

| | |
|--|-----|
| 1. _InclusionsLibrary | 9 |
| 1.1 _SDK Known Issues | 9 |
| 1.2 _Flagship Image for Plugin Development Platform | 9 |
| 1.3 _Package Name | 9 |
| 1.4 _Quality Standards | 10 |
| 1.5 _Note about Maven Bundled with SDK | 10 |
| 1.6 _Version Compatibility for Plugin Development Platform | 10 |
| 1.7 _Confluence Plugin Tutorials | 10 |
| 1.8 _Plugin SDK Supported Applications and Default Ports | 10 |
| 1.9 _Plugin Platform Version | 11 |
| 1.10 _About Plugin Framework 2 Documentation | 11 |
| 1.11 _JIRA Plugin Tutorials | 11 |
| 1.12 _Plugin Tutorial Step - Build and Test | 11 |
| 1.13 _Plugin Tutorial Step - Create Plugin | 11 |
| 1.14 _PACSearchBox | 11 |
| 1.15 _Getting Help on Plugin SDK | 12 |
| 1.16 _Components of Plugin Development Platform | 12 |
| 1.17 _Plugin SDK Version | 13 |
| 1.18 _Plugin SDK Download Latest | 13 |
| 1.19 _Development Hubs | 13 |
| 1.20 _NoteDocMigration | 13 |
| 2. .bookmarks | 14 |
| 2.1 Installing the Atlassian Plugin SDK - Atlassian Developer Network - Atlassian Documentation - Confluence | 14 |
| 3. Obsolete Documentation | 14 |
| 3.1 20051011 Developer Network Announcement | 14 |
| 3.2 Add unit and functional test capability | 16 |
| 3.3 Atlassian IDEA Plugin | 19 |
| 3.4 Building and Deploying Confluence in IDEA | 21 |
| 3.4.1 Compiling and Running Confluence | 21 |
| 3.4.2 Creating an IDEA project with a Confluence distribution | 22 |
| 3.4.3 Creating an IDEA project with a Confluence source release Maven1 | 23 |
| 3.5 Building and Deploying Plugins in IDEA | 26 |
| 3.5.1 Adding a plugin project to IDEA | 26 |
| 3.5.2 Setting up a new Confluence plugin | 28 |
| 3.5.3 Working on an existing Confluence plugin | 29 |
| 3.6 Community Authors | 29 |
| 3.7 Confluence Plugin Development Kit | 29 |
| 3.8 Convert pom and directory structure | 30 |
| 3.9 Creating a new plugin | 34 |
| 3.10 Developer Subversion Repository | 35 |
| 3.10.1 Subversion Repository Feed | 35 |
| 3.11 Developing with Eclipse | 37 |
| 3.12 Guide to Testing Plugins | 42 |
| 3.13 Initial development environment setup | 43 |
| 3.13.1 Atlassian Maven repositories OLD | 44 |
| 3.13.2 Example settings.xml OLD | 45 |
| 3.14 List of Product Dependencies | 48 |
| 3.14.1 Atlassian IDE Plugin 1.0 Dependencies | 63 |
| 3.14.2 Bamboo 1.0 Dependencies | 64 |
| 3.14.3 Bamboo 2.0 Dependencies | 66 |
| 3.14.4 Bamboo 2.1 Dependencies | 68 |
| 3.14.5 Bamboo 2.2 Dependencies | 71 |
| 3.14.6 Bamboo 2.3 Dependencies | 76 |
| 3.14.7 Clover 2.2 Dependencies | 81 |
| 3.14.8 Confluence 2.4 Dependencies | 82 |
| 3.14.9 Confluence 2.8 Dependencies | 84 |
| 3.14.10 Crowd 1.0.6 Dependencies | 87 |
| 3.14.11 Crowd 1.3.2 Dependencies | 88 |
| 3.14.12 FishEye Crucible 1.5 Dependencies | 90 |
| 3.14.13 JIRA 3.8 Dependencies | 93 |
| 3.14.14 JIRA 3.9 Dependencies | 96 |
| 3.14.15 JIRA 3.10 Dependencies | 98 |
| 3.14.16 JIRA 3.11 Dependencies | 101 |
| 3.14.17 JIRA 3.12 Dependencies | 104 |
| 3.14.18 JIRA 3.13 Dependencies | 106 |
| 3.14.19 JIRA 4.0 Dependencies | 109 |
| 3.15 Obsolete Confluence Documentation | 111 |
| 3.15.1 Confluence Plugin Repository | 111 |
| 3.15.1.1 Confluence Plugin Repository Details | 112 |
| 3.15.2 Plugins - Recursive JarClassLoader | 116 |
| 3.16 Obsolete JIRA Documentation | 121 |
| 3.16.1 Building and deploying JIRA from source | 122 |
| 3.16.2 Configure IDEA Global Settings | 124 |
| 3.16.3 Configure the Database | 124 |
| 3.16.4 Create Idea Project Files | 124 |
| 3.16.5 Debug JIRA | 124 |
| 3.16.6 Deploy JIRA | 125 |
| 3.16.7 Developing with Eclipse2 | 125 |
| 3.16.8 Disable Velocity Caching | 125 |
| 3.16.9 Edit build.properties | 125 |

| | |
|--|-----|
| 3.16.10 Edit More Configuration | 125 |
| 3.16.11 External Resources | 126 |
| 3.16.11.1 Maven Resources | 126 |
| 3.16.11.2 Workflow Function plugin | 126 |
| 3.16.12 Get the Source | 126 |
| 3.16.13 Getting Started | 126 |
| 3.16.14 Getting what you need -- Prerequisites | 127 |
| 3.16.15 How to create a JIRA Project Tab | 127 |
| 3.16.16 How to set up a development environment for a JIRA Plugin | 132 |
| 3.16.17 Install an Appserver | 133 |
| 3.16.18 Install Atlassian-IDEA Maven Plugin | 133 |
| 3.16.19 Setting up your plugin project | 133 |
| 3.17 Old Plugin Documentation | 134 |
| 3.17.1 Developing your plugin | 138 |
| 3.17.2 Testing your plugin | 140 |
| 3.18 Packaging and releasing a plugin | 142 |
| 3.19 Plugin Developer Kit for Maven1 | 142 |
| 3.20 Plugin Developer Kit for Maven2 | 145 |
| 3.20.1 Atlassian Plugin Archetype | 147 |
| 3.20.2 Confluence WARs | 148 |
| 3.20.3 How Integration Tests work in Plugins | 149 |
| 3.20.4 Legacy Plugin Archetype | 154 |
| 3.21 Plugin Development Kits | 155 |
| 3.22 Plugin Project Files | 155 |
| 3.22.1 build.properties | 155 |
| 3.22.2 project.properties | 157 |
| 3.22.3 project.xml | 157 |
| 3.23 Plugin Self-Configuration | 158 |
| 4. NavigationLinks | 159 |
| 5. TreeNavigation | 159 |
| 6. Welcome to the Atlassian Developer Network | 159 |
| 6.1 How to learn to program (in Java or JavaScript) | 161 |
| 6.2 Installing the Atlassian Plugin SDK | 162 |
| 6.2.1 Setting up your Plugin Development Environment | 163 |
| 6.2.1.1 Colour-Coding your Maven Output | 166 |
| 6.2.1.2 Configuring Eclipse to use the SDK | 167 |
| 6.2.1.3 Configuring IDEA to use the SDK | 168 |
| 6.2.1.4 Configuring NetBeans to use the SDK | 169 |
| 6.2.1.5 Creating a remote debug target | 170 |
| 6.2.1.6 Setting JAVA_HOME | 173 |
| 6.2.1.7 Using the SDK Maven as an Advanced Maven User | 174 |
| 6.2.2 Atlassian Maven Repositories | 174 |
| 6.2.3 Atlassian Plugin SDK FAQ | 175 |
| 6.2.3.1 Errors when Creating an Archetype | 175 |
| 6.2.3.2 How can I change the version of Java my plugin uses | 176 |
| 6.2.3.3 Maven Cannot Find Java Mail, Java Activation or JTA | 176 |
| 6.2.3.4 Maven is Unable to Download the Artifact from Any Repository | 177 |
| 6.2.3.5 Maven Parsing Error Unrecognised HTML Tag | 177 |
| 6.2.3.6 Maven Runs Out of Memory | 178 |
| 6.2.3.7 Maven Warning POM for X is Invalid | 178 |
| 6.2.3.8 SDK doesn't hear customizations for port or context path | 178 |
| 6.3 Writing your first plugin | 178 |
| 6.3.1 Developing your Plugin using the Atlassian Plugin SDK | 180 |
| 6.3.1.1 Quick Start for the Impatient | 181 |
| 6.3.1.2 Step 1. Create a Plugin Skeleton | 182 |
| 6.3.1.2.1 Synchronize your pom.xml and atlassian-plugin.xml | 183 |
| 6.3.1.3 Step 2. Start the Host Application with your Plugin Installed | 184 |
| 6.3.1.3.1 Finding your plugin in the host application | 185 |
| 6.3.1.4 Step 3. Generate Project Files for your IDE | 186 |
| 6.3.1.5 Step 4. Write Some Code | 186 |
| 6.3.1.6 Step 5. Run your Changes | 187 |
| 6.3.2 Packaging and Releasing your Plugin | 188 |
| 6.3.2.1 Plugin Release Checklist | 190 |
| 6.3.3 Writing Tests for your Plugin | 191 |
| 6.3.3.1 Product Data for Integration Testing | 195 |
| 6.3.4 Setting OSGi Manifest Instructions in your Plugin | 195 |
| 6.3.5 Plugin Hosting on JIRA Studio | 199 |
| 6.3.5.1 Building with Bamboo on Plugin Studio | 199 |
| 6.3.6 Plugin Developer Best Practices | 200 |
| 6.3.7 Plugin and Gadget Gotchas | 201 |
| 6.3.7.1 Cannot start plugin | 205 |
| 6.3.7.2 ClassNotFoundException | 205 |
| 6.3.7.3 Compilation failure - package does not exist | 205 |
| 6.3.7.4 java.lang.LinkageError - loader constraints violated | 206 |
| 6.3.7.5 NoClassDefFoundError | 206 |
| 6.3.7.6 PluginException | 207 |
| 6.3.7.7 SAXParserContextProvider | 207 |
| 6.3.7.8 UnsatisfiableDependenciesException | 207 |
| 6.3.7.9 UnsatisfiedDependencyException - Error creating bean with name | 207 |
| 6.3.8 Atlassian JavaDoc | 208 |

| | |
|--|-----|
| 6.3.9 Plugin Module Index | 208 |
| 6.3.10 Writing your first plugin FAQ | 227 |
| 6.3.10.1 Adding Selenium to Functional Tests | 227 |
| 6.3.10.2 Build Failure - Manifest Validation Errors | 228 |
| 6.3.10.3 Cannot Log In to Confluence using Admin Account | 229 |
| 6.3.10.4 Choosing a Logging Framework | 229 |
| 6.3.10.5 Choosing a Package Name | 229 |
| 6.3.10.6 Eclipse Maven Plugin Build Error | 230 |
| 6.3.10.7 If you change pom.xml, you may need to restart the atlas-cli | 230 |
| 6.3.10.8 Instant Loading of Plugin Resources | 230 |
| 6.3.10.9 Overriding the application's webapp when developing your plugin | 230 |
| 6.3.10.10 Specifying a particular version of the host application | 230 |
| 6.3.10.11 Using the Atlassian Plugin SDK with a Source Code License | 231 |
| 6.3.10.12 Using your own log4j configuration for your plugin | 231 |
| 6.4 Advanced Plugin Development | 232 |
| 6.4.1 Atlassian API Changes | 233 |
| 6.4.2 Atlassian Plugin Development Platform | 234 |
| 6.4.2.1 Plugin Development Platform Version Matrix | 237 |
| 6.4.2.2 Using the Plugin Development Platform Dependency Management POMs | 239 |
| 6.4.3 Early Access Programs | 240 |
| 6.4.4 Using the Atlassian RefApp | 241 |
| 6.4.5 Advanced Plugin Development FAQ | 242 |
| 6.4.5.1 Adding WebSudo Support to your Plugin | 243 |
| 6.4.5.2 BeanCreationException from Spring Framework | 243 |
| 6.4.5.3 Best Practices for ensuring a plugin works with a new version of the product | 243 |
| 6.4.5.4 Bundling extra dependencies in an OBR | 244 |
| 6.4.5.5 Converting a Plugin to Plugins 2 | 247 |
| 6.4.5.6 Deployment stalled due to spaces in directory path | 247 |
| 6.4.5.7 Detecting the Presence or Absence of Classes in Different OSGI Bundles | 247 |
| 6.4.5.8 Functional Testing with Multiple Products | 250 |
| 6.4.5.9 Getting Custom fields from Confluence V2 Searches | 252 |
| 6.4.5.10 Plugins that Cannot be Reloaded with pi | 252 |
| 6.4.5.11 Using Plugins for Testing and Development | 254 |
| 6.5 Atlassian Plugin SDK Documentation | 255 |
| 6.5.1 Plugin SDK Commands | 259 |
| 6.5.1.1 atlas-clean | 260 |
| 6.5.1.2 atlas-cli | 261 |
| 6.5.1.3 atlas-clover | 262 |
| 6.5.1.4 atlas-compile | 263 |
| 6.5.1.5 atlas-create-bamboo-plugin | 264 |
| 6.5.1.6 atlas-create-confluence-plugin | 265 |
| 6.5.1.7 atlas-create-crowd-plugin | 266 |
| 6.5.1.8 atlas-create-fecru-plugin | 268 |
| 6.5.1.9 atlas-create-home-zip | 269 |
| 6.5.1.10 atlas-create-jira-plugin | 269 |
| 6.5.1.11 atlas-create-refapp-plugin | 271 |
| 6.5.1.12 atlas-debug | 272 |
| 6.5.1.13 atlas-help | 274 |
| 6.5.1.14 atlas-install-idea-plugin (not published) | 275 |
| 6.5.1.15 atlas-install-plugin | 276 |
| 6.5.1.16 atlas-integration-test | 277 |
| 6.5.1.17 atlas-mvn | 278 |
| 6.5.1.18 atlas-package | 279 |
| 6.5.1.19 atlas-release | 280 |
| 6.5.1.20 atlas-release-rollback | 281 |
| 6.5.1.21 atlas-run | 282 |
| 6.5.1.22 atlas-run-standalone | 284 |
| 6.5.1.23 atlas-unit-test | 286 |
| 6.5.1.24 atlas-version | 286 |
| 6.5.2 Atlassian Plugin SDK Release Notes | 287 |
| 6.5.2.1 Atlassian Plugin SDK 3.0 Beta 2 Release Notes | 288 |
| 6.5.2.2 Atlassian Plugin SDK 3.0 Beta 1 Release Notes | 288 |
| 6.5.2.3 Atlassian Plugin SDK 3.0.5 Release Notes | 288 |
| 6.5.2.4 Atlassian Plugin SDK 3.0.4 Release Notes | 289 |
| 6.5.2.5 Atlassian Plugin SDK 3.0 Beta 5 Release Notes | 289 |
| 6.5.2.6 Atlassian Plugin SDK 3.0.3 Release Notes | 289 |
| 6.5.2.7 Atlassian Plugin SDK 3.0.2 Release Notes | 290 |
| 6.5.2.8 Atlassian Plugin SDK 3.0.1 Release Notes | 290 |
| 6.5.2.9 Atlassian Plugin SDK 3.0 Release Candidate 1 Release Notes | 290 |
| 6.5.2.10 Atlassian Plugin SDK 3.0 Release Notes | 291 |
| 6.5.2.11 Atlassian Plugin SDK 3.0 Beta 8 Release Notes | 291 |
| 6.5.2.12 Atlassian Plugin SDK 3.0 Beta 7 Release Notes | 291 |
| 6.5.2.13 Atlassian Plugin SDK 3.0 Beta 4 Release Notes | 291 |
| 6.5.2.14 Atlassian Plugin SDK 3.0 Beta 3 Release Notes | 292 |
| 6.5.2.15 Atlassian Plugin SDK 3.0 Beta 6 Release Notes | 292 |
| 6.5.2.16 Atlassian Plugin SDK 3.0.6 Release Notes | 292 |
| 6.5.2.17 Atlassian Plugin SDK 3.1 Release Notes | 293 |
| 6.5.2.18 Atlassian Plugin SDK 3.1.1 Release Notes | 293 |
| 6.5.2.19 Atlassian Plugin SDK 3.1.2 Release Notes | 294 |
| 6.5.2.20 Atlassian Plugin SDK 3.1.3 Release Notes | 295 |

| | |
|---|-----|
| 6.5.2.21 Atlassian Plugin SDK 3.2 Release Notes | 295 |
| 6.5.2.22 Atlassian Plugin SDK 3.2.3 Release Notes | 296 |
| 6.5.2.23 Atlassian Plugin SDK 3.2.4 Release Notes | 297 |
| 6.5.2.24 Atlassian Plugin SDK 3.3 Release Notes | 298 |
| 6.5.2.25 Atlassian Plugin SDK 3.3.1 Release Notes | 299 |
| 6.5.2.26 Atlassian Plugin SDK 3.4 Release Notes | 300 |
| 6.5.2.27 Atlassian Plugin SDK 3.3.2 Release Notes | 301 |
| 6.5.2.28 Atlassian Plugin SDK 3.3.3 Release Notes | 302 |
| 6.5.2.29 Atlassian Plugin SDK 3.3.4 Release Notes | 304 |
| 6.5.3 AMPS Plugin for Maven | 305 |
| 6.5.4 Getting Help with the Atlassian Plugin SDK | 312 |
| 6.5.5 Converting your Plugin to the new Atlassian Plugin SDK | 313 |
| 6.5.6 Plugin SDK Snapshots | 317 |
| 6.5.7 SDK Feedback Form | 318 |
| 6.5.8 Validation with XML schemas | 318 |
| 6.6 Other Information | 320 |
| 6.6.1 Documentation | 320 |
| 6.6.2 How to develop without the Atlassian Plugin SDK | 321 |
| 6.6.2.1 Example pom.xml | 321 |
| 6.6.2.2 Example settings.xml | 323 |
| 6.6.2.3 Obsolete Atlassian Maven PDK | 323 |
| 6.6.2.3.1 Speeding Up Confluence Plugin Development Using Maven CLI | 325 |
| 6.6.2.4 Example atlassian-plugin.xml | 326 |
| 6.6.2.5 Maven Requirements | 329 |
| 6.6.2.6 Atlassian Plugin Archetypes | 330 |
| 6.6.3 Developer Bamboo Server | 333 |
| 6.6.4 Getting Started with Remote APIs | 333 |
| 6.6.5 Confluence Build Information | 334 |
| 6.7 FAQ | 337 |
| 6.8 Tutorials | 338 |
| 6.8.1 Plugin Gadget Tutorial - Using the FishEye REST API to Write a Gadget to Monitor Recent Changes | 340 |
| 6.8.2 Plugin Gadget Tutorial - Writing a Gadget that Displays the Days Left in a Version | 345 |
| 6.8.3 Plugin Tutorial - Adding a custom action to Confluence | 372 |
| 6.8.4 Plugin Tutorial - Adding a JQL Function to JIRA | 380 |
| 6.8.5 Plugin Tutorial - Adding your own Menu Items to Confluence | 384 |
| 6.8.6 Plugin Tutorial - Adding your own Menu Items to JIRA | 386 |
| 6.8.7 Plugin Tutorial - Creating a JIRA Report | 391 |
| 6.8.8 Plugin Tutorial - Defining a Pluggable Service in a Confluence Plugin | 401 |
| 6.8.9 Plugin Tutorial - Scheduling Events via SAL | 404 |
| 6.8.10 Plugin Tutorial - Writing event listeners with the atlassian-event library | 414 |
| 6.8.11 Plugin Tutorial - Writing a Confluence Macro that Uses JSON | 420 |
| 6.8.12 Plugin Tutorial - Writing a Confluence Theme | 428 |
| 6.8.13 Plugin tutorial - Writing an Admin Configuration Screen | 434 |
| 6.8.14 Plugin Tutorial - Writing Gadgets for JIRA | 448 |
| 6.8.15 Plugin Tutorial - Writing Integration Tests for your JIRA plugin | 453 |
| 6.8.16 Plugin Tutorial - Writing Macros for Confluence | 457 |
| 6.8.17 Plugin Tutorial - Writing REST Services | 460 |
| 6.8.18 Plugin Tutorial - Writing Unit Tests for your Plugin | 463 |
| 6.8.19 Standalone Gadget Tutorial - Writing a JQL Gadget | 471 |
| 6.8.20 Tutorial Requests | 478 |
| 6.8.21 Tutorial Templates | 479 |
| 6.8.21.1 Template for Plugin Tutorials | 479 |
| 6.9 Getting Involved in the Atlassian Developer Network | 481 |
| 6.9.1 DevNet Discussion | 482 |
| 6.9.1.1 IRC Chat Transcripts | 482 |
| 6.9.1.1.1 atlasiandev_log-2010-01-16 | 483 |
| 6.9.1.1.2 atlasiandev_log-2010-01-17 | 483 |
| 6.9.1.1.3 atlasiandev_log-2010-01-18 | 484 |
| 6.9.1.1.4 atlasiandev_log-2010-01-19 | 485 |
| 6.9.1.1.5 atlasiandev_log-2010-01-20 | 488 |
| 6.9.1.1.6 atlasiandev_log-2010-01-21 | 490 |
| 6.9.1.1.7 atlasiandev_log-2010-01-22 | 492 |
| 6.9.1.1.8 atlasiandev_log-2010-01-23 | 494 |
| 6.9.1.1.9 atlasiandev_log-2010-01-24 | 494 |
| 6.9.1.1.10 atlasiandev_log-2010-01-25 | 494 |
| 6.9.1.1.11 atlasiandev_log-2010-01-26 | 495 |
| 6.9.1.1.12 atlasiandev_log-2010-01-27 | 496 |
| 6.9.1.1.13 atlasiandev_log-2010-01-28 | 497 |
| 6.9.1.1.14 atlasiandev_log-2010-01-29 | 500 |
| 6.9.1.1.15 atlasiandev_log-2010-01-30 | 505 |
| 6.9.1.1.16 atlasiandev_log-2010-01-31 | 507 |
| 6.9.1.1.17 atlasiandev_log-2010-02-01 | 508 |
| 6.9.1.1.18 atlasiandev_log-2010-02-02 | 510 |
| 6.9.1.1.19 atlasiandev_log-2010-02-03 | 512 |
| 6.9.1.1.20 atlasiandev_log-2010-02-04 | 514 |
| 6.9.1.1.21 atlasiandev_log-2010-02-05 | 515 |
| 6.9.1.1.22 atlasiandev_log-2010-02-06 | 515 |
| 6.9.1.1.23 atlasiandev_log-2010-02-07 | 516 |
| 6.9.1.1.24 atlasiandev_log-2010-02-08 | 517 |
| 6.9.1.1.25 atlasiandev_log-2010-02-09 | 519 |

| | |
|--|-----|
| 6.9.1.1.26 atlassiandev_log-2010-02-10 | 520 |
| 6.9.1.1.27 atlassiandev_log-2010-02-11 | 522 |
| 6.9.1.1.28 atlassiandev_log-2010-02-12 | 523 |
| 6.9.1.1.29 atlassiandev_log-2010-02-13 | 525 |
| 6.9.1.1.30 atlassiandev_log-2010-02-14 | 526 |
| 6.9.1.1.31 atlassiandev_log-2010-02-15 | 526 |
| 6.9.1.1.32 atlassiandev_log-2010-02-16 | 529 |
| 6.9.1.1.33 atlassiandev_log-2010-02-17 | 530 |
| 6.9.1.1.34 atlassiandev_log-2010-02-18 | 533 |
| 6.9.1.1.35 atlassiandev_log-2010-02-19 | 535 |
| 6.9.1.1.36 atlassiandev_log-2010-02-20 | 535 |
| 6.9.1.1.37 atlassiandev_log-2010-02-21 | 535 |
| 6.9.1.1.38 atlassiandev_log-2010-02-22 | 537 |
| 6.9.1.1.39 atlassiandev_log-2010-02-23 | 537 |
| 6.9.1.1.40 atlassiandev_log-2010-02-24 | 539 |
| 6.9.1.1.41 atlassiandev_log-2010-02-25 | 540 |
| 6.9.1.1.42 atlassiandev_log-2010-02-26 | 542 |
| 6.9.1.1.43 atlassiandev_log-2010-02-27 | 542 |
| 6.9.1.1.44 atlassiandev_log-2010-02-28 | 543 |
| 6.9.1.1.45 atlassiandev_log-2010-03-01 | 543 |
| 6.9.1.1.46 atlassiandev_log-2010-03-02 | 545 |
| 6.9.1.1.47 atlassiandev_log-2010-03-03 | 545 |
| 6.9.1.1.48 atlassiandev_log-2010-03-04 | 545 |
| 6.9.1.1.49 atlassiandev_log-2010-03-05 | 547 |
| 6.9.1.1.50 atlassiandev_log-2010-03-06 | 548 |
| 6.9.1.1.51 atlassiandev_log-2010-03-07 | 548 |
| 6.9.1.1.52 atlassiandev_log-2010-03-08 | 548 |
| 6.9.1.1.53 atlassiandev_log-2010-03-09 | 549 |
| 6.9.1.1.54 atlassiandev_log-2010-03-10 | 550 |
| 6.9.1.1.55 atlassiandev_log-2010-03-11 | 551 |
| 6.9.1.1.56 atlassiandev_log-2010-03-12 | 552 |
| 6.9.1.1.57 atlassiandev_log-2010-03-13 | 552 |
| 6.9.1.1.58 atlassiandev_log-2010-03-14 | 552 |
| 6.9.1.1.59 atlassiandev_log-2010-03-15 | 552 |
| 6.9.1.1.60 atlassiandev_log-2010-03-16 | 553 |
| 6.9.1.1.61 atlassiandev_log-2010-03-17 | 554 |
| 6.9.1.1.62 atlassiandev_log-2010-03-18 | 554 |
| 6.9.1.1.63 atlassiandev_log-2010-03-19 | 556 |
| 6.9.1.1.64 atlassiandev_log-2010-03-20 | 557 |
| 6.9.1.1.65 atlassiandev_log-2010-03-21 | 557 |
| 6.9.1.1.66 atlassiandev_log-2010-03-22 | 558 |
| 6.9.1.1.67 atlassiandev_log-2010-03-23 | 559 |
| 6.9.1.1.68 atlassiandev_log-2010-03-24 | 561 |
| 6.9.1.1.69 atlassiandev_log-2010-03-25 | 562 |
| 6.9.1.1.70 atlassiandev_log-2010-03-26 | 563 |
| 6.9.1.1.71 atlassiandev_log-2010-03-27 | 564 |
| 6.9.1.1.72 atlassiandev_log-2010-03-28 | 564 |
| 6.9.1.1.73 atlassiandev_log-2010-03-29 | 566 |
| 6.9.1.1.74 atlassiandev_log-2010-03-30 | 566 |
| 6.9.1.1.75 atlassiandev_log-2010-03-31 | 567 |
| 6.9.1.1.76 atlassiandev_log-2010-04-01 | 567 |
| 6.9.1.1.77 atlassiandev_log-2010-04-02 | 568 |
| 6.9.1.1.78 atlassiandev_log-2010-04-03 | 569 |
| 6.9.1.1.79 atlassiandev_log-2010-04-04 | 569 |
| 6.9.1.1.80 atlassiandev_log-2010-04-05 | 569 |
| 6.9.1.1.81 atlassiandev_log-2010-04-06 | 569 |
| 6.9.1.1.82 atlassiandev_log-2010-04-07 | 570 |
| 6.9.1.1.83 atlassiandev_log-2010-04-08 | 570 |
| 6.9.1.1.84 atlassiandev_log-2010-04-09 | 571 |
| 6.9.1.1.85 atlassiandev_log-2010-04-10 | 571 |
| 6.9.1.1.86 atlassiandev_log-2010-04-11 | 571 |
| 6.9.1.1.87 atlassiandev_log-2010-04-12 | 571 |
| 6.9.1.1.88 atlassiandev_log-2010-04-13 | 572 |
| 6.9.1.1.89 atlassiandev_log-2010-04-14 | 573 |
| 6.9.1.2 Plugin Testing Resources and Discussion | 573 |
| 6.9.2 Mailing Lists and Support | 573 |
| 6.9.3 Contributing to the Atlassian Developer Documentation | 574 |
| 6.10 Codegeist V | 574 |
| 6.10.1 Speakeasy Admin Guide | 575 |
| 6.10.2 Speakeeasy Extension Development Guide | 575 |
| 6.10.3 Speakeeasy Extension Examples | 580 |
| 6.10.4 Speakeeasy Install Guide | 580 |
| 6.10.5 Speakeeasy Overview | 581 |
| 6.10.6 Speakeeasy Plugin Development | 582 |
| 6.11 Developer Relations State of the Union 2011 - Resources | 583 |
| 7. Plugin Libraries | 584 |
| 8. Plugin (Glossary Entry) | 584 |
| 9. Admin-Config Wishes | 584 |
| 9.1 Development Kit for Eclipse+Ant | 584 |
| 9.1.1 EclipseAntDevelopmentKit | 585 |

| | |
|---|-----|
| 9.2 NTLM Authentication | 586 |
| 9.3 Plugin Repository Submission Form | 586 |
| 10. Avail - JIRA | 586 |
| 10.1 Agile Charts and Graphs | 586 |
| 10.2 JIRA MS Project Integration | 587 |
| 11. Exists | 587 |
| 11.1 Completed Specs | 587 |
| 11.1.1 Mail This Page | 588 |
| 11.1.2 JIRA Labels | 589 |
| 11.1.3 IntelliJ Idea Plugin for Jira and Confluence | 590 |
| 11.1.4 Dynamic Announcement Banner | 591 |
| 11.1.5 Invite user to space plugin | 591 |
| 11.1.6 Popups or Tooltips | 592 |
| 11.1.7 SVN Project Tab | 592 |
| 11.1.8 Translation Plugin | 593 |
| 11.1.8.1 JN Comments on First Delivery | 597 |
| 11.1.9 BloggerRPC Plugin | 598 |
| 11.1.10 Search Keyword Macro | 599 |
| 11.1.11 SVN Commit Acceptance | 599 |
| 11.1.11.1 Phase 1 design sketches | 600 |
| 11.1.12 Vote-for Custom Field | 601 |
| 11.1.13 Mark For Review | 601 |
| 11.1.14 Math Equations Plugin | 602 |
| 11.1.15 Google Maps Plugin | 602 |
| 11.1.16 New Code Macro | 602 |
| 11.1.17 Confluence Contributors Plugin | 602 |
| 11.1.18 Comments Tab Theme | 603 |
| 11.1.19 JIRA Calendar Plugin | 603 |
| 11.1.20 Social Bookmarking | 603 |
| 11.1.20.1 UI Feedback | 605 |
| 11.1.21 NAnt builder | 606 |
| 11.1.22 Zip Search Extractor | 606 |
| 11.1.23 Content Searcher for OpenOffice docs | 607 |
| 11.1.24 Import Visio Diagrams | 607 |
| 11.1.25 WebDav Plugin | 607 |
| 11.1.26 Crowd Ruby Integration Libraries | 608 |
| 11.1.27 Crowd PHP Integration Libraries | 608 |
| 11.1.28 Usage Stats Plugin | 609 |
| 12. Trusted Application Authentication (Glossary Entry) | 609 |
| 13. Approval Macro | 610 |
| 14. CVS Report Macro | 612 |
| 15. Gallery RSS Feed | 612 |
| 16. HTML Diff | 612 |
| 17. Moved to JIRA | 613 |
| 17.1 Confluence plugin for JIRA | 613 |
| 17.2 Link Visualization | 613 |
| 17.3 Sectioned Editing Macro | 613 |
| 17.4 Set field value using XPath post-function | 614 |
| 18. Attach a file custom field plugin | 614 |
| 18.1 To Sort | 614 |
| 19. For Bamboo | 614 |
| 19.1 Bamboo - Mercurial SCM integration | 614 |
| 19.2 Builds triggered by commit | 615 |
| 20. Add Excel cell range to Viewfile Macro | 615 |
| 21. Avail - Confluence | 616 |
| 21.1 Add contents to all pages in space | 616 |
| 21.2 admin log in as | 616 |
| 21.3 Better Flowchart Macro | 617 |
| 21.4 Collapsible Lists | 617 |
| 21.5 Personal Homepage | 617 |
| 21.6 Recently added pages | 619 |
| 21.7 Show confluence Permissions | 619 |
| 21.8 Space Provisioning Plugin | 620 |
| 21.9 Suspended Specs | 620 |
| 21.9.1 Advanced Table Macro | 620 |
| 21.9.2 Tasklist Improvements | 621 |
| 21.9.2.1 Tasks | 623 |
| 21.9.2.2 Task Scheduler | 624 |
| 21.10 User Information | 625 |
| 21.11 UWC | 625 |
| 21.11.1 Chariot Solutions converter | 626 |
| 21.11.2 Converter from Dan Woodhams | 628 |
| 21.11.3 UWC Converter Packs | 628 |
| 21.12 Voting and Vote tabulation | 628 |
| 21.13 Wiki Markup Editor AutoCompletion | 628 |
| 21.14 Wiki Markup Editor Toolbar | 629 |
| 21.15 Word, Open Office and Excel Importing | 629 |
| 22. Confluence Gantt Charts | 629 |
| 23. Plugin to expose Confluence page ordering to the SOAP API | 630 |
| 24. Document Staging | 630 |

| | |
|---|-----|
| 25. Who can see this page? | 631 |
| 26. In Process | 631 |
| 27. Set Field Value Using Regex Plugin | 631 |
| 28. Theory of Constraints | 632 |
| 29. Better Recent Changes Macro | 632 |
| 30. JIRA Issues Maps | 633 |
| 31. Offline Confluence Editor | 633 |
| 32. Calendar Improvements | 634 |
| 33. Datatable | 634 |
| 34. HTML WysiWyg Editor like Google Docs | 635 |
| 35. Live log viewer | 635 |
| 36. New Themes | 635 |
| 36.1 Bamboo Theme | 636 |
| 36.2 More Themes | 636 |
| 36.3 Simple Theme | 636 |
| 36.3.1 Simple Theme Mockup 1 | 638 |
| 36.4 Simple Theme Mockup 2 | 639 |
| 36.5 Standards Compliant Theme | 640 |
| 37. Things You Wish Were Pluggable | 640 |
| 37.1 Plugin System Improvements | 641 |
| 37.2 User object customization | 644 |
| 38. Improve the MS Office Add-in | 644 |
| 39. JIRA-Mylar plugin | 645 |
| 40. JIRA-VSS Integration | 646 |
| 41. Outline Plugin | 646 |
| 42. Display Access Macro | 647 |
| 43. Cobertura build complete action | 648 |
| 44. SCP Articfact Copier | 648 |
| 45. Emma post build action | 648 |
| 46. Checkout Macro | 649 |
| 47. Crowd Shibboleth Authenticator | 649 |
| 48. Crowd .NET Integration Libraries | 650 |
| 49. Crowd Delegated-Crowd-Connector | 650 |
| 50. Google Chart Support | 651 |
| 51. Who is online | 651 |
| 52. Create Dependent Issue plugin | 652 |
| 53. Google Maps Custom Fields for JIRA | 652 |
| 54. Mail this bug | 652 |
| 55. Post-resolution voting | 652 |
| 56. Find Related Issues | 652 |
| 57. JIRA Macro Library | 653 |
| 58. Space Search and Replace | 653 |
| 59. AssignUP | 653 |
| 60. Chat Plugin | 654 |
| 61. Gantt Chart Plugin | 655 |
| 62. Numbered Sections Macro | 656 |
| 63. SERENA Dimension Integration | 656 |
| 64. Combined Filters JIRA Plugin | 657 |
| 65. Communication Plugin | 657 |
| 66. Confluence Bamboo Plugin | 658 |
| 67. Email multiple users, groups | 661 |
| 68. Video Chat Plugin for Confluence | 661 |
| 69. Embed Gantt Project | 662 |
| 70. File download and transformation plugin | 662 |
| 71. Repository Tagger build complete action | 662 |
| 72. Friendly User Groups | 663 |
| 73. Keyword Index | 663 |
| 74. RSS Mix macro | 665 |
| 75. User History | 666 |
| 76. Amazon API Macros | 666 |
| 77. Dictionary in Editor | 666 |
| 78. Drag and Drop Attachments | 667 |
| 79. Recently Commented | 667 |
| 80. Search and Replace and Date and Time | 667 |
| 81. Default Charts Tab Panel | 668 |
| 82. Improved index macro | 668 |
| 83. Google Maps Custom Field for JIRA | 668 |
| 84. Checkbox Tree Custom Field | 668 |
| 85. Reporter IP CustomField | 668 |
| 86. cli | 669 |
| 87. compress-resources | 669 |
| 88. copy-bundled-dependencies | 669 |
| 89. create | 669 |
| 90. create-home-zip | 669 |
| 91. debug | 669 |
| 92. filter-plugin-descriptor | 669 |
| 93. generate-manifest | 669 |
| 94. generate-obr-artifact | 669 |
| 95. idea | 669 |
| 96. install | 669 |

| | |
|------------------------------|-----|
| 97. integration-test | 669 |
| 98. jar | 670 |
| 99. run | 670 |
| 100. unit-test | 670 |
| 101. validate-manifest | 670 |

_InclusionsLibrary

The children of this page contain information which is **included in other pages**, both in the DEVNET space and **in other spaces too**. This is a library of re-usable information chunks.

If you want to change any of these pages, be aware that:

- Changing page names is problematic — you will need to change all the `include` and `excerpt-include` macros manually.
 - The content is used in many places — make sure your change is generic enough to fit the contexts in which the pages are used.
- [_SDK Known Issues](#)
 - [_Flagship Image for Plugin Development Platform](#)
 - [_Package Name](#)
 - [_Quality Standards](#)
 - [_Note about Maven Bundled with SDK](#)
 - [_Version Compatibility for Plugin Development Platform](#)
 - [_Confluence Plugin Tutorials](#)
 - [_Plugin SDK Supported Applications and Default Ports](#)
 - [_Plugin Platform Version](#)
 - [_About Plugin Framework 2 Documentation](#)
 - [_JIRA Plugin Tutorials](#)
 - [_Plugin Tutorial Step - Build and Test](#)
 - [_Plugin Tutorial Step - Create Plugin](#)
 - [_PACSearchBox](#)
 - [_Getting Help on Plugin SDK](#)
 - [_Components of Plugin Development Platform](#)
 - [_Plugin SDK Version](#)
 - [_Plugin SDK Download Latest](#)
 - [_Development Hubs](#)
 - [_NoteDocMigration](#)

_SDK Known Issues

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., <http://localhost:1990>) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

_Flagship Image for Plugin Development Platform



Out of date

Note that this diagram is out of date. I'm leaving it here because legacy documentation spaces refer to it. We should not use it in any current documentation.

| Name | Size | Creator | Creation Date | Comment |
|----------------------------------|-------|---|--------------------|---------|
| PNG File PluginPlatform-50pc.png | 31 kB | Sarah Maddox [Atlassian Technical Writer] | Oct 30, 2008 00:05 | |

_Package Name

Make sure your package names are unique. You can choose any package name, provided that it does not conflict with any existing package used in the Atlassian application you are developing for, or in any other plugins. Do *not* use `com.atlassian.confluence.plugins.*` — That is confusing to people who try to use the plugin. Use a real package name that corresponds to your organisation or project. For

example: `org.mycompany.confluence.plugins.*` The Sun Java documentation has some tips about conventions used to ensure unique package names.

Quality Standards

- be in the Atlassian Plugin Exchange.
- have a JIRA project on <http://studio.plugins.atlassian.com/>.
- have source-code available at <http://studio.plugins.atlassian.com/source/>
- have Bamboo builds set up on one of our build servers.
- have unit and functional tests with a certain amount of code-coverage.
- have complete, accurate and attractive documentation in <http://studio.plugins.atlassian.com/wiki/>
- display a compatibility matrix for the last three product releases.
- support all configurations of the product (e.g. clustered Confluence), except where stated otherwise in the plugin documentation
- be internationalised.

Note about Maven Bundled with SDK

When running Maven commands against your project, make sure that you use the version of Maven bundled with the Atlassian Plugin SDK. This is important if you have a local version of Maven installed, as well as the Atlassian Plugin SDK. The simplest way is to use the `atlas-mvn` wrapper command instead of `mvn`. Another way is to [put the bundled Maven on your path](#).

Version Compatibility for Plugin Development Platform

The matrix below shows the versions of each component in the Atlassian Plugin Development Platform, mapped to the versions of the platform itself. The components are listed horizontally across the top and the platform versions are listed vertically on the left.

Version numbers in brackets show a future release.

| Plugin Development Platform | AUI | Plugin Framework | REST Plugin | SAL | UPM |
|-----------------------------|---------|------------------------|------------------------|---------|-----------|
| Platform 2.8 | AUI 3.2 | Plugin Framework 2.6 | REST Plugin Module 2.2 | SAL 2.2 | |
| Platform 2.9 | AUI 3.2 | (Plugin Framework 2.7) | REST Plugin Module 2.2 | SAL 2.2 | (UPM 1.2) |

Confluence Plugin Tutorials

- Writing a Confluence Theme
- Adding a REST Service to Confluence
- Writing Macros for Confluence
- Writing a Confluence Macro 1
- Integration Testing Confluence Plugins – from our friends at Customware
- Adding a custom action to Confluence
- Adding your own Menu Items to Confluence
- Defining a Pluggable Service in a Confluence Plugin
- Writing a Confluence macro that uses JSON
- Upgrading and Migrating an Existing Confluence Macro to Confluence 4.0
- Creating a new Confluence 4.0 Macro
- Creating A Template Bundle
- Extending the V2 search API
- Searching using the V2 Search API
- Writing a search result renderer

Plugin SDK Supported Applications and Default Ports

The table below shows the applications currently supported by the Atlassian Plugin SDK, the minimum version of the application required, and the default port for each application.

| Atlassian Application | Version | Default Port | Product Key | Caveats |
|-----------------------|---------|--------------|-------------|---------|
| Bamboo | 2.3 + | 6990 | bamboo | |
| Confluence | 2.10 + | 1990 | confluence | |
| Crowd | 2.0.2 + | 4990 | crowd | |

| | | | | |
|----------|---------|------|--------|---|
| Crucible | 2.0.6 + | 3990 | fecru | |
| FishEye | 2.0.6 + | 3990 | fecru | |
| JIRA | 4.0 + | 2990 | jira | Plugins developed for versions of JIRA before 4.0 are supported, but using the SDK with versions of JIRA earlier than 4.0 is not. For developing plugins for JIRA 3.13 and earlier, take a look at the old JIRA PDK . |
| RefApp | 1.0 + | 5990 | refapp | |

Plugin Platform Version



Current released version: Atlassian Plugin Development Platform 2.10
Atlassian Plugin Development Platform 2.10 is now available – see the [release notes](#).

About Plugin Framework 2 Documentation

This documentation gives an overview of the development platform, libraries and utilities you will use when creating plugins in the **Atlassian Plugin Framework 2.x**. The Plugin Framework itself is documented in [Atlassian Plugin Framework 2](#). The Atlassian Plugin Framework 2 replaces the original Atlassian Plugin framework, described in the [Developer Network](#).

JIRA Plugin Tutorials

- Writing listeners with the atlassian-event library
- Adding a REST Service to JIRA
- Writing a Plugin Gadget for JIRA
- Adding a JQL Function to JIRA
- Adding your own Menu Items to JIRA
- Creating a JIRA Report
- Writing Integration Tests for your JIRA plugin
- WebWork Sample Plugin – from Matt Doar, describing how JIRA uses WebWork
- Writing a Gadget that Displays the Days Left in a Version

- Available Permissions
- How to create a new Custom Field Type
- How to create Custom Workflow Elements for JIRA 3

Plugin Tutorial Step - Build and Test

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Plugin Tutorial Step - Create Plugin

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

PACSearchBox



Do Not Delete This Page!

| Name | Size | Creator | Creation Date | Comment |
|-------------------------------------|--------|----------------|--------------------|---------|
| HTML File PAC_Search_Macro.html | 5 kB | Jonathan Nolen | Jan 25, 2011 14:28 | |
| GIF File threecol_module_bottom.gif | 1 kB | Jonathan Nolen | Jan 25, 2011 14:11 | |
| GIF File threecol_module_top.gif | 13 kB | Jonathan Nolen | Jan 25, 2011 14:11 | |
| GIF File threecol_module_bg.gif | 0.1 kB | Jonathan Nolen | Jan 25, 2011 14:11 | |
| PNG File search_icon.png | 0.5 kB | Jonathan Nolen | Jan 25, 2011 14:11 | |
| PNG File page_bg.png | 0.2 kB | Jonathan Nolen | Jan 25, 2011 14:11 | |
| GIF File button_search.gif | 3 kB | Jonathan Nolen | Jan 25, 2011 14:11 | |
| PNG File icon_default.png | 2 kB | Jonathan Nolen | Jan 21, 2011 19:01 | |

Getting Help on Plugin SDK



Help is at hand

Make sure you are not in the Maven command line interface (CLI) when you enter the help commands described below. If you are in the CLI, your command line will start with `maven2>`, and you will need to exit from the CLI first. Enter `quit`, `exit` or a friendly `bye`.

Getting an Overview of All the Scripts

Enter the following shell script to see an overview of all the scripts with a brief outline of their functionality:

```
atlas-help
```

Enter the following to see all possible help content:

```
atlas-help --verbose
```

Getting Help Text per Script

Each shell script provides help text if the first argument of the script is one of the following:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

Examples:

```
atlas-clean help  
atlas-cli -?
```

Reading the Reference Guide

See the detailed [guide to all scripts](#).

Components of Plugin Development Platform

These are the major components in the Atlassian Plugin Development Platform:

- **Plugin Framework** – The framework that executes the plugins and manages the available plugin modules.
- **Shared Access Layer (SAL)** – The API for accessing common services, regardless of the underlying Atlassian application interfaces.

- **Atlassian User Interface (AUI)** – A set of reusable, cross-browser tested JavaScript and CSS UI components.
- **Atlassian REST Plugin Module** – An Atlassian plugin module that you can use to create plugin points easily in Atlassian applications, by exposing services and data entities as REST APIs.
- **Atlassian Template Renderer** – A library that provides an abstraction on top of various templating libraries, making it easier to use the templating libraries. For example, instead of your having to create a VelocityEngine object and configure it, we provide a factory to do that for you. See the [Javadoc](#) and the [wiki documentation](#).
- **Universal Plugin Manager (UPM)** – The UI for installing, managing, upgrading, and sharing Atlassian plugins.
- **Application Links (AppLinks)** – An Atlassian plugin module that allows you to link your [JIRA](#), [Confluence](#), [FishEye](#), [Crucible](#) and [Bamboo](#) applications.

Plugin SDK Version



Current released version – Atlassian Plugin SDK 3.4

Currently-supported applications: **Confluence**, **JIRA**, **Bamboo**, **FishEye**, **Crucible** and **Crowd**.

Atlassian Plugin SDK 3.4 is now available – see the [release notes](#). If you have ideas for improvement or new features, or have found a bug, please raise a ticket on our issue tracker. [Snapshot builds](#) are also available for the stout-hearted.

Plugin SDK Download Latest



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

[Download Latest Version \(Windows\)](#)



[Download Latest Version \(Mac OS X, UNIX, Linux\)](#)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

Development Hubs

- [Developer Network](#)
- [Plugin Framework](#)
- [Gadgets](#)
- [REST APIs](#)
- [Confluence](#)
- [JIRA](#)
- [GreenHopper](#)
- [Bamboo](#)
- [Crowd](#)
- [FishEye/Crucible](#)
- [JIRA Mobile Connect](#)

NoteDocMigration

 The developer documentation is currently **read only**. Atlassian is working on the launch of a new developer portal. We are currently migrating all developer documentation to a new Confluence site. This will take a bit of time. We will keep it as short as possible and let you know when the migration is complete. Happy reading. 😊

.bookmarks



Atlassian Developer Network Bookmarks

This page is a container for all the bookmarks in this space. Do not delete or move it or you will lose all your bookmarks.
[Bookmarks in Atlassian Developer Network](#) | [Links for Atlassian Developer Network](#)



The 15 most recent bookmarks in Atlassian Developer Network

[Installing the Atlassian Plugin SDK - Atlassian Developer Network - Atlassian Documentation - Confluence](#)

Posted 229 days ago in Atlassian Developer Network by Frederick Tompkins | [View Bookmark Page](#)

Labels: (None)

[0 Comments](#)

Installing the Atlassian Plugin SDK - Atlassian Developer Network - Atlassian Documentation - Confluence



[Installing the Atlassian Plugin SDK - Atlassian Developer Network - Atlassian Documentation - Confluence](#)

Posted 229 days ago in Atlassian Developer Network by Frederick Tompkins

Labels: (None)

[View: Bookmarks in Atlassian Developer Network](#) | [Links for Atlassian Developer Network](#)

Obsolete Documentation

- [20051011 Developer Network Announcement](#)
- [Add unit and functional test capability](#)
- [Atlassian IDEA Plugin](#)
- [Building and Deploying Confluence in IDEA](#)
- [Building and Deploying Plugins in IDEA](#)
- [Community Authors](#)
- [Confluence Plugin Development Kit](#)
- [Convert pom and directory structure](#)
- [Creating a new plugin](#)
- [Developer Subversion Repository](#)
- [Developing with Eclipse](#)
- [Guide to Testing Plugins](#)
- [Initial development environment setup](#)
- [List of Product Dependencies](#)
- [Obsolete Confluence Documentation](#)
- [Obsolete JIRA Documentation](#)
- [Old Plugin Documentation](#)
- [Packaging and releasing a plugin](#)
- [Plugin Developer Kit for Maven1](#)
- [Plugin Developer Kit for Maven2](#)
- [Plugin Development Kits](#)
- [Plugin Project Files](#)
- [Plugin Self-Configuration](#)

20051011 Developer Network Announcement

Announcing the Atlassian Developer Network

We're pleased to announce the launch of the Atlassian Developer Network.

At Atlassian, we've put a lot of energy into making sure that there are easy and powerful ways to extend JIRA and Confluence. We know that our users are building incredibly cool stuff on top of our products. There is already a community of JIRA & Confluence developers out there who have been very generous with their time, expertise and code.

We want to connect those developers with each other, with the developers at Atlassian and with the broader user community. We want to make sure that you have the information and support you need to create brilliant solutions. And we want to share those solutions as widely as possible, for the benefit of everyone.

There are four principles that will guide us as we grow the Developer Network:

1. **Communicate openly** - We believe in open and honest communication. We strive to make our products and our processes as transparent as possible so that you can make better decisions.
2. **Encourage participation** - We believe in the power of the community. And we believe that collaboration on an equal footing will create the greatest value for everyone.
3. **Listen to the community** - We believe in listening closely to our developers. Your feedback and participation will guide how the Developer Network evolves.
4. **Iterate rapidly** - We believe in delivering value to our customers quickly and frequently. We plan to expand and improve the Developer Network on a frequent and ongoing basis.

Listed below are some of the resources that we've lined up first. As always, we'd love to hear (developer-support@atlassian.com) what else you'd like to see.

1. The Developer Network Homepage

We'll start with the most important link. Here's our new homepage:

<http://confluence.atlassian.com/display/DEVNET>

The DevNet Confluence space is homebase for our brand new developer network and it has links to all of the other resources we're outlining in this email.

2. The Developer Blog

Consider this your behind-the-scenes tour of our little software factory. We plan to have as many of our staff posting here as possible. We have a lot of talented developers 'behind the curtain' and we want to let you get to know the people who actually build the software. You'll get to see how we work and what we're working on – before it hits the download server.

<http://blogs.atlassian.com/developer>

3. New Developer Mailing Lists / Forums

We've set up two brand new mailing lists especially for people developing with JIRA or Confluence. This is how you can tap into the formidable resources of the Atlassian developer community.

(The mailinglist and forum cross-post to each other. So use the format you prefer – you won't miss out on anything.)

JIRA Dev Forum: <http://forums.atlassian.com/forum.jspa?forumID=100>
Confluence Dev Forum: <http://forums.atlassian.com/forum.jspa?forumID=101>

JIRA Dev MailingList: <http://atlassian.com/software/jira/mailinglist.jsp#developers>
Confluence Dev MailingList: <http://atlassian.com/software/confluence/mailinglist.jsp#developers>

4. Improved (and Improving) Developer Docs

We've also been working on improving the developer documentation. We've created some new "Getting Started"-style tutorials and added a brand new Confluence plugin development kit. Enhancing the developer docs is an on-going effort, but I think we're off to a good start.

The JIRA Development Hub: http://confluence.atlassian.com/x/_iw
The Confluence Development Hub: <http://confluence.atlassian.com/x/ZQ0C>

5. The Plugin Libraries

One of our major goals for the developer network is to share the great work you are doing with the rest of the world. We're big believers in the power of the community, and by contributing your plugins back to the community, everyone can benefit and participate in the evolution.

So we're going to be highlighting some of the plugins over the next few months, in order to let the rest of our users know about all the great things they can do thanks to the work that you have contributed.

6. Plugin Hosting

And to that end, we're extremely excited to offer project hosting for JIRA and Confluence plugins. If you have a plugin that you're willing to make available to the community Atlassian will provide you with a Subversion repository and a JIRA project to make collaborative

development as easy as possible.

If you're interested, just send an email to developer-support@atlassian.com

7. Developer Relations

Oh yeah, then there's me. I'm Jonathan Nolen (jonathan@atlassian.com), and I've just begun as Atlassian's new Director of Developer Relations. It's my job to make sure we're listening to the developer community and that we're delivering the tools you need. You have been doing some seriously impressive coding, and I'm here to make sure that you have the support of Atlassian and the developer community behind you.

If you're interested in knowing a little more about me and where I came from, you can find my blog at: <http://www.jnolen.com>.

This is the first of what will hopefully be many announcements about the expanding Atlassian Developer Network. We've got lots of exciting additions in the pipeline. But even more important is what you decide to do and how you decided to contribute. This is your invitation to jump in. Get involved. Tell us what you'd like to see. Ask someone for help, or help someone out. Share what you're working on. We can't wait to see what develops.

Cheers,
Jonathan & the Atlassian team

Add unit and functional test capability

Who broke what? Add unit and functional test to find out

Here are the steps needed to setup your plugin to run tests against an instance of confluence.

1. [Add configurations to pom.xml](#)
2. [Add configurations to settings.xml](#)
3. [Add and configure test-specific files](#)

Add configurations to pom.xml

add these configuration tags into your pom.xml

```

<project>
...
<dependencies>
...
    <!-- libraries needed for setting up a Confluence instance -->
    <dependency>
        <groupId>com.atlassian.confluence</groupId>
        <artifactId>confluence-plugins-setup</artifactId>
        <version>${confluence.version}</version>
        <scope>provided</scope>
    </dependency>

    <!-- WAR file used for cargo to launch a Confluence instance
    <dependency>
        <groupId>com.atlassian.confluence</groupId>
        <artifactId>confluence-webapp</artifactId>
        <version>rpc-${confluence.version}</version>
        <type>war</type>
        <scope>provided</scope>
    </dependency>
...
</dependencies>
...
<build>
...
    <plugins>
...
        <plugin>
            <groupId>org.codehaus.cargo</groupId>
            <artifactId>cargo-maven2-plugin</artifactId>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
        </plugin>
        <plugin>
            <groupId>com.atlassian.maven.plugins</groupId>
            <artifactId>maven-properties-initialiser-plugin</artifactId>
        </plugin>
...
    </plugins>
...
</build>
...
<properties>
...
    <!-- version of Confluence to test against -->
    <confluence.version>2.3-SNAPSHOT</confluence.version>
    <acceptanceTestSettings>test.properties</acceptanceTestSettings>
...
</properties>
...
</project>

```

Add configurations to settings.xml

In setting up your Confluence instance, you need to provide a license key.

To provide this for your instance, add the following to your **settings.xml** file located in **{user-home-directory}/.m2** (create this file if it does not exist):

```

<settings>
  <profiles>
    ...
      <profile>
        ...
          <activation>
            <activeByDefault>true</activeByDefault>
          </activation>
        ...
        <properties>
          ...
            <CONF_COMM>
              <!-- insert your license key here -->
              BlaBlabLablaBlaBlaBlaBlaBlaBlaBlaBlaBlaBlaBlaBla
              BlaBlabLablaBlaBlaBlaBlaBlaBlaBlaBlaBlaBlaBlaBlaBla
              BlaBlabLablaBlaBlaBlaBlaBlaBlaBlaBlaBlaBlaBlaBlaBla
              Bla
            </CONF_COMM>
          ...
        </properties>
        ...
      </profile>
    ...
  </profiles>
</settings>

```

Add and configure test-specific files

Here are the files that are needed:

test.properties - save this file in src/test/resources

```

# PDK Properties
confluence.dev.server.home=C:/home/mike/data/bucket-refactoring
confluence.dev.server.url=http://localhost:8080
confluence.dev.server.user=admin
confluence.dev.server.pass=admin
confluence.dev.server.pluginKey=confluence.extra.chart

# Cargo Properties
cargo.container.containerid=tomcat5x
cargo.container.zipurlinstaller.url=http://apache.mirror.pacific.net.au/tomcat/tomcat-5/v5.5.17/bin/apache-tomcat-5.5.17.zip
Atlassian-idea variables (only needed if you want to use atlassian-idea)
appserver.module.dependency=${pom.artifactId}
maven.war.src=${warInstallPath}

```



Customizing Ports

For more information on customizing ports for functional testing, see [here](#)

SetupInvokerAcceptanceTest.java - save this file in src/test/java/it/com/atlassian/confluence/

```

package it.com.atlassian.confluence;

import com.atlassian.confluence.SetupAcceptanceTest;

public class SetupInvokerAcceptanceTest extends SetupAcceptanceTest {
}

```

RunUploadInvokerAcceptanceTest.java - save this file in src/test/java/it/com/atlassian/confluence/

```

package it.com.atlassian.confluence;

import com.atlassian.confluence.RunUploadAcceptanceTest;

public class RunUploadInvokerAcceptanceTest extends RunUploadAcceptanceTest
{
}

```

now give 'mvn clean test' and 'mvn clean integration-test -Dmaven.test.unit.skip=true' a go

Currently, there are no set rules for the acceptance test folder yet. But a suggestion that we use it to make sure your files follow the naming structure of *AcceptanceTest.java. This is so Cargo can run the correct Tests. It may also be handy to have a /src/test/java/it folder so that your integration/acceptance tests are in the "it" folder and your unit tests are in the /src/test/java folder.



Suggestions

To make life easier, you can use the [Atlassian IDEA Plugin](#) to develop and debug tests against a Confluence instance in IntelliJ IDEA

Atlassian IDEA Plugin

Introduction

The Maven 2 Atlassian IDEA plugin helps you set up any configuration in IDEA that is specific to Atlassian. The purpose of this plugin is to set up the includes and excludes of the source and test folders and to setup the App Servers as modules.

Please note that this plugin extends the IDEA plugin and runs it automatically. So `mvn idea:idea` does not need to be run.

Installation

1. Check out the plugin from <http://svn.atlassian.com/svn/public/atlassian/maven-idea-plugin/>
2. Run `mvn install -Dmaven.test.skip=true`

App Servers

There are 4 App Servers that can be set up with this plugin.

- Resin 2
- Resin 3
- ~~Tomcat 4~~ Not supported any more
- Tomcat 5
- Orion

Each of these can be setup in one step by running `mvn atlassian-idea:all -Pall` or individually by running the plugin followed by the appropriate server as the goal. Below is the different goals available:

| App Server | Command |
|------------|---|
| Resin 2 | <code>mvn atlassian-idea:resin -Pidea</code> |
| Resin 3 | <code>mvn atlassian-idea:resin3 -Pidea</code> |
| Tomcat 5 | <code>mvn atlassian-idea:tomcat -Pidea</code> |
| Orion | <code>mvn atlassian-idea:orion -Pidea</code> |

A server will not be setup if the path to that server can not be found. The path is specified in the `settings.xml` under your `~/.m2/settings.xml`. Each server must have its path stored as a variable. Below is an example of the variable names a paths in `settings.xml`:

```

<atlassian.idea.resin.location>C:\resin-2.1.17</atlassian.idea.resin.location>
<atlassian.idea.resin3.location>C:\resin-3.0.19\resin-3.0.19</atlassian.idea.resin3.location>
<atlassian.idea.tomcat.location>C:\apache-tomcat-5.5.17</atlassian.idea.tomcat.location>
<atlassian.idea.orion.location>C:\orion2.0.7\orion</atlassian.idea.orion.location>

```

Variables

Required Variables

There are two variables which have to be specified in the pom.xml file for the Atlassian IDEA plugin to work.

- <maven.war.src> - this is the path to the webapp directory eg. confluence/src/webapp
- <appserver.module.dependency> - this is the module that the appserver depends on in IDEA eg. confluence

Optional Variables

Functional Tests

```
<atlassian.idea.includeAcceptanceTests>true</atlassian.idea.includeAcceptanceTests>
```

Adds Functional Tests to your test path as defaults so that you don't have to add them in each time after running Atlassian-IDEA.

JDK Versions for Project and Modules

```
<atlassian.idea.jdk.name>1.4</atlassian.idea.jdk.name>
```

Sets the JDK Version for the project.

```
<atlassian.idea.tomcat_env.jdk.name>1.5</atlassian.idea.tomcat_env.jdk.name>
```

Sets the JDK Version for a specific Server. To set any other server just substitute *tomcat_env* for the module name.

Datasource

To customise the database that you use in IDEA with your server, make sure you set the following variables.

```
<atlassian.idea.application.db.jndiname></atlassian.idea.application.db.jndiname>
<atlassian.idea.application.db.drivername></atlassian.idea.application.db.drivername>
<atlassian.idea.application.db.url></atlassian.idea.application.db.url>
<atlassian.idea.application.db.username></atlassian.idea.application.db.username>
<atlassian.idea.application.db.password></atlassian.idea.application.db.password>
```

Global Libraries to Enable

Example

```
<atlassian.idea.global.libraries.to.enable>hsqldb</atlassian.idea.global.libraries.to.enable>
```

This variable is a comma (,) separated list of global libraries to enable.

Sources Feature

There is a feature which allows you Atlassian-IDEA plugin to attach sources to your class libraries. To do this you need to have a sources.xml file. You can read up on this on the [sources.xml Doc](#)

Troubleshooting

- In Confluence, you must be in the top-level directory of your CVS checkout, which is the directory _above_ where you used to run the maven 1 IDEA plugin
- The first time you run the plugin, it will die with a message like this. Ignore it and run it again, and everything should work:

```
[ATLMVNDOC:INFO] Internal error in the plugin manager executing goal
'com.atlassian.maven.plugins:maven-atlassian-idea-plugin:1.0.3-SNAPSHOT:all': Unable
to find the mojo
'com.atlassian.maven.plugins:maven-atlassian-idea-plugin:1.0.3-SNAPSHOT:all' in the
plugin 'com.atlassian.maven.plugins:maven-atlassian-idea-plugin'
```

Settings.xml

If the command `mvn atlassian-idea:all -Pall` doesn't work then you're probably missing these lines in your `settings.xml` file. Make sure you have these lines under `~/.m2/settings.xml`

```
<settings>
  <pluginGroups>
    ...
    <pluginGroup>com.atlassian.maven.plugins</pluginGroup>
    ...
  </pluginGroups>
  ...
  <profiles>
    <profile>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      ...
      <pluginRepositories>
        <pluginRepository>
          <id>atlassian-m2-repository</id>
          <name>Atlassian Maven 2.x Repository</name>
          <url>http://repository.atlassian.com/maven2</url>
        <snapshots>
          <enabled>true</enabled>
          <updatePolicy>always</updatePolicy>
        </snapshots>
        </pluginRepository>
      </pluginRepositories>
      <properties>
        ...
        <atlassian.idea.resin.location>C:\resin-2.1.17</atlassian.idea.resin.location>
        <atlassian.idea.resin3.location>C:\resin-3.0.19\resin-3.0.19</atlassian.idea.resin3.location>
        <atlassian.idea.tomcat.location>C:\apache-tomcat-5.5.17</atlassian.idea.tomcat.location>
        <atlassian.idea.orion.location>C:\orion2.0.7\orion</atlassian.idea.orion.location>
        ...
      </properties>
      ...
    </profile>
  </profiles>
</settings>
```

Building and Deploying Confluence in IDEA



Please ensure you have completed the [Initial development environment setup](#) before following this guide

There are two main ways of deploying Confluence inside IDEA. The main reason you would want to is so that you can debug [your plugins](#).

Confluence Source

This is the preferred of the two, since it will allow you to step through Confluence code as well as your own plugin code when debugging. However, it is only available if you have a [source code license](#), which generally only comes with a full Commercial license.

Confluence Distribution

The other way is to get a regular Confluence distribution running inside IDEA. This works just fine, but you won't be able to step into code outside your own plugin.

I have the source code

I just have a regular distribution

Compiling and Running Confluence

Compiling and running Confluence

Now that it's set up, we should make sure it's all working by getting it to run inside IDEA.

1. Click the 'confluence' module and then select 'Build > Make Project' from the menu. Alternately, click the  button.
2. From the drop-down beside the  button, select your app server (eg. 'Tomcat').
3. Click the  button next to the app server drop-down.



The most common problem here is that the JDK has not been set up correctly. If you have a problem, check the following:

1. Right-click on the 'confluence' module in the project window and select 'Module Settings' at the bottom of the popup-window.
2. Select the 'Libraries' tab
3. Check that a valid JDK has been selected for the module.
4. Check the other modules in the project for the same.

Creating an IDEA project with a Confluence distribution



Please ensure you have completed the [Initial development environment setup](#) properly before following this guide



This page is useful for academic and open-source customers who do not have access to the Confluence source distribution . If you are a commercial customer, you should download the Confluence source and follow [these instructions](#).

Getting the EAR/WAR distribution

1. Download the latest **EAR/WAR** distribution release of Confluence (you'll need to click on "Show advanced downloads" on the downloads page to see it) and expand it to a location on your hard disk (we'll call it '\$CONFBASE')
2. Download [confluence-dist.zip](#) file, which contains stripped-down project files for Maven, expand it and copy its contents into root of your Confluence release (\$CONFBASE).
3. Open the contained [build.properties](#) file in a text editor and adjust it to meet your environment settings. In particular, pay attention to the following:
 - **JDK Version** - This should be set to whatever your target JDK is called **in IDEA**. On Mac OS X, this is usually '1.4' or '1.5' (without quotes). Eg:

```
atlassian.idea.jdk.name=1.4
```

- **Application Server Settings** - Uncomment these and point to your app server installation. Only uncomment the ones you actually have installed. Eg:

```
atlassian.idea.tomcat.location=~/apache-tomcat-5.5.12
#atlassian.idea.resin3.location=$PATH_TO_MY_INSTALLATION
#atlassian.idea.resin.location=$PATH_TO_MY_INSTALLATION
#atlassian.idea.orion.location=$PATH_TO_MY_INSTALLATION
```

- **Application Settings** - The settings in which your application server will launch confluence. The settings below will allow you to access the running server at <http://localhost:8080/confluence>:

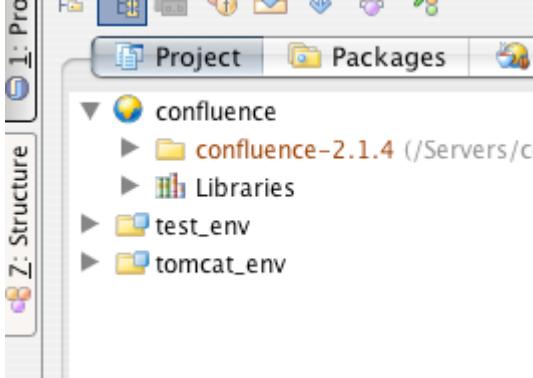
```
# port on which Confluence will listen
atlassian.idea.application.port=8080
# the context root at which Confluence will be deployed
atlassian.idea.application.webapp.contextpath=confluence
```

Creating the IDEA project files

1. Open a command prompt and enter the following:

```
cd $CONFBASE
maven atlassian-idea
```

2. That's it! Now just open up the freshly-generated 'confluence.ipr' file in IDEA. It should look something like this:



Setting up the Confluence environment

Once the project is loaded, you will need to set up where the 'confluence home' is.

1. Open `confluence/WEB-INF/classes/confluence-init.properties`
2. Set the 'confluence.home' property to the location you want confluence data stored. An example might be `$CONFHOME/confluence/home`, or somewhere in your home directory. If the location doesn't exist, Confluence will try to create it when you run the webapp.

Compiling and running Confluence

Now that it's set up, we should make sure it's all working by getting it to run inside IDEA.

1. Click the 'confluence' module and then select 'Build > Make Project' from the menu. Alternately, click the button.
2. From the drop-down beside the button, select your app server (eg. 'Tomcat').
3. Click the button next to the app server drop-down.



The most common problem here is that the JDK has not been set up correctly. If you have a problem, check the following:

1. Right-click on the 'confluence' module in the project window and select 'Module Settings' at the bottom of the popup-window.
2. Select the 'Libraries' tab
3. Check that a valid JDK has been selected for the module.
4. Check the other modules in the project for the same.

What's Next?

Now that we've got Confluence running, we can start working on plugins.

[I want to start one from scratch](#)

[I want to work on an existing plugin](#)

Creating an IDEA project with a Confluence source release Maven1



Please ensure you have completed the steps required to develop with IntelliJ IDEA before attempting the following.

Getting the source code

Firstly, you will probably only have access to the source code if you have a [commercial license](#). If you don't, but feel you should, contact [Atlassian](#) and see what you can work out.

Assuming the above, do the following:

1. Download the desired version of Confluence.

2. Expand the archive to your desired location (we'll call it \$CONFSRC)
3. Open the 'confluence' subdirectory under it.
4. Open the contained `build.properties` file in a text editor and adjust it to meet your environment settings. In particular, pay attention to the following:
 - *JDK Version* - This should be set to whatever your target JDK is called **in IDEA**. On Mac OS X, this is usually '1.4' or '1.5' (without quotes). Eg:

```
atlassian.idea.jdk.name=1.4
```

- *Source code location* - The location of extra source code. For some reason this is set incorrectly in the default source release. Set it to this:

```
atlassian.idea.src.relative.location=..
```

- *Application Server Settings* - Uncomment these and point to your app server installation. Only uncomment the ones you actually have installed. Eg:

```
atlassian.idea.tomcat.location=~/apache-tomcat-5.5.12
#atlassian.idea.resin3.location=$PATH_TO_MY_INSTALLATION
#atlassian.idea.resin.location=$PATH_TO_MY_INSTALLATION
#atlassian.idea.orion.location=$PATH_TO_MY_INSTALLATION
```

- *Application Settings* - The settings in which your application server will launch confluence. The settings below will allow you to access the running server at <http://localhost:8080/confluence>:

```
# port on which Confluence will listen
atlassian.idea.application.port=8080
# the context root at which Confluence will be deployed
atlassian.idea.application.webapp.contextpath=confluence
```

- *WAR Settings* - These should already be set for you in later versions, but it's good to check them. Eg:

```
maven.multiproject.type=war
maven.war.webxml=src/webapp/WEB-INF/web.xml
maven.war.webapp.dir=target/exploded
```

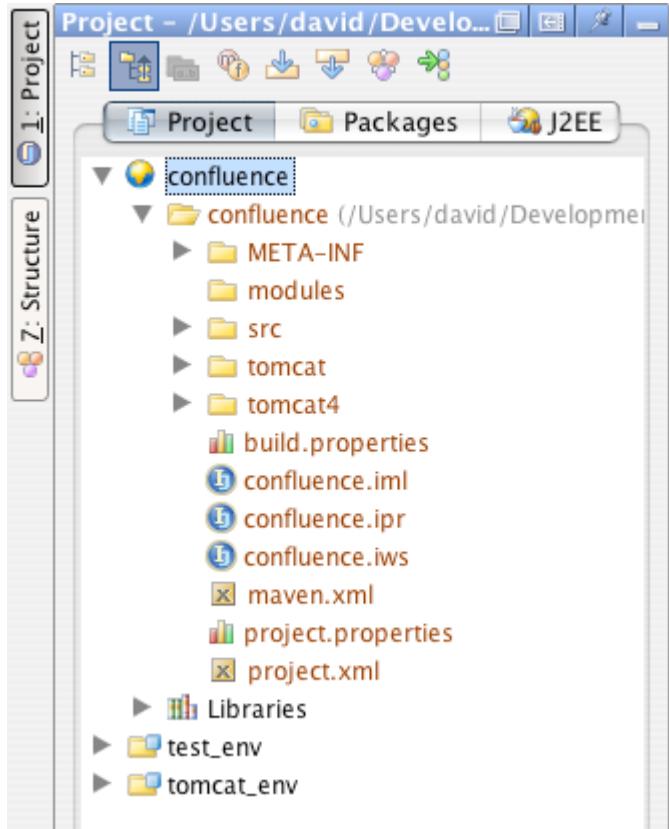
- *Database Settings* - If you are using an external database, set this up as documentation dictates.

Creating the IDEA project files

1. Open a command prompt and enter the following:

```
cd $CONFSRC/confluence
maven atlassian-idea
```

2. That's it! Now just open up the freshly-generated 'confluence.ipr' file in IDEA. It should look something like this:



Setting up the Confluence environment

Once the project is loaded, you will need to set up where the 'confluence home' is.

1. Open `src/etc/java/confluence-init.properties`
2. Set the '`confluence.home`' property to the location you want confluence data stored. An example might be `$CONF_SRC/confluence/home`, or somewhere in your home directory. If the location doesn't exist, Confluence will try to create it when you run the webapp.

Compiling and running/debugging Confluence

Now that it's set up, we should make sure it's all working by getting it to run inside IDEA.

1. Click the 'confluence' module and then select 'Build > Make Project' from the menu. Alternately, click the button.

The most common problem here is that the JDK has not been set up correctly. If you have a problem, check the following:

- a. Right-click on the 'confluence' module in the project window and select 'Module Settings' at the bottom of the popup-window.
- b. Select the 'Libraries' tab
- c. Check that a valid JDK has been selected for the module.
- d. Check the other modules in the project for the same.

2. From the drop-down beside the button, select your app server (eg. 'Tomcat').
3. Click the button next to the app server drop-down.

Hopefully your project will now run and you will be able to access Confluence through your web browser.

To debug, basically switch the last step of the above to clicking the button instead of the button.

What's Next?

Now that we've got Confluence running, we can start working on plugins.

[I want to start one from scratch](#)

I want to work on an existing plugin

Building and Deploying Plugins in IDEA



This guide assumes that you have already set IDEA to build and deploy Confluence.

There are two basic scenarios when starting plugin development with IDEA (well, in general really).

1. Starting from scratch

You have been inspired/ordered to make Confluence to something totally new and interesting.

2. Enhancing/maintaining an existing plugin

You want to fix some bugs or add some cool new functionality to an existing project, but you've just downloaded the source, or are switching from a different IDE.

I want to start one from scratch

I want to work on an existing plugin

Adding a plugin project to IDEA



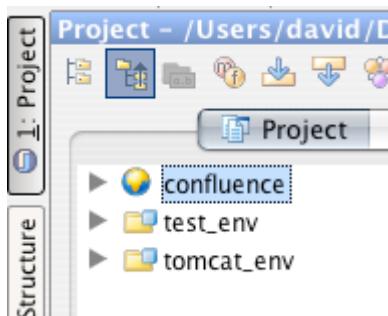
This guide assumes you have set up IDEA and have

- created a new plugin from scratch or
- downloaded and set up an existing plugin.

Now that we have our web application project and our plugin projects set up, we need to connect them together. The good news is, most of the hard work has been done now.

Adding a plugin

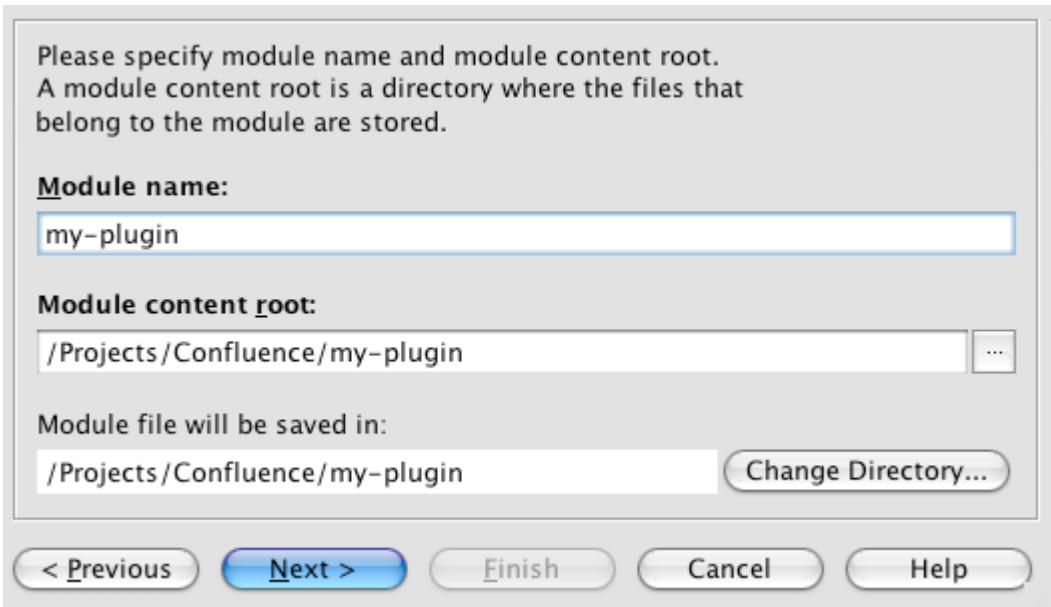
1. Open up your main Confluence project in IDEA. Your project should look something like this:



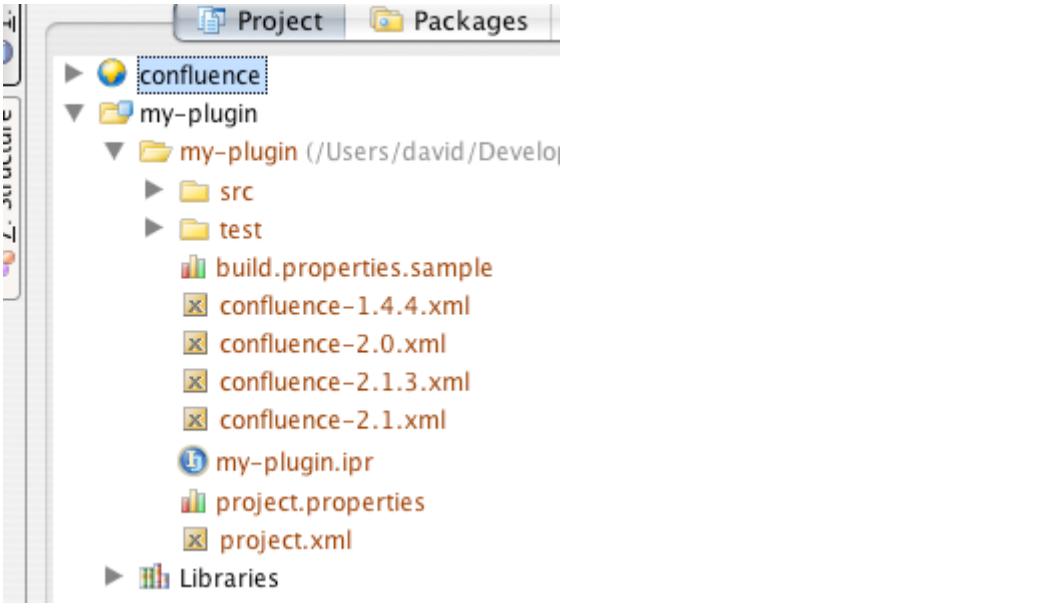
2. Now we add the plugin project you just created. First, select 'File > New Module'.

3. Select the 'Create new module' option and click next.

4. In 'Module Content Root', select to the root directory for the plugin.



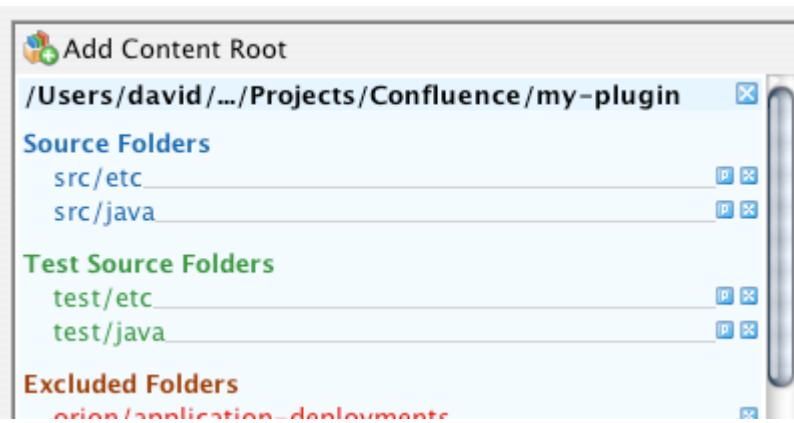
5. After continuing, the module should now have been added to your main project.



Checking the plugin settings

Next, we check that the plugin was actually generated correctly by atlassian-idea. Right-click on your new plugin module and select 'Module Settings'. We'll check through the tabs one-by-one.

Paths: In addition to the java/src/ path, you should add java/etc/. Your source paths look something like this:



| | |
|---|--|
| Libraries (Classpath): If your JDK is labeled 'Invalid', you need to select a JDK for your project (it should be the same one used by the main Confluence Module). | <p>Target JDK</p> <p><input type="radio"/> Use project JDK: <No JDK></p> <p><input checked="" type="radio"/> Use module-specific JDK: java version "1.5.0_05" [Invalid]</p> |
| Dependencies: Make sure that your plugin depends on the 'confluence' module. | <p>Modules:</p> <p><input type="button"/> Add <input type="button"/> Remove</p> <ul style="list-style-type: none"> confluence my-plugin test_env tomcat_env <p> Check modules this</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> confluence <input type="checkbox"/> test_env <input type="checkbox"/> tomcat_env |
| Order/Export: No changes here. | |
| JavaDoc: This one stays as-is. | |

Linking the plugin to Confluence

So far, the plugin should compile nicely, but it won't actually be hooked into Confluence if you run it in Tomcat (or your chosen application server). Let's set that up.

1. Open the Module Settings for the 'tomcat_env' project (Or your application server) and switch to **Dependencies**.
2. Select your new plugin as a dependency.

Modules:

Add Remove

- confluence
- my-plugin
- test_env
- tomcat_env

Check modules this

- confluence
- my-plugin
- test_env

Building and Testing your Plugin

Now, you just need to [develop your plugin](#). Once you have some code to test, do the following:

1. Start 'Build > Make Project'. This will put the necessary files into the webapp, ready to run, including your plugin source code.
2. Run the server ('Run > Run') or set some break-points and debug it ('Run > Debug').

What Next?

Now it's up to you to build something cool. Take a look at our [plugin guide](#) to see what the possibilities are.

Once you've completed your plugin, it's time to package it for distribution.

[I'm ready to package my plugin](#)

Setting up a new Confluence plugin



Please ensure you have set up IDEA with Confluence before following these instructions.

Creating the base plugin

Unable to render {include} Couldn't find a page to include called: CONFEXT:Creating a new plugin

Now what?

Next, we have to add your new project to the Confluence project you set up earlier.

I'm ready to add my plugin project

Working on an existing Confluence plugin

Get the plugin source code

Many plugin authors have been generous in sharing their plugin source code to allow other users to modify and contribute to their plugins. Some have chosen to use the [Atlassian Developer Network Subversion Repository](#) to maintain their source code.

You can generally find the source code location of a plugin by looking at table at the top of the plugin homepage ([like this one](#). Most will list a source-code location, either a download or a Subversion URL.

Subversion

If the plugin source code is stored in Subversion, you need to check out the source code using your Subversion client. If you're using the commandline client, you'd type:

```
svn co http://svn.atlassian.com/svn/public/contrib/confluence/sql-plugin/trunk/ sql-plugin
```

This should download all of the plugin's current source code into a directory called 'sql-plugin.'

Please note: If you want to check code back in, you will need to be an authorised user and use HTTPS instead of HTTP. But if you just want to check-out code, then you can just use HTTP.

Download

If the plugin author has attached the source code to the plugin homepage, just download the zip file and expand it.

Once you have the plugin source code, you're ready to move to the next step.

I'm ready to add my plugin project

Community Authors

Acting on the suggestion of one of our more prolific members, we've decided to create a new group for "Community Authors."

These folks have each contributed greatly in the past by writing plugins and answering questions in Confluence and on the mailing lists, and we're excited open up some of the semi-official documentation. Community Authors will, in addition to all of their normal rights, have permission to add and edit pages in the [JIRA](#) and the [Confluence](#) spaces.

I've seeded the group with all of the folk who already have Subversion access. However, if there are other users you would like to nominate – solid contributors to the mailing lists or the wiki – let me know and I can add them as well.

Right now, the following people are Community Authors:

- Dan Hardiker (from Adaptavist)
- David Peterson (author of plugins to numerous to list)
- Bob Swift (likewise)
- Brendan Patterson (Confluence developer extraordinaire)
- Danny Chen (original author of the scripting macros)
- Guy Fraser (from Adaptavist)
- Jon Pither (author of the JIRA Agile plugin)
- Simon Mittag (author of the JMAGE plugin)
- Binyan (author of the JIRA Clearcase Plugin)

Confluence Plugin Development Kit

| | |
|-----------------|-----------------------------------|
| Name | Confluence Plugin Development Kit |
| Plugin Version | 2.2.9 |
| Product Version | 2.2.9 |

| | |
|--------------------------|---|
| Author(s) | Atlassian |
| Home Page | Confluence Plugin Development Kit |
| Download Location | current version all versions |
| Download Source | http://svn.atlassian.com/svn/public/atlassian/confluence/confluence-development-kit/trunk/ |



The maven repository at ibiblio has moved to a new URL, so developers will have to update their maven configurations in order to build Confluence or its plugins. For more information, see [the Atlassian Developer Blog: Maven 1 Repository Changes](#).

Description/Features

A full development kit for Confluence plugins bundled with Confluence API and dependencies.

For more information, documentation and plugin tutorials, see our [Confluence Developer Documentation](#).

Included in this of the development kit are:

- Skeleton project templates for creating your own plugins
- Full API documentation of Confluence 2.2.9
- All Confluence 2.2.9 dependencies



For more details on plugins, see the [plugin guide](#).

Requirements

1. Sun Java JDK 1.4+ Confluence doesn't currently support Java 6.
2. Apache Maven 1.0 : <http://maven.apache.org/maven-1.x/start/download.html>
3. [Atlassian-IDEA Maven Plugin](#)
4. A running instance of Confluence: either [built from source](#) or a [standalone installation](#).

Maven is not strictly required to build your Confluence plugins, however, all of the example plugins have been developed with Maven as well as the template project. We highly recommend that you use Maven to build your plugins.



Idea is the Java Ide developed by JetBrains. This is totally optional for Confluence plugin development. You can use your own Ide or no Ide at all.

Instructions

1. Copy the templates directory and rename it for your plugin
2. Edit the project.xml for your plugin name
3. in your new directory, run 'maven atlassian-idea' to generate IDEA project files.
4. Launch your new project in IDEA by opening the \$MY_PLUGIN_NAME.ipr file.
5. Edit 'src/etc/atlassian-plugin.xml' for your project, as described in the [documentation](#).
6. Develop something cool!
7. When you're ready to deploy, run 'maven jar' to build your plugin jar in the target directory
8. [Deploy it](#) to your Confluence instance.

Version History

| Version | Comments |
|---------|--|
| 1.4 | initial version |
| 2.0 | upgraded for new version of Confluence |
| 2.1.1 | upgraded for new version of Confluence |
| 2.1.4 | upgraded for new version of Confluence |
| 2.2.9 | upgraded for new version of Confluence |

Convert pom and directory structure

I will work through how I converted the RSVP plugin over to Maven II as a step-by-step example
Here are the steps involved:

1. Convert project.xml to pom.xml
2. Convert Maven I Directory Structure

3. Add configurations to pom.xml

Original

project.xml

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<project>

<extend>confluence-2.2.2.xml</extend>
<pomVersion>3</pomVersion>

<id>rsvp</id>
<name>RSVP Plugin</name>
<currentVersion>1.1</currentVersion>

<organization>
  <name>Atlassian Software Systems Pty Ltd</name>
  <url>http://www.atlassian.com/</url>
</organization>

<test/etc|src/test/resources|
<package>com.atlassian.confluence.extra.rsvp</package>

<description>This plugin allows people to RSVP for an upcoming event</description>

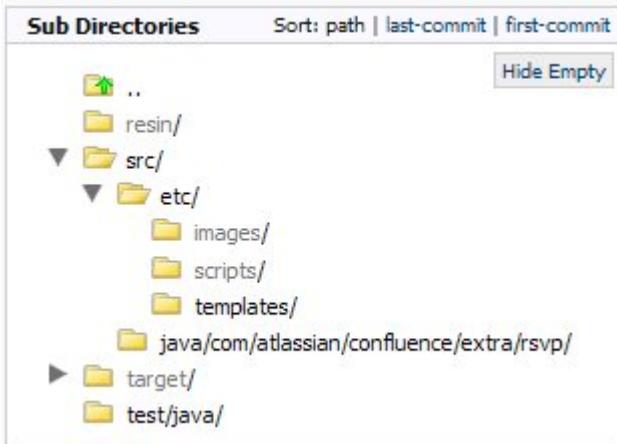
<build>
  <nagEmailAddress>jens@atlassian.com</nagEmailAddress>

  <sourceDirectory>src/java</sourceDirectory>
  <resources>
    <resource>
      <directory>src/etc</directory>
      <includes>**/*.*</includes>
    </resource>
    <resource>
      <directory>src/lib</directory>
      <includes>**/*.*</includes>
    </resource>
  </resources>

  <unitTestSourceDirectory>test/java</unitTestSourceDirectory>
  <unitTest>
    <includes>
      <include>**/Test*.java</include>
      <include>**/*TestCase.java</include>
    </includes>
    <excludes>
      <exclude>**/AbstractTestCase.java</exclude>
    </excludes>
    <resources>
      <resource>
        <directory>test/etc</directory>
        <includes>**/*.*</includes>
      </resource>
      <resource>
        <directory>src/etc</directory>
        <includes>**/*.*</includes>
      </resource>
    </resources>
  </unitTest>
</build>

</project>
```

directory structure



Convert project.xml to pom.xml

Here are the common XML tags that have an equivalence in Maven 2:

| Maven 1.x | Maven 2.x |
|----------------------------|---|
| <project> | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd"> |
| <extend> | deprecated |
| <pomVersion>3</pomVersion> | <modelVersion>4.0.0</modelVersion> |
| <id> | <artifactId> |
| <currentVersion> | <version> |
| <package> | <groupId> |
| <issueTrackingUrl> | <issueManagement>\<url></url> <issueManagement> |
| <repository> | <scm> |
| <nagEmailAddress> | <email> under <developer> tag |
| <extend> | use <dependency> of the project you want to use |
| <excludes> | <excludes>under <configurations> tag of surefire plugin |

So far...

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/maven-v4_0_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <artifactId>rsvp</artifactId>
  <name>RSVP Plugin</name>
  <version>1.1</version>

  <developers>
    <developer>
      <organization>Atlassian Software Systems Pty Ltd</organization>
      <organizationUrl>http://www.atlassian.com/</organizationUrl>
      <email>jens@atlassian.com</email>
      <name>Jens Schumacher</name>
    </developer>
  </developers>

  <groupId>com.atlassian.confluence.extra.rsvp</groupId>

  <description>This plugin allows people to RSVP for an upcoming event</description>

  <dependency>
    <groupId>com.atlassian.confluence</groupId>
    <artifactId>confluence</artifactId>
    <version>2.3-SNAPSHOT</version>
    <scope>provided</scope>
  </dependency>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <configuration>
          <excludes>
            <exclude>**/Abstract*TestCase.java</exclude>
          </excludes>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>

```



Scope Unneeded Dependencies

You may notice that the confluence dependency has scope provided. This is to ensure that the JAR compiled in the end does not include this dependency JAR.

Convert Maven I Directory Structure

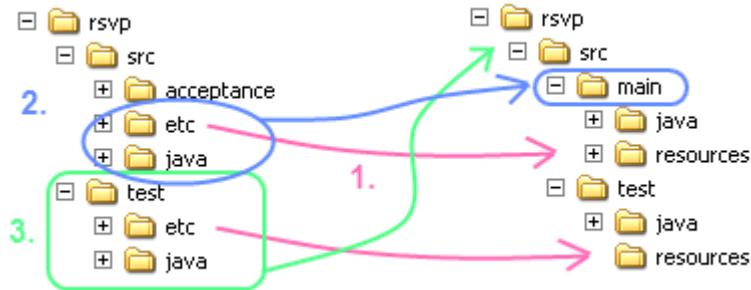
Here is the mapping between the old (Maven I) and the new (Maven II) directory structure:

| Maven 1.x | Maven 2.x Default |
|-----------|--------------------|
| src/java | src/main/java |
| src/etc | src/main/resources |
| test/etc | src/test/resources |
| test/java | src/test/java |

This will allow you to clean up the pom.xml as well in regards to <sourceDirectory> and other tags relating to directory structure.

So far...

directory structure



1. Rename the **etc** folders to **resources**
2. Create a **main** directory under /src. Move /src/java and /src/resources to /src/main.
3. Move /test/etc to /src/test.

Add configurations to pom.xml

add this plugin to the **pom.xml**. This will allow you to package the plugin with all its dependent JAR's

```
<project ...>
...
<parent>
<groupId>com.atlassian.confluence.plugin.base</groupId>
<artifactId>confluence-plugin-base</artifactId>
<version>1.3-SNAPSHOT</version>
</parent>
...
<packaging>atlassian-plugin</packaging>
...
<build>
...
<plugins>
...
<plugin>
<groupId>com.atlassian.maven.plugins</groupId>
<artifactId>atlassian-pdk</artifactId>
</plugin>
...
</plugins>
...
</build>
...
</project>
```

Now you have converted a Maven 1 directory structure to a Maven 2 structure.

Creating a new plugin

Method 1: Extract the sample project

1. Download [example-plugin-2.1.zip](#), and extract it somewhere where you can edit it.

Follow the instructions in the README.txt for generating an IDE project file and customising the plugin.

The contents of the sample project are like this:

```
example-plugin/pom.xml
example-plugin/README.txt
example-plugin/src/main/java/com/example/confluence/plugin/ExampleMacro.java
example-plugin/src/main/resources/atlassian-plugin.xml
example-plugin/src/test/java/com/example/confluence/plugin/TestExampleMacro.java
```

It includes a sample macro, ExampleMacro.java, a test case for that macro, TestExampleMacro.java, and a plugin descriptor, atlassian-plugin.xml.

Method 2: Use the Confluence plugin archetype

For more information on using the Confluence Archetype see the [Atlassian Plugin Archetypes](#) page.

IDE Integration

You can create project files using the normal maven plugins:

1. Run 'mvn eclipse:eclipse' to generate an Eclipse project file.
2. Run 'mvn idea:idea' to generate IntelliJ project files.

Related pages

For more information on writing Confluence plugins, see the [Confluence plugin guide](#).

Developer Subversion Repository

Atlassian can provide Subversion hosting for your JIRA or Confluence plugin.

Get a Subversion account

Mail developer-support@atlassian.com and request access. Please include the details about the plugin you would like us to host.

View or checkout from the Subversion repository.

Read-only access is available at <http://svn.atlassian.com/svn/public/> or you can browse and query the repository using [FishEye](#). However, if you have commit rights and intend on checking code *into* the repository, you must check out over HTTPS instead of HTTP.

Commit to the Subversion repository

There is an Atlassian-only directory at <https://svn.atlassian.com/svn/public/atlassian> and an open community directory at <https://svn.atlassian.com/svn/public/contrib> if you have an account (see above).

Every developer who has an account has full write access to the entire /contrib directory. So be careful about committing on a project other than yours.

If you import a project, be sure that you put it in the appropriate /JIRA or /confluence directory.

SVN Notify Mailing List

If you would like email notification of all changes to the repository, send an email with the "subscribe" in the body to devnet-svn-request@atlassian.com.

Recent Changes to the Atlassian Developer Subversion Repository

FishEye on public/contrib

(Recent activity in "public" under directory /contrib)



```
public (root): [maven-release-plugin] prepare for next development iteration
public (root): [maven-release-plugin] copy for tag atlassian-pdk-2.1.8
public (root): [maven-release-plugin] prepare release atlassian-pdk-2.1.8
public (root): Make the pdk use basic auth as well as os_username, when uploading with pi
public (root): Very basic swing UI added
public (root): [maven-release-plugin] prepare for next development iteration
public (root): [maven-scm] copy for tag cargo-test-runner-2.4
public (root): [maven-release-plugin] prepare release cargo-test-runner-2.4
public (root): cause I need authentication to svn ;P
public (root): fixing compilation error
public (root): fixing compilation error
public (root): cargo support for JUnit4 runners
public (root): [maven-release-plugin] prepare for next development iteration
public (root): [maven-scm] copy for tag cargo-test-runner-2.3
public (root): [maven-release-plugin] prepare release cargo-test-runner-2.3
```

Subversion Repository Feed

Recent Changes in the Atlassian Developer Subversion Repository

FishEye on public/contrib



(Recent activity in "public" under directory /contrib)

public (root): [maven-release-plugin] prepare for next development iteration

Joe Xie committed 41000 on branch root: in public

[maven-release-plugin] prepare for next development iteration

· /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/trunk/pom.xml (-4, +4) | History | Source | Diff

public (root): [maven-release-plugin] copy for tag atlassian-pdk-2.1.8

Joe Xie committed 40999 on branch root: in public

[maven-release-plugin] copy for tag atlassian-pdk-2.1.8

| | | | | |
|---|------------|---------|--------|------|
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/atlassian-pdk-2.1.8/src | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/atlassian-pdk-2.1.8 | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/.../com/atlassian/maven/plugins | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/.../main/java/com/atlassian/maven | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/.../src/main/java/com/atlassian | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/.../src/main/java/com | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/.../atlassian-pdk-2.1.8/src/main/java | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/atlassian-pdk-2.1.8/src/main | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/.../src/main/resources | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/.../src/main/resources/META-INF | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/tags/.../com/atlassian/maven/plugins/pdk | (-0, +0) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/.../plugins/pdk/DisablePluginMojo.java | (-0, +19) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/.../plugins/pdk/UninstallPluginMojo.java | (-0, +139) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/.../plugins/pdk/GenerateWarMojo.java | (-0, +340) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/.../plugins/pdk/RescanPluginsMojo.java | (-0, +19) | History | Source | Diff |
| ... 10 more files in changeset. | | | | |

public (root): [maven-release-plugin] prepare release atlassian-pdk-2.1.8

Joe Xie committed 40998 on branch root: in public

[maven-release-plugin] prepare release atlassian-pdk-2.1.8

· /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/trunk/pom.xml (-4, +4) | History | Source | Diff

public (root): Make the pdk use basic auth as well as os_username, when uploading with pi

Joe Xie committed 40990 on branch root: in public

Make the pdk use basic auth as well as os_username, when uploading with pi

| | | | | |
|---|-----------|---------|--------|------|
| · /contrib/maven-plugins/com.atlassian.maven.plugins/atlassian-pdk/.../plugins/pdk/InstallPluginMojo.java | (-4, +5) | History | Source | Diff |
| · /contrib/maven-plugins/com.atlassian.maven.plugins/.../plugins/pdk/BasePluginServerMojo.java | (-1, +16) | History | Source | Diff |

public (root): Very basic swing UI added

dkjellin committed 40979 on branch root: in public

Very basic swing UI added

| | | | | |
|--|------------|---------|--------|------|
| · /contrib/tools/atlassian-log-analysis/trunk/src/java/com/atlassian/util/ui/MainFrame.form | (-0, +100) | History | Source | |
| · /contrib/tools/atlassian-log-analysis/trunk/src/java/com/atlassian/util/ui/InputGlassPanel.java | (-0, +192) | History | Source | |
| · /contrib/tools/atlassian-log-analysis/trunk/src/java/com/atlassian/util/ui/InputGlassPanel.form | (-0, +143) | History | Source | |
| · /contrib/tools/atlassian-log-analysis/trunk/src/.../atlassian/util/log/analysis/AccessLogFileParser.java | (-1, +7) | History | Source | Diff |
| · /contrib/tools/atlassian-log-analysis/trunk/src/.../util/log/analysis/TomcatAccessLogRecordFactory.java | (-1, +1) | History | Source | Diff |
| · /contrib/tools/atlassian-log-analysis/trunk/src/java/com/atlassian/util/ui/ProgressDialog.java | (-0, +90) | History | Source | |
| · /contrib/tools/atlassian-log-analysis/trunk/src/java/com/atlassian/util/ui/ProgressDialog.form | (-0, +60) | History | Source | |
| · /contrib/tools/atlassian-log-analysis/trunk/src/.../atlassian/util/log/analysis/LogRecordCollection.java | (-5, +5) | History | Source | Diff |
| · /contrib/tools/atlassian-log-analysis/trunk/src/java/com/atlassian/util/ui/StartListener.java | (-0, +15) | History | Source | |

| | | | | |
|--|------------|-------------------------|------------------------|--|
| -/contrib/tools/atlassian-log-analysis/trunk/src/java/com/atlassian/util/ui | (-0, +0) | History | Source | |
| -/contrib/tools/atlassian-log-analysis/trunk/src/java/com/atlassian/util/ui/MainFrame.java | (-0, +206) | History | Source | |

public (root): [maven-release-plugin] prepare for next development iteration

Dariusz Kordonski committed 40893 on branch root: in public

[maven-release-plugin] prepare for next development iteration

| | | | | |
|---|----------|-------------------------|------------------------|----------------------|
| -/contrib/cargo-test-runner/trunk/pom.xml | (-4, +4) | History | Source | Diff |
|---|----------|-------------------------|------------------------|----------------------|

public (root): [maven-scm] copy for tag cargo-test-runner-2.4

Dariusz Kordonski committed 40892 on branch root: in public

[maven-scm] copy for tag cargo-test-runner-2.4

| | | | | |
|---|------------|-------------------------|------------------------|----------------------|
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/src | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4 | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/containers.properties | (-0, +111) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/pom.xml | (-0, +81) | History | Source | Diff |
| . | (-0, +58) | History | Source | Diff |
| /contrib/cargo-test-runner/tags/cargo-test-runner-2.4/.../atlassian/cargotestrunner/EnvironmentData.java | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/src/main/java/com/atlassian/cargotestrunner | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/src/main/java/com/atlassian | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/src/main/java/com | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/src/main/java | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/src/main | (-0, +0) | History | Source | Diff |
| . | (-0, +68) | History | Source | Diff |
| /contrib/cargo-test-runner/tags/cargo-test-runner-2.4/.../atlassian/cargotestrunner/webtest/WebTest.java | (-0, +0) | History | Source | Diff |
| . | (-0, +0) | History | Source | Diff |
| /contrib/cargo-test-runner/tags/cargo-test-runner-2.4/src/.../java/com/atlassian/cargotestrunner/webtest | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/cargo-test-runner-2.4/src/.../com/atlassian/cargotestrunner/reporting | (-0, +0) | History | Source | Diff |
| -/contrib/cargo-test-runner/tags/.../cargotestrunner/serverinformation/URLServerInformation.java | (-0, +43) | History | Source | Diff |
| . | (-0, +0) | History | Source | Diff |
| /contrib/cargo-test-runner/tags/cargo-test-runner-2.4/.../com/atlassian/cargotestrunner/serverinformation | (-0, +0) | History | Source | Diff |
| ... 23 more files in changeset. | | | | |

public (root): [maven-release-plugin] prepare release cargo-test-runner-2.4

Dariusz Kordonski committed 40891 on branch root: in public

[maven-release-plugin] prepare release cargo-test-runner-2.4

| | | | | |
|---|----------|-------------------------|------------------------|----------------------|
| -/contrib/cargo-test-runner/trunk/pom.xml | (-4, +4) | History | Source | Diff |
|---|----------|-------------------------|------------------------|----------------------|

public (root): cause I need authentication to svn ;P

Dariusz Kordonski committed 40890 on branch root: in public

cause I need authentication to svn ;P

| | | | | |
|--|----------|-------------------------|------------------------|----------------------|
| /contrib/cargo-test-runner/trunk/src/main/.../com/atlassian/cargotestrunner/PropertiesWebTestLoader.java | (-1, +1) | History | Source | Diff |
|--|----------|-------------------------|------------------------|----------------------|

public (root): fixing compilation error

Dariusz Kordonski committed 40889 on branch root: in public

fixing compilation error

| | | | | |
|--|----------|-------------------------|------------------------|----------------------|
| /contrib/cargo-test-runner/trunk/src/main/.../atlassian/cargotestrunner/testrunner/Junit4TestRunner.java | (-0, +5) | History | Source | Diff |
|--|----------|-------------------------|------------------------|----------------------|

Developing with Eclipse

How to use Eclipse for Confluence plugin development

Eclipse can be used to make Confluence plugin development easier.

General setup

- [Install Eclipse](#)
- Use Subversion
 - [Install the SVN plugin for Eclipse](#)
 - Add the Confluence Subversion site to Eclipse
 - Window->Open perspective and then find the Subversion perspective
 - <https://svn.atlassian.com/svn/public/contrib/confluence/>
- Use Maven
 - Install Maven 1.0.x
 - [Install the Maven \(1.x\) Eclipse plugin](#) by downloading the jar and copying to the Eclipse plugin directory
 - Setup your MAVEN_REPO property in Eclipse to point to your Eclipse workspace location

```
maven eclipse:add-maven-repo -Dmaven.eclipse.workspace=C:/...
```

Plugin specific

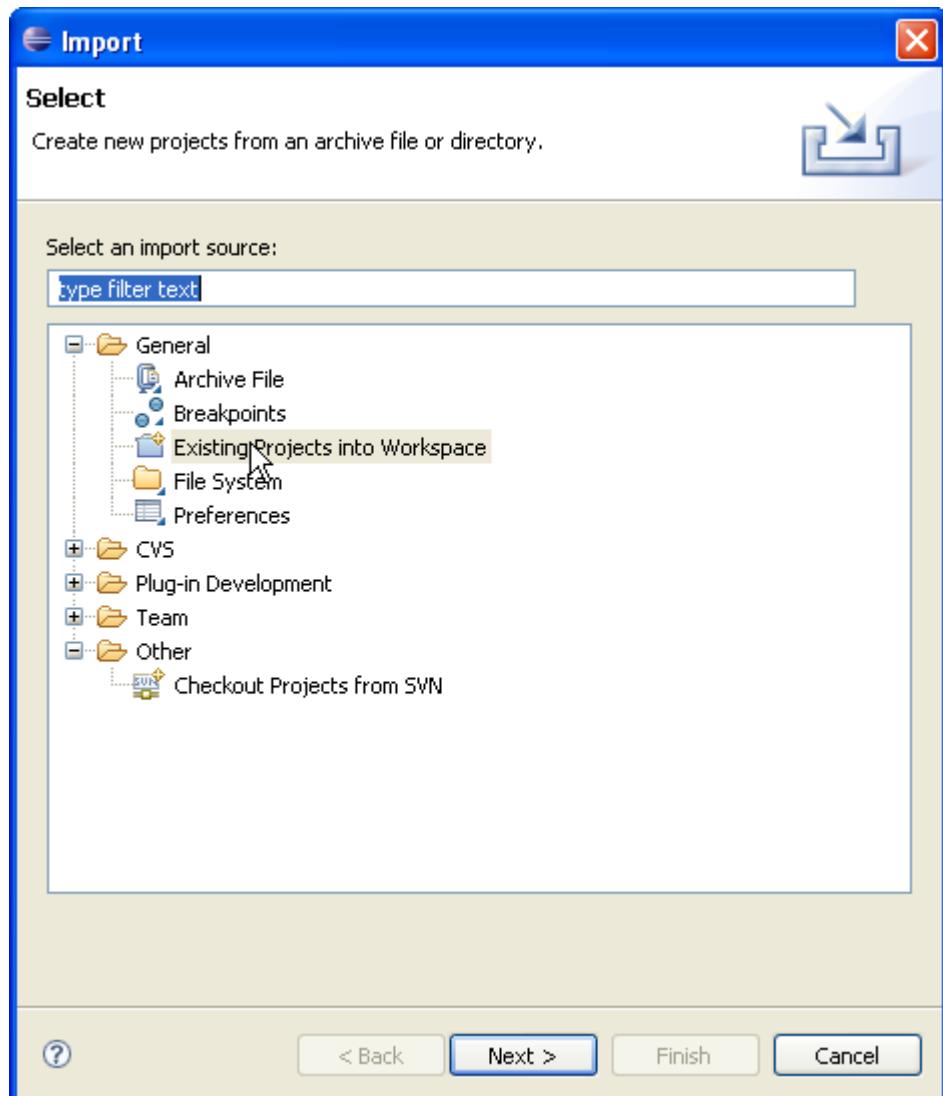
The following assumes you have a somewhat standard plugin directory structure that you created or checked out from Subversion. For example:

```
xxx-plugin
  trunk
    src
      etc
      java
    target
```

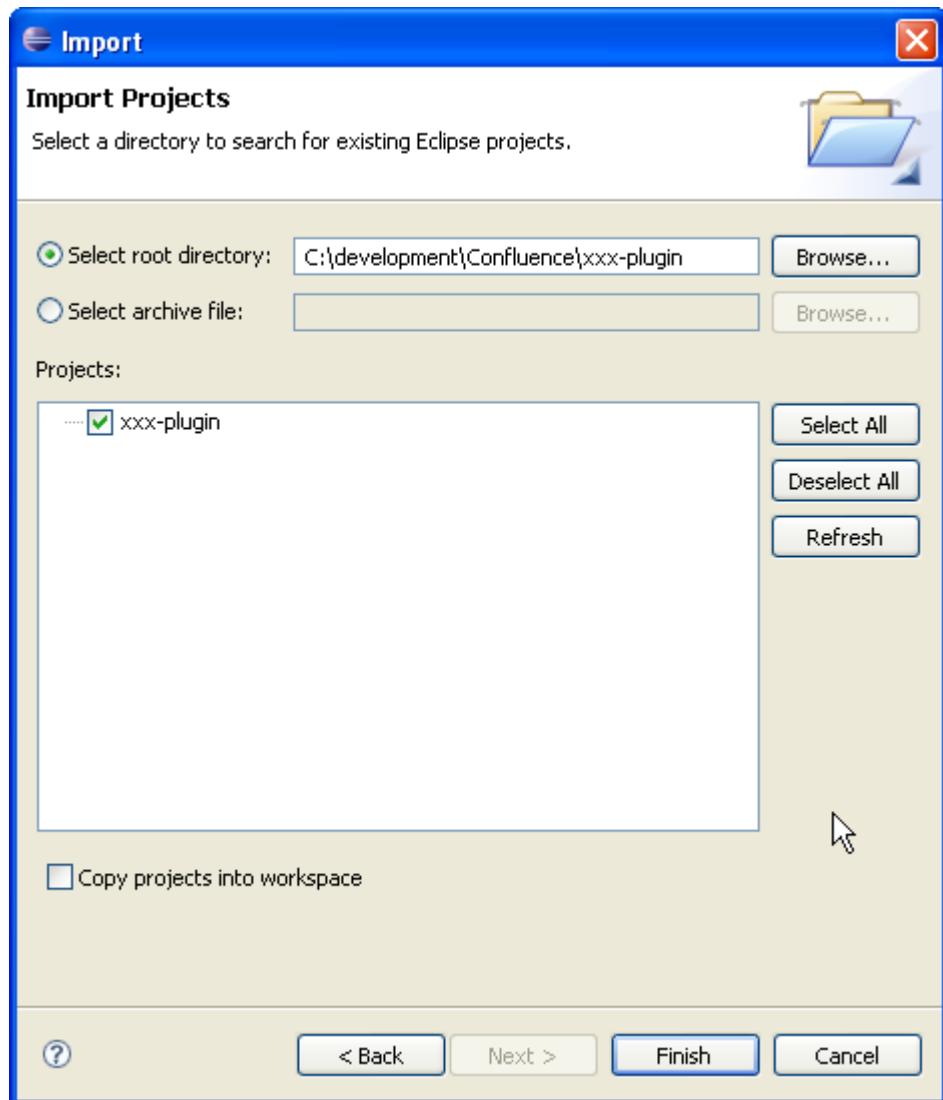
- Edit the project.xml file to make sure it contains all the dependencies needed
 - This must be done before generating the Eclipse project files
 - If the dependencies need to be changed later, the Eclipse project files will need to be re-generated and then in Eclipse, refresh your project using right click -> Refresh
- Generate the Eclipse project and classpath files
 - From a command line, cd to the trunk directory

```
cd .../xxx-plugin/trunk
maven eclipse
```

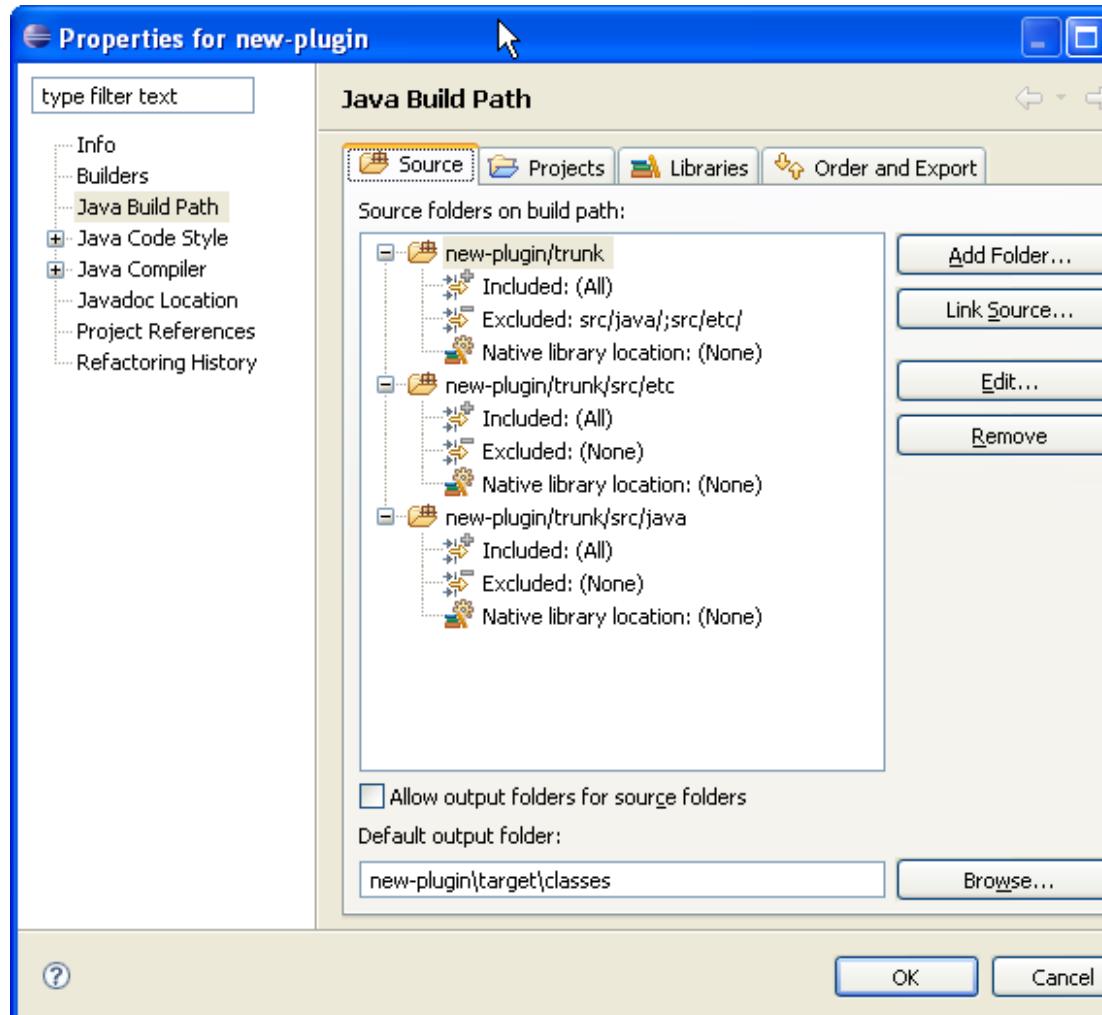
- Move the .project and .classpath files to the xxx-plugin directory
- Import the project by pointing at the plugin directory (that now contains a .project file!)
 - Right click **Import**



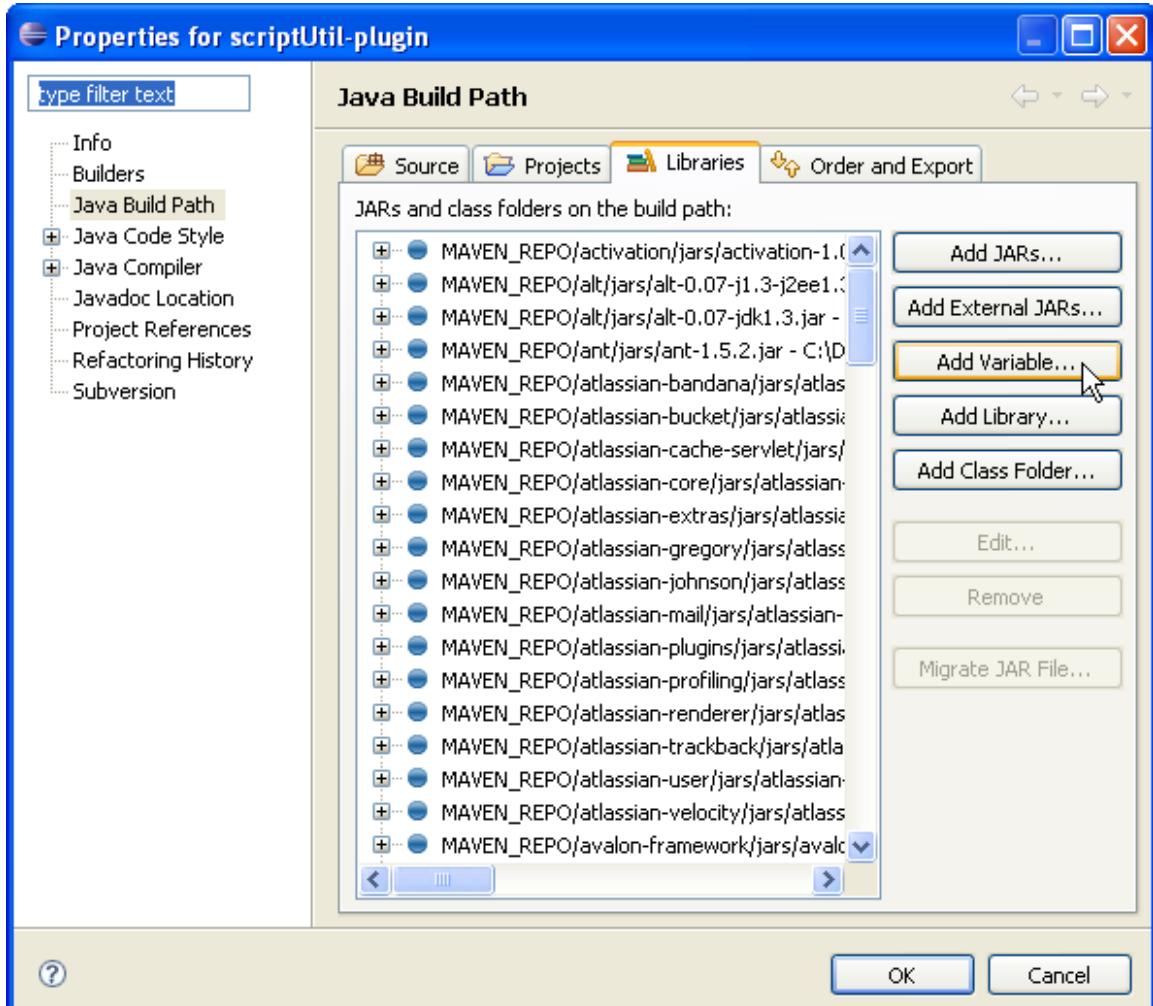
- Choose **Existing Projects into Workspace**



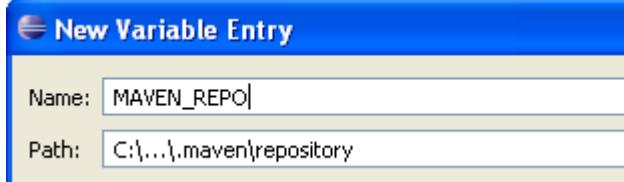
- Customize the build path **source** using right click on project -> Build Path -> Configure Build Path
 - removing the default location
 - adding /trunk/src/etc
 - adding /trunk/src/java
 - adding /trunk (last)



- Add MAVEN_REPO classpath variable to Eclipse
 - This only needs to be done for the first plugin you setup
 - MAVEN_REPO is used in the Maven generated .classpath file that contains references to all the dependencies defined to Maven in the project.xml and related files
 - Customize the build path **libraries** using right click on project -> Build Path -> Configure Build Path



- Add a new variable that points to your maven repository directory



Summary

- Eclipse should now analyse the plugin code and report errors
- Once all errors are fixed in Eclipse, a maven build (run outside Eclipse) should be clean

Guide to Testing Plugins

This page is designed to explain how to utilize Maven 2's testing facilities. Here are the following topics we will discuss:

1. Ports and Functional Testing

Ports and Functional Testing

For convenience, you can use the following variables to override the ports specified elsewhere.

Optional Variables

| Variable | Default | Type | Purpose |
|-----------------|------------|--------|--|
| http.port | No Default | String | Port on which cargo will start up to run integration-tests |
| controller.port | No Default | String | Port on which cargo will use as a controller for the cargo container |

Example of Usage

```
$ mvn integration-test -Dhttp.port=10548 -Dcontroller.port=10549
```

Initial development environment setup

The first time you set out to develop, you will need to make sure you have the supporting environment set up.

1. [DOC:Java Development Kit](#)
2. [DOC:Maven](#)
3. [DOC:Application Server](#)
4. [DOC:Database](#)
5. [DOC:Subversion](#)
6. [IDE](#)

Java Development Kit (JDK)

Any of the following should work:

- Sun JDK 1.4.2_x
- Sun JDK 1.5.0_x
- Sun JDK 1.6.0_x
- Jikes

Follow the installation instructions for your choice and platform.

If you're planning to distribute your work to the Confluence user community, we strongly encourage you to use JDK 1.4 instead of 1.5 or 1.6. We make sure that Confluence is compatible with JDK 1.4, and many Confluence users still run on the older JVM. If you're developing for your own use only, you are free to use the JDK version your internal Confluence instance is running on.

Maven 2.0.6

Quoting its [homepage](#), "Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information."

Both Confluence and Jira are set up to use Maven. And although it's not a requirement, most of the open-source plugins in the [Developer Subversion Repository](#) also use Maven. This guide assumes you will be using Maven for your plugins.

Confluence 2.3 and later is built using Maven 2. To configure Maven on your system:

1. [Download and install Maven 2.0.6 or later](#)
2. Set up your environment variables:
 - Define `$MAVEN_HOME` in your shell config
 - Add `$MAVEN_HOME/bin` to your path
3. Add the [DOC:Atlassian Maven repositories](#) to your Maven settings file.

Application Server Setup

You can use the Atlassian-IDEA Maven plugin to configure deployment environments for Resin 2, Resin 3, Tomcat 5.5.x and Orion 2. You can run on any or all of these appservers. Unless you're testing compatibility, you probably only need one. Resin seems to be the most popular, but Tomcat is the basis for the standalone dist, so either of those are good choices.

Tomcat Installation

1. [Download & install Tomcat 5.5.x](#)
2. If you want to run Tomcat 5.5 on a 1.4 JDK, download the [Compatibility package](#) and install it.

Resin 3 Installation

1. [Download & install Resin 3.0.x \(see also \[jira documentation\]\(#\)\)](#)

Resin 2 Installation

1. [Download & install Resin 2.1.x \(see also \[jira documentation\]\(#\)\)](#)

Orion 2 Installation

1. [Download & install Orion 2.0.x \(see also \[jira documentation\]\(#\)\)](#)

Database Setup (optional)

While both JIRA and Confluence come with the HSQLDB file-based database, it is sometimes useful to have a non-file-based database for development - it's faster and it's easier to find out what's going on inside. However, there are other cases where it is completely appropriate to stick with the HSQLDB as provided. If you want to use the standalone database, you can skip this step.

If you're using a database other than HSQLDB, then you need to create a database and a user which can access that database. If you're using HSQLDB, skip this step.

A wide variety of databases are supported. MySQL 4.1 is used in this example. Follow a similar procedure for your database of choice.

1. Download & install MySQL 4.1.
2. Make sure the database is running and will restart after reboot.
3. Create a new database (eg. '**confdb**').
4. Create a new username/password (eg. '**confuser**'/'**confuser**').
5. Grant the new user full rights on the new database.
6. Download & install the database driver into \$TOMCAT_HOME/common/lib, \$RESIN_HOME/lib, \$RESIN3_HOME/lib and/or \$ORION_HOME/???.

Subversion (optional)

Atlassian hosts a [Subversion](#) repository for developers to contribute open-source plugins. If you want to contribute your plugin, or get the latest version of helpful development resources, installing a subversion client is recommended.

Most modern IDEs include some support for Subversion, but you can also install the command-line client. This can also be helpful if the IDE's implementation doesn't quite do what you expected. You can find out where to get a command-line client (and other types of clients too) here:

http://subversion.tigris.org/project_packages.html

IDE

Now that the basic development environment is ready, we need to the the IDE (Integrated Development Environment) set up. Development can be done with any Java development environment. At Atlassian, we prefer IntelliJ's IDEA, and our instructions follow that path. We also have some user-contributed documentation about [Developing with Eclipse](#).

JetBrains IntelliJ IDEA Installation

1. Download & install the latest version of IDEA (5.0 or later).

Eclipse

1. Download & install the latest version of Eclipse.

What's Next?

Now that we've (finally) got the pieces in place, the next stage is to get Confluence setup and running inside IDEA. How you do this will depend on whether or not you have access to the Confluence source code. Currently this is only available to [commercial licensees](#). Choose your setup type:

I have the source code

I have a distribution

Related topics

[Increase memory for IDEA]

Atlassian Maven repositories OLD

This document covers the configuration of Maven repositories for Atlassian projects built using Maven 2. This includes:

- Confluence 2.3 or later
- Confluence dependencies such as atlassian-user, seraph, atlassian-renderer
- many recent Confluence plugins
- any other Atlassian project with a `pom.xml` file in the top folder.

Creating a personal Maven configuration file

Create a file in the following location which will include your personal settings for Maven 2:

```
~/.m2/settings.xml
```

For Windows users, this means C:\Documents and Settings\<username>\.m2\settings.xml.

The following file contains the required repositories for building Atlassian's software and contributed plugins. If you already have a settings.xml file, you need to update your pluginGroups settings and add the repositories below to an active profile.

```
<settings>
  <pluginGroups>
    <pluginGroup>com.atlassian.maven.plugins</pluginGroup>
    <pluginGroup>org.apache.maven.plugins</pluginGroup>
  </pluginGroups>
  <profiles>
    <profile>
      <id>default-profile</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <repositories>
        <repository>
          <id>atlassian-m2-repository</id>
          <name>Atlassian Maven 2 Repository</name>
          <url>http://repository.atlassian.com/maven2</url>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
        <repository>
          <id>atlassian-m2-contrib</id>
          <name>Atlassian Maven 2 Contributor Repository</name>
          <url>http://svn.atlassian.com/svn/public/contrib/repository/maven2</url>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
        <repository>
          <id>atlassian-ml</id>
          <url>http://repository.atlassian.com</url>
          <layout>legacy</layout>
        </repository>
      </repositories>
    </profile>
  </profiles>
</settings>
```

Installing non-distributable libraries

Building some projects will require the Java Transaction API (JTA), which cannot be distributed by Atlassian. You need to download this from the Sun website, and install it with:

```
mvn install:install-file \
-Dfile=../jta-1_0_1B-classes.zip \
-DgroupId=javax.transaction \
-DartifactId=jta -Dversion=1.0.1B \
-Dpackaging=jar
```

What next?

Continue with the [Initial development environment setup](#).

Example settings.xml OLD

This is an example settings.xml for Maven 2. It can be placed in your \$HOME/.m2/ directory and it will apply to all maven projects that you build. If you would rather, you can make this a per-project settings by including a profile.xml in your project base directory. See [Maven's documentation on Build Profile Settings](#).

```
<settings>
  <pluginGroups>
    <pluginGroup>com.atlassian.confluence.maven.plugins</pluginGroup>
    <pluginGroup>com.atlassian.maven.plugins</pluginGroup>
  </pluginGroups>
  <mirrors>
    <mirror>
      <id>ibiblio.net</id>
      <url>http://www.ibiblio.net/pub/packages/maven2</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
  <servers>
    <server>
      <id>atlassian-m2-repository</id>
    </server>
    <server>
      <id>atlassian-m1-repository</id>
    </server>
  </servers>
  <profiles>
    <profile>
      <id>default</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <repositories>
        <repository>
          <id>atlassian-m2-repository</id>
          <name>Atlassian Maven 2.x Repository</name>
          <url>http://repository.atlassian.com/maven2</url>
          <snapshots>
            <enabled>true</enabled>
            <updatePolicy>interval:30</updatePolicy>
          </snapshots>
        </repository>
        <repository>
          <id>atlassian-m1-repository</id>
          <name>Atlassian Maven 1.x Repository</name>
          <url>http://repository.atlassian.com</url>
          <layout>legacy</layout>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
        <repository>
          <id>Codehaus</id>
          <name>Codehaus Repository</name>
          <url>http://repository.codehaus.org/</url>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>atlassian-m2-repository</id>
          <name>Atlassian Maven 2.x Repository</name>
          <url>http://repository.atlassian.com/maven2</url>
          <snapshots>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
      <properties>
        <atlassian.idea.tomcat_env.jdk.name>1.4.2</atlassian.idea.tomcat_env.jdk.name>
        <atlassian.idea.jdk.name>1.4.2</atlassian.idea.jdk.name>
        <atlassian.pdk.server.url>http://localhost:8080</atlassian.pdk.server.url>
        <atlassian.pdk.server.username>admin</atlassian.pdk.server.username>
        <atlassian.pdk.server.password>admin</atlassian.pdk.server.password>
        <atlassian.idea.resin.location>C:\resin-2.1.17</atlassian.idea.resin.location>
      </properties>
    </profile>
  </profiles>

```

```
<atlassian.idea.resin3.location>C:\resin-3.0.19\resin-3.0.19</atlassian.idea.resin3.location>
<atlassian.idea.tomcat.location>C:\apache-tomcat-5.5.17</atlassian.idea.tomcat.location>
<atlassian.idea.orion.location>C:\orion2.0.7\orion</atlassian.idea.orion.location>
</properties>
```

```
</profile>
</profiles>
</settings>
```

List of Product Dependencies

This page shows the dependencies for the **latest** release of each product. To find the dependencies for earlier versions of the product, please refer to the children of this page — select the page corresponding to the product and version you want.

 Many, but not all, of these are open source components.

In this page:

- [Atlassian IDE Plugin 1.0 Dependencies](#)
- [Bamboo 2.2 Dependencies](#)
- [Clover 2.2 Dependencies](#)
- [Confluence 2.8 Dependencies](#)
- [Crowd 1.3.2 Dependencies](#)
- [FishEye/Crucible 1.5 Dependencies](#)
- [JIRA 4.0 Dependencies](#)

Atlassian IDE Plugin 1.0 Dependencies

```
org.picoccontainer(picoccontainer;jar)
org.easymock(easymock;jar)
com.intellij.idea(openapi;jar:7590)
gnu.trove(trove4j;jar)
org.ddsteps(ddsteps-httserver-mock;jar)
org.mortbay.jetty(jetty;jar)
org.mortbay.jetty(jetty-util;jar)
org.mortbay.jetty(servlet-api-2.5;jar)
junit-addons(junit-addons;jar)
junit(junit;jar)
com.intellij.idea(forms_rt;jar:7590)
xmlrpc(xmlrpc;jar)
joda-time(joda-time;jar)
log4j(log4j;jar)
com.intellij.idea(bootstrap;jar:7590)
com.intellij.idea(annotations;jar:7590)
net.sourceforge.htmlunit(htmlunit;jar)
jaxen(jaxen;jar)
commons-collections/commons-collections;jar
commons-lang/commons-lang;jar
rhino(js;jar)
xerces(xmlParserAPIs;jar)
nekohtml(nekohtml;jar)
xerces(xercesImpl;jar)
commons-io/commons-io;jar
com.intellij.idea(jgoodies-forms;jar:7590)
com.intellij.idea(javac2;jar:7590)
asm(asm-commons;jar)
asm(asm-tree;jar)
asm(asm;jar)
jdom(jdom;jar)
ant(ant;jar)
com.jgoodies(forms;jar)
com.intellij.idea/extensions;jar:7590)
net.sourceforge.cssparser(cssparser;jar;jdk15)
commons-httpclient/commons-httpclient;jar
commons-logging/commons-logging;jar
commons-codec/commons-codec;jar
```

Bamboo 2.2 Dependencies

| | |
|----------------|-------|
| acegi-security | 1.0.4 |
| activation | 1.1.1 |
| activeio-core | 3.0.1 |
| activemq-core | 4.1.2 |

| | |
|--|--------|
| activemq-ra | 4.1.2 |
| alt-jdk1.3 | 0.07 |
| annotations | 6.0.5 |
| ant | 1.7.1 |
| ant-launcher | 1.7.1 |
| aopalliance | 1.0 |
| asm | 1.5.3 |
| asm-util | 1.3.4 |
| aspectjrt | 1.5.3 |
| aspectjweaver | 1.5.3 |
| aspectwerkz-core | 0.8.1 |
| atlassian-aws | 0.8 |
| atlassian-bamboo-agent-bootstrap | 2.2 |
| atlassian-bamboo-agent-classserver | 2.2 |
| atlassian-bamboo-agent-core | 2.2 |
| atlassian-bamboo-agent-elastic | 2.2 |
| atlassian-bamboo-agent-elastic-server | 2.2 |
| atlassian-bamboo-agent-installer | 2.2 |
| atlassian-bamboo-agent-local | 2.2 |
| atlassian-bamboo-agent-remote | 2.2 |
| atlassian-bamboo-api | 2.2 |
| atlassian-bamboo-charts | 2.2 |
| atlassian-bamboo-core | 2.2 |
| atlassian-bamboo-language | 2.2 |
| atlassian-bamboo-license | 2.2 |
| atlassian-bamboo-migration | 2.2 |
| atlassian-bamboo-persistence | 2.2 |
| atlassian-bamboo-plugin-autofavourite | 2.2 |
| atlassian-bamboo-plugin-buildnumberstamper | 2.2 |
| atlassian-bamboo-plugin-clover | 2.2 |
| atlassian-bamboo-plugin-labeller | 2.2 |
| atlassian-bamboo-plugin-vcsversion | 2.2 |
| atlassian-bamboo-upgrader | 2.2 |
| atlassian-bamboo-web | 2.2 |
| atlassian-bonnie | 2.8 |
| atlassian-bucket | 0.17 |
| atlassian-config | 0.9 |
| atlassian-core | 4.0 |
| atlassian-event | 0.5 |
| atlassian-extras | 1.20.2 |
| atlassian-johnson | 0.9 |
| atlassian-mail | 1.9 |

| | |
|--------------------------|-----------------------|
| atlassian-plugins | 0.13 |
| atlassian-profiling | 1.4 |
| atlassian-seraph | 0.20 |
| atlassian-spring | 0.7 |
| atlassian-tunnel | 0.3 |
| atlassian-user | 1.9 |
| atlassian-velocity | 0.5 |
| atlassian-xwork-12 | 1.3 |
| atlassian-xwork-core | 1.3 |
| axis | 1.4 |
| axis-jaxrpc | 1.4 |
| axis-saaj | 1.4 |
| axis-wsdl4j | 1.5.1 |
| backport-util-concurrent | 2.1 |
| bamboo-agent | 2.2 |
| bamboo-agent-bootstrap | 2.2-with-dependencies |
| bamboo-jira-soap-client | 1.0 |
| c3p0 | 0.9.1.1 |
| cglib | 2.1 |
| cglib-nodep | 2.1_3 |
| commons-attributes-api | 2.1 |
| commons-beanutils | 1.8.0 |
| commons-codec | 1.3 |
| commons-collections | 3.1 |
| commons-configuration | 1.4 |
| commons-dbcp | 1.2 |
| commons-digester | 1.8 |
| commons-discovery | 0.2 |
| commons-el | 1.0 |
| commons-httpclient | 3.0.1 |
| commons-io | 1.4-backport-IO-168 |
| commons-jxpath | 1.2 |
| commons-lang | 2.3 |
| commons-logging | 1.0.4 |
| commons-logging-api | 1.0.4 |
| commons-pool | 1.4-RC2-atlassian-1 |
| crowd-integration-client | 1.6 |
| dom4j | 1.4 |
| dwr | 1.1 |
| easymock | 2.3 |
| ehcache | 1.2.3 |
| exml | 7.0 |

| | |
|-----------------------------------|------------|
| freemarker | 2.3.14 |
| geronimo-connector | 2.0.1 |
| geronimo-j2ee-connector_1.5_spec | 1.0 |
| geronimo-j2ee-management_1.0_spec | 1.0 |
| geronimo-jms_1.1_spec | 1.1.1 |
| geronimo-jta_1.0.1B_spec | 1.0.1 |
| geronimo-jta_1.1_spec | 1.0 |
| geronimo-spec-jta | 1.0.1B-rc4 |
| geronimo-transaction | 2.0.1 |
| hibernate | 2.1.8 |
| howl | 1.0.1-1 |
| hsqldb | 1.8.0.7 |
| isorelax | 20020414 |
| jasper-compiler | 5.0.28 |
| jasper-runtime | 5.0.28 |
| javacvs-atlassian | 20080407 |
| jaxb-api | 2.1 |
| jaxb-impl | 2.1.3 |
| jaxen | 1.0-FCS |
| jcip-annotations | 1.0 |
| jcommon | 1.0.12 |
| jdom | 1.0 |
| jencks | 2.1 |
| jencks-amqpool | 2.1 |
| jets3t | 0.6.1 |
| jettison | 1.0-RC2 |
| jetty | 6.1.5 |
| jetty-management | 6.1.5 |
| jetty-naming | 6.1.5 |
| jetty-plus | 6.1.5 |
| jetty-util | 6.1.5 |
| jfreechart | 1.0.9 |
| jmock | 1.0.1 |
| jms | 1.1 |
| jna | 3.0.3 |
| joda-time | 1.4 |
| jsch | 0.1.31 |
| jsp-api | 2.0 |
| jta | 1.0.1B |
| junit | 3.8.1 |
| ldaptemplate | 1.0.1 |
| lingo | 1.3 |

| | |
|--------------------------------|---------------------|
| log4j | 1.2.12 |
| lucene-analyzers | 2.3.2 |
| lucene-core | 2.3.2 |
| mail | 1.4.1 |
| mockobjects-alt-jdk1.3-j2ee1.3 | 0.09 |
| mockobjects-core | 0.09 |
| mockobjects-j1.3-j2ee1.3 | 0.07 |
| msv | 20020414 |
| nant_builder | 2.1.0 |
| odmg | 3.0 |
| ognl | 2.6.11 |
| oro | 2.0.8 |
| oscache-DEV | |
| oscore | 2.2.7 |
| osuser-atl.user | |
| p4java | 0.7.5-atlassian-5.5 |
| p6spy | 1.3 |
| pdfbox | 0.7.0 |
| pjl-comp-filter | 1.4 |
| poi | 2.0-final 20040126 |
| polsUtils | 1.0.18 |
| propertyset | 1.3 21Nov03 |
| qdox | 1.2 |
| quartz | 1.5.2 |
| relaxngDatatype | 20020414 |
| rife-continuations | 0.0.2 |
| rome | 0.8 |
| saxpath | 1.0-FCS |
| servlet-api | 2.4 |
| servlet-api | 2.5-6.1.5 |
| sitemesh | 2.2.1 |
| smack | 3.0.4 |
| smackx | 3.0.4 |
| spring | 2.0.7 |
| spring-aop | 2.0.7 |
| spring-beans | 2.0.7 |
| spring-context | 2.0.7 |
| spring-core | 2.0.7 |
| spring-dao | 2.0.7 |
| spring-hibernate2 | 2.0.7 |
| spring-jdbc | 2.0.7 |
| spring-jms | 2.0.7 |

| | |
|---------------------|----------------|
| spring-mock | 2.0.7 |
| spring-support | 2.0.7 |
| spring-web | 2.0.7 |
| stax-api | 1.0-2 |
| stax-api | 1.0.1 |
| stax-utils | 20040917 |
| svnkit | 1.2.1.5297 |
| tm-extractors | 0.4 |
| trilead-ssh2 | build213 |
| truezip | 6.6 |
| typica | 1.4.1 |
| urlrewrite | 2.6.0 |
| velocity | 1.4 |
| velocity-dep | 1.4 |
| velocity-tools | 1.2 |
| webwork | 2.2.5 |
| wrapper | 3.2.0 |
| wsdl4j | 1.6.1 |
| wstx-asl | 3.2.4 |
| xapool-without-pool | 1.4 |
| xbean-spring | 3.2 |
| xercesImpl | 2.8.1 |
| xfire-aegis | 1.2.6 |
| xfire-annotations | 1.2.6 |
| xfire-core | 1.2.6 |
| xfire-java5 | 1.2.6 |
| xfire-jsr181-api | 1.0-M1 |
| xml-apis | 1.3.03 |
| xmlParserAPIs | 2.2.1 |
| XmISchema | 1.1 |
| xpp3_min | 1.1.3.4.O |
| xstream | 1.3 |
| xwork | 1.2.3 20070717 |

Clover 2.2 Dependencies

| Library | Version | License |
|--------------------------|---------|------------------------|
| antlr | 2.7.1 | BSD License |
| backport-util-concurrent | 3.0 | Public Domain |
| fastutil | 4.4.3 | LGPL |
| itext | 0.96 | Mozilla Public License |
| jfreechart | 1.0.8 | LGPL |
| jcommon | 1.0.12 | LGPL |

| | | |
|----------|---------|-------------------|
| log4j | 1.2.13 | Apache License V2 |
| overLIB | 4.2.1 | Artistic License |
| velocity | 1.2-RC3 | Apache License V2 |

Confluence 2.8 Dependencies

| | |
|---|---------------|
| FontBox:FontBox | 0.1.0-dev |
| alt:alt | 0.07-jdk1.3 |
| alt:alt-0.07 | j1.3-j2ee1.3 |
| angosol-coherence:coherence | 3.3 |
| angosol-coherence:tangosol | 3.3 |
| ant:ant | 1.5.2 |
| aopalliance:aopalliance | 1.0 |
| asm:asm | 1.5.3 |
| asm:asm-util | 1.3.4 |
| aspectj:aspectjweaver | 1.5.3 |
| aspectwerkz:aspectwerkz-core | 0.8.1 |
| avalon-framework:avalon-framework | 4.2.0 |
| axis:axis | 1.2.1 |
| axis:axis-jaxrpc | 1.2.1 |
| axis:axis-saaj | 1.2.1 |
| backport-util-concurrent:backport-util-concurrent | 3.0 |
| batik:batik-all | 1.6 |
| bouncycastle:bcpprov-jdk14 | 136 |
| bsf:bsf | 2.3.0 |
| c3p0:c3p0 | 0.9.1.2 |
| cglib:cglib | 2.1 |
| com.servlets:cos | 09May2002 |
| com.sun:jai_codec | 1.1.3 |
| com.sun:jai_core | 1.1.3 |
| commons-beanutils:commons-beanutils | 1.6.1 |
| commons-codec:commons-codec | 1.3 |
| commons-collections:commons-collections | 3.2 |
| commons-digester:commons-digester | 1.5 |
| commons-discovery:commons-discovery | 0.2 |
| commons-fileupload:commons-fileupload | 1.2.1 |
| commons-httpclient:commons-httpclient | 3.0 |
| commons-io:commons-io | 1.2 |
| commons-jrcs:commons-jrcs | diff-0.1.7 |
| commons-lang:commons-lang | 2.1 |
| commons-logging:commons-logging | 1.0.4 |
| concurrent:concurrent | 1.3.4-patched |
| css2parser:ss_css2 | 0.9.3 |

| | |
|--|------------------------|
| dom4j:dom4j | 1.4-full |
| dwr:dwr | 1.1.4 |
| exml:exml | 7.1 |
| fop:fop | trunk-534713-atlassian |
| glue:glue | 5.0b2 |
| hibernate:hibernate | 2.1.8-atlassian |
| hsqldb:hsqldb | 1.8.0.1 |
| httpunit:httpunit | 1.5.4 |
| javax.activation:activation | 1.0.2 |
| javax.mail:mail | 1.3.3 |
| javax.servlet:servlet-api | 2.3 |
| javax.transaction:jta | 1.0.1B |
| jcaptcha:jcaptcha | all-1.0-RC2.0.1 |
| jdom:jdom | 1.0 |
| jfree:jcommon | 1.0.0 |
| jfree:jfreechart | 1.0.0 |
| joda-time:joda-time | 1.4 |
| jython:jython | 2.1-forked |
| log4j:log4j | 1.2.8 |
| m-extractors:tm-extractors | 0.4 |
| mockobjects:alt-jdk1.3 | 0.07 |
| mockobjects:mockobjects-alt-jdk1.3-j2ee1.3 | 0.09 |
| mockobjects:mockobjects-core | 0.09 |
| mockobjects:mockobjects-jdk1.3-j2ee1.2 | 0.09 |
| mx4j:mx4j | 3.0.1 |
| nekohtml:nekohtml | 0.9.5 |
| net.java.dev.urlrewrite:urlrewrite | 2.6.0 |
| net.sf.ehcache:ehcache | 1.2.3 |
| net.sfldaptemplate:ldaptemplate | 1.0.1 |
| net.sourceforge.jtds:jtds | 1.2 |
| odmg:odmg | 3.0 |
| ofbcore:ofbcore-share | 2.1.1 |
| ognl:ognl | 2.6.5 |
| opensymphony:oscore | 2.2.4 |
| opensymphony:propertyset | 1.3-21Nov03 |
| opensymphony:sitemesh | 2.3-atlassian |
| opensymphony:xwork | 1.0.3 |
| org.apache.lucene:lucene-analyzers | 2.2.0 |
| org.apache.lucene:lucene-core | 2.2.0 |
| org.codehaus.woodstox:wstx-asl | 3.2.0 |
| org.codehaus.xfire:xfire-all | 1.2.6 |
| org.hibernate:jtidy | r8-20060801 |

| | |
|---|------------------------|
| org.slf4j:slf4j-api | 1.4.3 |
| org.slf4j:slf4j-log4j12 | 1.4.3 |
| org.springframework:spring-aop | 2.0.6 |
| org.springframework:spring-beans | 2.0.6 |
| org.springframework:spring-context | 2.0.6 |
| org.springframework:spring-core | 2.0.6 |
| org.springframework:spring-dao | 2.0.6 |
| org.springframework:spring-hibernate2 | 2.0.6 |
| org.springframework:spring-jdbc | 2.0.6 |
| org.springframework:spring-jmx | 2.0.6 |
| org.springframework:spring-support | 2.0.6 |
| org.springframework:spring-web | 2.0.6 |
| oro:oro | 2.0.7 |
| oscache:oscache | 2.2 |
| osuser:osuser | atl.user |
| pdfbox:pdfbox | 0.7.2 |
| poi:poi | 2.0-final-20040126 |
| quartz:quartz | 1.5.2 |
| radeox:radeox | 1.0b2-forked-22Apr2004 |
| rome:rome | 0.8 |
| saxon:saxon | 6.5.3-patched |
| slide:slide | 2.1 |
| spring:spring-aopalliance | 1.0 |
| stax:stax-api | 1.0.1 |
| velocity-tools:velocity-tools | view-1.1 |
| velocity:velocity | 1.5 |
| wsdl4j:wsdl4j | 1.5.1 |
| xalan:xalan | 2.7.0 |
| xerces:xercesImpl | 2.2.1 |
| xerces:xmlParserAPIs | 2.2.1 |
| xml-apis:xml-apis | 1.0.b2 |
| xmlgraphics-commons:xmlgraphics-commons | 1.2svn |
| xmlrpc:xmlrpc | 2.0+xmlrpc61 |
| xpp3:xpp3 | 1.1.3.4d_b4_min |
| xstream:xstream | 1.1.1 |

Crowd 1.3.2 Dependencies

| | |
|---|-------|
| antlr:antlr | 2.7.6 |
| asm:asm | 1.5.3 |
| asm:asm-attrs | 1.5.3 |
| backport-util-concurrent:backport-util-concurrent | 3.0 |
| bouncycastle:bcpprov-jdk14 | 138 |

| | |
|---|--------------|
| c3p0:c3p0 | 0.9.1.2 |
| cenqua:fisheye | 1.2.1 |
| cglib:cglib | 2.1_3 |
| com.ibm.icu:icu4j | 3.4.4 |
| com.jivesoftware:jiveforums-atlassian | 5.5.1 |
| com.opensymphony:webwork | 2.2.6 |
| com.sun.jndi.ldap:ldapbp | 1.0 |
| commons-codec:commons-codec | 1.2 |
| commons-codec:commons-codec | 1.3 |
| commons-collections:commons-collections | 3.2 |
| commons-httpclient:commons-httpclient | 3.0.1 |
| commons-io:commons-io | 1.2 |
| commons-lang:commons-lang | 2.3 |
| commons-logging:commons-logging | 1.0.4 |
| dom4j:dom4j | 1.4 |
| freemarker:freemarker | 2.3.4 |
| isorelax:isorelax | 20020414 |
| javax.activation:activation | 1.1 |
| javax.mail:mail | 1.4 |
| javax.servlet:servlet-api | 2.3 |
| javax.transaction:jta | 1.0.1B |
| jaxen:jaxen | 1.0-FCS |
| jaxen:jaxen | 1.1-beta-9 |
| jdom:jdom | 1.0 |
| joda-time:joda-time | 1.4 |
| log4j:log4j | 1.2.8 |
| mockobjects:alt-jdk1.3 | 0.07 |
| msv:msv | 20020414 |
| net.java.dev.stax-utils:stax-utils | 20040917 |
| net.java.dev.urlrewrite:urlrewrite | 2.6.0 |
| net.sf.ehcache:ehcache | 1.2.3 |
| opensymphony:ognl | 2.6.11 |
| opensymphony:oscore | 2.2.4 |
| opensymphony:osuser | 1.0-20060106 |
| opensymphony:propertyset | 1.3-21Nov03 |
| opensymphony:sitemesh | 2.2.1 |
| opensymphony:xwork | 1.2.3 |
| org.acegisecurity:acegi-security | 1.0.5 |
| org.apache.ws.commons:XmlSchema | 1.1 |
| org.apache.xbean:xbean-spring | 2.8 |
| org.codehaus.woodstox:wstx-asl | 3.2.4 |
| org.codehaus.xfire:xfire-aegis | 1.2.6 |

| | |
|---------------------------------------|-----------------------------|
| org.codehaus.xfire:xfire-core | 1.2.6 |
| org.codehaus.xfire:xfire-spring | 1.2.6 |
| org.codehaus.xfire:xfire-xmlbeans | 1.2.6 |
| org.hibernate:hibernate | 3.2.5.ga |
| org.htmlparser:htmlparser | 1.6 |
| org.openid4java:openid4java | 0.9.3 |
| org.openxri:openxri-client | 1.0.1 |
| org.openxri:openxri-syntax | 1.0.1 |
| org.rifers:rife-continuations | 0.0.2 |
| org.springframework.ldap:spring-ldap | 1.2.1 |
| org.springframework:spring | 2.5.2 |
| org.springframework:spring-beans | 2.5.2 |
| org.springframework:spring-core | 2.5.2 |
| org.tuckey:urlrewrite | 2.5.2 |
| osuser:osuser | 1.0-dev-log4j-1.4jdk-7Dec05 |
| quartz:quartz | 1.5.1 |
| relaxngDatatype:relaxngDatatype | 20020414 |
| saxpath:sxpath | 1.0-FCS |
| spring-ldap-hacked:spring-ldap-hacked | 1.1.99 |
| stax:stax-api | 1.0.1 |
| wsdl4j:wsdl4j | 1.6.1 |
| xerces:xercesImpl | 2.6.2 |
| xerces:xercesImpl | 2.8.1 |
| xerces:xmlParserAPIs | 2.6.2 |
| xml-apis:xml-apis | 1.3.03 |
| xml-security:xmlsec | 1.3.0 |
| xmlbeans:xbean | 2.2.0 |

FishEye/Crucible 1.5 Dependencies

| | |
|--|-------|
| ant-contrib:ant-contrib | 1.0b2 |
| ant:ant | 1.6.5 |
| antlr:antlr | 2.7.7 |
| asm:asm | 3.1 |
| avalon-framework:avalon-framework | 4.1.3 |
| c3p0:c3p0 | 0.9.0 |
| cglib:cglib-nodep | 2.1_3 |
| colt:colt | 1.2.0 |
| com.atlassian.core:atlassian-core | 3.8 |
| com.atlassian.crowd:crowd-integration-client | 1.3 |
| com.atlassian.extras:atlassian-extras | 1.17 |
| com.atlassian.plugins:atlassian-plugins | 0.10 |
| com.atlassian.profiling:atlassian-profiling | 1.3 |

| | |
|---|--------------|
| com.atlassian.seraph:atlassian-seraph | 0.36 |
| com.atlassian:cenqua-licensing | 1.6 |
| com.boilerbay.infinitydb:infinitydb | v1.0.53_3192 |
| com.octocaptcha:jcaptcha-all | 1.0-RC6 |
| com.sun.xml.bind:jaxb-xjc | 2.0 |
| com.sun.xml.messaging.saaj:saaj-impl | 1.3 |
| com.svnkit:svnkit | 1.1.6 |
| com.svnkit:svnkit-cli | 1.1.6 |
| com.svnkit:svnkit-javahl | 1.1.6 |
| com.svnkit:svnkit-jna | 1.1.6 |
| com.svnkit:svnkit-trilead | 1.1.6 |
| com.zentus.sqlitejdbc:sqlitejdbc | v037-nested |
| commons-beanutils:commons-beanutils | 1.7.0 |
| commons-codec:commons-codec | 1.3 |
| commons-collections:commons-collections | 3.1 |
| commons-dbcp:commons-dbcp | 1.2.1 |
| commons-fileupload:commons-fileupload | 1.2 |
| commons-httpclient:commons-httpclient | 3.1 |
| commons-io:commons-io | 1.3.1 |
| commons-lang:commons-lang | 2.3 |
| commons-logging:commons-logging | 1.1 |
| commons-pool:commons-pool | 1.2 |
| concurrent:concurrent | 1.3.4 |
| dom4j:dom4j | 1.6.1 |
| freemarker:freemarker | 2.3.4 |
| hsqldb:hsqldb | 1.8.0.9 |
| javax.activation:activation | 1.1 |
| javax.mail:mail | 1.4 |
| javax.transaction:jta | 1.0.1B |
| javax.ws.rs:jsr311-api | 0.5ea |
| javax.xml.bind:jaxb-api | 2.0 |
| javax.xml.soap:saaj-api | 1.3 |
| javax.xml.ws:jaxws-api | 2.0 |
| jaxen:jaxen | 1.1.1 |
| jboss:javassist | 3.0 |
| jdom:jdom | 1.0 |
| jfree:jcommon | 1.0.12 |
| jfree:jfreechart | 1.0.9 |
| joda-time:joda-time | 1.4 |
| log4j:log4j | 1.2.11 |
| logkit:logkit | 1.0.1 |
| mockobjects:alt-jdk1.3 | 0.07 |

| | |
|---|-----------------|
| net.java.dev.jersey:jersey | 0.5ea |
| net.java.dev.stax-utils:stax-utils | 20040917 |
| net.java.dev.urlrewrite:urlrewrite | 3.0.4 |
| net.sf.ehcache:ehcache | 1.2.3 |
| net.sf.proguard:proguard | 3.8 |
| ognl:ognl | 2.7.1 |
| opensymphony:oscore | 2.2.6 |
| opensymphony:osuser | 1.0-20060106 |
| opensymphony:propertyset | 1.3-21Nov03 |
| opensymphony:sitemesh | 2.3 |
| opensymphony:webwork | 2.1.7 |
| opensymphony:xwork | 1.0.5 |
| org.apache.cxf:cxf-api | 2.0.3-incubator |
| org.apache.cxf:cxf-common-schemas | 2.0.3-incubator |
| org.apache.cxf:cxf-common-utilities | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-bindings-soap | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-bindings-xml | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-core | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-databinding-jaxb | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-frontend-jaxws | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-frontend-simple | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-transports-http | 2.0.3-incubator |
| org.apache.cxf:cxf-tools-common | 2.0.3-incubator |
| org.apache.geronimo.specs:geronimo-annotation_1.0_spec | 1.1 |
| org.apache.geronimo.specs:geronimo-ws-metadata_2.0_spec | 1.1.1 |
| org.apache.lucene:lucene-core | 2.2.0 |
| org.apache.neethi:neethi | 2.0.2 |
| org.apache.ws.commons.schema:XmlSchema | 1.3.2 |
| org.apache.xmlbeans:xmlbeans | 2.3.0 |
| org.bouncycastle:bcprov-jdk15 | 138 |
| org.codehaus.woodstox:wstx-asl | 3.2.4 |
| org.codehaus.xfire:xfire-aegis | 1.2.6 |
| org.codehaus.xfire:xfire-core | 1.2.6 |
| org.eclipse.jdt:core | 3.1.1 |
| org.hibernate:hibernate | 3.2.6.ga |
| org.mortbay.jetty:jetty | 6.1.6 |
| org.mortbay.jetty:jetty-ajp | 6.1.6 |
| org.mortbay.jetty:jetty-util | 6.1.6 |
| org.mortbay.jetty:jsp-2.1 | 6.1.6 |
| org.mortbay.jetty:jsp-api-2.1 | 6.1.6 |
| org.mortbay.jetty:servlet-api-2.5 | 6.1.6 |
| org.slf4j:slf4j-api | 1.4.3 |

| | |
|----------------------------|-------|
| org.slf4j:slf4j-simple | 1.4.3 |
| org.springframework:spring | 2.5 |
| p6spy:p6spy | 1.3 |
| quartz:quartz | 1.5.1 |
| rove:trove | 1.0.2 |
| stax:stax-api | 1.0.1 |
| sun-jaxb:jaxb-impl | 2.0.5 |
| velocity:velocity | 1.4 |
| wsdl4j:wsdl4j | 1.6.1 |
| xalan:xalan | 2.7.0 |
| xerces:xercesImpl | 2.6.2 |
| xerces:xmlParserAPIs | 2.6.2 |
| xml-resolver:xml-resolver | 1.2 |
| xom:xom | 1.1 |

JIRA 4.0 Dependencies

| | |
|-------------------------|----------|
| atlassian-core | 4.5.2 |
| sal-spi | 2.0.14.1 |
| atlassian-gadgets-api | 1.1.2 |
| atlassian-gadgets-spi | 1.1.2 |
| jsr305 | 1.3.2 |
| atlassian-bandana | 0.1.13 |
| atlassian-scheduler | 0.11 |
| atlassian-velocity | 0.8 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-seraph | 2.0.1 |
| atlassian-cache-api | 1.0 |
| atlassian-cache-memory | 1.0 |
| joda-time | 1.4 |
| atlassian-renderer | 3.14.1 |
| p6spy | 1.3 |
| osworkflow | 2.8.0 |
| oscore | 2.2.7 |
| propertyset | 1.3 |
| commons-digester | 1.4.1 |
| commons-beanutils | 1.6.1 |
| commons-logging | 1.0.4 |
| commons-collections | 3.2 |
| commons-lang | 2.4 |
| commons-configuration | 1.0 |
| commons-io | 1.4 |
| log4j | 1.2.15 |

| | |
|---------------------------|--------------------|
| oro | 2.0.7 |
| velocity | 1.4-atlassian-1 |
| velocity-tools | 1.3 |
| alt | 0.07-jdk1.3 |
| alt+j2ee | 0.07-jdk1.3 |
| junit | 3.8.1 |
| mockobjects | DEV |
| easymock | 2.4 |
| cglib-full | 2.0.1 |
| easymockextension | 2.4 |
| mockobjects+j2ee | 0.07-jdk1.3 |
| mockobjects+jdk | 0.07-jdk1.3 |
| activation | 1.1.1 |
| mail | 1.4.1 |
| servletapi | 2.3 |
| glue | 5.0b2 |
| bsf | 2.2 |
| bsh | 1.2b7 |
| ofbcore+jira+share | 2.1.5 |
| ofbcore+jira+entity | 2.1.5 |
| ofbcore+jira+service | 2.1.5 |
| ofbcore+jira+extutil | 2.1.5 |
| csv | 20 |
| quartz | 1.4.5 |
| picocontainer | 1.0 |
| jzlib | 1.0.5 |
| jsch | 0.1.23 |
| javacvs | 2007-04-04-patched |
| statcvs | 20060222-patched |
| commons-dbcp | 1.1 |
| commons-pool | 1.1 |
| hsqldb | 1.8.0.5 |
| jndi | 1.2.1 |
| jta | 1.0.1 |
| ots-jts | 1.0 |
| jotm | 1.4.3 |
| jotm+jrmp_stubs | 1.4.3 |
| jotm+iiop_stubs | 1.4.3 |
| jotm+jonas_timer | 1.4.3 |
| jotm+objectweb-datasource | 1.4.3 |
| carol | 1.5.2 |
| carol+properties | |

| | |
|---------------------------|------------------------|
| xapool | 1.3.1 |
| xml-apis | 1.3.04 |
| saxon+noaelfred | 6.5.5 |
| commons-jelly | 1.0 |
| commons-jelly+tags-junit | 1.0 |
| commons-jelly+tags-util | 1.1.1 |
| commons-jelly+tags-email | 1.0 |
| commons-jelly+tags-log | 1.0 |
| commons-jelly+tags-http | 1.0 |
| commons-jelly+tags-soap | 1.0 |
| commons-jelly+tags-sql | 1.0 |
| commons-jelly+tags-regexp | 1.0 |
| commons-jexl | 1.1 |
| dom4j | 1.4 |
| commons-httpclient | 3.0.1 |
| commons-codec | 1.3 |
| xmlrpc | 2.0 |
| axis | 1.3 |
| xpp3 | 1.1.3.4-RC8 |
| xstream | 1.1.1 |
| jfreechart | 1.0.4 |
| jfree | 1.0.8 |
| radeox | 1.0b2-forked-22Apr2004 |
| jtidy | r8-20050104 |

Atlassian IDE Plugin 1.0 Dependencies

Below is a straight list of the dependencies. You can also take a look at the [attached file](#) for a breakdown of the list.

```
org.picocontainer:picocontainer:jar
org.easymock:easymock:jar
com.intellij.idea:openapi:jar:7590
gnu.trove:trove4j:jar
org.ddsteps:ddsteps-httpsserver-mock:jar
org.mortbay.jetty:jetty:jar
org.mortbay.jetty:jetty-util:jar
org.mortbay.jetty:servlet-api-2.5:jar
junit-addons:junit-addons:jar
junit:junit:jar
com.intellij.idea:forms_rt:jar:7590
xmlrpc:xmlrpc:jar
joda-time:joda-time:jar
log4j:log4j:jar
com.intellij.idea:bootstrap:jar:7590
com.intellij.idea:annotations:jar:7590
net.sourceforge.htmlunit:htmlunit:jar
jaxen:jaxen:jar
commons-collections:commons-collections:jar
commons-lang:commons-lang:jar
rhino:js:jar
xerces:xmlParserAPIs:jar
nekohtml:nekohtml:jar
xerces:xercesImpl:jar
commons-io:commons-io:jar
com.intellij.idea:jgoodies-forms:jar:7590
com.intellij.idea:javac2:jar:7590
asm:asm-commons:jar
```

asm:asm-tree:jar
 asm:asm:jar
 jdom:jdom:jar
 ant:ant:jar
 com.jgoodies:forms:jar
 com.intellij.idea:extensions:jar:7590
 net.sourceforge.cssparser:cssparser:jar;jdk15
 commons-httpclient:commons-httpclient:jar
 commons-logging:commons-logging:jar
 commons-codec:commons-codec:jar

Bamboo 1.0 Dependencies

| Bamboo 1.0 Dependencies | |
|--------------------------|--------------|
| atlassian-config | 0.1 |
| atlassian-spring | 0.1 |
| atlassian-core | 2006.07.14 |
| atlassian-johnson | 0.5.15 |
| atlassian-extras | 0.7.32 |
| atlassian-plugins | 2006-11-28 |
| atlassian-velocity | 0.3.18 |
| atlassian-bucket | 2006.06.15 |
| atlassian-profiling | 1.1.6 |
| atlassian-user | 2006-08-23 |
| bonnie | 2006-09-04 |
| atlassian-mail | 1.3.22-DEV-2 |
| servlet-api | 2.4 |
| jsp-api | 2.0 |
| jwebunit-htmlunit-plugin | 1.3-rc2 |
| junit | 3.8.1 |
| activation | 1.0.2 |
| ant | 1.6.5 |
| optional | 1.5.4 |
| backport-util-concurrent | 3.0 |
| build-tools | 1.1 |
| lucene-core | 1.9.1 |
| c3p0 | 0.9.1-pre12 |
| cglib-nodep | 2.1_3 |
| commons-beanutils | 1.7.0 |
| commons-collections | 3.1 |
| commons-configuration | 1.3 |
| commons-el | 1.0 |
| commons-httpclient | 3.0.1 |
| commons-io | 1.2 |
| commons-lang | 2.1 |
| commons-logging | 1.0.4 |

| | |
|------------------------|--------------------|
| dom4j | 1.4 |
| dwr | 1.1 |
| hsqldb | 1.8.0.7 |
| javacvs | atlassian-20070112 |
| emma | 2.0.5312 |
| emma_ant | 2.0.5312 |
| exml | 7.0 |
| httpunit | 1.5.4 |
| jasper-compiler | 5.0.28 |
| jasper-runtime | 5.0.28 |
| svnkit | 1.1.0 |
| jdom | 1.0 |
| org.mortbay.jetty.plus | 5.1.4 |
| org.mortbay.jetty | 5.1.4. |
| jmock | 1.0.1 |
| joda-time | 1.3 |
| js | 1.5R4-RC3 |
| jsch | 0.1.29 |
| junit | 3.8.1 |
| jwebunit | 1.2 |
| jta | 1.0.1 |
| log4j | 1.2.7 |
| mail | 1.3.2 |
| nekohtml | 0.7.7 |
| odmg | 3.0 |
| oscore | 2.2.4 |
| osuser | atl.user |
| pjl-comp-filter | 1.4 |
| polsUtils | 1.0.18 |
| propertyset | 1.3-21Nov03 |
| qdox | 1.2 |
| quartz | 1.5.2 |
| rome | 0.8 |
| seraph | 0.7.17 |
| sitemesh | 2.2.1 |
| smack | 2.2.1 |
| smackx | 2.2.1 |
| urlrewrite | 2.6.0 |
| velocity | 1.4 |
| webwork | 2.2.2 |
| xwork | 1.1.3 |
| freemarker | 2.3.6 |

| | |
|--------------------|-----------|
| ognl | 2.6.7 |
| rife-continuations | 0.0.2 |
| wrapper | 3.2.0 |
| xercesImpl | 2.0.2 |
| xml-apis | 1.0.b2 |
| xmlrpc | 1.2-b1 |
| xpp3_min | 1.1.3.4.O |
| xstream | 1.0.2 |
| i4jruntime | 3.2.4 |
| acegi-security | 1.0.1 |
| jfreechart | 1.0.2 |
| spring | 1.2.8 |
| hibernate | 2.1.8 |
| velocity-tools | 1.2 |
| axis | 1.4 |
| xfire-all | 1.2.1 |
| wstx-asl | 2.9.3 |
| commons-codec | 1.3 |
| jdom | 1.0 |

Bamboo 2.0 Dependencies

| | |
|---|-------|
| ant:ant | 1.6.5 |
| ant:ant-optional | 1.5.1 |
| aopalliance:aopalliance | 1.0 |
| asm:asm | 1.5.3 |
| asm:asm-util | 1.3.4 |
| aspectwerkz:aspectwerkz-core | 0.8.1 |
| backport-util-concurrent:backport-util-concurrent | 2.1 |
| cglib:cglib | 2.1 |
| com.atlassian.bonnie:atlassian-bonnie | 2.8 |
| com.atlassian.bucket:atlassian-bucket | 0.17 |
| com.atlassian.config:atlassian-config | 0.9 |
| com.atlassian.core:atlassian-core | 3.6 |
| com.atlassian.event:atlassian-event | 0.5 |
| com.atlassian.johnson:atlassian-johnson | 0.9 |
| com.atlassian.mail:atlassian-mail | 1.6 |
| com.atlassian.plugins:atlassian-plugins | 0.13 |
| com.atlassian.profiling:atlassian-profiling | 1.4 |
| com.atlassian.seraph:atlassian-seraph | 0.20 |
| com.atlassian.spring:atlassian-spring | 0.7 |

| | |
|---|-------------|
| com.atlassian.user:atlassian-user | 1.9 |
| com.atlassian.velocity:atlassian-velocity | 0.5 |
| com.intellij:annotations | 6.0.5 |
| commons-beanutils:commons-beanutils | 1.7.0 |
| commons-codec:commons-codec | 1.3 |
| commons-collections:commons-collections | 3.1 |
| commons-configuration:commons-configuration | 1.4 |
| commons-digester:commons-digester | 1.8 |
| commons-io:commons-io | 1.3.1 |
| commons-jxpath:commons-jxpath | 1.2 |
| commons-lang:commons-lang | 2.3 |
| commons-logging:commons-logging | 1.0.4 |
| commons-logging:commons-logging-api | 1.0.4 |
| dom4j:dom4j | 1.4 |
| hibernate:hibernate | 2.1.8 |
| hsqldb:hsqldb | 1.8.0.7 |
| isorelax:isorelax | 20020414 |
| javax.mail:mail | 1.3.2 |
| javax.servlet:servlet-api | 2.4 |
| javax.transaction:jta | 1.0.1B |
| jaxen:jaxen | 1.0-FCS |
| jdom:jdom | 1.0 |
| jivesoftware:smack | 2.2.1 |
| jivesoftware:smackx | 2.2.1 |
| jmock:jmock | 1.0.1 |
| joda-time:joda-time | 1.4 |
| junit:junit | 3.8.1 |
| log4j:log4j | 1.2.12 |
| m-extractors:tm-extractors | 0.4 |
| mockobjects:alt-jdk1.3 | 0.07 |
| mockobjects:mockobjects-alt-jdk1.3-j2ee1.3 | 0.09 |
| mockobjects:mockobjects-core | 0.09 |
| mockobjects:mockobjects-j1.3-j2ee1.3 | 0.07 |
| msv:msv | 20020414 |
| net.jcip:jcip-annotations | 1.0 |
| net.sf.ehcache:ehcache | 1.2.3 |
| net.sf.ldaptemplate:ldaptemplate | 1.0.1 |
| odmg:odmg | 3.0 |
| opensymphony:ognl | 2.6.11 |
| opensymphony:oscore | 2.2.4 |
| opensymphony:propertyset | 1.3-21Nov03 |
| opensymphony:sitemesh | 2.2.1 |

| | |
|---------------------------------------|--------------------|
| opensymphony:webwork | 2.2.5 |
| opensymphony:xwork | 1.2.3-20070717 |
| org.acegisecurity:acegi-security | 1.0.4 |
| org.apache.lucene:lucene-analyzers | 2.2.0 |
| org.apache.lucene:lucene-core | 2.2.0 |
| org.rifers:rife-continuations | 0.0.2 |
| org.springframework:spring-beans | 2.0.7 |
| org.springframework:spring-context | 2.0.7 |
| org.springframework:spring-core | 2.0.7 |
| org.springframework:spring-dao | 2.0.7 |
| org.springframework:spring-hibernate2 | 2.0.7 |
| org.springframework:spring-jdbc | 2.0.7 |
| org.springframework:spring-mock | 2.0.7 |
| org.springframework:spring-support | 2.0.7 |
| org.springframework:spring-web | 2.0.7 |
| oro:oro | 2.0.8 |
| oscache:oscache | DEV |
| osuser:osuser | atl.user |
| p6spy:p6spy | 1.3 |
| pdfbox:pdfbox | 0.7.0 |
| poi:poi | 2.0-final-20040126 |
| polsUtils:polsUtils | 1.0.18 |
| quartz:quartz | 1.5.2 |
| relaxngDatatype:relaxngDatatype | 20020414 |
| rome:rome | 0.8 |
| saxpath:saxpath | 1.0-FCS |
| velocity:velocity | 1.4 |
| velocity:velocity-dep | 1.4 |
| xerces:xercesImpl | 2.8.1 |
| xerces:xmlParserAPIs | 2.2.1 |
| xml-apis:xml-apis | 1.3.03 |
| xmlrpc:xmlrpc | 1.2-b1 |

Bamboo 2.1 Dependencies

| | |
|---|---------|
| ant:ant | 1.6.5 |
| ant:ant-optional | 1.5.4 |
| aopalliance:aopalliance | 1.0 |
| asm:asm | 1.5.3 |
| asm:asm-util | 1.3.4 |
| aspectwerkz:aspectwerkz-core | 0.8.1 |
| backport-util-concurrent:backport-util-concurrent | 2.1 |
| c3p0:c3p0 | 0.9.1.1 |

| | |
|---|---------------------|
| cglib:cglib | 2.1_3 |
| com.atlassian.bamboo.jira:bamboo-jira-soap-client | 1.0 |
| com.atlassian.bonnie:atlassian-bonnie | 2.8 |
| com.atlassian.bucket:atlassian-bucket | 0.17 |
| com.atlassian.config:atlassian-config | 0.9 |
| com.atlassian.core:atlassian-core | 3.6 |
| com.atlassian.crowd:crowd-integration-client | 1.3 |
| com.atlassian.event:atlassian-event | 0.5 |
| com.atlassian.extras:atlassian-extras | 1.18 |
| com.atlassian.johnson:atlassian-johnson | 0.9 |
| com.atlassian.mail:atlassian-mail | 1.9 |
| com.atlassian.plugins:atlassian-plugins | 0.13 |
| com.atlassian.profiling:atlassian-profiling | 1.4 |
| com.atlassian.seraph:atlassian-seraph | 0.20 |
| com.atlassian.spring:atlassian-spring | 0.7 |
| com.atlassian.user:atlassian-user | 1.9 |
| com.atlassian.velocity:atlassian-velocity | 0.5 |
| com.intellij:annotations | 6.0.5 |
| com.tek42.perforce:p4java | 0.7.5-atlasian-4.5 |
| com.thoughtworks.xstream:xstream | 1.3 |
| commons-beanutils:commons-beanutils | 1.7.0 |
| commons-codec:commons-codec | 1.3 |
| commons-collections:commons-collections | 3.1 |
| commons-configuration:commons-configuration | 1.4 |
| commons-digester:commons-digester | 1.8 |
| commons-el:commons-el | 1.0 |
| commons-httpclient:commons-httpclient | 3.0.1 |
| commons-io:commons-io | 1.4-backport-IO-168 |
| commons-jxpath:commons-jxpath | 1.2 |
| commons-lang:commons-lang | 2.3 |
| commons-logging:commons-logging | 1.0.4 |
| commons-logging:commons-logging-api | 1.0.4 |
| commons-pool:commons-pool | 1.4-RC2-atlassian-1 |
| dom4j:dom4j | 1.4 |
| hibernate:hibernate | 2.1.8 |
| hsqldb:hsqldb | 1.8.0.7 |
| isorelax:isorelax | 20020414 |
| javacvs:javacvs | atlassian-20080407 |
| javax.activation:activation | 1.0.2 |
| javax.jms:jms | 1.1 |
| javax.mail:mail | 1.3.2 |
| javax.servlet:servlet-api | 2.4 |

| | |
|--------------------------------------|----------------|
| javax.transaction:jta | 1.0.1B |
| jaxen:jaxen | 1.0-FCS |
| jdom:jdom | 1.0 |
| jfree:jcommon | 1.0.12 |
| jfree:jfreechart | 1.0.9 |
| jivesoftware:smack | 2.2.1 |
| jivesoftware:smackx | 2.2.1 |
| jmock:jmock | 1.0.1 |
| joda-time:joda-time | 1.4 |
| junit:junit | 3.8.1 |
| log4j:log4j | 1.2.12 |
| m-extractors:tm-extractors | 0.4 |
| mockobjects:alt-jdk1.3 | 0.07 |
| mockobjects:mockobjects-alt-jdk | 0.09 |
| mockobjects:mockobjects-core | 0.09 |
| mockobjects:mockobjects-j1.3-j2ee1.3 | 0.07 |
| msv:msv | 20020414 |
| net.jcip:jcip-annotations | 1.0 |
| net.sf.ehcache:ehcache | 1.2.3 |
| net.sf.ldaptemplate:ldaptemplate | 1.0.1 |
| odmg:odmg | 3.0 |
| opensymphony:ognl | 2.6.11 |
| opensymphony:oscore | 2.2.7 |
| opensymphony:propertyset | 1.3-21Nov03 |
| opensymphony:sitemesh | 2.2.1 |
| opensymphony:webwork | 2.2.5 |
| opensymphony:xwork | 1.2.3-20070717 |
| org.acegisecurity:acegi-security | 1.0.4 |
| org.apache.activemq:activemq-core | 4.1.2 |
| org.apache.activemq:activeio-core | 3.0.1 |
| org.apache.lucene:lucene-analyzers | 2.3.2 |
| org.apache.lucene:lucene-core | 2.3.2 |
| org.apache.xbean:xbean-spring | 3.2 |
| org.jencks:jencks | 2.1 |
| org.jencks:jencks-amqpool | 2.1 |
| org.logicblaze.lingo:lingo | 1.3 |
| org.mortbay.jetty:jetty | 6.1.5 |
| org.mortbay.jetty:jetty-plus | 6.1.5 |
| org.mortbay.jetty:jetty-management | 6.1.5 |
| org.rifers:rife-continuations | 0.0.2 |
| org.springframework:spring | 2.0.7 |
| org.springframework:spring-aop | 2.0.7 |

| | |
|---------------------------------------|--------------------|
| org.springframework:spring-beans | 2.0.7 |
| org.springframework:spring-context | 2.0.7 |
| org.springframework:spring-core | 2.0.7 |
| org.springframework:spring-dao | 2.0.7 |
| org.springframework:spring-hibernate2 | 2.0.7 |
| org.springframework:spring-jdbc | 2.0.7 |
| org.springframework:spring-jms | 2.0.7 |
| org.springframework:spring-mock | 2.0.7 |
| org.springframework:spring-support | 2.0.7 |
| org.springframework:spring-web | 2.0.7 |
| org.tmatesoft:svnkit | 1.1.8 |
| oro:oro | 2.0.8 |
| oscache:oscache | DEV |
| osuser:osuser | atl.user |
| p6spy:p6spy | 1.3 |
| pdfbox:pdfbox | 0.7.0 |
| poi:poi | 2.0-final-20040126 |
| polsUtils:polsUtils | 1.0.18 |
| quartz:quartz | 1.5.2 |
| relaxngDatatype:relaxngDatatype | 20020414 |
| rome:rome | 0.8 |
| saxpath:saxpath | 1.0-FCS |
| velocity:velocity | 1.4 |
| velocity:velocity-dep | 1.4 |
| xerces:xercesImpl | 2.8.1 |
| xerces:xmlParserAPIs | 2.2.1 |
| xml-apis:xml-apis | 1.3.03 |
| xmlrpc:xmlrpc | 1.2-b1 |

Bamboo 2.2 Dependencies

| | |
|----------------|-------|
| acegi-security | 1.0.4 |
| activation | 1.1.1 |
| activeio-core | 3.0.1 |
| activemq-core | 4.1.2 |
| activemq-ra | 4.1.2 |
| alt-jdk1.3 | 0.07 |
| annotations | 6.0.5 |
| ant | 1.7.1 |
| ant-launcher | 1.7.1 |
| aopalliance | 1.0 |
| asm | 1.5.3 |
| asm-util | 1.3.4 |

| | |
|--|--------|
| aspectjrt | 1.5.3 |
| aspectjweaver | 1.5.3 |
| aspectwerkz-core | 0.8.1 |
| atlassian-aws | 0.8 |
| atlassian-bamboo-agent-bootstrap | 2.2 |
| atlassian-bamboo-agent-classserver | 2.2 |
| atlassian-bamboo-agent-core | 2.2 |
| atlassian-bamboo-agent-elastic | 2.2 |
| atlassian-bamboo-agent-elastic-server | 2.2 |
| atlassian-bamboo-agent-installer | 2.2 |
| atlassian-bamboo-agent-local | 2.2 |
| atlassian-bamboo-agent-remote | 2.2 |
| atlassian-bamboo-api | 2.2 |
| atlassian-bamboo-charts | 2.2 |
| atlassian-bamboo-core | 2.2 |
| atlassian-bamboo-language | 2.2 |
| atlassian-bamboo-license | 2.2 |
| atlassian-bamboo-migration | 2.2 |
| atlassian-bamboo-persistence | 2.2 |
| atlassian-bamboo-plugin-autofavourite | 2.2 |
| atlassian-bamboo-plugin-buildnumberstamper | 2.2 |
| atlassian-bamboo-plugin-clover | 2.2 |
| atlassian-bamboo-plugin-labeller | 2.2 |
| atlassian-bamboo-plugin-vcsversion | 2.2 |
| atlassian-bamboo-upgrader | 2.2 |
| atlassian-bamboo-web | 2.2 |
| atlassian-bonnie | 2.8 |
| atlassian-bucket | 0.17 |
| atlassian-config | 0.9 |
| atlassian-core | 4.0 |
| atlassian-event | 0.5 |
| atlassian-extras | 1.20.2 |
| atlassian-johnson | 0.9 |
| atlassian-mail | 1.9 |
| atlassian-plugins | 0.13 |
| atlassian-profiling | 1.4 |
| atlassian-seraph | 0.20 |
| atlassian-spring | 0.7 |
| atlassian-tunnel | 0.3 |
| atlassian-user | 1.9 |
| atlassian-velocity | 0.5 |
| atlassian-xwork-12 | 1.3 |

| | |
|-----------------------------------|-----------------------|
| atlassian-xwork-core | 1.3 |
| axis | 1.4 |
| axis-jaxrpc | 1.4 |
| axis-saaj | 1.4 |
| axis-wsdl4j | 1.5.1 |
| backport-util-concurrent | 2.1 |
| bamboo-agent | 2.2 |
| bamboo-agent-bootstrap | 2.2-with-dependencies |
| bamboo-jira-soap-client | 1.0 |
| c3p0 | 0.9.1.1 |
| cglib | 2.1 |
| cglib-nodep | 2.1_3 |
| commons-attributes-api | 2.1 |
| commons-beanutils | 1.8.0 |
| commons-codec | 1.3 |
| commons-collections | 3.1 |
| commons-configuration | 1.4 |
| commons-dbcp | 1.2 |
| commons-digester | 1.8 |
| commons-discovery | 0.2 |
| commons-el | 1.0 |
| commons-httpclient | 3.0.1 |
| commons-io | 1.4-backport-IO-168 |
| commons-jxpath | 1.2 |
| commons-lang | 2.3 |
| commons-logging | 1.0.4 |
| commons-logging-api | 1.0.4 |
| commons-pool | 1.4-RC2-atlassian-1 |
| crowd-integration-client | 1.6 |
| dom4j | 1.4 |
| dwr | 1.1 |
| easymock | 2.3 |
| ehcache | 1.2.3 |
| exml | 7.0 |
| freemarker | 2.3.14 |
| geronimo-connector | 2.0.1 |
| geronimo-j2ee-connector_1.5_spec | 1.0 |
| geronimo-j2ee-management_1.0_spec | 1.0 |
| geronimo-jms_1.1_spec | 1.1.1 |
| geronimo-jta_1.0.1B_spec | 1.0.1 |
| geronimo-jta_1.1_spec | 1.0 |
| geronimo-spec-jta | 1.0.1B-rc4 |

| | |
|--------------------------------|----------|
| geronimo-transaction | 2.0.1 |
| hibernate | 2.1.8 |
| howl | 1.0.1-1 |
| hsqldb | 1.8.0.7 |
| isorelax | 20020414 |
| jasper-compiler | 5.0.28 |
| jasper-runtime | 5.0.28 |
| javacvs-atlassian | 20080407 |
| jaxb-api | 2.1 |
| jaxb-impl | 2.1.3 |
| jaxen | 1.0-FCS |
| jcip-annotations | 1.0 |
| jcommon | 1.0.12 |
| jdom | 1.0 |
| jencks | 2.1 |
| jencks-amqpool | 2.1 |
| jets3t | 0.6.1 |
| jettison | 1.0-RC2 |
| jetty | 6.1.5 |
| jetty-management | 6.1.5 |
| jetty-naming | 6.1.5 |
| jetty-plus | 6.1.5 |
| jetty-util | 6.1.5 |
| jfreechart | 1.0.9 |
| jmock | 1.0.1 |
| jms | 1.1 |
| jna | 3.0.3 |
| joda-time | 1.4 |
| jsch | 0.1.31 |
| jsp-api | 2.0 |
| jta | 1.0.1B |
| junit | 3.8.1 |
| ldaptemplate | 1.0.1 |
| lingo | 1.3 |
| log4j | 1.2.12 |
| lucene-analyzers | 2.3.2 |
| lucene-core | 2.3.2 |
| mail | 1.4.1 |
| mockobjects-alt-jdk1.3-j2ee1.3 | 0.09 |
| mockobjects-core | 0.09 |
| mockobjects-j1.3-j2ee1.3 | 0.07 |
| msv | 20020414 |

| | |
|--------------------|---------------------|
| nant_builder | 2.1.0 |
| odmg | 3.0 |
| ognl | 2.6.11 |
| oro | 2.0.8 |
| oscache-DEV | |
| oscore | 2.2.7 |
| osuser-atl.user | |
| p4java | 0.7.5-atlassian-5.5 |
| p6spy | 1.3 |
| pdfbox | 0.7.0 |
| pjl-comp-filter | 1.4 |
| poi | 2.0-final 20040126 |
| polsUtils | 1.0.18 |
| propertyset | 1.3 21Nov03 |
| qdox | 1.2 |
| quartz | 1.5.2 |
| relaxngDatatype | 20020414 |
| rife-continuations | 0.0.2 |
| rome | 0.8 |
| saxpath | 1.0-FCS |
| servlet-api | 2.4 |
| servlet-api | 2.5-6.1.5 |
| sitemesh | 2.2.1 |
| smack | 3.0.4 |
| smackx | 3.0.4 |
| spring | 2.0.7 |
| spring-aop | 2.0.7 |
| spring-beans | 2.0.7 |
| spring-context | 2.0.7 |
| spring-core | 2.0.7 |
| spring-dao | 2.0.7 |
| spring-hibernate2 | 2.0.7 |
| spring-jdbc | 2.0.7 |
| spring-jms | 2.0.7 |
| spring-mock | 2.0.7 |
| spring-support | 2.0.7 |
| spring-web | 2.0.7 |
| stax-api | 1.0-2 |
| stax-api | 1.0.1 |
| stax-utils | 20040917 |
| svnkit | 1.2.1.5297 |
| tm-extractors | 0.4 |

| | |
|---------------------|----------------|
| trilead-ssh2 | build213 |
| truezip | 6.6 |
| typica | 1.4.1 |
| urlrewrite | 2.6.0 |
| velocity | 1.4 |
| velocity-dep | 1.4 |
| velocity-tools | 1.2 |
| webwork | 2.2.5 |
| wrapper | 3.2.0 |
| wsdl4j | 1.6.1 |
| wstx-asl | 3.2.4 |
| xapool-without-pool | 1.4 |
| xbean-spring | 3.2 |
| xercesImpl | 2.8.1 |
| xfire-aegis | 1.2.6 |
| xfire-annotations | 1.2.6 |
| xfire-core | 1.2.6 |
| xfire-java5 | 1.2.6 |
| xfire-jsr181-api | 1.0-M1 |
| xml-apis | 1.3.03 |
| xmlParserAPIs | 2.2.1 |
| XmISchema | 1.1 |
| xpp3_min | 1.1.3.4.O |
| xstream | 1.3 |
| xwork | 1.2.3 20070717 |

Bamboo 2.3 Dependencies

| | |
|------------------------------------|-------|
| acegi-security | 1.0.4 |
| activation | 1.1.1 |
| activeio-core | 3.1.0 |
| activemq-core | 5.2.0 |
| activemq-ra | 5.2.0 |
| annotations | 6.0.5 |
| ant | 1.7.1 |
| ant-launcher | 1.7.1 |
| aopalliance | 1.0 |
| aspectjrt | 1.5.3 |
| aspectjweaver | 1.5.3 |
| atlassian-aws | 0.11 |
| atlassian-bamboo-agent-bootstrap | 2.3.1 |
| atlassian-bamboo-agent-classserver | 2.3.1 |
| atlassian-bamboo-agent-core | 2.3.1 |

| | |
|--|---------|
| atlassian-bamboo-agent-elastic | 2.3.1 |
| atlassian-bamboo-agent-elastic-server | 2.3.1 |
| atlassian-bamboo-agent-local | 2.3.1 |
| atlassian-bamboo-agent-remote | 2.3.1 |
| atlassian-bamboo-api | 2.3.1 |
| atlassian-bamboo-charts | 2.3.1 |
| atlassian-bamboo-core | 2.3.1 |
| atlassian-bamboo-language | 2.3.1 |
| atlassian-bamboo-license | 2.3.1 |
| atlassian-bamboo-migration | 2.3.1 |
| atlassian-bamboo-persistence | 2.3.1 |
| atlassian-bamboo-plugin-autofavourite | 2.3.1 |
| atlassian-bamboo-plugin-buildnumberstamper | 2.3.1 |
| atlassian-bamboo-plugin-clover | 2.3.1 |
| atlassian-bamboo-plugin-labeller | 2.3.1 |
| atlassian-bamboo-plugin-phpbuilder | 2.3.1 |
| atlassian-bamboo-plugin-vcsversion | 2.3.1 |
| atlassian-bamboo-upgrader | 2.3.1 |
| atlassian-bamboo-web | 2.3.1 |
| atlassian-bandana | 1.0 |
| atlassian-bonnie | 3.2 |
| atlassian-bucket | 0.17 |
| atlassian-cache-api | 1.0 |
| atlassian-config | 0.9 |
| atlassian-core | 4.4-rc1 |
| atlassian-event | 0.5 |
| atlassian-extras | 1.21 |
| atlassian-johnson | 0.10 |
| atlassian-mail | 1.9 |
| atlassian-plugins-core | 2.2.4 |
| atlassian-plugins-osgi | 2.2.4 |
| atlassian-plugins-osgi-events | 2.2.4 |
| atlassian-plugins-servlet | 2.2.4 |
| atlassian-plugins-spring | 2.2.4 |
| atlassian-plugins-webfragment | 2.2.4 |
| atlassian-plugins-webresource | 2.2.4 |
| atlassian-profiling | 1.8 |
| atlassian-seraph | 0.38.3 |
| atlassian-spring | 0.7 |
| atlassian-tunnel | 0.4 |
| atlassian-user | 1.9 |
| atlassian-velocity | 0.5 |

| | |
|-----------------------------------|---------------------|
| atlassian-xwork | 12-1.7 |
| atlassian-xwork-core | 1.7 |
| auiplugin | 1.1.1 |
| axis | 1.4 |
| axis-jaxrpc | 1.4 |
| axis-saaj | 1.4 |
| axis-wsdl4j | 1.5.1 |
| backport-util-concurrent | 2.1 |
| bamboo-jira-soap-client | 1.0 |
| bcprov-jdk14-138 | |
| bndlbin | 0.0.255 |
| c3p0-0.9.1.2 | |
| camel-core | 1.5.0 |
| cglib-nodep | 2.1_3 |
| commons-attributes-api | 2.1 |
| commons-beanutils | 1.8.0 |
| commons-codec | 1.3 |
| commons-collections | 3.1 |
| commons-configuration | 1.4 |
| commons-dbcpc | 1.2 |
| commons-digester | 1.8 |
| commons-discovery | 0.2 |
| commons-el | 1.0 |
| commons-fileupload | 1.2.1 |
| commons-httpclient | 3.0.1 |
| commons-io | 1.4-backport-IO-168 |
| commons-jxpath | 1.2 |
| commons-lang | 2.4 |
| commons-logging | 1.0.4 |
| commons-pool | 1.4-RC2-atlassia-1 |
| crowd-integration-client | 1.6.1 |
| dom4j | 1.4 |
| easymock | 2.3 |
| ehcache | 1.2.3 |
| exml | 7.0 |
| freemarker | 2.3.15 |
| geronimo-connector | 2.0.1 |
| geronimo-j2ee-connector_1.5_spec | 1.0 |
| geronimo-j2ee-management_1.0_spec | 1.0 |
| geronimo-jms_1.1_spec | 1.1.1 |
| geronimo-jta_1.0.1B_spec | 1.0.1 |
| geronimo-jta_1.1_spec | 1.0 |

| | |
|-------------------------------------|------------|
| geronimo-spec-jta | 1.0.1B-rc4 |
| geronimo-transaction | 2.0.1 |
| google-collections | 1.0-rc2 |
| hibernate | 2.1.8 |
| howl | 1.0.1-1 |
| hsqldb | 1.8.0.7 |
| isorelax | 20020414 |
| jasper-compiler | 5.0.28 |
| jasper-runtime | 5.0.28 |
| javacvs-atlassian | 20080407 |
| javassist | 3.6.ga |
| jaxb-api | 2.1 |
| jaxb-impl | 2.1.10 |
| jaxen | 1.0-FCS |
| jcip-annotations | 1.0 |
| jcl-over-slf4j | 1.5.6 |
| jcommon | 1.0.12 |
| jdom | 1.0 |
| jencks | 2.1 |
| jencks-amqpool | 2.1 |
| jettison | 1.0.1 |
| jetty | 6.1.15 |
| jetty-management | 6.1.15 |
| jetty-naming | 6.1.15 |
| jetty-plus | 6.1.15 |
| jetty-util | 6.1.15 |
| jfreechart | 1.0.9 |
| jmock | 1.0.1 |
| jms | 1.1 |
| jna | 3.0.9 |
| joda-time | 1.6 |
| jsch | 0.1.38 |
| jsp-api | 2.0 |
| jta | 1.0.1B |
| junit | 3.8.1 |
| ldaptemplate | 1.0.1 |
| lingo | 1.3 |
| log4j | 1.2.15 |
| lucene-analyzers | 2.3.2 |
| lucene-core | 2.3.2 |
| mail | 1.4.1 |
| mockobjects-alt-jdk1.3-j2ee1.3-0.09 | |

| | |
|----------------------------|---------------------|
| msv | 20020414 |
| nant_builder | 2.1.14 |
| odmg | 3.0 |
| ognl | 2.7.1 |
| org.apache.felix.framework | 1.2.1-atlassian-7 |
| org.apache.felix.main | 1.2.1-atlassian-7 |
| oro | 2.0.8 |
| oscore | 2.2.7 |
| osuser | 1.0-20060106 |
| p4java | 0.7.5-atlassian-5.6 |
| package-scanner | 0.7.7 |
| pdfbox | 0.7.0 |
| pjl-comp-filter | 1.4 |
| plexus-utils | 1.5.9 |
| poi | 3.0.2-FINAL |
| poi-scratchpad | 3.0.2-FINAL |
| polsUtils | 1.0.18 |
| propertyset | 1.3-21Nov03 |
| qdox | 1.2 |
| quartz | 1.6.5 |
| relaxngDatatype | 20020414 |
| rife-continuations | 0.0.2 |
| rome | 0.8 |
| sal-spi | 2.0.7 |
| sal-spring | 2.0.7 |
| saxpath | 1.0-FCS |
| servlet-api | 2.4 |
| servlet-api | 2.5-20081211 |
| sitemesh | 2.2.1 |
| slf4j-api | 1.5.6 |
| slf4j-log4j12-1.5.6 | |
| smack | 3.0.4 |
| smackx | 3.0.4 |
| spring | 2.0.7 |
| spring-aop | 2.0.7 |
| spring-beans | 2.0.7 |
| spring-context | 2.0.7 |
| spring-core | 2.0.7 |
| spring-dao | 2.0.7 |
| spring-hibernate2-2.0.7 | |
| spring-jdbc | 2.0.7 |
| spring-jms | 2.0.7 |

| | |
|------------------------------|------------|
| spring-mock | 2.0.7 |
| spring-support | 2.0.7 |
| spring-web | 2.0.7 |
| stax-api | 1.0-2 |
| stax-utils | 20040917 |
| svnkit | 1.3.0.5847 |
| tm-extractors | 0.4 |
| trilead-ssh2-build213-svnkit | 1.3-patch |
| truezip | 6.6 |
| typica | 1.4.1 |
| urlrewrite | 2.6.0 |
| velocity | 1.4 |
| velocity-dep | 1.4 |
| velocity-tools | 1.2 |
| webwork | 2.2.5 |
| wrapper | 3.2.0 |
| wsdl4j | 1.6.1 |
| wstx-asl | 3.2.4 |
| xapool-without-pool | 1.4 |
| xbean-spring | 3.2 |
| xercesImpl | 2.9.1 |
| xfire-aegis | 1.2.6 |
| xfire-annotations | 1.2.6 |
| xfire-core | 1.2.6 |
| xfire-java5-1.2.6 | |
| xfire-jsr181-api | 1.0-M1 |
| xml-apis | 1.3.03 |
| xmlParserAPIs | 2.2.1 |
| xpp3_min | 1.1.4c |
| xstream | 1.3.1 |
| xwork | 1.2.5-rc1 |

Clover 2.2 Dependencies

| Library | Version | License |
|--------------------------|---------|------------------------|
| antlr | 2.7.1 | BSD License |
| backport-util-concurrent | 3.0 | Public Domain |
| fastutil | 4.4.3 | LGPL |
| itext | 0.96 | Mozilla Public License |
| jfreechart | 1.0.8 | LGPL |
| jcommon | 1.0.12 | LGPL |
| log4j | 1.2.13 | Apache License V2 |
| overLIB | 4.2.1 | Artistic License |

Confluence 2.4 Dependencies

| Confluence 2.4 Dependencies | |
|-----------------------------|-----------------------|
| shared-methods | 1.0 |
| maven-utils | 1.0 |
| maven-model | 2.0.4 |
| atlassian-mail | 1.3.23 |
| atlassian-velocity | 0.1.9 |
| atlassian-core | 2007-02-21 |
| atlassian-config | 0.4 |
| atlassian-spring | 0.2 |
| atlassian-event | 0.2 |
| atlassian-bucket | 2007.01.23 |
| atlassian-profiling | 1.1.6 |
| atlassian-trackback | 0.8.5 |
| atlassian-extras | 0.7.32 |
| atlassian-johnson | 0.3 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-plugins | 2007-03-19 |
| atlassian-bandana | 0.2.0 |
| atlassian-gregory | 0.2 |
| atlassian-user | 2007-04-05 |
| atlassian-renderer | 2007-04-03_2_4_stable |
| bonnie | 2007-04-18 |
| joda-time | 0.98 |
| backport-util-concurrent | 3.0 |
| seraph | 0.7.23 |
| javamail | 1.3.2 |
| activation | 1.0.2 |
| log4j | 1.2.8 |
| commons-io | 1.2 |
| commons-beanutils | 1.6.1 |
| dwr | 1.1.1 |
| hibernate | 2.1.8-atlassian |
| cglib-full | 2.0.2 |
| dom4j | 1.4-full |
| odmg | 3.0 |
| c3p0 | 0.9.0.4 |
| ant | 1.5.2 |

| | |
|--------------------|---------------------------|
| jta | 1.0.1 |
| webwork | 2.1.5 |
| xwork | 1.0.3 |
| pell | multipart |
| ognl | 2.6.5 |
| oscore | 2.2.4 |
| oscache | 2.2 |
| osuser | atl.user |
| propertyset | 1.3-21Nov03 |
| velocity | 1.3.1 |
| servlet-api | 2.3 |
| spring | 1.1.5 |
| spring-aopalliance | 1.0 |
| ldaptemplate | 1.0.1 |
| sitemesh | 2.2.1-atlassian |
| velocity-tools | view-1.1 |
| radeox | 1.0b2-forked-22Apr2004 |
| jython | 2.1-forked |
| lucene-core | 1.9.1-atlassian |
| lucene-analyzers | 1.9.1 |
| commons-digester | 1.5 |
| tm-extractors | 0.4 |
| pdfbox | 0.7.2 |
| poi | 2.0-final-20040126 |
| fop | 0.20.5-atlassian-20060626 |
| jimi | 1.0 |
| ss_css2 | 0.9.3 |
| avalon-framework | 4.1.4 |
| batik-1.5-fop | 0.20-5 |
| saxon | 6.5.3-patched |
| jtidy | r8-20050104 |
| quartz | 1.5.2 |
| hsqldb | 1.8.0.1 |
| jwebunit | 1.2 |
| rhino | 1.5R4.1 |
| mockobjects-core | 0.09 |
| mockobjects | 0.07-j1.3-j2ee1.3 |
| alt-0.07 | j1.3-j2ee1.3 |
| alt | 0.07-jdk1.3 |
| exml | 7.1 |
| rome | 0.8 |
| jdom | 1.0 |

| | |
|---------------------|--------------------------|
| commons-jrcs | diff-0.1.7 |
| commons-httpclient | 3.0 |
| httpunit | 1.5.4 |
| nekohtml | 0.9.5 |
| xercesImpl | 2.2.1 |
| xmlParserAPIs | 2.2.1 |
| xmlrpc | 2.0+xmlrpc61 |
| glue | 5.0b2 |
| axis | 1.2.1 |
| axis-jaxrpc | 1.2.1 |
| axis-saaj | 1.2.1 |
| axis-wsdl4j | 1.2.1 |
| commons-discovery | 0.2 |
| commons-codec | 1.3 |
| oro | 2.0.7 |
| slide | 2.1 |
| xstream | 1.1.1 |
| xpp3 | 1.1.3.4d_b4_min |
| concurrent | 1.3.4-patched |
| bsf | 2.3.0 |
| jfreechart | 1.0.0 |
| xalan | 2.7.0 |
| selenium | 0.6-atlassian-2005.12.15 |
| jcaptcha | all-1.0-RC2.0.1 |
| tangosol | 3.2-b365-atlassian |
| coherence | 3.2-b365 |
| jtds | 1.2 |
| urlrewrite | 2.6.0 |
| commons-collections | 3.1 |
| ehcache | 1.2.3 |
| stax-api | 1.0.1 |
| wstx-asl | 2.9.3 |
| xfire-all | 1.2.1 |

Confluence 2.8 Dependencies

| | |
|-----------------------------|--------------|
| FontBox:FontBox | 0.1.0-dev |
| alt:alt | 0.07-jdk1.3 |
| alt:alt-0.07 | j1.3-j2ee1.3 |
| angosol-coherence:coherence | 3.3 |
| angosol-coherence:tangosol | 3.3 |

| | |
|---|------------------------|
| ant:ant | 1.5.2 |
| aopalliance:aopalliance | 1.0 |
| asm:asm | 1.5.3 |
| asm:asm-util | 1.3.4 |
| aspectj:aspectjweaver | 1.5.3 |
| aspectwerkz:aspectwerkz-core | 0.8.1 |
| avalon-framework:avalon-framework | 4.2.0 |
| axis:axis | 1.2.1 |
| axis:axis-jaxrpc | 1.2.1 |
| axis:axis-saaj | 1.2.1 |
| backport-util-concurrent:backport-util-concurrent | 3.0 |
| batik:batik-all | 1.6 |
| bouncycastle:bcprov-jdk14 | 136 |
| bsf:bsf | 2.3.0 |
| c3p0:c3p0 | 0.9.1.2 |
| cglib:cglib | 2.1 |
| com.servlets:cos | 09May2002 |
| com.sun:jai_codec | 1.1.3 |
| com.sun:jai_core | 1.1.3 |
| commons-beanutils:commons-beanutils | 1.6.1 |
| commons-codec:commons-codec | 1.3 |
| commons-collections:commons-collections | 3.2 |
| commons-digester:commons-digester | 1.5 |
| commons-discovery:commons-discovery | 0.2 |
| commons-fileupload:commons-fileupload | 1.2.1 |
| commons-httpclient:commons-httpclient | 3.0 |
| commons-io:commons-io | 1.2 |
| commons-jrcs:commons-jrcs | diff-0.1.7 |
| commons-lang:commons-lang | 2.1 |
| commons-logging:commons-logging | 1.0.4 |
| concurrent:concurrent | 1.3.4-patched |
| css2parser:ss_css2 | 0.9.3 |
| dom4j:dom4j | 1.4-full |
| dwr:dwr | 1.1.4 |
| exml:exml | 7.1 |
| fop:fop | trunk-534713-atlassian |
| glue:glue | 5.0b2 |
| hibernate:hibernate | 2.1.8-atlassian |
| hsqldb:hsqldb | 1.8.0.1 |
| httpunit:httpunit | 1.5.4 |
| javax.activation:activation | 1.0.2 |
| javax.mail:mail | 1.3.3 |

| | |
|--|-----------------|
| javax.servlet:servlet-api | 2.3 |
| javax.transaction:jta | 1.0.1B |
| jcaptcha:jcaptcha | all-1.0-RC2.0.1 |
| jdom:jdom | 1.0 |
| jfree:jcommon | 1.0.0 |
| jfree:jfreechart | 1.0.0 |
| joda-time:joda-time | 1.4 |
| jython:jython | 2.1-forked |
| log4j:log4j | 1.2.8 |
| m-extractors:tm-extractors | 0.4 |
| mockobjects:alt-jdk1.3 | 0.07 |
| mockobjects:mockobjects-alt-jdk1.3-j2ee1.3 | 0.09 |
| mockobjects:mockobjects-core | 0.09 |
| mockobjects:mockobjects-jdk1.3-j2ee1.2 | 0.09 |
| mx4j:mx4j | 3.0.1 |
| nekohtml:nekohtml | 0.9.5 |
| net.java.dev.urlrewrite:urlrewrite | 2.6.0 |
| net.sf.ehcache:ehcache | 1.2.3 |
| net.sf.ldaptemplate:ldaptemplate | 1.0.1 |
| net.sourceforge.jtds:jtds | 1.2 |
| odmg:odmg | 3.0 |
| ofbcore:ofbcore-share | 2.1.1 |
| ognl:ognl | 2.6.5 |
| opensymphony:oscore | 2.2.4 |
| opensymphony:propertyset | 1.3-21Nov03 |
| opensymphony:sitemesh | 2.3-atlassian |
| opensymphony:xwork | 1.0.3 |
| org.apache.lucene:lucene-analyzers | 2.2.0 |
| org.apache.lucene:lucene-core | 2.2.0 |
| org.codehaus.woodstox:wstx-asl | 3.2.0 |
| org.codehaus.xfire:xfire-all | 1.2.6 |
| org.hibernate:jtidy | r8-20060801 |
| org.slf4j:slf4j-api | 1.4.3 |
| org.slf4j:slf4j-log4j12 | 1.4.3 |
| org.springframework:spring-aop | 2.0.6 |
| org.springframework:spring-beans | 2.0.6 |
| org.springframework:spring-context | 2.0.6 |
| org.springframework:spring-core | 2.0.6 |
| org.springframework:spring-dao | 2.0.6 |
| org.springframework:spring-hibernate2 | 2.0.6 |
| org.springframework:spring-jdbc | 2.0.6 |
| org.springframework:spring-jmx | 2.0.6 |

| | |
|---|------------------------|
| org.springframework:spring-support | 2.0.6 |
| org.springframework:spring-web | 2.0.6 |
| oro:oro | 2.0.7 |
| oscache:oscache | 2.2 |
| osuser:osuser | atl.user |
| pdfbox:pdfbox | 0.7.2 |
| poi:poi | 2.0-final-20040126 |
| quartz:quartz | 1.5.2 |
| radeox:radeox | 1.0b2-forked-22Apr2004 |
| rome:rome | 0.8 |
| saxon:saxon | 6.5.3-patched |
| slide:slide | 2.1 |
| spring:spring-aopalliance | 1.0 |
| stax:stax-api | 1.0.1 |
| velocity-tools:velocity-tools | view-1.1 |
| velocity:velocity | 1.5 |
| wsdl4j:wsdl4j | 1.5.1 |
| xalan:xalan | 2.7.0 |
| xerces:xercesImpl | 2.2.1 |
| xerces:xmlParserAPIs | 2.2.1 |
| xml-apis:xml-apis | 1.0.b2 |
| xmlgraphics-commons:xmlgraphics-commons | 1.2svn |
| xmlrpc:xmlrpc | 2.0+xmlrpc61 |
| xpp3:xpp3 | 1.1.3.4d_b4_min |
| xstream:xstream | 1.1.1 |

Crowd 1.0.6 Dependencies

Crowd 1.0.6 Dependencies

| | |
|--------------------------|------------|
| hibernate | 3.2.2.ga |
| atlassian-extras | 0.7.32 |
| atlassian-bucket | 2006.06.15 |
| atlassian-core | 2007-02-28 |
| atlassian-user | 2007-04-05 |
| atlassian-config | 0.4 |
| seraph | 0.7.23 |
| atlassian-spring | 0.2 |
| commons-collections | 3.1 |
| commons-logging | 1.0.4 |
| log4j | 1.2.8 |
| mail | 1.3.2 |
| oscore | 2.2.4 |
| osuser | atl.user |
| propertyset | 1.3 |
| sitemesh | 2.2.1 |
| servlet-api | 2.3 |
| quartz | 1.5.1 |
| xfire-all | 1.2.1 |
| ldapbp | 1.0 |
| jiveforums-atlassian | 5.0.4 |
| webwork | 2.2.4 |
| xercesImpl | 2.8.1 |
| xalan | 2.7.0 |
| swizzle-confluence | 1.1 |
| jta | 1.0.1 |
| fisheye | 1.2.1 |
| joda-time | 1.4 |
| spring-ldap-hacked | 1.1.99 |
| ehcache | 1.2.3 |
| jmock | 1.1.0 |
| spring-mock | 1.2.8 |
| dbunit | 2.2 |
| backport-util-concurrent | 3.0 |
| apacheds-server-unit | 1.0.1 |
| slf4j-log4j12 | 1.2 |

Crowd 1.3.2 Dependencies

| | |
|-------------|-------|
| antlr:antlr | 2.7.6 |
|-------------|-------|

| | |
|---|--------------|
| asm:asm | 1.5.3 |
| asm:asm-attrs | 1.5.3 |
| backport-util-concurrent:backport-util-concurrent | 3.0 |
| bouncycastle:bcprov-jdk14 | 138 |
| c3p0:c3p0 | 0.9.1.2 |
| cenqua:fisheye | 1.2.1 |
| cglib:cglib | 2.1_3 |
| com.ibm.icu:icu4j | 3.4.4 |
| com.jivesoftware:jiveforums-atlassian | 5.5.1 |
| com.opensymphony:webwork | 2.2.6 |
| com.sun.jndi.ldap:ldapbp | 1.0 |
| commons-codec:commons-codec | 1.2 |
| commons-codec:commons-codec | 1.3 |
| commons-collections:commons-collections | 3.2 |
| commons-httpclient:commons-httpclient | 3.0.1 |
| commons-io:commons-io | 1.2 |
| commons-lang:commons-lang | 2.3 |
| commons-logging:commons-logging | 1.0.4 |
| dom4j:dom4j | 1.4 |
| freemarker:freemarker | 2.3.4 |
| isorelax:isorelax | 20020414 |
| javax.activation:activation | 1.1 |
| javax.mail:mail | 1.4 |
| javax.servlet:servlet-api | 2.3 |
| javax.transaction:jta | 1.0.1B |
| jaxen:jaxen | 1.0-FCS |
| jaxen:jaxen | 1.1-beta-9 |
| jdom:jdom | 1.0 |
| joda-time:joda-time | 1.4 |
| log4j:log4j | 1.2.8 |
| mockobjects:alt-jdk1.3 | 0.07 |
| msv:msv | 20020414 |
| net.java.dev.stax-utils:stax-utils | 20040917 |
| net.java.dev.urlrewrite:urlrewrite | 2.6.0 |
| net.sf.ehcache:ehcache | 1.2.3 |
| opensymphony:ognl | 2.6.11 |
| opensymphony:oscore | 2.2.4 |
| opensymphony:osuser | 1.0-20060106 |
| opensymphony:propertyset | 1.3-21Nov03 |
| opensymphony:sitemesh | 2.2.1 |
| opensymphony:xwork | 1.2.3 |
| org.acegisecurity:acegi-security | 1.0.5 |

| | |
|---------------------------------------|-----------------------------|
| org.apache.ws.commons:XmlSchema | 1.1 |
| org.apache.xbean:xbean-spring | 2.8 |
| org.codehaus.woodstox:wstx-asl | 3.2.4 |
| org.codehaus.xfire:xfire-aegis | 1.2.6 |
| org.codehaus.xfire:xfire-core | 1.2.6 |
| org.codehaus.xfire:xfire-spring | 1.2.6 |
| org.codehaus.xfire:xfire-xmlbeans | 1.2.6 |
| org.hibernate:hibernate | 3.2.5.ga |
| org.htmlparser:htmlparser | 1.6 |
| org.openid4java:openid4java | 0.9.3 |
| org.openxri:openxri-client | 1.0.1 |
| org.openxri:openxri-syntax | 1.0.1 |
| org.rifers:rife-continuations | 0.0.2 |
| org.springframework.ldap:spring-ldap | 1.2.1 |
| org.springframework:spring | 2.5.2 |
| org.springframework:spring-beans | 2.5.2 |
| org.springframework:spring-core | 2.5.2 |
| org.tuckey:urlrewrite | 2.5.2 |
| osuser:osuser | 1.0-dev-log4j-1.4jdk-7Dec05 |
| quartz:quartz | 1.5.1 |
| relaxngDatatype:relaxngDatatype | 20020414 |
| saxpath:sxpath | 1.0-FCS |
| spring-ldap-hacked:spring-ldap-hacked | 1.1.99 |
| stax:stax-api | 1.0.1 |
| wsdl4j:wsdl4j | 1.6.1 |
| xerces:xercesImpl | 2.6.2 |
| xerces:xercesImpl | 2.8.1 |
| xerces:xmlParserAPIs | 2.6.2 |
| xml-apis:xml-apis | 1.3.03 |
| xml-security:xmlsec | 1.3.0 |
| xmlbeans:xbean | 2.2.0 |

FishEye Crucible 1.5 Dependencies

| | |
|-----------------------------------|-------|
| ant-contrib:ant-contrib | 1.0b2 |
| ant:ant | 1.6.5 |
| antlr:antlr | 2.7.7 |
| asm:asm | 3.1 |
| avalon-framework:avalon-framework | 4.1.3 |
| c3p0:c3p0 | 0.9.0 |
| cglib:cglib-nodep | 2.1_3 |
| colt:colt | 1.2.0 |
| com.atlassian.core:atlassian-core | 3.8 |

| | |
|--|--------------|
| com.atlassian.crowd:crowd-integration-client | 1.3 |
| com.atlassian.extras:atlassian-extras | 1.17 |
| com.atlassian.plugins:atlassian-plugins | 0.10 |
| com.atlassian.profiling:atlassian-profiling | 1.3 |
| com.atlassian.seraph:atlassian-seraph | 0.36 |
| com.atlassian:cenqua-licensing | 1.6 |
| com.boilerbay.infinitydb:infinitydb | v1.0.53_3192 |
| com.octocaptcha:jcaptcha-all | 1.0-RC6 |
| com.sun.xml.bind:jaxb-xjc | 2.0 |
| com.sun.xml.messaging.saaj:saaj-impl | 1.3 |
| com.svnkit:svnkit | 1.1.6 |
| com.svnkit:svnkit-cli | 1.1.6 |
| com.svnkit:svnkit-javahl | 1.1.6 |
| com.svnkit:svnkit-jna | 1.1.6 |
| com.svnkit:svnkit-trilead | 1.1.6 |
| com.zentus.sqlitejdbc:sqlitejdbc | v037-nested |
| commons-beanutils:commons-beanutils | 1.7.0 |
| commons-codec:commons-codec | 1.3 |
| commons-collections:commons-collections | 3.1 |
| commons-dbcpc:commons-dbcpc | 1.2.1 |
| commons-fileupload:commons-fileupload | 1.2 |
| commons-httpclient:commons-httpclient | 3.1 |
| commons-io:commons-io | 1.3.1 |
| commons-lang:commons-lang | 2.3 |
| commons-logging:commons-logging | 1.1 |
| commons-pool:commons-pool | 1.2 |
| concurrent:concurrent | 1.3.4 |
| dom4j:dom4j | 1.6.1 |
| freemarker:freemarker | 2.3.4 |
| hsqldb:hsqldb | 1.8.0.9 |
| javax.activation:activation | 1.1 |
| javax.mail:mail | 1.4 |
| javax.transaction:jta | 1.0.1B |
| javax.ws.rs:jsr311-api | 0.5ea |
| javax.xml.bind:jaxb-api | 2.0 |
| javax.xml.soap:saaj-api | 1.3 |
| javax.xml.ws:jaxws-api | 2.0 |
| jaxen:jaxen | 1.1.1 |
| jboss:javassist | 3.0 |
| jdom:jdom | 1.0 |
| jfree:jcommon | 1.0.12 |
| jfree:jfreechart | 1.0.9 |

| | |
|---|-----------------|
| joda-time:joda-time | 1.4 |
| log4j:log4j | 1.2.11 |
| logkit:logkit | 1.0.1 |
| mockobjects:alt-jdk1.3 | 0.07 |
| net.java.dev.jersey:jersey | 0.5ea |
| net.java.dev.stax-utils:stax-utils | 20040917 |
| net.java.dev.urlrewrite:urlrewrite | 3.0.4 |
| net.sf.ehcache:ehcache | 1.2.3 |
| net.sf.proguard:proguard | 3.8 |
| ognl:ognl | 2.7.1 |
| opensymphony:oscore | 2.2.6 |
| opensymphony:osuser | 1.0-20060106 |
| opensymphony:propertyset | 1.3-21Nov03 |
| opensymphony:sitemesh | 2.3 |
| opensymphony:webwork | 2.1.7 |
| opensymphony:xwork | 1.0.5 |
| org.apache.cxf:cxf-api | 2.0.3-incubator |
| org.apache.cxf:cxf-common-schemas | 2.0.3-incubator |
| org.apache.cxf:cxf-common-utilities | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-bindings-soap | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-bindings-xml | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-core | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-databinding-jaxb | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-frontend-jaxws | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-frontend-simple | 2.0.3-incubator |
| org.apache.cxf:cxf-rt-transports-http | 2.0.3-incubator |
| org.apache.cxf:cxf-tools-common | 2.0.3-incubator |
| org.apache.geronimo.specs:geronimo-annotation_1.0_spec | 1.1 |
| org.apache.geronimo.specs:geronimo-ws-metadata_2.0_spec | 1.1.1 |
| org.apache.lucene:lucene-core | 2.2.0 |
| org.apache.neethi:neethi | 2.0.2 |
| org.apache.ws.commons.schema:XmlSchema | 1.3.2 |
| org.apache.xmlbeans:xmlbeans | 2.3.0 |
| org.bouncycastle:bcpprov-jdk15 | 138 |
| org.codehaus.woodstox:wstx-asl | 3.2.4 |
| org.codehaus.xfire:xfire-aegis | 1.2.6 |
| org.codehaus.xfire:xfire-core | 1.2.6 |
| org.eclipse.jdt:core | 3.1.1 |
| org.hibernate:hibernate | 3.2.6.ga |
| org.mortbay.jetty:jetty | 6.1.6 |
| org.mortbay.jetty:jetty-ajp | 6.1.6 |
| org.mortbay.jetty:jetty-util | 6.1.6 |

| | |
|-----------------------------------|-------|
| org.mortbay.jetty:jsp-2.1 | 6.1.6 |
| org.mortbay.jetty:jsp-api-2.1 | 6.1.6 |
| org.mortbay.jetty:servlet-api-2.5 | 6.1.6 |
| org.slf4j:slf4j-api | 1.4.3 |
| org.slf4j:slf4j-simple | 1.4.3 |
| org.springframework:spring | 2.5 |
| p6spy:p6spy | 1.3 |
| quartz:quartz | 1.5.1 |
| rove:trove | 1.0.2 |
| stax:stax-api | 1.0.1 |
| sun-jaxb:jaxb-impl | 2.0.5 |
| velocity:velocity | 1.4 |
| wsdl4j:wsdl4j | 1.6.1 |
| xalan:xalan | 2.7.0 |
| xerces:xercesImpl | 2.6.2 |
| xerces:xmlParserAPIs | 2.6.2 |
| xml-resolver:xml-resolver | 1.2 |
| xom:xom | 1.1 |

JIRA 3.8 Dependencies

| JIRA 3.8 Dependencies | |
|-------------------------------|--------------|
| atlassian-core | 2007-04-23 |
| atlassian-bandana | 0.1.13 |
| atlassian-ofbiz | 0.3.8 |
| atlassian-profiling | 1.1.4 |
| atlassian-jdk-utilities | 0.1 |
| atlassian-configurableobjects | 0.4.22 |
| atlassian-mail | 2007_03_23 |
| atlassian-scheduler | 2007-04-05 |
| atlassian-velocity | 0.3.19 |
| atlassian-johnson | 2006-11-03 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-plugins | 2006-11-10 |
| seraph | 0.7.21 |
| atlassian-extras | 0.7.29 |
| atlassian-tagutil | 0.1 |
| atlassian-renderer | 2007.01.17 |
| bonnie | 2007-01-11-1 |
| p6spy | 1.3 |
| osworkflow | 2.8.0 |

| | |
|-----------------------|-----------------------------|
| oscore | 2.2.6-dev-9Nov05 |
| osuser | 1.0-dev-log4j-1.4jdk-7Dec05 |
| propertyset | 1.3 |
| sitemesh | 2.3-dev-25Oct-05 |
| webwork | 16May06-jiratld |
| commons-digester | 1.4.1 |
| commons-beanutils | 1.6.1 |
| commons-logging | 1.0.4 |
| commons-collections | 3.1 |
| commons-lang | 2.1 |
| commons-configuration | 1.0 |
| log4j | 1.2.7 |
| lucene+core | 1.9.1 |
| lucene+analyzers | 1.9.1 |
| oro | 2.0.7 |
| jregex | 1.2_01 |
| velocity | 1.4 |
| dwr | 1.1-rc1 |
| alt | 0.07-jdk1.3 |
| alt+j2ee | 0.07-jdk1.3 |
| junit | 3.8.1 |
| mockobjects | DEV |
| easymock | 1.1 |
| cglib-full | 2.0.1 |
| easymockextension | 1.1 |
| mockobjects+j2ee | 0.07-jdk1.3 |
| mockobjects+jdk | 0.07-jdk1.3 |
| activation | 1.0.2 |
| javamail | 1.3.2 |
| servletapi | 2.3 |
| glue | 5.0b2 |
| bsf | 2.2 |
| bsh | 1.2b7 |
| ofbcore+share | 2.1.1 |
| ofbcore+entity | 2.1.1-atlassian-16Oct06 |
| ofbcore+service | 2.1.1 |
| ofbcore+extutil | 2.1.1 |
| csv | 20 |
| ofbcore+xerces | serializer |
| quartz | 1.4.5 |
| picocontainer | 1.0 |
| jzlib | 1.0.5 |

| | |
|---------------------------|--------------------------------------|
| jsch | 0.1.23 |
| jcaptcha | all-1.0-RC2.0.1-atlassian-2007-03-07 |
| javacvs | 2007-04-04-patched |
| statcvs | 20060222-patched |
| commons-dbcp | 1.1 |
| commons-pool | 1.1 |
| hsqldb | 1.8.0.5 |
| jndi | 1.2.1 |
| jta | 1.0.1 |
| ots-jts | 1.0 |
| jotm | 1.4.3 |
| jotm+jrmp_stubs | 1.4.3 |
| jotm+iiop_stubs | 1.4.3 |
| jotm+jonas_timer | 1.4.3 |
| jotm+objectweb-datasource | 1.4.3 |
| carol | 1.5.2 |
| carol+properties | |
| xapool | 1.3.1 |
| xml-apis | 1.0.b2 |
| saxon+noaelfred | 6.5.5 |
| commons-jelly | 1.0 |
| commons-jelly+tags-junit | 1.0 |
| commons-jelly+tags-util | 1.1.1 |
| commons-jelly+tags-email | 1.0 |
| commons-jelly+tags-log | 1.0 |
| commons-jelly+tags-http | 1.0 |
| commons-jelly+tags-soap | 1.0 |
| commons-jelly+tags-sql | 1.0 |
| commons-jexl | 1.0 |
| dom4j | 1.4 |
| atlassian-trackback | 2007-01-25 |
| backport-util-concurrent | 2.2 |
| commons-httpclient | 3.0 |
| commons-codec | 1.3 |
| xmlrpc | 2.0 |
| axis | 1.3 |
| xml-apis | 1.0.b2 |
| xpp3 | 1.1.3.4-RC8 |
| xstream | 1.1 |
| jfreechart | 1.0.4 |
| jfree | 1.0.8 |
| radeox | 1.0b2-forked-22Apr2004 |

| | |
|-------|-------------|
| jtidy | r8-20050104 |
| jdom | 1.0 |

JIRA 3.9 Dependencies

| JIRA 3.9 Dependencies | |
|-------------------------------|-----------------------------|
| | |
| atlassian-core | 2007-04-23 |
| atlassian-bandana | 0.1.13 |
| atlassian-ofbiz | 0.3.8 |
| atlassian-profiling | 1.1.4 |
| atlassian-jdk-utilities | 0.1 |
| atlassian-configurableobjects | 0.4.22 |
| atlassian-mail | 2007_03_23 |
| atlassian-scheduler | 2007-04-05 |
| atlassian-velocity | 0.3.19 |
| atlassian-johnson | 2006-11-03 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-plugins | 2006-11-10 |
| seraph | 0.7.21 |
| atlassian-extras | 0.7.29 |
| atlassian-tagutil | 0.1 |
| atlassian-renderer | 2007.01.17 |
| bonnie | 2007-01-11-1 |
| p6spy | 1.3 |
| osworkflow | 2.8.0 |
| oscore | 2.2.6-dev-9Nov05 |
| osuser | 1.0-dev-log4j-1.4jdk-7Dec05 |
| propertyset | 1.3 |
| sitemesh | 2.3-dev-25Oct-05 |
| webwork | 30Apr07-jiratld |
| commons-digester | 1.4.1 |
| commons-beanutils | 1.6.1 |
| commons-logging | 1.0.4 |
| commons-collections | 3.1 |
| commons-lang | 2.1 |
| commons-configuration | 1.0 |
| log4j | 1.2.7 |
| lucene+core | 1.9.1 |
| lucene+analyzers | 1.9.1 |
| oro | 2.0.7 |

| | |
|---------------------------|--------------------------------------|
| jregex | 1.2_01 |
| velocity | 1.4 |
| dwr | 1.1.4 |
| alt | 0.07-jdk1.3 |
| alt+j2ee | 0.07-jdk1.3 |
| junit | 3.8.1 |
| mockobjects | DEV |
| easymock | 1.1 |
| cglib-full | 2.0.1 |
| easymockextension | 1.1 |
| mockobjects+j2ee | 0.07-jdk1.3 |
| mockobjects+jdk | 0.07-jdk1.3 |
| activation | 1.0.2 |
| javamail | 1.3.2 |
| servletapi | 2.3 |
| glue | 5.0b2 |
| bsf | 2.2 |
| bsh | 1.2b7 |
| ofbcore+share | 2.1.1 |
| ofbcore+entity | 2.1.1-atlassian-16Oct06 |
| ofbcore+service | 2.1.1 |
| ofbcore+extutil | 2.1.1 |
| csv | 20 |
| ofbcore+xerces | serializer |
| quartz | 1.4.5 |
| picocontainer | 1.0 |
| jzlib | 1.0.5 |
| jsch | 0.1.23 |
| jcaptcha | all-1.0-RC2.0.1-atlassian-2007-03-07 |
| javacvs | 2007-04-04-patched |
| statcvs | 20060222-patched |
| commons-dbcp | 1.1 |
| commons-pool | 1.1 |
| hsqldb | 1.8.0.5 |
| jndi | 1.2.1 |
| jta | 1.0.1 |
| ots-jts | 1.0 |
| jotm | 1.4.3 |
| jotm+jrmp_stubs | 1.4.3 |
| jotm+iiop_stubs | 1.4.3 |
| jotm+jonas_timer | 1.4.3 |
| jotm+objectweb-datasource | 1.4.3 |

| | |
|--------------------------|------------------------|
| carol | 1.5.2 |
| carol+properties | |
| xapool | 1.3.1 |
| xml-apis | 1.0.b2 |
| saxon+noaelfred | 6.5.5 |
| commons-jelly | 1.0 |
| commons-jelly+tags-junit | 1.0 |
| commons-jelly+tags-util | 1.1.1 |
| commons-jelly+tags-email | 1.0 |
| commons-jelly+tags-log | 1.0 |
| commons-jelly+tags-http | 1.0 |
| commons-jelly+tags-soap | 1.0 |
| commons-jelly+tags-sql | 1.0 |
| commons-jexl | 1.0 |
| dom4j | 1.4 |
| atlassian-trackback | 2007-01-25 |
| backport-util-concurrent | 2.2 |
| commons-httpclient | 3.0 |
| commons-codec | 1.3 |
| xmlrpc | 2.0 |
| axis | 1.3 |
| xml-apis | 1.0.b2 |
| xpp3 | 1.1.3.4-RC8 |
| xstream | 1.1 |
| jfreechart | 1.0.4 |
| jfree | 1.0.8 |
| radeox | 1.0b2-forked-22Apr2004 |
| jtidy | r8-20050104 |
| jdom | 1.0 |

JIRA 3.10 Dependencies

| JIRA 3.10 Dependencies | |
|-------------------------------|------------|
| atlassian-core | 2007-04-23 |
| atlassian-bandana | 0.1.13 |
| atlassian-ofbiz | 0.3.8 |
| atlassian-profiling | 1.1.4 |
| atlassian-jdk-utilities | 0.1 |
| atlassian-configurableobjects | 0.4.22 |
| atlassian-mail | 2007_03_23 |

| | |
|-------------------------|-----------------------------|
| atlassian-scheduler | 2007-04-05 |
| atlassian-velocity | 0.3.19 |
| atlassian-johnson | 2007-06-01 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-plugins | 2006-11-10-2 |
| seraph | 0.7.21 |
| atlassian-extras | 0.7.29 |
| atlassian-tagutil | 0.1 |
| atlassian-renderer | 2007.01.17 |
| bonnie | 2007-01-11-1 |
| p6spy | 1.3 |
| osworkflow | 2.8.0 |
| oscore | 2.2.6-dev-9Nov05 |
| osuser | 1.0-dev-log4j-1.4jdk-7Dec05 |
| propertyset | 1.3 |
| sitemesh | 2.3-dev-25Oct-05 |
| webwork | 30Apr07-jiratld |
| commons-digester | 1.4.1 |
| commons-beanutils | 1.6.1 |
| commons-logging | 1.0.4 |
| commons-collections | 3.1 |
| commons-lang | 2.1 |
| commons-configuration | 1.0 |
| log4j | 1.2.7 |
| lucene+core | 1.9.1 |
| lucene+analyzers | 1.9.1 |
| oro | 2.0.7 |
| jregex | 1.2_01 |
| velocity | 1.4 |
| dwr | 1.1.4 |
| alt | 0.07-jdk1.3 |
| alt+j2ee | 0.07-jdk1.3 |
| junit | 3.8.1 |
| mockobjects | DEV |
| easymock | 1.1 |
| cglib-full | 2.0.1 |
| easymockextension | 1.1 |
| mockobjects+j2ee | 0.07-jdk1.3 |
| mockobjects+jdk | 0.07-jdk1.3 |
| activation | 1.0.2 |
| javamail | 1.3.2 |
| servletapi | 2.3 |

| | |
|---------------------------|--------------------------------------|
| glue | 5.0b2 |
| bsf | 2.2 |
| bsh | 1.2b7 |
| ofbcore+share | 2.1.1 |
| ofbcore+entity | 2.1.1-atlassian-2007-05-07 |
| ofbcore+service | 2.1.1 |
| ofbcore+extutil | 2.1.1 |
| csv | 20 |
| ofbcore+xerces | serializer |
| quartz | 1.4.5 |
| picocontainer | 1.0 |
| jzlib | 1.0.5 |
| jsch | 0.1.23 |
| jcaptcha | all-1.0-RC2.0.1-atlassian-2007-03-07 |
| javacvs | 2007-04-04-patched |
| statcvs | 20060222-patched |
| commons-dbcp | 1.1 |
| commons-pool | 1.1 |
| hsqldb | 1.8.0.5 |
| jndi | 1.2.1 |
| jta | 1.0.1 |
| ots-jts | 1.0 |
| jotm | 1.4.3 |
| jotm+jrmp_stubs | 1.4.3 |
| jotm+iiop_stubs | 1.4.3 |
| jotm+jonas_timer | 1.4.3 |
| jotm+objectweb-datasource | 1.4.3 |
| carol | 1.5.2 |
| carol+properties | |
| xapool | 1.3.1 |
| xml-apis | 1.0.b2 |
| saxon+noaelfred | 6.5.5 |
| commons-jelly | 1.0 |
| commons-jelly+tags-junit | 1.0 |
| commons-jelly+tags-util | 1.1.1 |
| commons-jelly+tags-email | 1.0 |
| commons-jelly+tags-log | 1.0 |
| commons-jelly+tags-http | 1.0 |
| commons-jelly+tags-soap | 1.0 |
| commons-jelly+tags-sql | 1.0 |
| commons-jexl | 1.0 |
| dom4j | 1.4 |

| | |
|--------------------------|------------------------|
| atlassian-trackback | 2007-01-25 |
| backport-util-concurrent | 2.2 |
| commons-httpclient | 3.0 |
| commons-codec | 1.3 |
| xmlrpc | 2.0 |
| axis | 1.3 |
| xml-apis | 1.0.b2 |
| xpp3 | 1.1.3.4-RC8 |
| xstream | 1.1 |
| jfreechart | 1.0.4 |
| jfree | 1.0.8 |
| radeox | 1.0b2-forked-22Apr2004 |
| jtidy | r8-20050104 |
| jdom | 1.0 |

JIRA 3.11 Dependencies

| JIRA 3.11 Dependencies | |
|-------------------------------|-----------------------------|
| atlassian-core | 2007-04-23 |
| atlassian-bandana | 0.1.13 |
| atlassian-ofbiz | 0.3.8 |
| atlassian-profiling | 1.1.4 |
| atlassian-jdk-utilities | 0.1 |
| atlassian-configurableobjects | 0.5 |
| atlassian-gzipfilter | 1.2 |
| atlassian-mail | 1.5 |
| atlassian-scheduler | 2007-04-05 |
| atlassian-velocity | 0.3.19 |
| atlassian-johnson | 2007-06-01 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-plugins | 2006-11-10-2 |
| seraph | 0.7.21 |
| atlassian-extras | 1.7 |
| atlassian-tagutil | 0.1 |
| atlassian-renderer | 2007.01.17 |
| bonnie | 2.3 |
| p6spy | 1.3 |
| osworkflow | 2.8.0 |
| oscore | 2.2.6-dev-9Nov05 |
| osuser | 1.0-dev-log4j-1.4jdk-7Dec05 |

| | |
|-----------------------|--------------------------------------|
| propertyset | 1.3 |
| sitemesh | 2.3-dev-25Oct-05 |
| webwork | 30Apr07-jiratld |
| commons-digester | 1.4.1 |
| commons-beanutils | 1.6.1 |
| commons-logging | 1.0.4 |
| commons-collections | 3.1 |
| commons-lang | 2.1 |
| commons-configuration | 1.0 |
| log4j | 1.2.7 |
| lucene+core | 2.2.0 |
| lucene+analyzers | 2.2.0 |
| oro | 2.0.7 |
| jregex | 1.2_01 |
| velocity | 1.4 |
| dwr | 1.1.4 |
| alt | 0.07-jdk1.3 |
| alt+j2ee | 0.07-jdk1.3 |
| junit | 3.8.1 |
| mockobjects | DEV |
| easymock | 1.2_Java1.3 |
| cglib-full | 2.0.1 |
| easymockextension | 1.2.1 |
| mockobjects+j2ee | 0.07-jdk1.3 |
| mockobjects+jdk | 0.07-jdk1.3 |
| activation | 1.0.2 |
| javamail | 1.3.2 |
| servletapi | 2.3 |
| glue | 5.0b2 |
| bsf | 2.2 |
| bsh | 1.2b7 |
| ofbcore+share | 2.1.1 |
| ofbcore+entity | 2.1.1-atlassian-2007-05-07 |
| ofbcore+service | 2.1.1 |
| ofbcore+extutil | 2.1.1 |
| csv | 20 |
| ofbcore+xerces | serializer |
| quartz | 1.4.5 |
| picocontainer | 1.0 |
| jzlib | 1.0.5 |
| jsch | 0.1.23 |
| jcaptcha | all-1.0-RC2.0.1-atlassian-2007-03-07 |

| | |
|---------------------------|------------------------|
| javacvs | 2007-04-04-patched |
| statcvs | 20060222-patched |
| commons-dbcp | 1.1 |
| commons-pool | 1.1 |
| hsqldb | 1.8.0.5 |
| jndi | 1.2.1 |
| jta | 1.0.1 |
| ots-jts | 1.0 |
| jotm | 1.4.3 |
| jotm+jrmp_stubs | 1.4.3 |
| jotm+iiop_stubs | 1.4.3 |
| jotm+jonas_timer | 1.4.3 |
| jotm+objectweb-datasource | 1.4.3 |
| carol | 1.5.2 |
| carol+properties | |
| xapool | 1.3.1 |
| xml-apis | 1.0.b2 |
| saxon+noaelfred | 6.5.5 |
| commons-jelly | 1.0 |
| commons-jelly+tags-junit | 1.0 |
| commons-jelly+tags-util | 1.1.1 |
| commons-jelly+tags-email | 1.0 |
| commons-jelly+tags-log | 1.0 |
| commons-jelly+tags-http | 1.0 |
| commons-jelly+tags-soap | 1.0 |
| commons-jelly+tags-sql | 1.0 |
| commons-jexl | 1.0 |
| dom4j | 1.4 |
| atlassian-trackback | 2007-01-25 |
| backport-util-concurrent | 2.2 |
| commons-httpclient | 3.0 |
| commons-codec | 1.3 |
| xmlrpc | 2.0 |
| axis | 1.3 |
| xml-apis | 1.0.b2 |
| xpp3 | 1.1.3.4-RC8 |
| xstream | 1.1 |
| jfreechart | 1.0.4 |
| jfree | 1.0.8 |
| radeox | 1.0b2-forked-22Apr2004 |
| jtidy | r8-20050104 |
| jdome | 1.0 |

JIRA 3.12 Dependencies

| | |
|-------------------------------|-----------------------------|
| atlassian-core | 2007-04-23 |
| atlassian-bandana | 0.1.13 |
| atlassian-ofbiz | 0.3.8 |
| atlassian-profiling | 1.1.4 |
| atlassian-jdk-utilities | 0.1 |
| atlassian-configurableobjects | 0.7 |
| atlassian-gzipfilter | 1.2 |
| atlassian-mail | 1.5 |
| atlassian-scheduler | 2007-04-05 |
| atlassian-velocity | 0.3.19 |
| atlassian-johnson | 2007-06-01 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-plugins | 2006-11-10-2 |
| atlassian-seraph | 0.33 |
| bouncycastle | bcprov-jdk14-138 |
| atlassian-extras | 1.12 |
| atlassian-tagutil | 0.1 |
| atlassian-renderer | 3.14 |
| atlassian-bonnie | 2.3 |
| p6spy | 1.3 |
| osworkflow | 2.8.0 |
| oscore | 2.2.6 |
| osuser | 1.0-dev-log4j-1.4jdk-7Dec05 |
| propertyset | 1.3 |
| sitemesh | 2.3-dev-25Oct-05 |
| webwork | 30Apr07-jiratld |
| pell-multipart-request | 1.31.0 |
| commons-digester | 1.4.1 |
| commons-beanutils | 1.6.1 |
| commons-logging | 1.0.4 |
| commons-collections | 3.1 |
| commons-lang | 2.1 |
| commons-configuration | 1.0 |
| log4j | 1.2.7 |
| lucene+core | 2.2.0 |
| lucene+analyzers | 2.2.0 |
| oro | 2.0.7 |

| | |
|---------------------------|--------------------------------------|
| jregex | 1.2_01 |
| velocity | 1.4 |
| dwr | 1.1.4 |
| alt | 0.07-jdk1.3 |
| alt+j2ee | 0.07-jdk1.3 |
| junit | 3.8.1 |
| mockobjects | DEV |
| easymock | 1.2_Java1.3 |
| cglib-full | 2.0.1 |
| easymockextension | 1.2.1 |
| mockobjects+j2ee | 0.07-jdk1.3 |
| mockobjects+jdk | 0.07-jdk1.3 |
| activation | 1.0.2 |
| javamail | 1.3.2 |
| servletapi | 2.3 |
| glue | 5.0b2 |
| bsf | 2.2 |
| bsh | 1.2b7 |
| ofbcore+share | 2.1.1 |
| ofbcore+entity | 2.1.1-atlassian-2007-05-07 |
| ofbcore+service | 2.1.1 |
| ofbcore+extutil | 2.1.1 |
| csv | 20 |
| ofbcore+xerces | serializer |
| quartz | 1.4.5 |
| picocontainer | 1.0 |
| jzlib | 1.0.5 |
| jsch | 0.1.23 |
| jcaptcha | all-1.0-RC2.0.1-atlassian-2007-03-07 |
| javacvs | 2007-04-04-patched |
| statcvs | 20060222-patched |
| commons-dbcp | 1.1 |
| commons-pool | 1.1 |
| hsqldb | 1.8.0.5 |
| jndi | 1.2.1 |
| jta | 1.0.1 |
| ots-jts | 1.0 |
| jotm | 1.4.3 |
| jotm+jrmp_stubs | 1.4.3 |
| jotm+iiop_stubs | 1.4.3 |
| jotm+jonas_timer | 1.4.3 |
| jotm+objectweb-datasource | 1.4.3 |

| | |
|--------------------------|------------------------|
| carol | 1.5.2 |
| carol+properties | |
| xapool | 1.3.1 |
| xml-apis | 1.0.b2 |
| saxon+noaelfred | 6.5.5 |
| commons-jelly | 1.0 |
| commons-jelly+tags-junit | 1.0 |
| commons-jelly+tags-util | 1.1.1 |
| commons-jelly+tags-email | 1.0 |
| commons-jelly+tags-log | 1.0 |
| commons-jelly+tags-http | 1.0 |
| commons-jelly+tags-soap | 1.0 |
| commons-jelly+tags-sql | 1.0 |
| commons-jexl | 1.0 |
| dom4j | 1.4 |
| atlassian-trackback | 2007-01-25 |
| backport-util-concurrent | 2.2 |
| commons-httpclient | 3.0 |
| commons-codec | 1.3 |
| xmlrpc | 2.0 |
| axis | 1.3 |
| axis-jaxrpc | 1.3 |
| axis-saaj | 1.3 |
| axis-wsdl4j | 1.5.1 |
| commons-discovery | 0.2 |
| xml-apis | 1.0.b2 |
| xerces-impl | 2.6.2 |
| xml-security | 1.1.0 |
| datafile | 1.3.3 |
| xpp3 | 1.1.3.4-RC8 |
| xstream | 1.1 |
| jfreechart | 1.0.4 |
| jfree | 1.0.8 |
| urlrewrite | 2.5.2 |
| radeox | 1.0b2-forked-22Apr2004 |
| jtidy | r8-20050104 |
| jdom | 1.0 |
| ehcache | 1.2.3 |
| stax-api | 1.0.1 |
| wstx-asl | 2.9.3 |
| xfire-all | 1.2.1 |

JIRA 3.13 Dependencies

| | |
|-------------------------------|-----------------------------|
| atlassian-core | 3.10.2 |
| atlassian-bandana | 0.1.13 |
| atlassian-profiling | 1.1.4 |
| atlassian-configurableobjects | 0.9 |
| atlassian-scheduler | 0.11 |
| atlassian-velocity | 0.8 |
| atlassian-johnson | 0.10 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-plugins | 0.23.2 |
| atlassian-seraph | 0.38.3 |
| bouncycastle | 138 |
| atlassian-extras | 1.21 |
| joda-time | 1.4 |
| atlassian-tagutil | 0.1 |
| atlassian-renderer | 3.14 |
| atlassian-bonnie | 2.3 |
| p6spy | 1.3 |
| osworkflow | 2.8.0 |
| oscore | 2.2.7 |
| osuser | 1.0-dev-log4j-1.4jdk-7Dec05 |
| propertyset | 1.3 |
| sitemesh | 2.3-dev-25Oct-05 |
| webwork | jira-1.4.4 |
| commons-digester | 1.4.1 |
| commons-beanutils | 1.6.1 |
| commons-logging | 1.0.4 |
| commons-collections | 3.2 |
| commons-lang | 2.4 |
| commons-configuration | 1.0 |
| commons-io | 1.4 |
| log4j | 1.2.7 |
| lucene+core | 2.2.0 |
| lucene+analyzers | 2.2.0 |
| oro | 2.0.7 |
| jregex | 1.2_01 |
| velocity | 1.4 |
| alt | 0.07-jdk1.3 |
| alt+j2ee | 0.07-jdk1.3 |
| junit | 3.8.1 |
| mockobjects | DEV |
| easymock | 1.2_Java1.3 |

| | |
|---------------------------|----------------------------|
| cglib-full | 2.0.1 |
| easymockextension | 1.2.1 |
| mockobjects+j2ee | 0.07-jdk1.3 |
| mockobjects+jdk | 0.07-jdk1.3 |
| activation | 1.0.2 |
| javamail | 1.3.3 |
| servletapi | 2.3 |
| glue | 5.0b2 |
| bsf | 2.2 |
| bsh | 1.2b7 |
| ofbcore+share | 2.1.1-atlassian-2008-04-23 |
| ofbcore+entity | 2.1.1-atlassian-2008-04-23 |
| ofbcore+service | 2.1.1 |
| ofbcore+extutil | 2.1.1 |
| csv | 20 |
| ofbcore+xerces | serializer |
| quartz | 1.4.5 |
| picocontainer | 1.0 |
| jzlib | 1.0.5 |
| jsch | 0.1.23 |
| javacvs | 2007-04-04-patched |
| statcvs | 20060222-patched |
| commons-dbcp | 1.1 |
| commons-pool | 1.1 |
| hsqldb | 1.8.0.5 |
| jndi | 1.2.1 |
| jta | 1.0.1 |
| ots-jts | 1.0 |
| jotm | 1.4.3 |
| jotm+jrmp_stubs | 1.4.3 |
| jotm+iiop_stubs | 1.4.3 |
| jotm+jonas_timer | 1.4.3 |
| jotm+objectweb-datasource | 1.4.3 |
| carol | 1.5.2 |
| carol+properties | |
| xapool | 1.3.1 |
| xml-apis | 1.0.b2 |
| saxon+noaelfred | 6.5.5 |
| commons-jelly | 1.0 |
| commons-jelly+tags-junit | 1.0 |
| commons-jelly+tags-util | 1.1.1 |
| commons-jelly+tags-email | 1.0 |

| | |
|---------------------------|------------------------|
| commons-jelly+tags-log | 1.0 |
| commons-jelly+tags-http | 1.0 |
| commons-jelly+tags-soap | 1.0 |
| commons-jelly+tags-sql | 1.0 |
| commons-jelly+tags-regexp | 1.0 |
| commons-jexl | 1.0 |
| dom4j | 1.4 |
| atlassian-trackback | 0.11 |
| backport-util-concurrent | 3.1 |
| commons-httpclient | 3.0.1 |
| commons-codec | 1.3 |
| xmlrpc | 2.0 |
| axis | 1.3 |
| xpp3 | 1.1.3.4-RC8 |
| xstream | 1.1.1 |
| jfreechart | 1.0.4 |
| jfree | 1.0.8 |
| radeox | 1.0b2-forked-22Apr2004 |
| jtidy | r8-20050104 |

JIRA 4.0 Dependencies

| | |
|-------------------------|----------|
| atlassian-core | 4.5.2 |
| sal-spi | 2.0.14.1 |
| atlassian-gadgets-api | 1.1.2 |
| atlassian-gadgets-spi | 1.1.2 |
| jsr305 | 1.3.2 |
| atlassian-bandana | 0.1.13 |
| atlassian-scheduler | 0.11 |
| atlassian-velocity | 0.8 |
| atlassian-cache-servlet | 0.5.4 |
| atlassian-seraph | 2.0.1 |
| atlassian-cache-api | 1.0 |
| atlassian-cache-memory | 1.0 |
| joda-time | 1.4 |
| atlassian-renderer | 3.14.1 |
| p6spy | 1.3 |
| osworkflow | 2.8.0 |
| oscore | 2.2.7 |
| propertyset | 1.3 |
| commons-digester | 1.4.1 |
| commons-beanutils | 1.6.1 |
| commons-logging | 1.0.4 |

| | |
|-----------------------|--------------------|
| commons-collections | 3.2 |
| commons-lang | 2.4 |
| commons-configuration | 1.0 |
| commons-io | 1.4 |
| log4j | 1.2.15 |
| oro | 2.0.7 |
| velocity | 1.4-atlassian-1 |
| velocity-tools | 1.3 |
| alt | 0.07-jdk1.3 |
| alt+j2ee | 0.07-jdk1.3 |
| junit | 3.8.1 |
| mockobjects | DEV |
| easymock | 2.4 |
| cglib-full | 2.0.1 |
| easymockextension | 2.4 |
| mockobjects+j2ee | 0.07-jdk1.3 |
| mockobjects+jdk | 0.07-jdk1.3 |
| activation | 1.1.1 |
| mail | 1.4.1 |
| servletapi | 2.3 |
| glue | 5.0b2 |
| bsf | 2.2 |
| bsh | 1.2b7 |
| ofbcore+jira+share | 2.1.5 |
| ofbcore+jira+entity | 2.1.5 |
| ofbcore+jira+service | 2.1.5 |
| ofbcore+jira+extutil | 2.1.5 |
| csv | 20 |
| quartz | 1.4.5 |
| picocontainer | 1.0 |
| jzlib | 1.0.5 |
| jsch | 0.1.23 |
| javacvs | 2007-04-04-patched |
| statcvs | 20060222-patched |
| commons-dbcp | 1.1 |
| commons-pool | 1.1 |
| hsqldb | 1.8.0.5 |
| jndi | 1.2.1 |
| jta | 1.0.1 |
| ots-jts | 1.0 |
| jotm | 1.4.3 |
| jotm+jrmp_stubs | 1.4.3 |

| | |
|---------------------------|------------------------|
| jotm+iiop_stubs | 1.4.3 |
| jotm+jonas_timer | 1.4.3 |
| jotm+objectweb-datasource | 1.4.3 |
| carol | 1.5.2 |
| carol+properties | |
| xapool | 1.3.1 |
| xml-apis | 1.3.04 |
| saxon+noaelfred | 6.5.5 |
| commons-jelly | 1.0 |
| commons-jelly+tags-junit | 1.0 |
| commons-jelly+tags-util | 1.1.1 |
| commons-jelly+tags-email | 1.0 |
| commons-jelly+tags-log | 1.0 |
| commons-jelly+tags-http | 1.0 |
| commons-jelly+tags-soap | 1.0 |
| commons-jelly+tags-sql | 1.0 |
| commons-jelly+tags-regexp | 1.0 |
| commons-jexl | 1.1 |
| dom4j | 1.4 |
| commons-httpclient | 3.0.1 |
| commons-codec | 1.3 |
| xmlrpc | 2.0 |
| axis | 1.3 |
| xpp3 | 1.1.3.4-RC8 |
| xstream | 1.1.1 |
| jfreechart | 1.0.4 |
| jfree | 1.0.8 |
| radeox | 1.0b2-forked-22Apr2004 |
| jtidy | r8-20050104 |

Obsolete Confluence Documentation

- [Confluence Plugin Repository](#)
- [Plugins - Recursive JarClassLoader](#)

Confluence Plugin Repository

Hello! Welcome to the [Confluence Plugin Repository](#) - the easy way to manage your uploadable plugins.

The repository can contain any plugin which can be uploaded through the standard Confluence Plugin Manager without the need for further modification. This documentation is not for those wishing to use the repository, please refer to the [Confluence Repository Client](#) - instead this is for plugin developers who want to utilise the repository.

Step 1: Place your plugin's JAR somewhere publicly accessible via the web

Your plugin needs to be downloaded by the [Confluence Repository Client](#), and as such it needs to be publicly accessible. Please note that the client can deal with basic authentication - which is useful for commercial plugins. In theory it should also deal with HTTPS, however that has not yet been tested.

Step 2: Checking out the existing Repository Metadata

You will need the following:

- A subversion account for svn.atlassian.com - you may need to speak to [Jonathan Nolen](#)
- A subversion client
- An editor capable of dealing with xml (preferably one that validate against a schema - if not you can use a [web validator instead](#))

Once you are setup:

```
svn checkout  
https://svn.atlassian.com/svn/public/contrib/repository/atlassian-plugin-repository/trunk/metadata.v2  
repository-metadata
```

You should end up with a `repository-metadata` directory containing all the existing metadata. The files you are interested in are `*-metadata.xml` and `repository.xml`.

Read through at least 3 existing `-metadata.xml` files to get a feel for the data structure.

Step 3: Adding your own

Create your file with the name `plugin.key-metadata.xml`, where `plugin.key` is your plugin key from your `atlassian-plugin.xml` file. This is just a convention, but it helps if everyone sticks to it.

Copy the metadata in from another plugin and modify the xml. Not everything is required, some node values have preset values, and correct element order is essential! See the schema [documentation](#) - or try to read the schema file listed in the metadata at the top; it's not as complex as it sounds!

Step 4: Validation

I cannot stress this enough: **validate your xml against the schema**. If your plugin's metadata does not successfully validate **you will break the repository when it generates!**

The easiest way is to use an editor (like IntelliJ IDEA) that validates inline, however you can always use the [online XML Schema validator](#). The output of the online validator isn't pretty - but it does the job.

Again, if you check in broken metadata, you will **kill the repository generation dead** - no further updates to the repository will be possible until the issue is fixed.

Step 5: Check-in

Once you are happy with your metadata, check it back in:

```
svn -m "adding plugin x.y.z to the repository" commit
```

Step 7: Repository Macros (options)

The repository macros pull from the generated repository data and are cached.

Step 8: Finished

Your plugin version is now in the repository - now you can get on with building the next plugin / version!

Confluence Plugin Repository Details

Confluence Plugin Repository

For what seems like an age, the ability to browse a repository of Confluence plugins to see what is available and then to simply click "install" and have all the work done for you has all been a pipe dream - until now! May I present the **Plugin Repository** which contains metadata on plugins that is eventually digested by the [Confluence Repository Client](#) and served up in a sexy interface for you to navigate through.

Validation through XML Schema

Just a quick word to say that we use XML Schema to validate the XML files. It's crucial that you make sure your XML validates **before** placing it in the repository. The easiest way is to use <http://www.w3.org/2001/03/webdata/xsv> - an online XML Schema validator. When using the online validator, if everything is returned ok - I suggest that you break it (add a `<bogus/>` element in somewhere) and validate again just to be sure that the validation is working fine.

Structure

The source of the repository's information is held in [CONFEXT:metadata files](#) inside [subversion](#). These files are then processed by the [CONFEXT:repository build generator](#) into files which the Confluence Repository Client can use.

The Metadata Files

The metadata files are fully validated using XML Schema, and also have a XSL rendering sheet which will be applied if you look at the xml files through a browser which can process XSLT instructions. There are two metadata xml files:

repository.xml

The `repository.xml` file contains the following information:

- The confluence versions supported by the repository.
- The categories available for plugins to associate with.
- The URL's to the `*-metadata.xml` files defined below.

***-metadata.xml**

Currently, the recommended practice is to create a `$pluginKey-metadata.xml` file for each plugin - as this is how the `maven metadata generator` will churn them out. However you can go against this putting multiple plugins in one file, but you could encounter problems when using the maven tools. You're best bet is to look at the other XML files in there and pick one close to your plugin - the easiest way to find out what you can or cannot do is to read the XSD Schema file (it's far more straight forward than what I thought it'd be like).

Here is the simplest structure you can have:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="plugin-metadata.xsl"?>
<plugins xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://svn.atlassian.com/svn/public/contrib/repository/confluence/plugins.xsd">
  <plugin name="Google Maps Plugin" key="com.atlassian.confluence.ext.gmaps">
    <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com/" />
    <homepage>http://confluence.atlassian.com/display/CONFEXT/Google+Maps+Plugin</homepage>
    <description>
      <![CDATA[
        <p>A macro for including <a href="http://maps.google.com/">Google Maps</a>
        of particular addresses within a page.</p>
      ]]>
    </description>
    <categories>
      <category key="repository.content-macros"/>
    </categories>
    <plugin-version version="0.1" build="1" date="2006-02-27" state="stable">
      <product-versions startBuild="320"></product-versions>
      <license><open-source license="BSD" /></license>
    <binary>http://svn.atlassian.com/svn/public/contrib/confluence/gmaps-plugin/jars/gmaps-confluence-plugin-0.1.jar</binary>
    <source>http://svn.atlassian.com/svn/public/contrib/confluence/gmaps-plugin/tags/0.1</source>
    <docs>http://confluence.atlassian.com/display/CONFEXT/Google+Maps+Plugin</docs>
    <contributors>
      <contributor name="Mike Cannon-Brookes">
        <role>Developer</role>
      </contributor>
    </contributors>
  </plugin>
</plugins>
```

Conventions

- **bold** are required, normal are optional
- @ denotes an attribute, *text* denotes the textual contents (including any CDATA), all nodes are elements
- Elements are order specific - so please make sure they are as described.

Above all else, **VALIDATE** your XML before placing it in the repository. While the repository generator will do its best to filter out duff information, you should **not** rely on it to validate your data for you.

Element: plugins

This must be the base element, and **must** contain the two attributes shown for Schema validation.

| Node | Value |
|------|-------|
|------|-------|

| | |
|--------|-------------------------------------|
| plugin | There must be at least one of these |
|--------|-------------------------------------|

Element: *plugins > plugin*

Metadata for a specific plugin.

| Node | Value |
|----------------|--|
| @name | Descriptive name - same as in atlassian-plugin.xml |
| @key | Plugin key - same as in atlassian-plugin.xml |
| vendor | Optionally one can occur |
| homepage | Optionally one can occur (the <i>text</i> being the homepage url) |
| jira | Optionally one can occur (the <i>text</i> being the JIRA url - usually http://jira.server.dom/browse/KEY) |
| description | Optionally one can occur (the <i>text</i> being the XHTML description of this plugin - usually as CDATA) |
| categories | One and only one must occur, containing at least one category element |
| plugin-version | At least only one must occur |

Element: *plugins > plugin > categories > category*

One for each of the categories this plugin is to appear in. A plugin must appear in at least one category, but cannot appear in the same category twice.

| Node | Value |
|------|---|
| @key | Category key - taken from the categories listed in repository.xml |

Element: *plugins > plugin > plugin-version*

This is where the repository's power is - in the metadata here that describes an individual plugin-version.

| Node | Value |
|------------------|--|
| @version | This is the textual version number - same as in atlassian-plugin.xml - such as "1.0" or "2.1.4". This version number is used to find and match the plugin that is installed so it's crucial that the version numbers in your atlassian-plugin.xml match those in this metadata. |
| @build | This is the integerial build number. It does not have to be sequential, but they do have to be unique and a bigger number means a newer release. |
| @date | This is the date of the release in YYYY-MM-DD format. The XML schema validates both the format and content of this field. |
| @state | This is optional but highly recommended to be filled in. The field (if entered) can contain one of the following: alpha, beta, rc, stable |
| product-versions | This element must occur once and only once. It is the information about which versions of confluence this plugin version works (or doesn't work) on. |
| requirements | This optional element can only occur once. It contains at least one require element and specifies the pre-installation requirements for this version to install. |
| license | This must also only occur, and occur once. It is the license type and must contain either open-source , freeware or commerical as described below. |
| binary | This element occurs once and only once. Its <i>text</i> is the direct URL to download this version of the plugin. This can be over http or https and can handle HTTP Authentication. HTTP Authentication is usually used by commerical plugins, who will issue their users a username/password after purchase. |
| source | Optionally one can occur (the <i>text</i> being the url to the source - usually a zip or the path to the trunk / tag in subversion) |
| javaDocs | Optionally one can occur (the <i>text</i> being the url to the JavaDocs) |
| docs | This optional element can occur once. Its <i>text</i> is the direct URL to the documentation for this plugin version. |
| contributors | This optional element defines who has contributed to this release and must contain at least 1 contributor element if specified. |
| release-notes | This optional element's <i>text</i> is the XHTML release notes for this version - usually as CDATA |

Element: *plugins > plugin > plugin-version > product-versions*

This defines the builds of confluence that this plugin is known to work and known not to. Grey areas are treated as unknown - and these are filterable in the [Confluence Repository Client](#). Once a plugin-version is known to work on a build of confluence, it's best practice to add a working product node to this element. Same for if it's known to break on a confluence build. Please do not assume that the plugin works on a confluence build without testing it first - leave it as unknown.

Here is a list of [product versions and build numbers](#).

| Node | Value |
|-------------|--|
| @startBuild | The first confluence build this plugin could possibly work on (for big API changes). Assumes 0 if omitted. |
| @endBuild | The last confluence build this plugin could possibly work on (for big API changes). Assumes infinity if omitted. |
| product | This optional element can occur many times |

Element: *plugins > plugin > plugin-version > product-versions > product*

Defines whether a specific confluence build definitely works or not with this plugin version. Only use this element if it has been tested and been proved to work or not.

| Node | Value |
|----------|---|
| @id | Set this always to "confluence" |
| @build | The confluence build number from repository.xml |
| @working | true / false for the working state |

Element: *plugins > plugin > plugin-version > requirements > require*

These requirements are not acted upon, they are only checked during installation of a plugin version and either fail or pass. Should it fail, the error message is shown - should it pass the plugin version is installed.

| Node | Value |
|---------|---|
| @plugin | The plugin key to make sure is installed |
| @class | The class which must be loadable in the confluence webapp |
| text | The error message to show if it fails |

Element: *plugins > plugin > plugin-version > license > open-source*

| Node | Value |
|------------|---|
| @donateURL | URL where donations can be taken |
| @license | Must be an accepted license type, which is one of the following: BSD, ASL, LGPL |

Element: *plugins > plugin > plugin-version > license > freeware*

| Node | Value |
|------------|----------------------------------|
| @donateURL | URL where donations can be taken |

Element: *plugins > plugin > plugin-version > license > commercial*

| Node | Value |
|--------------|--|
| @purchaseURL | URL where the product can be bought (or information on purchasing) |

Element: *plugins > plugin > plugin-version > contributors > contributor*

| Node | Value |
|--------|--|
| @name | Contributor's name |
| @email | Contributor's email address |
| @url | Contributor's website |
| role | Optionally an unlimited number of role elements can be added, each role's <i>text</i> being the role they played |

The Repository Build Generator

| | |
|----------------------|---|
| Generator Output | http://files.adaptavist.com/repository/ |
| Generation Frequency | Hourly |

This is a Java app which acts as a buffer between the CONFEXT:metadata files and the confluence build xml files that the Confluence Repository Client uses for its data. This means that should broken data be put into the CONFEXT:metadata files, the output will remain as resilient as possible (of course - this doesn't account for incorrect data).

The generator runs the following procedure:

1. Read in the `repository.xml` file and validating using the schema.
2. Read in each plugin url specified in the `repository.xml` file and validating using the schema.
3. Pulls out each plugin from the plugin url xml files and places them in a list (making sure the plugin key is unique).
4. Create a `repository-full.xml` file with all the information about all the plugins collected together.
5. For each confluence build specified in the `repository.xml`## Add the current build product element.
 - a. Add the latest build product element (useful for knowing if you're running an old version of confluence).
 - b. Copy in the categories.
 - c. Filter out any of the plugin versions known not to work on this confluence build.
 - d. Filter out any plugins with no versions notwithstanding the above filter.
 - e. Write out to `repository-b###.xml` where ### is the confluence build number.

The generator can be found in [subversion](#).

Plugins - Recursive JarClassLoader



This patch is now included as a standard part of **Confluence 2.2.1** and beyond.

Problem

Previously, the only way for a plugin to use libraries is to have those libraries installed into `WEB-INF/lib` - a pain at best, and impossible at worst if you want a version of the library different to that installed with Confluence itself.

Solution



This needs more extensive testing - and I was unable to get the `maven test` to run the `atlassian-plugins` unit tests. However so far it holds up to my preliminary testing - your experiences are welcome!



This no longer extracts dependencies! They are loaded dynamically from the plugin jars ... woot!

Binary Patch

If you would rather not patch the source code yourself, but would prefer just to install a binary patch - this is the one for you.

- Download the DISC:patched `atlassian-plugins` jar
- Delete the `confluence/WEB-INF/atlassian-plugins-0.3.12.jar` file
- Replace it with the patched version downloaded.
- Restart confluence
- DISC:Test the patch works!

Source Patching



Instead of following the instructions, you could simply apply DISC:the patch file.

Apply a patch to the `atlassian-plugins` project (the source is available with the Confluence source) to have its classloader do two things:

1. Load the Libraries

- Cycle through the files in the plugin JAR, looking for those in the `META-INF/lib` dir and ending with the `.jar` extension.
- Create a `InnerJar` inner class representing the Jar - passing the ZipEntry and the JarFile - and add it to the `innerLibraries` list.

2. File Lookup

- Look up the requested resource in the following order:
 1. Ask the cache of responses (using the simple `FileBytes` wrapper inner class for storage)
 2. Ask the plugin's JarClassLoader
 - a. Check the files in the plugin JAR first.
 - b. Check the `InnerJar` objects in `innerLibraries` (if the list is null, load the libraries first) - this will create a `JarInputStream` so that extraction isn't needed.
 3. Ask the parent's classloader (the webapp)

Preface

You will need to download the latest Confluence source (it's specifically the `plugins` subdirectory we are interested in) and have maven up and running. Our aim is to change a few files, and compile a new `atlassian-plugin` JAR which can be dropped into Confluence's `WEB-INF/lib` over the existing one.

This patch was created against **Atlassian Plugins 0.3.12** - other versions may need changes to these patches.

com.atlassian.plugin.loaders.classloading.JarClassLoading

Open up `plugins/src/java/com/atlassian/plugin/loaders/classloading/JarClassLoading.java` for editing.

Add the inner classes to the top of the class around line 17 to read:

```

private static class FileBytes {
    private byte[] data;

    private FileBytes(byte[] data) {
        this.data = data;
    }
}

private static class InnerJar {
    private JarFile jar;
    private ZipEntry entry;

    private InnerJar(JarFile jar, ZipEntry entry) {
        this.jar = jar;
        this.entry = entry;
    }

    private byte[] getFile(String path) {
        // Check for a null path
        if (path == null) return null;
        // Get the file
        try {
            // Get a new input stream
            JarInputStream jarStream = new JarInputStream( jar.getInputStream(entry) );
            // Find the entry for this path
            ZipEntry eachEntry;
            while ((eachEntry = jarStream.getNextEntry()) != null) {
                if (path.equals( eachEntry.getName() )) {
                    break;
                }
            }
            // Check I found something
            if (eachEntry == null) return null;
            // Grab the content
            ByteArrayOutputStream out = new ByteArrayOutputStream();
            try {
                byte[] buffer = new byte[2048];
                int read;
                while (jarStream.available() > 0) {
                    read = jarStream.read(buffer, 0, buffer.length );
                    if (read < 0) break;
                    out.write(buffer, 0, read);
                }
                // Return the contents
                return out.toByteArray();
            } finally {
                out.close();
            }
        } catch (IOException e) {
            log.error(e, e);
            return null;
        }
    }
}

```

Change the variable declarations now around line 72 to read:

```

private JarFile jar;
private File file;
private LinkedHashMap innerLibraries;
private HashMap cachedFiles = new HashMap();

```

Add the following below the `openJar` method around line 96:

```
public synchronized void loadInnerLibraries() {
    // Check I have a jar
    if (jar == null) return;
    innerLibraries = new LinkedHashMap();
    // Cycle through the entries
    Enumeration entries = jar.entries();
    while (entries.hasMoreElements()) {
        ZipEntry entry = (ZipEntry) entries.nextElement();
        String name = entry.getName();
        if (name.startsWith("META-INF/lib/") && name.endsWith(".jar")) {
            innerLibraries.add( new InnerJar(jar, entry) );
        }
    }
}
```

Finally in this file, change the `getByte` method to now around line 111 read as follows

```

public synchronized byte[] getFile(String path)
{
    // Check the cache
    FileBytes cacheLookup = (FileBytes) cachedFiles.get(path);
    if (cacheLookup != null) {
        return cacheLookup.data;
    }
    //
    InputStream in = null;
    try
    {
        openJar();
        ZipEntry entry = jar.getEntry(path);
        // Dan Hardiker :: Check the libraries if there are any
        if (entry == null) {
            if (innerLibraries == null) {
                loadInnerLibraries();
            }
            // Cycle through them trying to grab the file
            byte[] data;
            for (Iterator iter = innerLibraries.iterator(); iter.hasNext();) {
                InnerJar innerJar = (InnerJar) iter.next();
                data = innerJar.getFile(path);
                if (data != null) {
                    cachedFiles.put(path, new FileBytes(data));
                    return data;
                }
            }
            // Still nothing? - oh dear, what a pity, never mind
            cachedFiles.put(path, new FileBytes(null));
            return null;
        }
        // Dan Hardiker ::

        in = jar.getInputStream(entry);
        int size = (int) entry.getSize();
        byte[] data = readStream(in, size);

        // Cache it
        cachedFiles.put(path, new FileBytes(data));

        return data;
    }
    catch (IOException e)
    {
        return null;
    }
    finally
    {
        // ensure that we close the jar inputStream. Can not rely upon the readStream
        // method to do this.
        if (in != null)
        {
            try
            {
                in.close();
            }
            catch (IOException e)
            {
                // noop.
            }
        }
    }
}
}

```

com.atlassian.plugin.loaders.classloading.PluginsClassLoader

Open up `plugins/src/java/com/atlassian/plugin/loaders/classloading/PluginsClassLoader.java` for editing.

Change the `loadClass` method to read:

```

protected synchronized Class loadClass(String name, boolean resolve) throws ClassNotFoundException
{
    Class c = (Class) cache.get(name);
    if (c != null) return c;
    /*
    boolean handles = false;
    if (packages != null)
    {
        for (int i = 0; i < packages.length; i++)
        {
            if (name.startsWith(packages[i]))
            {
                handles = true;
                break;
            }
        }
    }
    if (!handles)
    {
        return super.loadClass(name, resolve);
    }
    */
    try
    {
        c = findClass(name);
    }
    catch (ClassNotFoundException ex)
    {
        return super.loadClass(name, resolve);
    }
    cache.put(name, c);
    return c;
}

```

Compile and Deploy

From a command line with `conf-source/plugins/` as your current working directory, run `maven jar` to build the JAR. Copy the resulting JAR file from the `target` subdirectory into Confluence's `WEB-INF/lib` (don't forget to delete any old versions). Restart Confluence and upload a plugin with bundled libraries - see [Bundling Libraries in Plugins using Maven](#) for an example of how to automate the bundling of libraries into your plugins.

Testing the Patch

You've got the new plugin system running now ... so what next? You will want to test it and you can do that one of two ways:

1. Create your own plugin with bundled libraries (see [Bundling Libraries in Plugins using Maven](#) for instructions on making compiling bundled libraries trivially easy)
2. Download the [DISC:test plugin](#) and install it into Confluence by uploading it or placing it into `conf.home/plugins`.

If you have installed the [DISC:test plugin](#) then you will have a `{load-class}` macro which simply loads a Class and outputs its name. If it fails to load then it gives you an error - this is a simple way of testing whether a plugin has had its bundled libraries loaded.

Try the following:

```

{load-class:com.adaptavist.test.LoadClass}
{load-class:org.apache.commons.httpclient.HttpClient}
{load-class:it.unimi.dsi.fastutil.Hash}

```

That should output the textual class names 3 times. If it works, the HttpClient will actually be v3.0 not the v2.0 which ships with Confluence v2.1.x (bundled libraries take precedence), and that Hash class doesn't exist anywhere in Confluence except in this plugin jar as a bundled library.

Congratulations - go forth and bundle. Happy plugging!

PS: If this patch makes it into Confluence v2.2 (as I hope it does) then you can use the [Plugin Repository's](#) metadata to limit who can download your new bundled library plugin to simplify your distribution.

Obsolete JIRA Documentation

- Building and deploying JIRA from source

- Configure IDEA Global Settings
- Configure the Database
- Create Idea Project Files
- Debug JIRA
- Deploy JIRA
- Developing with Eclipse2
- Disable Velocity Caching
- Edit build.properties
- Edit More Configuration
- External Resources
- Get the Source
- Getting Started
- Getting what you need – Prerequisites
- How to create a JIRA Project Tab
- How to set up a development environment for a JIRA Plugin
- Install an Appserver
- Install Atlassian-IDEA Maven Plugin
- Setting up your plugin project

Building and deploying JIRA from source

Getting what you need -- Prerequisites

Before you start developing for JIRA, you'll need a few things. Start by downloading and installing the software listed below.

Java Tools - JDK and Compiler

- [JDK 1.4.2_X](#)
- [JDK 1.5.0_X](#) (optional, for Tomcat 5.5. You can also run Tomcat 5.5 with JSK 1.4.2 by installing the compatibility libraries. See below.)
- [Jikes](#)

Database

While JIRA Standalone comes with the [HSQLDB](#) file-based database, it is sometimes useful to have a non-file-based database for development – it's faster and it's easier to find out what's going on inside. However, there are other cases where it is completely appropriate to stick with the HSQLDB as provided. If you want to use the standalone database, you can skip this step.

- [MySQL](#) (install instructions) or some other database of your choice

Build Tools

- [Maven 1.0.2](#) (install instructions) **NOTE: Maven 2.0 will NOT work.**
 1. Define `$MAVEN_HOME` in your shell config
 2. Add `$MAVEN_HOME/bin` to your path
 3. create maven repository (`$MAVEN_HOME/bin/install_repo.sh $HOME/.maven/repository`)

IDE

- [IDEA](#)



At Atlassian, we use JetBrains' [IDEA](#) for development. This documentation is going to focus initially on IDEA. Eventually, we hope to provide instructions for other popular IDEs. However, if you are using a different IDE for developing JIRA, feel free to add your instructions to these docs.

- [Eclipse](#)

You can use Eclipse to develop plugins for JIRA quite successfully. You can use the [Eclipse Maven Plugin](#) with the [JIRA Plugin Development Kit](#).

Install an Appserver

You can use the Atlassian-IDEA Maven plugin to configure deployment environments for Resin 2, Resin 3, Tomcat 5.5.x and Orion 2. You can run on any or all of these appservers. Unless you're testing compatibility, you probably only need one. Resin seems to be the most popular, but Tomcat is the basis for the standalone dist, so either of those seem like good choices.

Tomcat Installation

1. download & install Tomcat 5.5.x (see also [jira documentation](#))
2. download & install [extra jars](#) to `$TOMCAT_HOME/common/lib`.
3. If you want to run Tomcat 5.5 on a 1.4 JDK, download the [Compatibility package](#) and [install](#) it.

Resin 3 Installation

1. download & install Resin 3.0.x (see also [jira documentation](#))
2. download & install [extra jars](#) to \$RESIN3_HOME/lib.

Resin 2 Installation

1. download & install Resin 2.1.x (see also [jira documentation](#))
2. download & install [extra jars](#) to \$RESIN_HOME/lib.

Orion 2 Installation

1. download & install Orion 2.0.x (see also [jira documentation](#))

JIRA:Configure the database

If you're using a database other than HSQLDB, then you need to create a database and a user for JIRA. If you're using HSQLDB, skip this step.

1. make sure the database is running and will restart after reboot, if you prefer
2. create a new databases **jiradb**
3. create anew user/password **jirauser/jirauser**
4. grant **jirauser** full rights on **jiradb**
5. download & install the database driver into \$TOMCAT_HOME/common/lib, \$RESIN_HOME/lib, \$RESIN3_HOME/lib and/or \$ORION_HOME/???.
 - mysql connector
 - other databases

Install Atlassian-IDEA Maven Plugin

The Atlassian-IDEA Maven plugin will automatically create the IDEA project files (.ipr, .iml, .iws) necessary for developing and running JIRA from within IDEA.

1. download the latest [Atlassian-IDEA Maven plugin](#)
2. unpack the distribution in \$MY_WORKING_DIR
3. cd \$MY_WORKING_DIR/atlassian-idea-plugin-X.X.X
4. run **maven plugin:install**

Get the Source

1. download the source distribution from the Atlassian Website.
2. Unpack it in \$MY_WORKING_DIR
3. cd \$MY_WORKING_DIR/atlassian-jira-enterprise-X.X.X-source/jira/
4. run **maven jira:buildutils** to create the BuildUtils.java file from the BuildUtils.template in \$MY_WORKING_DIR/atlassian-jira-enterprise-X.X.X-source/jira/src/java/com/atlassian/jira/util/

Configure IDEA Global Settings

1. Launch IDEA
2. Go to 'Settings' > 'JDK & Global Libraries'
3. In the 'JDK' tab, make sure there is at least one JDK defined for 1.4.2_X (and 1.5.0_X if you wish to use Tomcat without the compatibility libs). If there is not, then create one by clicking the 'Add JSDK' icon and selecting the home directory of the java version you want. Name the JDK something useful and remember it.

Edit build.properties

- Edit \$MY_WORKING_DIR/atlassian-jira-enterprise-X.X.X-source/jira/build.properties for your environment.

Create Idea Project Files

1. cd \$MY_WORKING_DIR/atlassian-jira-enterprise-X.X.X-source/jira
2. maven atlassian-idea

That should create the IDEA project files. Now you can (finally) open the project 'atlassian-jira.ipr' in IDEA.

Edit More Configuration

1. In 'Settings' > "Compiler" uncheck "Report use of deprecated features."
2. Go to 'Settings' > 'Modules' and make sure that the tomcat_env is set to use your 1.5 JDK (unless you installed the Compatibility Package). All the other modules can use the default 1.4.2 JDK.
3. Edit \$MY_WORKING_DIR/src/atlassian/jira/src/webapp/WEB-INF/classes/entityengine.xml to correspond to your database type. Particularly, you'll need to change the 'field-type-name' in the datasource config. See the [JIRA documentation](#).
4. If you are going to deploy to Tomcat, you'll need to change the JNDI paths as mentioned [here](#). See the [JIRA documentation](#).

5. Disable Velocity Caching for faster development.

Deploy JIRA

In the 'Select Run/Debug Configuration' dropdown, you should see an entry corresponding to each of the appservers you configured. If any of them have red Xs on top of the icons, then open the configuration panel and try to fix any warnings/errors.

At this point, you should be able hit the 'Run' triangle, and watch as IDEA compiles JIRA and deploys it to your chosen appserver. You'll see the logs in the 'Run' panel. Once it's all up and running, hit [http://localhost:\\$atlassian.idea.application.port/\\$atlassian.idea.application.webapp.contextpath_](http://localhost:$atlassian.idea.application.port/$atlassian.idea.application.webapp.contextpath_) (e.g. <http://localhost:8080/jira>) and you should see the app's start page.

Remember, you have to manually edit your entityengine.xml to make sure it conforms to the app server you are currently deploying to. (At this point, that means Tomcat5.5.x vs. everything else.)

Debug JIRA

Once you're sure that JIRA can run in normal mode, you can try running with the debugger attached. Terminate the 'Run' session by clicking the red 'X'. Then Select the 'Debug' icon and IDEA will start the appserver and attach the debugger. You can now set breakpoints and step through the app.



OSX Users

If you see an error message about your appserver not listening on the debug port, you may need to put an entry in your hosts file for your local machine (whatever hostname IDEA thinks your machine is). See this [IDEA bug](#) for further info.

If you plan to develop a JIRA plugin, proceed to the next tutorial: [How to set up a development environment for a JIRA Plugin](#)

Configure IDEA Global Settings

Configure IDEA Global Settings

1. Launch IDEA
2. Go to 'Settings' > 'JDK & Global Libraries'
3. In the 'JDK' tab, make sure there is at least one JDK defined for 1.4.2_x (and 1.5.0_x if you wish to use Tomcat without the compatibility libs). If there is not, then create one by clicking the 'Add JSRDK' icon and selecting the home directory of the java version you want. Name the JDK something useful and remember it.

Configure the Database

JIRA: Configure the database

If you're using a database other than HSQLDB, then you need to create a database and a user for JIRA. If you're using HSQLDB, skip this step.

1. make sure the database is running and will restart after reboot, if you prefer
2. create a new databases **jiradb**
3. create anew user/password **jirauser/jirauser**
4. grant **jirauser** full rights on **jiradb**
5. download & install the database driver into \$TOMCAT_HOME/common/lib, \$RESIN_HOME/lib, \$RESIN3_HOME/lib and/or \$ORION_HOME/???.
 - mysql connector
 - other databases

Create Idea Project Files

Create Idea Project Files

1. cd \$MY_WORKING_DIR/atlassian-jira-enterprise-X.X.X-source/jira
2. maven atlassian-idea

That should create the IDEA project files. Now you can (finally) open the project 'atlassian-jira.ipr' in IDEA.

Debug JIRA

Debug JIRA

Once you're sure that JIRA can run in normal mode, you can try running with the debugger attached. Terminate the 'Run' session by clicking the red 'X'. Then Select the 'Debug' icon and IDEA will start the appserver and attach the debugger. You can now set breakpoints and step through the app.



OSX Users

If you see an error message about your appserver not listening on the debug port, you may need to put an entry in your hosts file for your local machine (whatever hostname IDEA thinks your machine is). See this [IDEA bug](#) for further info.

Deploy JIRA

Deploy JIRA

In the 'Select Run/Debug Configuration' dropdown, you should see an entry corresponding to each of the appservers you configured. If any of them have red Xs on top of the icons, then open the configuration panel and try to fix any warnings/errors.

At this point, you should be able hit the 'Run' triangle, and watch as IDEA compiles JIRA and deploys it to your chosen appserver. You'll see the logs in the 'Run' panel. Once it's all up and running, hit [http://localhost:\\$atlassian.idea.application.port/\\$atlassian.idea.application.webapp.contextpath_](http://localhost:$atlassian.idea.application.port/$atlassian.idea.application.webapp.contextpath_) (e.g. <http://localhost:8080/jira>) and you should see the app's start page.

Remember, you have to manually edit your entityengine.xml to make sure it conforms to the app server you are currently deploying to. (At this point, that means Tomcat5.5.x vs. everything else.)

Developing with Eclipse2



Also see the [Eclipse page](#) in Confluence Space.

For those of you using [Eclipse](#) rather than IDEA, you can still generate your project files using the [Eclipse plugin](#) bundled with Maven 1.0.x (2.0 will not work). This is not as advanced as the [atlassian-idea](#) plugin, but is still quite usable.

Below are steps you need to follow.

1. Ensure you have the Maven 1.0.x installed
2. Run `maven eclipse:add-maven-repo -Dmaven.eclipse.workspace=D:/dev/src/atlassian/`. This will setup your MAVEN_REPO property in `eclipse.maven.eclipse.workspace` has to be where your workspace is.
3. Run `maven eclipse` in the project you are working on (where the `project.xml` file is)
4. Run `maven jira:buildutils` to create the build-specific class info. Do this from the JIRA project root folder (where `project.xml` is)
5. Open up Eclipse and import or refresh the project

This will setup a project with all the dependencies and should compile.

Disable Velocity Caching

When you are developing for JIRA, it is often useful to disable the caching of the velocity templates so that you don't have to restart the server to see velocity changes. You can do this by editing the file `./src/webapp/WEB-INF/classes/velocity.properties`. Set the "class.resource.loader.cache" to false and set "velocimacro.library.autoreload" to true (uncomment the line if necessary).

Edit build.properties

Edit build.properties

- Edit `$MY_WORKING_DIR/atlassian-jira-enterprise-X.X.X-source/jira/build.properties` for your environment.

Edit More Configuration

Edit More Configuration

1. In 'Settings' > "Compiler" uncheck "Report use of deprecated features."
2. Go to 'Settings' > 'Modules' and make sure that the `tomcat_env` is set to use your 1.5 JDK (unless you installed the Compatibility Package). All the other modules can use the default 1.4.2 JDK.
3. Edit `$MY_WORKING_DIR/src/atlassian/jira/src/webapp/WEB-INF/classes/entityengine.xml` to correspond to your database type. Particularly, you'll need to change the 'field-type-name' in the datasource config. See the [JIRA documentation](#).
4. If you are going to deploy to Tomcat, you'll need to change the JNDI paths as mentioned [here](#). See the [JIRA documentation](#).

5. Disable Velocity Caching for faster development.

External Resources

External Resources

Links to other useful resources that are relevant to JIRA and JIRA development:

- [Maven Resources](#) — There are several [Maven](#) resources that are useful if you're planning to check out the [tutorials](#)
- [Workflow Function plugin](#) — Vincent Massol's [excellent tutorial](#) can be found [here](#)

Maven Resources

There are several [Maven](#) resources that are useful if you're planning to check out the [tutorials](#). We've gathered them here.

- [Maven home page](#)
- [Installing Maven](#)
- The [JAR](#) and [Deploy](#) plugins used for building your projects
- IDE plugins for [IDEA](#) and [Eclipse](#)

Workflow Function plugin

Vincent Massol's [excellent tutorial](#) can be found [here](#)

Get the Source

Get the Source

1. download the source distribution from the [Atlassian Website](#).
2. Unpack it in \$MY_WORKING_DIR
3. `cd $MY_WORKING_DIR/atlassian-jira-enterprise-X.X.X-source/jira/`
4. run `maven jira:buildutils` to create the BuildUtils.java file from the BuildUtils.template in
\$MY_WORKING_DIR/atlassian-jira-enterprise-X.X.X-source/jira/src/java/com/atlassian/jira/util/

Getting Started

1. Getting started with Atlassian plugins

This tutorial will show you how to set up your development environment, create an empty plugin template, and the basic principles of building, debugging, and testing a plugin. It will take you through the prerequisites and introduce you to some of the resources that Atlassian provides for plugin developers.

2. The JIRA Plugin Guide

These documents are specifically about plugins for JIRA. There's a page for each plugin module type that JIRA supports. You can combine multiple plugin modules inside a single plugin to accomplish complex tasks.

3. Understanding how JIRA works

These documents go some of the way to explaining what's really going on inside JIRA. Some of this information is useful to plugin developers. Other pieces are more relevant to the JIRA development team, but we've published them here in the spirit of open documentation.

Want to modify the JIRA source code?

You'll want to download a copy of the [source distribution](#) and then follow the instructions in [Building and deploying JIRA from source](#).

1. [Building and deploying JIRA from source](#)
2. [Developer Tutorials](#)

Help and documentation

- [The JIRA Documentation](#)
- [The JIRA API](#)
- [The JIRA Developer FAQ](#)
- [The JIRA Developer Forums](#)

Getting what you need -- Prerequisites

Getting what you need -- Prerequisites

Before you start developing for JIRA, you'll need a few things. Start by downloading and installing the software listed below.

Java Tools - JDK and Compiler

- [JDK 1.4.2_x](#)
- [JDK 1.5.0_x](#) (optional, for Tomcat 5.5. You can also run Tomcat 5.5 with JSK 1.4.2 by installing the compatibility libraries. See below.)
- [Jikes](#)

Database

While JIRA Standalone comes with the [HSQLDB](#) file-based database, it is sometimes useful to have a non-file-based database for development – it's faster and it's easier to find out what's going on inside. However, there are other cases where it is completely appropriate to stick with the HSQLDB as provided. If you want to use the standalone database, you can skip this step.

- [MySQL \(install instructions\)](#) or some other database of your choice

Build Tools

- Maven 1.0.2 (install instructions) **NOTE: Maven 2.0 will NOT work.**
 1. Define `$MAVEN_HOME` in your shell config
 2. Add `$MAVEN_HOME/bin` to your path
 3. create maven repository (`$MAVEN_HOME/bin/install_repo.sh $HOME/.maven/repository`)

IDE

- [IDEA](#)



At Atlassian, we use [JetBrains' IDEA](#) for development. This documentation is going to focus initially on IDEA. Eventually, we hope to provide instructions for other popular IDEs. However, if you are using a different IDE for developing JIRA, feel free to add your instructions to these docs.

- [Eclipse](#)

You can use Eclipse to develop plugins for JIRA quite successfully. You can use the [Eclipse Maven Plugin](#) with the [JIRA Plugin Development Kit](#).

How to create a JIRA Project Tab

Overview

Project Tab and Issue Tab plugins can be used to add tabs to the project and issue view. These tabs can contain almost any kind of information: lists of issues, static information, reports, statistics or charts.

With JIRA 3.0, the introduction of the plugin system aimed to provide a simple point of extensibility for custom features users may wish to add to JIRA. This tutorial will describe how to create a custom Project Tab within JIRA 3, using this plugin interface.

A Project Tab Plugin Primer

In order to make a custom Project Tab available within JIRA, it is necessary to create a Project Tab plugin. Like all plugins, the Project Tab plugin will consist of:

- Java classes encapsulating Project Tab logic
- Resource templates for display of the Project Tab
- Plugin descriptor to enable the portlet module in JIRA

all contained within a single JAR file.

Project Tab Logic

The Java classes include the necessary logic to retrieve the data used in configuring and displaying the Project Tab. The module class must implement `com.atlassian.jira.plugin.projectpanel.ProjectTabPanel` and will usually extend `com.atlassian.jira.plugin.projectpanel.impl.GenericProjectTabPanel`. The `GenericProjectTabPanel` interface provides methods to initialise the Project Tab through the parameters defined in the descriptor (name, key, order etc.) and retrieve the Project Tab template location. It is also possible to extend existing Project Tabs to add further custom functionality.

Resource Templates

The second component consists of templates used to render the Projects - Velocity templates can be used here.

Other Resources

It is possible to include i18n property files also - so as to allow other users to easily translate the strings used in the Project Tab for different languages. The following Project Tab examples are fully internationalized and include default property files.

Plugin Descriptor

The Project Tab module descriptor is the only mandatory part of the plugin. It must be called `atlassian-plugin.xml` and be located in the root of the JAR file.

Here is a sample portlet module descriptor element:

```
<!--  
The class defined should implement  
com.atlassian.jira.plugin.projectpanel.ProjectTabPanel  
and it may be useful to use the functionality provided by  
com.atlassian.jira.plugin.projectpanel.impl.GenericProjectTabPanel.  
-->
```

In this sample, the portlet logic is encapsulated in the `VersionsProjectTabPanel` Java class. The view template location is specified in the `templates/plugins/jira/projectpanels` directory.

JIRA Development Kit

You can choose to develop your plugins however you wish. However, we recommend using [Maven 1.0](#) and the [JIRA Plugin Developement Kit](#).

Maven is an ant-like build tool that downloads any specified project dependencies automatically (just one of the many features).

The [JIRA Plugin Developement Kit](#) consists of a number of examples (including the ones discussed here) to help developers extend JIRA through the plugin interface as easily as possible.

Once the [JIRA Plugin Developement Kit](#) has been setup, you need only run the command:

```
maven jar
```

to build the desired plugin. Using Maven is **not** a requirement - you can use ant or any other build tool, however, it *will* make your life a lot easier.

Example



First, complete the instructions in [How to Build and Atlassian Plugin] to set up your development environment.

Step 1: Hello World

Our first step is to get the Tab to appear in JIRA.

To do this, we need to:

1. Fill out the `/src/etc/atlassian-plugin.xml` file.

```

<atlassian-plugin key="com.atlassian.jira.plugin.examples" name="Examples">
    <plugin-info>
        <description>Example Project Tab</description>
        <version>1.0</version>
        <application-version min="3.5" max="3.5"/>
        <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com"/>
    </plugin-info>

    <project-tabpanel key="example-project-tab"
                      name="Example Project Tab"
                      class="com.atlassian.jira.plugin.projectpanel.impl.GenericProjectTabPanel">
        <description key="example.project-tab.description"/>
        <label key="example.project-tab.label" />
        <!-- this is a number defining the order of all project tabs.
        The system panels are 10, 20, 30 and 40. -->
        <order>50</order>
        <!-- this template produces the HTML for the panel -->
        <resource type="velocity" name="view" location="templates/example-project-tab.vm" />
        <!-- this properties files contains i18n keys for the panel -->
        <resource type="i18n" name="i18n" location="com.atlassian.jira.plugin.example-project-tab">
    />
    </project-tabpanel>
</atlassian-plugin>

```

2. Create a `/src/etc/com/atlassian/jira/plugin/example-project-tab.properties` file that will contain the i18n-ready text strings for your plugin.

```

example.project-tab.description = A hello world project tab
example.project-tab.name = Example Project Tab
example.project-tab.label = Example

```

3. Create `/src/etc/templates/example-project-tab.vm` that will print "Hello World."

```
<h1>Hello World</h1>
```

Now if you run `maven jar` from your working directory and then copy the plugin to WEB-INF/lib you should be able to restart your appserver and see the new project tab on the project screen. You should also be able to go to 'Administration > Plugin > Examples' and see your new plugin installed and enabled.

Step 2: Make your Project Tab do something interesting

Now that you've gotten the basic project tab to display, this is the point where you could take off on your own. The example I'm going to code below may have no similarity to what you want your project tab to do. However, if you want your project tab to display a list of recently updated issues (or something reasonably similar), then carry on!

1. Add a java class that retrieves a list of issues.

We'll want our new class to extend `GenericProjectTabPanel`. Basically, what's going to happen is that we will retrieve a list of recently updated issues and then push them into the velocity context to use in the page. Here's the class:

```

package com.atlassian.jira.plugin;

import com.atlassian.jira.issue.IssueFieldConstants;
import com.atlassian.jira.issue.IssueManager;
import com.atlassian.jira.issue.customfields.converters.DateConverter;
import com.atlassian.jira.issue.fields.NavigableField;
import com.atlassian.jira.issue.index.DocumentConstants;
import com.atlassian.jira.issue.search.*;
import com.atlassian.jira.issue.search.parameters.lucene.*;
import com.atlassian.jira.plugin.projectpanel.impl.GenericProjectTabPanel;
import com.atlassian.jira.util.map.EasyMap;
import com.atlassian.jira.web.action.browser.Browser;
import com.atlassian.jira.web.bean.PagerFilter;
import org.apache.log4j.Category;

import java.util.Collection;
import java.util.Date;

```

```

import java.util.List;
import java.util.Map;

public class RecentlyUpdatedIssuesProjectTabPanel extends GenericProjectTabPanel {
    protected final Category log = Category.getInstance(this.getClass());

    private final SearchProvider searchProvider;

    private static final int MAX_ISSUES_TO_DISPLAY = 50;
    private static final PagerFilter PAGER_FILTER = new PagerFilter(MAX_ISSUES_TO_DISPLAY);
    protected final DateConverter dateConverter;

    final long ONE_WEEK = 1000 * 60 * 60 * 24 * 7;

    public RecentlyUpdatedIssuesProjectTabPanel(SearchProvider searchProvider, DateConverter
dateConverter) {
        this.searchProvider = searchProvider;
        this.dateConverter = dateConverter;
    }

    public String getHtml(Browser browser) {
        final Map startingParams = EasyMap.build("action", browser);

        // Get the list of recently updated issues and add it to the velocity context
        startingParams.put("recentlyUpdatedIssues", getRecentlyUpdatedIssues(browser));

        return descriptor.getHtml("view", startingParams);
    }

    public Collection getRecentlyUpdatedIssues(Browser browser) {
        List issues = null;
        try {
            // Create a Search Request to get the list of issues
            SearchRequest searchRequest = new SearchRequest(browser.getRemoteUser());

            // Only get issues for the current project
            searchRequest.addParameter(new
ProjectParameter(browser.getSelectedProject().getLong("id")));

            // Only get issues updated in the last week
            Date startDate = new Date(System.currentTimeMillis() - ONE_WEEK);
            searchRequest.addParameter(new
AbsoluteDateRangeParameter(DocumentConstants.ISSUE_UPDATED, startDate, null));

            // Sort the issues by issue updated time
            searchRequest.addSearchSort(new SearchSort(NavigableField.ORDER_DESCENDING,
IssueFieldConstants.UPDATED));

            // Do the search, return the list of issues.
            issues = searchProvider.search(searchRequest, browser.getRemoteUser(),
PAGER_FILTER).getIssues();
        }
        catch (Exception e) {
            log.error("Error finding recently updated issues: " + e, e);
        }
    }

    return issues;
}

```

```
    }  
}
```



For more information on searching, check out the [JIRA:How to search for Issues from within a Plugin page](#).

2. Change the atlassian-plugin.xml to point to your new class

In `atlassian-plugin.xml` change the `class` property on your plugin to point to your new class rather than the generic implementation.

```
<project-tabpanel key="example-project-tab"  
                  name="Example Project Tab"  
                  class="com.atlassian.jira.plugin.projectpanel.impl.GenericProjectTabPanel">
```

to

```
<project-tabpanel key="example-project-tab"  
                  name="Example Project Tab"  
                  class="com.atlassian.jira.plugin.RecentlyUpdatedIssuesProjectTabPanel">
```

3. Add some better translations to your properties file

```
example.project-tab.description = A recently updated issues project tab  
example.project-tab.name = Recently Updated Issues Project Tab  
example.project-tab.label = Recently Updated Issues  
example.project-tab.header = Recently Updated Issues
```

4. Add to the velocity template to display the issues.

The velocity template will generate the HTML that displays inside the project tab. We'll replace our Hello World text with some more interesting display. Note that we're taking the `recentlyUpdatedIssues` list that we put into the velocity context in our Java action and using it to display the issues.

```
<div class="projectPanel">  
    <div class="header">  
        <!-- Use the $i18n bean to get any of the translated keys from your plugin -->  
        <h3 class="formtitle">$i18n.getText("example.project-tab.header")</h3>  
    </div>  
    <table cellpadding=3 cellspacing=0 border=0 width="100%" bgcolor="fffff0">  
  
        <!-- $recentlyUpdatedIssues is the object that we placed in the velocity context in our  
        java method -->  
        #if ($recentlyUpdatedIssues.size() > 0)  
            #foreach ($issue in $recentlyUpdatedIssues)  
                <tr bgcolor=ffffff>  
                    <td width=5% nowrap>&nbsp;&nbsp;</td>  
                    <!-- This calls a velocity macro in JIRA that constructs the rows for listing  
                    issues -->  
                    #issueLineItem ($action $issue)  
                </tr>  
            #end  
        #else  
            <tr>  
                <td bgcolor=ffffff>&nbsp;</td>  
                <td colspan=6 bgcolor=fffff0>  
                    <font size=1>$action.getText("common.concepts.noissues").</font>  
                </td>  
            </tr>  
        #end  
    </table>  
</div>
```

5. ???

6. Profit!

Conclusion

That should get you a basic project tab. You can use this as a basis to extend or expand for your own purposes.

How to set up a development environment for a JIRA Plugin



This tutorial assumes that you have access to the JIRA source code and have already followed the steps outlined in [Building and deploying JIRA from source](#). If you don't have a source license, you should follow this tutorial instead.

You now have JIRA successfully deploying and running inside the an appserver and talking to the database of your choice. The next step is to prepare to develop your own [custom plugin](#).

Get the Plugin Development Kit

1. Download the JIRA Plugin Development Kit.
2. Unpack it in \$MY_WORKING_DIR

Create a directory for your new plugin

1. `cp $MY_WORKING_DIR/jira-development-kit-X.X/template/ $MY_WORKING_DIR/`
2. rename the directory `jira-atlassian-project-template` for your plugin, e.g. `my-cool-plugin`
3. `cp $MY_WORKING_DIR/jira-development-kit-X.X/common/project.xml $MY_WORKING_DIR/$MY_COOL_PLUGIN/parent-project.xml`
4. edit `$MY_WORKING_DIR/$MY_COOL_PLUGIN/project.xml`.
 - change `<extend>../../common/project.xml</extend>` to `<extend>parent-project.xml</extend>`
 - enter the correct id, name, version and organization info for your project.



If you're planning on using source control (and we do recommend it), now would be a good time to import your `$MY_COOL_PLUGIN` to your source control repository.

Generate project files for your plugin

Instructions below for for [IntelliJ IDEA](#). You can find instructions for Eclipse [here](#).

Note that you must use Maven 1.0.x. Maven 2.0 will not work

1. `cd $MY_COOL_PLUGIN/`
2. run `maven atlassian-idea`
3. this will create the IDEA project files you'll need in the next step.

Import your plugin module into IDEA

1. Launch IDEA and open the .ipr file that you created in [the previous tutorial](#).
2. go to IDEA > 'Settings' > 'Modules'.
3. Click 'Add'.
4. Choose to 'Import an existing module' and click the '...' button.
5. Select the .iml file that you just created in the `$MY_COOL_PLUGIN` directory.
6. In the 'Dependencies' tab, check 'atlassian-jira' as a dependency.
7. In the lefthand pane, select any of the appserver_env you happen to be using and check 'my-cool-plugin' as a dependency for each of them.
8. Select 'OK' at the bottom of the dialog. You're done configuring!

Start developing

Start developing your plugin as described in the [documentation](#) and the [tutorials](#). Don't forget to properly configure the `atlassian-plugin.xml` so that JIRA will recognize your plugin.

When you 'run' or 'debug' JIRA your new plugin classes will be running and available also.

Writing Something Cool.

Magic happens here.

Package your new plugin for deployment.

When you are satisfied with your development, go to your `my-cool-plugin` directory and run `maven jar`. This will create a jar of your plugin

which you can deploy as described here in JIRA:Deploying a Plugin in JIRA.

Contribute your plugin back to the [JIRA Plugin Library](#)

Install an Appserver

Install an Appserver

You can use the Atlassian-IDEA Maven plugin to configure deployment environments for Resin 2, Resin 3, Tomcat 5.5.x and Orion 2. You can run on any or all of these appservers. Unless you're testing compatibility, you probably only need one. Resin seems to be the most popular, but Tomcat is the basis for the standalone dist, so either of those seem like good choices.

Tomcat Installation

1. [download & install Tomcat 5.5.x](#) (see also [jira documentation](#))
2. [download & install extra jars](#) to `$TOMCAT_HOME/common/lib`.
3. If you want to run Tomcat 5.5 on a 1.4 JDK, download the [Compatibility package](#) and [install](#) it.

Resin 3 Installation

1. [download & install Resin 3.0.x](#) (see also [jira documentation](#))
2. [download & install extra jars](#) to `$RESIN3_HOME/lib`.

Resin 2 Installation

1. [download & install Resin 2.1.x](#) (see also [jira documentation](#))
2. [download & install extra jars](#) to `$RESIN_HOME/lib`.

Orion 2 Installation

1. [download & install Orion 2.0.x](#) (see also [jira documentation](#))

Install Atlassian-IDEA Maven Plugin

Install Atlassian-IDEA Maven Plugin

The Atlassian-IDEA Maven plugin will automatically create the IDEA project files (.ipr, .iml, .iws) necessary for developing and running JIRA from within IDEA.

1. [download the latest Atlassian-IDEA Maven plugin](#)
2. unpack the distribution in `$MY_WORKING_DIR`
3. `cd $MY_WORKING_DIR/atlassian-idea-plugin-X.X.X`
4. run `maven plugin:install`

Setting up your plugin project

Setting up your plugin project



We recommend that you use the [JIRA Plugin Development Kit](#) when developing plugins for JIRA. The kit contains all the dependencies required for JIRA as well as a skeleton project. You should download the version development kit that is relevant for the JIRA version you're building the plugin for. e.g (jira-development-kit-0.5.zip for JIRA 3.2 etc.)

You can choose to develop your plugins however you wish. However, we'd recommend the following using [Maven 1.0](#) and the [skeleton plugin in the development kit](#). You only need to run the command `maven jar` in the correct folder to build your plugin. What could be simpler! The development kit comes complete with the source code for the tutorial as well as other plugins for you to check out. Using Maven is **not** a requirement - you can use ant or any other build tool, however, it *will* make your life a lot easier.

Configuring your project for Maven

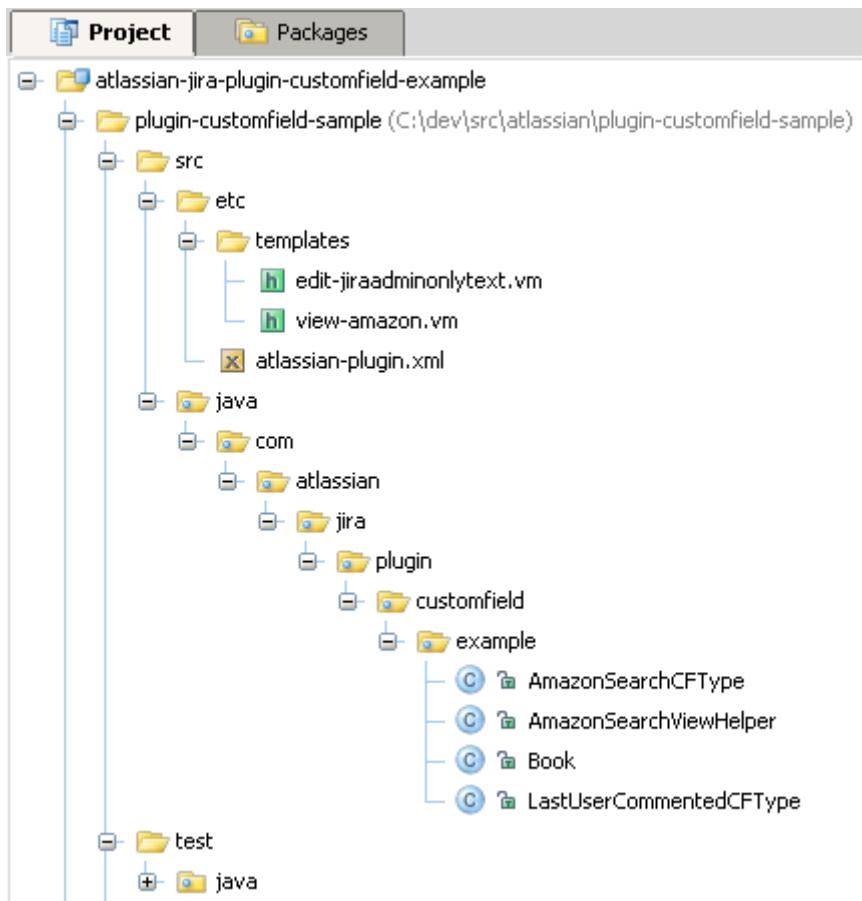
You will first need to download and install [Maven](#). Maven is an ant-like build tool that allows downloads any specified project dependencies automatically (just one of the many [features](#)).

We have provided a project skeleton in the development kit that will work with Maven (and is used for the example). Navigate to the `project-template/jira-atlassian-project-template` folder. You will see the following files.

- `project.xml`

- `project.properties`
- `src/etc/atlassian-plugin.xml`

Maven uses the `project.xml` to build your project. Update this with your relevant project information and put any other dependencies that your plugin require here. Note this file extends the `project.xml` in the `common` folder and all of JIRA's dependencies as well as common maven tasks are there. See the Maven documentation for more on the [project descriptor](#). If you access the Internet through a proxy, you'll also need to configure the `project.properties` file. Lastly, the `atlassian-plugin.xml` configures your plugin for JIRA. See [A brief guide to JIRA plugins](#) for more details.



Maven also provides plugins to help you setup your projects in popular IDEs such as [Eclipse](#) and [IDEA](#). We'd recommend using this if you're planning to use either of the two editors.

Building and Deploying your Plugin

To build using Maven run:

```
maven jar
```

in root directory of your project. Any dependencies will be downloaded and a jar created in the target directory. To deploy, copy the JAR to the lib folder of your JIRA instance. If you have extra libraries that's required beyond what is available in JIRA, you'll need to copy them over as well. Restart your server and your custom fields will be there ready and waiting!

Old Plugin Documentation



Try the new SDK

We encourage you to try the new Atlassian Plugin SDK, which supports all Atlassian products.

- Overview
- Quickstart for the impatient
- Requirements
- Settings.xml
- Creating a skeleton plugin
- Special section for commercial license holders with Confluence source access.
- Maven and your IDE
 - Using a different build tool
- Developing your plugin

- Testing your plugin
- A note about quality
- Packaging and Releasing your Plugin
- Resources

Overview

This documentation contains all of the information you'll need to start building your own plugins and extensions for Atlassian's products. We set several goals for our plugin development process:

- Allow you to set up a complete plugin development environment with just a few commands.
- Ensure that all plugins are set up, built, tested and released in a consistent way.
- To allow you to use the IDE you prefer.
- To make the code/deploy/test cycle as efficient as possible.
- To encourage unit and functional testing
- To use continuous integration and code coverage metrics to ensure plugins are of high quality, and that they stay compatible with new releases of the products.
- Make it easy to release your plugins to the community.
- To encourage collaboration around plugin development.

Quickstart for the impatient

- Download and install [JDK 1.4](#) or [JDK 5](#)
- [Download and install the latest release of Maven 2](#).
- Follow the instructions at [Example settings.xml](#)
- Create a plugin template using one of the [Atlassian Plugin Archetypes](#).
- Run `mvn idea:idea` or `mvn eclipse:eclipse` in the top-level directory of your new plugin.
- Open your IDE and [start coding](#).
- Write unit and functional tests.
- [Release your plugin](#).



Please note that the new Eclipse plugin is unable to deal with source 'includes' and 'excludes'. See [Eclipse Maven Plugin Build Error](#), and [\[Plugin Development Tips, FAQ and Troubleshooting\]](#) for more information.

Requirements

Java: Depending on which Atlassian product (and which version of the product) you are building plugins for, you may need either JDK 5 or JDK 1.4. Many Atlassian customers are still using Java 1.4, either by choice or because their app servers still require it. If you are building a plugin exclusively for internal use, you're free to target whichever JDK you wish. However, if you are planning to share your plugin with other Atlassian users, you *must* check what requirements your targetted product has, and unless stated explicitly we recommend you make sure that it is compatible with Java 1.4.

- [Download and install JDK 1.4](#)
- [Download and install JDK 5](#)

Read [\[How do I specify a particular version of Java?\]](#) to learn how to use Java 5 language features or enforce building with a particular version of Java.

Maven 2: The Atlassian plugin development process depends heavily on [Maven](#). Calling Maven a "build tool" would be an understatement - but if you have no experience with Maven yet, start thinking of it as a tool for simplifying your life by building your plugin. Quoting its homepage, "Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information."

Maven is particularly useful in this instance because it can automatically manage massive dependency trees - especially transitive dependencies where `Plugin X` depends on `Product Y`, which depends on `Shared Library Z`. Using Maven, you should be able to quickly and easily retrieve all the dependencies that your plugin will need, and create a working development environment in just a few steps.

- [Download and install the latest release of Maven 2](#). You must install Maven 2.0.9 or newer.

Obsolete Atlassian Maven PDK: We've written a Maven extension specifically for plugin developers. The extension provides a new packaging type for `atlassian-plugins`, gives you some utility methods for uploading, enabling, disabling and uninstalling a plugin to the product, and provides methods for integration and functional testing. This plugin is automatically installed when you build an Atlassian plugin.

- [More information about the Atlassian Maven PDK](#)

Settings.xml

In order for Maven to find and download all of the necessary dependencies, we have to tell it where everything is stored. The best option is to do that once, in your `$HOME/.m2/settings.xml` file. We've included instructions and an example that you can copy and paste at [Example settings.xml](#). If you look closely, you will find two lines containing placeholders for username & password. Don't worry about them at this

moment — you don't need them to get started.

Your plugin will depend on many different artifacts: the Atlassian product itself, Atlassian modules and various open-source libraries. All of those artifacts are stored in different maven repositories scattered around the net. However, to simplify configuration and speed downloads, Atlassian provides a [Maven proxy](#) that contains *all* of the dependencies for all of our products. The example settings file contains just one repository entry for this proxy, and saves Maven from having to check several different repositories for each artifact it needs. You can find more information about the [repositories that are behind the Maven Proxy](#).

Atlassian will release binary and javadoc artifacts for all of our product releases and their Atlassian-managed dependencies. We will release source artifacts wherever we can freely do so. Some of our closed-source artifacts will only be available via manual download by source-license-holders.

We will also be releasing various milestone and snapshot releases during product development to aid plugin developers in testing their work against upcoming product releases.

You will be able to find all of these artifacts in the [Maven proxy](#).

Creating a skeleton plugin

Now you're ready to create your (skeleton) plugin. This involves creating plenty of directories and subdirectories, two XML files and initial Java files. Fortunately, Maven can help here, too, and simplify your life a lot! Maven has a concept called an [Archetype](#), which is basically a plugin template. We've created plugin templates for each of our pluggable products, so you can use those templates to set up your initial plugin skeleton easily. You can find the [Atlassian Plugin Archetypes](#) and how you use them here. Visit the page, create a plugin for your targetted Atlassian product, and return to this page later.

Running an plugin archetype will download dependencies from the internet, set up a project skeleton, and prepare you to run unit and functional tests. Once you run the Maven archetype command, you'll find that it creates a new directory structure that looks like this: Unable to render embedded object: File (atlassian-archetype-directory-structure.png) not found.

Here a a few points to note:

- the "pom.xml", which stands for [Project Object Model](#) definition file.
- Directories for your sources and test sources.
- Directories for your resources (templates, javascript, css, images, etc.)
- The "[it](#)" package for your integration tests.
- An [atlassian-plugin.xml](#) plugin definition file.
- A license file.

It's important to understand a little bit about the pom.xml, and what's in it. The [Project Object Model](#) describes everything that maven needs to know about your plugin, from dependencies, to source control, to authors, to tests, to deployment locations. The POM is also hierarchical, so many properties are inherited. You'll note that this pom.xml inherits from something like [confluence-plugin-base](#), which contains lots of information about the products and the Atlassian development environment. Some properties are later overridden in your plugin's POM.

We've written a page with an [Example pom.xml](#) that explains what's going on in your plugin's POM.

Please note: The plugin now knows about the target product's binaries, because it has downloaded the JAR files - but it has not downloaded the actual source code. You don't need to have access to the target product's sourcecode to be able to develop a plugin, the JAR files alone will enable you to go and write the plugin, using the product's API. If you **do** have access to the source code (e.g. as a commercial license holder) then it helps to download and set up the source code now:

Special section for commercial license holders with Confluence source access.

Since you have purchased a commercial license to Confluence, you have access to the Confluence source code. You can download a source distribution by visiting <http://my.atlassian.com>. You can attach the Confluence source code to your plugin project so that you can step through the Confluence code in your debugger. It's often helpful to do this to better understand that API, or to track down problems.

Once you've downloaded and unpacked the source distribution, you'll see a bunch of directories and files, exactly as they are checked out of our SVN repository. This is perfect for reading and exploring, but it doesn't do you much good in the brave new world of Maven 2. So what you'll need to do is create and install source artifacts into your local maven repository. Once you've done that, the maven IDE plugins will recognize that they are there and link them up for debugging purposes.

From the top level of the source distribution, run:

```
mvn source:jar install -Dmaven.test.skip=true
```

This will build JAR files containing most of our source-code into the `target` directory of each subproject. There are still a few items that aren't yet set up for this, but the important ones are, like the core product source code. Likewise, we will soon be enabling this same process for JIRA.

The JAR files can be copied into your local Maven repository to make them available in your IDE when you run `mvn idea:idea` or `mvn eclipse:eclipse`. For example, to install the source code for Confluence 2.10 core, copy `confluence-project/confluence/target/confluence-2.10-sources.jar` to `$HOME/.m2/repository/com/atlassian/confluence/confluence/2.10`.

A note about memory: This is a resource intensive process, and you may need to allocate more memory to maven in order to complete it. You can do so by setting an environment variable called `MAVEN_OPTS`, like so `export MAVEN_OPTS=-Xmx512m`

A note about versions: Make sure that you download and install the **same** version of the source code that is set in the `atlassian.product.version` property in your `pom.xml`.

Maven and your IDE

We have our plugin skeleton, and now it's time to get it set up in your IDE, if you use one. Once again, Maven comes to the rescue. Maven has plugins to generate project files for both [Eclipse](#) and [IntelliJ IDEA](#). Our instructions will cover these two IDEs, but you're free to use whichever IDE or editor you prefer.

To set up your IDE project, simply run `mvn eclipse:eclipse` or `mvn idea:idea` from the root folder of the newly created directory (e.g. where the `pom.xml` resides).

First, Maven will download all of the dependencies referenced in your `pom.xml`, and then all the inherited dependencies from the parent POMs. Your `settings.xml` specifies that maven should download the javadoc and source files as well, where they are available. This is likely to take quite a while: Depending on what product you are targeting, Maven will now download hundred MB or more from the internet. Go relax, get some coffee, read a blog. Fortunately, after you've downloaded these files for the first time, they are stored in your local Maven repository (`$HOME/.m2/repository/`) and won't be downloaded again unless they change.

In case this process ended with an error message (e.g. "required artifact missing") please refer to [\[Plugin Development Tips, FAQ and Troubleshooting\]](#)

Second, Maven will construct project files that your IDE can understand. It will reference all the dependent jars in the right projects, and attach the sources and javadoc where they are available. Note, when you're ready to check in your plugin, don't commit these project files as they are generally specific to each developer's machine.

Open the project files in your IDE, and you should see the mostly-empty template directory for your new plugin, matching the directory structure in the screenshot above.



Eclipse Workspace Setup

Eclipse requires an additional one-time step to configure your workspace with the Maven repository location:

```
mvn -Declipse.workspace=<path-to-eclipse-workspace> eclipse:add-maven-repo
```

You can also accomplish the same thing by adding the `M2_REPO` classpath variable, pointing to your `.m2/repository` directory, in Eclipse preferences

(see also the [Maven Eclipse Plugin Usage guide](#)

Using a different build tool

These instructions depend on using Maven. However, if you want to use your own build environment (e.g. Ant), the following Maven command will export all the JAR files you need to develop against to the directory you specify, and you can then ensure that these JAR files are in your CLASSPATH using your build tool.

```
mvn dependency:copy-dependencies -DoutputDirectory=<specify a directory>
```

Developing your plugin

So now what? Find out how the [development/test/debug cycle](#) works. We've tried to make it as efficient as possible.

Testing your plugin

Atlassian plugins depend on Maven 2 for running their test suites. This is advantageous because all plugins are always tested the same way, and because Bamboo, our continuous integration server, can do likewise.

Building a suite of tests for your plugin is **extremely** important. We're big believers in [Test Driven Development](#), and we think that writing tests as you go is one of the best ways to ensure you build a high-quality and functional application. Additionally, having a comprehensive suite of tests means that other developers in the community (or even Atlassian developers) can jump in and collaborate on your plugin without fear of breaking anything inadvertently.

Find out how to make it happen on the [Testing your plugin page](#).

A note about quality

We have set certain criteria for our own plugin development. If you are working on a plugin that you wish to share with the community, we encourage you to try and meet these same criteria in your work. It will result in a higher quality product and make it easier for other to collaborate.

Your plugin should:

- be available in [Developer Network SVN](#).
- have its dependencies in the [Developer Maven Repository](#).
- have a [\[JIRA project\]](#).
- have a [Bamboo Project](#).
- be in the [plugin repository](#) (for Confluence).
- have unit and functional tests with a significant amount of code-coverage.
- have complete, accurate and attractive documentation in the [Plugin Library](#) on Confluence.
- display a compatibility matrix for the last three product releases.
- be [internationalized](#).

Atlassian offers hosting for plugins to all authors. Find out more about the resources and how to take advantage of them [here](#).

Packaging and Releasing your Plugin

Atlassian offers lots of services to help you host your plugin development project. We can provide an SVN repository with Fisheye and Crucible, a JIRA project, Confluence space, a Bamboo Project, a Maven repository for artifacts, and more. When you're ready to start collaborating with other developers on your plugin project, read about [Packaging and Releasing your Plugin](#). [Contact us](#) and we'll grant you account and get everything set up for you.

Resources

Where do you go for help? For questions about plugin development for Atlassian products, head to the [Developer Forum](#). These are community run forums where other plugin developers ask and answer questions, get advice, and publicize their latest work.

- [\[Plugin Development Tips, FAQ and Troubleshooting\]](#)
- [Other Information](#)
- [Developer Mailing Lists and Support](#)
- [Report bugs with the plugin development process](#)

NOTE: Please do not file support tickets at <http://support.atlassian.com> regarding plugin development questions. The engineers on support are not equipped to answer those sorts of questions.

Developing your plugin



Try the new SDK

We encourage you to try the new [Atlassian Plugin SDK](#), which supports all Atlassian applications.

There are several steps involved in actually coding a plugin. We're not going to discuss the details of *what* you need to do on this page, we'll leave that for the product specific plugin guides. But we will outline here how you need will work and test your plugin efficiently.

In order to test your plugin, you need to deploy it into a running instance of the product. While you could start and stop the server manually and reinstall in the latest version of your plugin after each code-change, you can also use Maven to automate a lot of tasks: A simple Maven task can set up and start an application server, a simple database with some basic settings, and enable you to tear the application down easily too. You can even drop the complete setup by just removing one directory and start all over again, at no extra configuration cost.

Basically, what you do is run the command

```
> mvn -Pplugin-debug
```

This creates a "plugin-debug" folder, starts up Tomcat, puts all the logfiles into the appropriate subdirectories, and enables you to use the product straight away: Once you see a note on the commandline interface telling you that you can stop the server using CTRL-C, you will know it is up and running and accepting browsers to connect to it.

There are some subtle differences in how this works for Confluence and the other products:

JIRA, Bamboo and other products

Plugins in JIRA are not dynamically loadable. They must be installed in JIRA's `WEB-INF/lib` and the application restarted. The Atlassian Plugin Maven Archetypes come with a profile called `plugin-debug`. Running the command

```
> mvn -Pplugin-debug
```

will compile your plugin's classes, create a jar in the `/target` directory, copy that jar into JIRA's `WEB-INF/lib` directory and restart JIRA. The process will continue running in the foreground, so when you're ready to stop or restart JIRA, you just need to hit CTRL-C to quit.

Maven will also install some default data into this instance: things like a license, database configuration, and an admin account. This will save you the effort of having to set up JIRA each time. It's important to note that the license installed is a unique one: it is perpetually valid, but only for a period of three hours (which should be more than enough time to run your tests.) Should you keep going longer than three hours, JIRA will lock you out, and you'll need to restart it to proceed.

JIRA will be running at <http://localhost:1990/jira>, so you can go directly there in your browser and see your latest changes. You'll be able to log in with the username `admin` and the password `admin`.

As you make changes, you'll have to kill the JIRA process (ctrl-c) and the run `mvn -Pplugin-debug` again.

Confluence

Confluence works very similarly, but it has the advantage that plugins are dynamically reloadable. So we take advantage of this to speed the development process. When you run

```
> mvn -Pplugin-debug
```

in a Confluence plugin we follow the same steps at first: mvn compiles the plugin, creates a jar, and starts Confluence. As with JIRA, the process will continue running in the foreground, so you just need to hit CTRL-C to quit.

Maven will also install some default data into this instance: things like a license, database configuration, and an admin account. This will save you the effort of having to set up Confluence each time. It's important to note that the license installed is a unique one: it is perpetually valid, but only for a period of three hours (which should be more than enough time to run your tests.) Should you keep going longer than three hours, Confluence will lock you out, and you'll need to restart it to proceed.

However, rather than copying the JAR to `WEB-INF/lib` then in uses the [Obsolete Atlassian Maven PDK](#) to dynamically deploy the plugin. From there, you can run the PDK goals like `uninstall` and `install` to deploy your changes without having to wait for Confluence to restart. You'll want to keep another terminal window open so that you can run these commands while the Tomcat process is running in the other window.

Confluence will also be running at <http://localhost:1990/confluence>, so you can go directly there in your browser and see your latest changes. You'll be able to log in with the username `admin` and the password `admin`.

As you make changes to your code, you can redeploy your plugin by using the [Obsolete Atlassian Maven PDK](#) and running

```
> mvn atlassian-pdk:install
```

The main logfile you will be interested in is located in "target" and called "output.log".



Limitations in Confluence versions 2.6 and before:

1. The default admin/admin username/password will not let you log in. This problem was corrected with Confluence 2.6.1, so if you change your `atlassian.product.version` or your `atlassian.product.data.version` property to 2.6.1 or later, you'll be able to log in.
2. However, even after you log in, you might find that the Confluence license is expired. This was corrected in Confluence 2.7, so if you bump your `atlassian.product.version` property up to 2.7 or later, it will work normally. If you still need to compile against 2.6, you can retrieve a new license from <http://my.atlassian.com> and install that in your test server using the web interface.

Debugging your plugin

When you run JIRA or Confluence with the plugin-debug profile, they are automatically configured to accept a remote debugger - in this case IDEA or Eclipse. We've also made sure to include source-code artifacts for most of the important code in the products: both the core applications and key libraries like `atlassian-plugins` or `atlassian-renderer`.

Maven attaches those source code artifacts to your plugin automatically, so you can pull up most classes and read the code or insert breakpoints as needed.

You'll want to [create a remote debug target](#) in your IDE. But once you've done that, you can start the debugger to JIRA or Confluence and step through the code in your plugin, or in the product itself. (You will not need to explicitly use VM parameters when running your plugin, as IntelliJ indicates when you are creating a remote debug target, because the parameters have already been set up for you.)

Please see also the [Ports used by applications](#) section in the remote target creation page.

Testing your plugin



Developing for Confluence, JIRA or Bamboo? Use the new SDK

If you are developing a plugin for Confluence, JIRA or Bamboo, try the new [Atlassian Plugin SDK].

Atlassian plugins depend on Maven 2 for running their test suites. This is advantageous because all plugins are always tested the same way, and because Bamboo, our continuous integration server, can do likewise.

Building a suite of tests for your plugin is **extremely** important. We're big believers in [Test Driven Development](#), and we think that writing tests as you go is one of the best ways to ensure you build a high-quality and functional application. Additionally, having a comprehensive suite of tests means that other developers in the community (or even Atlassian developers) can jump in and collaborate on your plugin without fear of breaking anything inadvertently.

Our products are always moving forward. We enhance and change things. We move or deprecate APIs. A plugin that works perfectly in Confluence 2.5 might change in Confluence 2.6, in subtle or profound ways. That's the primary reason that we run the [Developer Bamboo Server](#) – to alert us to any changes that might break plugins. When those breakages occur, we might be able to go back and do things another way that preserves compatibility. Or, at the very least, we can notify developers in advance of a new release about what has changed. Continuous Integration is a crucial safeguard, and it depends largely on a comprehensive test suite to alert us when things break.

Unit Tests

The first tests you should write for your plugin are [unit tests](#). Unit tests are designed to exercise individual chunks of your code in isolation. To run unit tests, the default Maven 'test' command works:

```
> mvn clean test
```

For examples of unit tests, take a look at the [Confluence Chart Plugin test](#) or the [JIRA Fisheye Plugin tests](#).

Integration, or Functional, Tests

Start by reading this [great testing tutorial](#) from Customware.

[Integration Testing](#) is designed to test your code inside the the Atlassian product. The goal is confirm that the plugin functionality works as designed, and is compatible with various versions of the product. To achieve this goal, Maven will actually deploy your plugin to a running version of JIRA or Confluence and then run tests that you write to exercise its functionality. To run integration tests, the command is:

```
> mvn clean integration-test
```

When you run this command, Maven will start up a new instance of JIRA or Confluence, running inside Tomcat, and install some default data into it: things like a license, database configuration, and an admin account. This will save you the effort of having to set up the product each

time. It's important to note that the license installed is a unique one: it is perpetually valid, but only for a period of three hours (which should be more than enough time to run your tests.) Should you keep going longer than three hours, the license system will lock you out, and you'll need to restart it to proceed.

For examples of integration tests, take a look at the [JIRA Fisheye Plugin tests](#). A Confluence example is forthcoming.

Skipping tests

To skip the various test phases, add one of the following options to your Maven command:

```
> -Dmaven.test.skip=true - skips both unit and integration tests  
> -Dmaven.test.unit.skip=true - skips unit tests  
> -Dmaven.test.it.skip=true - skips integration tests
```

Notes

Oftentimes, your functional tests will need to set up the product in a particular state. You may wish to do this by creating an XML export of JIRA or Confluence data in a known state, and importing that data at the beginning of your tests. XML data that is usually imported by the tests into the running product should be placed in `src/test/xml/` directory provided.

Packages containing integration tests should start with `it.`. For instance:

- `it.com.atlassian.jira.plugins.fisheye`
- `it.com.atlassian.jira.plugins.calendar`

This is due to inability of Maven to discriminate unit tests from integration tests. This will be resolved in the coming Maven 2.1 and we will take care of the migration for you then.

System Properties

The following system properties are available to control the behaviour of the integration test harness:

- **tomcat.installer.url** - path to the tomcat zip file to install. Default:
<https://m2proxy.atlassian.com/repository/public/org/apache/tomcat/apache-tomcat/5.5.23/apache-tomcat-5.5.23-jdk14.zip>
- **cargo.wait** - if this is set to true tomcat will be started up and then cargo suspends allowing you to manually access this JIRA from the browser.
- **atlassian.product.version** - version of the Atlassian product to compile and test against. Depending on which product you're developing a plugin for, this could be the version of Bamboo, JIRA or Confluence. Below are the defaults:

| Product | Version |
|------------|---------|
| Bamboo | 2.0 |
| Confluence | 2.9 |
| JIRA | 3.12.2 |

- **atlassian.product.data.version** - version of the test resource bundle that contains the basic Atlassian product configuration data for the integration test environment. These versions mimic the actual Atlassian product versions. However we might only modify and release the relevant projects for the reasons of non-backwards compatibility of the new versions of Atlassian products. Therefore not every version of Atlassian products will have a corresponding version of the resource bundle. Below are the default versions of test resource bundle used according to product.

| Product | Version |
|------------|---------|
| Bamboo | 2.0 |
| Confluence | 2.9 |
| JIRA | 3.12.2 |

- **atlassian.product.url** - base URL of the Atlassian product deployed during integration tests. Below tables out the defaults according to products

| Product | URL template | Default |
|---------|---|---|
| Bamboo | <code>http://localhost:\$ {http.port}/ \${atlassian.product.context}</code> | <code>http://localhost:1990/bamboo</code> |
| JIRA | <code>http://localhost:\$ {http.port}/ \${atlassian.product.context}</code> | <code>http://localhost:1990/jira</code> |

| | | |
|------------|--|---|
| Confluence | <code>http://localhost:\$ {http.port}/\${atlassian.product.context}</code> | <code>http://localhost:1990/confluence</code> |
|------------|--|---|

- **http.port** - port on which the deployed instance of Atlassian product listens on. Default: 1990
- **rmi.port**

Debugging your failing tests

All of the necessary output generated by tests will be found under `target` subdirectory of your module. The following files are useful places to look at dire times:

- `surefire-reports/*.txt` — a log file per test class of failed tests (both unit tests and functional tests)
- `<product>/output.log` — where `product` is one of `bamboo`, `confluence` or `jira` depending on the plugin you're developing.
This is the Atlassian product log file (e.g. `atlassian-jira.log`).¹

¹ — The `<product>` directory will be found under `debug` if the Maven profile, `plugin-debug` is activated. That means, when the `plugin-debug` profile is activated, the directory becomes `debug/<product>/output.log`. This isolates debug resources and the actual integration test resources.

Packaging and releasing a plugin

Packaging your plugin

In order to install your new plugin on your production instance of Confluence, you'll need to package it up and install it there.

1. Navigate to your plugin's root directory.
2. Make sure that your `POM.xml` is depending to the version of Confluence on which you plan to deploy.
3. Run the command `mvn package`
4. That will create `my-plugin-X.X.X.jar` in the target directory.
5. You can now [install this plugin in Confluence](#).

* If your plugin has dependent jars, see this [page](#).

Releasing your plugin

We encourage you to open-source (we recommend a [BSD-style license](#)) your plugin and put it on Atlassian's [Developer Subversion Repository](#) and add it to the [Confluence Extensions Library](#) for other Confluence users to use and enhance. Never underestimate the power of collaboration!

1. Get Subversion access (email developer-relations@atlassian.com).
2. Request a [JIRA project](#) for your plugin (email developer-relations@atlassian.com).
3. Check in your code to Subversion.
4. [Add your plugin to the Plugin Repository](#)
5. Create a homepage for your plugin in the [Confluence Plugin Library](#).
6. Announce your plugin on the [Confluence Mailing Lists](#).

Plugin Developer Kit for Maven1

| | |
|------------------|---|
| Name | Plugin Developer Utilities |
| Classification | Maven Plugin |
| Version | 1.2 |
| Status | Stable |
| Product Versions | All |
| Author(s) | Dan Hardiker |
| Homepage | Plugin Developer Kit for Maven1 |
| Price | free! |
| License | BSD |
| Issue Tracking | http://jira.adaptavist.com/browse/CDU |
| JavaDocs | none |



If building plugins with Confluence 2.3+, use the [Maven 2 PDK](#) instead.



You can now download it with out needing to compile it yourself! (remove the \'s if you place it all on one line)

```
maven plugin:download \
-Dmaven.repo.remote=http://files.adaptavist.com/maven \
-DgroupId=plugin-dev-utils \
-DartifactId=plugin-dev-utils \
-Dversion=1.2
```

Description

A collection of utilities in the form of a Maven Plugin for automating boring development tasks. Primarily aimed at helping anyone developing plugins for confluence using maven as their build environment to get a more productive development environment.

Dependencies

- Maven 1.0.2

Installation

While this library doesn't yet exist in any public repository, you can do one of the following.

Installation from Source

- Download the [CONFEXT:latest source](#) to a directory on your local machine
- Run `maven plugin:install` from that directory

This will build and install the plugin into your local maven build system. The plugin can be uninstalled by running `maven plugin:uninstall` from the same place.

Installation from Release

You can now download it with out needing to compile it yourself: (remove the \'s if you place it all on one line)

```
maven plugin:download \
-Dmaven.repo.remote=http://files.adaptavist.com/maven \
-DgroupId=plugin-dev-utils \
-DartifactId=plugin-dev-utils \
-Dversion=1.2
```

Required Build-time Properties

This plugin looks for configuration information in a number of properties which must be specified somewhere in your build hierarchy (in your project's `project.properties` / `build.properties` or your user's `build.properties`). Which goals need which properties is shown in the [Usage](#) section.

A sample `build.properties` file.

Dependencies

This plugin is now capable of bundling dependancies into your plugin jar automatically (either as jar's compatible with Confluence 2.2.1a or exploded as classes for backward compatibility), however it needs to be told which of the dependancies are to be bundled. The following example depends on MySQL at build time (and assumes its in the live env if needed), and [Dan Hardiker's](#) DWR hack at run time (and wants it bundled).

```

<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>3.1.12</version>
    </dependency>
    <dependency>
        <id>dwr</id>
        <version>1.1.1-confPluginHack</version>
        <properties>
            <plugin.bundle>true</plugin.bundle>
        </properties>
    </dependency>
</dependencies>

```

The property you specifically need to add is:

```
<plugin.bundle>true</plugin.bundle>
```

Usage

This is the simple part! Where you would run `maven jar`, you now run one of the following goals using maven.

All of them require the following variables:

- `confluence.dev.server.url`
- `confluence.dev.server.user`
- `confluence.dev.server.pass`

Some require more, and those with dependancies inherit those variable requirements too.

plugin-dev-utils:lib-jar

For the new recursive jar classloader which can deal with bundled libraries.

| | |
|---------------------------|----------------------|
| Depends on | <code>jar:jar</code> |
| Required Variables | <code>none</code> |

- Builds the plugin jar
- Extracts it to a temp directory
- Adds the dependancies as jar's to the temp dir appropriately
- Compresses the temp directory back into the plugin jar

plugin-dev-utils:class-jar

Backward compatibility with existing classloaders.

| | |
|---------------------------|----------------------|
| Depends on | <code>jar:jar</code> |
| Required Variables | <code>none</code> |

- Builds the plugin jar
- Extracts it to a temp directory
- Extracts the dependancies into the temp dir
- Compresses the temp directory back into the plugin jar

plugin-dev-utils:rescan

| | |
|---------------------------|----------------------|
| Depends on | <code>nothing</code> |
| Required Variables | <code>none</code> |

Uses a HTTP Client to rescan the plugins using the plugin manager.

plugin-dev-utils:uninstall

| | |
|-------------------|----------------------|
| Depends on | <code>nothing</code> |
|-------------------|----------------------|

| | |
|---------------------------|---------------------------------|
| Required Variables | confluence.dev.server.pluginKey |
|---------------------------|---------------------------------|

Uses a HTTP Client to uninstall (and delete) the plugin by the key defined using the plugin manager.

plugin-dev-utils:reinstall

| | |
|---------------------------|--|
| Depends on | plugin-dev-utils:uninstall, plugin-dev-utils:lib-jar |
| Required Variables | confluence.dev.server.home |

- Uninstalls the plugin using `plugin-dev-utils:uninstall`
- Builds the plugin jar
- Copies the jar to `confluence.dev.server.home/plugins`
- Rescans for new plugins using `plugin-dev-utils:rescan`

plugin-dev-utils:upload

| | |
|---------------------------|--|
| Depends on | plugin-dev-utils:uninstall, plugin-dev-utils:lib-jar |
| Required Variables | <code>none</code> |

- Uninstalls the plugin using `plugin-dev-utils:uninstall`
- Builds the plugin jar
- Uploads the plugin jar to the plugin manager using a HTTP Client
- Enables the plugin (workaround for a bug in Confluence 2.2, fixed in Confluence 2.2.1a - the work around will be removed in 1.3)

plugin-dev-utils:enable

| | |
|---------------------------|-------------------|
| Depends on | <code>none</code> |
| Required Variables | <code>none</code> |

- Enables the plugin

plugin-dev-utils:disable

| | |
|---------------------------|-------------------|
| Depends on | <code>none</code> |
| Required Variables | <code>none</code> |

- Disables the plugin

Release History

| Version | Date | Author | Notes |
|---------|-------------|----------------|--|
| 1.0 | 22-Feb-2006 | Adaptavist.com | Initial version. |
| 1.1 | 23-Feb-2006 | Adaptavist.com | <ul style="list-style-type: none"> • Using a maven plugin now • Moved to jellybean's instead of ant:java • Corrected documentation link |
| 1.2 | 17-May-2006 | Adaptavist.com | <ul style="list-style-type: none"> • Renamed to plugin-dev-utils • Added enable / disable goals • Added bundling lib-jar and class-jar goals • Made the plugin binary downloadable |

Plugin Developer Kit for Maven2

| | |
|-------------------------|---|
| Name | Atlassian Plugin Developer Kit for Maven2 |
| Version | 2.0 |
| Product Versions | All |

| | |
|------------------------|---|
| Author(s) | Dan Hardiker, Michael Mekaail, Tony Truong, David Peterson |
| Price | free |
| License | |
| JavaDocs | |
| IssueTracking | http://developer.atlassian.com/jira/browse/CPDK |
| Download JAR | atlassian-pdk-2.0.jar |
| Download Source | Subversion |

Description/Features

This is a Maven 2 plugin which assists in building and deploying Atlassian plugins.

Usage

Getting started (Simple)

To start off with, you should set up the **Project Object Model (POM)** of your project. A good way to do this is to use a **Maven Archetype**. Archetypes will create a basic project structure for you automatically.

Maven ships with some default archetypes, but there are also some custom archetypes available to streamline building an Atlassian Plugin:

- [Atlassian Plugin Archetype](#) - This is best if you are creating a new plugin from scratch.
- [Legacy Plugin Archetype](#) - This will assist in converting an existing plugin to Maven 2.

Usage

`mvn package`

| Command | Description |
|--------------------------|--|
| <code>mvn package</code> | Creates a Jar in the target directory with any dependencies your plugin depends on. |

Variables

| Variable | Default | Type | Purpose |
|---------------------|---------|---------|---|
| extractDependencies | false | boolean | If true the plugin will contain all dependencies extracted into the root of the jar. If false the plugin will contain the dependencies as jars files in <code>META-INF/lib</code> . |

Note: Any dependency that has a `<scope>compile</scope>` or `<scope>runtime</scope>` will be packaged in the final JAR. All other dependencies will be ignored. To exclude particular dependencies, use the `provided` scope, which indicates that the environment is expected to provide the library at runtime.

Plugin Goals

The following goals manage the plugin's installation in a specified Confluence server. There are several properties which should be specified in your local `settings.xml` file which are used by most of these goals:

| Property Name | Default | Description |
|--|---------------------|---|
| <code>atlassian.pdk.server.url</code> | <code>None</code> | The URL for the base address of the Confluence server the plugin should be installed on. Eg. ' <code>http://localhost:8080/confluence</code> '. |
| <code>atlassian.pdk.server.username</code> | <code>None</code> | The username to log in as when installing. This user must have administration privillages on the server. |
| <code>atlassian.pdk.server.password</code> | <code>None</code> | The password to log in with when installing. |
| <code>atlassian.pdk.installation.method</code> | <code>upload</code> | The method to use when installing to the server. May be ' <code>upload</code> ' or ' <code>copy</code> '. See <code>atlassian-pdk:install</code> for details. |

| | | |
|----------------------------------|------|--|
| atlassian.pdk.server.home | None | This is only used if <code>atlassian.pdk.installation.method</code> is set to 'copy'. It should point to the Confluence home directory on your local machine, and is used for copying the plugin for installation. |
|----------------------------------|------|--|

atlassian-pdk:disable

Disables the current plugin in the server specified in `atlassian.pdk.server.url`.

atlassian-pdk:enable

Disables the current plugin in the server specified in `atlassian.pdk.server.url`.

atlassian-pdk:install

Packages the plugin, uninstalls any existing copies, and installs the new version to the server specified in `atlassian.pdk.server.url`.

There are two methods for installing, specified by setting the `atlassian.pdk.installation.method` parameter to one of the following:

1. **upload**: The plug is uploaded to the webserver. This is the same as going to the plugin admin page, navigating to the plugin jar file and clicking 'Upload'.
2. **copy**: The plugin is copied to a local directory containing the 'Confluence Home' directory, and the server is told to rescan the directory for new plugins.

atlassian-pdk:uninstall

Uninstalls the current plugin from the server specified in `atlassian.pdk.server.url`, if it exists.

atlassian-pdk:rescan

Asks the server specified in `atlassian.pdk.server.url` to rescan the plugins directory for new plugins.

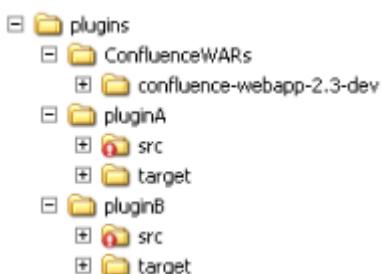
Examples

Version History

Open Issues

jiraissues: JIRA project does not exist or you do not have permission to view it.

Screenshots



Atlassian Plugin Archetype

| | |
|-------------------------|--|
| Name | Atlassian Plugin Archetype for Maven2 |
| Version | 2.3 |
| Product Versions | All |
| Author(s) | David Peterson |
| Price | Free |
| License | BSD |
| IssueTracking | (soon to be implemented) |
| Download JAR | atlassian-plugin-archetype-2.3.jar |

Description/Features

This is a Maven 2 archetype which assists in creating a new Atlassian Plugin project.

Usage

To use this archetype, open a command prompt and navigate to the directory where you would like to create the plugin project, and enter the following:

```
mvn archetype:create -DarchetypeGroupId=com.atlassian.maven.archetypes  
-DarchetypeArtifactId=atlassian-plugin-archetype -DarchetypeVersion=2.3 \  
-DremoteRepositories=http://svn.atlassian.com/svn/public/contrib/repository/maven2 \  
-DgroupId=[your group id] -DartifactId=[your artifact id] -Dversion=[initial version]  
-DpackageName=[your package]
```



Once you've run the latest version of the archetype once, you don't need to specify the `remoteRepository` any more.

Replace the `[CONFEXT:your group id]`, `[CONFEXT:your artifact id]` and `[CONFEXT:initial version]` with the values appropriate for your plugin. Eg. "com.foo.confluence", "bar-plugin", and 2.5-SNAPSHOT", respectively. The 'version' and 'packageName' parameters are optional - 'version' defaults to '1.0-SNAPSHOT' and 'packageName' defaults to the 'groupId'.

Once you execute this, it will create a new subdirectory in your current location, called whatever you put for the artifactId (eg. "bar-plugin"). Inside this directory you should find a directory named 'src' and a file named 'pom.xml'. Open the 'pom.xml' file and edit the properties therin appropriate to your plugin.



The sections marked 'TODO:' will need to be changed by you for the specific plugin project you're working on.

Some of the important sections you will need to check are:

- `<name>` - This should be customised to a human-readable name for the plugin.
- `<atlassian.plugin.key>` - This must be set to the unique key for the plugin. This will be automatically put into your 'atlassian-plugin.xml' file when the plugin is built.

Example

I want to create a new plugin called 'Demo Plugin'. All of my company's plugins are in the 'com.acme.confluence.plugins' group, but we put the code for each into its own sub-package. I enter the following at the command line:

```
mvn archetype:create -DarchetypeGroupId=com.atlassian.maven.archetypes  
-DarchetypeArtifactId=atlassian-plugin-archetype -DarchetypeVersion=2.3 \  
-DremoteRepositories=http://svn.atlassian.com/svn/public/contrib/repository/maven2 \  
-DgroupId=com.acme.confluence.plugins -DartifactId=demo-plugin  
-DpackageName=com.acme.confluence.plugins.demo
```

Confluence WARs

For the plugin toolkit to run integration tests a Confluence WAR dependency must be specified.

Confluence 2.2

Confluence 2.2 has not been fully converted to Maven 2. It is possible to build plugins against it, however it is not a fully-fledged WAR file so you cannot use the `mvn integration-test` or `atlassian-pdk:generateWar` with your projects.

To build against Confluence 2.2 your confluence dependency should look like this:

```

<dependency>
    <groupId>com.atlassian.confluence</groupId>
    <artifactId>confluence-core</artifactId>
    <version>2.2.9</version>
    <scope>provided</scope>
</dependency>

```

Currently 2.2.9 is currently the only version available to build against.

The main differences to note between 2.2 and 2.3 are that the artifactId is 'confluence-core' in 2.2 and 'confluence-webapp' in 2.3, and that the <type> is 'war' for 2.3.

Confluence 2.3

Below is a list of the Confluence WARs for 2.3. These must be put in the following form in your POM.

```

<dependency>
    <groupId>com.atlassian.confluence</groupId>
    <artifactId>confluence-webapp</artifactId>
    <version>${confluenceVersion}</version>
    <type>war</type>
    <scope>provided</scope>
</dependency>

<properties>
    <confluenceVersion>2.3-SNAPSHOT</confluenceVersion> <!-- or any other version -->
</properties>

```

The Confluence WARs available are:

- Confluence 2.3-SNAPSHOT



Maven 1 Wars

Confluence Maven 1 Wars will not work... so you can only test it against new versions of Confluence which now use Maven 2 (Post 2.3)

How Integration Tests work in Plugins

Since Integration tests in plugins seems complicated, this page has been setup to answer any questions and put any useful information you may require as a plugin developer.

Contents

- Confluence Plugin Parent
- FAQ

Confluence Plugin Parent

Here is the main section of the parent pom that is essential to make the integration tests work. (This is taken from version 1.1 of the confluence-plugins-base)

NOTE: it is recommended that you simply have this parent as:

```

<parent>
    <groupId>plugin.developer.utils</groupId>
    <artifactId>confluence-plugins-base</artifactId>
    <version>1.1</version>
</parent>

```

and dont forget to put in the repository so it know where to download it from. You should put this in `~/.m2/settings.xml`

```

<repositories>
    <repository>
        <id>atlassian-m2-repository</id>
        <name>Atlassian Maven 2.x Repository</name>
        <url>http://repository.atlassian.com/maven2</url>
        <snapshots>
            <enabled>true</enabled>
            <updatePolicy>interval:30</updatePolicy>
        </snapshots>
    </repository>
</repositories>

```

Here is the code in the parent pom if your interested, but please avoid using this manually in your pom unless you know what you're doing.
pom.xml - Integration Test section

```

<dependencies>
    <!-- Cargo dependencies -->
    <dependency>
        <groupId>tomcat</groupId>
        <artifactId>jasper-compiler</artifactId>
        <version>5.0.28</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>tomcat</groupId>
        <artifactId>jasper-runtime</artifactId>
        <version>5.0.28</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.mortbay.jetty</groupId>
        <artifactId>jetty</artifactId>
        <version>6.0.0beta15</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId> servlet-api</artifactId>
        <version>2.4</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>jwebunit</groupId>
        <artifactId>jwebunit</artifactId>
        <version>1.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>httpunit</groupId>
        <artifactId>httpunit</artifactId>
        <version>1.5.4</version>
        <scope>test</scope>
    </dependency>
</dependencies>
<build>
    <pluginManagement>
        <plugins>
            <plugin>

```

```

<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-surefire-plugin</artifactId>
<executions>
    <execution>
        <id>test</id>
        <phase>test</phase>
        <goals>
            <goal>test</goal>
        </goals>
        <configuration>
            <skip>false</skip>
            <excludes>
                <exclude>**/*AcceptanceTest.java</exclude>
            </excludes>
        </configuration>
    </execution>
    <execution>
        <id>Setup</id>
        <phase>integration-test</phase>
        <goals>
            <goal>test</goal>
        </goals>
        <configuration>
            <skip>false</skip>
            <includes>
                <include>it/com/atlassian/confluence/SetupInvokerAcceptanceTest.java</include>
            </includes>
        </configuration>
    </execution>
    <execution>
        <id>UploadPlugin</id>
        <phase>integration-test</phase>
        <goals>
            <goal>test</goal>
        </goals>
        <configuration>
            <skip>false</skip>
            <includes>
                <include>it/com/atlassian/confluence/RunUploadInvokerAcceptanceTest.java</include>
            </includes>
        </configuration>
    </execution>
    <execution>
        <id>RunTests</id>
        <phase>integration-test</phase>
        <goals>
            <goal>test</goal>
        </goals>
        <configuration>
            <skip>false</skip>
            <includes>
                <include>**/*AcceptanceTest.java</include>
            </includes>
            <excludes>
                <include>**/Abstract*.java</include>
            </excludes>
        </configuration>
    </execution>
<exclude>it/com/atlassian/confluence/SetupInvokerAcceptanceTest.java</exclude>
<exclude>it/com/atlassian/confluence/RunUploadInvokerAcceptanceTest.java</exclude>
</executions>
<configuration>
    <skip>true</skip>
    <systemProperties>
        <property>
            <name>acceptanceTestSettings</name>
            <value>${acceptanceTestSettings}</value>
        </property>
        <property>
            <name>pluginFile</name>
            <value>${project.build.finalName}.jar</value>
        </property>
    </systemProperties>
</configuration>

```

```

        </property>
        <property>
            <name>http.port</name>
            <value>${http.port}</value>
        </property>
        <property>
            <name>controller.port</name>
            <value>${controller.port}</value>
        </property>
    </systemProperties>
</configuration>
</plugin>
<plugin>
    <groupId>org.codehaus.cargo</groupId>
    <artifactId>cargo-maven2-plugin</artifactId>
    <version>0.3-SNAPSHOT</version>
    <configuration>
        <wait>false</wait>
        <container>
            <containerId>${cargo.container.containerid}</containerId>
            <zipUrlInstaller>
                <url>${cargo.container.zipurlinstaller.url}</url>
                <installDir>${installDir}</installDir>
            </zipUrlInstaller>
            <timeout>120000</timeout>
            <output>${project.build.directory}/output.log</output>
            <log>${project.build.directory}/cargo-log.log</log>
            <systemProperties>
<confluence.home>${project.build.directory}/confluence-home</confluence.home>
            </systemProperties>
        </container>
        <configuration>
            <home>${project.build.directory}/tomcat5x/container</home>
            <properties>
                <cargo.servlet.port>${http.port}</cargo.servlet.port>
                <cargo.rmi.port>${controller.port}</cargo.rmi.port>
                <cargo.jvmargs>-Djava.awt.headless=true -Xms256m
-Xmx512m</cargo.jvmargs>
            </properties>
        </configuration>
    </configuration>
    <executions>
        <execution>
            <id>start-container</id>
            <phase>pre-integration-test</phase>
            <goals>
                <goal>start</goal>
                <goal>deploy</goal>
            </goals>
            <configuration>
                <deployer>
                    <deployables>
                        <deployable>
                            <groupId>com.atlassian.confluence</groupId>
                            <artifactId>confluence-webapp</artifactId>
                            <type>war</type>
                            <location>${warInstallPath}</location>
                            <pingURL>${baseUrl}</pingURL>
                            <pingTimeout>240000</pingTimeout>
                            <properties>
                                <context>confluence</context>
                            </properties>
                        </deployable>
                    <deployables>
                </deployer>
            </configuration>
        </execution>
        <execution>
            <id>stop-container</id>
            <phase>post-integration-test</phase>
            <goals>
                <goal>stop</goal>
            </goals>
        </execution>
    </executions>
</plugin>

```

```

        </execution>
    </executions>
</plugin>
<plugin>
<groupId>com.atlassian.maven.plugins</groupId>
<artifactId>maven-properties-initialiser-plugin</artifactId>
<executions>
    <execution>
        <phase>validate</phase>
        <goals>
            <goal>setproperties</goal>
        </goals>
    </execution>
</executions>
</plugin>
</plugins>
</pluginManagement>
<plugins>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-antrun-plugin</artifactId>
    <executions>
        <execution>
            <phase>validate</phase>
            <id>delete</id>
            <goals>
                <goal>run</goal>
            </goals>
            <configuration>
                <tasks>
                    <echo>Deleting Confluence Configurations...</echo>
                    <delete dir="${basedir}/target/confluence-home" />
                </tasks>
            </configuration>
        </execution>
    </executions>
</plugin>
</plugins>
</build>
<repositories>
    <repository>
        <id>central</id>
        <name>Maven Repository Switchboard</name>
        <url>http://repol.maven.org/maven2</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </repository>
    <repository>
        <id>ibiblio</id>
        <name>ibiblio</name>
        <url>http://www.ibiblio.org/maven2</url>
    </repository>
    <repository>
        <id>atlassian-m2-repository</id>
        <name>Atlassian Maven 2.x Repository</name>
        <url>http://repository.atlassian.com/maven2</url>
        <snapshots>
            <enabled>true</enabled>
            <updatePolicy>interval:30</updatePolicy>
        </snapshots>
    </repository>
    <repository>
        <id>atlassian-ml-repository</id>
        <name>Atlassian Maven 1.x Repository</name>
        <url>http://repository.atlassian.com</url>
        <layout>legacy</layout>
        <snapshots>
            <enabled>true</enabled>

```

```

        </snapshots>
    </repository>
</repositories>

```

Please feel free to add Questions under this section.

FAQ

Legacy Plugin Archetype

| | |
|-------------------------|---|
| Name | Legacy Plugin Archetype for Maven2 |
| Version | 1.3 |
| Product Versions | All |
| Author(s) | David Peterson |
| Price | Free |
| License | BSD |
| IssueTracking | (soon to be implemented) |
| Download JAR | legacy-plugin-archetype-1.3.jar |
| Download Source | Subversion |

Description/Features

This is a Maven 2 archetype which assists in converting an existing plugin project to use Maven 2.

Usage

To use this archetype, open a command prompt and navigate to the directory where you would like to create the plugin project, and enter the following:

```

mvn archetype:create -DarchetypeGroupId=com.atlassian.maven.archetypes
-DarchetypeArtifactId=legacy-plugin-archetype -DarchetypeVersion=1.3 \
-DremoteRepositories=http://svn.atlassian.com/svn/public/contrib/repository/maven2 \
-DgroupId=[your group id] -DartifactId=[your artifact id] -Dversion=[initial version]
-DpackageName=[your package]

```



Once you've run the latest version of the archetype once, you don't need to specify the `remoteRepositories` parameter any more.

Replace the `[CONFEXT:your group id]`, `[CONFEXT:your artifact id]` and `[CONFEXT:initial version]` with the values appropriate for your plugin. Eg. "com.foo.confluence", "bar-plugin", and 2.5-SNAPSHOT", respectively. The 'version' and 'packageName' parameters are optional - 'version' defaults to '1.0-SNAPSHOT' and 'packageName' defaults to the 'groupId'.

Once you execute this, it will create a new subdirectory in your current location, called whatever you put for the artifactId (eg. "bar-plugin"). Inside this directory you should find a directory named '`src`' and a file named '`pom.xml`'. Open the '`pom.xml`' file and edit the properties therin appropriate to your plugin.



The sections marked 'TODO:' will need to be changed by you for the specific plugin project you're working on.

Some of the important sections you will need to check are:

- `<name>` - This should be customised to a human-readable name for the plugin.
- `<atlassian.plugin.key>` - This must be set to the unique key for the plugin. This will be automatically put into your 'atlassian-plugin.xml' file when the plugin is built.

Example

I have an existing plugin called 'Demo Plugin' which builds with Maven 1. It's checked out in the 'demo-plugin' subdirectory of my projects

directory. I want to upgrade it to use Maven 2 and the [Atlassian PDK](#).

1. Create the archetype

```
mvn archetype:create -DarchetypeGroupId=com.atlassian.maven.archetypes  
-DarchetypeArtifactId=legacy-plugin-archetype -DarchetypeVersion=1.3 \  
-DremoteRepositories=http://svn.atlassian.com/svn/public/contrib/repository/maven2 \  
-DgroupId=com.acme.confluence.plugins -DartifactId=demo-plugin  
-DpackageName=com.acme.confluence.plugins.demo
```

This will create two new files in my 'demo-plugin' directory:

1. `pom.xml` - This is a basic version of a Maven 2 project descriptor with the group ID, artifact ID and other standard settings set up for you. It is not complete, however, you will have to copy over some settings from your existing `project.xml` file.
2. `src/etc/atlassian-plugin.xml.m2` - This is a minimal version of the standard `atlassian-plugin.xml` file, with values which will retrieve the project settings such as the plugin version, name and description from the Maven `pom.xml` file. The advantage of this is you only have to update those details in one location. You can copy the example settings in this file into your real `atlassian-plugin.xml` file as desired. Otherwise, just delete the `.m2` file.

Once you've run it, modify your new `pom.xml` file. Search for the 'TODO' items in the file. Generally, however, the settings you will need to modify are:

- Project Name
- Developer information
- Dependencies

One thing to note about dependencies is that Maven 2 will recursively add dependencies for you. This can greatly reduce the number of dependencies you have to include in your own project. However, it won't hurt to just include all your old dependencies, even if they are automatically included.

Also, Maven 2 dependencies *must* declare their `<groupId>` and `<artifactId>`, as opposed to just supplying a single `<id>`. For older projects, the `groupId` and `artifactId` are often both the same value (the original `id`), but newer projects will have switched to a Java-package-like `groupId` (eg. `'org.springframework'` instead of `'spring'`). The best idea is to look in the public Maven repositories (eg. [ibiblio](#)) and see if you can find the current details for the dependency you're after.

Finally, if your project does not use the '/src/java' project structure, you will need to modify the `<build>` section to match your settings.

Plugin Development Kits

The Development Kits contains all the resources you need to get started developing a plugin.

| JIRA Plugin Dev Kit |
|---|
| • JIRA Plugin Development Kit |
| Confluence Plugin Dev Kit |
| • Confluence Plugin Development Kit |
| Bamboo Plugin Dev Kit |
| • Bamboo Plugin Development Kit |

Plugin Project Files

There are several files common to most plugin projects. Here is a summary of the areas of interest to plugin developers.

Files

- `build.properties`
- `project.properties`
- `project.xml`

build.properties

The `build.properties` file contains project properties specific to the specific developer's environment. It should **NOT** be checked into source control, or it will break other people's build environment. Also, properties which should apply to the project independent of the build environment should be put in `project.properties` instead.

As well as the properties listed below, you can override any of the properties specified in `project.properties`, so it may be worth checking those options too.

Below are some examples of common properties which may need to be set in your environment.

Maven Properties

Remote Repositories

This property allows you to specify remote repositories which Maven can download dependent libraries from. In general, this should be specified in `project.properties`, however there are some occasions where a repository isn't available (eg. it is a private repository inside an organisation) and outsiders will have to get the libraries from elsewhere. This should be avoided if possible.

```
maven.repo.remote=file:///lib,http://repository.atlassian.com,http://www.ibiblio.org/maven
```

Proxy Settings

If you are behind a firewall, you may need to specify how Maven can get through your proxy server. Only complete the ones which are required for your server.

```
## Specify these properties if you are behind a proxy server
maven.proxy.host=192.168.0.1
maven.proxy.port=8080
maven.proxy.username=xxx
maven.proxy.password=yyy
maven.proxy.ntlm.host=server
maven.proxy.ntlm.domain=domain
```

IntelliJ IDEA Properties

If you are using the 'atlassian-idea' plugin to build your project files, there are several custom properties available to make project setup simpler.

Jikes

You can specify that IDEA uses the open-source **Jikes** compiler if you wish.

```
atlassian.idea.jikespath=/path/to/jikes
```

JDK

Usually, however, you'll be using one of the standard JDKs set up in IDEA. Specify the name that IDEA has given the JDK here. You can find the name by opening IDEA and selecting 'File > Settings', then 'JDK & Global Libraries'. The value you need is in the 'Name' field of the JDK.

```
atlassian.idea.jdk.name=1.4
```

Source Code

If you have access to the Confluence or JIRA source code, it can be handy to have your linked libraries referencing that source code when debugging. You can specify a basic set of source code links by pointing the plugin at the root of the downloaded source distribution. Debugging will still work fine without it, but it will get lost once the trace leaves your plugin code.

```
atlassian.idea.src.relative.location=../confluence-2.1-src
```

Global Libraries

Sometimes you will need to specify extra global libraries that need to be linked to by your project. Often these are items like JDBC drivers, and are more often required when setting up Confluence or JIRA as a whole, rather than a plugin project. However, this is where you do it.

```
atlassian.idea.global.libraries.to.enable=mysql
```

Application Server Settings

These properties will only be relevant in the project files for Confluence or JIRA, not plugin projects. They point to your app server installation(s). Currently supported are Tomcat 4 & 5, Resin 2 & 3, and Orion.

Only include/uncomment the ones you actually have installed.

```
atlassian.idea.tomcat.location=~/apache-tomcat-5.5.12  
atlassian.idea.resin3.location=~/resin3  
atlassian.idea.resin.location=~/resin2  
atlassian.idea.orion.location=~/orion
```

Application Server Settings

Once you've picked your application server, the following wil specify how the web application is launched.

The settings below will allow you to access the running server at <http://localhost:8080/confluence>:

```
# port on which Confluence will listen  
atlassian.idea.application.port=8080  
# the context root at which Confluence will be deployed  
atlassian.idea.application.webapp.contextpath=confluence
```

WAR Settings

Again, these are not relevant for plugin projects and in fact **should not be present**. These should already be set for you in later versions of Confluence and JIRA, but it's good to check them.

```
maven.multiproject.type=war  
maven.war.webxml=src/webapp/WEB-INF/web.xml  
maven.war.webapp.dir=target/exploded
```

Database Settings

If you are using an external database, set this up as documentation dictates. Confluence and JIRA build.properties files have some instructions for common database engines which should be fairly easy to adapt to your environment.

project.properties

This file contains properties relating to the project which do not change in a given development environment.

project.xml

The project.xml file describes the project structure to Maven, which is used to build and deploy the plugin. It can also be used to create project files for IntelliJ IDEA and Eclipse.

File Contents

1. **<extend>** - This determines most of the dependencies for your plugin and should be set to the oldest version of Confluence your plugin will support. There are often significant changes between different versions of Confluence, so it's probably best to stick with the most recent version initially. This value should match one of the 'confluence-x.x.xml' files in the project directory. Start with something like this:

```
<extend>confluence-2.1.3.xml</extend>
```

2. **<id>** - The name of the plugin. Since this will turn into file names, it's best to stick with alphanumerics and '-' for the name (eg. 'my-plugin' rather than 'My Most Excellent Plugin').
3. **<name>** - This is your human-readable name for the plugin. 'My Most Excellent Plugin!' will work fine here.
4. **<currentVersion>** - The current version of the project. This will be updated each time you do a major release. You will also need to update the number in the `atlassian-plugin.xml` file at the same time (more on that later).

```

<id>my-plugin</id>
<name>My Most Excellent Plugin</name>
<currentVersion>1.0</currentVersion>

```

5. <organization> - Details about your organisation go here.

```

<organization>
  <name>My Organization</name>
  <url>http://www.myorganization.com/</url>
</organization>

```

6. <package> - The base package for your project. This is used when generating the JavaDocs for the project, so make it the lowest common package name. Eg, if you have classes in 'com.myorg.confluence' and 'com.myorg.utilities' in the plugin, then make this value 'com.myorg'.

```

<package>com.myorg.confluence.plugin</package>

```

7. <description> - Just a regular, human-readable description of the project.

8. <dependencies> - This is where you can specify extra dependencies specific to your plugin. Don't bother listing libraries which are already in the standard Confluence build - they're already taken care of by the <extend> tag above. If your extra library is not in one of the standard Maven repositories, you will have to do some extra work here to get it going, which will, for the moment, be left as an exercise for the reader.

Also, this example uses <id>, which is not Maven best-practice. The overall guide is being updated to Maven 2, and this will be addressed during that process. Stay tuned.

```

<dependencies>
  <dependency>
    <id>extra-library</id>
    <version>1.2-dr3</version>
  </dependency>
</dependencies>

```

9. <build> - There isn't too much in this section that you will need to change, other than the <nagEmailAddress>.

```

<build>
  <nagEmailAddress>me@myorg.com</nagEmailAddress>
  <!-- The rest of the settings stay the same... -->
</build>

```

Plugin Self-Configuration



Plugin configuration is available in Confluence 2.2 and later. Versions of Confluence before 2.1 will simply ignore any of these parameters

Plugins can specify internal links within Confluence to configure themselves. This is useful where your plugin requires any configuration or user specific settings to work. For example, the [Google Maps plugin](#) requires a Google API Key from Google (which needs to be configured on each server) before it will work properly.

- Configuration links will *most often* point to [XWork plugin modules](#) within the plugin itself
- Configuration links can be provided for a whole plugin and/or for any module within a plugin
- Configuration links are relative to the Confluence application

Plugin configuration - to add a configuration link for the whole plugin, place a single `param` element with the name `configure.url` within the `plugin-info` element at the top of the plugin descriptor:

```

<plugin-info>
  <description>A macro which displays Google maps within a Confluence page.</description>
  <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com/" />
  <version>0.1</version>
  <param name="configure.url">/admin/plugins/gmaps/configurePlugin.action</param>
</plugin-info>

```

Plugin module configuration - to add a configuration link for a single module, place the same `param` element with the name

`configure.url` within the `descriptor` element for that module:

```
<macro name="gmap" class="com.atlassian.confluence.ext.gmaps.GmapsMacro" key="gmap">
    <description>The individual map macro.</description>
    <param name="configure.url"/>/admin/plugins/gmaps/configureMacro.action</param>
</macro>
```

Here is an image showing where the `Configure` links appear for both a plugin and an individual module:

The screenshot shows the Atlassian Plugins page. On the left, there's a sidebar titled "Plugins" with a list of modules: Advanced Macros, Attachment Extractors, Basic Macros, Compatibility Macros, Dashboard Macros, and Google Maps. The "Google Maps" entry is highlighted. To the right, there's a detailed view for the Google Maps plugin. It shows the vendor (Atlassian Software Systems Pty Ltd) and plugin version (0.1). A description states: "A macro which displays Google maps within a Confluence page." Below this, there are two checkboxes: "Configure plugin" and "Disable plugin". Under the "Configure plugin" checkbox, there are three entries: "gmap" (with a description "The individual map macro."), "Configure GMaps", and "gmapsManager". Each entry has a "Configure" and a "Disable" link next to it.

NavigationLinks

[Developer Blog](#)
[Forums / Mailing Lists](#)

TreeNavigation

Welcome to the Atlassian Developer Network

Welcome!

These pages will teach you everything you need to know in order to extend and customize Atlassian products. (If you're interested in finding plugins to use in your Atlassian tools, you should check out our plugin library on <http://plugins.atlassian.com>.)

Getting Started

We've organized our documentation to answer the questions posed by developers at different experience levels.

"I'm totally new to this sort of thing."

You've never programmed before, or it was long enough ago that you've forgotten everything useful. We'll show you how to get up to speed.

"I've done some programming before, and I know enough Java (or JavaScript) to get in trouble with it."

Code doesn't scare you, and you've worked on Java long enough to have a good feel for it. We'll get you going with the [Atlassian Plugin SDK](#).

"I've got the SDK installed and I'm ready to start on a plugin."

You want to get started on something specific. We'll teach you what you need to know about programming in the Atlassian world and point you in the right direction.

"I'm writing a plugin and I'm stuck."

You're in the middle of writing a plugin, and you've hit the wall. We'll try to get you back on the road.

Product information

Here you'll find direct access to the development hubs maintained by our product teams.

| | | |
|---|--|--|
|  JIRA |  Confluence |  Bamboo |
|  FishEye |  Crucible |  Crowd |

FAQ

We've organized our [FAQ](#) into sections depending on your experience level to help you find what you need.

Getting help

Help is never far away.

IRC

If you have a quick question or comment, you can usually find us on IRC.

Server: [irc.freenode.net](irc://irc.freenode.net)
Channel: [#atlassiandev](#)

[Clicking this link](#) will launch an IRC session in your web browser.

(A bunch of Atlassians and former Atlassians also hang out in [#atlassian](#), if you're looking to get to know us better.)

Atlassian Answers

To get the widest possible audience for your question, post in the [Atlassian forums](#).

Contact Developer Relations

For documentation questions, suggestions, and project hosting, email us at developer-relations@atlassian.com.

Atlassian Plugin SDK

Get the latest version here!



Atlassian Plugin Exchange

Looking for a particular plugin? You can find it on the [Atlassian Plugin Exchange](#).

Search the Atlassian Plugin Exchange



All Products JIRA Confluence Fis
hEye Bamboo Crowd Crucible

Search

Developer Relations on Twitter

Atlassian Developer Blog

Atlassian Developer Blog

RSS

Plugin Driven Development- Developing JIRA's new Project Configuration pages

One of the biggest pain points we hear from customers is how complex configuring projects in JIRA is. There are so many buttons and dials available to push in the administration screens...

The road to HAMS 3.0 - Transaction boundaries

HAMS is Atlassian's order processing system; if you've ever bought an Atlassian product it's HAMS that's been doing the work in the back-end. HAMS has served us well, but is over...

Make your plugin sizzle with new Plugin Exchange enhancements

Plugin Devs! We recently released version 1.3 of the Atlassian Plugin Exchange and I wanted to highlight a few improvements we've made. First, we want to make sure that you can...

Testing's In Session

At Atlassian, we use test sessions to manually test our products. If you're a tester and haven't heard of test sessions before, you might be surprised to learn that you're probably using...

AtlasCamp 2011

Good morning, Campers! AtlasCamp 2011 registration is now open! AtlasCamp is one of the highlights of the Atlassian Calendar, our once a year, developers-only extravaganza. If you're interesting extending the Atlassian...

How to learn to program (in Java or JavaScript)

Prerequisites

This page is the starting point for aspiring developers who want to someday build an Atlassian plugin or gadget.

This page is for you if:

- You know programming language(s) that aren't Java or JavaScript

OR if:

- You aren't already familiar with any programming language
- You have a very logical and persistent attitude
- You are prepared to spend months studying a difficult, arbitrary space of knowledge

Customizing without programming

Atlassian products are flexible and powerful, and many common customizations are available to you without requiring you to write a single line of code. See page *Product Customizations Without Code* for examples of what you can do.

Learning Java

The Atlassian products are written in the Java™ programming language, originally a product of Sun Microsystems and now owned by Oracle Corporation. Java is an object-oriented language, which means that it models program code as a set of self-contained chunks that communicate with each other. Most hip modern languages, like C++, Ruby, Python, JavaScript, and Objective-C, use this same paradigm.

Why learn Java?

In general, Java is a good choice for study if:

- You want to learn how object-oriented design works
- You prefer (or are required) to program for the server instead of the client
- You want to get involved in the extensive open source community around Java

For Atlassian applications, Java can help you:

- Understand exactly how the application behaves (since Atlassian [provides the source code to paying customers](#), you can study the product in the most minute detail)
- Write plugins to add awesome functionality for yourself, your business, or other Atlassian users

Where should I go to learn Java?

There are a wealth of online tutorials and dead-tree books for learning Java.

- [The Java Tutorial](#) – Oracle's online Java tutorial, started in 1995 and kept up to date as the language evolves. Thousands of programmers have learned Java here, including several Atlassians.
- [The Java Tutorial: A Short Course on the Basics, 4th Edition](#) – The online tutorial in book form.
- [Head First Java, 2nd Edition](#) – A well-respected book introducing new developers to Java.

Learning JavaScript

Atlassian Gadgets are written in a combination of XML, a document formatting standard, and JavaScript™, the language of the web browser. JavaScript is a different programming language from Java. JavaScript code is downloaded from the web server and run inside the browser, interacting both with you and the page itself. JavaScript code is used for many websites that offer rich interfaces to the user, such as [Gmail](#), [Facebook](#), and [YouTube](#).

Why learn JavaScript?

In general, JavaScript is a good choice for study if:

- You want to write "thick client" Web applications or richly-featured interactive web pages
- You prefer (or are required) to program for the client instead of the server
- You enjoy a high degree of flexibility and style in your work

For Atlassian applications, JavaScript can help you:

- Write Gadgets – little programs that run in all Atlassian products
- Customize the look and feel of the application

Where should I go to learn JavaScript?

As with Java, there are a wealth of online tutorials and dead-tree books.

- [Mozilla Developer Network JavaScript Guide](#) – A comprehensive online guide to the features of JavaScript, as well as an introduction to the Document Object Model (DOM).
- [JavaScript: The Good Parts](#) – A slim, fast-reading book on core JavaScript features and good programming practices, but harder to understand for people who've never done programming.

Programming is hard

The most effective way to learn something is to practice with it at least a little bit every day. Learning programming is difficult but very rewarding. There will be lots of things that don't make sense at first. Study examples wherever you can. Experiment. Try not to get frustrated. Don't expect to become an expert in a week, a month, or even a year. Good programmers never stop learning.

Good luck!

Installing the Atlassian Plugin SDK

- Prerequisites
- [What to learn here](#)
- Setting up with the Atlassian Plugin SDK
 - [Download Latest Version \(Windows\)](#)
 - [Download Latest Version \(Mac OS X, UNIX, Linux\)](#)
- Advanced Maven configuration

Prerequisites

This page is the starting point for Java developers who are new to working with Atlassian plugins and products.

This page is for you if:

- You are familiar with and competent at Java or JavaScript programming
- You are comfortable using your machine's command line or terminal window

If you don't feel ready to proceed, please check our [New Developers page](#) for tips on how to start.

What to learn here

On this page you'll learn the following:

- How to install the Atlassian Plugin SDK
- How to use the SDK commands to create a plugin skeleton
- How to start the host product using the SDK

Setting up with the Atlassian Plugin SDK

1

Set up: Start here to set up your development environment, including Java and the Atlassian Plugin SDK.



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

[Download Latest Version \(Windows\)](#)



[Download Latest Version \(Mac OS X, UNIX, Linux\)](#)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.



Current released version – Atlassian Plugin SDK 3.4

Currently-supported applications: **Confluence, JIRA, Bamboo, FishEye, Crucible and Crowd.**

Atlassian Plugin SDK 3.4 is now available – see the [release notes](#). If you have ideas for improvement or new features, or have found a bug, please raise a ticket on our [issue tracker](#). [Snapshot builds](#) are also available for the stout-hearted.

Advanced Maven configuration

Need super-custom access to Atlassian's Maven repositories? Find details [here](#).

Setting up your Plugin Development Environment



Current released version – Atlassian Plugin SDK 3.4

Currently-supported applications: **Confluence**, **JIRA**, **Bamboo**, **FishEye**, **Crucible** and **Crowd**.

Atlassian Plugin SDK 3.4 is now available – see the [release notes](#). If you have ideas for improvement or new features, or have found a bug, please raise a ticket on our issue tracker. **Snapshot builds** are also available for the stout-hearted.



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

Download Latest Version (Windows)



Download Latest Version (Mac OS X, UNIX, Linux)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

This page tells you how to set up your development environment. You will need to follow these setup instructions only once. If you have already done everything on this page, go to the page about [developing your plugin](#).

On this page:

- Prerequisites, product keys and default ports
- Step 1. Install Java
- Step 2. Install the Atlassian Plugin SDK
- Step 3. Install your IDE
 - Instructions for specific IDEs
- Next Step. Develop your First Plugin

Prerequisites, product keys and default ports

The table below shows the applications currently supported by the Atlassian Plugin SDK, the minimum version of the application required, and the default port for each application.

| Atlassian Application | Version | Default Port | Product Key | Caveats |
|-----------------------|---------|--------------|-------------|---|
| Bamboo | 2.3 + | 6990 | bamboo | |
| Confluence | 2.10 + | 1990 | confluence | |
| Crowd | 2.0.2 + | 4990 | crowd | |
| Crucible | 2.0.6 + | 3990 | fecru | |
| FishEye | 2.0.6 + | 3990 | fecru | |
| JIRA | 4.0 + | 2990 | jira | Plugins developed for versions of JIRA before 4.0 are supported, but using the SDK with versions of JIRA earlier than 4.0 is not. For developing plugins for JIRA 3.13 and earlier, take a look at the old JIRA PDK . |
| RefApp | 1.0 + | 5990 | refapp | |

Step 1. Install Java

You will need a [Java](#) development kit. Atlassian applications support **JDK 6**.

1. Download and install [JDK 6](#).
2. Set your `JAVA_HOME` environment variable. (See [our instructions](#)).
3. Read [how to specify a version of Java](#), to learn how to use Java version-specific language features or enforce building with a particular version of Java. Failure to follow the proper procedure for choosing a Java version can produce cryptic errors, especially with OSX.

Step 2. Install the Atlassian Plugin SDK

The Atlassian Plugin SDK includes [Maven](#) and a correctly-configured Maven `settings.xml` file, as well as a number of shell scripts to speed up and simplify plugin development.

1. Download the [Atlassian Plugin SDK](#) and unzip it into a folder of your choice. (Latest version is linked to the 'Download' button above.)
2. Add the SDK's `bin` directory to your path environment variable.

Windows

For example, if you unzipped the SDK into this location:

`C:\Atlassian\atlassian-plugin-sdk-3.3`

Then you should add the following to your path:

`C:\Atlassian\atlassian-plugin-sdk-3.3\bin`

NB: On Windows, any applications started after you have set the PATH will get the new path. Open a new command window to see the new path.

Unix

For example, if you unzipped the SDK into the following location:

`/Users/example/bin/atlassian-plugin-sdk-3.3`

Then you should add the following to your path:

`PATH=/Users/example/bin/atlassian-plugin-sdk-3.3/bin:$PATH`

OS X

For example, if you unzipped the SDK into the following location:

`/Users/example/bin/atlassian-plugin-sdk-3.3`

Next, create a file called `/etc/paths.d/atlassian`. You must create this file as `root` for it to work:

`sudo touch /etc/paths.d/atlassian`

Adding the following to the file:

`/Users/example/bin/atlassian-plugin-sdk-3.3/bin`

Save the file and your path is set.



If for some reason you have decided not to use the Atlassian Plugin SDK, you will need to install and configure Maven yourself, as described in the page on [Maven requirements](#).



If you need to access internet resources through an HTTP proxy, you will need to configure a [proxy](#) in the `apache-maven/conf/settings.xml` file after you have extracted the archive.

Step 3. Install your IDE

Decide on your code development tool, if you do not already have one. We recommend [Eclipse](#), [IntelliJ IDEA](#) or [NetBeans](#), because these all support Maven 2.

Other IDEs: The Atlassian plugin development process depends heavily on [Maven](#), so it will be easier if your IDE has Maven 2 support. The IDE must allow you to open a POM file.

Instructions for specific IDEs

- Configuring Eclipse to use the SDK

- Configuring IDEA to use the SDK
- Configuring NetBeans to use the SDK

Next Step. Develop your First Plugin

Now you can use the handy Atlassian archetypes (templates) to [create your first plugin](#).

RELATED TOPICS

Welcome to the Atlassian Developer Network

- How to learn to program (in Java or JavaScript)
- Installing the Atlassian Plugin SDK
- Writing your first plugin
- Advanced Plugin Development
- Atlassian Plugin SDK Documentation
- Other Information
- FAQ
- Tutorials
- Getting Involved in the Atlassian Developer Network
- Codegeist V
- Developer Relations State of the Union 2011 - Resources

Colour-Coding your Maven Output

 For **UNIX**, **Linux** and **OS X** only.

If you are using the [Atlassian Plugin SDK](#) and are running a UNIX-based operating system, you can ensure that your Maven output comes in true colour.

Just add an environment variable with the name `MAVEN_COLOR` and value `true`.

```
export MAVEN_COLOR=true
```

Your Maven output will look something like this:

```
~/dev $ mvn
Colorizing console...
[INFO] Scanning for projects...
[INFO] -----
[ERROR] BUILD FAILURE
[INFO] -----
[INFO]
```

You must specify at least one goal or lifecycle phase to perform build steps.
The following list illustrates some commonly used build commands:

```
mvn clean
    Deletes any build output (e.g. class files or JARs).
mvn test
    Runs the unit tests for the project.
mvn install
    Copies the project artifacts into your local repository.
mvn deploy
    Copies the project artifacts into the remote repository.
mvn site
    Creates project documentation (e.g. reports or Javadoc).
```

Please see
<http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>
for a complete description of available lifecycle phases.

Use "mvn --help" to show general usage information about Maven's command line.

```
[INFO] -----
[INFO] For more information, run Maven with the -e switch
[INFO] -----
[INFO] Total time: < 1 second
[INFO] Finished at: Mon Aug 31 16:41:42 EST 2009
[INFO] Final Memory: 1M/3M
[INFO] -----
```

RELATED TOPICS

[Developing your Plugin using the Atlassian Plugin SDK](#)

Configuring Eclipse to use the SDK

This page explains how to create an Eclipse project from the Atlassian Plugin SDK.



M2Eclipse is not supported

Modern versions of Sonatype's [M2Eclipse plugin](#) will not work correctly with the Atlassian Plugin SDK. We will fix this, but for now the instructions below will let you work.

- Prerequisites
- Setting up the project for the first time
- Picking up updates
- Further reading

Prerequisites

- Have Eclipse Helios installed and configured.
- Create or check out your plugin project (see [here](#) for instructions on starting from scratch).

Setting up the project for the first time

- Open a terminal window.

- Change to your plugin directory.
- Run the command:

```
atlas-mvn eclipse:add-maven-repo -Declipse.workspace=<path to your Eclipse workspace>
```

This command adds the classpath variable M2_REPO to Eclipse. The variable allows Eclipse to load dependencies from your local Maven repository. If you prefer, you can add this variable yourself through your Eclipse preferences.

- Run the command:

```
atlas-mvn eclipse:eclipse
```

This command creates an Eclipse project metadata file (.classpath and .project). During the execution, the SDK's embedded Maven will download all required dependencies, along with source code and Javadoc artifacts, and put them in the project metadata so that Eclipse can find them.



Product source code

The source code for the product itself is still only available through the my.atlassian.com account where you purchased the product. See [this page](#) for instructions on using your product source code with the SDK.

- Start Eclipse.
- Select **File->Import**.
- Expand the **General** folder tree and select **Existing Projects into Workspace**.
- Select the folder your plugin project lives in.

The project will now be imported.

Picking up updates

If you change your POM, Eclipse will not be automatically aware of the changes and may be unable to rebuild your project or locate dependency files. Whenever you change the pom.xml, run the command:

```
atlas-mvn eclipse:eclipse
```

This will update the on-disk project metadata with the new POM information. Refresh the project in Eclipse to effect the changes.

You should be ready to go!

Further reading

Complete information on the capabilities of the Maven Eclipse plugin are available [here](#).

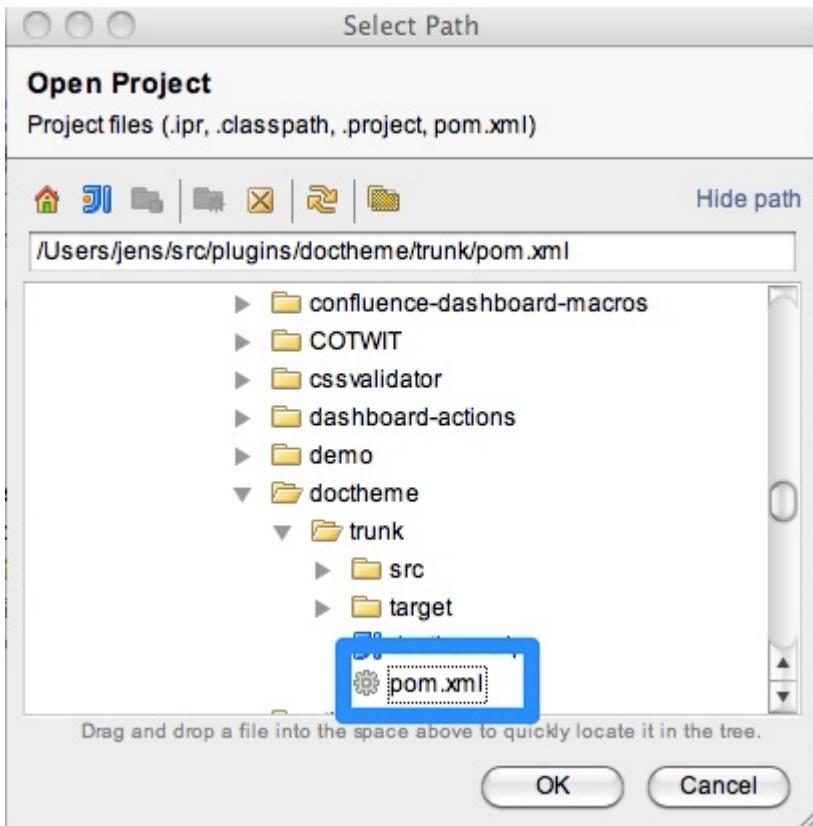
Configuring IDEA to use the SDK

Opening SDK projects in IDEA

As of version 9.0, [IDEA integrates Maven](#) out of the box by importing the entire IntelliJ IDEA project from pom.xml files. This makes setting up the plugin in IDEA a breeze. Simply create a plugin project using the atlas-create-*-plugin (eg. [atlas-create-confluence-plugin](#)) command and load the project's pom file into IntelliJ.

To load the plugin project in IDEA:

- Click **Open Project** in the **File** menu
- Browse to your plugins directory
- Select the **pom.xml** file and click **ok**
- The project will be loaded



Increasing the amount of memory available to IDEA

You may find it effective to increase the default amount of memory allocated for IDEA. You can do this by modifying the VM options located in:

for Windows

```
<Install-dir>\bin\idea.exe.vmoptions
```

for Unix

```
<Install-dir>\bin\idea.vmoptions
```

for Mac OS X

```
/Applications/IntelliJ\ IDEA\ 9.0.4.app/Contents/Info.plist
```

For example, if you installed the application in its default location on Windows you may see something like this:

```
C:\Program Files\JetBrains\IntelliJ IDEA 9.0.4\bin\idea.exe.vmoptions
```

Below is an example of settings:

```
-Xms32m  
-Xmx256m  
-XX:MaxPermSize=128m  
-ea
```

Configuring NetBeans to use the SDK

Partial integration between [NetBeans](#) and the Atlassian Plugin SDK can be achieved. The NetBeans IDE includes native support for Maven projects. The Atlassian Plugin SDK includes a bundled version of Maven.

An existing NetBeans user is unlikely to want to switch to the instance of Maven that comes with the Atlassian SDK.

Step 1 - Setup the environment so that atlas Maven commands can be invoked from the command line. See: [Step 2. Install the Atlassian Plugin SDK](#)

Step 2 - Create an empty plugin project using "atlas-create-confluence-plugin".

Step 3 - Modify the pom.xml

In order for NetBeans to pick up the necessary files to recognize this correctly you need to add the following to the project's **pom.xml**

```
<repositories>
    <repository>
        <id>atlassian</id>
        <name>Atlassian Repository</name>
        <url>https://maven.atlassian.com/content/groups/public/</url>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
        <releases>
            <enabled>true</enabled>
        </releases>
    </repository>
</repositories>

<pluginRepositories>
    <pluginRepository>
        <id>atlassian-public</id>
        <url>https://m2proxy.atlassian.com/repository/public</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </pluginRepository>
</pluginRepositories>
```

and to the properties section add a reference to your locally installed Atlassian Maven repository, so that your pom.xml contains something like:

```
<properties>

<maven.local.repo>C:\Atlassian\atlassian-plugin-sdk-3.2\repository</maven.local.repo>
    <confluence.version>3.3</confluence.version>
    <confluence.data.version>3.1</confluence.data.version>
</properties>
```

After this Netbeans should recognize the plugin and be able to perform all the usual dependency maintenance and code editing as per usual. You just need to remember to do any actual compilation and packaging using the atlas-mvn command line tools.

Creating a remote debug target

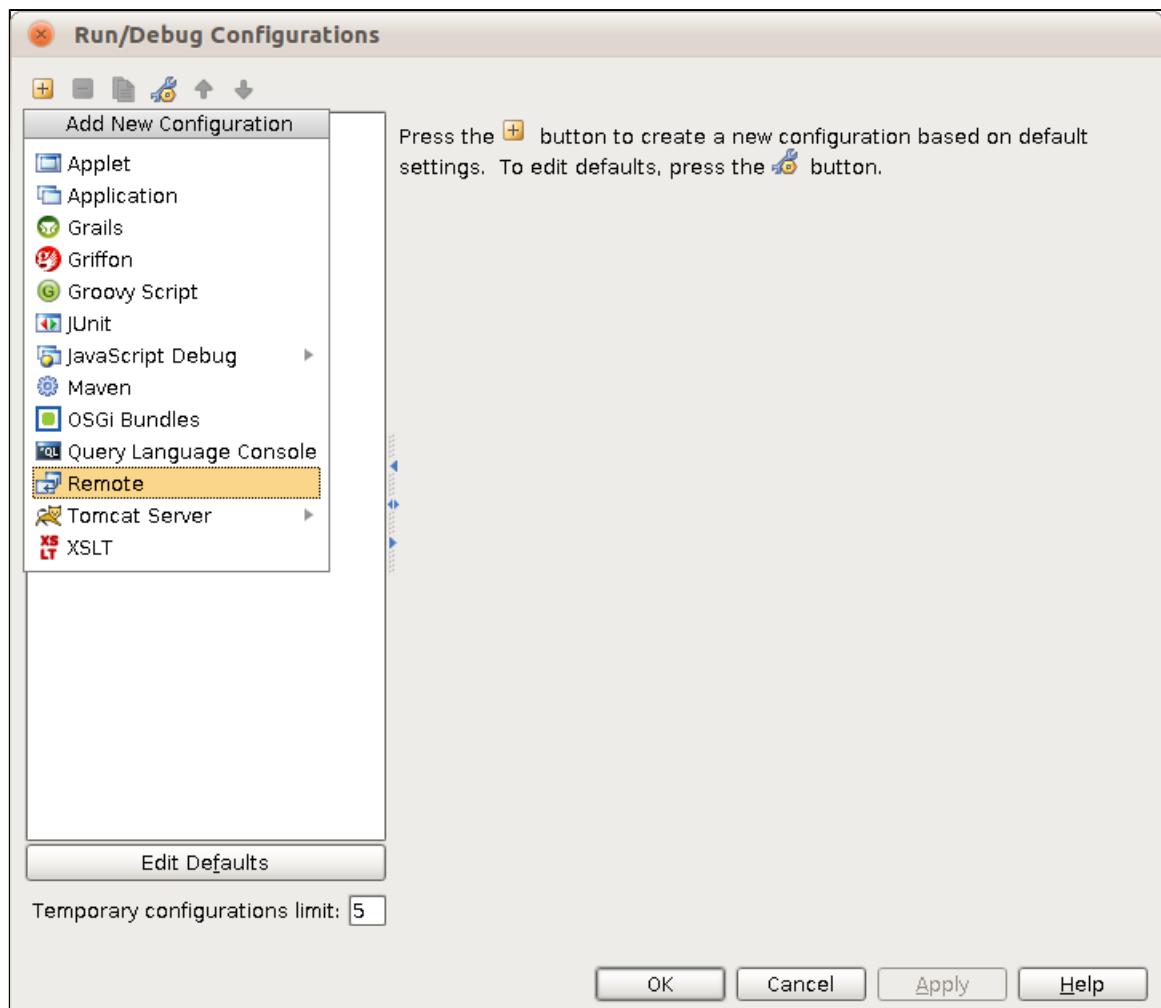
Basics

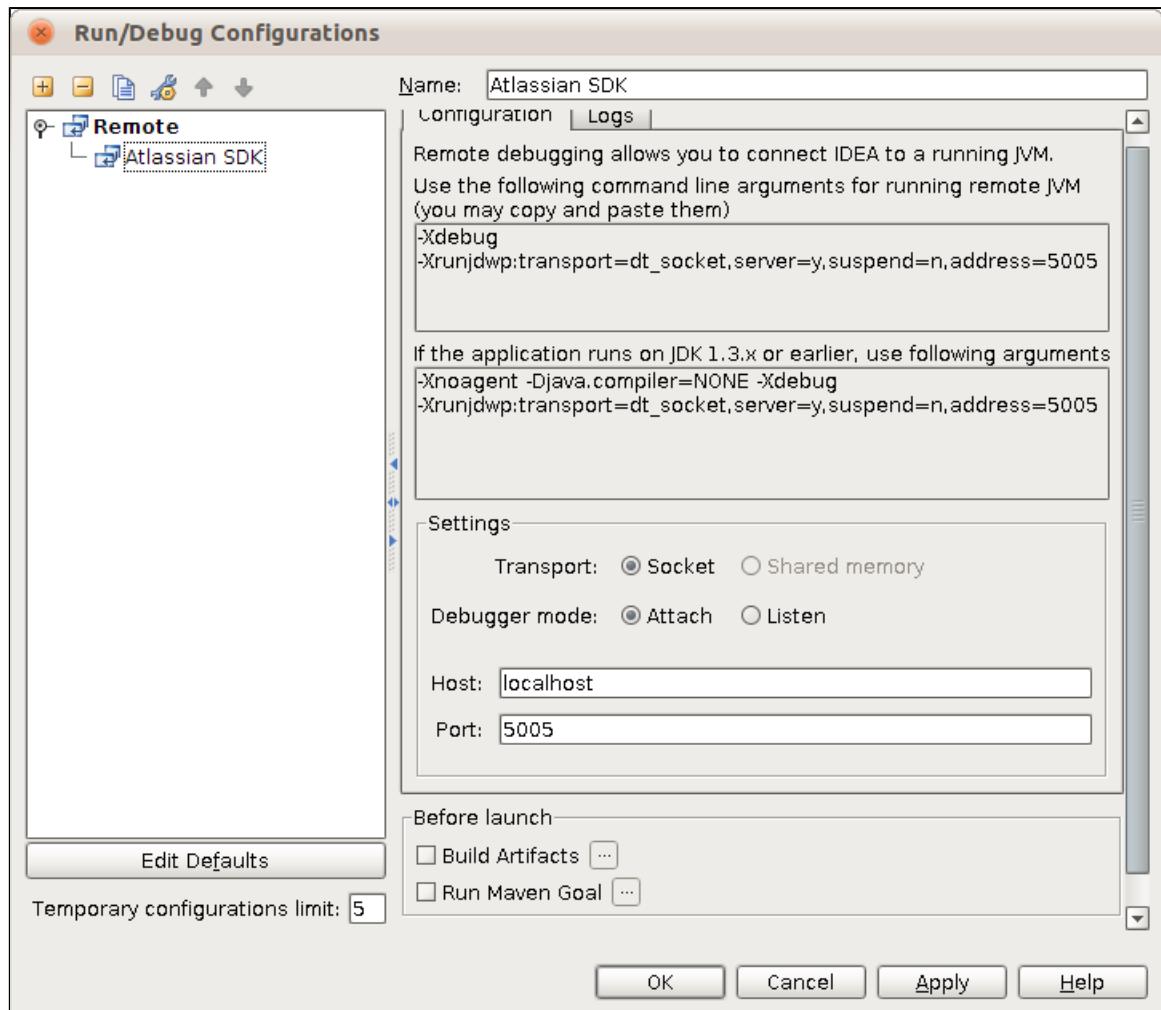
Since your plugin is deployed as part of another application you can't just start your IDE in debug mode to debug the plugin. You have to make your IDE connect to the (Confluence) server that runs your plugin. In order to do this, the server needs to open up a port so your IDE can hook into it. Fortunately, most IDE's tell you what parameters you need.

For more details on what "remote debugging" is and how it works with your particular tools, a quick internet search will help. Below are two examples for IDEA and Eclipse.

Creating a remote debug target in IDEA

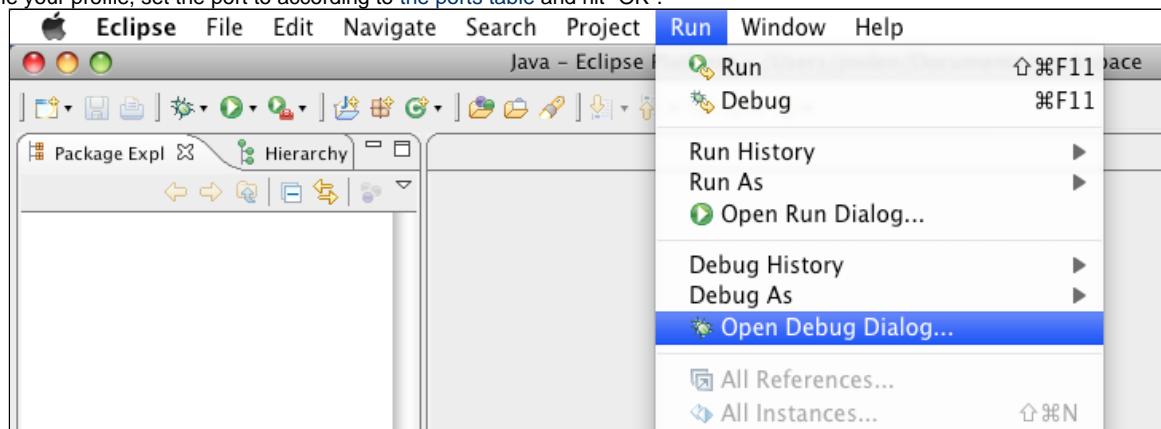
1. Go to Run > Edit Configurations.
2. Click the plus icon and choose Remote.
3. As you can see in the screenshot below, IDEA creates a default set of parameters to attach to the debugger. These are the same ones the SDK uses by default, so you don't need to change these.
4. Name your profile and hit "OK".
5. Go to Run > Debug to connect to begin debugging.



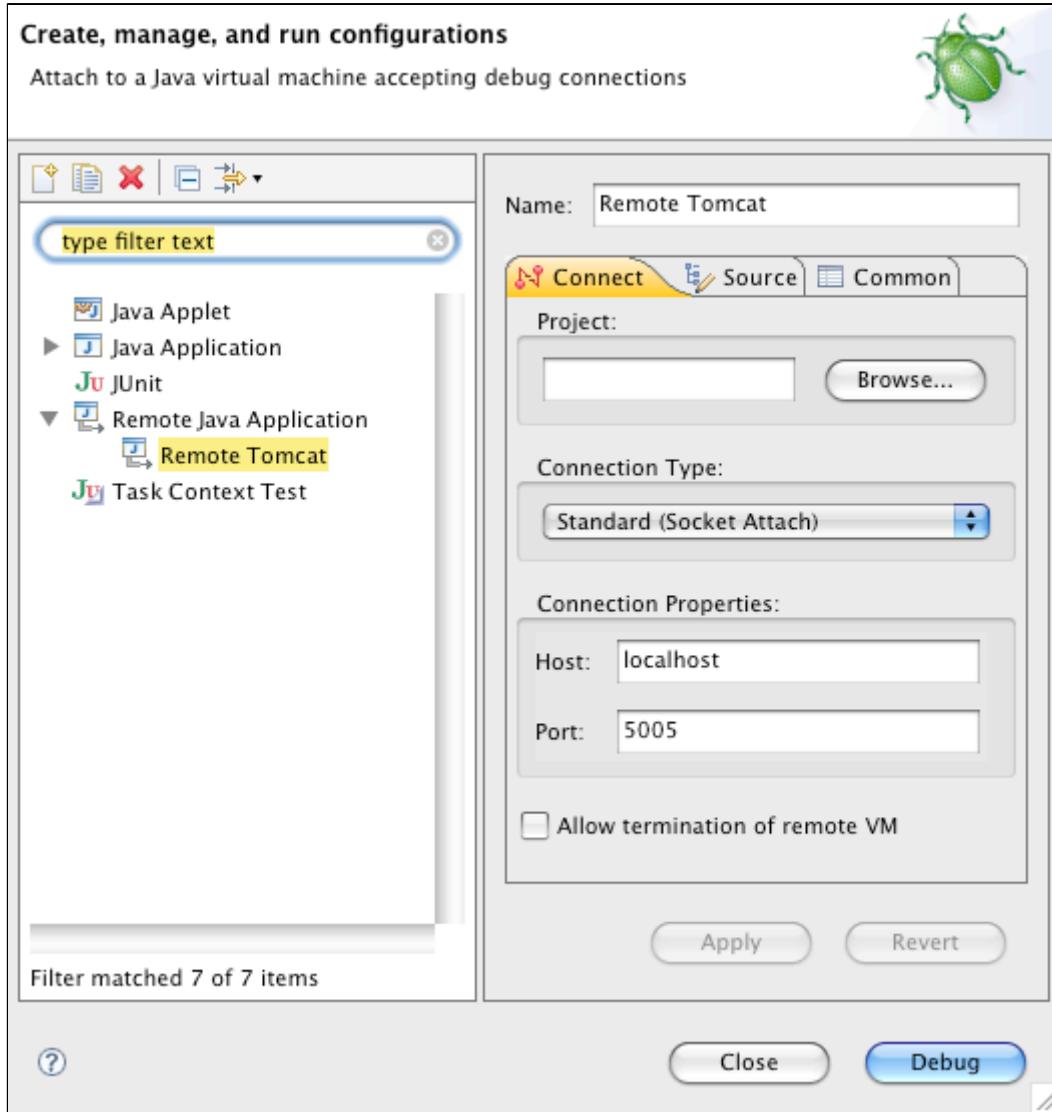


Creating a remote debug target in Eclipse

1. Go to Run > Open Debug Dialog.....
2. Click the Remote Java Application.
3. Name your profile, set the port to according to the ports table and hit "OK".



4. Make sure to apply the correct JVM settings to your Confluence server so it will accept the debugging session



Ports used by applications

All applications started by the SDK with `atlas-debug` will open a debugger on port 5005, which is the default for Java debugging. Since IDEA and Eclipse both default to port 5005, no further configuration should be necessary.

Setting JAVA_HOME

Once you have installed the JDK (see [Setting up your Plugin Development Environment](#)), you need to set the `JAVA_HOME` environment variable.

Setting JAVA_HOME on Windows

1. Right-click the 'My Computer' icon on your desktop and select 'Properties'.
2. Click the 'Advanced' tab.
3. Click the 'Environment Variables' button.
4. Click 'New'.
5. In the 'Variable name' field, enter 'JAVA_HOME'.
6. In the 'Variable value' field, enter the directory (including its full path) where you installed the JDK.
7. Restart the computer.

Setting JAVA_HOME on UNIX and Mac OS X

There are many ways you can set the `JAVA_HOME` environment variable on 'nix based systems (including Mac OS X). Here are two:

For your current user,

1. Open up a shell / terminal window
2. `vi ~/.profile` (replace vi with your favourite text editor)
3. Add `export JAVA_HOME=/path/to/java/home/dir` on its own line at the end of the file
4. Add `export PATH=$JAVA_HOME/bin:$PATH` on its own line immediately after

5. Save, and restart your shell
6. Running `java -version` should give you the desired results

For all users in the system,

1. Open up a shell / terminal window
2. `vi /etc/profile` (replace vi with your favourite text editor)
3. Add `export JAVA_HOME=/path/to/java/home/dir` on its own line at the end of the file
4. Add `export PATH=$JAVA_HOME/bin:$PATH` on its own line immediately after
5. Save, and restart your shell
6. Running `java -version` should give you the desired results

If you are using a GUI, you may not need to open up the shell. Instead, you might be able to open the file directly in a graphical text editor.

RELATED TOPICS

[Setting up your Plugin Development Environment](#)

Using the SDK Maven as an Advanced Maven User

This is a hint for advanced Maven users. You may find it useful to hook up to the version of Maven supplied with the Atlassian Plugin SDK. This would allow you to make use of the [colour-coded Maven output](#) and any new features added in future.

1. Download the [Atlassian Plugin SDK](#) and unzip it into a folder of your choice.
 2. Add the SDK's **Maven bin** directory to your path environment variable. For example, if you unzipped the SDK into this location:
`C:\Atlassian\atlassian-plugin-sdk-3.0`
Then you should add the following to your path:
`C:\Atlassian\atlassian-plugin-sdk-3.0\apache-maven\bin`
-  On Windows, you may need to restart your computer to ensure the new path variable is registered.

RELATED TOPICS

[Developing your Plugin using the Atlassian Plugin SDK](#)

Atlassian Maven Repositories

Atlassian provides various Maven repositories for Plugin Developers. We recommend using the [Atlassian Plugin SDK](#) to make sure you've got the correct settings to use our servers.

The Plugin SDK handles almost all of this Maven tweaking for you. The SDK includes an embedded Maven installation and correct `settings.xml` that will be kept up to date as necessary. We believe it makes the plugin development process much easier. For more information, see [here](#).

Atlassian Maven Proxy

The Atlassian Maven proxy sits in front of all of our other Maven repositories, as well as third-party repositories like iBiblio and Codehaus. This should provide all artifacts needed to build our products and plugins. In the basic case, this is the only repository you need to know about.

- <https://maven.atlassian.com/repository/public>

Legacy Maven 1 Repository

The Legacy Maven repository contains older components that are necessary for compatibility with previous releases of our products. This is a read-only repository. Any new artifacts should be deployed to one of the Maven 2 repositories below.

- <http://maven.atlassian.com/maven1>

Atlassian Private Repository

The Atlassian private repository contains the source artifacts of our closed-source components. Atlassian staff has read and write access.

- <http://maven.atlassian.com/private>
- <http://maven.atlassian.com/private-snapshot>

Atlassian 3rdParty Repository

The third-party directory contains jars that we are [not allowed to redistribute](#), such as those from Sun. Atlassian staff has read and write

access.

- <http://maven.atlassian.com/3rdparty>

Atlassian Public Repository

The Atlassian public repository contains all artifacts necessary to build a plugin for our products. It contains binaries, source and javadoc of our open-source components, and binaries and javadoc for our closed-source components. Anyone can read from this repository. Only Atlassian staff can write to it.

- <http://maven.atlassian.com/public>
- <http://maven.atlassian.com/public-snapshot>

Atlassian Contrib Repository

The Atlassian contrib repository is a resource that we make available for our third-party developers. You can use this to deploy and release your plugins as well as the shared libraries that many developers have created. Anyone can read from this repository. Atlassian staff and Developer Network Committers have write access using your Developer Network credentials.

- <http://maven.atlassian.com/contrib>
- <http://maven.atlassian.com/contrib-snapshot>

If you are not using the [Atlassian Plugin SDK Documentation](#), those wishing to deploy their plugins and libraries to this repository need to add the following to their `pom.xml`:

```
<distributionManagement>
    <repository>
        <id>atlassian-contrib</id>
        <name>Atlassian Contrib Repository</name>
        <url>dav:https://maven.atlassian.com/contrib</url>
    </repository>
    <snapshotRepository>
        <id>atlassian-contrib-snapshot</id>
        <name>Atlassian Contrib Snapshot Repository</name>
        <url>dav:https://maven.atlassian.com/contrib-snapshot</url>
    </snapshotRepository>
</distributionManagement>
```

Also make sure your credentials for deploying are correctly configured in your `settings.xml` as shown below. You can use a `settings.xml` in your home folder for this.

```
<settings>
    ...
    <servers>
        <server>
            <id>atlassian-contrib</id>
            <username>yourusername</username>
            <password>yourpassword</password>
        </server>
        <server>
            <id>atlassian-contrib-snapshot</id>
            <username>yourusername</username>
            <password>yourpassword</password>
        </server>
    </servers>
    ...
</settings>
```

Atlassian Plugin SDK FAQ

- Errors when Creating an Archetype
- How can I change the version of Java my plugin uses
- Maven Cannot Find Java Mail, Java Activation or JTA
- Maven is Unable to Download the Artifact from Any Repository
- Maven Parsing Error Unrecognised HTML Tag
- Maven Runs Out of Memory
- Maven Warning POM for X is Invalid
- SDK doesn't hear customizations for port or context path

Errors when Creating an Archetype

When creating a new archetype (plugin skeleton), you may see errors that look like this:
"[ERROR] ResourceManager : unable to find resource 'VM_global_library.vm' in any resource loader."

These errors are harmless, and you can safely ignore them. We hope to eliminate them eventually, and the issue about them is filed internally as XPR-229.

RELATED TOPICS

[Atlassian Plugin SDK FAQ](#)
[Installing the Atlassian Plugin SDK](#)

How can I change the version of Java my plugin uses

How do I specify the default Java version?

By default, Maven uses the JDK that you run it with to compile your sources. To change the default version of Java that is selected on your machine, please see the specific instructions for your platform. Check with `java -version` from a command line prompt.

Windows - use the [Java Control Panel](#)

Linux - use the `alternatives` command. More information can be found [here](#)

OSX - run Applications, Utilities, Java Preferences. Making changes to soft links by hand is error prone and can produce cryptic errors.

How do I specify a particular version of Java for my plugin?

If you would like to enforce a particular JDK version or version range, add these parameters to your plugin's POM file.

```
<project>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-enforcer-plugin</artifactId>
        <executions>
          <execution>
            <id>enforce-java</id>
            <goals>
              <goal>enforce</goal>
            </goals>
            <configuration>
              <rules>
                <requireJavaVersion>
                  <version>[1.4,1.5)</version>
                </requireJavaVersion>
              </rules>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

Plugins are pre-configured to use Java 1.5 as the source level when compiling. If you would like to use a different source level, you'll need to set the `jdkLevel` property:

```
<project>
  ...
  <properties>
    ...
    <jdkLevel>1.4</jdkLevel>
  </properties>
  ...
</project>
```

This property is also used by the Maven IDEA plugin to set the source level for the project generated when you run `mvn idea:idea`.

Maven Cannot Find Java Mail, Java Activation or JTA

When you run the shell scripts in the Atlassian Plugin SDK, Maven may complain that it cannot find **Java Mail**, **Java Activation** or **JTA**.

For example, for the Java Activation, Maven will give this error:

| *Embedded error: Missing: javax.activation:activation:jar:1.0.2*

Background to this Problem

Sun will not allow Maven to redistribute its binaries. Instead, all users must install Sun binaries manually by downloading them from Sun's website and running the `atlas-mvn install` command. You'll find instructions in the Maven documentation [here](#) and [here](#).

Installing the Sun Binaries

You will find the Sun binaries in these locations:

- JavaMail
- JTA
- Java Activation

The Maven install command looks like this:

```
atlas-mvn install:install-file -DgroupId(javax.XXXXXX -DartifactId=XXXXXX \
-Dversion=X.X.X -Dpackaging=jar -Dfile=/path/to/XXXX-X_X_X.jar
```



Issue with Atlassian Plugin SDK

A known issue with the Atlassian Plugin SDK is causing problems with maven parameters. You may need to run the above command directly from your maven directory. Please see [AMPS-197](#) for more information.

Below, as an example of the detailed installation process, we give specific instructions on installing the JavaBeans Activation Framework (JAF).

Installing the JavaBeans Activation Framework (JAF)

1. Download [JAF](#).
2. Unzip the downloaded file.
3. Go to the folder which contains the `activation.jar` file that you have just unzipped.
4. Install JAF using the following command:

```
atlas-mvn install:install-file -DgroupId(javax.activation -DartifactId=activation \
-Dversion=1.0.2 -Dpackaging=jar -Dfile=activation.jar
```



Issue with Atlassian Plugin SDK

A known issue with the Atlassian Plugin SDK is causing problems with maven parameters. You may need to run the above command directly from your maven directory. Please see [AMPS-197](#) for more information.

RELATED TOPICS

[Atlassian Plugin SDK FAQ](#)

[Installing the Atlassian Plugin SDK](#)

Maven is Unable to Download the Artifact from Any Repository

You may find that Maven complains it is "Unable to download the artifact from any repository."

We use the Maven project's [Archiva Maven Proxy](#) to act as our Maven repository. Sometimes, however, artifact downloads fail because of Archiva's instabilities. If you run into messages like this, sometimes a second attempt will succeed in downloading the artifact. If it doesn't work after a few times, then the artifact may be truly missing. You can search our [Archiva](#) to confirm this.

RELATED TOPICS

[Atlassian Plugin SDK FAQ](#)

[Installing the Atlassian Plugin SDK](#)

Maven Parsing Error Unrecognised HTML Tag

You may receive a Maven error: "Parsing error.... Unrecognised tag: 'html'"

Occasionally, our Maven proxy will return a 404 file instead of the POM that it should. We are investigating this error (XPR-259) with the Archiva developers. In the mean time, should this happen to you, you should try remove the artifact and its POM from your local repository to force a re-download. If that does not work, contact developer-relations@atlassian.com and we will correct the problem at the source.

RELATED TOPICS

[Atlassian Plugin SDK FAQ](#)
[Installing the Atlassian Plugin SDK](#)

Maven Runs Out of Memory

You may need to allocate more memory to Maven in order to complete your build. You can do so by setting an environment variable called MAVEN_OPTS. For example:

```
export MAVEN_OPTS=-Xmx512m
```

RELATED TOPICS

[Atlassian Plugin SDK FAQ](#)
[Installing the Atlassian Plugin SDK](#)

Maven Warning POM for X is Invalid

You may see a Maven warning "[WARN] POM for 'X' is invalid.... Reason: Not a v4.0.0 POM"

The warning just means that this particular library has not been upgraded to have a Maven 2 POM yet. It will still work fine in your build, and you can safely ignore these errors. As we get all our dependencies moved over to Maven 2, and these errors will become less and less frequent.

RELATED TOPICS

[Atlassian Plugin SDK FAQ](#)
[Installing the Atlassian Plugin SDK](#)

SDK doesn't hear customizations for port or context path

Problem

Passing context-path or http-port parameters to atlas-run or atlas-debug doesn't change the base URL of the application instance.

Solution

Make sure to run atlas-clean every time you change the runtime parameters through the SDK scripts; the scripts will cache previous installations and settings, which must be manually cleared.

Writing your first plugin

On this page:

- Prerequisites
- Creating a new plugin
- Java in Atlassian applications
- The Atlassian plugin system
- Plugin modules
- Security considerations
- Javadoc
- Plugin Developer Best Practices
- Plugin and Gadget Gotchas

Prerequisites

This page is the starting point for Atlassian Plugin SDK users who are ready to start writing a plugin.

This page is for you if:

- You have a working Java development environment
- You have a working Atlassian Plugin SDK installed and configured

If you don't feel ready to proceed, please check our guide to [learning how to program](#) and [starting with the Atlassian Plugin SDK](#).

Creating a new plugin

-  **1 Develop:** Create your plugin from an archetype (template) and use the awesome Atlassian tools that make plugin development a breeze. Click through to the tutorials on specific plugin types.
-  **2 Test:** Find out how to include automated tests for your plugin. In our quest for quality, we encourage plugin developers to supply unit and functional tests with a significant amount of code coverage.
-  **3 Package & Release:** Share your killer plugin with the world. Read about licensing, plugin repositories, issue trackers, documentation and how to announce your plugin.

Find Out More: Check out our plugin development resources including forums, reading, example files and more.

Convert: See how to convert your existing plugin to use the new [Atlassian Plugin SDK Documentation](#).

Java in Atlassian applications

Atlassian applications require Java 1.5 or higher.

The Atlassian plugin system

Atlassian products provide a plugin system which allows developers to extend or alter the product:

- The plugin system contains an *OSGi container*, and plugins are understood as *bundles* inside that container.
- A *bundle* is a standard .jar file that contains extra OSGi instructions in the manifest.
- OSGi insulates bundles from each other, primarily by guaranteeing that each bundle has its own classloader. This buys you several good things, including:
 - **Predictable class importing:** You know that the library your code is using at runtime is the one you supplied with it
 - **Talk to other plugins:** Communicate with other plugins; extend them or use their power for your own ends
 - **Multiple versions:** Have several versions of the same library or plugin active at once; you don't need to upgrade just because someone else does

For more information on the plugin framework, see its [home page](#).

Plugin modules

Each product exposes a number of **plugin modules**; some defined by the plugin framework (and thus common to all Atlassian products) and some defined by the product itself. (For example, all products support a servlet plugin module, which allows developers to write a JEE servlet and have it hosted by the product when the plugin is installed. Confluence alone provides a macro plugin module, which allows developers to create custom wiki markup/XHTML elements that will be rendered at page display time.) It is also possible to create your own plugin modules.

All plugin modules are listed in the [Plugin Module Index](#), which will give you an idea of what is possible with the builtin plugin modules and direct you to detailed instructions on how to write with them.

Security considerations

It is easy to overlook vulnerabilities in your code if this is the first time you write a plugin. These, in turn, can introduce security vulnerabilities into the product you plug into.

We have created several APIs to be used for securing your plugins. See this [AtlasCamp 2010 presentation](#) for advice on how to use these APIs and practices.

Javadoc

You can find information on our Javadoc [here](#).

Plugin Developer Best Practices

We've collected a page of best practices for plugin developers. Everybody should know these important tips.

Plugin and Gadget Gotchas

Have a problem you just can't explain? Find some likely solutions [here](#).

Developing your Plugin using the Atlassian Plugin SDK



Current released version – Atlassian Plugin SDK 3.4

Currently-supported applications: **Confluence**, **JIRA**, **Bamboo**, **FishEye**, **Crucible** and **Crowd**.

Atlassian Plugin SDK 3.4 is now available – see the [release notes](#). If you have ideas for improvement or new features, or have found a bug, please raise a ticket on our [issue tracker](#). [Snapshot builds](#) are also available for the stout-hearted.



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

Download Latest Version (Windows)



Download Latest Version (Mac OS X, UNIX, Linux)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

Please make sure that you have set up your development environment and installed the Atlassian Plugin SDK, as described on the [setup page](#). Now you can start building your plugin.



Do not use your local Maven

When running Maven commands against your project, make sure that you use the version of Maven bundled with the Atlassian Plugin SDK. This is important if you have a local version of Maven installed, as well as the Atlassian Plugin SDK. The simplest way is to use the `atlas-mvn` wrapper command instead of `mvn`. Another way is to [put the bundled Maven on your path](#).



No spaces in your directory path

When creating the plugin skeleton, ensure you do **NOT** use spaces in the path or the artifactId. For example use: `C:\Development\MyPlugin` instead of `C:\Documents and Settings\My Plugin`

Quick Start for the Impatient

Step 1. Create a Plugin Skeleton

Step 2. Start the Host Application with your Plugin Installed

Step 3. Generate Project Files for your IDE

Step 4. Write Some Code

Step 5. Run your Changes

Getting Help



Help is at hand

Make sure you are not in the Maven command line interface (CLI) when you enter the help commands described below. If you are in the CLI, your command line will start with `maven2>`, and you will need to exit from the CLI first. Enter `quit`, `exit` or a friendly `bye`.

Getting an Overview of All the Scripts

Enter the following shell script to see an overview of all the scripts with a brief outline of their functionality:

Enter the following to see all possible help content:

Getting Help Text per Script

Each shell script provides help text if the first argument of the script is one of the following:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

Examples:

Reading the Reference Guide

See the detailed [guide](#) to all scripts.

RELATED TOPICS

Welcome to the Atlassian Developer Network

- How to learn to program (in Java or JavaScript)
- Installing the Atlassian Plugin SDK
- Writing your first plugin
- Advanced Plugin Development
- Atlassian Plugin SDK Documentation
- Other Information
- FAQ
- Tutorials
- Getting Involved in the Atlassian Developer Network
- Codegeist V
- Developer Relations State of the Union 2011 - Resources
- Creating a remote debug target

Quick Start for the Impatient

Already know what's what? Ready, set, go:

1. Download and install the latest JDK.
2. Download and install the [Atlassian Plugin SDK](#).
3. Create a plugin skeleton using `atlas-create-APPLICATION-plugin`, e.g. `atlas-create-confluence-plugin`.
4. Run your plugin's host application with the plugin skeleton installed, using `atlas-run`.
5. To install your plugin in the host application, use the Atlassian CLI (command line interface) to dynamically install your plugin. In a new terminal window, run `atlas-cli` then `pi` (plugin install).
6. Open the project in your IDE and start coding.
7. Remember to [make your plugin secure](#)
8. As you make changes, type `pi` again to load your latest code.
9. Write [unit and functional tests](#).
10. When you're done, [Release your plugin](#).

Something didn't work? See the [setup guide](#) and the full instructions on [Developing your Plugin using the Atlassian Plugin SDK](#).

Step 1. Create a Plugin Skeleton

Run the following shell script to create your plugin skeleton. The shell script is part of the [Atlassian Plugin SDK](#). It will download the directories and files that make up a 'Hello World' plugin for your chosen Atlassian application.

1. Add a new folder where you would like to create your plugin.
2. Open a command window and navigate into the folder where you want to create your plugin.
3. Run `atlas-create-APPLICATION-plugin`, where `APPLICATION` is the Atlassian application you are developing your plugin for, e.g. `atlas-create-confluence-plugin` or `atlas-create-jira-plugin`.
4. When prompted, supply the following values:



Make sure your package names are unique

Make sure your package names are unique. You can choose any package name, provided that it does not conflict with any existing package used in the Atlassian application you are developing for, or in any other plugins. Do *not* use `com.atlassian.confluence.plugins.*` — That is confusing to people who try to use the plugin. Use a real package name that corresponds to your organisation or project. For example: `org.mycompany.confluence.plugins.*` The [Sun Java documentation](#) has some tips about conventions used to ensure unique package names.

- '`groupId`' — This would typically be your company or project. This value will also be used as the default for your package name, but you can change it in the next few steps below. For example:
`com.mycompany.confluence.plugins`
 - '`artifactId`' — This is an identifier for your plugin. Do **NOT** use spaces in the artifactId. For example:
`sarahtest`
 - '`version`' — This is the version number of your plugin. You can leave it at the default or enter your own version number. For example:
`1.0-SNAPSHOT`
 - '`package`' — You can leave the default value, as derived from the `groupId` that you entered above. For example:
`com.mycompany.confluence.plugins`
5. Press 'Enter' to confirm your selections, or press 'N' if you want to go back and change the values you have supplied.
 6. You should see a `BUILD SUCCESSFUL` message. (Ignore any warnings. It's all good if the build is successful.)

Troubleshooting this Step

If this process ends with an error message, please refer to the [FAQ and troubleshooting](#) section.

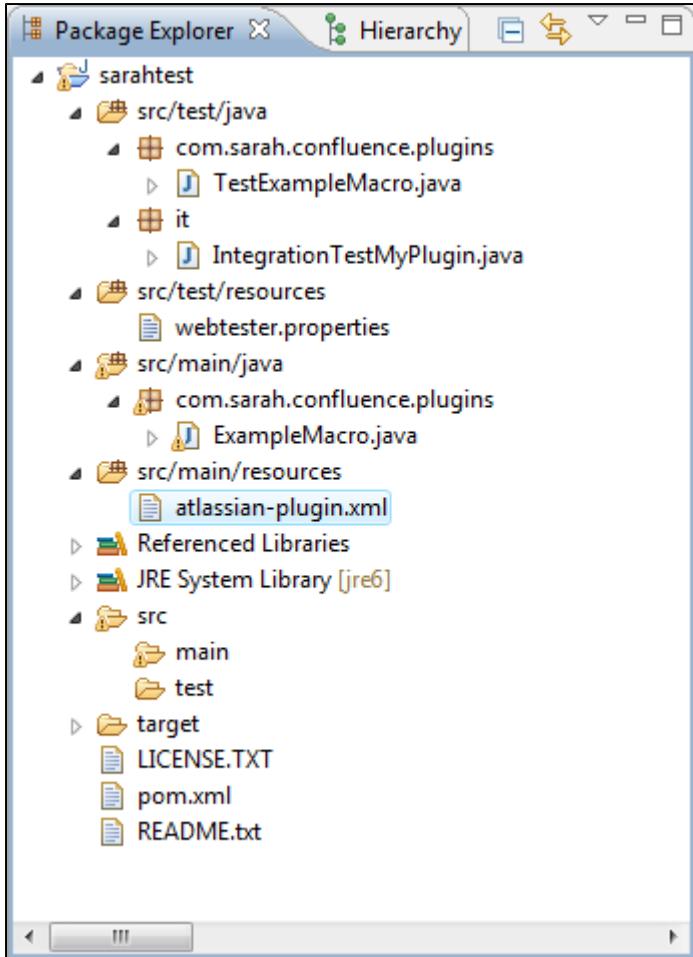
About the Plugin Skeleton

The first step when starting a new plugin is to create a skeleton for that plugin. This involves creating plenty of directories and subdirectories, XML files and Java files. This is what the above steps will do for you. We use a Maven [archetype](#), which is basically a plugin template. We have created plugin templates, or archetypes, for each of our pluggable applications. You can read more about the [Atlassian Plugin Archetypes](#).

When you create your plugin skeleton, Maven will download (a lot of) dependencies from the internet, set up a project skeleton, and prepare you to run unit and functional tests. It creates a new directory structure that looks like this (screenshot taken from Eclipse IDE):

Take a look at the directory where you created your plugin. Note the following files and directories:

- The `pom.xml` file in the root folder of your plugin. POM stands for [Project Object Model](#) definition file.
- Directories for your source code files (`src/main`) and test source code files (`src/test`).
- Directories for your resources (templates, javascript, css, images, etc).
- The `it` package for your [integration tests](#).
- An `atlassian-plugin.xml` [plugin descriptor file](#) in your `src/main/resources` directory.
- A [license file](#) in the root folder of your plugin.



About the pom.xml

The [Project Object Model](#), POM or `pom.xml`, describes everything that Maven needs to know about your plugin: dependencies, source control, authors, tests and deployment locations. The POM is hierarchical, so many properties are inherited.

Take a look at our page with an [example pom.xml](#), explaining your plugin's POM.

About your Local Maven Repository

You will notice that your Maven `.m2` directory now has a 'repository' sub-directory, e.g. `$HOME/.m2/repository`, with a lot of new directories and files in it. Maven keeps local copies of every dependency it ever downloads in this local Maven repository.

You're now ready to move on to [Step 2](#).

Synchronize your pom.xml and atlassian-plugin.xml

The atlassian SDK does this automatically when you create your plugin with the `atlas-create-<product>-plugin`.

Only in specific cases this page may apply to your project.

After setting up your plugin skeleton it is generally a good idea to get your `atlassian-plugin.xml` synchronized with your `pom.xml`. This way you don't have to maintain your project's key, version, name and description in two places.

Setup Maven to filter your resources

Maven has the ability to filter resources and replace values in those resources with values and properties from your `pom.xml` file. So the `pom.xml` is the place to maintain your projects key, version, etc.

First add the following to your `pom.xml` right after the `<build>`-tag

```

<build>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>true</filtering>
      <includes>
        <include>**/*.xml</include>
      </includes>
    </resource>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>false</filtering>
      <excludes>
        <exclude>**/*.xml</exclude>
      </excludes>
    </resource>
  </resources>
  ...

```

This filters all the XML files in your resources directory, excluding binary files.

Replace your atlassian-plugin.xml

Now that your `atlassian-plugin.xml` is filtered by maven it's time to change the header of this file to utilize the filtering.

```

<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.name}"
  plugins-version="2">
  <plugin-info>
    <description>${project.description}</description>
    <version>${project.version}</version>
    <vendor name="${project.organization.name}" url="${project.organization.url}" />
  </plugin-info>

```

That's all! Now your `pom.xml` is the single source for information on your plugin, making your plugin-development-life much easier 😊

Step 2. Start the Host Application with your Plugin Installed

In this step, the Atlassian Plugin SDK will download the host application binaries, run your plugin's unit tests, install your plugin (skeleton) and start the host application. At the end of this procedure, you will be able to visit the application in your browser, log in as '**admin/admin**' and see your plugin in action.

1. Open a command window and go to the folder where your plugin's `pom.xml` is located, i.e. the folder where you created your plugin skeleton.
2. Type `atlas-run` and hit enter.
3. Have a cup of coffee. The download may be a bit slow the first time you do it. But the next time you execute the 'run' command, it will be much faster because all the binaries will be saved in your local Maven repository.
4. If you are on Windows, a Windows Security Alert may pop up, asking if you want to allow the program to accept incoming network connections. Click '**Unblock**' to allow incoming network connections.
5. When all the downloads and installation are complete, you will see a message something like this. (This one is for Confluence):

```

[WARNING] [talledLocalContainer] INFO: Server startup in 87246 ms
[INFO] [talledLocalContainer] Tomcat 6.x started on port [1990]
[INFO] confluence started successfully and available at
http://localhost:1990/confluence
[INFO] Type CTRL-C to exit

```

Open your browser and go to the URL given in the message, such as <http://localhost:1990/confluence> for Confluence. (See [this listing](#) of the ports for the other applications.)

6. The application's login screen will appear. Log in with username '**admin**' and password '**admin**'.
7. [Finding your plugin in the host application](#) to confirm that it was deployed.

Congratulations! You are running the Atlassian application from your local repository. Your simple 'Hello World' plugin has been built and installed into the application.

Troubleshooting this Step

- [Maven Cannot Find Java Mail, Java Activation or JTA](#)

- BeanCreationException from Spring Framework
- Specifying a particular version of the host application
- For other errors, please refer to the FAQ and troubleshooting section.

About the Application Source Code

Please note: The plugin now knows about the host application's binaries, because it has downloaded the JAR files. But it has not downloaded the application source code. You do not need to have access to the host application's source code to be able to develop a plugin. The JAR files alone will enable you to go and write the plugin, using the application's API.

If you hold a commercial license for an Atlassian application with access to the source code, you can attach the application source code to your plugin project. See how to [use the Atlassian Plugin SDK with a source code license](#).

You're now ready to [set up your IDE in Step 3](#).

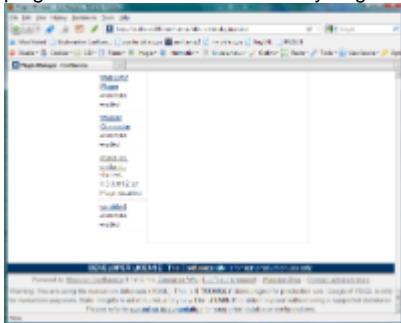
Finding your plugin in the host application

Your plugin now has been installed into your instance of the application. The plugin name will be the 'artifactId' that you gave when creating your plugin skeleton in [Step 1](#).

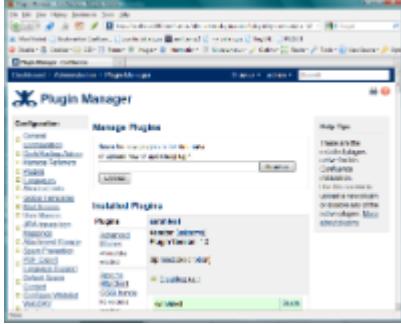
In Confluence

For example, let's assume your application is Confluence:

1. Open the '**Browse**' menu and click '**Confluence Admin**'.
2. The Administration Console opens. Click '**Plugins**' in the left-hand menu.
3. The Plugin Manager opens, showing a list of the plugins installed on your Confluence instance. Scroll down the page to find your plugin. It will have the name that you gave it in step 1 above, e.g sarahtest:



4. Click the plugin name.
5. The plugin details will appear in the highlighted section on the Plugin Manager screen:



6. The Confluence plugin skeleton provides a macro. To use it, edit a page in your Confluence instance and insert a macro with your plugin name as the macro name. For example, if your plugin name is 'sarahtest' then insert a macro like this: {sarahtest}
7. Alternatively, you can see your plugin's macro in the Confluence macro browser.

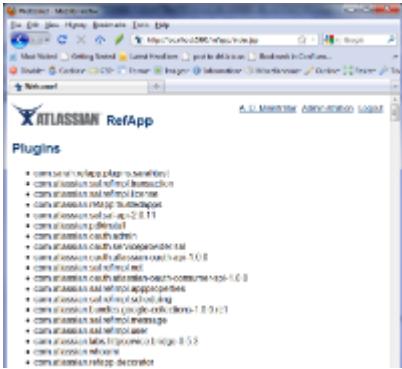
In JIRA

For example, let's assume your application is JIRA:

1. Open the '**Administration**' menu and click '**Plugins**'.
2. The Administration Console opens to the Plugin Manager
3. The Plugin Manager opens, showing a list of the plugins installed on your JIRA instance. Scroll down the page to find your plugin. It will have the name that you gave it in step 1 above, e.g sarahtest, similar to Confluence above.
4. Click the plugin name.
5. The plugin details will appear in the highlighted section on the Plugin Manager screen, similar to Confluence above.
6. The sample JIRA plugin has no module types, so the next step is to create your own.

In the Refapp

Let's assume your application is the Reference Application (refapp). To see details of your plugin and other useful information in the refapp, go to <http://localhost:5990/refapp/index.jsp>.



In Crowd

Crowd, unfortunately, does not yet have a Plugin Management interface. Please try some of the functionality in your plugin to see if it works.

Step 3. Generate Project Files for your IDE

Now it's time to get it set up in your IDE, if you use one. Once again, Maven comes to the rescue. Maven has plugins to generate project files for both [Eclipse](#) and [IntelliJ IDEA](#). Our instructions will cover these two IDEs, but you're free to use whichever IDE or editor you prefer.

1. Open a command window and go to the folder where your plugin's `pom.xml` is located, i.e. the folder where you created your plugin skeleton.
2. Open your plugin project in the IDE:
 - If you are using [IntelliJ IDEA](#), just open the POM in your IDE following [these instructions](#).
 - If you are using [Eclipse](#), see [these instructions](#).
3. Have some more coffee. If you are a technical writer, it's OK to have a hot chocolate instead.
4. Looking at your project in your IDE, you should see the mostly-empty template folders for your new plugin, matching the folder structure in the screenshot in step 1.

Troubleshooting this Step

If this process ends with an error message, please refer to the [FAQ](#) and troubleshooting section.

What's Happening

Maven will download all of the dependencies referenced in your `pom.xml`, and then all the inherited dependencies from the parent POMs (unless it has already done so). Your `settings.xml` specifies that Maven should download the JavaDoc and source artifacts as well, where available. This is likely to take quite a while. Depending on your plugin's host application, Maven will now download 100MB or more from the internet. Go relax, get some more tea or coffee, read a blog. Fortunately, after you've downloaded these files for the first time, they are stored in your local Maven repository (`$HOME/.m2/repository/`) and need not be downloaded again unless they change.

After this is complete, Maven will construct project files* that your IDE can understand. It will reference all the dependent JARs in the right projects, and attach the sources and Javadoc where they are available.

* **NB:** When you are ready to check in your plugin, do not commit these project files. They are specific to each developer's machine.

You are now ready to [write some code](#) in Step 4.

Step 4. Write Some Code

Now you can start editing the plugin files, to transform your plugin from a skeleton to a fully fleshed-out application. The Java classes and the other components you write will depend on your plugin's host Atlassian application, and on the functionality of the plugin.

A plugin consists of the following components, all contained within a single JAR file:

- Java classes encapsulating the plugin logic.
- Resources for display of the plugin UI, if required by the plugin.
- A plugin descriptor, `atlassian-plugin.xml`, that enables the plugin module in your host application, e.g. JIRA or Confluence.

Below are the general guidelines to what you need to do, with links to specific documentation for your plugin's host application.

1. Edit the `pom.xml` file in the root folder of your plugin:
 - Add information about your project, its developers and your organisation.
 - Check that the version of the Atlassian application, e.g. Confluence, specified in the dependencies section is as you want it.
2. Edit the plugin descriptor at `src/main/resources/atlassian-plugin.xml`. A plugin is made up of one or more 'plugin modules'. These modules define the various types of things you can add to the host application. For example, JIRA has a custom field module type, and Confluence has a macro module type. You put new modules into your plugin by adding a section to the `atlassian-plugin.xml` for each new module and implementing the required classes. For details of the plugin module types available, refer to the 'development hub' in the documentation for your host application:
 - [Confluence](#)
 - [JIRA](#)
 - [Crowd](#)
 - [Bamboo](#)
 - [FishEye/Crucible](#)
 - See also the shared guidelines in the [Atlassian Plugin Framework](#) for modules available to all applications.
3. Edit the plugin code in `src/main/java/` adding new classes as needed.
4. Edit the unit and integration tests in `src/test/java/` to test your functional. (Read more about [testing](#).)
5. Learn more about [what you can do with your plugin](#).

To get you started, we have written some tutorials for specific types of plugin:

- [Plugin Tutorial - Writing Gadgets for JIRA](#)
- [Plugin Tutorial - Writing REST Services](#)
- [Plugin Tutorial - Writing Macros for Confluence](#)
- [Plugin Tutorial - Securing your Plugin](#)

As you make changes and additions to your plugin, you'll want to deploy your changes to test. Proceed to the [next step](#) to find out how.

Troubleshooting this Step

- For other errors, please refer to the [FAQ](#) and troubleshooting section.

Now you're ready to [test and debug your plugin](#) in Step 5.

Step 5. Run your Changes

Once you have done the initial `atlas-create-APPLICATION-plugin` and `atlas-run`, you can keep the application running in one command window and use the CLI (command line interface) in another window to dynamically re-install your plugin after each change.

1. Make your changes in your IDE.
2. Open a new command window and go to the plugin's root folder (where the `pom.xml` is located).
3. Run `atlas-cli` to start the CLI.
4. Wait until you see a message, Waiting for commands.
5. Run `pi` (plugin install) to compile, package and install the plugin.
6. Go back to your browser. The updated plugin will have been installed into the application, and you can test your changes. (You may need to refresh the browser page first.)
7. Lather, rinse, repeat.

Note: You can use the shell script directly instead of using the CLI, although the performance is less efficient, just run `atlas-install-plugin`.

Other handy commands

- Run `atlas-compile` to compile the plugin.
- Run `atlas-unit-test` to run the unit tests.
- Run `atlas-package` to produce the JAR.

The Atlassian Plugin SDK provides many more scripts supporting various goals, as described in the [detailed documentation](#).

Running in Debug Mode

You can use `atlas-debug` instead of `atlas-run`. This sets up the host application so that you can attach your IDE's remote debugger to it. Read the instructions on [setting up a remote debugger](#).

Writing Tests for your Plugin

Atlassian plugins depend on Maven 2 to run their test suites. This is advantageous because all plugins are always tested the same way, and because Bamboo, our continuous integration server, can do likewise. Find out how to [make it happen](#).

Troubleshooting this Step



This dynamic deployment method works *most* of the time, but not all plugins may be eligible for dynamic installation. Specifically, some versions of some applications, including JIRA 4.0.0, mark some module types as requiring a restart. If a plugin uses one of the module types that require a restart, the plugin will be installed but not activated until the application is restarted. And occasionally the host application fails to detect new changes. If you suspect this may be happening, just hit `ctrl-c` in the first window and type `atlas-run` again to completely restart the host app with your latest changes included. See the list of [plugin modules that cannot be reloaded with pi](#).

- If you change `pom.xml`, you may need to restart the `atlas-cli`
- For other errors, please refer to the [FAQ](#) and troubleshooting section.

You're ready to learn how to [package and release your plugin](#).

Packaging and Releasing your Plugin

So you've developed a killer plugin and you want to share it with the world. How do you do it?

Atlassian offers a number of services to help you host your plugin development project. We can provide an [JIRA Studio](#) project that will give you an SVN repository with Fisheye and Crucible, a JIRA project, a Confluence space, and a Bamboo Project. We also have a 'Contrib' Maven repository for your artifacts. When you are ready to start collaborating with other developers on your plugin project, read about [Packaging and Releasing your Plugin](#). [Contact us](#) and we can get everything set up for you.

On this page:

- Packaging your plugin
- Releasing your plugin
 - 1. Choose a License
 - 2. Upload your Code to a Subversion Repository
 - 3. Create a Project for Tracking Issues
 - 4. Build and Test your Plugin with Bamboo
 - 5. Release your Plugin
 - 6. Create a Documentation Page
 - 7. Add your Plugin to the Atlassian Plugin Exchange
 - 8. Announce your Plugin
 - 9. Collaborate

Packaging your plugin

Before you release your plugin you might want to package it in order to [dogfood](#) it for example. With the Plugin SDK it couldn't be easier. Simply use the `atlas-package` command. This will produce a `.jar` file in the `target` directory of maven that you can then install in your applications instances.

Releasing your plugin

A Note about Quality

We have set certain criteria for our own plugin development. If you are working on a plugin that you wish to share with the community, we encourage you to try and meet these same criteria in your work. It will result in a higher quality product and make it easier for others to collaborate.

- be in the [Atlassian Plugin Exchange](#).
- have a JIRA project on <http://studio.plugins.atlassian.com/>.
- have source-code available at <http://studio.plugins.atlassian.com/source/>
- have Bamboo builds set up on one of our build servers.
- have unit and functional tests with a certain amount of code-coverage.
- have complete, accurate and attractive documentation in <http://studio.plugins.atlassian.com/wiki/>
- display a compatibility matrix for the last three product releases.
- support all configurations of the product (e.g. clustered Confluence), except where stated otherwise in the plugin documentation
- be internationalised.

1. Choose a License

If you're going to be distributing your plugin, it's a good idea to explicitly license your software. There are many choices, and for the most part you are free to choose whichever one you'd like. The only caveat is that you can't use the GPL, as Confluence and JIRA are not themselves GPL'd. (There is much argument about this point, but we've had GPL authors complain, and we'd rather err on the side of doing the right thing.)

If you don't know which license to use, we generally recommend the [Apache2 license](#). It simple and completely open: people are free to do whatever they want to the software, and you're free of any liability. It's the license we use for most of our plugins.

Of course, you're also free to charge for your plugin, if you want. If so, you should create an appropriate commercial software license.



2. Upload your Code to a Subversion Repository

One of the best reasons to share your plugin is that you might attract other developers to help you with it. In order for them to do that, you'll need a way to share source code. Atlassian provides a [Plugins Subversion Repository](#) where you can commit your source code for other people to check out and use. Hosting your code in *our* Subversion isn't required, but it's a good idea to host it somewhere public. If you would like access to the Plugins Subversion Repository, just send us an email.



3. Create a Project for Tracking Issues

As soon as you make your plugin available, the very first question people will ask is, "where can I file feature requests?" So to make that process easier, Atlassian provides [Plugins JIRA Studio](#) exclusively for your plugin projects. Send us an [email](#), and we'll create a project for your plugin.



4. Build and Test your Plugin with Bamboo

Bamboo is Atlassian's continuous integration server, and <http://studio.plugins.atlassian.com/> contains a Bamboo server exclusively for plugin projects. Bamboo will compile and test your plugin any time anyone makes a change, and alert you instantly to any failures. It can also compile and test your plugin whenever we make a change to the products, alerting us whenever we introduce an incompatibility.



5. Release your Plugin

You can use the Maven Release plugin to automatically upload your plugin to our [Atlassian Contrib Maven Repository](#). That will make it

available to any other Maven project that might need it. You should copy the web location of your JAR in the Maven repository to fill in your plugin metadata in the next step. Also, because Maven repositories are web-accessible, anyone with a browser can download your plugin directly as well. Point all of your download links here. (You can apply for an account by following the procedure [here](#).)



6. Create a Documentation Page

You should create a wiki space on <http://studio.plugins.atlassian.com/> to serve as documentation for your plugin (see this great example).

You should add a description of the plugin's features, instructions for installation and usage, and some screenshots of your plugin. (People like pictures!) You might even consider adding a screencast of your plugin in action. That's sometimes the best way to get your point across.

Lastly, you should be sure to "watch" the page so that you notice any comments or changes and can respond to them.



7. Add your Plugin to the Atlassian Plugin Exchange

Now that you've got your code committed, your issue-tracking project ready and your plugin documentation up, it's time to make a little noise. Go to <http://plugins.atlassian.com> and create a new entry for your plugin there. You'll find full instructions for the Atlassian Plugin Exchange [here](#).



8. Announce your Plugin

You should also announce the release of your plugin on [Atlassian's forums](#). (Everything posted to the forums will be automatically copied to the mailing lists, and vice versa.)

9. Collaborate

Once people start to use your plugin, they're going to want to get involved. Some people will file bugs. Some people will request new features. And best of all, some users might pitch in an add new a feature themselves. You can encourage this behaviour by being responsive, answering questions, making fixes and additions as quickly as you can. Demonstrating momentum is the surest way to keep people interested and motivate them to contribute themselves.

Plugin Release Checklist

Each time you release a new version of a plugin, there are certain steps you should perform.

Creating the release

1. Make sure your plugin's **pom.xml** is up-to-date. This includes accurate sections for:
 - *Parent POM* — Make sure that the plugin is using the correct parent POM for the product and the most recent version of it.
 - *version* — It should be <version_to_be_released>-SNAPSHOT.
 - *scm* — Source control repository locations.
 - *atlassian.product.version* - See [atlassian.product.version](#) description in Testing your plugin.
 - *atlassian.product.data.version* — See [atlassian.product.data.version](#) description in Testing your plugin.

2. Run **mvn release:prepare** — This will update your pom.xml release numbers and make a new tag for the release.
3. Run **mvn release:perform** — This will:
 - a. Run tests
 - b. Compiles code and package it
 - c. Put the binary package in an appropriate [Atlassian Maven Repository](#) and your local Maven repository.
4. Copy the created binary package from your target sub-directory to the project's jars sub-directory (in SVN). Then schedule it for addition to SVN before finally committing the added resource.

Updating the plugin's project

#You'll need to release the current version of the project in JIRA and move any unfinished issues for the project to a later version. Here's how:

1.
 - a. Go to the project page in JIRA.
 - b. Click on **Administer Project**.
 - c. Under **Versions**, click on **Manage versions**.
 - d. From this screen, choose "release" for the version that you just created. Make sure the date is correct, and move any remaining open issues to the next unreleased version.
2. Update plugins documentation on on <http://studio.plugins.atlassian.com/wiki/>.
3. Update the plugin's listing on <http://plugins.atlassian.com>
4. Communicate about release (probably with a blog post and a post to the appropriate mailing list.)

Writing Tests for your Plugin



Current released version – Atlassian Plugin SDK 3.4

Currently-supported applications: **Confluence**, **JIRA**, **Bamboo**, **FishEye**, **Crucible** and **Crowd**.

Atlassian Plugin SDK 3.4 is now available – see the [release notes](#). If you have ideas for improvement or new features, or have found a bug, please raise a ticket on our issue tracker. Snapshot builds are also available for the stout-hearted.

On this page:

- [Why Automated Testing is Important](#)
- [Unit Tests](#)
- [Integration, or Functional, Tests](#)
- [Skipping Tests](#)
- [Hints and Notes](#)
- [Debugging your Failing Tests](#)
- [Getting Help](#)

Why Automated Testing is Important

Atlassian plugins depend on Maven 2 for running their test suites. This is advantageous because all plugins are always tested the same way, and because Bamboo, our continuous integration server, can do likewise.

Building a suite of tests for your plugin is **extremely** important. We are big believers in [Test Driven Development](#), and we think that writing tests as you go is one of the best ways to ensure you build a high-quality and functional application. Additionally, having a comprehensive suite of tests means that other developers in the community (or even Atlassian developers) can jump in and collaborate on your plugin without fear of breaking anything inadvertently.

Our products are always moving forward. We enhance and change things. We move or deprecate APIs. A plugin that works perfectly in Confluence 2.5 might change in Confluence 2.6, in subtle or profound ways. That is the primary reason that we run the [Developer Bamboo Server](#) – to alert us to any changes that might break plugins. When those breakages occur, we might be able to go back and do things another way that preserves compatibility. Or, at the very least, we can notify developers in advance of a new release about what has changed. Continuous Integration is a crucial safeguard, and it depends largely on a comprehensive test suite to alert us when things break.

Unit Tests

The first tests you should write for your plugin are [unit tests](#). Unit tests are designed to exercise individual chunks of your code in isolation.

Here are some useful examples of plugins with unit tests:

- [JIRA Labels Plugin](#), using the jMock test library.
- [JIRA Chart Plugin](#).
- [JIRA FishEye Plugin](#).
- [Confluence Chart Plugin](#).

Running the Unit Tests

To run unit tests, use the following shell script supplied by the Atlassian Plugin SDK:

```
atlas-unit-test
```

The Maven command also works:

```
atlas-mvn clean test
```

Structure Provided in the Plugin Skeleton

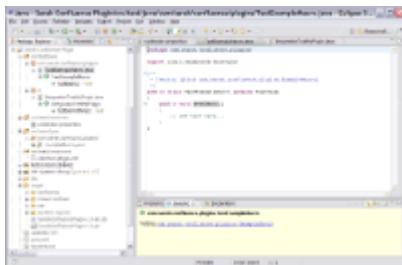
Here is an example of the default unit test included in your plugin, when you create it via the Atlassian Plugin SDK as described in the step-by-step guide:

```
package com.sarah.confluence.plugins;

import junit.framework.TestCase;

/**
 * Testing {@link com.sarah.confluence.plugins.ExampleMacro}
 */
public class TestExampleMacro extends TestCase
{
    public void testBasic()
    {
        // add test here...
    }
}
```

Screenshot: Default unit test provided in the plugin skeleton



(Click the above image to enlarge it.)

Integration, or Functional, Tests

Start by reading this [great testing tutorial](#) from CustomWare.

Integration Testing is designed to test your code inside the the Atlassian application. The goal is confirm that the plugin functionality works as designed, and is compatible with various versions of the application. To achieve this goal, Maven will actually deploy your plugin to a running version of JIRA or Confluence and then run tests that you write to exercise its functionality.

For examples of integration tests, take a look at the [JIRA Fisheye Plugin](#) tests.

Running the Integration Tests

To run integration tests, the shell script is:

```
atlas-integration-test
```

The Maven command also works:

```
atlas-mvn clean integration-test
```

When you run this command, Maven will start up a new instance of JIRA or Confluence, running inside Tomcat, and install some default data into it: things like a license, database configuration, and an admin account. This will save you the effort of having to set up the product each time.

i Note that the license installed is a unique one: it is perpetually valid, but only for a period of three hours (which should be more than enough time to run your tests). If you keep going longer than three hours, the license system will lock you out, and you will need to restart it

to proceed.

Structure Provided in the Plugin Skeleton

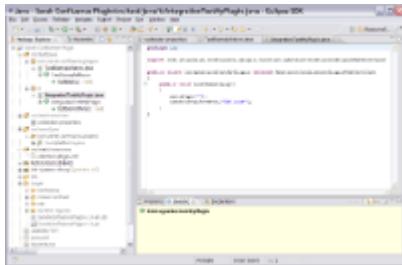
Here is an example of the default integration test included in your plugin, when you create it via the Atlassian Plugin SDK as described in the step-by-step guide:

```
package it;

import com.atlassian.confluence.plugin.functest.AbstractConfluencePluginWebTestCase;

public class IntegrationTestMyPlugin extends AbstractConfluencePluginWebTestCase
{
    public void testSomething()
    {
        gotoPage( "" );
        assertTextPresent( "Welcome" );
    }
}
```

Screenshot: Default integration test provided in the plugin skeleton



(Click the above image to enlarge it.)

Data for Integration Testing

See our page on product data for integration testing.

Skipping Tests

To skip the various test phases, use the following shell script:

```
atlas-integration-test --skip-tests
```

Alternatively, using Maven directly you can add one of the following options to your Maven command:

```
> -Dmaven.test.skip=true - skips both unit and integration tests  
> -Dmaven.test.unit.skip=true - skips unit tests  
> -Dmaven.test.it.skip=true - skips integration tests
```

Hints and Notes

1. **Importing data into your application:** Often your functional tests will need to set up the application in a particular state. You may wish to do this by creating an XML export of JIRA or Confluence data in a known state, and importing that data at the beginning of your tests. XML data that is usually imported by the tests into the running product should be placed in the `src/test/xml/` directory provided.
2. **Generating test data for re-use:** Use the `atlas-create-home-zip` command to create a zip file of your application's home directory, and then load the contents of the home directory into the application every time you start it up.
3. **Naming conventions for test packages:** Packages containing integration tests should start with 'it'. For example:
 - `it.com.atlassian.jira.plugins.fisheye`
 - `it.com.atlassian.jira.plugins.calendar`
4. **Running a different version of the application:**
 - Change the version of your plugin's host application, in the dependencies section of your POM. Note that your POM may use a property to hold the version number, like this:

```
<properties>
    <confluence.version>VERSION_NUMBER</confluence.version>
</properties>
```

- Run `atlas-clean`.
- Run `atlas-run` to run the specified version of the application.

5. Testing against a different version of the application on different containers:

- Run `atlas-clean && atlas-integration-test -v 3.1-m7 -c tomcat5x`.
 - This will clean the build output directory and run the integration tests against version 3.1-m7 of Confluence on the Apache Tomcat 5.5 container.
- Run `atlas-integration-test --help` to get a full listing of the available command line options.

Debugging your Failing Tests

All of the necessary output generated by tests will be found under the `target` subdirectory of your plugin module. The following files are useful to look at if you are having trouble:

- `surefire-reports/*.txt` – A log file per test class of failed tests (both unit tests and functional tests).
- `<product>/output.log` – Where `product` is the host application of your product, e.g. `bamboo`, `confluence` or `jira`. This is the Atlassian application's log file, e.g. `atlassian-jira.log`.

Note: The `<product>` directory will be found under `debug` if the Maven profile, `plugin-debug` is activated. That means, when the `plugin-debug` profile is activated, the directory becomes `debug/<product>/output.log`. This isolates debug resources and the actual integration test resources.

Getting Help



Help is at hand

Make sure you are not in the Maven command line interface (CLI) when you enter the help commands described below. If you are in the CLI, your command line will start with `maven2>`, and you will need to exit from the CLI first. Enter `quit`, `exit` or a friendly `bye`.

Getting an Overview of All the Scripts

Enter the following shell script to see an overview of all the scripts with a brief outline of their functionality:

Enter the following to see all possible help content:

Getting Help Text per Script

Each shell script provides help text if the first argument of the script is one of the following:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

Examples:

Reading the Reference Guide

See the detailed [guide to all scripts](#).

RELATED TOPICS

[Plugin Testing Resources and Discussion](#)

[Welcome to the Atlassian Developer Network](#)

- [How to learn to program \(in Java or JavaScript\)](#)
- [Installing the Atlassian Plugin SDK](#)
- [Writing your first plugin](#)
- [Advanced Plugin Development](#)
- [Atlassian Plugin SDK Documentation](#)
- [Other Information](#)
- [FAQ](#)
- [Tutorials](#)

- Getting Involved in the Atlassian Developer Network
- Codegeist V
- Developer Relations State of the Union 2011 - Resources

Product Data for Integration Testing

The AMPS plugin and by extension the plugin SDK rely on product data for integration testing. This data is simply a zip archive of the home directory of a fully configured application. This usually contains an [HSQL database](#) that will be used during testing. The plugin data also contains a license valid for plugin testing.

For convenience this archive is deployed to Atlassian's maven repository and is automatically downloaded by the AMPS plugin (resp. the plugin SDK) when required. You can configure the version of the data you want to use in your plugin via the `<product>.data.version` properties. We generally only release major versions (3.x) and not minor versions (3.x.x) of the product data.

Note that the product data does not *have to* match the application version you are testing. In many cases, you can use an older version of the product-data. The application will automatically upgrade that data when it starts up. So, for example, you can continue to use the Confluence 3.1 product data with 3.1.2, or even with 3.2 and 3.3.

By default this version is set to `LATEST`, which means maven will figure out which one is the latest version from the remote repositories and use that one.

Here are the artifacts as deployed in the maven repository and used by the plugin:

| Product | Group ID | Artifact ID | Latest Version | |
|------------------|----------------------------------|----------------------------------|----------------------|--------------|
| JIRA | com.atlassian.jira.plugins | jira-plugin-test-resources | 4.2 | all versions |
| Confluence | com.atlassian.confluence.plugins | confluence-plugin-test-resources | 3.1 | all versions |
| Bamboo | com.atlassian.bamboo.plugins | bamboo-plugin-test-resources | 2.6.1 | all versions |
| Crowd | com.atlassian.crowd.distribution | crowd-plugin-test-resources | 2.0.6 | all versions |
| FishEye/Crucible | com.atlassian.fecru | amps-fecru | 2.3.4-20100712032131 | all versions |

Note: Plugin data is not (yet) automatically generated with each new product release. So versioning of the data might be lagging behind the versioning of their products. We will aim to get them in sync when possible. If you have any questions don't hesitate to ask

Setting OSGi Manifest Instructions in your Plugin

- What's this, then?
- Preliminaries
 - Plugins, bundles and OSGi
 - Specifying manifest instructions
 - Package bundling instructions
 - Package name specification
 - Version specification
 - Using Atlassian code
 - Using third-party code
 - Other OSGi instructions

What's this, then?

This page will describe:

- How the Atlassian plugin system supplies libraries to your plugin
- How to use Atlassian code in your plugin
- How to use third-party code in your plugin

Preliminaries

First, we'll explain a few background topics you'll need to understand.

Plugins, bundles and OSGi

An Atlassian plugin is an OSGi bundle. This implies a lot of things, but there are two fundamentals you must understand:

- **Classloader isolation:** A new classloader is created for each plugin at startup, and it's used to load your plugin's classes. This is not the same as the system classloader, nor can you access any class the system knows about through it. This provides several benefits: different plugins can use different versions of the same library simultaneously, for example. However, it also means that several assumptions you may have made about how class loading works in Java are no longer valid. This page will explain how to do what you want without needing to understand every detail of the system.
- **Runtime dependencies:** At plugin start, the plugin system will ask your plugin for the dependencies it requires to run and then attempt to locate them somewhere in the available OSGi publicly-visible packages. If they can't be found, the plugin will not start and

you will see an error message, usually a `NoClassDefFoundError`. If they are found, the plugin classloader is created and told where to load the libraries and versions requested – and ONLY those libraries.

You will follow the same process every time you decide to use another piece of code in your plugin:

- "Where will I get the artifacts to compile my code against?" Answering this question will tell you what you need to add as `<dependencies>` in your `pom.xml`.
- "Where will my code find these artifacts at runtime?" Answering this question will tell you which manifest instructions you need to specify in your `pom.xml` (inside the AMPS configuration section).

In most cases, you either need something from the Atlassian product or you want to use something you package inside your plugin, like a third-party library. There are a few other complications which we'll discuss below.

Specifying manifest instructions

Manifest instructions tell the plugin system what Java packages – and, optionally, which version of those packages – your plugin needs to have available in its classloader at runtime. You add instructions to your plugin using the `pom.xml` file, which is created when you use one of the `atlas-create` commands to create a new plugin:

```
<project>
  <build>
    <plugins>
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-***-plugin</artifactId> <!-- *** is the name of the application
(product) you're using -->
        <version>3.2.3</version>
        <extensions>true</extensions>
        <configuration>
          <productVersion>${product.version}</productVersion>
          <instructions>
            <!-- OSGi instructions go here -->
          </instructions>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```



What other configuration options are available?

See [this page](#) for details.

Package bundling instructions

Specifying runtime dependencies is done through a custom declaration syntax called a **bundling instruction**. The SDK uses the [Apache Felix Maven plugin \(BND\)](#) behind the scenes to handle this, so the instructions explained here are passed to BND at package time and used to create the OSGi manifest.

Bundling instructions are specified by the format:

```
package-name(wildcard);version="<major>.<minor>.<maintenance>"
```

Multiple instructions are separated by commas.

Here's a real-world example:

```
com.atlassian.confluence.events.event.space;version="[3.4,4.0)"
com.atlassian.confluence.search.lucene.*;version="[3.4,4.0)"
com.atlassian.confluence.json*;version="[3.4,4.0)"
```

Package name specification

There are three different ways of specifying the package name:

- **com.atlassian.confluence.events.event.space**: Imports every class in the `com.atlassian.confluence.events.event.space` package.
- **com.atlassian.confluence.search.lucene.***: Imports every class in **subpackages** of `com.atlassian.confluence.search.lucene` while **not** importing any classes `com.atlassian.confluence.search.lucene` itself

- **com.atlassian.confluence.json***: Imports every class in `com.atlassian.confluence.json` and all classes in all subpackages of `com.atlassian.confluence.json`

Version specification

There are several ways to specify which version of a package to import:

- `version="0.0"`: Matches any version found. This is used when your code requires the package but doesn't care which version of the code is used.
- `version="1.4"`: Matches any version equal to the version specified or any later version. Any package versioned as 1.4, 1.5, 2.0, or 9.0 will satisfy this specification
- `version="[2.0,2.8]"`: Matches any version in the range from 2.0 (exclusive) to 2.8 (inclusive). Either end of a range can be exclusive or inclusive.

In the Confluence example above, the version specification `[3.4, 4.0)` translates to:

For the preceding package, import any version from 3.4 inclusive to 4.0 exclusive; that is, any version from 3.4 itself to any version less than 4.0.



Use ranges wherever possible

The ranges in the above example allow the code to work with any future 3.x version of Confluence (assuming the APIs being used don't break in one of those releases). In general, set your upper bound to the next major version (exclusive) – whatever "major version" means for that library.

Using Atlassian code

Each OSGi system has a special bundle called the **system bundle**, which functions similarly to the system classloader in vanilla Java applications. Atlassian products export the following through the system bundle:

- The usual Java runtime libraries (everything in the `java.* / javax.*` namespaces)
- Everything in the product itself that we think a plugin author might want to use (most of the product's core classes and any proprietary libraries we use, which means most of `com.atlassian.*`)
- Several open source libraries the product uses (things like the Apache Commons libraries `org.apache.commons.*`, Google Collections `com.google.collect.*`, etc.)

All plugins can use packages exported by the system bundle; in fact, the plugin system inspects your code at runtime, sees what code you want to use from the system bundle, and automatically adds the necessary directives. This works in simple cases, and it's good enough to start with, but we strongly recommend that you specify exactly what you need as part of the polishing-before-release process.



Adding JAR files to WEB-INF/lib

In the past, you could add a JAR file to the `WEB-INF/lib` directory and make it available to all plugins, as well as the product itself. **This no longer works**, as the plugin system doesn't know about any code in the product beyond what it's been told to use; placing random JARs in `WEB-INF/lib` or class files in `WEB-INF/classes` will **not** make them available in the system bundle.

To use a package from the system bundle in your plugin, do the following:

- Add the library to your `pom.xml` under the `<dependencies>` block.

```
<project>
  ...
  <dependencies>
    <dependency>
      <!--
        we're using Confluence in this example;
        this is taken directly from the plugin
        generated when you run the command
        atlas-create-confluence-plugin.
      -->
      <groupId>com.atlassian.confluence</groupId>
      <artifactId>confluence</artifactId>
      <version>${confluence.version}</version>
      <scope>provided</scope>    <!-- important! -->
    </dependency>
  </dependencies>
  ...
</project>
```

Adding a `<dependency>` tells the SDK that your code will use this artifact, but specifying `<scope>provided</scope>` refines that to mean "these classes will be provided at runtime by the product". The SDK will then use these classes when compiling your plugin, but it won't

actually put Confluence itself inside your plugin when it is built.

Similarly, if you were writing a [servlet plugin](#), you'd need to tell the SDK where to find the servlet API classes to compile against, but you wouldn't want to actually include the servlet APIs in your plugin. To do that, you'd add this to your `pom.xml`:

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId> servlet-api</artifactId>
      <version>2.4</version>
      <scope>provided</scope>      <!-- important! -->
    </dependency>
  </dependencies>
  ...
</project>
```

- Add import instructions for the packages you're using

Having taken care of the compile time side, we turn our focus to runtime. We need to tell the plugin system what packages to make available to our classloader. This is done with the `<Import-Package>` instruction.

Assume your plugin requires classes from Confluence's JSON support, space events, and Lucene mechanisms. To import these Confluence packages at runtime, use the following:

```
<configuration>
  ...
  <instructions>
    <Import-Package>
      com.atlassian.confluence.events.event.space;version="[3.4,4.0)" ,
      com.atlassian.confluence.search.lucene.*;version="[3.4,4.0)" ,
      com.atlassian.confluence.json*;version="[3.4,4.0)"
    </Import-Package>
  </instructions>
</configuration>
```

Using third-party code

To use third-party code in your plugin, you need to add a `<dependency>` just like above, but with an important change:

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>com.mycompany.awesome.stuff</groupId>
      <artifactId>awesome-utils</artifactId>
      <version>1.0.4</version>
      <scope>compile</scope>      <!-- important! -->
    </dependency>
  </dependencies>
  ...
</project>
```

By specifying `<scope>compile</scope>`, you tell the SDK that the `awesome-utils` library is needed at compile time and runtime. The SDK will **bundle** the `awesome-utils` library into your plugin, and the plugin system will make it available in the plugin classloader when the plugin is started.

Other OSGi instructions

In the background the Atlassian Plugin SDK uses Felix's [Maven Bundle Plugin](#) to process those instructions. For detailed information about which instructions are allowed and how they work, refer to the [documentation on the Felix website](#).

RELATED TOPICS

John Kodumal's [AtlasCamp 2010 presentation](#) covered these and other modular plugin development topics.

[Developing your Plugin using the Atlassian Plugin SDK](#)
[OSGi, Spring and the Plugin Framework](#)
[Going from Plugin to OSGi Bundle](#)

Plugin Hosting on JIRA Studio

Atlassian can provide a full project hosting infrastructure for your plugin. We'll give you a JIRA Studio project on <http://studio.plugins.atlassian.com/>, which gives you JIRA for bug tracking, a Confluence wiki, a Subversion repository, FishEye and Crucible for source visualisation and code review and Bamboo for doing continuous integration builds. Just contact developer-relations@atlassian.com and we will set you up.

In the meantime, you can look at all the other projects here: <http://studio.plugins.atlassian.com/>.

Some more useful documentation:

- [Plugin Studio How-To](#)
- [Building with Bamboo on Plugin Studio](#)

Building with Bamboo on Plugin Studio



Good News, Everyone!

We have completed the process of enabling Bamboo build plan creation for Atlassian plugin developers! This page will provide a brief overview, on top of Bamboo's regular [product docs](#), of how to set up a build plan for your plugin projects hosted on Studio.

Prerequisites

To set up automated builds / continuous integration for your project on Studio, there are two things we want to make sure you have first:

1. **A user account:** If you don't already have an account, you can create one yourself on the [Signup](#) page.
2. **A project hosted on Studio:** Okay, this may sound obvious, but it's probably worth noting that to use Bamboo on Studio, your project needs to be hosted on Studio, not elsewhere. If your project isn't already hosted here, send us email at developer-relations@atlassian.com, with some basic information about the name and functionality of the plugin, and we'll set up the hosting for you. Have a look at the [overview and introduction](#) to the functionality we provide in Studio, for more information.

With these two things, your user account should be associated with the *developers* and *FOO-developers* groups (assuming your project has the key "FOO," although we would like to assure you we can come up with much more cleverly named project keys than that). Membership in the *FOO-developers* group will give you privileges to do most administrative tasks for project FOO, and membership in the *developers* group is what is specifically necessary for Bamboo build plan creation.

The Fun Part

Okay, you have battled the two-headed dragon of user account and project creation, and somehow you have survived. There's no turning back now, it's time to create a build plan. Click on the "Builds" tab, then on the "Create Plan" sub-tab (or just click [here](#)), to get started. The first two steps, Plan Details and Source Repository, are essentially as described in the [regular Bamboo documentation](#) on how to create a build plan. For the third step, [Configuring Tasks](#), we've done a bit of extra work on our end to simplify the specification of build details for plugin developers. This is the fun part.

Normally, plugin developers use the scripts packaged with the [Atlassian Plugin SDK Documentation](#), such as `atlas-compile` and `atlas-run`, to avoid much of the complexity of the Maven environment and the various differences in deployment between the various Atlassian web

applications, while providing a consistent framework for implementing testing, deployment, and so on. Bamboo does not typically provide a builder specific to the Atlassian Plugin SDK, but because the SDK is the preferred tool for plugin development, we felt it was important to provide an **SDK-specific builder** for the Plugin Studio.

When you select the Atlassian Plugin SDK builder from the options on the "Builder Configuration" page, you will see options specific to the SDK, including which Atlassian product should host your plugin, which web application container should host the product, the location of any unit test output, etc. In most cases it's perfectly fine to leave these at their default settings (i.e. choose the product and container automatically, etc.) For particularly sophisticated plugins, such as cross-product plugins, you may choose to set up multiple build plans, each targeting a different specific product.



Deploying Snapshots

A common scenario and best practice for continuous integration, particularly when Maven is used, is the automatic deployment of snapshot artifacts for every successful build. For plugin developers, we have always encouraged deployment to our "contrib" and "contrib-snapshot" repositories. However, as much as we would like to provide the ability to deploy snapshots to contrib directly from Bamboo, we have not found a good way to do so without security liability. Therefore, unfortunately, "deploy" is not an option in the "Build phase" list.

One Last Thing

After configuring the builder, there is one final piece of required information to provide, which is the build plan's capability requirements. Again, this step is essentially as described in the [regular Bamboo documentation](#), but it's worth mentioning that there is presently only a single image configured for use in a Bamboo remote agent. This image is basically the default image provided by JIRA Studio, customized slightly with an attached EBS volume containing a pre-populated Maven repository and a recent version of the Plugin SDK. Most importantly, this image exposes **no custom capabilities**, and we do not anticipate being able to accommodate individual requests for custom capabilities, so please avoid custom requirements if possible.

Finally, you can save and enable your new build plan immediately, without the need to configure artifacts, notifications, post actions or permissions. You can come back and edit these at any time. For starters, kick off your first build, watch it run, and wait for the green to come. Enjoy!

Plugin Developer Best Practices

Use your own package name

Make sure your package names are unique. You can choose any package name, provided that it does not conflict with any existing package used in the Atlassian application you are developing for, or in any other plugins. Do *not* use `com.atlassian.confluence.plugins.*` — That is confusing to people who try to use the plugin. Use a real package name that corresponds to your organisation or project. For example: `org.mycompany.confluence.plugins.*` The [Sun Java documentation](#) has some tips about conventions used to ensure unique package names.

Plan Details
Source Repository
Builder
Requirements
Artifacts
Not

Builder Configuration

Builder:

Build JDK: *

Which JDK do you need to build the build?

Build Phase:

Compile only
 Compile, then run unit tests
 Compile, then run unit and integration tests
 Clean Before Building?

Integration Test Environment

Product:

Product Version:

Container:

Where should Bamboo look for the test result files?

The build will produce test results.
Note that test files must be in JUnit XML format. For more information, see the [NCover output will be produced](#).

Test Results Directory: Look in the standard test results directory.
 Specify custom results directories
Where should Bamboo look for the test result files?

When should bamboo determine a build has hung?

Override default hanging build detection criteria
Enable and override hanging build criteria for this build.

Would you like to view Clover Code Coverage for this plan?

Use Clover to collect Code Coverage for this build.
Clover is a code coverage tool that helps ensure that your code is well tested.

Namespace your CSS and Javascript

We can't promise that our products CSS and Javascript isn't going to change from version to version. For that reason, you are generally best off writing CSS and Javascript that does not depend on the products' at all. One of the best ways to do this is namespace your CSS and Javascript.

In the CSS realm, there is an [official namespace proposal](#), but that's not what I'm talking about as the current browser crop doesn't support these enhancements yet. All I'm really talking about is intelligently naming your selectors so that you don't bleed styles between your plugin and the product itself. For example, rather than having an element like:

```
<table class="table">...</table>
```

you might try

```
<table class="MyPluginName_table">...</table>
```

That will ensure that your CSS rules don't get applied to anyone else's plugins, and you don't inadvertently pick up a style from Confluence's table class that may change in the future.

On the Javascript side, you run an even greater risk of conflicting with the product or with another plugin. For that reason, you should namespace your javascript as outlined in [this tutorial](#). Our products don't actually follow this rule yet, so it's doubly important for plugins to do so.

Application-Specific Best Practices

Confluence:

- Community Development Practices
- Ten common mistakes in plugin development
- Development Hub

JIRA:

- JIRA Community Code Examples
- JIRA Development Hub

Plugin and Gadget Gotchas

This page contains guidelines for resolving errors related to classes not being found by a plugin. Some of these errors are due to unresolved Maven dependencies, some are due to OSGI dependencies. These errors were all generated using JIRA plugins but apply to all Atlassian applications.



Some of these bundle gotchas can be debugged using the [Felix Web Console](#) or better still, the [OSGI](#) tab in the plugin manager (UPM) version 1.3 and later (JIRA 4.3+).

Compilation failure: package does not exist
Cannot start plugin
NoClassDefNotFoundError
UnsatisfiedDependencyException
ClassNotFoundException
UnsatisfiableDependenciesException
java.lang.LinkageError
PluginException
SAXParserContextProvider

Compile Time Errors

These errors are triggered by any PDK command that starts a javac compile such as `atlas-compile`, `atlas-package` or even `mvn package`.

These go beyond the usual "cannot find symbol" [Java compilation errors](#) due to typos in the method or class name, or missing import statements, or typos in a package name.

Symptom: "Compilation failure: package does not exist"

Cause: the javac compile command did not have one or more required .jar files specified on the classpath

Solution: add a dependency element in `pom.xml` for the artifact that contains the missing class

Example: package javax.servlet.http does not exist

Add this element to `pom.xml`:

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>2.3</version>
    <scope>provided</scope>
</dependency>
```



How do I find the correct artifact name and version? Best practice is to use the same version as the jar file shipped with the version of JIRA, Confluence etc you are building with.
The `provided` scope element indicates that the jar file doesn't need to be packaged with the plugin since the main application can already provide it.

Start Up Errors

These errors are triggered when your new plugin is deployed and the application starts up. The application log file contains errors or warnings about it even before you try to use the plugin.

Symptom: "Cannot start plugin"

This is a WARNING not an ERROR in the log. The plugin will also be shown as Disabled on the Admin, Plugins screen.

Cause: the OSGI plugin does not know anything about a class used when the plugin is started up

Solution: correct any typos in the `interface` attribute in the `component-import` element. Also make sure that the component is exported from elsewhere with a `<component>` element.

Example:

```
2010-02-24 16:18:52,882 main WARN [plugin.osgi.factory.OsgiPlugin] Unable to enable plugin
'com.example.jira.plugins.example-errors'
com.atlassian.plugin.osgi.container.OsgiContainerException: Cannot start plugin:
com.example.jira.plugins.example-errors
at com.atlassian.plugin.osgi.factory.OsgiPlugin.enableInternal(OsgiPlugin.java:385)
```

Note that the error messages do not show the name of the component that the OSGI container failed to import. In this case the problem was

```
<component-import key="foo" interface="com.example.does.not.exist"/>
```

Run Time Errors



- Some of these errors and warnings only appear in `logs/catalina.out` and not in `atlassian-jira.log`
- If one REST resource in a plugin has a problem, then all the other resources and their URLs will produce HTTP 404 errors

These errors are triggered when a new plugin is used and the application log file contains an error the first time the plugin is used.

Symptom: "NoClassDefFoundError: com/atlassian/jira/gadgets/system/AbstractResource"

Cause: a plugin is lazily instantiated and then fails to find a needed class

Solution: make sure the class and package name are correct and that the class is declared as a component in its `atlassian-plugin.xml`.

Simple Example:

```

2010-02-23 13:17:10,611 http-8080-Processor19 ERROR admin 47830x17x3 1bvaudl
http://localhost:8080/rest/gadget/1.0/stats/generate
[atlassian.plugin.servlet.DefaultServletModuleManager] Unable to create filter
com.atlassian.plugin.servlet.util.LazyLoadedReference$InitializationException:
java.lang.NoClassDefFoundError: com/atlassian/jira/gadgets/system/AbstractResource
at com.atlassian.plugin.servlet.util.LazyLoadedReference.get(LazyLoadedReference.java:94)
...

```

In this case, the class using `AbstractResource` class was in the wrong package. Unfortunately, the name of the actual class in the plugin that required the missing class does not appear in the log file.

A More Complex Example

```

2010-02-23 16:20:22,917 http-8080-Processor25 ERROR admin 58822x1x1 17hs118
http://localhost:8080/rest/example/1.0/myplugin
[atlassian.plugin.servlet.DefaultServletModuleManager] Unable to create filter
com.atlassian.plugin.servlet.util.LazyLoadedReference$InitializationException:
java.lang.NoClassDefFoundError: com/atlassian/jira/gadgets/system/SearchQueryBackedResource
at com.atlassian.plugin.servlet.util.LazyLoadedReference.get(LazyLoadedReference.java:94)
...

```

In this case, the `SearchQueryBackedResource` class was not exported by the `atlassian-plugin.xml` file. The workaround is to copy the required class (and all the classes it requires in turn) to your gadget source tree. Alternatively [JRA-18986](#) recommends creating a simple class in your plugin that wraps the core JIRA class and just calls `super()`.

Symptom: "*UnsatisfiedDependencyException: Error creating bean with name*", "*Unsatisfied dependency expressed through constructor argument with index 1*"

Cause: the class is unknown to the OSGI plugin

Solution: add a `component-import` element in the `atlassian-plugin.xml` file to import the necessary class. See the [Component Import](#) plugin module for an example of what this element does.

Other useful introductions include <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/OSGi,+Spring+and+the+Plugin+Framework> and <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Automatic+Generation+of+Spring+Configuration>.

Example:

```

2010-02-23 14:15:46,831 http-8080-Processor25 ERROR admin 51345x1x1 1hageqe
http://localhost:8080/rest/example/1.0/myplugin
[atlassian.plugin.servlet.DefaultServletModuleManager] Unable to create filter
com.atlassian.plugin.servlet.util.LazyLoadedReference$InitializationException:
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name
'com.example.jira.plugins.MyPlugin':
Unsatisfied dependency expressed through constructor argument with index 1 of type
[com.atlassian.sal.api.user.UserManager]: :
No unique bean of type [com.atlassian.sal.api.user.UserManager] is defined:
Unsatisfied dependency of type [interface com.atlassian.sal.api.user.UserManager]: expected at
least 1 matching bean;
nested exception is org.springframework.beans.factory.NoSuchBeanDefinitionException: No unique
bean of type [com.atlassian.sal.api.user.UserManager] is defined:
Unsatisfied dependency of type [interface com.atlassian.sal.api.user.UserManager]: expected at
least 1 matching bean
at com.atlassian.plugin.servlet.util.LazyLoadedReference.get(LazyLoadedReference.java:94)
...

```

Note that the first argument to the constructor has index 0, not index 1.

Add this to `atlassian-plugin.xml` so that the plugin knows about the missing class:

```
<component-import key="userManager" interface="com.atlassian.sal.api.user.UserManager" />
```

A good rule of thumb is to be careful with any class whose package doesn't start with `com.atlassian.jira`

Beyond DI

If using a `component-import` doesn't work you may be able to instantiate the object using `ManagerFactory.getMyObject()` or `ComponentManager.getInstance().getMyObject()`, or even

```
ComponentManager.getInstance().getComponentInstanceOfType(MyObject.class).
```

Symptom: "ClassNotFoundException"

Cause: the class is unknown to the OSGI plugin

Solution: add an explicit <Import-Package> element to the plugin-info element in atlassian-plugin.xml

See <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Troubleshooting+a+ClassNotFoundException> for more information.

Example: TODO

Symptom: "UnsatisfiableDependenciesException"

Cause: you can refer to a Pico component in an OSGI plugin by getting the container, but Pico cannot reference OSGI components

Solution: TODO

Example:

```
org.picocontainer.defaults.UnsatisfiableDependenciesException:  
com.atlassian.plugins.tutorial.workflow.sla.condition.TimeElapsedSinceCreationCondition doesn't  
have any satisfiable constructors.  
Unsatisfiable dependencies: [[interface  
com.atlassian.plugins.tutorial.workflow.sla.util.SLATimestampUtils]]
```

Symptom: "BeanCreationException: Error creating bean with name", "java.lang.LinkageError: loader constraints violated"

Cause: the class has already been loaded from different location than the plugin

Solution: see <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Troubleshooting+a+LinkageError>

Example:

```
2010-02-23 13:29:51,421 http-8080-Processor25 ERROR admin 48590x7x1 b0gp1r  
http://localhost:8080/rest/gadget/1.0/stats/generate  
[atlassian.plugin.servlet.DefaultServletModuleManager] Unable to create filter  
com.atlassian.plugin.servlet.util.LazyLoadedReference$InitializationException:  
org.springframework.beans.factory.BeanCreationException: Error creating bean with name  
'com.atlassian.jira.gadgets.stats.IssueTableResource':  
Instantiation of bean failed; nested exception is java.lang.LinkageError: loader constraints  
violated when linking org/apache/log4j/Logger class  
at com.atlassian.plugin.servlet.util.LazyLoadedReference.get(LazyLoadedReference.java:94)  
...
```

In this case the log4j class appears to be the cause of the error. Note that this error will also occur if your plugin has imported but not otherwise used the class that is using the missing class (so beware of import foo.* statements).

Symptom: "PluginException: Unable to instantiate class"

Cause: No Java class file referring to the class exists, so the automatic scanner for injection didn't know it needed to make the class importable

Solution: Explicitly add <component-import> elements or create a subclass with a delegating constructor

Example:

Create+a+New+JIRA+Custom+Field+Type

See also <https://studio.atlassian.com/browse/PLUG-535>

Symptom: The provider class, class com.sun.jersey.core.impl.provider.xml.SAXParserContextProvider, could not be instantiated

Cause: Your pom.xml file likely has a compile (not test) dependency on jersey version 1.1.2-ea

<http://jersey.576304.n2.nabble.com/Jersey-in-Jetty-Exceptions-td3481925.html> suggests that different versions of jersey-core are involved in

this problem.

Solution: Changing the jersey version 1.1.2 to 1.1.0 makes the error go away.

It's not clear what version of jersey is shipped with JIRA.



There is a JIRA Forum discussion about this here.

Resources

<http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Troubleshooting+a+ClassNotFoundException>

<http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Accessing+Classes+from+Another+Plugin>

<http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Troubleshooting+Velocity+in+OSGi>

<http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Creating+your+Plugin+Descriptor#CreatingyourPluginDescriptor-%7B%7Burl>

Cannot start plugin

Symptom: "Cannot start plugin"

This is a WARNING not an ERROR in the log. The plugin will also be shown as Disabled on the Admin, Plugins screen.

Cause: the OSGI plugin does not know anything about a class used when the plugin is started up

Solution: correct any typos in the interface attribute in the component-import element. Also make sure that the component is exported from elsewhere with a <component> element.

Example:

```
2010-02-24 16:18:52,882 main WARN      [plugin.osgi.factory.OsgiPlugin] Unable to enable plugin
'com.example.jira.plugins.example-errors'
com.atlassian.plugin.osgi.container.OsgiContainerException: Cannot start plugin:
com.example.jira.plugins.example-errors
at com.atlassian.plugin.osgi.factory.OsgiPlugin.enableInternal(OsgiPlugin.java:385)
```

Note that the error messages do not show the name of the component that the OSGI container failed to import. In this case the problem was

```
<component-import key="foo" interface="com.example.does.not.exist"/>
```

ClassNotFoundException

Symptom: "ClassNotFoundException"

Cause: the class is unknown to the OSGI plugin

Solution: add an explicit <Import-Package> element to the plugin-info element in atlassian-plugin.xml

See <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Troubleshooting+a+ClassNotFoundException> for more information.

Example: TODO

Compilation failure - package does not exist

Symptom: "Compilation failure: package does not exist"

Cause: the javac compile command did not have one or more required .jar files specified on the classpath

Solution: add a dependency element in pom.xml for the artifact that contains the missing class

Example: package javax.servlet.http does not exist

Add this element to pom.xml:

```

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>2.3</version>
    <scope>provided</scope>
</dependency>

```



How do I find the correct artifact name and version? Best practice is to use the same version as the jar file shipped with the version of JIRA, Confluence etc you are building with.
The provided scope element indicates that the jar file doesn't need to be packaged with the plugin since the main application can already provide it.

java.lang.LinkageError - loader constraints violated

Symptom: "BeanCreationException: Error creating bean with name", "java.lang.LinkageError: loader constraints violated"

Cause: the class has already been loaded from different location than the plugin

Solution: see <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Troubleshooting+a+LinkageError>

Example:

```

2010-02-23 13:29:51,421 http-8080-Processor25 ERROR admin 48590x7x1 b0gplr
http://localhost:8080/rest/gadget/1.0/stats/generate
[atlassian.plugin.servlet.DefaultServletModuleManager] Unable to create filter
com.atlassian.plugin.servlet.util.LazyLoadedReference$InitializationException:
org.springframework.beans.factory.BeanCreationException: Error creating bean with name
'com.atlassian.jira.gadgets.stats.IssueTableResource':
Instantiation of bean failed; nested exception is java.lang.LinkageError: loader constraints
violated when linking org/apache/log4j/Logger class
    at com.atlassian.plugin.servlet.util.LazyLoadedReference.get(LazyLoadedReference.java:94)
...

```

In this case the log4j class appears to be the cause of the error. Note that this error will also occur if your plugin has imported but not otherwise used the class that is using the missing class (so beware of import foo.* statements).

NoClassDefFoundError

Symptom: "NoClassDefFoundError: com/atlassian/jira/gadgets/system/AbstractResource"

Cause: a plugin is lazily instantiated and then fails to find a needed class

Solution: make sure the class and package name are correct and that the class is declared as a component in its atlassian-plugin.xml.

Simple Example:

```

2010-02-23 13:17:10,611 http-8080-Processor19 ERROR admin 47830x17x3 1bvaudl
http://localhost:8080/rest/gadget/1.0/stats/generate
[atlassian.plugin.servlet.DefaultServletModuleManager] Unable to create filter
com.atlassian.plugin.servlet.util.LazyLoadedReference$InitializationException:
java.lang.NoClassDefFoundError: com/atlassian/jira/gadgets/system/AbstractResource
    at com.atlassian.plugin.servlet.util.LazyLoadedReference.get(LazyLoadedReference.java:94)
...

```

In this case, the class using AbstractResource class was in the wrong package. Unfortunately, the name of the actual class in the plugin that required the missing class does not appear in the log file.

A More Complex Example

```
2010-02-23 16:20:22,917 http-8080-Processor25 ERROR admin 58822x1x1 17hs118
http://localhost:8080/rest/example/1.0/myplugin
[atlassian.plugin.servlet.DefaultServletModuleManager] Unable to create filter
com.atlassian.plugin.servlet.util.LazyLoadedReference$InitializationException:
java.lang.NoClassDefFoundError: com/atlassian/jira/gadgets/system/SearchQueryBackedResource
at com.atlassian.plugin.servlet.util.LazyLoadedReference.get(LazyLoadedReference.java:94)
...
```

In this case, the `SearchQueryBackedResource` class was not exported by the `atlassian-plugin.xml` file. The workaround is to copy the required class (and all the classes it requires in turn) to your gadget source tree. Alternatively [JRA-18986](#) recommends creating a simple class in your plugin that wraps the core JIRA class and just calls `super()`.

PluginException

Symptom: "PluginException: Unable to instantiate class"

Cause: No Java class file referring to the class exists, so the automatic scanner for injection didn't know it needed to make the class importable

Solution: Explicitly add `<component-import>` elements or create a subclass with a delegating constructor

Example:

Create+a+New+JIRA+Custom+Field+Type

See also <https://studio.atlassian.com/browse/PLUG-535>

SAXParserContextProvider

Symptom: The provider class, class `com.sun.jersey.core.impl.provider.xml.SAXParserContextProvider`, could not be instantiated

Cause: Your `pom.xml` file likely has a compile (not test) dependency on jersey version 1.1.2-ea
<http://jersey.576304.n2.nabble.com/Jersy-in-Jetty-Exceptions-td3481925.html> suggests that different versions of jersey-core are involved in this problem.

Solution: Changing the jersey version 1.1.2 to 1.1.0 makes the error go away.

It's not clear what version of jersey is shipped with JIRA.



There is a JIRA Forum discussion about this here.

UnsatisfiableDependenciesException

Symptom: "UnsatisfiableDependenciesException"

Cause: you can refer to a Pico component in an OSGI plugin by getting the container, but Pico cannot reference OSGI components

Solution: TODO

Example:

```
org.picocontainer.defaults.UnsatisfiableDependenciesException:
com.atlassian.plugins.tutorial.workflow.sla.condition.TimeElapsedSinceCreationCondition doesn't
have any satisfiable constructors.
Unsatisfiable dependencies: {[interface
com.atlassian.plugins.tutorial.workflow.sla.util.SLATimestampUtils]}
```

UnsatisfiedDependencyException - Error creating bean with name

Symptom: "UnsatisfiedDependencyException: Error creating bean with name", "Unsatisfied dependency expressed through constructor argument with index 1"

Cause: the class is unknown to the OSGI plugin

Solution: add a `<component-import>` element in the `atlassian-plugin.xml` file to import the necessary class. See the [Component Import](#) plugin module for an example of what this element does.

Other useful introductions include <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/OSGi,+Spring+and+the+Plugin+Framework> and <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Automatic+Generation+of+Spring+Configuration>.

Example:

```
2010-02-23 14:15:46,831 http-8080-Processor25 ERROR admin 51345x1x1 lhageqe
http://localhost:8080/rest/example/1.0/myplugin
[atlassian.plugin.servlet.DefaultServletModuleManager] Unable to create filter
com.atlassian.plugin.servlet.util.LazyLoadedReference$InitializationException:
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name
'com.example.jira.plugins.MyPlugin':
Unsatisfied dependency expressed through constructor argument with index 1 of type
[com.atlassian.sal.api.user.UserManager]: :
No unique bean of type [com.atlassian.sal.api.user.UserManager] is defined:
Unsatisfied dependency of type [interface com.atlassian.sal.api.user.UserManager]: expected at
least 1 matching bean;
nested exception is org.springframework.beans.factory.NoSuchBeanDefinitionException: No unique
bean of type [com.atlassian.sal.api.user.UserManager] is defined:
Unsatisfied dependency of type [interface com.atlassian.sal.api.user.UserManager]: expected at
least 1 matching bean
at com.atlassian.plugin.servlet.util.LazyLoadedReference.get(LazyLoadedReference.java:94)
...
...
```

Note that the first argument to the constructor has index 0, not index 1.

Add this to `atlassian-plugin.xml` so that the plugin knows about the missing class:

```
<component-import key="userManager" interface="com.atlassian.sal.api.user.UserManager"/>
```

A good rule of thumb is to be careful with any class whose package doesn't start with `com.atlassian.jira`

Beyond DI

If using a `<component-import>` doesn't work you may be able to instantiate the object using `ManagerFactory.getMyObject()` or `ComponentManager.getInstance().getMyObject()`, or even `ComponentManager.getInstance().getComponentInstanceOfType(MyObject.class)`.

Atlassian JavaDoc

Atlassian provides JavaDoc for all of our code. Atlassian's JavaDoc is available in two ways:

1. **From the Maven Repository:** All of our released libraries have JavaDoc artifacts deployed to our [Atlassian Maven Repositories](#).
2. **From Atlassian Docs:** We publish all of the JavaDoc from our released modules on <http://docs.atlassian.com>. You can browse them there.

Plugin Module Index

- What this is
 - Modules common to all applications
 - Code sharing
 - Component
 - Component Import
 - JEE container integration
 - Servlet Context Listener
 - Servlet Context Parameter
 - Servlet Filter
 - Servlet
 - User interface
 - Web Item
 - Web Section
 - Web Panel
 - Web Panel Renderer
 - Web Resource
 - Web Resource Transformer

- Other types
 - REST
 - Gadget
- Product-specific modules
- JIRA
 - Adding custom remote APIs
 - SOAP
 - XML-RPC
 - Adding operations to tabs, views or screens
 - Project tab panel
 - Component tab panel
 - Version tab panel
 - Issue tab panel
 - Search request view
 - Custom workflow operations
 - Workflow conditions
 - Workflow validators
 - Workflow functions
 - Custom fields
 - Custom field types
 - Custom field searchers
 - Other types
 - Reports
 - Custom actions
 - JQL functions
- Confluence
 - Custom remote APIs
 - SOAP
 - XML-RPC
 - Custom markup
 - User macros
 - Custom macros
 - Code formatting
 - System tasks
 - Job
 - Lifecycle
 - Triggers
 - Look and feel
 - Decorators
 - Editors
 - Language
 - Theme
 - Keyboard shortcuts
 - Custom actions
 - XWork/WebWork
 - Other types
 - Search index extractor
 - Path converters
 - Velocity context

What this is

This page lists the plugin modules for Atlassian applications. The idea is for the plugin developer to quickly identify the plugin module they need and give them an outline of what to expect and what they can change.

Each plugin module listed here comes with:

- A brief description of what it's for
- Glosses for the crucial configuration elements
- An example of its use in `atlassian-plugin.xml`

The name of each module is a link to its full description in the product development hub. If you decide to write one of the modules listed below, you **must** read the full description at the link to succeed.

Modules common to all applications

These modules are provided by the plugin framework and operate identically in all applications.

Code sharing

The Component and Component Import module types are for modularizing and sharing code within a plugin and between plugins:

Component

Declares that a class in the plugin is available for dependency injection within that plugin, and optionally, available for dependency injection in other plugins in the application.

Frequently used attributes:

- **class**: The class providing the implementation for the component to inject.
- **public**: Whether this implementation is available for dependency injection into other plugins, via the [Component Import](#) declaration.

Frequently used elements:

- **interface**: The interface for the component to inject.

Example 1

```
<!-- declare a component for injection in this plugin -->
<component
    key="helloWorldService"
    class="com.myapp.DefaultHelloWorldService">
    <description>Provides hello services.</description>
    <interface>com.myapp.HelloWorldService</interface>
</component>
```

Example 2

```
<!-- declare a component shared with other plugins -->
<component
    key="goodbyeWorldService"
    class="com.myapp.DefaultGoodbyeWorldService"
    public="true">
    <description>Provides goodbye services.</description>
    <interface>com.myapp.GoodbyeWorldService</interface>
</component>
```

Component Import

Requests that some implementation of a specific interface, exported either by another plugin using a [Component](#) declaration or by the host application, be made available for dependency injection into this plugin.

Frequently used attributes:

- **interface**: The interface for the component to inject.

Frequently used elements:

- **interface**: The interface for the component to inject (identical to the `interface` attribute).

Example

```
<!-- import a component shared by another plugin -->
<component-import
    key="helloWorldService"
    interface="com.myapp.HelloWorldService">
    <description>Provides hello services.</description>
</component>
```

JEE container integration

Servlet Context Listener

Deploys the specified class as a Java EE servlet context listener.

Frequently used attributes:

- **class**: The class to use as a servlet context listener.

Example

```
<!-- declare a custom servlet context listener -->
<servlet-context-listener
    key="myServletContextListener"
    interface="com.myapp.MyServletContextListener">
    <description>Custom listener for the servlet context in this application.</description>
</servlet-context-listener>
```

Servlet Context Parameter

Sets the specified name and value as a parameter in the Java EE servlet context.

Frequently used elements:

- param-name: The name of the parameter.
- param-value: The value of the parameter.

Example

```
<!-- declare a custom servlet context parameter -->
<servlet-context-param
    key="myServletContextParam">
    <description>Custom parameter for the servlet context in this application.</description>
    <param-name>myParam</param-name>
    <param-value>myParamValue</param-value>
</servlet-context-param>
```

Servlet Filter

Deploys the specified Java EE servlet filter in the product.

Frequently used attributes:

- class: The class to use as a servlet filter,

Frequently used elements:

- url-pattern: The URL pattern this servlet filter should be applied to.

Example

```
<!-- declare a custom servlet filter -->
<servlet-filter
    key="myServletFilter"
    class="com.myapp.MyServletFilter">
    <description>Custom parameter for the servlet context in this application.</description>
    <url-pattern>/myapp</url-pattern>
</servlet-filter>
```

Servlet

Deploys the specified Java EE servlet in the product.

Frequently used attributes:

- class: The servlet class to use.

Frequently used elements:

- url-pattern: The URL pattern this servlet should be deployed to.

Example

```
<!-- declare a custom servlet filter -->
<servlet
    key="myServletFilter"
    class="com.myapp.MyServlet">
<description>Servlet to deploy in this application.</description>
<url-pattern>/myapp</url-pattern>
</servlet>
```

User interface

Web Item

Defines a new link in an application menu.

Frequently used attributes:

- **section:** The [web section](#) to insert this web item into.

Frequently used elements:

- **label:** Points to an internationalized resource property that contains the display text of the link.
- **link:** Defines where the link should point.

Example

```
<!-- declare a web item -->
<web-item
    key="myWebItem"
    section="system.admin/example1">
<description>Example link appearing in the system admin menu.</description>
<label key="atlassian.home"/>
<link>http://atlassian.com</link>
</web-item>
```

Web Section

Defines a new section in an application menu.

Frequently used attributes:

- **section:** The [web section](#) to insert this web section into.
- **weight:** The order in which this section should appear in the menu relative to other items; heavier weights display lower in the menu.

Frequently used elements:

- **label:** Points to an internationalized resource property that contains the user-visible name of this section.

Example

```
<!-- declare a web section -->
<web-section
    key="myWebItem"
    section="system.admin/example1"
    weight="100">
<description>Example web section appearing in the system admin menu.</description>
<label key="example1.web.section.name"/>
</web-section>
```

Web Panel

Defines a web panel – a set of HTML content that can be inserted verbatim into a page.

Frequently used attributes:

- **location:** The product-specific location to insert this web section. Locations are different between products.

Frequently used elements:

- **resource:** The HTML resource to use for the panel's contents.

Example

```
<!-- declare a web panel -->
<web-panel
    key="myWebPanel"
    location="atl.confluence.comments">
    <description>Example web panel appearing in the Confluence comments.</description>
    <resource name="view" type="velocity" location="templates/comments-web-panel.vm"/>
</web-panel>
```

Web Panel Renderer

Defines a custom rendering engine for a <web-panel>.

Frequently used attributes:

- **class:** The class that implements the web panel renderer.

Example

```
<!-- web panel renderer example -->
<web-panel-renderer
    key="myWebPanelRenderer"
    class="com.myapp.FreemarkerWebPanelRenderer" />
```

Web Resource

Defines downloadable resources (files) for a plugin, such as CSS or JavaScript files.

Frequently used elements:

- **resource:** Specifies the resources that should be made available for download.

Example

```
<!-- web resource example -->
<web-resource
    key="myWebResources">
    <resource name="mylib.js" type="download" location="js/mylib.js"/>
    <resource name="mylib2.js" type="download" location="js/mylib2.js"/>
</web-resources>
```

Web Resource Transformer

Defines transformers which allow changing web resources before being served to the browser.

Frequently used attributes:

- **class:** The class to create and use as a transformer.

Example

```
<!-- web resource transformer example -->
<web-resource-transformer
    key="template"
    class="com.atlassian.labs.template.TemplateTransformer" />
```

Other types

REST

Exposes data and services as REST resources.

Frequently used attributes:

- path: Path to the API defined by these resources.
- version: Version of the REST API being exposed at this path.

Example

```
<!-- REST module example -->
<rest
    key="helloWorldRest"
    path="/helloworld"
    version="1.0">
    <description>Provides hello world services.</description>
</rest>
```

Gadget

Defines an [Atlassian Gadget](#) provided by this plugin.

Frequently used attributes:

- location: Path to the gadget XML specification in this plugin.

Example

```
<!-- gadget module example -->
<gadget
    key="myGadget"
    location="gadgets/public/myGadget.xml"/>
```

Product-specific modules

These modules are unique to the product they originate in.

JIRA

Adding custom remote APIs

SOAP

Adds custom SOAP services to JIRA in addition to the [builtin SOAP services](#).

Frequently used attributes:

- class: The SOAP service implementing the published interface (see below).

Frequently used elements:

- service-path: The path (under /rpc/soap) the service will be published at.
- published-interface: The interface implemented by the service class and exposed to the end user.

Example

```
<rpc-soap
    key="mySoapService"
    class="com.myapp.MySoapServiceImpl">
    <description>Custom SOAP services for this installation.</description>
    <service-path>customsoap-v1</service-path>
    <published-interface>
        com.myapp.MySoapService
    </published-interface>
</rpc-soap>
```

XML-RPC

Adds custom XML-RPC services to JIRA in addition to the builtin XML-RPC service.

Frequently used attributes:

- `class`: The class to publish as an XML-RPC service

Frequently used elements:

- `service-path`: The path (under `/rpc/xmlrpc`) the service will be published at.

Example

```
<rpc-xmlrpc
    key="myXmlRpcService"
    class="com.myapp.MyXmlRpcService">
    <description>Custom XML-RPC services for this installation.</description>
    <service-path>custom-xmlrpc</service-path>
</rpc-xmlrpc>
```

Adding operations to tabs, views or screens

Project tab panel

Adds new tabs to the Browse Projects page.

Frequently used attributes:

- `class`: The class implementing the project tab panel.

Frequently used elements:

- `label`: The user-visible name of the label on the page. Can be specified directly in the element content or picked up from the i18n resource by attribute `key`.
- `order`: The order in which the label should appear in the menu.
- `resource (velocity)`: The template which renders the HTML for the project tab.
- `resource (i18n)`: The .properties file containing i18n values for user-visible text.

Example

```
<project-tabpanel key="roadmap-panel" name="Road Map Panel"
    class="com.atlassian.jira.plugin.projectpanel.impl.VersionsProjectTabPanel">
    <description key="projectpanels.roadmap.description">
        A roadmap of the upcoming versions in this project.
    </description>
    <label key="common.concepts.roadmap" />
    <order>20</order>
    <resource type="velocity" name="view"
        location="templates/plugins/jira/projectpanels/roadmap-panel.vm" />
    <resource type="i18n" name="i18n"
        location="com.atlassian.jira.plugins.projectpanels.roadmap" />
</project-tabpanel>
```

Component tab panel

Adds new tabs to the Browse Component page.

Frequently used attributes:

- `class`: The class implementing the component tab panel.

Frequently used elements:

- `label`: The user-visible name of the label on the page. Can be specified directly in the element content or picked up from the i18n resource by attribute `key`.
- `order`: The order in which the label should appear in the menu.
- `resource (velocity)`: The template which renders the HTML for the component tab.
- `resource (i18n)`: The .properties file containing i18n values for user-visible text.

Example

```
<!-- module example -->
<component-tabpanel
    key="component-openissues-panel"
    i18n-name-key="componentpanels.openissues.name"
    name="Open Issues Panel"
    class="com.atlassian.jira.plugin.componentpanel.impl.GenericTabPanel">
    <description
        key="componentpanels.openissues.description">
        Show the open issues for this component.
    </description>
    <label key="common.concepts.openissues" />
    <order>10</order>
    <resource type="velocity" name="view"
        location="templates/plugins/jira/projectentitypanels/openissues-component-panel.vm" />
    <resource type="i18n" name="i18n"
        location="com.atlassian.jira.plugins.componentpanels.openissues" />
</component-tabpanel>
```

Version tab panel

Adds new tabs to the Browse Version page.

Frequently used attributes:

- **class:** The class implementing the version tab panel.

Frequently used elements:

- **label:** The user-visible name of the label on the page. Can be specified directly in the element content or picked up from the i18n resource by attribute key.
- **order:** The order in which the label should appear in the menu.
- **resource (velocity):** The template which renders the HTML for the version tab.
- **resource (i18n):** The .properties file containing i18n values for user-visible text.

Example

```
<!-- module example -->
<version-tabpanel
    key="version-openissues-panel"
    i18n-name-key="versionpanels.openissues.name"
    name="Open Issues Panel"
    class="com.atlassian.jira.plugin.versionpanel.impl.GenericTabPanel">
    <description key="versionpanels.openissues.description">Show the open issues for this
version.</description>
    <label key="common.concepts.openissues" />
    <order>10</order>
    <resource type="velocity" name="view"
        location="templates/plugins/jira/projectentitypanels/openissues-version-panel.vm" />
    <resource type="i18n" name="i18n"
        location="com.atlassian.jira.plugins.versionpanels.openissues" />
</version-tabpanel>
```

Issue tab panel

Adds new tabs to the View Issue panel.

Frequently used attributes:

- **{class}:** The class implementing the issue tab panel.

Frequently used elements:

- **label:** The user-visible name of the label on the page.
- **resource:** The template which renders the HTML for the issue tab.

Example

```
<!-- issue tab panel example -->
<issue-tabpanel
    key="environment-tabpanel"
    name="Environment Tab Panel"
    class="com.icontact.jira.environmentmanor.issuetabpanels.EnvironmentPanelPlugin">
    <description>Show Environment Status and controls in an issue tab panel.</description>
    <label>Environment Control Panel</label>
    <resource type="velocity" name="view"
        location="templates/plugins/environmentmanor/issuetabpanels/environment.vm"/>
</issue-tabpanel>
```

Search request view

Allows display of search results in the [issue navigator](#) in custom formats (such as XML or MS Excel format).

Frequently used attributes:

- **class:** The class implementing the search request view.
- **contentType:** The MIME content type to return when displaying this view.

Frequently used elements:

- **resource (header):** The header to render for this view.
- **resource (singleissue):** What to render for each individual issue in this view.
- **resource (footer):** The footer to render for this view.

Example

```
<!-- search request view example -->
<search-request-view
    key="simple-searchrequest-xml"
    name="Simple XML"
    class="com.atlassian.jira.sample.searchrequest.SimpleSearchRequestXmlView"
    state='enabled'
    fileExtension="xml"
    contentType="text/xml">
    <resource type="velocity" name="header" location="templates/searchrequest-xml-header.vm"/>
    <resource type="velocity" name="singleissue"
    location="templates/searchrequest-xml-singleissue.vm"/>
    <resource type="velocity" name="footer" location="templates/searchrequest-xml-footer.vm"/>
    <order>100</order>
</search-request-view>
```

Custom workflow operations

Workflow conditions

Checks whether a user can perform a [workflow transition](#).

Frequently used attributes:

- **class:** The class implementing the workflow condition (usually a builtin product class rather than one the plugin developer writes)

Frequently used elements:

- **condition-class:** The class implementing the condition's logic.
- **resource:** Renders the view for the condition.

Example

```
<!-- workflow condition example -->
<workflow-condition key="onlyassignee-condition"
    name="Only Assignee Condition"
    i18n-name-key="admin.workflow.condition.onlyassignee.display.name"
    class="com.atlassian.jira.plugin.workflow.WorkflowAllowOnlyAssigneeConditionFactoryImpl">
    <description key="admin.workflow.condition.onlyassignee">Condition to allow only the assignee
to execute a transition.</description>
    <condition-class>com.atlassian.jira.workflow.condition.AllowOnlyAssignee</condition-class>
    <resource type="velocity" name="view"

        location="templates/jira/workflow/com/atlassian/jira/plugin/onlyassignee-condition-view.vm"/>
</workflow-condition>
```

Workflow validators

Checks that the data supplied to a workflow transition is valid.

Frequently used attributes:

- **class**: The class implementing the workflow validator (usually a builtin product class rather than one the plugin developer writes)

Frequently used elements:

- **validator-class**: The class implementing the validator's logic.
- **resource**: Renders the view for the condition.

Example

```
<!-- workflow validator example -->
<workflow-validator
    key="permission-validator"
    name="Permission Validator"
    class="com.atlassian.jira.plugin.workflow.WorkflowPermissionValidatorPluginFactory">
    <description>Validates that the user has a permission.</description>
    <validator-class>
        com.atlassian.jira.workflow.validator.PermissionValidator
    </validator-class>
    <resource type="velocity" name="view"
        location="templates/jira/.../permission-validator-view.vm"/>
    <resource type="velocity" name="input-parameters"
        location="templates/jira/.../permission-validator-input-params.vm"/>
</workflow-validator>
```

Workflow functions

Performs actions after a workflow transition has executed.

Frequently used attributes:

- **class**: The class implementing the workflow function (usually a builtin product class rather than one the plugin developer writes)

Frequently used elements:

- **function-class**: The class implementing the function's logic.
- **resource**: Renders the view for the condition.

Example

```
<!-- workflow function example -->
<workflow-function key="update-issue-field-function"
    name="Update Issue Field"
    class="com.atlassian.jira.plugin.workflow.UpdateIssueFieldFunctionPluginFactory">
    <description>Updates a simple issue field to a given value.</description>
    <function-class>
        com.atlassian.jira.workflow.function.issue.UpdateIssueFieldFunction
    </function-class>
    <orderable>true</orderable>
    <unique>false</unique>
    <deletable>true</deletable>
    <resource type="velocity" name="view"
        location="templates/jira/.../update-issue-field-function-view.vm"/>
    <resource type="velocity" name="input-parameters"
        location="templates/jira/.../update-issue-field-function-input-params.vm"/>
</workflow-function>
```

Custom fields

Custom field types

Defines a custom field and its type.

Frequently used attributes:

- **class**: The class that implements the custom field and its behaviors.

Frequently used elements:

- **resource (view)**: Template for rendering this custom field on the View Issue page.
- **resource (edit-userpicker)**: Template for rendering this custom field on the Create/Edit Issue page.
- **resource (xml)**: Template for rendering this custom field in XML.

Example

```
<!-- custom field type example -->
<customfield-type key="userpicker" name="User Picker"
    class="com.atlassian.jira.issue.customfields.impl.UserCFType">
    <description>
        Choose a user from the user base via a popup picker window.
    </description>
    <resource type="velocity" name="view"
        location="templates/plugins/fields/view-user.vm" />
    <resource type="velocity" name="edit"
        location="templates/plugins/fields/edit-userpicker.vm" />
    <resource type="velocity" name="xml"
        location="templates/plugins/fields/xml-user.vm" />
</customfield-type>
```

Custom field searchers

Defines a search method for indexing a custom field.

Frequently used attributes:

- **class**: The class implementing this custom field searcher.

Frequently used elements:

- **valid-customfield-type**: Defines the custom field this searcher applies to.
- **resource**: Renders this searcher on the issue navigator search form.

Example

```
<!-- custom field searcher example -->
<customfield-searcher key="userpickersearcher" name="User Picker Searcher"
class="com.atlassian.jira.issue.customfields.searchers.UserPickerSearcher">
    <description>
        Allow to search for a user using a userpicker.
    </description>
    <resource type="velocity" name="search"
        location="templates/plugins/fields/search-userpicker.vm" />
    <valid-customfield-type
        package="com.atlassian.jira.plugin.system.customfieldtypes" key="userpicker" />
</customfield-searcher>
```

Other types

Reports

Defines a report in JIRA.

Frequently used attributes:

- class: The class implementing this report.

Frequently used elements:

- label: The user-visible name of the label on the page.
- resource (view): Renders the HTML version of the report.
- resource (i18n): Provides the i18n values for the keys in the report displays.
- properties: Properties this report requires to run correctly.

Example

```
<!-- report example -->
<report key="time-tracking"
    name="Time Tracking Report"
    class="com.atlassian.jira.plugin.report.impl.TimeTrackingReport">
    <description key="report.timetracking.description">
        This report shows the time tracking details for a specific project.
    </description>
    <label key="report.timetracking.label" />
    <resource type="velocity" name="view"
        location="templates/plugins/jira/reports/time-tracking-report.vm" />
    <resource type="i18n" name="i18n"
        location="com.atlassian.jira.plugins.reports.timetracking" />
    <properties>
        <property>
            <key>versionId</key>
            <name>common.concepts.version</name>
            <description>report.timetracking.version.description</description>
            <type>select</type>
            <values class="com.atlassian.jira.portal.VersionOptionalValuesGenerator"/>
        </property>
    </properties>
</report>
```

Custom actions

Defines custom WebWork actions (functionality that can be triggered by visiting a URL) or overrides an existing JIRA action.

Frequently used elements:

- actions: Defines the WebWork actions this module will provide.

Example

```
<!-- webwork example -->
<webwork1
    key="qquserissue"
    name="Quick Create User Issue"
    class="java.lang.Object">
    <actions>
        <action
            name="com.atlassian.jira.toolkit.action.QuickCreateUserIssueAction"
            alias="QuickCreateUserIssue">
            <view name="createuserissue"/>/templates/quickcreateuser.vm</view>
        </action>
    </actions>
</webwork1>
```

JQL functions

Defines new functions for use in the [JIRA Query Language \(JQL\)](#).

Frequently used attributes:

- **class**: The class implementing the JQL function logic.

Frequently used elements:

Example

```
<!-- jql example -->
<jql-function
    key="role-members"
    i18n-name-key="rolefunc.name"
    name="Role Members Function"
    class="com.atlassian.example.jira.jqlfunc.RoleFunction">
    <resource type="i18n" name="i18n"
        location="com.atlassian.example.jira.jqlfunc.RoleFunction" />
    <description key="rolefunc.description">
        JQL function to return the members of a particular role
    </description>
</jql-function>
```

Confluence

Custom remote APIs

SOAP

Adds custom SOAP services to Confluence in addition to the [builtin SOAP services](#).

Frequently used attributes:

- **class**: The SOAP service implementing the published interface (see below).

Frequently used elements:

- **service-path**: The path (under /rpc/soap) the service will be published at.
- **published-interface**: The interface implemented by the service class and exposed to the end user.

Example

```
<!-- SOAP example -->
<rpc-soap
    key="mySoapService"
    class="com.myapp.MySoapServiceImpl">
    <description>Custom SOAP services for this installation.</description>
    <service-path>customsoap-v1</service-path>
    <published-interface>
        com.myapp.MySoapService
    </published-interface>
</rpc-soap>
```

XML-RPC

Adds custom XML-RPC services to Confluence in addition to the [builtin XML-RPC service](#).

Frequently used attributes:

- **class**: The class to publish as an XML-RPC service

Frequently used elements:

- **service-path**: The path (under /rpc/xmlrpc) the service will be published at.

Example

```
<rpc-xmlrpc
    key="myXmlRpcService"
    class="com.myapp.MyXmlRpcService">
    <description>Custom XML-RPC services for this installation.</description>
    <service-path>custom-xmlrpc</service-path>
</rpc-xmlrpc>
```

Custom markup

User macros

Defines simple [user macros](#) as plugin modules without requiring any new Java code.

Frequently used elements:

- **template**: The body of the user macro (in HTML). Can access the Velocity context.

Example

```
<!-- user macro example -->
<user-macro
    name='helloworld'
    key='helloworld'>
    <description>Hello, user macro</description>
    <template><![CDATA[Hello, $body!]]></template>
</user-macro>
```

Custom macros

Defines a [macro](#) -- a piece of code that can be invoked from inside a page. Usually this code is replaced in the rendered page by its output.

Frequently used attributes:

- **class**: The class implementing the macro

Frequently used elements:

Example

```
<!-- macro example -->
<macro
    name='tasklist'
    class='com.atlassian.confluence.extra.tasklist.TaskListMacro'
    key='tasklist'>
    <description>Creates a very simple task list, with user checkable tasks</description>
</macro>
```

Code formatting

Adds support for new languages to the [builtin code macro](#).

Frequently used attributes:

- **class**: Class implementing the new code formatter.

Example

```
<!-- code formatter example -->
<codeformatter
    name="ruby"
    key="ruby"
    class="com.example.confluence.formatters.RubyFormatter">
    <description>Code formatter for the Ruby programming language</description>
</codeformatter>
```

System tasks

Job

Adds repeatable tasks to Confluence which can be scheduled by [triggers].

Frequently used attributes:

- **class**: Class implementing the job.

Example

```
<!-- job example -->
<job
    key="myJob"
    name="My Job"
    class="com.example.myplugin.jobs.MyJob"/>
```

Lifecycle

Adds tasks to be run on Confluence startup and shutdown.

Frequently used attributes:

- **class**: Implements the lifecycle module for startup or shutdown.

Example

```
<!-- lifecycle example -->
<lifecycle
    key="frobozz"
    name="Frobozz Service"
    class="com.example.frobozz.Lifecycle"
    sequence="1200">
    <description>Start and stop the Frobozz service</description>
</lifecycle>
```

Triggers

Schedules jobs to run.

Frequently used elements:

- job: The key of the job module to schedule.
- schedule: When to run the job. Can be expressed as a cron job or by repeat intervals.

Example

```
<!-- trigger example -->
<trigger
    key="myTrigger"
    name="My Trigger">
    <job key="myJob" />
    <schedule
        repeat-interval="3600000"
        repeat-count="5" />
</trigger>
```

Look and feel

Decorators

Allows the user to add Sitemesh Velocity [decorators](#) around Confluence pages.

Frequently used attributes:

- page: Name of the Velocity template to render as the decorator.

Frequently used elements:

- pattern: The URL pattern for which pages should have this decorator applied.

Example

```
<!-- decorator example -->
<decorator
    name="myDecorator"
    page="myDecorator.vmd"
    key="myDecorator">
    <description>My sample decorator.</description>
    <pattern>/plugins/sampleplugin/*</pattern>
</decorator>
```

Editors

Defines a WYSIWYG editor for Confluence.

Frequently used attributes:

- class: The class implementing the interface between Confluence and the editor's code.

Example

```
<!-- editor example -->
<editor
    name="tinymceeditor"
    class="com.atlassian.confluence.extra.tinymceplugin.TinyMceEditor"
    key="tinymceeditor">
    <description>TinyMCE Editor</description>
</editor>
```

Language

Defines new languages for the Confluence UI.

Frequently used attributes:

- **language**: The language defined by this module (must exist in java.util.Locale)
- **country**: The country this language is intended for

Frequently used elements:

- **resource (download)**: Defines a flag image to show for this resource.

| Example |
|--|
| <pre><!-- language example --> <language name="German" key="de_DE" language="de" country="DE"> <resource name="de_DE.gif" type="download" location="templates/languages/de_DE/de_DE.gif"> <property key="content-type" value="image/gif"/> </resource> </language></pre> |

Theme

Defines a new theme (of stylesheets and images) for Confluence.

Frequently used attributes:

- **class**: The class implementing the theme. (Don't change this.)

Frequently used elements:

- **resource (download)**: CSS stylesheets implementing the theme (and images)
- **resource (icon)**: Theme icon image used in the Choose Theme menu.

| Example |
|--|
| <pre><!-- theme example --> <theme key="simple-theme" name="Simple Demo Theme" class="com.atlassian.confluence.themes.BasicTheme"> <resource type="download" name="default-theme.css" location="/includes/css/default-theme.css"/> <resource type="download" name="image-theme.css" location="image-theme.css"/> <resource type="download" name="home-16.png" location="home-16.png"/> <resource key="icon" name="themeicon.gif" type="download" location="your-theme-icon.gif"/> </theme></pre> |

Keyboard shortcuts

Defines a keyboard shortcut within Confluence.

Frequently used elements:

- **order**: The order in which this shortcut appears in the Keyboard Shortcuts dialog box.
- **description**: A human-readable description of this shortcut.
- **shortcut**: The keyboard shortcut keystroke sequence.

- **operation:** The target of the keyboard shortcut.
- **context:** Which pages this shortcut will be active on.

Example

```
<!-- keyboard shortcut example -->
<keyboard-shortcut
    key="goto.space"
    i18n-name="admin.keyboard.shortcut.goto.space.name"
    name="Browse Space">
    <order>20</order>
    <description key="admin.keyboard.shortcut.goto.space.desc">Browse Space</description>
    <shortcut>gs</shortcut>
    <operation type="followLink">#space-pages-link</operation>
    <context>global</context>
</keyboard-shortcut>
```

Custom actions

XWork/WebWork

Defines new XWork/WebWork actions, adding URL-addressable functionality to Confluence.

Frequently used elements:

The body of the `<xwork>` element contains XWork action markup. An example is below.

Example

```
<!-- xwork example -->
<xwork
    name="livesearchaction"
    key="livesearchaction">
    <package
        name="livesearch"
        extends="default"
        namespace="/plugins/livesearch">
        <default-interceptor-ref name="defaultStack" />

        <action name="livesearch"
            class="com.atlassian.confluence.extra.livesearch.LiveSearchAction">
            <result name="success" type="velocity">
                /templates/extra/livesearch/livesearchaction.vm
            </result>
        </action>
    </package>
</xwork>
```

Other types

Search index extractor

Defines an extractor for adding information to the Confluence search index.

Frequently used attributes:

- **class:** The class implementing the extractor.
- **priority:** The order in which the extractor is run; lower-priority extractors run first.

Example

```
<!-- extractor example -->
<extractor
    name="Page Metadata Extractor"
    key="pageMetadataExtractor"
    class="com.atlassian.confluence.extra.extractor.PageMetadataExtractor"
    priority="1000">
    <description>Extracts certain keys from a page's metadata and adds them to the search index.</description>
</extractor>
```

Path converters

Defines path converters which provide custom path mapping in a plugin.

Frequently used attributes:

- **class**: The class implementing the path converter.
- **weight**: The order in which this path converter is executed relative to any other converters.

Example

```
<!-- path converter example -->
<path-converter
    weight="10"
    key="example-converter"
    class="com.mycompany.confluence.plugin.ExamplePathConverter"/>
```

Velocity context

Defines new components to be added to the Confluence Velocity context.

Frequently used attributes:

- **class**: The class representing the component to be added.
- **context-key**: The Velocity variable that will be created to reference this component.

Example

```
<!-- velocity context example -->
<velocity-context-item
    key="myVelocityHelper"
    name="My Plugin's Velocity Helper"
    context-key="myVelocityHelper"
    class="com.example.mybatisplus.helpers.MyVelocityHelper" />
```

Writing your first plugin FAQ

- Adding Selenium to Functional Tests
- Build Failure - Manifest Validation Errors
- Cannot Log In to Confluence using Admin Account
- Choosing a Logging Framework
- Choosing a Package Name
- Eclipse Maven Plugin Build Error
- If you change pom.xml, you may need to restart the atlas-cli
- Instant Loading of Plugin Resources
- Overriding the application's webapp when developing your plugin
- Specifying a particular version of the host application
- Using the Atlassian Plugin SDK with a Source Code License
- Using your own log4j configuration for your plugin

Adding Selenium to Functional Tests

With more and more UI being rendered and processed dynamically with JavaScript, in-browser testing has become a requirement for

functional testing. [Selenium](#) is a popular choice for executing tests within the native browser but still driven by Java. This tip tells you how you can easily add Selenium testing to your plugin.

Step 1 - Add atlassian-selenium-browsers-auto to your pom.xml

Add this XML to the <dependencies> section in your pom.xml:

```
<dependency>
    <groupId>com.atlassian.selenium</groupId>
    <artifactId>atlassian-selenium-browsers-auto</artifactId>
    <version>2.0.0.m3</version>
    <scope>test</scope>
</dependency>
```

Replace "2.0.0.m3" with the latest version.

Step 2 - Write a Selenium test

The Plugin SDK comes with support for integration/functional tests that will run with the target Atlassian application started and your plugin installed. You just have to create one or more tests in the `src/test/java/it` directory. The convention is if your code is in the `com.mycompany.myplugin` package, your integration tests would be in the `it.com.mycompany.myplugin` package or `src/test/java/it/com/mycompany/myplugin` directory.

To use Selenium in these tests, the `atlassian-selenium-browsers-auto` artifact provides a few static helper methods. These methods, when used for the first time:

1. Detect your operating system
2. Install the appropriate browser (FireFox 3.5 is the default) for your OS in `target/seleniumTmp`
3. If Linux and `xfvb.enable` is set to true, start `Xvfb` on an open DISPLAY detected automatically
4. Start the Selenium server
5. Start the newly installed browser

This all happens in the background, so you should just be able to focus on your test.

Here is a simple test that hits the home page and looks for the text of "Hello world":

```
import com.atlassian.selenium.SeleniumClient;
import static com.atlassian.selenium.browsers.AutoInstallClient.assertThat;
import static com.atlassian.selenium.browsers.AutoInstallClient.seleniumClient;
public class TestHelloWorld extends TestCase
{
    public void testHelloWorld() throws Exception
    {
        SeleniumClient client = seleniumClient();
        client.open("/");
        assertThat().textPresent("Hello world");
    }
}
```

RELATED TOPICS

[Plugin Testing Resources and Discussion](#)

Build Failure - Manifest Validation Errors

If you use a vendor name of "Atlassian" in your `atlassian-plugin.xml` file, you may receive errors like this:

```
[ERROR] BUILD FAILURE
[INFO] -----
[INFO] Manifest must contain versions for all imports. Suggested changes:
javax.servlet.http;version="2.3",
com.opensymphony.user;version="1.1.1",
com.atlassian.plugins.rest.common.security;version="1.0.2",
com.atlassian.jira.t*;version="4.0",
net.jcip.annotations;version="1.0",
javax.xml.bind.annotation;version="2.1"
```

The vendor should be set to something other than "Atlassian", since we have a number of additional validations triggered by that vendor string. For example:

```
<plugin-info>
<description>A sample plugin showing how to add a REST service to JIRA.</description>
<version>1.0</version>
<vendor name="Example Company" url="http://www.example.com" />
<application-version min="4.0"/>
</plugin-info>
```

To skip the manifest validation, you can add the following tag to the configuration element in your pom.xml

```
<skipManifestValidation>true</skipManifestValidation>
```

Note: This is not recommended if you want to make your plugin available to multiple Atlassian applications.

RELATED TOPICS

[Writing your first plugin FAQ](#)

[Writing your first plugin](#)

Cannot Log In to Confluence using Admin Account

If you find that you cannot log in to your test Confluence instance using the admin account, set your plugin to build against Confluence 2.6.1 or later using the `atlassian.product.version` parameter (see [system properties](#)).

RELATED TOPICS

[Atlassian Plugin SDK FAQ](#)

[Installing the Atlassian Plugin SDK](#)

Choosing a Logging Framework

Three logging frameworks are available to plugins:

- Log4j 1.2.15
- Commons Logging 1.1.1
- SLF4J 1.5

We recommend Simple Logging Facade for Java (SLF4J) as a the primary logging framework.

You should ensure that none of these libraries are bundled with your plugin (in `META-INF/lib`) as this will prevent the host application from logging any of your plugin's log messages.

You can check your plugin's dependencies by using `mvn dependency:tree` and then exclude any transitive dependencies in the `pom.xml`:

```
<exclusions>
<exclusion>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-api</artifactId>
</exclusion>
<exclusion>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-log4j12</artifactId>
</exclusion>
</exclusions>
```

Choosing a Package Name

Make sure your package names are unique. You can choose any package name, provided that it does not conflict with any existing package used in the Atlassian application you are developing for, or in any other plugins. Do *not* use `com.atlassian.confluence.plugins.*` — That is confusing to people who try to use the plugin. Use a real package name that corresponds to your organisation or project. For example: `org.mycompany.confluence.plugins.*` The Sun Java documentation has some tips about conventions used to ensure unique package names.

RELATED TOPICS

[Writing your first plugin FAQ](#)

[Writing your first plugin](#)

Eclipse Maven Plugin Build Error

Please note that the new Eclipse plugin is unable to deal with source 'includes' and 'excludes'.

Use the following command `mvn org.apache.maven.plugins:maven-eclipse-plugin:2.6:eclipse` as this will force mvn to use version 2.6 of the maven eclipse plugin.

The error that you will see if 2.7 is used in your mvn build :

```
[ERROR] BUILD ERROR
[INFO] -----
[INFO] Request to merge when 'filtering' is not identical.
[INFO] Original=resource src/main/resources: output=target/classes,
[INFO]     include=[atlassian-plugin.xml], exclude=[**/*.java], test=false,
[INFO]     filtering=true, merging with=resource src/main/resources:
[INFO]     output=target/classes, include=[], exclude=[atlassian-plugin.xml|**/*.java],
[INFO]     test=false, filtering=false
[INFO] -----
[INFO] For more information, run Maven with the -e switch
[INFO] -----
```

RELATED TOPICS

[Writing your first plugin FAQ](#)
[Writing your first plugin](#)

If you change pom.xml, you may need to restart the atlas-cli

If you are using `atlas-cli` and `pi` for a fast development cycle of your plugin and you get compilation errors about "Unable to run mojo" and no Java line number in your plugin source code, it may be that you changed the dependencies in `pom.xml` but didn't restart `atlas-cli`.

Instant Loading of Plugin Resources

During development, one of the main things that can slow you down is reloading the plugin just to see the effect of a change to a static CSS or JavaScript file. To solve this problem during development, you can point the plugin framework at a directory containing your resource files and have the plugin framework load them first before looking in the plugin itself.

To enable this feature, set the system property `plugin.resource.directories` to a comma-delimited list of directories containing plugin resources. For example, to point the framework at your current project's resource directory when running an Atlassian application from Maven, you could set the `MAVEN_OPTS` variable:

(Source: [Instant Loading of Plugin Resources](#))

Overriding the application's webapp when developing your plugin

The [Atlassian Plugin SDK](#) makes it easy to override any component of the application's webapp.

1. Create a `src/test/resources/***-app` directory, where `***` is the name of the application you're developing your plugin against. For example `confluence`.
2. To this directory add the resources you want to add (or override) following the webapp directory structure.

Now when you run your application via `atlas-run`, `atlas-debug` and/or `atlas-integration-test`, it will use the updated WAR built from the application configured in your `pom.xml` and the added resources you have defined.

Example

This can be useful for example to disable the velocity caches by overriding `velocity.properties` when developing a Confluence plugin.

Create a `src/test/resources/confluence-app/WEB-INF/classes/velocity.properties` file specifying the properties as defined in the [Confluence developer documentation](#). That's it.

RELATED TOPICS

- [Using your own log4j configuration for your plugin](#)
- [Developing your Plugin using the Atlassian Plugin SDK](#)

Specifying a particular version of the host application

The `atlas-run` command will download the latest version of the application binaries into your local Maven repository. If you want to develop against a version of the host application other than the very latest, you can specify the version to use as a parameter of your `atlas-run` command.

For example, let's assume you want to use Confluence 3.1:

1. Run `atlas-clean`.
2. Run `atlas-run -v 3.1`
or `atlas-run --version 3.1`.

 Alternatively, you can permanently change the application version number in the dependencies section of your POM. Note that your POM may use a property to hold the version number, like this:

```
<properties>
  <confluence.version>RELEASE</confluence.version>
</properties>
```

You plugin will always build and deploy against this version, until you change this property.

For example, if you are happy to use Confluence 3.1 for a while, you would change the version to the following:

```
<properties>
  <confluence.version>3.1</confluence.version>
</properties>
```

Using the Atlassian Plugin SDK with a Source Code License

This page is relevant to developers who hold a commercial license for an Atlassian application, with access to the source code.

Since you have purchased a commercial license to an Atlassian application, such as Confluence, you have access to the application source code. You can download a source distribution by visiting <http://my.atlassian.com>. Then you can attach the application source code to your plugin project, so that you can step through the application code in your debugger. It is often helpful to do this to better understand the API or to track down problems.

Once you have downloaded and unpacked the source distribution, you will see a number of directories and files, exactly as they are checked out of our SVN repository. This is perfect for reading and exploring, but it does not do you much good in the brave new world of Maven 2. You'll need to create and install source artifacts into your local Maven repository. Once you have done that, the Maven IDE plugins will recognise that they are there and link them up for debugging purposes.

From the top level of the source distribution, run:

```
atlas-mvn source:jar install -Dmaven.test.skip=true -DskipTests=true
```

This will build JAR files containing most of our source code into the `target` directory of each subproject. The important items are now set up, including the core application source code.

You can copy the JAR files into your local Maven repository to make them available in your IDE. For example, to install the source code for Confluence 3.4 core, copy `confluence-project/confluence/target/confluence-3.4-sources.jar` to `$HOME/.m2/repository/com/atlassian/confluence/confluence/2.10`. After doing so, run `atlas-mvn idea:idea` or `atlas-mvn eclipse:eclipse` again, close and reopen the project; the sources should now be available.

A Note about Memory

This is a resource intensive process, and you may need to allocate more memory to Maven in order to complete it. You can do so by setting an environment variable called `MAVEN_OPTS`, like this:

```
export MAVEN_OPTS=-Xmx512m
```

A Note about Versions

Make sure that you download and install the **same** version of the source code that is set in the `atlassian.product.version` property in your `pom.xml`.

RELATED TOPICS

[Developing your Plugin using the Atlassian Plugin SDK](#)

Using your own log4j configuration for your plugin

The [Atlassian Plugin SDK](#) makes it easy to override the application's `log4j.properties` file, so that you can do custom logging for your plugin.

1. Create your `log4j.properties` file. A simple way to do this is to copy the application's `log4j.properties` file and change it to suit your needs.
2. Place the `log4j.properties` file somewhere in your source tree, for example in `src/aps/log4j.properties` ('aps' means the Atlassian Plugin SDK).
3. Configure your `pom.xml` like this:

```
<plugin>
  <groupId>com.atlassian.maven.plugins</groupId>
  <artifactId>maven-***-plugin</artifactId> <!-- *** is the name of the
application (product) you're using -->
  <version>3.0.4</version>
  <extensions>true</extensions>
  <configuration>
    <productVersion>${product.version}</productVersion>
    <log4jProperties>src/aps/log4j.properties</log4jProperties>
  </configuration>
</plugin>
```

Now you can run your application via `atlas-run` or `atlas-debug`.



A plugin specific `log4j.properties` file does not get picked up using this method once you deploy your plugin to another stand alone server. It only works using `atlas-run` or `atlas-debug`

RELATED TOPICS

[Developing your Plugin using the Atlassian Plugin SDK](#)

Advanced Plugin Development

- [Prerequisites](#)
- [Product development hubs](#)
- [Atlassian product technologies](#)
 - [SAL](#)
 - [AUI](#)
 - [Atlassian REST](#)
 - [Atlassian Template Renderer](#)
- [Reference Application \(RefApp\)](#)
- [Atlassian Plugin Development Platform](#)
- [API Changes](#)
- [Early Access Programs](#)

Prerequisites

This page is the starting point for developers who are currently writing a plugin.

This page is for you if:

- You have started developing a plugin
- You want to find out how to develop a certain kind of plugin

If you've never written a plugin before, please check our [Atlassian Developers page](#) to catch up on what you need to know to develop successfully.

Product development hubs

Product-specific plugin modules, tutorials, and other questions are answered here.

| | | |
|---|--|--|
|  JIRA |  Confluence |  Bamboo |
|  FishEye |  Crucible |  Crowd |

Atlassian product technologies

Each Atlassian product ships a common set of libraries in addition to the product itself. These libraries are available for plugin developers' use.

SAL

Unable to render {include} Couldn't find a page to include called: PLUGINFRAMEWORK:_SAL Overview

[Click here for more information.](#)

AUI

The Atlassian User Interface (AUI) is a set of reusable, cross-browser tested UI components (markup, CSS and Javascript). We developed AUI for use in Atlassian applications such as [JIRA](#), [Confluence](#) and [others](#). As a plugin developer, you will find AUI useful to provide standard UI components that fit in to the host application's user interface. Refer to the [version matrix](#) to see which AUI version is included in each application.

[Click here for more information.](#)

Atlassian REST

You can use the REST plugin module to create plugin points easily in Atlassian applications, by exposing services and data entities as REST APIs. The Atlassian REST plugin module is bundled with our applications.

REST APIs provide access to resources via URI paths. To use a REST API, your plugin or script will make an HTTP request and parse the response. You can choose JSON or XML for the response format. Your methods will be the standard HTTP methods like GET, PUT, POST and DELETE. Because the REST API is based on open standards, you can use any web development language to access the API.

[Click here for more information.](#)

Atlassian Template Renderer

The Atlassian Template Renderer provides an abstraction on top of various templating libraries, making it easier to use the templating libraries. For example, instead of you having to create a VelocityEngine object and configure it, we provide a factory to do that for you. See the [Javadoc](#) and the [wiki documentation](#).

Reference Application (RefApp)

The RefApp exists to enable plugin development across more than one Atlassian application. The RefApp implements the plugin system and the Atlassian product technologies.

See [this page](#) for more information.

Atlassian Plugin Development Platform

The plugin development platform is an ongoing effort to specify what each Atlassian application contains that is of interest to plugin developers.

See [this page](#) for more information.

API Changes

As products evolve, their API evolves too. Check our published API diffs [here](#).

Early Access Programs

Get access to milestone and beta builds of our products to see how your plugin works with new releases.

Atlassian API Changes

We provide generated reports of API differences between one version of a product to the next. We are working on having these generated automatically at release time, but here are some we've already done manually.

Confluence

- Changes from 2.7.4 to 2.8.3

- Changes from 2.8.3 to 2.9.3
- Changes from 2.9.3 to 2.10.4
- Changes from 2.10.4 to 3.0.2
- Changes from 3.0.2 to 3.1.2
- Changes from 3.1.2 to 3.2.2
- Changes from 3.2.2 to 3.3.3
- Changes from 3.3.3 to 3.4
- Changes from 3.4.9 to 3.5

JIRA

- Changes from 4.1.1 to 4.2
- Changes from 4.2.4 to 4.3 [JIRA API] [JIRA core]

Bamboo

- Changes from 2.6.2 to 2.7 beta 1

FishEye/Crucible

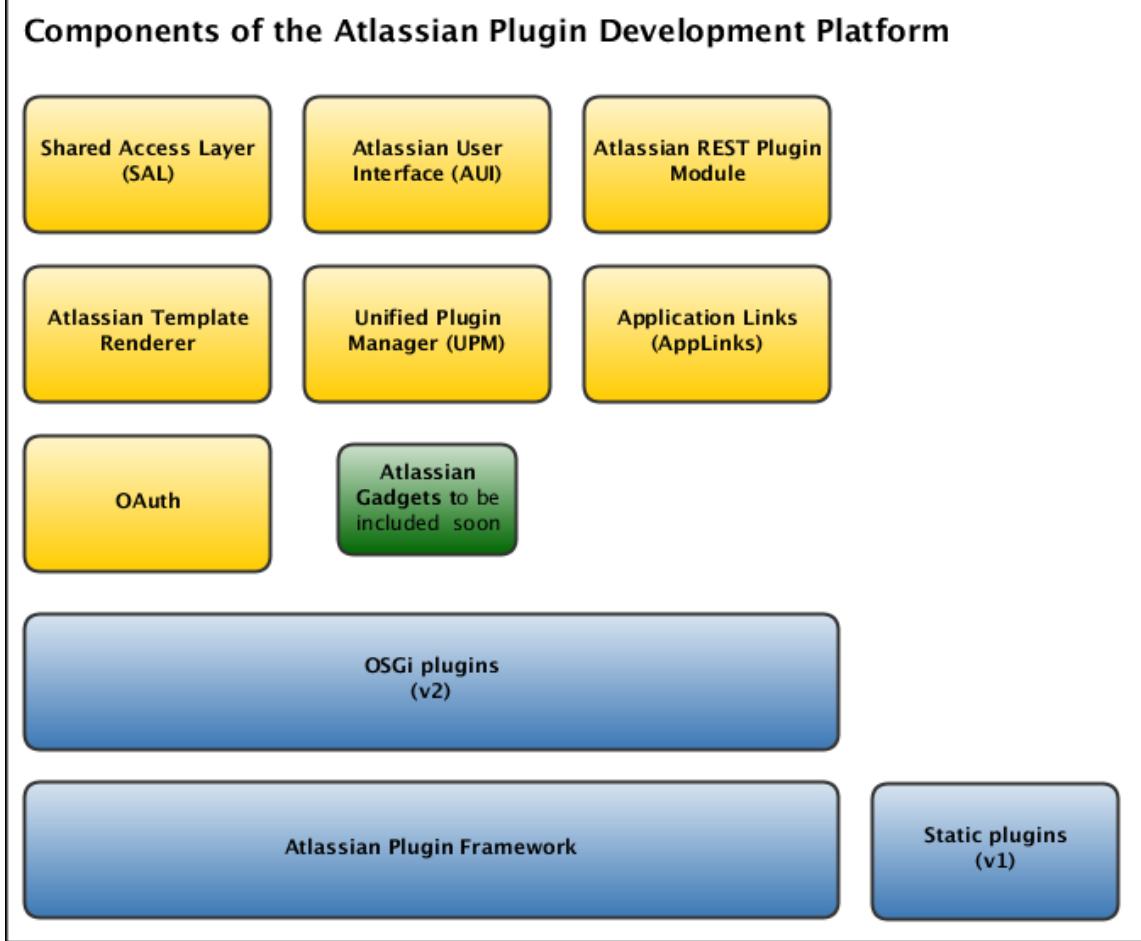
Atlassian Plugin Development Platform

Using Atlassian's plugin development tools, developers can create plugins that extend the functionality of Atlassian applications such as [JIRA](#), [Confluence](#) and [others](#). The Atlassian Plugin Development Platform defines the set of tools a plugin developer can use. This document covers the pieces that make up the platform.

On this page:

- Components of the Plugin Development Platform
- About Plugins
- Atlassian Plugin Framework
- Shared Access Layer (SAL)
- Atlassian User Interface Library (AUI)
- Atlassian REST Plugin Module
- Universal Plugin Manager (UPM)
- Atlassian Template Renderer
- Application Links
- Version Matrix
- Using the Plugin Development Platform Dependency Management POMs

Components of the Plugin Development Platform



These are the major components in the Atlassian Plugin Development Platform:

- **Plugin Framework** – The framework that executes the plugins and manages the available plugin modules.
- **Shared Access Layer (SAL)** – The API for accessing common services, regardless of the underlying Atlassian application interfaces.
- **Atlassian User Interface (AUI)** – A set of reusable, cross-browser tested JavaScript and CSS UI components.
- **Atlassian REST Plugin Module** – An Atlassian plugin module that you can use to create plugin points easily in Atlassian applications, by exposing services and data entities as REST APIs.
- **Atlassian Template Renderer** – A library that provides an abstraction on top of various templating libraries, making it easier to use the templating libraries. For example, instead of you having to create a VelocityEngine object and configure it, we provide a factory to do that for you. See the [Javadoc](#) and the [wiki documentation](#).
- **Universal Plugin Manager (UPM)** – The UI for installing, managing, upgrading, and sharing Atlassian plugins.
- **Application Links (AppLinks)** – An Atlassian plugin module that allows you to link your [JIRA](#), [Confluence](#), [FishEye](#), [Crucible](#) and [Bamboo](#) applications.

About Plugins

A plugin is a bundle of code, resources and configuration files that can be dropped into an Atlassian product to add new functionality or change the behaviour of existing features.

Every plugin is made up of one or more plugin modules. A single plugin may do many things, while a plugin module represents a single function of the plugin.

There are two versions of plugins in the Atlassian Plugin Framework 2:

- **Version 1** — These may be static (deployed in `WEB-INF/lib`) or dynamic (via the web UI, only in Confluence) and should work the same as they did in version 1 of the Atlassian Plugin Framework. The capabilities and features available to version 1 plugins vary significantly across products.
- **Version 2** — These plugins are dynamically deployed on an internal OSGi container to provide a consistent set of features and behaviours, regardless of the application the plugin is running on. Version 2 plugins have to be specifically declared as such, using the `plugins-version="2"` attribute in `atlassian-plugin.xml`.

Atlassian Plugin Framework

As you can see in the diagram, the plugin framework supports several types of plugins, including OSGi-based plugins. OSGi is a dynamic module system for Java that the framework uses to enable plugins to depend on each other and more easily share services. Because the plugin framework supports multiple versions of plugins simultaneously, legacy plugins or those built into the application can exist side-by-side with newer dynamic plugins that leverage the plugin-sharing benefits of OSGi.

[More information....](#)

Shared Access Layer (SAL)

The Shared Access Layer, or SAL for short, provides a consistent, cohesive API to common plugin tasks, regardless of the Atlassian application into which your plugin is deployed. SAL is most useful for cross-application plugin development. If you are developing your plugin for a single application only, you can simply use the application's own API. If your plugin will run in two applications or more, you will find SAL's services useful. These common services include, but are not limited to:

- Job scheduling
- Internationalisation lookups
- Persistence for plugin settings
- A plugin upgrade framework

[More information....](#)

Atlassian User Interface Library (AUI)

The Atlassian User Interface (AUI) is a set of reusable, cross-browser tested UI components (markup, CSS and Javascript). We developed AUI for use in Atlassian applications such as [JIRA](#), [Confluence](#) and [others](#). As a plugin developer, you will find AUI useful to provide standard UI components that fit in to the host application's user interface. Refer to the [version matrix](#) to see which AUI version is included in each application.

[More information....](#)

Atlassian REST Plugin Module

You can use the REST plugin module to create plugin points easily in Atlassian applications, by exposing services and data entities as REST APIs. The Atlassian REST plugin module is bundled with our applications.

REST APIs provide access to resources via URI paths. To use a REST API, your plugin or script will make an HTTP request and parse the response. You can choose JSON or XML for the response format. Your methods will be the standard HTTP methods like GET, PUT, POST and DELETE. Because the REST API is based on open standards, you can use any web development language to access the API.

[More information....](#)

Universal Plugin Manager (UPM)

The Universal Plugin Manager gives the administrator of Atlassian applications a consistent way to install, manage and upgrade plugins. It also makes it easy to share plugins with other users. From the developer's perspective, the Universal Plugin Manager provides a consistent programmatic interface for managing plugins.

[More information....](#)

Atlassian Template Renderer

The Atlassian Template Renderer provides an abstraction on top of various templating libraries, making it easier to use the templating libraries. For example, instead of you having to create a VelocityEngine object and configure it, we provide a factory to do that for you. See the [Javadoc](#) and the [wiki documentation](#).

Application Links

Application Links (AppLinks) is a bundled plugin that allows you to link your [JIRA](#), [Confluence](#), [FishEye](#), [Crucible](#) and [Bamboo](#) applications. Linking two applications allows you to share information and access one application's functions from within the other. For example, if you link JIRA and Confluence, you can view JIRA issues on a Confluence page via the [JIRA Issues macro](#). You can even link your individual projects, spaces and repositories with each other, across the different applications.

Note that the Application Links plugin is bundled and shipped with the Atlassian applications. You cannot install it yourself. Applications Links is bundled with FishEye 2.4, Confluence 3.5, JIRA 4.3, and all later versions of those applications. In addition, Bamboo 3.1 is compatible with AppLinks. You can configure JIRA-to-Bamboo links via the JIRA administration screens.

[More information...](#)

Version Matrix



Current released version: Atlassian Plugin Development Platform 2.10
Atlassian Plugin Development Platform 2.10 is now available – see the [release notes](#).

See [Plugin Development Platform Version Matrix](#).

Using the Plugin Development Platform Dependency Management POMs

If you would like to have just one version to change when upgrading to a new version of the Atlassian Plugin Development Platform, you can use the `atlassian-platform-libraries` and `atlassian-platform-plugins` POM artifacts to set the versions via Maven's "[Import](#)" scope. This will allow you to change the platform version and have all the appropriate versions of platform modules set automatically.

See [Using the Plugin Development Platform Dependency Management POMs](#).

Plugin Development Platform Version Matrix

This page tells you which versions of the Atlassian Plugin Development Platform and its components are included in your host application.

Matrix of Platform Versions and Component Versions

The matrix below shows the versions of each component in the Atlassian Plugin Development Platform, mapped to the versions of the platform itself. The components are listed horizontally across the top and the platform versions are listed vertically on the left.

Version numbers in brackets show a future release.

| Plugin Development Platform | AUI | Plugin Framework | REST Plugin | SAL | UPM |
|-----------------------------|---------|------------------------|------------------------|---------|-----------|
| Platform 2.8 | AUI 3.2 | Plugin Framework 2.6 | REST Plugin Module 2.2 | SAL 2.2 | |
| Platform 2.9 | AUI 3.2 | (Plugin Framework 2.7) | REST Plugin Module 2.2 | SAL 2.2 | (UPM 1.2) |

Matrix of Application Versions and Component Versions

See which versions of the Atlassian applications use each component:

Plugin Framework

Almost every Atlassian application has a version of the Atlassian Plugin Framework. However, they may not all have the same version. Eventually, each Atlassian application ([Confluence](#), [FishEye](#), [Crucible](#), [Crowd](#), etc) will be upgraded to version 2.x, but many are running version 1 today. If you are developing a plugin, you need to know what your version is capable of, and how it will interact with other versions.



The Atlassian Plugin Framework 2 is available in all current Atlassian applications, but may not be available in some older versions.

The matrix below lists the current applications horizontally across the top and the Plugin Framework 2.x versions vertically on the left.

- Version numbers next to a tick show the **earliest** release of the application which supports the relevant framework version.
- Version numbers in brackets show a future application release expected to support the relevant framework version.

| Plugin Framework | Bamboo | Confluence | Crowd | Crucible | FishEye | JIRA |
|------------------|------------|-----------------|-----------|--------------|-------------|----------|
| 2.0 | | | Crowd 1.5 | Crucible 1.6 | FishEye 1.6 | |
| 2.1 | | Confluence 2.10 | Crowd 1.6 | | | |
| 2.2 | Bamboo 2.3 | Confluence 3.0 | Crowd 2.0 | Crucible 2.0 | FishEye 2.0 | |
| 2.3 | Bamboo 2.4 | | | Crucible 2.1 | FishEye 2.1 | JIRA 4.0 |
| 2.4 | Bamboo 2.5 | Confluence 3.1 | Crowd 2.1 | Crucible 2.2 | FishEye 2.2 | JIRA 4.1 |
| 2.5 | | Confluence 3.3 | | | | JIRA 4.2 |
| 2.6 | Bamboo 2.7 | Confluence 3.4 | Crowd 2.2 | Crucible 2.4 | FishEye 2.4 | JIRA 4.3 |
| 2.7 | | Confluence 3.5 | | | | |
| 2.8 | | | Crowd 2.3 | | | JIRA 4.4 |
| 2.9 | | Confluence 4.0 | | | | |

Shared Access Layer (SAL)

The matrix below shows the applications which include and support the Shared Access Layer. The applications are listed horizontally across the top and the SAL versions are listed vertically on the left.

- Version numbers next to a tick show the **earliest** release of the application which supports the relevant SAL version.
- Version numbers in brackets show a future application release expected to support the relevant SAL version.

| | Bamboo | Confluence | Crowd | Crucible | FishEye | JIRA |
|----------------|------------|------------------|-----------|--------------|-------------|----------|
| SAL 2.0 | Bamboo 2.3 | Confluence 3.0 | Crowd 2.0 | 1.6.5 | 1.6.5 | JIRA 4.0 |
| SAL 2.1 | | Confluence 3.3 | | | | JIRA 4.2 |
| SAL 2.2 | Bamboo 3.0 | Confluence 3.4 | Crowd 2.2 | Crucible 2.4 | FishEye 2.4 | JIRA 4.3 |
| SAL 2.3 | | | | | | |
| SAL 2.6 | | (Confluence 4.0) | Crowd 2.3 | | | JIRA 4.4 |

There is also a detailed list of services available per application.

Atlassian User Interface (AUI)

Although almost every Atlassian application uses AUI, they may not all have the same version. If you are using AUI you need to know what version of AUI is supported in your application (or your plugin's host application).

| AUI | Bamboo | Confluence | Crowd | Crucible | FishEye | JIRA | RefApp | PAC |
|----------------|------------|--|-------------|--------------|-------------|-------------|--------|-----|
| 1.0.0 | | | Crowd 2.0 | | | | | |
| 1.0.2 | | Confluence 3.0 Confluence 3.0.1 Confluence 3.0.2 | | | | | | |
| 1.1.1 | | | | | | JIRA 3.13.5 | | |
| 1.1.3 | | | Crowd 2.0.1 | | | | | |
| 1.1.5 | | | Crowd 2.0.2 | | | | | |
| 1.2.1 | | | | | | JIRA 4.0 | | |
| 1.2.4 | | | | | | JIRA 4.01 | | |
| 1.2.5 | Bamboo 2.5 | | | | | | | |
| 2.0.2.1 | | Confluence 3.1 | | | | | | |
| 2.0.4 | | | Crowd 2.0.3 | | | | | |
| 2.0.7 | | Confluence 3.2 | | | | | | |
| 2.1.3 | | | | | | JIRA 4.1 | | |
| 2.2.2 | Bamboo 2.6 | | Crowd 2.0.4 | | | | | |
| 3.0.5 | | Confluence 3.3 | | | | | | |
| 3.0.6 | Bamboo 2.7 | | | | | | | |
| 3.0.9 | | | | | | JIRA 4.2 | | |
| 3.1.0 | | | | Crucible 2.4 | FishEye 2.4 | | | |

| | | | | | | | |
|--------------|------------|----------------------|-------------|--------------|-------------|----------|-------|
| 3.2.0 | | Confluence 3.4 | | | | | |
| 3.2.1 | | | | Crucible 2.5 | FishEye 2.5 | | |
| 3.3.0 | | | Crowd 2.2.2 | | | | |
| 3.3.1 | Bamboo 3.0 | Confluence 3.5 | | | | JIRA 4.3 | 2.9.0 |
| 3.4.0 | | | | | | | |
| 3.4.1 | | Confluence 4.0 (TBC) | | | | | |
| 3.4.2 | | | Crowd 2.3 | | | JIRA 4.4 | |
| 3.5.0 | | | | | | | |
| 3.6.0 | | | | | | | |

Notes:

- Atlassian Gadgets and Embedded Crowd use the version of AUI provided by the host/container application
- 'RefApp' is the Atlassian reference implementation, used for plugin development and testing.
- 'PAC' is plugins.atlassian.com, the Atlassian Plugin Exchange.

Atlassian REST Plugin Module

The matrix below shows the applications which include and support the REST API plugin. The applications are listed horizontally across the top and the REST plugin versions are listed vertically on the left.

- Version numbers next to a tick show the **earliest** release of the application which supports the relevant REST plugin version.
- Version numbers in brackets show a future application release expected to support the relevant version of the REST plugin.

| | Bamboo | Confluence | Crowd | Crucible | FishEye | JIRA |
|------------------------|--------------|------------------|-----------|--------------|-------------|----------|
| REST plugin 1.0 | Bamboo 2.4 | | | Crucible 2.3 | FishEye 2.3 | JIRA 4.0 |
| REST plugin 1.1 | Bamboo 2.6 | Confluence 3.1 | Crowd 2.1 | | | |
| REST plugin 2.0 | | Confluence 3.3 | | | | |
| REST plugin 2.1 | | | | | | JIRA 4.2 |
| REST plugin 2.2 | (Bamboo 3.0) | Confluence 3.4 | Crowd 2.2 | Crucible 2.4 | FishEye 2.4 | JIRA 4.3 |
| REST plugin 2.5 | | (Confluence 4.0) | Crowd 2.3 | | | JIRA 4.4 |

Using the Plugin Development Platform Dependency Management POMs

If you would like to have just one version to change when upgrading to a new version of the [Atlassian Plugin Development Platform](#), you can use the `atlassian-platform-libraries` and `atlassian-platform-plugins` POM artifacts to set the versions via Maven's "[import](#)" scope. This will allow you to change the platform version and have all the appropriate versions of platform modules set automatically.

For your web application, add this to your dependencyManagement section:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.atlassian.refapp</groupId>
      <artifactId>atlassian-platform-libraries</artifactId>
      <version>2.7.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

For your bundled plugins module that builds the bundled plugins zip file, add this:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.atlassian.refapp</groupId>
      <artifactId>atlassian-platform-plugins</artifactId>
      <version>2.7.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

With these elements in place, you can now remove the version elements for all platform libraries and plugins.

For consistency, we recommend that you use a Maven property for the version to allow easy overriding: `platform.version`



If you are depending on a project that contains the above `<dependencyManagement>` stanza, you may receive an error similar to:

```
[INFO] Unable to find resource
'com.atlassian.refapp:atlassian-platform-libraries:pom:2.7.0' in repository central
(http://repol.maven.org/maven2)
[INFO] -----
[ERROR] BUILD ERROR
[INFO] -----
[INFO] Error building POM (may not be this project's POM).

Project ID: com.atlassian.refapp:atlassian-platform-libraries

Reason: POM 'com.atlassian.refapp:atlassian-platform-libraries' not found in
repository: Unable to download the artifact from any repository

  com.atlassian.refapp:atlassian-platform-libraries:pom:2.7.0

from the specified remote repositories:
  central (http://repol.maven.org/maven2)

for project com.atlassian.refapp:atlassian-platform-libraries

[INFO] -----
[INFO] For more information, run Maven with the -e switch
[INFO] -----
```

This is due to a bug ([MNG-4148](#)) in Maven 2.x where profiles specified in `settings.xml` are ignored.

Apart from upgrading to Maven 3.x, there is currently no workaround for this.

Note that this does not affect projects that have the above `<dependencyManagement>` stanza in their pom. For example, if project A has the stanza, it will build successfully. However, if project B depends on project A, it will cause the error mentioned above.

RELATED TOPICS

[Atlassian Plugin Development Platform](#)
[Plugin Framework 2.6 Migration Guide](#)

Early Access Programs

We sometimes offer Early Access (aka Beta) to preview releases of our products, primarily to help plugin developers prepare for changes in between [major releases](#). The releases, when available, can be found here:

EAP Releases

- [JIRA](#)

- Confluence
- Bamboo
- Fisheye
- Crucible
- Crowd
- Clover

EAP Release Notes:

- JIRA Beta Release Notes
- Confluence Beta Release Notes
- Bamboo Beta Release Notes
- Fisheye Beta Release Notes
- Crucible Beta Release Notes
- Crowd Beta Release Notes
- Clover Beta Release Notes

Using the Atlassian RefApp

The Atlassian Reference Application, or reference implementation, is known as the 'RefApp'. It is a model implementation of the Atlassian Plugin Framework 2. It takes the form of a simple web application, which implements the plugin framework and does not do much more than that.

 Another example of a reference implementation: [GlassFish](#) is the reference implementation of Java EE.

This page describes the purpose of the RefApp and tells you how to run your plugin in the RefApp.

On this page:

- Purpose and Scope of the RefApp
 - When Would You Use the Refapp?
 - What Does the RefApp Do?
- Creating your Plugin for the RefApp
- Finding your Plugin in the RefApp

Purpose and Scope of the RefApp

When Would You Use the Refapp?

The RefApp is useful to developers who wish to develop cross-product plugins, meaning plugins which can be added to more than one Atlassian application. It is probably not very interesting to developers wanting to write plugins for a specific Atlassian application.

For example, if you are creating a plugin:

- To be run in Confluence as well as JIRA, or any two Atlassian applications – This is when you may find it helpful to use the RefApp, by running `atlas-create-refapp-plugin`.
- For JIRA only – You will probably not find the RefApp useful. Instead, use `atlas-create-jira-plugin`.
- For Confluence only – You will probably not find the RefApp useful. Instead, use `atlas-create-confluence-plugin`.
- For any other Atlassian application, where the plugin will be used in that application only – You will probably not find the RefApp useful. Instead, use the application-specific '`atlas-create`' command.

What Does the RefApp Do?

The RefApp is an application with no features and very little functionality. Instead, it represents the lowest common denominator of all the Atlassian applications — a way of codifying the shared framework in all the applications.

Atlassian's applications were built independently, by different teams. We made many of the same technology choices, but no application is exactly like any other. By developing the RefApp, we can make sure that we do not depend on any functionality that JIRA provides but Confluence does not (for example). In the same way, the RefApp gives the applications an agreed-upon target to make sure they can support cross-application plugins.

Creating your Plugin for the RefApp

Follow the steps to [create your plugin using the Atlassian Plugin SDK](#). When you reach the step where you [create a plugin skeleton](#), use the command `atlas-create-refapp-plugin`.

Now you can start the RefApp as described in the [SDK guide](#), using `atlas-run` to download and run the RefApp with your new plugin already installed.

This is the RefApp home page as displayed in a web browser:

Welcome!

 **ATLASSIAN® RefApp**

[Login](#)

Welcome to the Atlassian RefApp!

This application has no useful features whatsoever. It is intended to be used as a platform for testing cross product Atlassian plugins.

- [Old RefApp Home Page](#)

Wondering how the above links got there? Just add a web item to your plugin:

```
<web-item key="myLink" section="index.links" weight="40" application="refapp">
    <label key="My Link Name or I18n Key">
        <link linkId="myLinkId"/>/relative/or/absolute/path/to/my/link</link>
    </web-item>
```

Other places you can add links to include `system.admin/general`, `header.links` and `footer.links`. You might want to log in, by default, three users are available, admin, fred and barney. Their passwords are the same as their usernames.

Atlassian Plugins - [Issues](#) | [Documentation](#) | [Builds](#) | [Login](#)

Finding your Plugin in the RefApp

Once you have created your plugin and started the RefApp, you can see details of your plugin and other useful information in the RefApp by going to <http://localhost:5990/refapp/index.jsp>.

Welcome!

 **ATLASSIAN® RefApp**

[A. D. Ministrator](#) [Administration](#) [Logout](#)

Plugins

- com.sarah.refapp.plugins.sarahtest
- com.atlassian.sal.refimpl.transaction
- com.atlassian.sal.refimpl.license
- com.atlassian.refapp.trustedapps
- com.atlassian.sal.sal-api-2.0.11
- com.atlassian.pdkinstall
- com.atlassian.oauth.admin
- com.atlassian.oauth.serviceprovider.sal
- com.atlassian.oauth.atlassian-oauth-api-1.0.0
- com.atlassian.sal.refimpl.net
- com.atlassian.oauth.atlassian-oauth-consumer-spi-1.0.0
- com.atlassian.sal.refimpl.appproperties
- com.atlassian.sal.refimpl.scheduling
- com.atlassian.bundles.google-collections-1.0.0.rc1
- com.atlassian.sal.refimpl.message
- com.atlassian.sal.refimpl.user
- com.atlassian.labs.httpservice.bridge-0.5.3
- com.atlassian.whoami
- com.atlassian.refapp.decorator

RELATED TOPICS

[atlas-create-refapp-plugin](#)
Developing your Plugin using the Atlassian Plugin SDK

Advanced Plugin Development FAQ

- Adding WebSudo Support to your Plugin
- BeanCreationException from Spring Framework
- Best Practices for ensuring a plugin works with a new version of the product
- Bundling extra dependencies in an OBR
- Converting a Plugin to Plugins 2
- Deployment stalled due to spaces in directory path
- Detecting the Presence or Absence of Classes in Different OSGI Bundles
- Functional Testing with Multiple Products
- Getting Custom fields from Confluence V2 Searches
- Plugins that Cannot be Reloaded with pi
- Using Plugins for Testing and Development

Adding WebSudo Support to your Plugin

Support for Secure Administrator Sessions, called 'WebSudo', was added in Confluence 3.3 (see [documentation](#)) and JIRA 4.3 (see [documentation](#)). When an administrator who is logged into Confluence or JIRA attempts to access an administration function, they are prompted to log in again.

All the Atlassian applications will support WebSudo sessions at some point. As of [SAL 2.2](#) and [REST 2.2](#) it is possible to enforce WebSudo from within a plugin if the host application supports it.

[More...](#)

BeanCreationException from Spring Framework

If you get an error like the one below when you attempt to use your plugin, check that your `atlassian-plugin.xml` file (i.e. your plugin descriptor) contains `plugins-version="2"`.

The Error

```
Unable to render content due to system error: org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'com.sarah.confluence.plugins.ExampleMacro': Instantiation of bean failed; nested exception is org.springframework.beans.BeanInstantiationException: Could not instantiate bean class [com.sarah.confluence.plugins.ExampleMacro]: No default constructor found; nested exception is java.lang.NoSuchMethodException: com.sarah.confluence.plugins.ExampleMacro.<init>()
```

The Solution

To fix the problem:

1. Stop the Confluence server. (Go to the server command window and press Ctrl-C.)
2. Add `plugins-version="2"` to the `<atlassian-plugin>` element in your `atlassian-plugin.xml` file. It will look something like this:

```
<atlassian-plugin key="${groupId}.${artifactId}" name="${artifactId}" plugins-version="2">
```

Or this:

```
<atlassian-plugin key="com.atlassian.confluence.plugins.example" name="Example Plugin" plugins-version="2">
```

3. Run `amps-clean`.
4. Run `amps-run` again.
5. Go to the application in your browser and try again.

RELATED TOPICS

[Advanced Plugin Development FAQ](#)
[Advanced Plugin Development](#)

Best Practices for ensuring a plugin works with a new version of the product

Read the Upgrade Guide

The product developers publish upgrade guides each time they release a new version of the product. They often use these upgrade guides to mention things that have changes in the code: new APIs, deprecations, new libraries, etc. Read **all** the upgrades between the version you are currently running on and the one you wish to move to.

- JIRA Upgrade Guides
- Confluence Release Notes (Upgrade guides are attached to each release notes page)
- Bamboo Upgrade Guides
- Crowd Upgrade Guides
- Fisheye Upgrade Guides
- Crucible Upgrade Guides

Upgrade your POM and recompile the plugin against the new product.version

Change your project's POM so that you specify the correct target version. For example, you might change the property `confluence.version` from 2.4.5 to 2.7.1. Once you've done this, do a `mvn clean compile` to see if anything has been broken.

Watch for deprecation warnings

Watch for deprecation warnings during compilation or in your IDE. It's a good idea to correct these warnings as early as possible.

Run the test suite

Be sure you run the full test-suite over your newly-compiled plugin. (You do have a [test suite](#), don't you?)

Test your plugin's functionality manually.

Be sure to test your plugin functionality manually inside a running instance of the product.

Test the product upgrade (including your plugin) with a backup copy of your data

If at all possible, we strongly recommend attempting the upgrade in a test-bed before doing so with your live system. You'll want to run the upgrade and install the new plugin, making sure that all the data is upgraded properly, that the new plugin works as expected, that there are not unexpected errors in the logs, etc.

Backup all your data

This is critical. You **must** perform a full backup of your data (database, attachments, indexes, `confluence.home`, etc) before attempting an upgrade.

Perform the real upgrade

Perform the real upgrade with your real data and install your new plugin. Test everything again before you bring your system back online.

Bundling extra dependencies in an OBR

With the ability to create dynamic dependencies on other plugins and libraries with OSGi and the Plugins 2 framework, the challenge becomes making sure that all the other required bundles are actually installed into the application along with your plugin. The Plugin Exchange lets you define 'dependent' plugins, but currently that information isn't used anywhere currently.

A solution is to use an OSGi Bundle Repository (OBR) file, which is essentially a JAR file containing your plugin, any dependent plugins, and the information required to install them.

Requirements

UPM only

Currently, you have to use the Universal Plugin Manager (UPM) to install OBR files. Installing via the older 'Plugins' console or the 'Plugin Repository' from Confluence 3.3 and earlier will not work. From Confluence 3.5 onwards, the UPM is the default and only option for installing plugins via the Administration Console.

OSGi bundles only

Any library/plugin you want to bundle must be an OSGi bundle in its own right, *with a valid `META-INF/MANIFEST.MF` file defining the OSGi bundle*. This is mainly an issue for any Atlassian-based plugins you might have, which do not generate a `MANIFEST.MF` by default. To do so, you need to declare the `<instructions>` section within the definition for the '`com.atlassian.maven.plugins`' plugin for the application you're targeting. More information about this is available [here](#).

pom.xml Configuration

The Atlassian Plugin SDK automatically builds an OBR for your plugin, but actually getting it to work as expected takes a bit more work. There are a few key things required to get your plugin building and installing correctly via the UPM.

<properties>

This step is not strictly required, but it will make it easier to ensure that you are building and deploying the same versions of your dependencies. Here, we simply create a property value for the version number of the bundled dependency.

```
<properties>
  <my.library.version>1.0</my.library.version>
  ...
</properties>
```

<dependencies>

All libraries you want to have bundled **must** have an entry in the `<dependencies>` section of the `pom.xml`. They should also have a scope of 'provided' or 'test', otherwise they will get included directly into the plugin jar instead. Eg:

```
<dependencies>
  <dependency>
    <groupId>my.company.whatever</groupId>
    <artifactId>my-library</artifactId>
    <version>${my.library.version}</version>
    <scope>provided</scope>
  </dependency>
  ...
</dependencies>
```

AMPS

There are two things to configure for the AMPS plugin: `<pluginDependencies>` and `<instructions>`.

<pluginDependencies>

Within the `<configuration>` section of your plugin's 'maven-<application>-plugin' definition for AMPS, you need to set the `<pluginDependencies>` details for the dependencies you want bundled.

<instructions>

The package for the dependent plugin **must** be listed in the `<Import-Package>` section of the `<instructions>` for the OSGi bundle. More details about that are [here](#). Some plugins do not technically need to be included for your plugin to build, but if they're not listed here, they won't get installed, since the UPM does dependency checking to ensure it doesn't install libraries that it doesn't need to.

It also seems that on some platforms a spurious 'CONF_COMM' property is added to the `MANIFEST.MF` when building this way. The fix is to include an empty `<CONF_COMM/>` element in your `<instructions>`.

Example

In this example we're targeting Confluence, but similar values would be required for any other targeted application.

```

<build>
    <plugins>
        <plugin>
            <groupId>com.atlassian.maven.plugins</groupId>
            <artifactId>maven-confluence-plugin</artifactId>
            <!-- use the latest version of the SDK -->
            <version>3.2.4</version>
            <extensions>true</extensions>
            <configuration>
                <productVersion>${atlassian.product.version}</productVersion>
                <testResourcesVersion>${atlassian.product.data.version}</testResourcesVersion>
                <!-- Specify what to bundle in the OBR -->
                <pluginDependencies>
                    <pluginDependency>
                        <groupId>my.company.library</groupId>
                        <artifactId>my-library</artifactId>
                    </pluginDependency>
                </pluginDependencies>
                <instructions>
                    <!-- Specify what package to include. Ensure that any packages from OBRS
are also listed. -->
                    <Import-Package>
                        my.company.library;version="${my.library.version}",
                        ...
                    </Import-Package>
                    <CONF_COMM/>
                    ...
                </instructions>
            </configuration>
        </plugin>
        ...
    </plugins>
    ...
</build>

```

Build

You should now be able to build your OBR successfully. To check that it is including what you expect, look into the '/target/obr' directory in your project after running a build.

Once the .obr file is built, you should be able to install it via the UPM by manually uploading it from the 'Install' tab.

Troubleshooting

Actually getting your OBR to install successfully can be tricky. One issue is that the UPM doesn't always report useful error messages in the log about what is causing the problem. Here are a few tips:

- 1. Ensure you can install the dependencies independently.** Install each dependency into the application individually first to ensure that there aren't any problems with specific libraries. The UPM reports more useful errors this way also. Often the problem is missing imports, either due to not being listed in the <Import-Package> section of the library, or being missing from the target application, or having a bad version number.
- 2. Check the MANIFEST.MF.** This is for both the set of dependencies, and the plugin you're building. Take a look and make sure it's getting generated as you expect.
- 3. Check the obr.xml.** Inside the OBR (and the '/target/obr' directory) is the obr.xml file, which defines the links between the plugin and its dependencies. Make sure the extra libraries/plugins are mentioned in the <require> sections for your target plugin.
- 4. All dependencies must be in Maven.** You won't be able to build without this, but it's essential any plugins and libraries you use are available in your development Maven repository. However, they do not have to be available to the general public, since one the OBR is built it is self-contained.

FAQ

Why do I have to define all my imports manually now?

The default mechanism used by Atlassian Plugins doesn't generate a MANIFEST.MF file, which is currently required for the UPM to know how to install the OBR file. As such, it needs to be defined manually.

OBR vs. bundling internally and using <Export-Package>

You can achieve almost the same effect as an OBR by bundling your dependent libraries in the same way you could for Plugins 1, and then adding an <Export-Package> entry for the common libraries. This is not a bad way to go, and allows plugins to share code and classes, but it has a couple of disadvantages.

Firstly, it doesn't work for dependent *plugins*, since they have extra work that is done, as defined in the `atlassian-plugin.xml`, as well as potentially Spring configurations, etc. You can only export/import the classes, not have extra lifecycle stuff happen.

Secondly, it's not upgradable independently. The versions you use are based on what's available in whatever other plugins you have installed.

On the other hand, a potential disadvantage of OBRs is that the extra bundles will remain installed after the plugin is removed, thus consuming extra memory and/or resources.

Converting a Plugin to Plugins 2

See the plugin framework documentation: [Converting a Plugin to Plugin Framework 2](#).

Deployment stalled due to spaces in directory path

If you get an error like the one below after running `atlas-run`, please check that the directory path to your plugin does not contain spaces. (Issue tracking is at [AMPS-126](#).)

```
34K downloaded  (cargo-core-container-weblogic-1.0-beta-2.jar)
[INFO] [cargo:start]
[INFO] [stalledLocalDeployer] Deploying [C:\Documents and Settings\name\My Documents\source\test-plugin\plugintest\target\confluence\confluence.war] to [C:\Documents and Settings\name\My Documents\source\test-plugin\plugintest\target\container\tomcat6x\cargo-confluence-home\webapps]...
[INFO] [talledLocalContainer] Tomcat 6.x starting...
[WARNING] [talledLocalContainer] java.lang.NoClassDefFoundError: and
[WARNING] [talledLocalContainer] Exception in thread "main"
[WARNING] [talledLocalContainer] Java Result: 1
[INFO] -----
[ERROR] BUILD ERROR
[INFO] -----
[INFO] Unable to execute mojo

Deployable [http://localhost:1990/cargocpc/index.html] failed to finish deploying within the timeout period [120000]. The Deployable state is thus unknown.
[INFO] -----
[INFO] For more information, run Maven with the -e switch
[INFO] -----
[INFO] Total time: 6 minutes 36 seconds
[INFO] Finished at: Tue Jun 15 09:05:51 EDT 2010
[INFO] Final Memory: 49M/97M
[INFO] -----
```

Detecting the Presence or Absence of Classes in Different OSGI Bundles



Problems have been found with this approach and it is not recommended. We are currently working on a better solution.



This page is about 'Plugins 2' plugins, i.e. plugins written for version 2 of the Atlassian Plugin Framework. The procedure below will not work with plugins written against version 1 of the plugin framework.

Occasionally you will need to write a plugin that behaves differently depending on the presence or absence of another plugin. An example where we at Atlassian confront this problem is with the [AppLinks](#) plugin: if the AppLinks plugin is available, we want to use it to auto-detect known servers and connect to them automatically, but we still want our plugin to behave normally if AppLinks is not present.

There are a few ways to detect whether a specific plugin is present in your host application or not. On this page, we will show you how to add a **listener** to track when the availability of that plugin changes.

To start with, there is a bit of housekeeping. Add a dependency on spring-extender in your `pom.xml`:

```
<dependency>
  <groupId>org.springframework.osgi</groupId>
  <artifactId>spring-osgi-extender</artifactId>
  <version>1.2.0</version>
  <scope>provided</scope> <!-- Provided by application -->
</dependency>
```

Add a DynamicImport-Package element to the plugin-info element in atlassian-plugin.xml:

```
<plugin-info>
...
<bundle-instructions>
...
<DynamicImport-Package>com.example.plugin.package*;version="1.2.3",com.example.another.plugins.packag
<!-- packages are separated by commas (,) and package information is separated by semicolons (;)
-->
</bundle-instructions>
</plugin-info>
```

Wire up the class you will use, in atlassian-plugin.xml:

```
<component key="exampleKey" name="Example Name" class="com.example.somepackage.ExampleClass">
  <interface>com.example.somepackage.ExampleInterface</interface>
</component>
```

Now to the meat of it all. The class should implement BundleContextAware and DisposableBean as follows:

ExampleClass.java

```
package com.example.somepackage;

import org.springframework.osgi.context.BundleContextAware;
import org.springframework.beans.factory.DisposableBean;

public class ExampleClass implements ExampleInterface, BundleContextAware, DisposableBean
{
    private ServiceTracker tracker = null;
    private boolean exampleIsPresent = false;
    public boolean isExamplePresent() { return exampleIsPresent; }

    /**
     * Set the {@link BundleContext} that this bean runs in. Normally this can
     * be used to initialize an object.
     *
     * @param bundleContext the <code>BundleContext</code> object to be used
     * by this object
     */
    public void setBundleContext(final BundleContext bundleContext) // comes from BundleContextAware
    {
        ServiceTracker tracker = new ServiceTracker(bundleContext,
"com.example.plugin.package.ExampleDependency", new ServiceTrackerCustomizer(){
            public Object addingService(ServiceReference serviceReference)
            {
                final ExampleDependency exampleDependency = (ExampleDependency)
bundleContext.getService(serviceReference);
                exampleIsPresent = true;
                return exampleDependency;
            }

            public void modifiedService(ServiceReference serviceReference, Object o)
            {
                /* Do nothing for this simple case */
            }

            public void removedService(ServiceReference serviceReference, Object o)
            {
                exampleIsPresent = false;
            }
        });

        // Start the tracker.
        tracker.open();
        this.tracker = tracker;
    }

    /**
     * Invoked by a BeanFactory on destruction of a singleton.
     *
     * @throws Exception in case of shutdown errors.
     *          Exceptions will get logged but not rethrown to allow
     *          other beans to release their resources too.
     */
    public void destroy() throws Exception // comes from DisposableBean
    {
        // stick your cleanup code here
        tracker.close();
    }
}
```

ExampleInterface.java

```
package com.example.somepackage;

public interface ExampleInterface{}
```

Just query the method **isExamplePresent** to find out if the example dependency is available, and execute different code based on the result.

RELATED TOPICS

[Advanced Plugin Development FAQ](#)
[Installing the Atlassian Plugin SDK](#)

Functional Testing with Multiple Products



This feature requires Atlassian Plugin SDK 3.2 or later. Get the latest version [here](#).

If your plugin interacts with other external Atlassian products, you may want to run your functional tests with multiple applications up and running. Even more, you may want to run your tests against different products, different versions of products, or even different configurations of the same product. This tip will show you how to set up multiple product configurations then use them in creating a functional testing group.

The configuration bits in the following steps take place in the `<configuration>` section of your configured Atlassian Maven plugin. For example, if you were using the `maven-jira-plugin`, look for something like:

```
<build>
  <plugins>
    <plugin>
      <groupId>com.atlassian.maven.plugins</groupId>
      <artifactId>maven-jira-plugin</artifactId>
      <version>3.2</version>
      <configuration>
        <!-- your product and test group configuration goes here -->
      </configuration>
    </plugin>
  </plugins>
</build>
```

Step 1 - Define your products in pom.xml

For each instance of a product, you can define its configuration. If you set the `<productsInheritConfiguration> flag`, then configuration items not in a product will be inherited by all products.

For example, this configuration defines a Confluence and JIRA instance, both sharing an external plugin:

```
<configuration>
  <pluginArtifacts>
    <pluginArtifact>
      <groupId>com.mycompany.myplugin</groupId>
      <artifactId>myplugin</artifactId>
      <version>1.0</version>
    </pluginArtifact>
  </pluginArtifacts>
  <products>
    <product>
      <!-- id, version and dataVersion are all REQUIRED
parameters -->
      <id>confluence</id>
      <!-- must be one of: jira, confluence, bamboo,
fecru, crowd, refapp -->
      <version>3.3.1</version>
      <!-- version of the product to use (same as
productVersion) -->
      <dataVersion>3.0</dataVersion>
      <!-- version of test data to use (same as
productTestData) -->
    </product>
    <product>
      <id>jira</id>
      <version>4.2.1</version>
      <dataVersion>4.0</dataVersion>
    </product>
  </products>
</configuration>
```

To specify multiple instances of the same product, you can give the products instance identifiers. This example creates two Confluence instances:

```

<configuration>
  ...
  <products>
    <product>
      <id>confluence</id>
      <instanceId>confluence-3.3.1</instanceId>
      <version>3.3.1</version>
      <dataVersion>3.0</dataVersion>
    </product>
    <product>
      <id>confluence</id>
      <instanceId>confluence-3.4.0</instanceId>
      <version>3.4.0</version>
      <dataVersion>3.0</dataVersion>
    </product>
  </products>
</configuration>

```

Step 2 - Define a test group using the products

Test groups are a way to have further control over what tests are included in your integration tests and what products should be available when you run them.

This example creates a test group called "confjira" that will start up Confluence, then JIRA, then run all the tests found ending with "Test" in a package starting with "it" and having a subpackage called "confjira".

```

<configuration>
  <testGroups>
    <testGroup>
      <id>confjira</id>
      <productIds>
        <productId>confluence</productId>
        <productId>jira</productId>
      </productIds>
      <includes>
        <include>it/**/confjira/*Test.java</include>
      </includes>
    </testGroup>
  </testGroups>
</configuration>

```

If you specified multiple instances of the same application like in the second example of step 1, you could refer to them like this:

```

<configuration>
  <testGroups>
    <testGroup>
      <id>conf2</id>
      <productIds>
        <productId>confluence-3.3.1</productId>
        <productId>confluence-3.4.0</productId>
      </productIds>
      <includes>
        <include>it/**/conf2/*Test.java</include>
      </includes>
    </testGroup>
  </testGroups>
</configuration>

```

If your integration tests use Selenium, Selenium will automatically be configured to point at the first instance.

Accessing test instances

During each test group execution, the following system properties are set for each product definition:

| | |
|--------------------------|---|
| baseurl.INSTANCE_ID | The base url of the application, e.g. http://localhost:8990/jira |
| http.INSTANCE_ID.port | The HTTP port that the application is listening on, e.g. 8990 |
| context.INSTANCE_ID.path | The context path of the application, e.g. /jira |

Use these values to co-ordinate communication between your integration tests and running application instances.

RELATED TOPICS

[Plugin Testing Resources and Discussion](#)

Getting Custom fields from Confluence V2 Searches

If you use an [extractor module](#) in your Confluence plugin to add additional information to the Lucene search index there is currently no way to access these fields using the [V2 Search API](#). You can use these fields in custom search queries or sorts, but actually reading these fields from the returned data is not possible with the current API.

Limitations of the Current API

When you do a search using the V2 Search API the `SearchManager.search(...)` method Confluence will give you back a `SearchResults` object. You can then retrieve each of the individual `SearchResult` objects that matched your search.

The `SearchResult` interface contains a `getExtraFields()` method which is supposed to return additional fields that cannot be directly accessed by this interface. Unfortunately the `LuceneSearchResult` implementation has hardcoded the fields that are returned. This means that if you have an extractor module that has added custom fields to the index they can not be retrieved using the `SearchResult` interface.

Wrapping the `SearchResult` Object with custom implementation

If you look at the source code for the `LuceneSearchResult` class, you will see that all of the index document fields are stored in a private map called `results`. You can use java reflection to get access to this field within your own plugin classes.

The following example wraps the `LuceneSearchResult` providing access to the internal results field.

```
public class LuceneV2Result {

    static private Field resultsField;

    static {
        try {
            resultsField = LuceneSearchResult.class.getDeclaredField("results");
            resultsField.setAccessible(true);
        } catch (Exception e) {
            // :(
        }
    }

    Map<String, String> fields;

    @SuppressWarnings("unchecked")
    public LuceneV2Result(SearchResult result) {
        try {
            fields = (Map) resultsField.get(result);
        } catch (Exception e) {
            // :(
        }
    }

    public Map<String, String> getFields(){
        return fields;
    }
}
```

 It should be noted that this implementation is highly likely to break in future releases if Atlassian change the internal implementation of this class. Hopefully by that time there will be a way to access custom extractor fields directly.

Plugins that Cannot be Reloaded with `pi`

During development it is useful to use the `maven-cli pi` command to reload a plugin in an actively running application, to provide rapid testing of changes.

However, plugin modules can be marked with a `@RequiresRestart` annotation which means the plugin will not be reloaded until the application server is restarted. This means that the `pi` command does not work for these plugins.

Below is a list of plugin modules indicating if they require a restart or not.

JIRA

| Module Type | Can be reloaded |
|-------------------------|-----------------|
| Component | No |
| ComponentImport | Yes |
| ComponentTabPanel | No |
| CustomFieldSearcher | No |
| CustomFieldType | No |
| Gadget | Yes |
| IssueOperation | No |
| IssueTabPanel | No |
| JQLFunction | No |
| ModuleType | Yes |
| Portlet | No |
| ProjectTabPanel | No |
| Report | No |
| Resource | Yes |
| REST | Yes |
| RPC-SOAP | No |
| RPC-XMLRPC | No |
| SearchRequestView | No |
| ServletContextListener | Yes |
| ServletContextParameter | Yes |
| ServletFilter | Yes |
| Servlet | Yes |
| UserFormat | No |
| VersionTabPanel | No |
| WebItem | Yes |
| WebResource | Yes |
| WebSection | Yes |
| Webwork | No |
| WorkflowCondition | No |
| WorkflowFunction | No |
| WorkflowValidator | No |

Confluence

All modules except Component will be reloaded

Bamboo

Information to be completed. Component will not be reloaded.

Fisheye/Crucible

Information to be completed. Component will not be reloaded.

Other information

- See <http://jira.atlassian.com/browse/JRA-19242>

Using Plugins for Testing and Development



This feature requires Atlassian Plugin SDK 3.3 or later. Get the latest version [here](#).

Some features are hard to test without providing testing specific functionality in the product. Atlassian Plugin SDK 3.3 introduced a feature that allows creating plugins that will only be installed when the product is started with Atlassian Plugin SDK.

This technique is useful for:

1. Integration testing where you need to execute some code on the server
2. Development shortcut urls you don't want to ship

The only difference between a testing plugin and a normal plugin is that test plugin files are placed under src/test directory hierarchy. So for example atlassian-plugin.xml would be placed in src/test/resources directory instead of src/main/resources directory.

Step 1 - Add a plugin descriptor in src/test/resources directory

Create a new atlassian-plugin.xml in src/test/resources directory. If the project already contains atlassian-plugin.xml file in src/main/resources directory, ensure that these two files will have different plugin keys.

```
<atlassian-plugin key="com.atlassian.example-test" name="Atlassian Example Test Plugin"
  pluginsVersion="2">
  <plugin-info>
    <version>1.0</version>
    <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com/" />
  </plugin-info>
</atlassian-plugin>
```

Step 2 - Implement testing plugin modules

Testing plugins can use existing plugin modules, but often a new plugin module is required. We will create a hello world servlet which will be placed in src/test/java/com/atlassian/example directory:

```
package com.atlassian.example;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloWorldTestServlet extends HttpServlet
{
    @Override
    protected void doGet(final HttpServletRequest request, final HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/plain");
        response.getWriter().println("Hello World!");
    }
}
```

Step 3 - Add plugin modules to module descriptor

Next step is to specify plugin modules in the testing plugin module descriptor (src/test/resources/atlassian-plugin.xml). In this example we define HelloWorldTestServlet as a plugin module:

```
<servlet name="Hello World Test Servlet" key="helloWorldTestServlet"
  class="com.atlassian.example.HelloWorldTestServlet">
  <url-pattern>/hello-world</url-pattern>
</servlet>
```

Step 4 - Configure Atlassian Plugin SDK

If the project was created using Atlassian Plugin SDK there is no need to anything. Otherwise, add the following lines to pom.xml to make it a minimal Atlassian Plugin project:

```
<packaging>atlassian-plugin</packaging>

<build>
  <plugins>
    <plugin>
      <groupId>com.atlassian.maven.plugins</groupId>
      <artifactId>maven-amps-plugin</artifactId>
      <extensions>true</extensions>
    </plugin>
  </plugins>
</build>
```

See [AMPS Plugin for Maven](#) for configuration details.

Step 5 - Using the testing plugin

Testing plugin will be enabled in the product both when running the product and when running integration tests:

```
atlas-run
atlas-integration-test
```

HTTP requests to <http://localhost:5990/refapp/plugins/servlet/hello-world> would now get responded with "Hello World!".

RELATED TOPICS

[Plugin Testing Resources and Discussion](#)

Atlassian Plugin SDK Documentation



Current released version – Atlassian Plugin SDK 3.4

Currently-supported applications: **Confluence**, **JIRA**, **Bamboo**, **FishEye**, **Crucible** and **Crowd**.

Atlassian Plugin SDK 3.4 is now available – see the [release notes](#). If you have ideas for improvement or new features, or have found a bug, please raise a ticket on our [issue tracker](#). Snapshot builds are also available for the stout-hearted.



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

[Download Latest Version \(Windows\)](#)



[Download Latest Version \(Mac OS X, UNIX, Linux\)](#)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

The Atlassian Plugin SDK includes [Maven](#), a correctly-configured Maven `settings.xml` file, and a number of shell scripts to speed up and simplify plugin development. The scripts hide the complexities of Maven and provide the developer with less verbose commands,

command-line help and auto-completion.

When running Maven commands against your project, make sure that you use the version of Maven bundled with the Atlassian Plugin SDK. This is important if you have a local version of Maven installed, as well as the Atlassian Plugin SDK. The simplest way is to use the `atlas-mvn` wrapper command instead of `mvn`. Another way is to put the bundled Maven on your path.

On this page:

- Supported Atlassian Applications and Default Ports
- Getting Started
- Shell Script Reference Guide
- Getting Help
- Functional Overview and Examples
- Benefits of Using the Atlassian Plugin SDK

Supported Atlassian Applications and Default Ports

The table below shows the applications currently supported by the Atlassian Plugin SDK, the minimum version of the application required, and the default port for each application.

| Atlassian Application | Version | Default Port | Product Key | Caveats |
|-----------------------|---------|--------------|-------------|---|
| Bamboo | 2.3 + | 6990 | bamboo | |
| Confluence | 2.10 + | 1990 | confluence | |
| Crowd | 2.0.2 + | 4990 | crowd | |
| Crucible | 2.0.6 + | 3990 | fecru | |
| FishEye | 2.0.6 + | 3990 | fecru | |
| JIRA | 4.0 + | 2990 | jira | Plugins developed for versions of JIRA before 4.0 are supported, but using the SDK with versions of JIRA earlier than 4.0 is not. For developing plugins for JIRA 3.13 and earlier, take a look at the old JIRA PDK . |
| RefApp | 1.0 + | 5990 | refapp | |

The SDK supports both [static](#) and [dynamic](#) plugins. The focus is on [version 2 plugins](#), as supported by the [Atlassian Plugin Framework](#).

Getting Started

Follow the [setup guide](#) to download and install the SDK.

Follow the [step-by-step guide](#) to create your first plugin. Alternatively, if you have an existing plugin, follow the [conversion guide](#).

Shell Script Reference Guide

Click the links below to see the detailed descriptions of each shell script provided by the Atlassian Plugin SDK, along with the available parameters and some usage examples:

- [Plugin SDK Commands](#)
- [Atlassian Plugin SDK Release Notes](#)
- [AMPS Plugin for Maven](#)
- [Getting Help with the Atlassian Plugin SDK](#)
- [Converting your Plugin to the new Atlassian Plugin SDK](#)
- [Plugin SDK Snapshots](#)
- [SDK Feedback Form](#)
- [Validation with XML schemas](#)

Getting Help



Help is at hand

Make sure you are not in the Maven command line interface (CLI) when you enter the help commands described below. If you are in the CLI, your command line will start with `maven2>`, and you will need to exit from the CLI first. Enter `quit`, `exit` or a friendly `bye`.

Getting an Overview of All the Scripts

Enter the following shell script to see an overview of all the scripts with a brief outline of their functionality:

Enter the following to see all possible help content:

Getting Help Text per Script

Each shell script provides help text if the first argument of the script is one of the following:

- -?
- -h
- help
- -help
- --help

Examples:

Reading the Reference Guide

See the detailed [guide to all scripts](#).

Functional Overview and Examples

The Atlassian Plugin SDK provides a number of shell scripts wrapping the commands you will need to create, install and build your plugin.

The scripts are of the form:

```
atlas-xxxx [--parameter-name parameter-value] [--parameter-name parameter-value]
```

- Some scripts are purely Maven wrappers, and pass all parameters straight through to Maven.
- Some scripts support interpreted parameters.
- Some scripts have no parameters.

The details of all scripts are in the [reference section](#).

In this section, we will look at some typical tasks and give examples of the SDK scripts you can use.

Creating a New Plugin from Scratch

Say you want to create a new Confluence plugin skeleton. Simply open a command window, go to the directory where you want to create the plugin and type:

```
atlas-create-confluence-plugin
```

Similarly, you would enter one of the following to create a plugin for JIRA, Bamboo or the RefApp:

```
atlas-create-jira-plugin  
atlas-create-bamboo-plugin  
atlas-create-refapp-plugin
```

Running a Plugin in an Application

Say you want to install and run your plugin in your host Atlassian application. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-run
```

Note that the above shell script will work for any host application, including Confluence, JIRA, etc. The script will determine the application, based on your plugin's specifications.

Specifying a Version of the Application

Say you want to run your plugin with Confluence 2.10.3. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-clean  
atlas-run --version 2.10.3
```

Say you want to run your plugin with JIRA 4.0 snapshot. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-clean  
atlas-run --version 4.0-SNAPSHOT
```

Note: Running `atlas-clean` will clear the previous version of the host application from your build output directory. You only need to do this if the previous application version was different from the one you need now.

Specifying an Application Server (Container)

Say you want to run your plugin with Confluence 2.10.3 and JBoss 4.2.x. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-run --version 2.10.3 --container jboss42x
```

Specifying a Version of SAL

Say you want to run that plugin but with Confluence 2.10.3 and SAL 2.0.5:

```
atlas-run --version 2.10.3 --sal-version 2.0.5
```

Running Integration Tests against a Different Application

Say you have a RefApp plugin but want to run your integration tests against Confluence:

```
atlas-integration-test --product confluence
```

Running your Plugin in Multiple Applications

Say you want to run that RefApp plugin in multiple applications simultaneously. In three separate tabs in your terminal (command window):

- Type the following to run the plugin in the RefApp:

```
atlas-run
```

- Type the following to run the plugin in Confluence:

```
atlas-run --product confluence --version 3.0-m9
```

- Type the following to run the plugin in JIRA:

```
atlas-run --product jira --version 4.0-SNAPSHOT
```

Using the Command Line Interface

The SDK bundles the Maven CLI plugin and pre-configures it properly with the `pi` and `pu` commands. To use it with your plugin's host application, go to the plugin's project directory (where you created the plugin) and type:

```
atlas-cli
```

Generating Test Data for Re-Use

Use the `atlas-create-home-zip` command to create a zip file of your application's home directory, and then load the contents of the home directory into the application every time you start it up.

For example, let's assume you want to prepopulate JIRA with a few projects and issues for testing on every clean startup. First, run JIRA using `atlas-run` and add your projects and issues. Stop JIRA, go to the project directory (where you created the plugin) and type:

```
atlas-create-home-zip
```

Copy the `generated-test-resources.zip` file to the `test /src/test/resources/` directory. Then add the following element to the configuration of the `maven-jira-plugin` in your POM:

```
<productDataPath>${basedir}/src/test/resources/generated-test-resources.zip</productDataPath>
```

See [atlas-create-home-zip](#).

Benefits of Using the Atlassian Plugin SDK

The Atlassian Plugin SDK makes your life easier by providing just one tool to help you do the following, no matter which Atlassian application your plugin is for:

- Create a plugin skeleton from a Maven archetype, specific to the Atlassian application you are developing for.
- Download the application binaries, install your plugin and start the application.
- Dynamically re-install your plugin after changes during development. No restart required, not even for JIRA.
- Write quality unit tests and integration tests.
- Speed up the all-important code-deploy-test cycle.
- Use the same commands for all Atlassian applications.

The SDK provides the above functionality by:

- Shielding you from Maven complexity. With the Atlassian Plugin SDK, the days of fiddling with Atlassian parent POMs are over. You no longer have to hack the Atlassian Maven infrastructure into your existing one. You can compose Atlassian plugin development functionality into your existing Maven hierarchy. This makes it much easier to have a plugin as part of a multimodule project, for example.
- Incorporating the [Maven CLI plugin](#) and [PDK](#) for rapid development.
- Working with the [Maven ITBlast plugin](#) for multi-container integration tests.
- Supplying the boiler-plate POM for bundling dependencies in META-INF/lib. (See [Accessing Classes from Another Plugin](#).)
- Using the OSGi Bundle Repository ([OBR](#)) to resolve plugin dependencies. OBR provides a way to install a number of plugins and their dependencies via a single zip file.
- Automatically configuring the [IDEA CLI plugin](#) for rapid deployment.
- Supporting debug mode, both remote and within your IDE.
- Allowing easy upgrades to the latest version of the Atlassian Plugin SDK.

The SDK offers a set of well-defined goals that make it easy to do common development tasks. If you want to run the integration tests – and only the integration tests – you can say:

```
atlas-integration-test
```

Similarly, you can run the unit tests in isolation:

```
atlas-unit-test
```

You can also install your plugin from the command line, instead of having to browse to the correct page and make several clicks:

```
atlas-install-plugin
```

Or:

```
atlas-cli then pi
```

And so on. Refer to the details of all the [available shell scripts](#).

RELATED TOPICS

Developing your Plugin using the Atlassian Plugin SDK

- [How to learn to program \(in Java or JavaScript\)](#)
- [Installing the Atlassian Plugin SDK](#)
- [Writing your first plugin](#)
- [Advanced Plugin Development](#)
- [Atlassian Plugin SDK Documentation](#)
- [Other Information](#)
- [FAQ](#)
- [Tutorials](#)
- [Getting Involved in the Atlassian Developer Network](#)
- [Codegeist V](#)
- [Developer Relations State of the Union 2011 - Resources](#)

Plugin SDK Commands

- **atlas-clean** — atlas-clean [options] – Removes files from the project directory, that were generated during the build. (Runs mvn clean.) Passes all parameters straight through to Maven.
- **atlas-cli** — atlas-cli [options] – Starts up a command line interface to your plugin running in the host application. Then use pi (plugin install). (Runs mvnamps:cli.) Interpreted parameters: http-port, context-path, server, cli-port.
- **atlas-clover** — atlas-clover [options] – Runs unit and integration tests, generating a code coverage report with Clover. Passes all parameters straight through to Maven. Report is available in target/site/clover/index.html.
- **atlas-compile** — atlas-compile [options] – Compiles the sources of your project. (Runs mvn compile.) Passes all parameters straight through to Maven.
- **atlas-create-bamboo-plugin** — atlas-create-bamboo-plugin [options] – Creates an example of a Bamboo plugin, which you can adapt to suit your own plugin's needs. (Runs mvn bamboo:create.) Interpreted parameters: artifact-id,

- group-id, version, package, non-interactive.
- **atlas-create-confluence-plugin** — atlas-create-confluence-plugin [options] – Creates an example of a Confluence plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn confluence:create`.) Interpreted parameters: artifact-id, group-id, version, package, non-interactive.
- **atlas-create-crowd-plugin** — atlas-create-crowd-plugin [options] – Creates an example of a Crowd plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn crowd:create`.) Interpreted parameters: artifact-id, group-id, version, package, non-interactive.
- **atlas-create-fecru-plugin** — atlas-create-fecru-plugin [options] – Creates an example of a FishEye or Crucible plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn fecru:create`.) Interpreted parameters: artifact-id, group-id, version, package, non-interactive.
- **atlas-create-home-zip** — atlas-create-home-zip – Creates a test-resources zip of the current application's home directory. This zip file can then be pointed to in the AMPS productDataPath property to auto-populate application data during startup.
- **atlas-create-jira-plugin** — atlas-create-jira-plugin [options] – Creates an example of a JIRA plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn jira:create`.) Interpreted parameters: artifact-id, group-id, version, package, non-interactive.
- **atlas-create-refapp-plugin** — atlas-create-refapp-plugin [options] – Creates an example of a RefApp plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn refapp:create`.) Interpreted parameters: artifact-id, group-id, version, package, non-interactive.
- **atlas-debug** — atlas-debug [options] – Runs the application in debug mode with your plugin installed. (Runs `mvn amps:debug`.) Interpreted parameters: version, container, http-port, context-path, server, jvmargs, log4j, test-version, sal-version, rest-version, plugins, lib-plugins, bundled-plugins, product, jvm-debug-port, jvm-debug-suspend.
- **atlas-help** — atlas-help [--verbose] – Displays help text for the shell scripts incorporated into the Atlassian Plugin SDK. Interpreted parameters: verbose.
- **atlas-install-idea-plugin (not published)** — atlas-install-idea-plugin – Configures IntelliJ IDEA for CLI integration. (Runs `mvn amps:idea`.)
- **atlas-install-plugin** — atlas-install-plugin [options] – Installs the plugin into a running application. (Runs `mvn amps:install`.) Interpreted parameters: http-port, context-path, server, username, password, plugin-key.
- **atlas-integration-test** — atlas-integration-test [options] – Runs the integration tests for the plugin. (Runs `mvn integration-test`.) Interpreted parameters: version, container, http-port, context-path, server, jvmargs, log4j, test-version, sal-version, rest-version, plugins, lib-plugins, bundled-plugins, product, no-webapp, skip-tests.
- **atlas-mvn** — atlas-mvn [options] – Allows you to execute any Maven command using the version of Maven bundled with your Atlassian Plugin SDK. (Runs `mvn`.) Passes all parameters straight through to Maven.
- **atlas-package** — atlas-package[options] – Packages the plugin artifacts and produces the JAR. (Runs `mvn package`.) Passes all parameters straight through to Maven.
- **atlas-release** — atlas-release [options] – Performs a release of the current plugin. (Runs `mvn release:prepare release:perform`.) Passes all parameters straight through to Maven.
- **atlas-release-rollback** — atlas-release-rollback [options] – Rolls back a release of the current plugin. (Runs `mvn release:rollback`.) Passes all parameters straight through to Maven.
- **atlas-run** — atlas-run [options] – Runs the application with your plugin installed. (Runs `mvn amps:run`.) Interpreted parameters: version, container, http-port, context-path, server, jvmargs, log4j, test-version, sal-version, rest-version, plugins, lib-plugins, bundled-plugins, product.
- **atlas-run-standalone** — atlas-run-standalone [options] – Runs an Atlassian application standalone, without a plugin project (i.e. not requiring `atlas-create-<product>-plugin`). Interpreted parameters: version, container, http-port, context-path, server, jvmargs, log4j, test-version, sal-version, rest-version, plugins, lib-plugins, bundled-plugins, product.
- **atlas-unit-test** — atlas-unit-test [options] – Runs the unit tests for your plugin. (Runs `mvn test`.) Passes all parameters straight through to Maven.
- **atlas-version** — atlas-version – Displays version and runtime information for the Atlassian Plugin SDK. (Runs `mvn --version`.)

atlas-clean

This page describes the shell script `atlas-clean`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-clean [options]` – Removes files from the project directory, that were generated during the build. (Runs `mvn clean`.) Passes all parameters straight through to Maven.

Parameters

This shell script is a Maven wrapper script. All parameters are passed straight through to Maven.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-clean -?
atlas-clean -help
```

Examples

Let's assume that you want to use a specific version of your plugin's host application. You can specify the version of the host application to use as a parameter of your `atlas-run` command. If you have already used a different version of the same application, you will need to clear the previous version of the host application from your build output directory.

For example, let's assume that you want to use version 3.1 of your host application (e.g. Confluence):

1. Run `atlas-clean`.
2. Run `atlas-run -v 3.1`

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-cli

This page describes the shell script `atlas-cli`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-cli [options]` – Starts up a command line interface to your plugin running in the host application. Then use `pi` (plugin install). (Runs `mvn amps:cli`.) Interpreted parameters: `http-port`, `context-path`, `server`, `cli-port`.

Available CLI commands:

- `pi` — Plugin install. Installs your updated plugin into the application.
- `quit`, `exit` or `bye` — Exits from the CLI.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted Parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|-----------------------------|-----------------|------------------|--|
| <code>--http-port</code> | <code>-p</code> | Yes | HTTP port for the servlet container, e.g. 1990. The defaults are as described in the SDK overview . You may need to change this if you already have a process listed for the default port, such as when you want to bring up two instances of Confluence. |
| <code>--context-path</code> | | Yes | The application context path. You will need to include the leading forward slash. For example, if your application is running at <code>http://localhost:1990/confluence</code> then you should enter <code>/confluence</code> . If your application is running at <code>http://localhost:1990/</code> then you should enter <code>/</code> . |
| <code>--server</code> | | Yes | Host name of the application server. The default is <code>localhost</code> . |

| | | | |
|------------|--|-----|--|
| --cli-port | | Yes | The port the CLI will listen to for commands. The default is 4330. You may need to change this if you already have a process listed for this port. |
|------------|--|-----|--|

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-cli -?
atlas-cli -help
```

Examples

Once you have done the initial `atlas-create-APPLICATION-plugin` and `atlas-run`, you can keep the application running in one command window and use the CLI (command line interface) to dynamically re-install your plugin after each change.

1. Make your changes in your IDE.
2. Open a command window and go to the plugin's root folder (where the `pom.xml` is located).
3. Run `atlas-cli` to start the CLI.
4. Wait until you see a message, Waiting for commands.
5. Run `pi` (plugin install) to compile, package and install the plugin.
6. Go back to your browser. The updated plugin will have been installed into the application, and you can test your changes. (You may need to refresh the browser page first.)

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-clover

This page describes the shell script `atlas-clover`, part of the [Atlassian Plugin SDK](#).

On this page:

- Basic Usage
- Parameters
- Getting Help
- Examples

Basic Usage

`atlas-clover [options]` – Runs unit and integration tests, generating a code coverage report with [Clover](#). Passes all parameters straight through to Maven. Report is available in `target/site/clover/index.html`.

Parameters

This shell script is a Maven wrapper script. All parameters are passed straight through to Maven.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-clover -?  
atlas-clover -help
```

Examples

```
atlas-clover
```

will run all your tests with Clover enabled and generate a source level, HTML code coverage report in `target/site/clover/index.html`

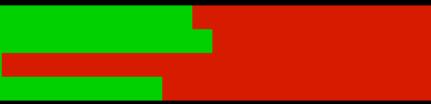
A coverage summary will also be printed to stdout at the end of the build and will be pretty if you have `MAVEN_COLOR=true`:

```
Coverage Timestamp: Mon Sep 14 16:23:26 EST 2009
```

```
Coverage Overview -
```

```
Coverage:-
```

```
    Methods: 92/127 (72.4%)  
    Statements: 385/518 (74.3%)  
    Branches: 101/210 (48.1%)  
    Total: 67.6%
```



```
Complexity:-
```

```
    Avg Method: 1.9527559  
    Density: 0.47876447  
    Total: 248
```

```
[INFO] -----  
[INFO] BUILD SUCCESSFUL
```

```
[INFO] -----
```

```
[INFO] Total time: 2 minutes 12 seconds
```

```
[INFO] Finished at: Mon Sep 14 16:23:35 EST 2009
```

```
[INFO] Final Memory: 55M/108M
```

```
[INFO] -----
```

```
[INFO] -----
```

RELATED TOPICS

[maven-clover2-plugin Site Docs](#)

[Maven Clover User Guide](#)

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-compile

This page describes the shell script `atlas-compile`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-compile [options]` – Compiles the sources of your project. (Runs `mvn compile`.) Passes all parameters straight through to Maven.

Parameters

This shell script is a Maven wrapper script. All parameters are passed straight through to Maven.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-compile -?  
atlas-compile -help
```

Run the following command to see all options provided by `mvn compile`:

```
atlas-mvn compile --help
```

Refer to the [Maven documentation](#) for more information about the `mvn compile` command.

Examples

Run the following command to compile your source code:

```
atlas-compile
```

Run the following command to compile your source code without failing the build, even if there are errors

```
atlas-compile --fail-never
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-create-bamboo-plugin

This page describes the shell script `atlas-create-bamboo-plugin`, part of the [Atlassian Plugin SDK](#).

 There is a specific version of this shell script for each Atlassian application. The shell script described on this page is for **Bamboo**.

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-create-bamboo-plugin [options]` – Creates an example of a Bamboo plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn bamboo:create`.) Interpreted parameters: `artifact-id`, `group-id`, `version`, `package`, `non-interactive`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|----------------------------|-----------------|------------------|--|
| <code>--artifact-id</code> | <code>-a</code> | Yes | Name of the project. This is an identifier for your plugin. It corresponds to the Maven artifactId. If you do not enter this value, the script will prompt you for it. |
| <code>--group-id</code> | <code>-g</code> | Yes | Identifier for the logical group of artifacts associated with the project. Will be used as the default value for <code>package</code> . Corresponds to the Maven groupId. If you do not enter this value, the script will prompt you for it. |

| | | | |
|-------------------|----|-----|---|
| --version | -v | Yes | Version of your plugin. Default is 1.0-SNAPSHOT. If you do not enter this value, the script will prompt you for it. |
| --package | -p | Yes | Name of the Java package that will contain the plugin source code. Default is the value given for group-id. |
| --non-interactive | | No | If this parameter is present, interactive mode is turned off, so that the script will not prompt the user for input. In this case, the script will fail if you do not provide the artifact-id and group-id. |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-create-bamboo-plugin -?
atlas-create-bamboo-plugin -help
```

Examples

Let's assume you want to create a new Bamboo plugin skeleton. Simply open a command window, go to the directory where you want to create the plugin and type:

```
atlas-create-bamboo-plugin
```

Let's assume you want to create a new Bamboo plugin by supplying all the necessary information to the script and allowing the package name to be the same as the group ID. Let's assume you do not want the script to prompt you for information. Type:

```
atlas-create-bamboo-plugin --artifact-id myfooplugin --group-id com.mycompany.plugins --version 1.0 --non-interactive
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)
[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-create-confluence-plugin

This page describes the shell script `atlas-create-confluence-plugin`, part of the [Atlassian Plugin SDK](#).

 There is a specific version of this shell script for each Atlassian application. The shell script described on this page is for **Confluence**.

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-create-confluence-plugin [options]` – Creates an example of a Confluence plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn confluence:create`.) Interpreted parameters: `artifact-id`, `group-id`, `version`, `package`, `non-interactive`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just

support the full version.

- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|-------------------|-----------|------------------|--|
| --artifact-id | -a | Yes | Name of the project. This is an identifier for your plugin. It corresponds to the Maven artifactId. If you do not enter this value, the script will prompt you for it. |
| --group-id | -g | Yes | Identifier for the logical group of artifacts associated with the project. Will be used as the default value for package. Corresponds to the Maven groupId. If you do not enter this value, the script will prompt you for it. |
| --version | -v | Yes | Version of your plugin. Default is 1.0-SNAPSHOT. If you do not enter this value, the script will prompt you for it. |
| --package | -p | Yes | Name of the Java package that will contain the plugin source code. Default is the value given for group-id. |
| --non-interactive | | No | If this parameter is present, interactive mode is turned off, so that the script will not prompt the user for input. In this case, the script will fail if you do not provide the artifact-id and group-id. |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-create-confluence-plugin -?  
atlas-create-confluence-plugin -help
```

Examples

Let's assume you want to create a new Confluence plugin skeleton. Simply open a command window, go to the directory where you want to create the plugin and type:

```
atlas-create-confluence-plugin
```

Let's assume you want to create a new Confluence plugin by supplying all the necessary information to the script and allowing the package name to be the same as the group ID. Let's assume you do not want the script to prompt you for information. Type:

```
atlas-create-confluence-plugin --artifact-id myfooplugin --group-id com.mycompany.plugins  
--version 1.0 --non-interactive
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-create-crowd-plugin

This page describes the shell script `atlas-create-crowd-plugin`, part of the [Atlassian Plugin SDK](#).

 There is a specific version of this shell script for each Atlassian application. The shell script described on this page is for **Crowd**.

On this page:

- Basic Usage
- Parameters
- Getting Help

- Examples

Basic Usage

`atlas-create-crowd-plugin [options]` – Creates an example of a Crowd plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn crowd:create`.) Interpreted parameters: `artifact-id`, `group-id`, `version`, `package`, `non-interactive`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|--------------------------------|-----------------|------------------|--|
| <code>--artifact-id</code> | <code>-a</code> | Yes | Name of the project. This is an identifier for your plugin. It corresponds to the Maven artifactId. If you do not enter this value, the script will prompt you for it. |
| <code>--group-id</code> | <code>-g</code> | Yes | Identifier for the logical group of artifacts associated with the project. Will be used as the default value for <code>package</code> . Corresponds to the Maven groupId. If you do not enter this value, the script will prompt you for it. |
| <code>--version</code> | <code>-v</code> | Yes | Version of your plugin. Default is <code>1.0-SNAPSHOT</code> . If you do not enter this value, the script will prompt you for it. |
| <code>--package</code> | <code>-p</code> | Yes | Name of the Java package that will contain the plugin source code. Default is the value given for <code>group-id</code> . |
| <code>--non-interactive</code> | | No | If this parameter is present, interactive mode is turned off, so that the script will not prompt the user for input. In this case, the script will fail if you do not provide the <code>artifact-id</code> and <code>group-id</code> . |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-create-crowd-plugin -?
atlas-create-crowd-plugin -help
```

Examples

Let's assume you want to create a new Crowd plugin skeleton. Simply open a command window, go to the directory where you want to create the plugin and type:

```
atlas-create-crowd-plugin
```

Let's assume you want to create a new Crowd plugin by supplying all the necessary information to the script and allowing the package name to be the same as the group ID. Let's assume you do not want the script to prompt you for information. Type:

```
atlas-create-crowd-plugin --artifact-id myfooplugin --group-id com.mycompany.plugins --version 1.0
--non-interactive
```

RELATED TOPICS

atlas-create-fecru-plugin

This page describes the shell script `atlas-create-fecru-plugin`, part of the Atlassian Plugin SDK.

 There is a specific version of this shell script for each Atlassian application. The shell script described on this page is for **FishEye** and **Crucible**. Because these two applications share the same code base, you will use the same shell script to create your plugin for either application.

On this page:

- Basic Usage
- Parameters
- Getting Help
- Examples

Basic Usage

`atlas-create-fecru-plugin [options]` – Creates an example of a FishEye or Crucible plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn fecru:create`.) Interpreted parameters: `artifact-id`, `group-id`, `version`, `package`, `non-interactive`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|--------------------------------|-----------------|------------------|--|
| <code>--artifact-id</code> | <code>-a</code> | Yes | Name of the project. This is an identifier for your plugin. It corresponds to the Maven artifactId. If you do not enter this value, the script will prompt you for it. |
| <code>--group-id</code> | <code>-g</code> | Yes | Identifier for the logical group of artifacts associated with the project. Will be used as the default value for <code>package</code> . Corresponds to the Maven groupId. If you do not enter this value, the script will prompt you for it. |
| <code>--version</code> | <code>-v</code> | Yes | Version of your plugin. Default is <code>1.0-SNAPSHOT</code> . If you do not enter this value, the script will prompt you for it. |
| <code>--package</code> | <code>-p</code> | Yes | Name of the Java package that will contain the plugin source code. Default is the value given for <code>group-id</code> . |
| <code>--non-interactive</code> | | No | If this parameter is present, interactive mode is turned off, so that the script will not prompt the user for input. In this case, the script will fail if you do not provide the <code>artifact-id</code> and <code>group-id</code> . |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-create-fecru-plugin -?
atlas-create-fecru-plugin -help
```

Examples

Let's assume you want to create a new FishEye or Crucible plugin skeleton. Simply open a command window, go to the directory where you want to create the plugin and type:

```
atlas-create-fecru-plugin
```

Let's assume you want to create a new FishEye or Crucible plugin by supplying all the necessary information to the script and allowing the package name to be the same as the group ID. Let's assume you do not want the script to prompt you for information. Type:

```
atlas-create-fecru-plugin --artifact-id myfooplugin --group-id com.mycompany.plugins --version 1.0  
--non-interactive
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)
[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-create-home-zip

This page describes the shell script `atlas-create-home-zip`, part of the [Atlassian Plugin SDK](#).

 This command is only available in [Atlassian Plugin SDK](#) version 3.1 and later

On this page:

- [Basic Usage](#)
- [Examples](#)

Basic Usage

`atlas-create-home-zip` – Creates a test-resources zip of the current application's home directory. This zip file can then be pointed to in the AMPS productDataPath property to auto-populate application data during startup.

Examples

Let's assume you want to prepopulate JIRA with a few projects and issues for testing on every clean startup. First, run the application as you normally would using `atlas-run`. Next, use the JIRA web interface to create your projects and issues. Stop JIRA (CTRL+C). Finally, go to the root directory of your plugin and type:

```
atlas-create-home-zip
```

This will create a file called **generated-test-resources.zip** in the plugins target folder. Copy this file to a known location (such as, `src/test/resources`). Finally add:

```
<productDataPath>${basedir}/src/test/resources/generated-test-resources.zip</productDataPath>
```

to the configuration of the maven-jira-plugin in your pom.

Now every time you start JIRA using `atlas-run` or `atlas-debug` JIRA's home directory will be prepopulated using the contents of the generated zip file.

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)
[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-create-jira-plugin

This page describes the shell script `atlas-create-jira-plugin`, part of the [Atlassian Plugin SDK](#).

 There is a specific version of this shell script for each Atlassian application. The shell script described on this page is for **JIRA**.

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)

- Examples

Basic Usage

`atlas-create-jira-plugin [options]` – Creates an example of a JIRA plugin, which you can adapt to suit your own plugin's needs.
(Runs `mvn jira:create`.) Interpreted parameters: `artifact-id`, `group-id`, `version`, `package`, `non-interactive`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|--------------------------------|-----------------|------------------|--|
| <code>--artifact-id</code> | <code>-a</code> | Yes | Name of the project. This is an identifier for your plugin. It corresponds to the Maven artifactId. If you do not enter this value, the script will prompt you for it. In this case, the script will fail if you do not provide the <code>artifact-id</code> and <code>group-id</code> . |
| <code>--group-id</code> | <code>-g</code> | Yes | Identifier for the logical group of artifacts associated with the project. Will be used as the default value for <code>package</code> . Corresponds to the Maven groupId. If you do not enter this value, the script will prompt you for it. |
| <code>--version</code> | <code>-v</code> | Yes | Version of your plugin. Default is <code>1.0-SNAPSHOT</code> . If you do not enter this value, the script will prompt you for it. |
| <code>--package</code> | <code>-p</code> | Yes | Name of the Java package that will contain the plugin source code. Default is the value given for <code>group-id</code> . |
| <code>--non-interactive</code> | | No | If this parameter is present, interactive mode is turned off, so that the script will not prompt the user for input. |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-create-jira-plugin -?
atlas-create-jira-plugin -help
```

Examples

Let's assume you want to create a new JIRA plugin skeleton. Simply open a command window, go to the directory where you want to create the plugin and type:

```
atlas-create-jira-plugin
```

Let's assume you want to create a new JIRA plugin by supplying all the necessary information to the script and allowing the package name to be the same as the group ID. Let's assume you do not want the script to prompt you for information. Type:

```
atlas-create-jira-plugin --artifact-id myfooplugin --group-id com.mycompany.plugins --version 1.0
--non-interactive
```

RELATED TOPICS

atlas-create-refapp-plugin

This page describes the shell script `atlas-create-refapp-plugin`, part of the Atlassian Plugin SDK.

 There is a specific version of this shell script for each Atlassian application. The shell script described on this page is for the **Atlassian Reference Application**, also called the [RefApp](#).

On this page:

- Basic Usage
- Parameters
- Getting Help
- Examples

Basic Usage

`atlas-create-refapp-plugin [options]` – Creates an example of a RefApp plugin, which you can adapt to suit your own plugin's needs. (Runs `mvn refapp:create`.) Interpreted parameters: `artifact-id`, `group-id`, `version`, `package`, `non-interactive`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|--------------------------------|-----------------|------------------|--|
| <code>--artifact-id</code> | <code>-a</code> | Yes | Name of the project. This is an identifier for your plugin. It corresponds to the Maven artifactId. If you do not enter this value, the script will prompt you for it. |
| <code>--group-id</code> | <code>-g</code> | Yes | Identifier for the logical group of artifacts associated with the project. Will be used as the default value for <code>package</code> . Corresponds to the Maven groupId. If you do not enter this value, the script will prompt you for it. |
| <code>--version</code> | <code>-v</code> | Yes | Version of your plugin. Default is <code>1.0-SNAPSHOT</code> . If you do not enter this value, the script will prompt you for it. |
| <code>--package</code> | <code>-p</code> | Yes | Name of the Java package that will contain the plugin source code. Default is the value given for <code>group-id</code> . |
| <code>--non-interactive</code> | | No | If this parameter is present, interactive mode is turned off, so that the script will not prompt the user for input. In this case, the script will fail if you do not provide the <code>artifact-id</code> and <code>group-id</code> . |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-create-refapp-plugin -?
atlas-create-refapp-plugin -help
```

Examples

Let's assume you want to create a new RefApp plugin skeleton. Simply open a command window, go to the directory where you want to

create the plugin and type:

```
atlas-create-refapp-plugin
```

Let's assume you want to create a new RefApp plugin by supplying all the necessary information to the script and allowing the package name to be the same as the group ID. Let's assume you do not want the script to prompt you for information. Type:

```
atlas-create-refapp-plugin --artifact-id myfooplugin --group-id com.mycompany.plugins --version 1.0 --non-interactive
```

RELATED TOPICS

[Using the Atlassian RefApp](#)

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-debug

This page describes the shell script `atlas-debug`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-debug [options]` – Runs the application in debug mode with your plugin installed. (Runs `mvnamps:debug`.) Interpreted parameters: `version`, `container`, `http-port`, `context-path`, `server`, `jvmargs`, `log4j`, `test-version`, `sal-version`, `rest-version`, `plugins`, `lib-plugins`, `bundled-plugins`, `product`, `jvm-debug-port`, `jvm-debug-suspend`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|----------------|-----------|------------------|--|
| --version | -v | Yes | Version of the application to run. Default is RELEASE, i.e. the latest publicly-available milestone release. Examples: <ul style="list-style-type: none">• 3.1• 3.1-m7 |
| --container | -c | Yes | Container to run in. Default is tomcat6x. Other available values are tomcat5x, resin3x and jboss42x. |
| --http-port | -p | Yes | HTTP port for the servlet container. The defaults are as described in the SDK overview . You may need to change this if you already have a process listed for the default port, such as when you want to bring up two instances of Confluence. |

| | | | |
|---------------------|--|-----|--|
| --context-path | | Yes | The application context path. You will need to include the leading forward slash. For example, if your application is running at {{ http://localhost:1990/confluence }} then you should enter /confluence. If your application is running at {{ http://localhost:1990/ }} then you should enter /. |
| --server | | Yes | Host name of the application server. The default is localhost. |
| --jvmargs | | Yes | Additional JVM arguments if required. |
| --log4j | | Yes | Path and file name of the Log4j properties file. |
| --test-version | | Yes | Version to use for test resources. Default is LATEST. |
| --sal-version | | Yes | Version of SAL (Shared Access Layer) to use. |
| --rest-version | | Yes | Version of the Atlassian REST module to use. |
| --plugins | | Yes | A list of plugin artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. These plugins will be installed into your local version of your plugin's host application. |
| --lib-plugins | | Yes | A list of lib artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. Use this to add additional JARs into your /lib folder. |
| --bundled-plugins | | Yes | A list of bundled plugin artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. These plugins will be loaded as bundled plugins in your local version of your plugin's host application. |
| --product | | Yes | The application to launch with the plugin. You might use this if your plugin is written for one application (as specified in the POM) but you want to install the plugin into another application. |
| --jvm-debug-port | | Yes | Port open to accept connections for remote debugging. Default is 5005. |
| --jvm-debug-suspend | | No | If this parameter is present, the JVM will be suspended until the debugger connects. |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-debug -?
atlas-debug -help
```

Examples

Running a Plugin in an Application

Say you want to install and run your plugin in your host Atlassian application, attaching your debugger. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-debug
```

Note that the above shell script will work for any host application, including Confluence, JIRA, etc. The script will determine the application, based on your plugin's specifications.

Specifying a Version of the Application

Say you want to run your plugin with Confluence 2.10.3, attaching your debugger. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-clean  
atlas-debug --version 2.10.3
```

Say you want to run your plugin with JIRA 4.0 snapshot. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-clean  
atlas-debug --version 4.0-SNAPSHOT
```

Note: Running `atlas-clean` will clear the previous version of the host application from your build output directory. You only need to do this if the previous application version was different from the one you need now.

Specifying an Application Server (Container)

Say you want to run your plugin with Confluence 2.10.3 and JBoss 4.2.x. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-debug --version 2.10.3 --container jboss42x
```

Specifying a Version of SAL

Say you want to run that plugin but with Confluence 2.10.3 and SAL 2.0.5:

```
atlas-debug --version 2.10.3 --sal-version 2.0.5
```

Running your Plugin in Multiple Applications

Say you want to run that RefApp plugin in multiple applications simultaneously, with debugger attached. In three separate tabs in your terminal (command window):

- Type the following to run the plugin in the RefApp:

```
atlas-debug
```

- Type the following to run the plugin in Confluence:

```
atlas-debug --product confluence --version 3.0-m9
```

- Type the following to run the plugin in JIRA:

```
atlas-debug --product jira --version 4.0-SNAPSHOT
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-help

This page describes the shell script `atlas-help`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Examples and More about Getting Help](#)

Basic Usage

`atlas-help [--verbose]` – Displays help text for the shell scripts incorporated into the Atlassian Plugin SDK. Interpreted parameters: `verbose`.

Parameters

| Full Parameter | Shortened | Accepts a Value? | Description |
|----------------|-----------|------------------|-------------|
|----------------|-----------|------------------|-------------|

| | | | |
|-----------|--|----|--|
| --verbose | | No | Lists all the shell scripts and the full help text for each command. |
|-----------|--|----|--|

Examples and More about Getting Help



Help is at hand

Make sure you are not in the Maven command line interface (CLI) when you enter the help commands described below. If you are in the CLI, your command line will start with `maven2>`, and you will need to exit from the CLI first. Enter `quit`, `exit` or a friendly `bye`.

Getting an Overview of All the Scripts

Enter the following shell script to see an overview of all the scripts with a brief outline of their functionality:

Enter the following to see all possible help content:

Getting Help Text per Script

Each shell script provides help text if the first argument of the script is one of the following:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

Examples:

Reading the Reference Guide

See the detailed [guide](#) to all scripts.

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-install-idea-plugin (not published)



Not published

This page is visible to **Atlassian staff only**, because the `atlas-install-idea-plugin` script has not yet been released as part of the SDK.

This page describes the shell script `atlas-install-idea-plugin`, part of the Atlassian Plugin SDK.

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-install-idea-plugin` – Configures IntelliJ IDEA for CLI integration. (Runs `mvn amps:idea`.)

Parameters

None.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-install-idea-plugin -?
atlas-install-idea-plugin -help
```

Examples



To be completed

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-install-plugin

This page describes the shell script `atlas-install-plugin`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-install-plugin [options]` – Installs the plugin into a running application. (Runs `mvnamps:install`.) Interpreted parameters: `http-port`, `context-path`, `server`, `username`, `password`, `plugin-key`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted Parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|-----------------------------|-----------------|------------------|--|
| <code>--http-port</code> | <code>-p</code> | Yes | HTTP port for the servlet container. The defaults are as described in the SDK overview . You may need to change this if you already have a process listed for the default port, such as when you want to bring up two instances of Confluence. |
| <code>--context-path</code> | | Yes | The application context path. You will need to include the leading forward slash. For example, if your application is running at {{ http://localhost:1990/confluence }} then you should enter /confluence. If your application is running at {{ http://localhost:1990/ }} then you should enter /. |
| <code>--server</code> | | Yes | Host name of the application server. The default is <code>localhost</code> . |
| <code>--username</code> | | Yes | Username for the administrator account on the plugin's host application. Default is <code>admin</code> . |
| <code>--password</code> | | Yes | Password for the above administrator account on the plugin's host application. Default is <code>admin</code> . |
| <code>--plugin-key</code> | | Yes | Unique key identifying the plugin. This is the full key specified in your <code>atlassian-plugin.xml</code> , including the group and artifact IDs. For example, "com.atlassian.confluence.plugins.example". |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-install-plugin -?
atlas-install-plugin -help
```

Examples

Go to the plugin's project directory (where you created the plugin) and run the following command to install your plugin com.atlassian.confluence.plugins.example into the host application, where the administration username is 'myadmin' with password 'secret'.

```
atlas-install-plugin --username myadmin --password secret --plugin-key
com.atlassian.confluence.plugins.example
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-integration-test

This page describes the shell script `atlas-integration-test`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-integration-test [options]` – Runs the integration tests for the plugin. (Runs `mvn integration-test`.) Interpreted parameters: `version`, `container`, `http-port`, `context-path`, `server`, `jvmargs`, `log4j`, `test-version`, `sal-version`, `rest-version`, `plugins`, `lib-plugins`, `bundled-plugins`, `product`, `no-webapp`, `skip-tests`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|----------------|-----------|------------------|---|
| --version | -v | Yes | Version of the application to run. Default is RELEASE. Examples: <ul style="list-style-type: none">• 3.1• 3.1-m7 |
| --container | -c | Yes | Container to run in. Default is tomcat6x. Other available values are tomcat5x, resin3x and jboss42x. |

| | | | |
|-------------------|----|-----|---|
| --http-port | -p | Yes | HTTP port for the servlet container. The defaults are as described in the SDK overview . You may need to change this if you already have a process listed for the default port, such as when you want to bring up two instances of Confluence. |
| --context-path | | Yes | The application context path. You will need to include the leading forward slash. For example, if your application is running at <code>http://localhost:1990/confluence</code> then you should enter <code>confluence</code> . If your application is running at <code>http://localhost:1990/</code> then you should enter <code>/</code> . |
| --server | | Yes | Host name of the application server. The default is <code>localhost</code> . |
| --jvmargs | | Yes | Additional JVM arguments if required. |
| --log4j | | Yes | Log4j properties file. |
| --test-version | | Yes | Version to use for test resources. Default is <code>LATEST</code> . |
| --sal-version | | Yes | Version of SAL (Shared Access Layer) to use. |
| --rest-version | | Yes | Version of the Atlassian REST module to use. |
| --plugins | | Yes | A list of plugin artifacts, separated by commas, in the form <code>GROUP_ID:ARTIFACT_ID:VERSION</code> . Version is optional. Default is <code>LATEST</code> . These plugins will be installed into your local version of your plugin's host application. |
| --lib-plugins | | Yes | A list of lib artifacts, separated by commas, in the form <code>GROUP_ID:ARTIFACT_ID:VERSION</code> . Version is optional. Default is <code>LATEST</code> . Use this to add additional JARs into your <code>/lib</code> folder. |
| --bundled-plugins | | Yes | A list of bundled plugin artifacts, separated by commas, in the form <code>GROUP_ID:ARTIFACT_ID:VERSION</code> . Version is optional. Default is <code>LATEST</code> . These plugins will be loaded as bundled plugins in your local version of your plugin's host application. |
| --product | | Yes | The application to launch with the plugin. You might use this if your plugin is written for one application (as specified in the POM) but you want to install the plugin into another application. |
| --no-webapp | | No | If this parameter is present, then the script will not start up the plugin's host application. |
| --skip-tests | | No | If this parameter is present, the script will not execute the unit or integration tests. |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-integration-test -?
atlas-integration-test -help
```

Examples

You can run your integration tests against a different application. Say you have a RefApp plugin but want to run your integration tests against Confluence:

```
atlas-integration-test --product confluence
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)
[Developing your Plugin using the Atlassian Plugin SDK](#)
[Plugin Testing Resources and Discussion](#)

atlas-mvn

This page describes the shell script `atlas-mvn`, part of the [Atlassian Plugin SDK](#).



Do not use your local Maven

When running Maven commands against your project, make sure that you use the version of Maven bundled with the Atlassian Plugin SDK. This is important if you have a local version of Maven installed, as well as the Atlassian Plugin SDK. The simplest way is to use the `atlas-mvn` wrapper command instead of `mvn`. Another way is to [put the bundled Maven on your path](#).

On this page:

- Basic Usage
- Parameters
- Getting Help
- Examples

Basic Usage

`atlas-mvn [options]` – Allows you to execute any Maven command using the version of Maven bundled with your Atlassian Plugin SDK. (Runs `mvn`.) Passes all parameters straight through to Maven.

Parameters

This shell script is a Maven wrapper script. All parameters are passed straight through to Maven.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-mvn -?
atlas-mvn --help
```

Examples

Run the following command to compile the Java JUnit test classes:

```
atlas-mvn test-compile
```

Run the following command to get help on the Maven `test-compile` goal:

```
atlas-mvn test-compile --help
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-package

This page describes the shell script `atlas-package`, part of the [Atlassian Plugin SDK](#).

On this page:

- Basic Usage
- Parameters
- Getting Help
- Examples

Basic Usage

`atlas-package[options]` – Packages the plugin artifacts and produces the JAR. (Runs `mvn package`.) Passes all parameters straight through to Maven.

Parameters

This shell script is a Maven wrapper script. All parameters are passed straight through to Maven.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-package -?  
atlas-package -help
```

Run the following command to see all options provided by mvn package:

```
atlas-mvn package --help
```

Examples

Run the following command to produce your JAR:

```
atlas-package
```

Run the following command to produce your JAR without failing the build, even if there are errors

```
atlas-package --fail-never
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-release

This page describes the shell script `atlas-release`, part of the [Atlassian Plugin SDK](#).

On this page:

- Basic Usage
- Parameters
- Getting Help
- Examples

Basic Usage

`atlas-release [options]` – Performs a release of the current plugin. (Runs `mvn release:prepare release:perform`.) Passes all parameters straight through to Maven.

Parameters

This shell script is a Maven wrapper script. All parameters are passed straight through to Maven.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-release -?  
atlas-release -help
```

Run the following command to see all options provided by `mvn release`:

```
atlas-mvn release --help
```

Examples

Run the following command to release your plugin:

```
atlas-release
```

Run the following command to release your plugin without failing the build, even if there are errors

```
atlas-release --fail-never
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-release-rollback

This page describes the shell script `atlas-release-rollback`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-release-rollback [options]` – Rolls back a release of the current plugin. (Runs `mvn release:rollback`.) Passes all parameters straight through to Maven.

Parameters

This shell script is a Maven wrapper script. All parameters are passed straight through to Maven.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-release-rollback -?  
atlas-release-rollback -help
```

Run the following command to see all options provided by `mvn release`:

```
atlas-mvn release --help
```

Examples

Run the following command to roll back a previous release of your plugin:

```
atlas-release-rollback
```

Run the following command to roll back a previous release of your plugin without failing the build, even if there are errors

```
atlas-release-rollback --fail-never
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-run

This page describes the shell script `atlas-run`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-run [options]` – Runs the application with your plugin installed. (Runs `mvnamps:run`.) Interpreted parameters: `version`, `container`, `http-port`, `context-path`, `server`, `jvmargs`, `log4j`, `test-version`, `sal-version`, `rest-version`, `plugins`, `lib-plugins`, `bundled-plugins`, `product`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|----------------|-----------|------------------|--|
| --version | -v | Yes | Version of the application to run. Default is <code>RELEASE</code> . Examples: <ul style="list-style-type: none">• <code>3.1</code>• <code>3.1-m7</code> |
| --container | -c | Yes | Container to run in. Default is <code>tomcat6x</code> . Other available values are <code>tomcat5x</code> , <code>resin3x</code> and <code>jboss42x</code> . |
| --http-port | -p | Yes | HTTP port for the servlet container. The defaults are as described in the SDK overview . You may need to change this if you already have a process listed for the default port, such as when you want to bring up two instances of Confluence. |
| --context-path | | Yes | The application context path. You will need to include the leading forward slash. For example, if your application is running at {{ http://localhost:1990/confluence }} then you should enter <code>/confluence</code> . If your application is running at {{ http://localhost:1990 }} then you should enter <code>/</code> . |
| --server | | Yes | Host name of the application server. The default is <code>localhost</code> . |
| --jvmargs | | Yes | Additional JVM arguments if required. |

| | | | |
|-------------------|--|-----|---|
| --log4j | | Yes | Log4j properties file. |
| --test-version | | Yes | Version to use for test resources. Default is LATEST. |
| --sal-version | | Yes | Version of SAL (Shared Access Layer) to use. |
| --rest-version | | Yes | Version of the Atlassian REST module to use. |
| --plugins | | Yes | A list of plugin artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. These plugins will be installed into your local version of your plugin's host application. |
| --lib-plugins | | Yes | A list of lib artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. Use this to add additional JARs into your /lib folder. |
| --bundled-plugins | | Yes | A list of bundled plugin artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. These plugins will be loaded as bundled plugins in your local version of your plugin's host application. |
| --product | | Yes | The application to launch with the plugin. You might use this if your plugin is written for one application (as specified in the POM) but you want to install the plugin into another application. |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-run -?
atlas-run -help
```

Examples

Running a Plugin in an Application

Say you want to install and run your plugin in your host Atlassian application. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-run
```

Note that the above shell script will work for any host application, including Confluence, JIRA, etc. The script will determine the application, based on your plugin's specifications.

Specifying a Version of the Application

Say you want to run your plugin with Confluence 2.10.3. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-clean
atlas-run --version 2.10.3
```

Say you want to run your plugin with JIRA 4.0 snapshot. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-clean
atlas-run --version 4.0-SNAPSHOT
```

Note: Running `atlas-clean` will clear the previous version of the host application from your build output directory. You only need to do this if the previous application version was different from the one you need now.

Specifying an Application Server (Container)

Say you want to run your plugin with Confluence 2.10.3 and JBoss 4.2.x. Go to the plugin's project directory (where you created the plugin) and type:

```
atlas-run --version 2.10.3 --container jboss42x
```

Specifying a Version of SAL

Say you want to run that plugin but with Confluence 2.10.3 and SAL 2.0.5:

```
atlas-run --version 2.10.3 --sal-version 2.0.5
```

Running your Plugin in Multiple Applications

Say you want to run that RefApp plugin in multiple applications simultaneously. In three separate tabs in your terminal (command window):

- Type the following to run the plugin in the RefApp:

```
atlas-run
```

- Type the following to run the plugin in Confluence:

```
atlas-run --product confluence --version 3.0-m9
```

- Type the following to run the plugin in JIRA:

```
atlas-run --product jira --version 4.0-SNAPSHOT
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-run-standalone

This page describes the shell script `atlas-run-standalone`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)
- [Limitations](#)

Basic Usage

`atlas-run-standalone [options]` – Runs an Atlassian application standalone, without a plugin project (i.e. not requiring `atlas-create-<product>-plugin`). Interpreted parameters: `version`, `container`, `http-port`, `context-path`, `server`, `jvmargs`, `log4j`, `test-version`, `sal-version`, `rest-version`, `plugins`, `lib-plugins`, `bundled-plugins`, `product`.

Parameters

This shell script supports some interpreted parameters, specified below. All other parameters are passed straight through to Maven.

Explaining the table columns:

- 'Full Parameter' and 'Shortened' – Some parameters allow you to choose a full or a shortened version of the parameter, others just support the full version.
- 'Accepts a Value?' – 'Yes' means that you should specify a value in addition to the parameter, e.g. for the 'version' parameter you should specify the version number. 'No' means that the parameter does not accept a value.

Interpreted parameters:

| Full Parameter | Shortened | Accepts a Value? | Description |
|--------------------------|-----------------|------------------|---|
| <code>--version</code> | <code>-v</code> | Yes | Version of the application to run. Default is <code>RELEASE</code> . Examples: <ul style="list-style-type: none">• <code>3.1</code>• <code>3.1-m7</code> |
| <code>--container</code> | <code>-c</code> | Yes | Container to run in. Default is <code>tomcat6x</code> . Other available values are <code>tomcat5x</code> , <code>resin3x</code> and <code>jboss42x</code> . |

| | | | |
|-------------------|----|-----|--|
| --http-port | -p | Yes | HTTP port for the servlet container. The defaults are as described in the SDK overview . You may need to change this if you already have a process listed for the default port, such as when you want to bring up two instances of Confluence. |
| --context-path | | Yes | The application context path. You will need to include the leading forward slash. For example, if your application is running at http://localhost:1990/confluence then you should enter '/confluence'. If your application is running at http://localhost:1990/ then you should enter '/'. |
| --server | | Yes | Host name of the application server. The default is localhost. |
| --jvmargs | | Yes | Additional JVM arguments if required. |
| --log4j | | Yes | Log4j properties file. |
| --test-version | | Yes | Version to use for test resources. Default is LATEST. |
| --sal-version | | Yes | Version of SAL (Shared Access Layer) to use. |
| --rest-version | | Yes | Version of the Atlassian REST module to use. |
| --plugins | | Yes | A list of plugin artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. These plugins will be installed into the application. |
| --lib-plugins | | Yes | A list of lib artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. Use this to add additional JARs into your /lib folder. |
| --bundled-plugins | | Yes | A list of bundled plugin artifacts, separated by commas, in the form GROUP_ID:ARTIFACT_ID:VERSION. Version is optional. Default is LATEST. These plugins will be loaded as bundled plugins in the application. |
| --product | | Yes | The application to launch. |

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- -?
- -h
- help
- -help
- --help

For example:

```
atlas-run-standalone -?
atlas-run-standalone -help
```

Examples

Running an Application

Say you want to quickly spin up an instance of JIRA, without having previously created a JIRA plugin project using `atlas-create-jira-plugin`. At the command line, type:

```
atlas-run-standalone --product jira
```

This will create a directory called `amps-standalone` in your current working directory, which will contain a JIRA home directory, database and server logs.

Specifying a Version of the Application

Say you want to start up Confluence version 3.2.2 specifically, rather than the latest version. At the command line, type:

```
atlas-run-standalone --product confluence --version 3.2.2
```

Limitations

Currently if you want to run multiple products, or multiple versions of each product, you must either use different working directories to contain the `amps-standalone` data for each product, or you must manually remove the `target` directory under `amps-standalone`. See issue [AMPS-403](#).

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)
[Developing your Plugin using the Atlassian Plugin SDK](#)

atlas-unit-test

This page describes the shell script `atlas-unit-test`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)
- [Getting Help](#)
- [Examples](#)

Basic Usage

`atlas-unit-test [options]` – Runs the unit tests for your plugin. (Runs `mvn test`.) Passes all parameters straight through to Maven.

Parameters

This shell script is a Maven wrapper script. All parameters are passed straight through to Maven.

Getting Help

The shell script will display some help text if you enter one of the following as the first argument:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

For example:

```
atlas-unit-test -?  
atlas-unit-test -help
```

Run the following command to see all options provided by `mvn test`:

```
atlas-mvn test --help
```

Examples

Run the following command to run your unit tests:

```
atlas-unit-test
```

Run the following command to run your unit tests without failing the build, even if there are errors:

```
atlas-unit-test --fail-never
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)
[Developing your Plugin using the Atlassian Plugin SDK](#)
[Plugin Testing Resources and Discussion](#)

atlas-version

This page describes the shell script `atlas-version`, part of the [Atlassian Plugin SDK](#).

On this page:

- [Basic Usage](#)
- [Parameters](#)

- Examples

Basic Usage

atlas-version – Displays version and runtime information for the Atlassian Plugin SDK. (Runs mvn --version.)

Parameters

None.

Examples

To see the version and runtime information for your installed version of the Atlassian Plugin SDK, run the following shell script:

```
atlas-version
```

Results:

```
ATLAS Version: 3.0-m8
ATLAS Home: C:\Atlassian\atlassian-plugin-sdk-3.0-m8
ATLAS Scripts: C:\Atlassian\atlassian-plugin-sdk-3.0-m8\bin
ATLAS Maven Home: C:\Atlassian\atlassian-plugin-sdk-3.0-m8\apache-mav
-----
Executing: C:\Atlassian\atlassian-plugin-sdk-3.0-m8\apache-maven\bin\
Apache Maven 2.1.0 (r755702; 2009-03-19 06:10:27+1100)
Java version: 1.6.0_04
Java home: C:\Sun\SDK\jdk\jre
Default locale: en_AU, platform encoding: Cp1252
OS name: "windows xp" version: "5.1" arch: "x86" Family: "windows"
```

RELATED TOPICS

[Atlassian Plugin SDK Documentation](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

Atlassian Plugin SDK Release Notes



Current released version – Atlassian Plugin SDK 3.4

Currently-supported applications: **Confluence**, **JIRA**, **Bamboo**, **FishEye**, **Crucible** and **Crowd**.

Atlassian Plugin SDK 3.4 is now available – see the release notes. If you have ideas for improvement or new features, or have found a bug, please raise a ticket on our issue tracker. Snapshot builds are also available for the stout-hearted.



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

[Download Latest Version \(Windows\)](#)



[Download Latest Version \(Mac OS X, UNIX, Linux\)](#)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

Release Notes

- [Atlassian Plugin SDK 3.0 Beta 2 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Beta 1 Release Notes](#)
- [Atlassian Plugin SDK 3.0.5 Release Notes](#)
- [Atlassian Plugin SDK 3.0.4 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Beta 5 Release Notes](#)
- [Atlassian Plugin SDK 3.0.3 Release Notes](#)
- [Atlassian Plugin SDK 3.0.2 Release Notes](#)
- [Atlassian Plugin SDK 3.0.1 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Release Candidate 1 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Beta 8 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Beta 7 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Beta 4 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Beta 3 Release Notes](#)
- [Atlassian Plugin SDK 3.0 Beta 6 Release Notes](#)
- [Atlassian Plugin SDK 3.0.6 Release Notes](#)
- [Atlassian Plugin SDK 3.1 Release Notes](#)
- [Atlassian Plugin SDK 3.1.1 Release Notes](#)
- [Atlassian Plugin SDK 3.1.2 Release Notes](#)
- [Atlassian Plugin SDK 3.1.3 Release Notes](#)
- [Atlassian Plugin SDK 3.2 Release Notes](#)
- [Atlassian Plugin SDK 3.2.3 Release Notes](#)
- [Atlassian Plugin SDK 3.2.4 Release Notes](#)
- [Atlassian Plugin SDK 3.3 Release Notes](#)
- [Atlassian Plugin SDK 3.3.1 Release Notes](#)
- [Atlassian Plugin SDK 3.4 Release Notes](#)
- [Atlassian Plugin SDK 3.3.2 Release Notes](#)
- [Atlassian Plugin SDK 3.3.3 Release Notes](#)
- [Atlassian Plugin SDK 3.3.4 Release Notes](#)

Atlassian Plugin SDK 3.0 Beta 2 Release Notes

8 September 2009

Version 3.0 Beta 2 of the Atlassian Plugin SDK is now available and contains some important bug fixes and improvements. The SDK scripts will now pick up the configuration settings from the plugin's `pom.xml`. The JIRA home directory was being overwritten on every startup – this problem is now fixed. And the SDK will turn on Atlassian developer mode in the host applications (`-Datlassian.dev.mode=true`).

Complete List of Fixes in this Release

jiraissues: Unable to determine if sort should be enabled.

Atlassian Plugin SDK 3.0 Beta 1 Release Notes

30 August 2009

Atlassian is delighted to present version 3.0 Beta 1 of the Atlassian Plugin SDK. This is the first public release of the SDK, available in time for [Codegeist](#). We will extend and polish the SDK leading up to [AtlasCamp](#).

The aim of the SDK is to simplify the process of building a basic plugin and installing it into a local version of JIRA, Confluence or another Atlassian application. Then plugin developers can concentrate on the code that makes their plugin do its thing.

Our documentation tells you how to [install the SDK](#) and [create your plugin](#). There's also a detailed [reference guide](#) to all the SDK scripts and features.

Complete List of Fixes in this Release

jiraissues: Unable to determine if sort should be enabled.

Atlassian Plugin SDK 3.0.5 Release Notes

4 February 2010

Version 3.0.5 of the Atlassian Plugin SDK is now available. This is a maintenance release which fixes several issues with archetypes and the behavior of `atlas-debug`. Users are encouraged to upgrade.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.0.4 Release Notes

14 December 2009

Version 3.0.4 of the Atlassian Plugin SDK is now available. This is a maintenance release which fixes several issues with archetypes and the behavior of `atlas-debug`. Users are encouraged to upgrade.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.0 Beta 5 Release Notes

18 September 2009

Version 3.0 Beta 5 of the Atlassian Plugin SDK is now available. This release is a highly recommended upgrade for other beta users.

Notable features in this release:

- FishEye and Crucible support
- New JIRA version

Known Issues

The SDK does not work if installed in a directory with spaces in its name ([AMPS-128](#)) or if used in a plugin directory with spaces in the name ([AMPS-126](#)). To work around this, install the SDK in a directory without spaces and make sure the plugin directories don't have spaces.

Complete List of Fixes in this Release

jiraissues: Unable to determine if sort should be enabled.

Atlassian Plugin SDK 3.0.3 Release Notes

6 December 2009

Version 3.0.3 of the Atlassian Plugin SDK is now available. This is a maintenance release which should fix problems with certain Maven plugin configurations. Users are encouraged to upgrade.



The version number is actually 3.0.3.2 due to some issues when releasing.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.0.2 Release Notes

4 November 2009

Version 3.0.2 of the Atlassian Plugin SDK is now available. This is a maintenance release which should fix the problem JIRA users have with downloading certain artifacts. Users are encouraged to upgrade.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.0.1 Release Notes

27 October 2009

Version 3.0.1 of the Atlassian Plugin SDK is now available. This is a maintenance release which fixes the automatic configuration of Plugins `2 plugin.resource.directories`. Users are encouraged to upgrade.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

- The sample data for Fisheye/Crucible plugins has been updated with sample reviews to make initial plugin development easier.
- Data created in a Fisheye/Crucible application is now only erased when `atlas-clean` is run.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.0 Release Candidate 1 Release Notes

12 October 2009

Version 3.0 Release Candidate 1 of the Atlassian Plugin SDK is now available. This release is a highly recommended upgrade for other beta users.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Atlassian Plugin SDK 3.0 Release Notes

21 October 2009

Version 3.0 of the Atlassian Plugin SDK is now available. This is the first officially supported release.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

- Fisheye and Crucible need extra commands when they are being debugged. When running `atlas-debug`, the SDK will detect if you are using Fisheye or Crucible and print the environment variables you need to use to debug successfully. This sucks, and it will be fixed in a future release.
- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Atlassian Plugin SDK 3.0 Beta 8 Release Notes

6 October 2009

Version 3.0 Beta 8 of the Atlassian Plugin SDK is now available. This release is a highly recommended upgrade for other beta users.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Atlassian Plugin SDK 3.0 Beta 7 Release Notes

30 September 2009

Version 3.0 Beta 7 of the Atlassian Plugin SDK is now available. This release is a highly recommended upgrade for other beta users.

Notable features in this release:

- The SDK is now Open Source, licensed under the [Apache License 2.0](#). All are encouraged to play with the source and make whatever improvements they wish; we're happy to accept community patches under that license's terms.
- The SDK's embedded Maven instance now sets a private copy of the `MAVEN_OPTS` variable. The values for maximum Java heap size and permgen space are 256m, which are in line with the [recommended settings for Atlassian applications](#). If you have set this variable in your own environment, your value will **not** be used.
- Batch scripts have been converted to use CRLF line endings.
- Some missing artifacts, which were causing spurious run failures in JIRA, have been installed in the SDK's Maven repository.

Known Issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Complete List of Fixes in this Release

jiraissues: Unable to determine if sort should be enabled.

Atlassian Plugin SDK 3.0 Beta 4 Release Notes

11 September 2009

Version 3.0 Beta 4 of the Atlassian Plugin SDK is now available. This release contains an important bug fix that fixes an issue that prevented the downloading of Atlassian artifacts. It is a highly recommended upgrade for other beta users.

Notable features in this release:

- Clover support
- Bundling of all jars needed to run plugins in sdk repository

Complete List of Fixes in this Release

jiraissues: Unable to determine if sort should be enabled.

Atlassian Plugin SDK 3.0 Beta 3 Release Notes

8 September 2009

Version 3.0 Beta 3 of the Atlassian Plugin SDK is now available. This release contains an important bug fix that fixes many of the atlas scripts. It is a required upgrade for beta 2 users and a highly recommended upgrade for beta 1 users.

Complete List of Fixes in this Release

jiraissues: Unable to determine if sort should be enabled.

Atlassian Plugin SDK 3.0 Beta 6 Release Notes

26 September 2009

Version 3.0 Beta 6 of the Atlassian Plugin SDK is now available. This release is a highly recommended upgrade for other beta users.

Notable features in this release:

- Crowd support (available with the next release of Crowd, due soon)
- Fix for [AMPS-128](#)

Known Issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.

Complete List of Fixes in this Release

jiraissues: Unable to determine if sort should be enabled.

Atlassian Plugin SDK 3.0.6 Release Notes

1 March 2010

Version 3.0.6 of the Atlassian Plugin SDK is now available. This is a maintenance release which updates the built-in products to the latest available versions. Users are encouraged to upgrade.

This is expected to be the last release in the 3.0 series. 3.1 is expected to ship with JIRA 4.1 and will feature support for that product.

Products supported in this release

| Product | Version |
|------------------|---------|
| JIRA | 4.0.2 |
| Confluence | 3.1.1 |
| Bamboo | 2.5.2 |
| Fisheye/Crucible | 2.2.0 |
| Crowd | 2.0.3 |



Attention Confluence plugin developers

Support for Confluence 3.1.1 requires an upgrade of the functional test library, which is now version 2.3-beta1. This version is not backwards compatible with the previous 2.1.1 shipped in the SDK thus far, so any existing functional tests will have to be slightly modified. (Sorry, but there was no other option.)

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.1 Release Notes

26 April 2010

Version 3.1 of the Atlassian Plugin SDK is now available. This release updates the built-in products to the latest available versions, fixes numerous issues and introduces initial support for "test groups" which facilitate cross-product plugin testing. Users are encouraged to upgrade.

Products supported in this release

| Product | Version |
|------------------|---------|
| JIRA | 4.1.0 |
| Confluence | 3.1.1 |
| Bamboo | 2.5.2 |
| FishEye/Crucible | 2.2.0 |
| Crowd | 2.0.3 |



Attention JIRA plugin developers

There are known security vulnerabilities in JIRA 4.1. Developers using `atlas-run` or `atlas-debug` to run and test JIRA plugins should be aware that these will be running JIRA 4.1 by default. This may not be a problem, for example, if the testing instance is running on a development machine not accessible to others. However, if security of the testing instance is a concern, please open the `pom.xml` file (generated by `atlas-create-jira-plugin`) and update the `<jira.version>` property to 4.1.1.

Notable features in this release:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.1.1 Release Notes

29 April 2010

Hot on the heels of 3.1, version 3.1.1 of the Atlassian Plugin SDK is now available. This release updates the built-in products to the latest available versions (notably JIRA 4.1.1) and fixes an issue concerning packaging of OSGi bundle repository artifacts. Also note that the 3.1 series is the first to include contributions from external developers – thanks Jonathan!

Products supported in this release

| Product | Version |
|------------------|---------|
| JIRA | 4.1.1 |
| Confluence | 3.2 |
| Bamboo | 2.5.4 |
| FishEye/Crucible | 2.2.2 |
| Crowd | 2.0.3 |

Issues resolved in 3.1.1:

jiraissues: Unable to determine if sort should be enabled.

Notable features in 3.1:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., <http://localhost:1990>) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.1.2 Release Notes

3 May 2010

Version 3.1.2 of the Atlassian Plugin SDK is now available. This release fixes a broken Maven compilation dependency on the JIRA test database, which is used when testing JIRA plugins with `atlas-run` or `atlas-debug` (see also Known Issues). JIRA plugin developers using SDK 3.1.1 should update to 3.1.2; others are unaffected.

Products supported in this release

| Product | Version |
|------------------|---------|
| JIRA | 4.1.1 |
| Confluence | 3.2 |
| Bamboo | 2.5.4 |
| FishEye/Crucible | 2.2.2 |
| Crowd | 2.0.3 |

Issues resolved in 3.1.2:

jiraissues: Unable to determine if sort should be enabled.

Issues resolved in 3.1.1:

jiraissues: Unable to determine if sort should be enabled.

Notable features in 3.1:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.1.3 Release Notes

5 May 2010

Version 3.1.3 of the Atlassian Plugin SDK is now available. This release updates the built-in products to the latest available versions, each of which include security fixes and other improvements. Users are encouraged to upgrade.

Products supported in this release

| Product | Version |
|------------------|----------|
| JIRA | 4.1.1 |
| Confluence | 3.2.1_01 |
| Bamboo | 2.5.5 |
| FishEye/Crucible | 2.2.3 |
| Crowd | 2.0.4 |

Issues resolved in 3.1.1 through 3.1.3:

jiraissues: Unable to determine if sort should be enabled.

Notable features in 3.1:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.2 Release Notes

24 August 2010

Version 3.2 of the Atlassian Plugin SDK is now available. This release includes many bugfixes and test configuration improvements, and updates the built-in products to the latest available versions. Users are encouraged to upgrade.

Products supported in this release

| Product | Version |
|------------|---------|
| JIRA | 4.1.2 |
| Confluence | 3.3 |

| | |
|------------------|-------|
| Bamboo | 2.6.1 |
| FishEye/Crucible | 2.2.3 |
| Crowd | 2.0.4 |

Issues resolved and notable features implemented in 3.2:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.2.3 Release Notes

2 November 2010

Version 3.2.3 ("Election Day") of the Atlassian Plugin SDK is now available. This release includes many bugfixes and test configuration improvements, and updates the built-in products to the latest available versions. Users are encouraged to upgrade.

Products supported in this release

| Product | Version |
|------------------|---------|
| JIRA | 4.2 |
| Confluence | 3.4.1 |
| Bamboo | 2.6.2 |
| FishEye/Crucible | 2.4 |
| Crowd | 2.0.7 |

Compatibility changes

- To support the now-bundled Universal Plugin Manager, the Confluence functional test library has been upgraded from 2.3-beta1 to 2.3 final. New plugins created with this SDK release will default to the correct versions. The 2.3 release may not be compatible with versions of Confluence before 3.4.
- The `pi` command supported by `atlas-cli` has had an important bugfix ([AMPS-349](#)). This will cause plugins using older versions of the SDK to break when using `atlas-cli`. To fix, ensure that your plugin's `pom.xml` is configured to use the same version of the Maven product plugin as the SDK. For example:

```
<project>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-confluence-plugin</artifactId>
        <version>3.2.3</version> <!-- make sure this is updated to use the same version as your
SDK -->
        </plugin>
      </plugins>
    </build>
  </project>
```

Changelog

Issues resolved and notable features implemented in 3.2.3:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., <http://localhost:1990>) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.2.4 Release Notes

27 January 2011

Version 3.2.4 ("Happy Birthday, Mozart!") of the Atlassian Plugin SDK is now available. This release fixes a critical bug with the `atlas-create-home-zip` command and updates the built-in products to the latest available versions. Users are encouraged to upgrade.

Products supported in this release

| Product | Version |
|------------------|------------|
| JIRA | 4.2.2-b589 |
| Confluence | 3.4.7 |
| Bamboo | 2.7.3 |
| FishEye/Crucible | 2.4 |
| Crowd | 2.0.7 |

Compatibility changes

- To support the now-bundled Universal Plugin Manager, the Confluence functional test library has been upgraded from 2.3-beta1 to 2.3 final. New plugins created with this SDK release will default to the correct versions. The 2.3 release may not be compatible with versions of Confluence before 3.4.
- The `pi` command supported by `atlas-cli` has had an important bugfix ([AMPS-349](#)). This will cause plugins using older versions of the SDK to break when using `atlas-cli`. To fix, ensure that your plugin's `pom.xml` is configured to use the same version of the Maven product plugin as the SDK. For example:

```
<project>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-confluence-plugin</artifactId>
        <version>3.2.4</version> <!-- make sure this is updated to use the same version as your
SDK -->
        </plugin>
      </plugins>
    </build>
  </project>
```

Changelog

Issues resolved and notable features implemented in 3.2.4:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly (AMPS-126). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) (AMPS-122). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.3 Release Notes

22 March 2011

Version 3.3 ("New and Improved!") of the Atlassian Plugin SDK is now available. This release updates the built-in products to the latest available versions, adds a new `atlas-run-standalone` command to run any Atlassian product (without first needing to use `atlas-create-<product>-plugin`) and includes several bugfixes. Users are encouraged to upgrade.

Products supported in this release

| Product | Version |
|------------------|---------|
| JIRA | 4.3 |
| Confluence | 3.5 |
| Bamboo | 3.0 |
| FishEye/Crucible | 2.5.2 |
| Crowd | 2.2.2 |

Compatibility changes

- To support the now-bundled Universal Plugin Manager, the Confluence functional test library has been upgraded from 2.3-beta1 to 2.3 final. New plugins created with this SDK release will default to the correct versions. The 2.3 release may not be compatible with versions of Confluence before 3.4.
- The `pi` command supported by `atlas-cli` has had an important bugfix (AMPS-349). This will cause plugins using older versions of the SDK to break when using `atlas-cli`. To fix, ensure that your plugin's `pom.xml` is configured to use the same version of the Maven product plugin as the SDK. For example:

```
<project>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-confluence-plugin</artifactId>
        <version>3.3</version> <!-- make sure this is updated to use the same version as your SDK
-->
        </plugin>
      </plugins>
    </build>
  </project>
```

New requirements

The SDK now requires **JDK 6**. You can download the latest version of JDK 6 [here](#).

Changelog

Issues resolved and notable features implemented in 3.3:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly (AMPS-126). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., <http://localhost:1990>) (AMPS-122). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.3.1 Release Notes

26 April 2011

Version 3.3.1 ("Brian's Invaluable Help") of the Atlassian Plugin SDK is now available. This release updates the built-in products to the latest available versions and adds automatic XML schema validation to the plugin archetypes. Users are encouraged to upgrade.



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

Download Latest Version (Windows)



Download Latest Version (Mac OS X, UNIX, Linux)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

Products supported in this release

| Product | Version |
|------------------|----------------|
| JIRA | 4.3.2 |
| Confluence | 3.5.2 |
| Bamboo | 3.0.3 |
| FishEye/Crucible | 2.5.4 |
| Crowd | 2.2.4 |

Compatibility changes

- To support the now-bundled Universal Plugin Manager, the Confluence functional test library has been upgraded from 2.3-beta1 to 2.3 final. New plugins created with this SDK release will default to the correct versions. The 2.3 release may not be compatible with versions of Confluence before 3.4.
- The pi command supported by `atlas-cli` has had an important bugfix (AMPS-349). This will cause plugins using older versions of the SDK to break when using `atlas-cli`. To fix, ensure that your plugin's `pom.xml` is configured to use the same version of the Maven product plugin as the SDK. For example:

```

<project>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-confluence-plugin</artifactId>
        <version>3.3.1</version> <!-- make sure this is updated to use the same version as your
SDK -->
      </plugin>
    </plugins>
  </build>
</project>

```

New requirements

The SDK now requires **JDK 6**. You can download the latest version of JDK 6 [here](#).

Changelog

- The plugin archetypes now include XML schemas for validation and autocompletion. See [Validation with XML schemas](#) for further details and for instructions on applying schemas to your existing plugins.
- An issue preventing the `atlas-cli` and `atlas-install-plugin` commands from working properly with Fisheye/Crucible has been addressed.

Issues resolved and notable features implemented in 3.3.1:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.4 Release Notes

13 May 2011

Atlassian is proud to release version 3.4 of the Atlassian Plugin SDK.

In this update we have provided:

- Updates to the Bamboo default project with an example of the new [Atlassian Bamboo 3.1 Task](#) plugins.

Products supported in this release

| Product | Version |
|------------------|-------------|
| JIRA | 4.3.2, 4.4* |
| Confluence | 3.5.2 |
| Bamboo | 3.1 |
| FishEye/Crucible | 2.5.4 |
| Crowd | 2.2.4 |

New requirements

As with the Atlassian Plugin SDK 3.3 release, **The SDK now requires JDK 6**. You can download the latest version of JDK 6 [here](#).

Changelog

Issues resolved and notable features implemented in 3.4:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

JIRA 4.4

This version of the SDK supports JIRA 4.4, which was not available as of the SDK release. To use JIRA 4.4, follow the same procedure for changing the product and data versions in the POM:

```
<properties>
    <jira.version>4.4</jira.version>
    <jira.data.version>4.4</jira.data.version>
</properties>
```

Atlassian Plugin SDK 3.3.2 Release Notes

17 May 2011

Version 3.3.2 ("Bamboo Petshop Boys Lady Ga-Ga Remix") of the Atlassian Plugin SDK is now available. This release updates the built-in products to the latest available versions and adds automatic XML schema validation to the plugin archetypes. Users are encouraged to upgrade.



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

Download Latest Version (Windows)



Download Latest Version (Mac OS X, UNIX, Linux)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

Products supported in this release

| Product | Version |
|---------|---------|
| JIRA | 4.3.2 |

| | |
|------------------|-------|
| Confluence | 3.5.2 |
| Bamboo | 3.1 |
| FishEye/Crucible | 2.5.4 |
| Crowd | 2.2.4 |

Compatibility changes

- To support the now-bundled Universal Plugin Manager, the Confluence functional test library has been upgraded from 2.3-beta1 to 2.3 final. New plugins created with this SDK release will default to the correct versions. The 2.3 release may not be compatible with versions of Confluence before 3.4.
- The `pi` command supported by `atlas-cli` has had an important bugfix ([AMPS-349](#)). This will cause plugins using older versions of the SDK to break when using `atlas-cli`. To fix, ensure that your plugin's `pom.xml` is configured to use the same version of the Maven product plugin as the SDK. For example:

```
<project>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-confluence-plugin</artifactId>
        <version>3.3.1</version> <!-- make sure this is updated to use the same version as your
SDK -->
        </plugin>
      </plugins>
    </build>
  </project>
```

New requirements

The SDK now requires **JDK 6**. You can download the latest version of JDK 6 [here](#).

Changelog

- Bamboo default project now relies on Bamboo 3.1 and contains an example [Task Plugin](#).

Issues resolved and notable features implemented in 3.3.2:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.3.3 Release Notes

18 May 2011

Version 3.3.3 ("Are friends electric?") of the Atlassian Plugin SDK is now available. This release updates a small bug found in the Bamboo SDK example.



We now have developer mailing lists for our products!

Sign up for [Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

Download Latest Version (Windows)



Download Latest Version (Mac OS X, UNIX, Linux)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

Products supported in this release

| Product | Version |
|------------------|---------|
| JIRA | 4.3.2 |
| Confluence | 3.5.2 |
| Bamboo | 3.1 |
| FishEye/Crucible | 2.5.4 |
| Crowd | 2.2.4 |

Compatibility changes

- To support the now-bundled Universal Plugin Manager, the Confluence functional test library has been upgraded from 2.3-beta1 to 2.3 final. New plugins created with this SDK release will default to the correct versions. The 2.3 release may not be compatible with versions of Confluence before 3.4.
- The `pi` command supported by `atlas-cli` has had an important bugfix (AMPS-349). This will cause plugins using older versions of the SDK to break when using `atlas-cli`. To fix, ensure that your plugin's `pom.xml` is configured to use the same version of the Maven product plugin as the SDK. For example:

```
<project>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-confluence-plugin</artifactId>
        <version>3.3.1</version> <!-- make sure this is updated to use the same version as your
SDK -->
        </plugin>
      </plugins>
    </build>
  </project>
```

New requirements

The **SDK now requires JDK 6**. You can download the latest version of JDK 6 [here](#).

Changelog

- Fix small bug in the Bamboo 3.1 SDK example where configuration is not correctly persisted.

Issues resolved and notable features implemented in 3.3.3:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly (AMPS-126). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., <http://localhost:1990>) (AMPS-122). To work around this, use a named context.

Product-specific issues

None for recent versions.

Atlassian Plugin SDK 3.3.4 Release Notes

18 May 2011

Version 3.3.4 ("Oh XML...") of the Atlassian Plugin SDK is now available. This release removes the XML schemas that were automatically added to `atlassian-plugin.xml` because of a [bug in the plugins system](#).



We now have developer mailing lists for our products!

[Sign up for Developer Updates](#) to get release notifications for new versions, API changes, exclusive offers, and much more.

Download Latest Version (Windows)



Download Latest Version (Mac OS X, UNIX, Linux)



Note: You'll be asked to log in to <http://my.atlassian.com> before you download the SDK. If you don't have an Atlassian account, you can create one for free.

Products supported in this release

| Product | Version |
|------------------|---------|
| JIRA | 4.3.2 |
| Confluence | 3.5.2 |
| Bamboo | 3.1 |
| FishEye/Crucible | 2.5.4 |
| Crowd | 2.2.4 |

Compatibility changes

- To support the now-bundled Universal Plugin Manager, the Confluence functional test library has been upgraded from 2.3-beta1 to 2.3 final. New plugins created with this SDK release will default to the correct versions. The 2.3 release may not be compatible with versions of Confluence before 3.4.
- The `pi` command supported by `atlas-cli` has had an important bugfix (AMPS-349). This will cause plugins using older versions of the SDK to break when using `atlas-cli`. To fix, ensure that your plugin's `pom.xml` is configured to use the same version of the Maven product plugin as the SDK. For example:

```

<project>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-confluence-plugin</artifactId>
        <version>3.3.4</version> <!-- make sure this is updated to use the same version as your
SDK -->
      </plugin>
    </plugins>
  </build>
</project>

```

New requirements

The **SDK now requires JDK 6**. You can download the latest version of JDK 6 [here](#).

Changelog

- Stopped adding xml schemas to `atlassian-plugin.xml`

Issues resolved and notable features implemented in 3.3.4:

jiraissues: Unable to determine if sort should be enabled.

Known Issues

General SDK issues

- If a plugin is in a directory with spaces in the name, the SDK commands will not work properly ([AMPS-126](#)). To work around this, make sure the plugin directory has no spaces.
- The SDK cannot deploy applications to the root context (e.g., `http://localhost:1990`) ([AMPS-122](#)). To work around this, use a named context.

Product-specific issues

None for recent versions.

AMPS Plugin for Maven

The Atlassian Maven Plugin Suite (AMPS) is a suite of Maven 2 plugins that make it easier to develop plugins for Atlassian applications. The project is housed on [StAC](#). This page is about AMPS specifically, and is just a part of the complete [Atlassian Plugin SDK Documentation](#).

On this page:

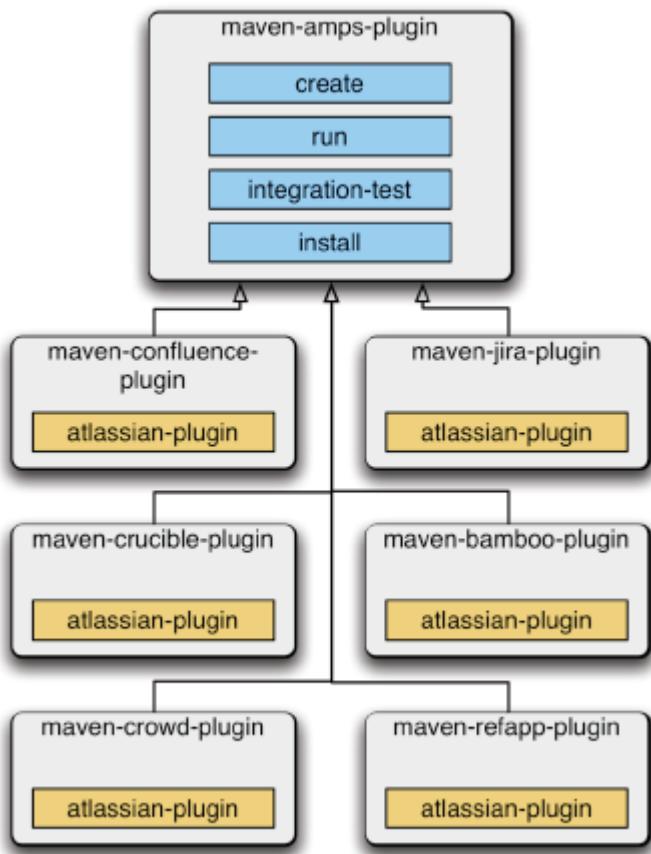
- Overview
- Basic Commands
- Examples
- Advanced Configuration
 - containerId
 - httpPort
 - cliPort
 - contextPath
 - server
 - productVersion
 - jvmArgs
 - systemPropertyVariables
 - log4jProperties
 - productDataVersion
 - productDataPath
 - pluginArtifacts
 - libArtifacts
 - bundledArtifacts
 - pluginDependencies
 - installPlugin
 - output
 - jvmDebugPort
 - jvmDebugSuspend
 - skipTests

- functionalTestPattern
- products
- testGroups
- configuredTestGroupsToRun
- noWebapp
- instructions
- skipManifestValidation

Overview

AMPS provides a set of plugins for Maven 2. AMPS combines multiple plugins, conventions, and POM configuration.

Diagram: AMPS architecture:



The application-specific configuration is as follows:

| Application | AMPS Plugin Type |
|------------------|-------------------------|
| Confluence | maven-confluence-plugin |
| JIRA | maven-jira-plugin |
| Bamboo | maven-bamboo-plugin |
| FishEye/Crucible | maven-fecru-plugin |
| Crowd | maven-crowd-plugin |

Basic Commands

The commands are of the form `mvn APPLICATION:CMD`, where:

- APPLICATION is the name of the product plugin above;
- CMD is one of the commands listed below.

Examples:

- `mvn confluence:run`
- `mvn confluence:create`
- `mvn jira:run`

```
• mvn jira:create
```

In the table below, replace APPLICATION with the name of Atlassian application you are developing your plugin for. Fisheye and Crucible use the same plugin, so APPLICATION should be replaced with fecru if you are developing for Fisheye or Crucible.

| | |
|--|--|
| APPLICATION:create | Create a plugin skeleton from a Maven archetype, specific to the Atlassian application you are developing for. Examples: mvn confluence:create or mvn jira:create |
| APPLICATION:run | Download the application binaries (if not already downloaded), install your plugin, start the application with the container (application server) of your choice and your deployed plugin. Examples: mvn confluence:run or mvn jira:run |
| APPLICATION:debug | Start the application with the container of your choice and your deployed plugin, with remote debugging enabled. |
| APPLICATION:integration-test | Start the application with the container of your choice, and run all <code>it/**</code> integration tests, then shut the application down. |
| APPLICATION:unit-test | Run the unit tests, excluding <code>it/**</code> . |
| APPLICATION:cli | Run the command line interface, listening to a port by default for remote commands |
| APPLICATION:create-home-zip | Creates a zip of the current application's data which can be used in future runs of the SDK. |
| APPLICATION:copy-bundled-dependencies | Copy all runtime and compile-scoped dependencies into <code>META-INF/lib</code> . |
| APPLICATION:install | Install your plugin (via the PDK). |
| APPLICATION:uninstall | Uninstall your plugin (via the PDK). |

Examples

Creating a New Confluence Plugin from Scratch

Say you want to create a new Confluence plugin skeleton. Simply go to the directory where you want to create the plugin and type:

```
mvn confluence:create
```

Running a Plugin in an Application

Say you wanted to run an arbitrary plugin in Confluence. Simply go to the plugin's project directory and type:

```
mvn confluence:run
```

Specifying a Version of the Application

Say you wanted to run your plugin with Confluence 3.4:

```
mvn confluence:run -Dproduct.version=3.4
```

Say you want to run your plugin with JIRA 4.2:

```
mvn jira:run -Dproduct.version=4.2
```

Specifying an Application Server (Container)

Say you wanted to run that plugin but with Confluence 3.4 and Tomcat 7:

```
mvn confluence:run -Dproduct.version=3.4 -Dcontainer=tomcat7x
```

Running your Plugin in Debug Mode from your IDE

Say you wanted to run that plugin in debug mode from your IDE. Set up a configuration in IDEA that executes from the project directory:

```
mvn confluence:run -Dcontainer=jetty6x
```

Writing Integration Tests

Say you wanted to write a new plugin for Confluence, with integration tests, but wanted minimal POM XML and you do not want to have to extend any other POM:

```
<packaging>confluence-plugin</packaging>
...
<plugin>
  <groupId>com.atlassian.maven.plugins</groupId>
  <artifactId>maven-confluence-plugin</artifactId>
  <extensions>true</extensions>
</plugin>
```

Running Integration Tests against a Different Application

Say you had a Refapp plugin but wanted to run your integration tests against Confluence:

```
mvn package confluence:integration-test
```

Running your Plugin in Multiple Applications

Say you want to run the plugin in multiple applications simultaneously. In three separate tabs in your terminal (command window), type:

```
mvn refapp:run
```

```
mvn confluence:run -Dproduct.version=3.4
```

```
mvn jira:run -Dproduct.version=4.2
```

You can now test your plugin on three applications side by side in your browser. This also means you can upgrade your plugin simultaneously:

```
mvn package confluence:install refapp:install
```

Using the Command Line Interface

AMPS bundles the [Maven CLI plugin](#) and pre-configures it properly with the `pi` command. To use it with Confluence:

```
mvn confluence:cli
```

Specifying a Version of AMPS

The short form of the `mvn APPLICATION:run` or `mvn APPLICATION:create` command will use the latest version of the AMPS plugins. If you want to get a specific version of an AMPS command, use the longer form of the command E.g:
`mvn com.atlassian.maven.plugins:maven-confluence-plugin:3.2.4:create`

Advanced Configuration

The plugin's behavior can be customized with parameters. Parameters can be passed to the mojo at runtime in two ways:

- On the command line as Java system variables:

```
mvn jira:run -D<parameter1Name>=<parameter1Value> -D<parameter2Name>=<parameter2Value> ...
```

- In the POM:

```

<project>
  ...
  <build>
    <plugins>
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-jira-plugin</artifactId>
        <version>3.2.3</version>
        <extensions>true</extensions>
        <configuration>
          <!-- parameter key/value pairs go here -->
          <parameter1Name>parameter1Value</parameter1Name>
        </configuration>
      </plugin>
    </plugins>
  </build>
  ...
</project>

```

Values specified on the command line will override values in the POM.

containerId

Specifies the [Cargo container](#) that should be used when starting and hosting the product. The default value is `tomcat6x` (for Tomcat 6.0.20).



While the underlying Cargo system can support many different containers, Atlassian only supports Tomcat installations. Check the supported platforms page for [JIRA](#) and [Confluence](#) to see which versions are supported.

httpPort

Specifies the port that the container should listen on. The default value varies by product; see the table [here](#).

cliPort

Specifies the port that the embedded Maven CLI plugin should open for interactive shell commands. The default value is `4330`.

contextPath

Specifies the context path at which the product should be deployed. The default value varies by product; see the table [here](#).

server

Specifies the hostname of the machine on which the container should be started. The default value is `localhost`.

productVersion

Specifies the version of the product that should be run. This value must correspond to the version number of a set of Maven artifacts available from the [Atlassian Maven repositories](#). The default value is defined for each product in the `com.atlassian.amps:atlassian-amps-parent` POM version that corresponds to the current SDK version.

jvmArgs

Specifies arguments to the forked Java VM that will host the container. The default value is blank.

systemPropertyVariables

Specifies system properties that should be set in the forked Java VM that will host the container. The default values are blank.

Example:

```

<configuration>
  <systemPropertyVariables>
    <property1Name>property1Value</property1Name>
    <property2Name>property2Value</property2Name>
  </systemPropertyVariables>
</configuration>

```

log4jProperties

Specifies a custom log4j.properties file to be used by the product, replacing the one shipped inside the product. Generally, there is no reason to do this, but it can be useful for debugging sometimes.

productDataVersion

Specifies the version of the test data resources the product will use at startup. The test data includes a developer license valid for three hours, a default product home directory, and a simple, pre-provisioned, in-memory or on-disk data store (usually provided via HSQL). These files are supplied by Atlassian with the product. The default value is defined for each product in the com.atlassian.amps:atlassian-amps-parent POM version that corresponds to the current SDK version.

If you'd like to override the default data with data from your own product installation, see the `productDataPath` parameter.

productDataPath

Specifies a path on the local filesystem to a custom product data archive. This allows you to start the product prepopulated with data of your choice. For more details, see the [atlas-create-home-zip](#) SDK command.

If you need to also make changes to the `webapp` directory, see [overriding the application's webapp when developing your plugin](#).

pluginArtifacts

Specifies other Atlassian plugins that your plugin depends on to work properly. These plugin artifacts will be installed into the product along with your plugin.

All plugins specified here must be installed in a repository your Maven installation can access through `settings.xml`.

Example:

```
<configuration>
  <pluginArtifacts>
    <pluginArtifact>
      <groupId>com.mycompany.atlassian.plugins</groupId>
      <artifactId>our-business-services-plugin</artifactId>
      <version>1.0.3</version>
    </pluginArtifact>
  </pluginArtifacts>
</configuration>
```

libArtifacts

Specifies arbitrary Java libraries that your plugin depends on to work properly. These libraries will be installed alongside the product's core libraries in `WEB-INF/lib`.

All libraries specified here must be installed in a repository your Maven installation can access through `settings.xml`.

Accessing these libraries from your plugin is a product-specific process, and usually you won't want to do this anyway, as this defeats the purpose of the plugin system. This feature is only useful for legacy code or very old plugins that you're not able to upgrade to version 2 of the plugin system.

To depend on a third-party library, add it to the `<dependencies>` in your plugin POM, which protects you from changes out of your control.

Example:

```
<configuration>
  <libArtifacts>
    <libArtifact>
      <groupId>com.mycompany.ancient.plugins</groupId>
      <artifactId>our-business-services-plugin</artifactId>
      <version>1.0.3</version>
    </libArtifact>
  </libArtifacts>
</configuration>
```

bundledArtifacts

Specifies other Atlassian plugins that your plugin depends on to work properly. These plugin artifacts will be treated as [bundled](#) by the product, meaning they can be disabled but not removed. The product will always make a bundled plugin available at startup, even if it was removed previously. By contrast, plugin artifacts can be removed after installation at any time, and they will stay gone until they are manually reinstalled.

All plugins specified here must be installed in a repository your Maven installation can access through `settings.xml`.

Unless you're working with a forked version of a plugin normally bundled in the product, you will not need to use this.

To depend on another second- or third-party plugin, add it to the `<pluginArtifacts>` above.

Example:

```
<configuration>
  <bundledArtifacts>
    <bundledArtifact>
      <groupId>com.mycompany.atlassian.plugins</groupId>
      <artifactId>our-forked-bundled-product-plugin</artifactId>
      <version>4.1-mycompany-fork</version>
    </bundledArtifact>
  </bundledArtifacts>
</configuration>
```

pluginDependencies

Specifies other plugins or OSGi bundles that will be bundled into an OBR archive. OBRs can be installed via the UPM, and the UPM will automatically install any dependencies that are packaged in the OBR which are not installed yet.

All dependencies specified here must be defined in the standard `<dependencies>` section of the `pom.xml`, and should be set to a `<scope>` of either 'test' or 'provided'.

Example:

```
<configuration>
  <pluginDependencies>
    <pluginDependency>
      <groupId>com.mycompany.atlassian.plugins</groupId>
      <artifactId>our-osgi-bundle</artifactId>
    </pluginDependency>
  </pluginDependencies>
</configuration>
```

installPlugin

Specifies whether to install the current plugin when the product is started. The default value is (obviously) `true`.

You can set this flag to `false` to start a product without being in a plugin working directory. This is useful for quickly starting a product with a specific configuration for testing or experimentation. For example, the command:

```
atlas-mvn confluence:run -DproductVersion=3.4 -DinstallPlugin=false
```

will simply start Confluence 3.4 with the default test data.

output

Specifies a file to pipe the container's output to. The container output includes the startup logs, messages sent to `System.out` and `System.err` while the product is running, and SDK-specific messages. By default, container output is sent to the standard output stream.

This is useful for comparing your plugin's log output (and possible error messages) between different products or different versions of the same product. It's also useful when [reporting bugs with the SDK](#).

jvmDebugPort

Specifies the port on which the product should listen for a debugger connection. The default value is 5005.

jvmDebugSuspend

Specifies whether the product, when started in debug mode, should wait for a debugger to connect before proceeding with startup. The default value is `false`.

skipTests

Specifies whether testing should be skipped in this invocation. The default value is `false`.

functionalTestPattern

Specifies the pattern to use for finding integration tests to run. The default value is `\it/**/*Test.java` (relative to `src/test/java`). If any `<testGroup>` elements are specified, this value will be ignored.

products

Defines a product that can be referenced in a `testGroup`. Test groups allow you to run a subset of tests on a specific product configuration; for example, if your plugin is designed for JIRA and Confluence, you can have a test group with JIRA-specific tests, another group with Confluence-specific tests, and a third with tests common to both configurations.

See the [functional testing page](#) for more information on, and examples of, product definitions.

testGroups

Defines a test group to run on a `<product>`. Test groups can be run on any number of products.

See the [functional testing page](#) for more information on, and examples of, test groups.

configuredTestGroupsToRun

Specifies which test groups should be run in this invocation. Test groups should be referenced by ID and separated by commas. The default is for all test groups to run.

Example:

```
atlas-mvn crowd:integration-test -DconfiguredTestGroupsToRun=loginTests,authTests,cookieTests
```

See the [functional testing page](#) for more information on, and examples of, test groups.

noWebapp

Specifies whether the product(s) required by `<products>` should actually be started, or if the tests should run blind and assume that the needed products are already running. The default value is `false`, which starts all required `<product>` instances.

This can be valuable if you are debugging integration tests and the product's state is not relevant to the tests being debugged; this will save you the wait of a product starting and stopping before you can see the test results.

Example:

```
atlas-mvn jira:integration-test -DconfiguredTestGroupsToRun=jiraTests,pluginTests -DnoWebapp=true
```

instructions

Specifies OSGi bundling instructions, interpreted by the Maven [BND plugin](#) and used to generate the OSGi manifest inside the plugin JAR. BND instructions give you complete control over the expression of your plugin's runtime dependencies, including package name and version; they communicate to the plugin system (and the product) exactly what is required for your plugin to operate correctly.

This configuration is essential when creating plugins designed to operate in more than one product, but is usually unnecessary in other cases.

skipManifestValidation

Specifies whether the SDK should attempt to validate the OSGi manifest (whether explicitly specified by the user or auto-generated). The default value is `false`.

This can be useful when debugging an OSGi validation problem or forcing the plugin to load in the container to see which bundles can not be located. However, in most cases the warnings suppressed by this flag indicate actual problems that need to be fixed before the plugin can be successfully installed.

RELATED TOPICS

[Developing your Plugin using the Atlassian Plugin SDK](#)
[Functional Testing with Multiple Products](#)

Getting Help with the Atlassian Plugin SDK



Help is at hand

Make sure you are not in the Maven command line interface (CLI) when you enter the help commands described below. If you are in the CLI, your command line will start with `maven2>`, and you will need to exit from the CLI first. Enter `quit`, `exit` or a friendly `bye`.

Getting an Overview of All the Scripts

Enter the following shell script to see an overview of all the scripts with a brief outline of their functionality:

Enter the following to see all possible help content:

Getting Help Text per Script

Each shell script provides help text if the first argument of the script is one of the following:

- `-?`
- `-h`
- `help`
- `-help`
- `--help`

Examples:

Reading the Reference Guide

See the detailed [guide to all scripts](#).

RELATED TOPICS

Welcome to the Atlassian Developer Network

- How to learn to program (in Java or JavaScript)
- Installing the Atlassian Plugin SDK
- Writing your first plugin
- Advanced Plugin Development
- Atlassian Plugin SDK Documentation
- Other Information
- FAQ
- Tutorials
- Getting Involved in the Atlassian Developer Network
- Codegeist V
- Developer Relations State of the Union 2011 - Resources

Converting your Plugin to the new Atlassian Plugin SDK



Current released version – Atlassian Plugin SDK 3.4

Currently-supported applications: **Confluence**, **JIRA**, **Bamboo**, **FishEye**, **Crucible** and **Crowd**.

Atlassian Plugin SDK 3.4 is now available – see the [release notes](#). If you have ideas for improvement or new features, or have found a bug, please raise a ticket on our [issue tracker](#). Snapshot builds are also available for the stout-hearted.

This page tells you how to convert an existing plugin to use the new Atlassian Plugin SDK. If you want to develop a *new* plugin using the SDK, you can follow the step-by-step instructions on [developing a new plugin](#).

On this page:

- Requirements
- Converting your Plugin
 - Step 1. Remove Outdated Maven Parent POM
 - Step 2. Update POM `<packaging>`
 - Step 3. Add AMPS Plugin and Product Dependencies to POM `<build>`
 - Step 4. Configure the Compiler Plugin
 - Step 5. Add distribution management information
 - Step 6. Test the Converted Plugin
- Sample Conversion Process
 - Getting the Source
 - Making Sure the Plugin Works as It Is

- Making the Changes for the Atlassian Plugin SDK
- Trying the SDK Commands

Requirements

You will need the following:

- Your existing plugin should be 'Mavenised' – that is, organised according to Maven's Standard Directory Layout™ and having a correct `pom.xml`.
- The plugin should be for one of the Atlassian applications supported by the SDK.

Converting your Plugin

Step 1. Remove Outdated Maven Parent POM

Your plugin probably inherits some plugin configuration from an Atlassian plugin POM. Previously, this Maven inheritance mechanism was used to supply your plugin with plugin-specific goals and configuration for them. This is no longer necessary, so if the outdated parent POM exists, it has to go.

Look for a `<parent>` XML element near the top of `pom.xml`. Depending on your plugin's host applications (say, Confluence) this will look something like:

```
<parent>
  <groupId>com.atlassian.confluence.plugin.base</groupId>
  <artifactId>confluence-plugin-base</artifactId>
  <version>12</version>
</parent>
```

JIRA, Bamboo, Crucible, and Crowd plugins may have similar configuration. Remove this entire element.

Step 2. Update POM `<packaging>`

Find the `<packaging>` element in your `pom.xml`. If it is not there, add it. The usual place for it is directly below `<name>`. Make sure it has the following value:

```
<packaging>atlassian-plugin</packaging>
```

Step 3. Add AMPS Plugin and Product Dependencies to POM `<build>`

The Atlassian Maven Plugin Suite (**AMPS**) forms part of the Atlassian Plugin SDK. You will need to add AMPS to your POM.

Find the `<build>` element in your POM, and then find underneath it the `<plugins>` element. If it is not there, add it.

```
<build>
  <plugins>
    ...
  </plugins>
</build>
```

Add a plugin configuration for the AMPS plugin. For Confluence, the configuration looks like this:

```
<build>
  <plugins>
    <plugin>
      <groupId>com.atlassian.maven.plugins</groupId>
      <artifactId>maven-confluence-plugin</artifactId>
      <version>3.1.3</version> <!-- use the latest version of the SDK -->
      <extensions>true</extensions>
      <configuration>
        <productVersion>${atlassian.product.version}</productVersion>
        <productDataVersion>${atlassian.product.data.version}</productDataVersion>
      </configuration>
    </plugin>
  </plugins>
</build>
```

There are a few important things to notice here:

1. The line `<extensions>true</extensions>`. This puts the unified AMPS goals into your plugin namespace. Without this, you will not be able to use the convenient commands like `atlas-unit-test` or `atlas-debug`.
2. The `<productVersion>` configuration property. This is used to specify which version of Confluence to use for dependency resolution and the integration test container. It is not required, but having it makes it clear which version you are using, and it is used as a sensible default in the AMPS goals.
3. The `<productDataVersion>` configuration property. This is used to specify the version of (test) pre-packaged Confluence home directory to use when launching Confluence in the integration-test phase. While the version number of this packaged home directory follows Confluence's (e.g. version 3.0 of the package is created with Confluence 3.0), that does not mean for every version of Confluence, there's the same version of the package. For instance, there *may* be no version 3.1 of the home directory package - that's because version 3.0 can be used with Confluence 3.1. When in doubt, please look [here](#)

If you do not have an `atlassian.product.version` property defined, you can add it to the `<properties>` element:

```

<properties>
    <atlassian.product.version>3.0</atlassian.product.version> <!-- or whichever
    SDK-supported product version you're using -->
    <atlassian.product.data.version>3.0</atlassian.product.data.version> <!-- This
    defines the version of Confluence home directory to use to launch Confluence in the
    integration-test phase. Try to use a recent version (e.g. 3.0 for Confluence > 3.1) -->
</properties>

```

You will also need to add the product libraries to your POM; previously, these were automatically inherited from the parent POMs.

JIRA configuration

```

<dependency>
    <groupId>com.atlassian.jira</groupId>
    <artifactId>atlassian-jira</artifactId>
    <version>${atlassian.product.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.jira</groupId>
    <artifactId>jira-func-tests</artifactId>
    <version>${atlassian.product.version}</version>
    <scope>test</scope>
</dependency>

```

Confluence configuration

```

<dependency>
    <groupId>com.atlassian.confluence</groupId>
    <artifactId>confluence</artifactId>
    <version>${atlassian.product.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.confluence.plugin</groupId>
    <artifactId>func-test-package</artifactId>
    <version>2.2-beta5</version> <!-- Version 2.2 of the test library contains fixes that make
    it to work with SDK more seamlessly. Please also note the change of artifact ID (from func-test to
    func-test-package) -->
    <scope>test</scope>
</dependency>

```

Bamboo configuration

```

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.bamboo</groupId>
    <artifactId>atlassian-bamboo-api</artifactId>
    <version>${atlassian.product.version}</version>
    <scope>provided</scope>
</dependency>

```

Step 4. Configure the Compiler Plugin

The old parent POM would use the `jdkLevel` property to set the `source` and `target` parameters used for compiling your plugin. Since there is no parent POM any more, you need to add this in

```
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <source>1.5</source>
    <target>1.5</target>
  </configuration>
</plugin>
```

Step 5. Add distribution management information

If you need to deploy your plugin's artifacts to Maven, please make sure your POM has the right distribution management information. For most Open Source plugins, the distribution management looks like this:

```
<distributionManagement>
  <repository>
    <id>atlassian-contrib</id>
    <name>Atlassian Contrib Repository</name>
    <url>davs://maven.atlassian.com/contrib</url>
  </repository>
  <snapshotRepository>
    <id>atlassian-contrib-snapshot</id>
    <name>Atlassian Contrib Snapshot Repository</name>
    <url>davs://maven.atlassian.com/contrib-snapshot</url>
  </snapshotRepository>
</distributionManagement>
```

Step 6. Test the Converted Plugin

After making the above changes, your plugin should be ready to use the Atlassian Plugin SDK. Test it out by starting your plugin's host application:

`atlas-run`

Point your browser to the URL as prompted (e.g. <http://localhost:1990/confluence>) and confirm that the server is active.

If something goes wrong, check that your environment is correct. Refer to the instructions on setting up your environment and developing a plugin.

Sample Conversion Process

We will show you how this conversion would be done on an existing, fully operational Atlassian plugin. We will use the [Confluence Basic Macros](#) plugin, which provides such useful things as the `{quote}`, `{color}`, and `{panel}` macros.

Getting the Source

We will use the 1.4 version for this guide. You can get the source with Subversion:

```
svn co https://studio.plugins.atlassian.com/svn/BASICMACROS/tags/confluence-basic-macros-1.4/
```

Making Sure the Plugin Works as It Is

Build the entire plugin with Maven:

```
cd confluence-basic-macros-1.4
mvn install
```

This may take several minutes to download the required dependencies, but it should build successfully.

Making the Changes for the Atlassian Plugin SDK

1. Open the `pom.xml` and look for the `<parent>` element:

```
<parent>
    <groupId>com.atlassian.confluence.plugin.base</groupId>
    <artifactId>confluence-plugin-base</artifactId>
    <version>12</version>
</parent>
```

Remove the element entirely.

- Now find the `<packaging>` element. Confirm that its value is:

```
<packaging>atlassian-plugin</packaging>
```

- Since this POM has no existing `<build>` element, we will add one:

```
<build>
    <plugins>
        <plugin>
            <groupId>com.atlassian.maven.plugins</groupId>
            <artifactId>maven-confluence-plugin</artifactId>
            <version>3.1.3</version>
            <extensions>true</extensions>
            <configuration>
                <productVersion>${atlassian.product.version}</productVersion>
            </configuration>
        </plugin>
    </plugins>
</build>
```



Reusing existing properties

Since there is already a property `atlassian.product.version`, we will reuse it instead of adding another one.

Trying the SDK Commands

Start Confluence:

```
atlas-run
```

Point your browser to <http://localhost:1990/confluence> and confirm that the server is active. The conversion is now complete.



Dynamically installable plugins

If you want to be able to reinstall/redploy your plugin without restarting the application when using the CLI interface via the [Atlassian Plugin SDK], your plugin needs to be a **version 2** plugin. At the very least, it needs the `plugins-version="2"` attribute in `atlassian-plugin.xml`. See [converting a version 1 plugin to a version 2 plugin](#).

RELATED TOPICS

[Getting Help with the Atlassian Plugin SDK](#)

[Benefits of Using the SDK](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

[Atlassian Plugin SDK Documentation](#)

[AMPS Plugin for Maven](#)

[Installing the Atlassian Plugin SDK](#)

Plugin SDK Snapshots

Feeling brave?

Want to live on the bleeding edge?

We upload every successful SDK build to our public Maven repository. If you'd like to try a new fix or feature without waiting for the next release, now you can.

A few words of caution are in order first:

- These are **unsupported** builds. They may work flawlessly. They may have crippling, data-destroying bugs. They may cause demons to fly out of your nose. If that does happen, send us a picture, not a support request.
- Don't use this on production code or data. Use a separate source checkout and test data.
- That being said, if you have problems, **please** let us know by filing bugs in the [SDK JIRA project](#).

Finally, if you're motivated enough to try a snapshot build, we think it's likely you can write some code yourself. If you do find a problem – and the fix is something you can do yourself – send us the fix along with your newly filed JIRA issue. The SDK is open source protected by the [Apache 2.0 license](#), and we're delighted to accept your contributions under those terms.

SDK Feedback Form

Validation with XML schemas



This page is currently atlassian-staff only until the underlying plugin system bug is fixed.

We're happy to present a new way to improve the readability and maintainability of your plugins: XML Schema validation. XML schemas are a W3C-recommended XML dialect which can be used to verify that the structure – not just the syntax – of a given XML document is valid. The Maven POMs in your existing plugins are XML files constrained by a schema, and now that same benefit is being brought to `atlassian-plugin.xml`.

Features

Schemas bring several benefits, but there are three essential ones: validation, autocompletion and documentation.



Currently supported products

JIRA 4.3+, Confluence 3.5+, Refapp 2.9+.

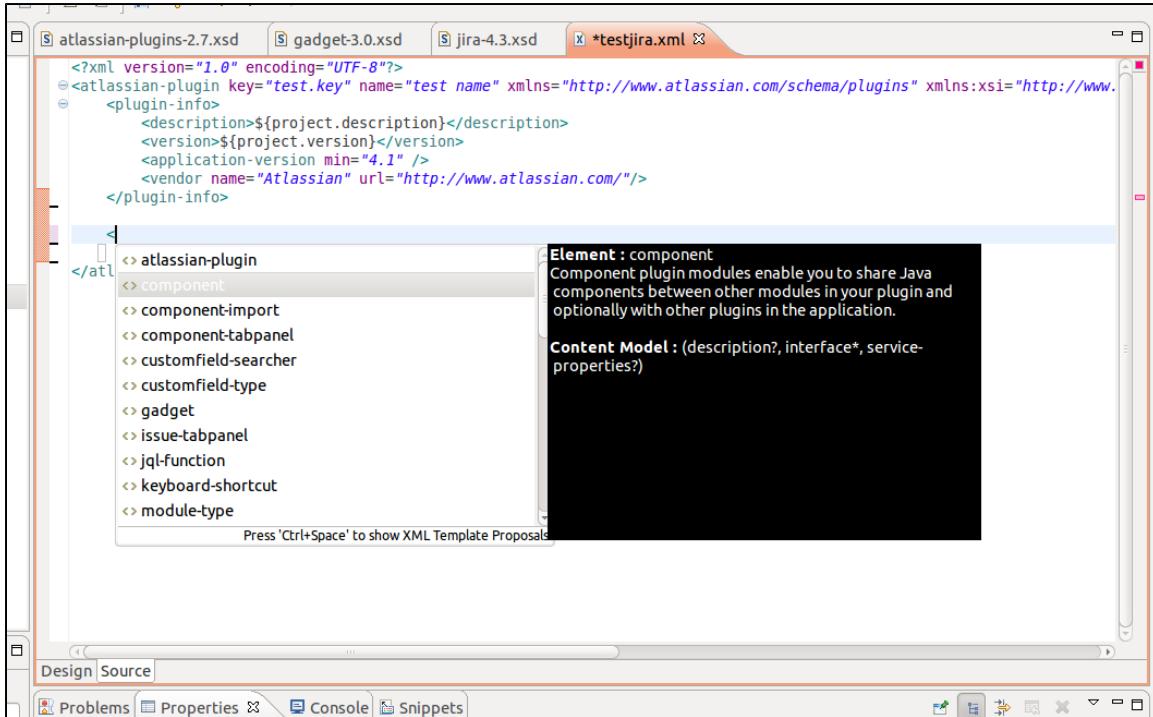
Validation

Plugin modules have different data schematics. Some require values for every configurable element. Many plugins require some values but not all, and some plugins don't require any configuration. The schemas are aware of this and will check that you have provided required information.

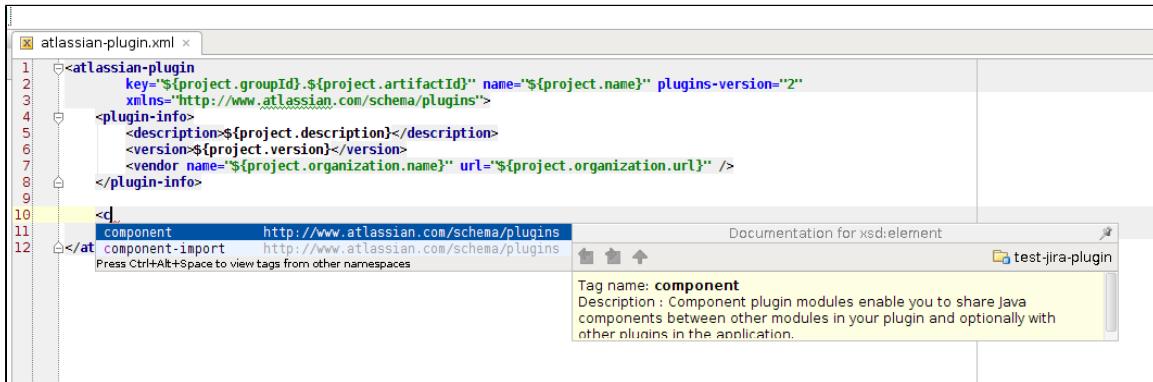
Autocompletion/Documentation

Many IDEs are smart enough to use their code suggestion features with schemas. XML-specific editors like [OxygenXML](#) do this, as well as the major Java IDEs. In addition, the schemas contain embedded documentation. The documentation explains which plugin module corresponds to the XML tag, what its acceptable parameters are, what's required and what's not, and so on. The documentation text comes from the plugin module pages on the development hubs for each supported product (see Confluence, JIRA, and the plugin framework).

Here's how the autocomplete looks in [Eclipse](#) with embedded documentation:



And here's how it looks in IntelliJ IDEA:



Documentation window in IDEA

You may need to press **Ctrl-Q** to get the documentation popup window to display.

Schemas in new plugins

For currently supported products, all plugins created with `atlas-create-<product>-plugin` as of SDK version 3.3.1 will automatically include the correct schema declarations. No other work on your part is necessary.

Schemas in existing plugins

To add schemas to your existing plugins, you need to add the validation syntax yourself.

- Open your `atlassian-plugin.xml` file in your editor
- Locate the root element, which should look like:

```
<atlassian-plugin key="your.plugin.key" name="Plugin Name" plugins-version="2">
```

- Add the following attributes inside the root element:

```

<atlassian-plugin key="your.plugin.key" name="Plugin Name" plugins-version="2"
    xmlns="http://www.atlassian.com/schema/plugins"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.atlassian.com/schema/plugins
    http://schema.atlassian.com/productName/productName-productVersion.xsd">

```

where `productName` and `productVersion` correspond to the Atlassian product you're using. Only major revisions are necessary, so JIRA 4.3.2 would use only 4.3.

| JIRA | Confluence | Refapp |
|----------|----------------|------------|
| jira-4.3 | confluence-3.5 | refapp-2.9 |

Known issues

Schemas enforce an order on elements that doesn't always correspond to `atlassian-plugin.xml`. As far as the product is concerned, any element and attribute ordering is allowable. For example, all modules take a `<description>` element that provides a human-readable description of the module's purpose. The schemas require that this element come first, but the plugin will still work properly regardless. Feel free to ignore validation warnings about out-of-order elements if you wish.

Reporting problems

We've made every reasonable effort to ensure that the schemas work properly. If you do find something you can't explain and that should work properly, please email developer-relations@atlassian.com.

Other Information

- Documentation
- How to develop without the Atlassian Plugin SDK
- Developer Bamboo Server — This page points to continuous integration builds for Atlassian supported plugins and libraries.
- Getting Started with Remote APIs
- Confluence Build Information — exact versions, dates and build numbers of Confluence releases.

Documentation

If you're want to build a plugin to extend or enhance your Atlassian tool, you've come to the right place. You'll find below a collection of documentation links that will help you in your task:

Building an Atlassian Plugin

- How to build an Atlassian Plugin — Start here to learn how to set up your development environment, create a basic plugin and start coding.
- Plugin Developer Best Practices
- Atlassian Plugin Development Platform

Atlassian Technologies you might run into:

- Shared Access Layer
- Atlassian User Interface Library (AUI/AJS)
- Atlassian Plugin Framework
- Gadgets
- Atlassian REST API

Product Specific Guides

JIRA Development

- JIRA Development Hub
- Developer FAQ
- JIRA JavaDoc

Confluence Development

- Confluence Development Hub
- Developer FAQ
- Confluence JavaDoc

Bamboo Development

- Bamboo Development Hub
- Developer FAQ
- Bamboo JavaDoc

Crowd Development

- Crowd Development Hub
- Developer FAQ
- Crowd JavaDoc

Fisheye/Crucible Development

- Fisheye/Crucible Development Hub
- Frequently Asked Questions
- Fisheye/Crucible JavaDoc

How to develop without the Atlassian Plugin SDK



These instructions are obsolete if you are using the [Atlassian Plugin SDK](#) (and you should be). Leaving here only as reference for those using very old products which do not support the SDK.

- Example pom.xml — This is the POM that will be created if you run the Confluence Archetype.
- Example settings.xml
- Obsolete Atlassian Maven PDK — This is a Maven 2 plugin which assists in building and deploying Atlassian plugins.
- Example atlassian-plugin.xml — The Plugin descriptor is an XML file that tells the application all about the plugin
- Maven Requirements
- Atlassian Plugin Archetypes — Atlassian has created Maven 2 archetypes for five of our pluggable products.

Example pom.xml



These instructions are obsolete if you are using the [Atlassian Plugin SDK](#) (and you should be). Leaving here only as reference for those using very old products which do not support the SDK.

This is the POM that will be created if you run the Confluence Archetype. (The JIRA one is quite similar.) We've included it here so you can better understand what the POM does for you. There are some notes at the bottom about each section.

- JIRA - [see source code](#)
- Confluence - [see source code](#)
- Bamboo - *forthcoming*
- Crowd - *forthcoming*

Confluence Example POM

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">

    <parent>
        <groupId>com.atlassian.confluence.plugin.base</groupId>
        <artifactId>confluence-plugin-base</artifactId>
        <version>17</version>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <groupId>com.this.that.groupId</groupId>
    <artifactId>artifactId</artifactId>
    <version>0.1</version>

    <name><!-- TODO: Add a name for your plugin --></name>
    <packaging>atlassian-plugin</packaging>

    <properties>
```

```

<atlassian.plugin.key>${pom.groupId}.${pom.artifactId}</atlassian.plugin.key>

<!-- Confluence version -->
<atlassian.product.version>2.9</atlassian.product.version>
<!-- Confluence plugin functional test library version -->
<atlassian.product.test-lib.version>1.4.1</atlassian.product.test-lib.version>
<!-- Confluence data version -->
<atlassian.product.data.version>2.9</atlassian.product.data.version>
</properties>

<!-- TODO: Add project description -->
<!--
<description></description>
<url></url>
-->

<!-- TODO: Complete developer details -->
<!--
<developers>
<developer>
<name>John Smith</name>
<organization>Example Company</organization>
</developer>
</developers>
-->

<!-- TODO: Complete source control details -->
<!--
<scm>
<connection></connection>
<developerConnection></developerConnection>
<url></url>
</scm>
-->

<!--You can define Ant tasks to be executed before and/or after integration tests --><!--
<build>
    <plugins>
        <plugin>
            <artifactId>maven-antrun-plugin</artifactId>
            <executions>
                <execution>
                    <id>pre-integration-test-user-ant-tasks</id>
                    <configuration>
                        <tasks>
                            <echo message="Ant task before integration tests."/>
                        </tasks>
                    </configuration>
                </execution>
                <execution>
                    <id>post-integration-test-user-ant-tasks</id>
                    <configuration>
                        <tasks>
                            <echo message="Ant task _after_ integration tests."/>
                        </tasks>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>

```

```
</build>-->  
</project>
```

Parent

POMs can inherit from other POMs. The Confluence Archetype POMs inherit many settings from the [Confluence Plugins Base POM](#). You can take a look at this POM to see all the neat things it does for you, but ideally, you'll never need to worry about it.

Group and Artifact

These two pieces of information identify your plugin. They will make up your plugin key, which is used inside JIRA and Confluence to reference your plugin, as well as in the [Atlassian Plugin Repository](#).

Details

This section gives you a chance to tell us about the plugin and about its authors. Add a description and credits. The SCM section should be modified to point to your plugin's location in the [Developer Subversion Repository](#). This SVN location will be used to automatically tag and release new versions of your plugin.

Example settings.xml



Try the new Plugin SDK

We've released a new Plugin SDK which handles almost all of this Maven tweaking for you. The SDK includes an embedded Maven installation and correct `settings.xml` that will be kept up to date as necessary. We believe it makes the plugin development process much easier. If you are using the SDK, you won't need this file. For more information, see [\[here\]](#).

This is an example `settings.xml` file for Maven 2. It can be placed in your `$HOME/.m2` directory and it will apply to all maven projects that you build. If you would rather, you can make this a per-project settings by including a `profile.xml` in your project base directory. See [Maven's documentation on Build Profile Settings](#).

If your network requires the use of an HTTP or HTTPS proxy, you'll need to add those to your `settings.xml`.

[Download here](#)

Obsolete Atlassian Maven PDK



These instructions are obsolete if you are using the [Atlassian Plugin SDK](#) (and you should be). Leaving here only as reference for those using very old products which do not support the SDK.

| | |
|-----------------|---|
| Name | Atlassian Plugin Developer Kit for Maven2 |
| Vendor | |
| Author(s) | Dan Hardiker, Michael Mekaail, Tony Truong, David Peterson, Kevin Ross |
| Homepage | Obsolete Atlassian Maven PDK |
| Issue Tracking | http://developer.atlassian.com/jira/browse/CPDK |
| Categories | |
| Version | 2.1.6 |
| Availability | All |
| State | Stable |
| License | Apache 2 |
| Price | Free |
| Java API Docs | |
| Download Source | Subversion |
| Download JAR | atlassian-pdk-2.1.6.jar |

Description/Features

This is a Maven 2 plugin which assists in building and deploying Atlassian plugins.

Installation

To start, you must be using Maven 2.0.9 or later.

Maven automatically downloads plugins that are configured in your pom and installs them into its repository when you use them. If you use Atlassian's [plugin archetypes](#) to create your plugin, you should not have to do anything else to install the Atlassian PDK.



Manual Installation

Sometimes automatic install doesn't work. If so, try this:

- Download the latest version of the PDK, listed above.
- Install it to your maven repository by running this command from the directory where the downloaded JAR resides:
`mvn install:install-file -DgroupId=com.atlassian.maven.plugins
-DartifactId=atlassian-pdk -Dpackaging=jar -Dfile=atlassian-pdk-2.1.6.jar
-Dversion=2.1.6`
- Create an issue in the [Developer Network CPDK JIRA project](#) to let us know the details of the problem you encountered so we can fix it or improve these instructions.

Configuration

There are several properties which should be specified in your local `settings.xml` file which are used by this plugin. These are all noted in the [Example settings.xml](#) file.

| Property Name | Default | Description |
|--|-------------------|---|
| <code>atlassian.pdk.server.url</code> | <code>None</code> | The URL for the base address of the Confluence server the plugin should be installed on e.g., ' <code>http://localhost:8080</code> '. |
| <code>atlassian.pdk.server.username</code> | <code>None</code> | The username to log in as when installing. This user must have administration privillages on the server. |
| <code>atlassian.pdk.server.password</code> | <code>None</code> | The password to log in with when installing. |



Speeding Up Confluence Plugin Development

Looking for faster turnaround times in Confluence plugin development? Try the guide to [Speeding Up Confluence Plugin Development Using Maven CLI](#).

Usage

`mvn package`

| Command | Description |
|--------------------------|--|
| <code>mvn package</code> | Creates a Jar in the target directory with any dependencies your plugin depends on. |

Variables

`extractDependencies`

Type: boolean
Default: false

Atlassian plugins often needs other libraries as dependencies. In Confluence, you can include dependent JARs *inside* your plugin, and the default `atlassian-plugin` package is configured to do this. Any dependency that you specify in your POM will be included inside the `META-INF/lib` of your plugin jar.

However, JIRA plugins cannot yet do the same thing, so it is necessary to explode your dependent JARs, and include their classes and resources inside your plugin. By specifying `extractDependencies=true` the Atlassian PDK can do this for you.

Note: Any dependency that has a `<scope>compile</scope>` or `<scope>runtime</scope>` will be packaged in the final JAR. All other dependencies will be ignored. To exclude particular dependencies, use the provided scope, which indicates that the environment is expected to provide the library at runtime.

`installMethod`

Type: text
Options: upload or copy **Default:** upload

Confluence allows plugins to be uploaded directly into a running instance of the server. JIRA, Bamboo and the other Atlassian products do not yet allow this. If building a plugin for those servers you will need to specify that the plugin should be installed via copy.

If you do, you also need to specify the `pluginsHome` option to tell the PDK where to copy to.

`pluginsHome`

Type: text
Default: N/A

This defines the location that the plugin will be copied to. Typically, this is something like this:

```
<pluginsHome>${jira.webapp.path}/WEB-INF/lib</pluginsHome>
```

The `${jira.webapp.path}` value is set in your `settings.xml` file and points to the absolute path to your local JIRA server. Substitute 'bamboo', 'fisheye', etc for other application servers.

Plugin Goals

The following goals manage the plugin's installation in a specified Confluence server.

- `atlassian-pdk:disable`: Disables the current plugin in the server specified in `atlassian.pdk.server.url`.
- `atlassian-pdk:enable`: Enables the current plugin in the server specified in `atlassian.pdk.server.url`.
- `atlassian-pdk:install`: Packages the plugin, uninstalls any existing copies, and installs the new version to the server specified in `atlassian.pdk.server.url`.
- `atlassian-pdk:uninstall`: Uninstalls the current plugin from the server specified in `atlassian.pdk.server.url`, if it exists.
- `atlassian-pdk:rescan`: Asks the server specified in `atlassian.pdk.server.url` to rescan the `plugins` directory for new plugins. (*Note: this has basically no effect since Confluence 2.5, due to plugins being stored in the database from that version onwards*)

Version History

| | |
|-----------------------|-------------------------------|
| 2.1.6 | Release Notes |
| 2.1.5 | Release Notes |
| 2.1.2 | Release Notes |

Speeding Up Confluence Plugin Development Using Maven CLI



These instructions are obsolete if you are using the [Atlassian Plugin SDK] (and you should be). Leaving here only as reference for those using very old versions of Confluence.

As everyone knows, running [Maven](#) can be pretty slow, which means deploying your plugin using the [Atlassian Maven PDK](#) takes quite a while. So I tried to find a way for faster turnaround times in Confluence plugin development. By using a combination of [Don Brown's excellent CLI Plugin](#) and the [Atlassian Maven PDK](#), my deployment time for a changed plugin is down to — TADA — **1 to 2 seconds**.

Setting Up a New Plugin Project

Here are the instructions to set up a new plugin project from scratch:

1. Create a new Maven project using the [plugin archetypes](#). You need to replace `$MY_PACKAGE` with your package name (e.g. `com.example.confluence`) and `$MY_PLUGIN` with your artifact ID (e.g. `fast-confluence-plugin`). More detailed instructions are available [here](#).
2. Update your `~/.m2/settings.xml` (see [Example settings.xml](#)) to include the following plugin repository for the CLI plugin:

```
<pluginRepository>
  <id>cli</id>
  <url>http://twdata-m2-repository.googlecode.com/svn/</url>
  <snapshots>
    <enabled>false</enabled>
  </snapshots>
  <releases>
    <enabled>true</enabled>
  </releases>
</pluginRepository>
```

- This will allow the plugin to be installed automatically when it is used for the first time.
3. Make sure that you have specified the location of **your** Confluence instance in your `settings.xml` when configuring the PDK:

```
<atlassian.pdk.server.url>http://localhost:1990/confluence</atlassian.pdk.server.url>
```

4. Add the following to your project's `pom.xml` as a child of the `<project>` element:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.twdata.maven</groupId>
      <artifactId>maven-cli-plugin</artifactId>
      <configuration>
        <commands>
          <pi>clean resources compile jar
com.atlassian.maven.plugins:atlassian-pdk:install</pi>
          <pu>com.atlassian.maven.plugins:atlassian-pdk:uninstall</pu>
        </commands>
      </configuration>
    </plugin>
  </plugins>
</build>
```

This will define the two commands `pi` (plugin install) and `pu` (plugin uninstall). Of course you can adjust the commands to your needs or add more customs commands.

5. Now start up your Confluence development instance.
6. Run `mvn cli:execute` and type `pi` at the `maven2>` command prompt.

After the first run, which takes about 2 seconds, all subsequent runs take about 1 second on my machine. And because the CLI plugin supports command history, running it again is as easy as switching to your terminal, doing `<Arrow Up>+<Enter>`, switching to your browser and reloading your page.

The above procedure is very reliable and is as close to the usual plugin runtime environment as possible.

Example `atlassian-plugin.xml`



These instructions are obsolete if you are using the [Atlassian Plugin SDK](#) (and you should be). Leaving here only as reference for those using very old products which do not support the SDK.



Plugin Framework 2

Please refer to the documentation for the [Atlassian Plugin Framework 2](#), especially the [plugin](#) description and [plugin module types](#).

The `atlassian-plugin.xml` Plugin Descriptor

The Plugin descriptor is an XML file that tells the application all about the plugin, and the modules contained within it. The descriptor must be a single file named `atlassian-plugin.xml` and must be located at the root of the plugin's jar file. Here's a sample plugin descriptor:

```

<!-- Every plugin must have a key, which identifies the plugin uniquely to the system -->
<!-- and a name, which is used to display the plugin in menus. -->
<atlassian-plugin key="com.atlassian.confluence.plugins.example" name="Example Plugin"
plugins-version="2">

    <!-- The plugin info block allows you to provide more information about your plugin -->
    <plugin-info>
        <description>
            A sample plugin for demonstrating the file format.
        </description>
        <!-- This version is displayed in the application's Plugin Manager. -->
        <version>1.0</version>
        <!-- The versions of the application this plugin is compatible with -->
        <application-version min="1.3" max="1.3"/>
        <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com/" />
        <!-- The location of any plugin configuration (optional) -->
        <param name="configure.url">/admin/plugins/example/configurePlugin.action</param>
    </plugin-info>

    <!-- Here is where you define your modules. The code you use      -->
    <!-- to define a module depends on the module itself. This is just -->
    <!-- a sample, which will not load if installed into Confluence      -->

    <!-- Modules must have a key that is unique within the plugin, a name -->
    <!-- and an implementing class. -->
    <example key="module1" name="Example Module"
class="com.atlassian.confluence.plugins.example.ExampleModule">
        <!-- All modules can optionally have a description -->
        <description>An example module</description>
    </example>
</atlassian-plugin>

```

Each plugin has a **plugin key** which must be unique to the plugin. We suggest using the Java convention of reversing your domain name in order to ensure your key is unique. Each module has a **module key** which need only be unique within the plugin it is defined.



The plugin key has to be defined in lower-case in the plugin descriptor.

When you call the plugin in wiki markup you can use any capitalization (eg. {module1} or {Module1})

Use `plugins-version="2"` to create an OSGi plugin using version 2.0 or later of the [Atlassian Plugin Framework](#).

Sometimes you will need to uniquely identify a module - you do this with the **complete module key**. A module with key `fred` in a plugin keyed as `com.example.modules` will have a complete key of `com.example.modules:fred`

All plugin modules have a **class** attribute, which tells the plugin manager which Java class it should instantiate when loading the module. What class you should provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence (so you can write a theme-plugin without any Java code), but for macro and listener modules you need to write your own implementing class and include it in your plugin.

Java Classes

Because the plugin is a JAR that is dropped into the application's classpath, all Java classes contained within the JAR become a part of the application. You can include as many classes as you like, and have them interact with each other. Obviously, it's important to follow the Java package naming conventions to ensure your plugin's classes do not conflict with the application's classes, or other plugins'.

Plugin and Module Resources

Resources are non-Java files that a plugin may need in order to operate. Examples of possible resources might be:

- A velocity file used to generate HTML for a macro or layout plugin module
- A CSS file required by a theme layout plugin module
- An image referenced from within a layout plugin module
- A macro help file
- A localisation property file

Resource definitions look like this. They can be either a part of the plugin, or part of a particular plugin module:

```

<!-- A resource has a type, a name and a location. The resource definition maps -->
<!-- some arbitrary resource name to where that resource would be located in      -->
<!-- the server's classpath -->
<resource type="velocity" name="template" location="com/example/plugin/template.vm"/>

<!-- For the localisation property file below, it must be named exampleplugin.properties -->
<!-- located under the resources folder -->
<resource type="i18n" name="i18n" location="resources/exampleplugin" />

<!-- Resources may contain arbitrary key/value pairs -->
<resource type="download" name="style.css" location="com/example/plugin/style.css">
  <property key="content-type" value="text/css"/>
</resource>

```

The **name** of the resource defines how the plugin module can locate a particular resource. The **type** of a resource tells the module how that resource can be used. A module can look for resources of a certain type or name: for example the `layout` plugin required that its help file is a file of type `velocity` and name `help`.

The **location** of a resource tells the plugin where the resource can be found in the jar file (resources are loaded by Java's classpath resource-loader). The full path to the file - without a leading slash - is required.

The simplest kind of resource, supported with all plugin module types, is of type `download`, which makes a resource available for download from the application at a particular URL. See: [Downloadable Plugin Resources](#).

Modules can be **disabled by default** by specifying `state="disabled"`. Similarly, the entire plugin can be disabled by default with `<atlassian-plugin state="disabled"/>`

Plugin Configuration

Plugins can specify internal links within the application to configure themselves. This is useful where your plugin requires any configuration or user specific settings to work. For example, the [Google Maps plugin](#) requires a Google API Key from Google (which needs to be configured on each server) before it will work properly.

- Configuration links will *most often* point to XWork plugin modules within the plugin itself.
- Configuration links can be provided for a whole plugin and/or for any module within a plugin.
- Configuration links are relative to the application.

Plugin configuration - to add a configuration link for the whole plugin, place a single `param` element with the name `configure.url` within the `plugin-info` element at the top of the plugin descriptor:

```

<plugin-info>
  <description>A macro which displays Google maps within a Confluence page.</description>
  <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com/" />
  <version>0.1</version>
  <param name="configure.url">/admin/plugins/gmaps/configurePlugin.action</param>
</plugin-info>

```

Plugin module configuration - to add a configuration link for a single module, place the same `param` element with the name `configure.url` within the `descriptor` element for that module:

```

<macro name="gmap" class="com.atlassian.confluence.ext.gmaps.GmapsMacro" key="gmap">
  <description>The individual map macro.</description>
  <param name="configure.url">/admin/plugins/gmaps/configureMacro.action</param>
</macro>

```

Here is an image showing where the `Configure` links appear for both a plugin and an individual module:

| Plugins | |
|---|--|
| Advanced Macros All modules enabled. | Google Maps |
| Attachment Extractors All modules enabled. | Vendor: Atlassian Software Systems Pty Ltd Plugin Version: 0.1 |
| Basic Macros All modules enabled. | A macro which displays Google maps within a Confluence page. |
| Compatibility Macros All modules enabled. | <input type="checkbox"/> Configure plugin <input type="checkbox"/> Disable plugin |
| Dashboard Macros All modules enabled. | gmap The individual map macro. Configure GMaps gmapsManager |
| Google Maps All modules enabled. | Configure Disable Configure Disable Configure Disable |

Maven Requirements



These instructions are obsolete if you are using the [Atlassian Plugin SDK](#) (and you should be). This page is being retained only as reference for those using very old products which do not support the Atlassian Plugin SDK.

i Maven is included in the Atlassian SDK, as described in the page on [setting up your plugin development environment](#). **If you install the SDK, there is no need to install Maven separately.** If for some reason you have decided not to use the Atlassian SDK, you will need to install Maven as described below.

The Atlassian plugin development process depends on [Maven](#). You must have **Maven 2.1.0 or later**.

On this page:

- [What is Maven?](#)
- [Installing Maven](#)
- [Configuring your Maven settings.xml](#)
- [About Maven Settings, Repositories and Proxies](#)
- [Troubleshooting](#)

What is Maven?

Calling Maven a 'build tool' would be an understatement — but if you have no experience with Maven yet, start thinking of it as a tool for simplifying your life by building your plugin. Quoting the Maven home page:

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

Maven is particularly useful because it can automatically manage massive dependency trees, especially transitive dependencies where Plugin X depends on Application Y, which depends on Shared Library Z. Using Maven, you should be able to quickly and easily retrieve all the dependencies that your plugin will need, and create a working development environment in just a few steps.

Installing Maven

If for some reason you have decided not to use the Atlassian SDK, you will need to install Maven.

1. Download the latest release of [Maven 2](#).
2. Install Maven, following the instructions for your operating system on the [Maven download page](#).

Configuring your Maven settings.xml

The Atlassian SDK includes a correctly-configured Maven `settings.xml` file. We include the information below for those who have decided not to use the Atlassian SDK, or who wish to add the configurations to an existing `settings.xml` file.

1. If you do not already have a Maven `.m2` directory, create one now. By default, the directory should be in this location, where `USERNAME` is your username:
 - For Windows XP: `C:\Documents and Settings\USERNAME\.m2`
Note: Windows does not like the full stop at the beginning of the directory name, so you will probably have to create the directory via a command prompt, e.g: `mkdir C:\Documents and Settings\USERNAME\.m2`
 - For Windows Vista: `C:\Users\USERNAME\.m2`
Note: Windows does not like the full stop at the beginning of the directory name, so you will probably have to create the directory via a command prompt, e.g: `mkdir C:\Users\USERNAME\.m2`
 - For Mac: `/Users/USERNAME/.m2`
 - For UNIX/Linux: `/home/USERNAME/.m2`
 2. If you do not already have a Maven `settings.xml` file, create one now by copying our [example settings.xml](#). Put the file in your `.m2` directory.
 3. If you already have a Maven `settings.xml` file, edit it and include the components from our [example settings.xml](#).
-  If you look closely at our example settings, you will find two lines containing placeholders for username and password. Don't worry about them yet. You do not need them to get started.

About Maven Settings, Repositories and Proxies

In order for Maven to find and download all of the necessary dependencies, you need to tell it where everything is stored. This is done via the `settings.xml` file.

Your plugin will depend on many different artifacts: the Atlassian application itself, Atlassian modules and various open-source libraries. All of those artifacts are stored in different Maven repositories scattered around the net. However, to simplify configuration and speed up downloads, Atlassian provides a [Maven proxy](#) that contains *all* of the dependencies for all of our applications. The example settings file contains just one repository entry for this proxy, and saves Maven from having to check several different repositories for each artifact it needs. You can read more about the [repositories that are behind the Maven proxy](#).

Atlassian will release binary and Javadoc artifacts for all of our product releases and their Atlassian-managed dependencies. We will release source artifacts wherever we can freely do so. Some of our closed-source artifacts will only be available via manual download for source license holders.

We will also release various milestone and snapshot releases during application development, to aid plugin developers in testing their work against upcoming application releases.

You will be able to find all of these artifacts in the [Maven proxy](#).

Troubleshooting

Please refer to our [FAQ](#) and troubleshooting page.

RELATED TOPICS

Welcome to the Atlassian Developer Network

- How to learn to program (in Java or JavaScript)
- Installing the Atlassian Plugin SDK
- Writing your first plugin
- Advanced Plugin Development
- Atlassian Plugin SDK Documentation
- Other Information
- FAQ
- Tutorials
- Getting Involved in the Atlassian Developer Network
- Codegeist V
- Developer Relations State of the Union 2011 - Resources

Atlassian Plugin Archetypes

 These instructions are obsolete if you are using the [Atlassian Plugin SDK](#) (and you should be). Leaving here only as reference for those using very old products which do not support the SDK.

Atlassian has created Maven 2 archetypes for five of our pluggable products.

- **JIRA** - you can view the source code [here](#)
- **Confluence** - you can view the source code [here](#)
- **Bamboo** - you can view the source code [here](#)
- **Crowd** - you can view the source code [here](#)
- **Fisheye/Crucible** - you can view the source code [here](#)

See the build status for these archetypes [here](#).

Instructions

You can use a plugin archetype by running one of the Maven commands below. Maven will download all of the required dependencies, set up a project skeleton, and prepare you to run unit and functional tests. To set up your project template, run the command below, substituting your package name and plugin name where appropriate.

We recommend creating all of your archetypes at the top level of a new, empty directory. Be sure **not** to create a plugin archetype from a directory with an existing pom.xml, as it will try to add the plugin as a sub-module of the project in the current directory (which will fail unless the project has the packaging type "pom").

After creating the template plugin, it's a good idea to run some other command, like 'mvn compile', 'mvn idea:idea', or 'mvn eclipse:eclipse' to force mvn to download all of the project dependencies.

Atlassian Archetype Catalog

Recent versions of Maven support archetype discovery through catalog files. Catalog files always point to the latest available version of an archetype, and they can also prompt you for required properties through the command line.

```
mvn archetype:generate  
-DarchetypeCatalog=http://svn.atlassian.com/svn/public/atlassian/maven-plugins/archetype-catalog
```



Archetype catalogs require version 2.0-alpha-4 or later of the maven-archetype plugin. If the above commands don't work for you, try adding -cpu after mvn; this will force Maven to download the most recent version of the plugin.

Using Archetypes Manually

In general, we recommend using the archetype catalog in most cases, but if you prefer, you can manually create projects from specific archetypes. Follow the directions for the product you're targeting and the platform you're developing on. Note that you'll need to replace \$MY_PACKAGE and \$MY_PLUGIN (or %MY_PACKAGE% and %MY_PLUGIN%) with the actual package name and plugin ID that you wish to use.

JIRA

Unix

```
mvn archetype:create \  
-DarchetypeGroupId=com.atlassian.maven.archetypes \  
-DarchetypeArtifactId=jira-plugin-archetype \  
-DarchetypeVersion=15 \  
-DremoteRepositories=https://maven.atlassian.com/repository/public/ \  
-DgroupId=$MY_PACKAGE -DartifactId=$MY_PLUGIN
```

Windows

```
mvn archetype:create ^  
-DarchetypeGroupId=com.atlassian.maven.archetypes ^  
-DarchetypeArtifactId=jira-plugin-archetype ^  
-DarchetypeVersion=15 ^  
-DremoteRepositories=https://maven.atlassian.com/repository/public/ ^  
-DgroupId=%MY_PACKAGE% -DartifactId=%MY_PLUGIN%
```

Confluence

Unix

```
mvn archetype:create \  
-DarchetypeGroupId=com.atlassian.maven.archetypes \  
-DarchetypeArtifactId=confluence-plugin-archetype \  
-DarchetypeVersion=15 \  
-DremoteRepositories=https://maven.atlassian.com/repository/public/ \  
-DgroupId=$MY_PACKAGE -DartifactId=$MY_PLUGIN
```

Windows

```
mvn archetype:create ^
-DarchetypeGroupId=com.atlassian.maven.archetypes ^
-DarchetypeArtifactId=confluence-plugin-archetype ^
-DarchetypeVersion=15 ^
-DremoteRepositories=https://maven.atlassian.com/repository/public/ ^
-DgroupId=%MY_PACKAGE% -DartifactId=%MY_PLUGIN%
```

Bamboo

Unix

```
mvn archetype:create \
-DarchetypeGroupId=com.atlassian.maven.archetypes \
-DarchetypeArtifactId=bamboo-plugin-archetype \
-DarchetypeVersion=10 \
-DremoteRepositories=https://maven.atlassian.com/repository/public/ \
-DgroupId=$MY_PACKAGE -DartifactId=$MY_PLUGIN
```

Windows

```
mvn archetype:create ^
-DarchetypeGroupId=com.atlassian.maven.archetypes ^
-DarchetypeArtifactId=bamboo-plugin-archetype ^
-DarchetypeVersion=10 ^
-DremoteRepositories=https://maven.atlassian.com/repository/public/ ^
-DgroupId=%MY_PACKAGE% -DartifactId=%MY_PLUGIN%
```

Crowd

Unix

```
mvn archetype:create \
-DarchetypeGroupId=com.atlassian.maven.archetypes \
-DarchetypeArtifactId=crowd-plugin-archetype \
-DarchetypeVersion=1-SNAPSHOT \
-DremoteRepositories=https://maven.atlassian.com/repository/public/ \
-DgroupId=$MY_PACKAGE -DartifactId=$MY_PLUGIN
```

Windows

```
mvn archetype:create ^
-DarchetypeGroupId=com.atlassian.maven.archetypes ^
-DarchetypeArtifactId=crowd-plugin-archetype ^
-DarchetypeVersion=1-SNAPSHOT ^
-DremoteRepositories=https://maven.atlassian.com/repository/public/ ^
-DgroupId=%MY_PACKAGE% -DartifactId=%MY_PLUGIN%
```

Fisheye/Crucible

Unix

```
mvn archetype:create \
-DarchetypeGroupId=com.atlassian.maven.archetypes \
-DarchetypeArtifactId=crucible-plugin-archetype \
-DarchetypeVersion=1-SNAPSHOT \
-DremoteRepositories=https://maven.atlassian.com/repository/public/ \
-DgroupId=$MY_PACKAGE -DartifactId=$MY_PLUGIN
```

Windows

```
mvn archetype:create ^
-DarchetypeGroupId=com.atlassian.maven.archetypes ^
-DarchetypeArtifactId=crucible-plugin-archetype ^
-DarchetypeVersion=1-SNAPSHOT ^
-DremoteRepositories=https://maven.atlassian.com/repository/public/ ^
-DgroupId=%MY_PACKAGE% -DartifactId=%MY_PLUGIN%
```

Example

For example, if Joe Developer from Acme Corp were writing his first Confluence macro, he would create an empty folder (naming it for example "plugin-project"), enter that directory, and then — replacing \$MY_PACKAGE with "com.acme.confluence.plugins", and \$MY_PLUGIN with "awesome-macro" — he would run the following command (assuming he's using a Unix command line):

```
mvn archetype:create \
-DarchetypeGroupId=com.atlassian.maven.archetypes \
-DarchetypeArtifactId=confluence-plugin-archetype \
-DarchetypeVersion=15 \
-DremoteRepositories=https://maven.atlassian.com/repository/public/ \
-DgroupId=com.acme.confluence.plugins -DartifactId=awesome-macro
```

This would download some 50 small files from the net, create a folder named "awesome-macro" into his "plugin-project" folder, and set up all he needs to work on the plugin.

Please return to [Installing the Atlassian Plugin SDK](#) for the next steps.

Developer Bamboo Server

This page points to continuous integration builds for Atlassian supported plugins and libraries.

View the Bamboo server

Anyone can view the current status of the plugin builds on the server:

- [Developer Bamboo Server](#).

The Developer Bamboo is used mostly for supported plugins and public Atlassian libraries.

[Recent Builds on the Atlassian Developer Bamboo Server]

[Bamboo build results feed for all builds](#) 
(This feed is updated whenever a build gets built)

Recent Changes in the Atlassian Developer Bamboo Server

[Bamboo build results feed for all builds](#) 
(This feed is updated whenever a build gets built)

Getting Started with Remote APIs

This page has a short URL: <http://bit.ly/satellitesTalk>

Overview

This page complements the [Summit 2011](#) lightning talk "Satellite Apps around the Cloud: Integrating your infrastructure with JIRA Studio". It includes links and material about the remote APIs provided by Atlassian products.

Remote API Documentation



The links below point to the remote API documentation for the versions of each product currently included in JIRA Studio. If you aren't using JIRA Studio, check that your product version matches the documentation that you are using.

- REST Tutorial
 - REST API Reference
- SOAP Overview
 - SOAP API Reference
- XML-RPC Overview
 - XML-RPC API Reference

Confluence

- REST Overview
 - REST API Reference
- SOAP & XML-RPC API spec

Bamboo

- REST API Usage guide
 - REST API Reference

Fisheye / Crucible

- REST guide (start here)
 - Fisheye REST API Reference
 - Crucible REST API Reference

Code Samples

- Code samples from the presentation.
- Various examples of using Confluence's remote APIs (many in Python)
- See also the documentation links above - there are several code examples within the docs.

Relevant Issues

- [CONF-3877](#) : Overloaded methods in Confluence's WSDL cause issues with some SOAP toolkits
 - If you hit this, try the XML-RPC API instead, or see [here](#) for some generic workarounds.

Interesting Projects

- Marvin: IRC chatbot providing an interface to JIRA. Uses JIRA's SOAP interface. Written in Scala.
- Atlassian CLI: Re-exposes Atlassian products' remote APIs as a command-line interface. Written in Java.
- Confluence4r: Simple and useful wrapper around Confluence's XML-RPC for Ruby.
- Confluence4r Rails Plugin: enhanced version of Confluence4r for Rails.
- Hercules the support bot / log analysis tool

Confluence Build Information

Sometimes it's helpful to know the exact versions, dates and build numbers of Confluence releases. Here is a list of all the build information we've collected about Confluence - please add any missing information you have. The information is ordered by **date** and not necessarily version number, this is especially true for DRs of upcoming versions.

| Version | Codename | System Favourite Colour | Release Date | Build Number | Major features/API/Code Changes |
|---------|----------|-------------------------|--------------|--------------|---|
| 3.5.9 | | | | 2166 | |
| 3.5.7 | | | | 2163 | |
| 3.5.6 | | | | 2162 | |
| 3.5.5 | | | | 2160 | |
| 3.5.4 | | | | 2156 | |
| 3.5.3 | | | | 2154 | |
| 3.5.2 | | | | 2153 | |
| 3.5.1 | | | 30-Mar-2011 | 2149 | |
| 3.5 | Kermes | | 09-Mar-2011 | 2148 | Improved user management (Embedded Crowd) |
| 3.4.9 | | | | 2042 | |
| 3.4.8 | | | | 2041 | |

| | | | | | |
|----------|--|--------------|-------------|------|---|
| 3.4.7 | | | | 2037 | |
| 3.4.6 | | | | 2036 | |
| 3.4.5 | | | | 2035 | |
| 3.4.3 | | | | 2033 | |
| 3.4.2 | | | | 2032 | |
| 3.4.1 | | | | 2030 | |
| 3.4 | | Periwinkle | 12-Oct-2010 | 2029 | |
| 3.3.3 | | | | 1925 | |
| 3.3.1 | | | | 1923 | |
| 3.3 | | Spindrift | | 1911 | |
| 3.2.1_01 | | Heliotrope | | 1814 | |
| 3.2 | | Heliotrope | | 1810 | |
| 3.1.2 | | Heliotrope | | 1725 | |
| 3.1.1 | | Heliotrope | | 1724 | |
| 3.1 | | Heliotrope | | 1723 | |
| 3.0.2 | | LemonChiffon | 6-Oct-2009 | 1636 | |
| 3.0.1 | | LemonChiffon | 20-Aug-2009 | 1634 | |
| 3.0.0_01 | | LemonChiffon | | 1627 | |
| 3.0 | | LemonChiffon | 01-Jun-2009 | 1626 | Social Features, Macro Browser, Improved RTE |
| 2.10.4 | | | | 1520 | |
| 2.10.3 | | | | 1519 | |
| 2.10.2 | | | | 1518 | |
| 2.10.1 | | | | 1517 | |
| 2.10 | | Gamboge | 03-Dec-2008 | 1515 | Widget Connector, Office Connector, Quick Nav, Did-You-Mean, upgraded TinyMCE and more. |
| 2.10-m5 | | Sangria | 24-Oct-2008 | 1508 | |
| 2.9.2 | | Sangria | 14-Oct-2008 | 1419 | |
| 2.9.1 | | Sangria | 08-Sep-2008 | 1418 | |
| 2.9 | | Sangria | 07-Aug-2008 | 1415 | <ul style="list-style-type: none"> com.atlassian.confluence.renderer.RenderContent is finally gone Labels Editing (UI) totally redone |
| 2.9-M5 | | | ??? | 1408 | |
| 2.8.2 | | | 03-Jul-2008 | 1325 | |
| 2.8.1 | | | 21-May-2008 | 1322 | |
| 2.8 | | Myrtle | 24-Apr-2008 | 1318 | Drop-down menu + collapsible comment UI, Page ordering, JMX AttachmentRemoveEvent now takes User as remover |
| 2.7.3 | | Cinnabar | 19-Mar-08 | 1116 | |
| 2.7.2 | | Cinnabar | 05-Mar-08 | | |
| 2.7.1 | | Cinnabar | 23-Jan-08 | 1112 | |
| 2.7 | | Cinnabar | 12-Dec-07 | 1109 | JIRA Trusted Auth, System Administrator permission, Attach files or create |
| 2.6.2 | | | 27-Nov-07 | 919 | |
| 2.6.1 | | | 03-Nov-07 | 916 | |

| | | | | | |
|---------|------------|------------|-----------|-------|---|
| 2.6 | | Ecru | 27-Sep-07 | 913 | New theme, updated Joda Time, Backdate/rename news |
| 2.6-DR1 | | | | | |
| 2.5.8 | | | | 814 | |
| 2.5.7 | | | | 813 | |
| 2.5.6 | | | | | |
| 2.5.5 | | | | | |
| 2.5.4 | | | 13-Jun-07 | 809 | |
| 2.5.3 | | | 29-May-07 | 808 | |
| 2.5.2 | | | | | |
| 2.5.1 | | | | 806 | |
| 2.5 | | Myrtle | | | Multiple page permissions |
| 2.4.5 | | | 12-Apr-07 | 708 | |
| 2.4.3 | | | 21-Mar-07 | 705 | |
| 2.4.2 | | Periwinkle | 12-Mar-07 | 703 | |
| 2.3 | Snowy | Claret | 03-Jan-07 | 641 | Clustering, People directory |
| 2.3-DR2 | | Viridian | 18-Sep-06 | 628 | |
| 2.2.10 | | | 30-Nov-06 | 528 | |
| 2.2.9 | | Taupe | 07-Sep-06 | 527 | |
| 2.2.8 | | | 08-Aug-06 | 525 | |
| 2.3-DR1 | | | 24-Jul-06 | | |
| 2.2.3 | | | 09-Jun-06 | 518 | |
| 2.2.2 | | | | 516 | |
| 2.2.1a | | | 18-May-06 | 515 | Enabled the Recursive Plugin Classloader, Plugin Manager Fixes |
| 2.2.1 | | | 17-May-06 | 514 | LDAP bugfix |
| 2.2 | Shoalhaven | Taupe | 26-Apr-06 | 512 | User Spaces, Plugin Subsystem Improvements, Localisation, CAPTCHA |
| 2.1.5a | | | 17-Mar-06 | 411 ! | |
| 2.1.5 | | | 16-Mar-06 | 411 | |
| 2.1.4 | | | 15-Feb-06 | 410 | |
| 2.1.3 | | | 23-Jan-06 | 408 | |
| 2.1.2 | | | 12-Jan-06 | 407 | |
| 2.1.1 | | | 23-Dec-05 | 406 | |
| 2.1 | Bogan | Chartreuse | 23-Dec-05 | 405 | Autosave, concurrent edit warnings, atlassian-user |
| 2.0.3 | | | 11-Dec-05 | 323 | |
| 2.0.2 | | | 05-Dec-05 | 322 | |
| 2.0.1 | | | 28-Nov-05 | 321 | |
| 2.0 | Yarra | Chartreuse | 16-Nov-05 | 320 | Rich text editor, labels, custom dashboard, RSS builder |
| 1.4.4 | | | 23-Sep-05 | 221 | |
| 1.4.3 | | | 15-Aug-05 | 219 | |
| 1.4.2 | | | 30-Jun-05 | 214 | |
| 1.4.1 | | | 02-Jun-05 | 212 | |
| 1.3.6 | | | 02-Jun-05 | 123 | |
| 1.4 | Hunter | Lime | 23-May-05 | 211 | v2Renderer, new UI, page permissions, plugin API, |

| | | | | | |
|------------|--------------|-----------|-------------|-----|--|
| 1.3.5 | | | 01-Mar-05 | 122 | |
| 1.3.4 | | | 15-Feb-05 | 121 | |
| 1.3.3 | | | 08-Feb-2005 | 116 | |
| 1.3.2 | | | 21-Jan-05 | 114 | |
| 1.3.1 | | | 17-Dec-04 | 113 | |
| 1.3 | Murrumbidgee | Lime | 30-Nov-04 | 112 | Mail archiving, themes, trash can, fine-grained permissions |
| 1.2.3 | | | 08-Oct-04 | 60 | |
| 1.2.2 | | | 23-Sep-04 | 60 | |
| 1.2.1 | | | 09-Sep-04 | 59 | |
| 1.2 | Swan | Jet Black | 23-Aug-04 | 58 | Space content lists, image thumbnails, advanced search |
| 1.1.2 | | | 21-Jun-04 | 57 | |
| 1.1.1 | | | 18-Jun-04 | 56 | |
| 1.1 | Nymboida | Almond | 09-Jun-04 | 53 | Macro management, attachment versioning, Powerpoint/Excel sear |
| 1.0.3a | | | 05-May-04 | 50 | |
| 1.0.3 | | | 30-Apr-04 | 49 | |
| 1.0.2 | | | 13-Apr-04 | 48 | |
| 1.0.1 | | | 06-Apr-04 | 47 | |
| 1.0 | Murray | Puce | 12-Mar-04 | 46 | First commercial release |
| 1.0-rc1 | | | | 36 | Atlassian redefines 'feature freeze' |
| 1.0-beta1 | | | 19-Dec-03 | 30 | Limited beta release |
| 1.0-alpha1 | | | | 19 | First external release |

FAQ

This page collects the FAQ pages for different developer experience levels and the product development pages.

- FAQs for writing plugins
 - Atlassian Plugin SDK FAQ
 - Writing your first plugin FAQ
 - Advanced Plugin Development FAQ
- FAQs for product development
 - JIRA
 - Confluence
 - Fisheye/Crucible
 - Bamboo

FAQs for writing plugins

Atlassian Plugin SDK FAQ

- Errors when Creating an Archetype
- How can I change the version of Java my plugin uses
- Maven Cannot Find Java Mail, Java Activation or JTA
- Maven is Unable to Download the Artifact from Any Repository
- Maven Parsing Error Unrecognised HTML Tag
- Maven Runs Out of Memory
- Maven Warning POM for X is Invalid
- SDK doesn't hear customizations for port or context path

Writing your first plugin FAQ

- Adding Selenium to Functional Tests
- Build Failure - Manifest Validation Errors
- Cannot Log In to Confluence using Admin Account
- Choosing a Logging Framework

- Choosing a Package Name
- Eclipse Maven Plugin Build Error
- If you change pom.xml, you may need to restart the atlas-cli
- Instant Loading of Plugin Resources
- Overriding the application's webapp when developing your plugin
- Specifying a particular version of the host application
- Using the Atlassian Plugin SDK with a Source Code License
- Using your own log4j configuration for your plugin

Advanced Plugin Development FAQ

- Adding WebSudo Support to your Plugin
- BeanCreationException from Spring Framework
- Best Practices for ensuring a plugin works with a new version of the product
- Bundling extra dependencies in an OBR
- Converting a Plugin to Plugins 2
- Deployment stalled due to spaces in directory path
- Detecting the Presence or Absence of Classes in Different OSGI Bundles
- Functional Testing with Multiple Products
- Getting Custom fields from Confluence V2 Searches
- Plugins that Cannot be Reloaded with pi
- Using Plugins for Testing and Development

FAQs for product development

JIRA

No content found for label(s) faq_jira_dev.

Confluence

No content found for label(s) faq_conf_dev.

Fisheye/Crucible

This page contains answers to frequently asked questions posed by FishEye/Crucible developers.

Feel free to comment, make submissions, or pose your own question on FishEye/Crucible Development here.

- Q: I'm getting the error "API access is disabled" as a response from <http://fisheye/api/rest/repositories> on my installation. How do I enable the API as a Fisheye administrator?
▶ Click here to expand...
A: There is a toggle to enable the API under "Server Settings" in the web admin interface. See [Configuring the FishEye Web Server](#) for more details.
- Q: Is there any way to return unique results from an EyeQL query?
▶ Click here to expand...
A: It is not currently possible to return unique results.
An improvement request exists: [FE-1136](#). Your vote and comments on that issue are appreciated.
- Q: How do I use AUI on a page generated by a servlet plugin module?
▶ Click here to expand...
A: FishEye 2.4 and earlier don't include AUI by default, so your servlet will need to explicitly include it. There are two ways to do this:
 1. In FishEye/Crucible 2.4 and later, specify that the context of its `page decorator` requires the `com.atlassian.auiplugin:ajs` resource. i.e. your `atlassian-plugin.xml` file should include:
.....
assuming your generated page will request the `atl.general` decorator.
 2. If you are rendering a velocity template, include
`$webResourceManager.requireResource('com.atlassian.auiplugin:ajs')` in your template.

Bamboo

No content found for label(s) faq_bamboo_dev.

Tutorials

Below are some tutorials on writing plugins for the Atlassian products.

JIRA Tutorials

- Writing listeners with the atlassian-event library
- Adding a REST Service to JIRA
- Writing a Plugin Gadget for JIRA
- Adding a JQL Function to JIRA
- Adding your own Menu Items to JIRA
- Creating a JIRA Report
- Writing Integration Tests for your JIRA plugin
- WebWork Sample Plugin – from Matt Doar, describing how JIRA uses WebWork
- Writing a Gadget that Displays the Days Left in a Version
- Available Permissions
- How to create a new Custom Field Type
- How to create Custom Workflow Elements for JIRA 3

Confluence Tutorials

- Writing a Confluence Theme
- Adding a REST Service to Confluence
- Writing Macros for Confluence
- Writing a Confluence Macro 1
- Integration Testing Confluence Plugins – from our friends at Customware
- Adding a custom action to Confluence
- Adding your own Menu Items to Confluence
- Defining a Pluggable Service in a Confluence Plugin
- Writing a Confluence macro that uses JSON
- Upgrading and Migrating an Existing Confluence Macro to Confluence 4.0
- Creating a new Confluence 4.0 Macro
- Creating A Template Bundle
- Extending the V2 search API
- Searching using the V2 Search API
- Writing a search result renderer

FishEye/Crucible Tutorials

- Crucible SCM Plugin Tutorial — Crucible SCM modules are plugins that make version control systems accessible to Crucible.
- Event Listener Plugin Module Tutorial — This is a brief tutorial which teaches you how to write a trivial event listener plugin.
- FishEye Twitter Integration Plugin Tutorial — The plugin created in this tutorial sends each of your commit messages to your Twitter account.
- Gadget Tutorial — This tutorial will teach you how to write a simple gadget which can display information from Crucible on the JIRA dashboard.
- Gutter Renderer Plugin Tutorial — This tutorial teaches you how to add extra information to annotated views of files, and to diffs
- Rendering a Velocity Template from Your Servlet — Velocity allows your Servlet Plugins to render HTML pages from simple templates.
- REST Service Plugin Module Tutorial — provide your own REST API
- Storing Plugin Settings — This tutorial demonstrates how to use SAL (Shared Access Layer) to let your plugin store its configuration settings.
- Using Logging From Your Plugin — This tutorial describes how to log messages from your plugin.
- Writing a REST Client in Perl
- Writing a REST Client in Python
- Using the FishEye REST API to Write a Gadget to Monitor Recent Changes

Cross-Product Tutorials

- Scheduling Events using SAL
- Writing an admin configuration screen
- Writing Unit Tests for your Plugin
- Writing Cross Product Plugins – from Jonathan Mort and our friends at Adaptavist
- Plugin Architecture, Episode IV ("A New Hope"): introduction to the plugin software stack, including Maven, plugin and module descriptors, etc.
- Plugin Architecture, Episode V ("The 3-Tiered Architecture Strikes Back"): introduction to the REST module type and several important SAL services.
- Plugin Architecture, Episode VI (Return of the jQuery)
- Plugin Architecture, Episode I (The Phantom Mess)

Gadget-Centric Tutorials

- Standalone Gadget Tutorial - Writing a JQL Gadget
- Writing an Atlassian Gadget – from Bo Wang and our friends at Customware

More

- Converting your Plugin to the new Atlassian Plugin SDK
- Testing REST plugins – from Jonathan Doklovic
- Plugins 2.0 & OSGi Gotchas – Video of session at Atlassian Summit 2010
- Measuring Quality: Review Coverage with Crucible & FishEye – Blog post detailing the many "moving parts" involved in the creation of the Crucible bundled Review Coverage Report plugin.

Plugin Gadget Tutorial - Using the FishEye REST API to Write a Gadget to Monitor Recent Changes



Draft and under construction

This tutorial is a DRAFT and is under construction. (To the tutorial author: Please leave this warning in place until your tutorial has been reviewed and tested.)

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).



Level of experience: Beginner

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'beginner' level, so you can follow it even if you have never developed a plugin or gadget before.

On this page:

- Overview
 - What You Will Learn
 - Assumed Knowledge
- Step 1. Create the Plugin Project
- Step 2. Customise the plugin Vendor and Description in your `pom.xml`
- Step 3. Create the Gadget Specification
- Step 4. Register the Plugin Module in the Plugin Descriptor
- Step 5. Add the Gadget to a Dashboard for Testing
- Step 6. Make the Gadget Do Something Useful
 - Clear the Current Contents of the Gadget
 - Get the list of repositories
 - Find the Recent Changesets for Each Repository
 - Add the HTML for a Changeset
- Step 7. Build, Install and Run the Plugin

Overview

This tutorial shows you how to write a gadget that lists the most recent changes which have been committed to the repositories monitored by a Fisheye instance.

What You Will Learn

After completing this tutorial you will know how to:

- Create a gadget to be hosted by Fisheye and displayed on the JIRA dashboard.
- Use FishEye and Crucible's built in REST services from a Gadget.

Assumed Knowledge

- Understanding of what a gadget is and the relationship between the dashboard a gadget is displayed on and the host the gadget is served from.
- Familiarity with writing Javascript.
- Familiarity with the `jQuery` library.

Your gadget will be a 'plugin' gadget. That means that it will be embedded within an Atlassian plugin. The plugin will consist of the following parts:

- An Atlassian plugin descriptor, that is an XML file enabling the plugin in FishEye.
- A gadget specification, that is an XML file containing the gadget's HTML, CSS and Javascript.

All these components will be contained within a single JAR file. We will discuss each component in the examples below.

If you are interested, you can [compare standalone gadgets and gadgets embedded in plugins](#).

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

When you run `atlas-create-fecru-plugin` you'll be asked to supply a `groupId` and an `artifactId` – use `com.atlassian.tutorial.fisheye` and `recentchangesgadget` respectively. When you create a plugin of your own, you should use a `groupId` based on your company's domain.



You need to make sure that your project is using FishEye 2.2.0 or later. Edit your `pom.xml` and set the `fecru.version` property to the build number of the version of FishEye you want to use, e.g.:

```
<properties>
    <fecru.version>2.2.0-465</fecru.version>
    ...
</properties>
```

Step 2. Customise the plugin Vendor and Description in your `pom.xml`

Here are the parts of `pom.xml` which you may wish to change:

```
...
<organization>
    <name>Example Company</name>
    <url>http://www.example.com/</url>
</organization>
...
<description>This is the com.atlassian.tutorial.fisheye:recentchangesgadget plugin for Atlassian
FishEye/Crucible.</description>
...
```

You should change the `organization.name` and `url` attributes to those of your company, and the `description` to a description of your plugin.

These values are copied to your `atlassian-plugin.xml` plugin descriptor file, and are displayed on the plugin administration page of FishEye when your plugin is loaded.

Step 3. Create the Gadget Specification

Here is a basic gadget specification using the Atlassian Gadgets JavaScript Framework:

```

<?xml version="1.0" encoding="UTF-8" ?>
<Module>
<ModulePrefs title="Recent Changes" directory_title="Recent Changes" description="Recent Changes in FishEye repositories">
<Require feature="dynamic-height" />
<Optional feature="auth-refresh"/>
<Require feature="setprefs" />
<Require feature="setttitle" />
<Require feature="views" />
<Optional feature="atlassian.util" />
<Optional feature="gadget-directory">
    <Param name="categories">FishEye</Param>
</Optional>
</ModulePrefs>
<Content type="html">
<![CDATA[
<!-- The #-directive below will include all required JavaScript and CSS resources needed to use the Atlassian Gadgets JavaScript Framework. --&gt;
#requireResource("com.atlassian.gadgets.publisher:ajs-gadgets")
#includeResources()

&lt;script type="text/javascript"&gt;
(function () {
/* (2) Construct and initialise the gadget */
var gadget = AJS.Gadget({
    baseUrl: "__ATLASSIAN_BASE_URL__", /* (3) Used to make the application's base URL available to the gadget */
    view: { /* (4) Defines the view logic and initialises the view */
        enableReload: true,
        onResizeReload: false,
        onResizeAdjustHeight: true,
        template: function (args) {
            this.getView().html("&lt;div&gt;Hello World&lt;/div&gt;");
        },
        args: []
    }
});
})();
&lt;/script&gt;
]]&gt;
&lt;/Content&gt;
&lt;/Module&gt;
</pre>

```

Copy the XML code from the above gadget specification and put it in a new file named `gadget.xml` in the `src/main/resources` directory.

Step 4. Register the Plugin Module in the Plugin Descriptor

1. Add the `<gadget>` the plugin module to your plugin descriptor, `atlassian-plugin.xml`.

```

<atlassian-plugin key="${project.groupId}.${project.artifactId}"
name="${project.artifactId}" plugins-version="2">
<plugin-info>
    <description>${project.description}</description>
    <version>${project.version}</version>
    <vendor name="${project.organization.name}" url="${project.organization.url}" />
</plugin-info>
<gadget key="gadget" location="gadget.xml"/>
</atlassian-plugin>

```

2. Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Step 5. Add the Gadget to a Dashboard for Testing

The gadget you are creating in this tutorial will be 'served' by a FishEye instance, but 'hosted' on a JIRA dashboard. That is, the HTTP requests which serve the plugin descriptor you created above, and the plugin's data and resources will go to a FishEye instance in which the plugin is installed, but the gadget will be displayed on a JIRA dashboard.

You will need an instance of JIRA on which you have administrative privileges. This JIRA instance must also be able to 'see' the FishEye instance serving the plugin. A JIRA instance outside your firewall will not be able to see a FishEye instance running on your desktop development machine without special network configuration, for instance.

To add the plugin you will need to:

1. Start up your FishEye instance with the plugin SDK command `atlas-run`. This builds your plugin and starts a FishEye instance with the plugin installed in it.
2. Go to <http://localhost:3990/fecru/admin/viewplugins.do> and check that your plugin has loaded successfully. You should see this:

| recentchangesgadget | This is the com.atlassian.tutorial.fisheye:recentchangesgadget plugin for Atlassian FishEye/Crucible. | 1.0-SNAPSHOT | Enabled | All modules enabled. | Disable all |
|---|---|--------------|---------|----------------------|-------------------------|
| com.atlassian.tutorial.fisheye.recentchangesgadget:gadget | | | Enabled | | Disable |

3. Add the plugin to your JIRA dashboard:
 - a. Log in as an administrator.
 - b. Select Tools, Create Dashboard... and create a new, blank dashboard.
 - c. When your new dashboard displays, click 'add a new gadget'
 - d. Click the 'Add Gadget to Directory' button and type <http://localhost:3990/fecru/rest/gadgets/1.0/g/com.atlassian.tutorial.fisheye.recentchangesgadget:gadget> and click 'Add Gadget'.
 - e. Your gadget will appear highlighted in yellow. Click the 'Add it Now' button, and then the 'Finished' button.
4. You should see a new gadget containing the text 'Hello World' appear on your dashboard.



If you see the error message `Error loading gadget: org.apache.shindig.gadgets.GadgetException: Unable to retrieve gadget xml. HTTP error 504` try stopping and starting JIRA.



JIRA sometimes caches your gadget, so you'll need to restart JIRA when you change your `gadget.xml` file, otherwise you won't see your changes.

Step 6. Make the Gadget Do Something Useful

Now you will write the JavaScript and HTML code to retrieve recent changes data from FishEye and display the information in the gadget. As you can see below, using the built-in REST interfaces of FishEye to retrieve the information we want for this gadget involves many REST calls. If this did not perform satisfactorily you would need to consider writing your own REST endpoint to retrieve exactly the data you need in a single call. See [this tutorial](#) for an example of creating your own REST endpoint.

Because your gadget is embedded in a plugin, you can use the [Atlassian Gadgets JavaScript Framework](#) in addition to the OpenSocial JavaScript API.

Clear the Current Contents of the Gadget

We are going to build up the HTML displayed in the gadget as we go, so first we will remove its contents:

```
var gadget = this;
gadget.getView().html("<div class='main'><h1>Recent Changesets</h1></div>");
```

`gadget.getView()` returns a jQuery object representing the body of the Gadget. We replace any existing contents with a `div` which will contain all our content, and a `h1` containing our heading.

Get the list of repositories

First the gadget retrieves a list of all the repositories in the FishEye instance.

```

...
var reportError = function(request, textStatus, errorThrown) {
    console.log("Request:");
    console.log(request);
    console.log("TextStatus:");
    console.log(textStatus);
    console.log("ErrorThrown:");
    console.log(errorThrown);
};

AJS.$.ajax({
    url: "/rest-service-fe/repositories-v1",
    type: "GET",
    dataType: "xml",
    success: function(msg){
        AJS.$(msg).find('repository').each(function() {
            var name = AJS.$(this).attr('name');
            loadChangesetsForRepo(name);
        });
    },
    error: reportError // this function just dumps the error to the FireBug console
});

...

```

Some things to notice in the code above:

1. We are retrieving data from the REST endpoint as XML, and using jQuery to parse it. The Atlassian Gadgets Javascript framework prefixes jQuery with AJS, so the jQuery \$ function name becomes AJS.\$.
2. The `reportError` function uses the Firebug `console` to report errors.
3. Each repository name is passed to the `loadChangesetsForRepo` function, which we will write in the next step.

Find the Recent Changesets for Each Repository

Now we make another REST call for each repository, getting the recent changesets from the repository:

```

...
var loadChangesetsForRepo = function(repoName, repoDiv) {
    var repoDiv = AJS.$("<div class='repo'><h2>" + repoName + "</h2></div>");
    gadget.getView().find("div.main").append(repoDiv);
    AJS.$.ajax({
        url: "/rest-service-fe/revisionData-v1/changesetList/" + repoName +
"?maxReturn=10",
        type: "GET",
        dataType: "xml",
        success: function(msg){
            AJS.$(msg).find('csid').each(function() {
                var csid = AJS.$(this).text();
                addChangesetData(repoName, csid, repoDiv)
            });
        },
        error: reportError
    });
};
...

```

1. The first thing this function does is to use jQuery to create a div to hold the changesets from this repository. The div is added as a child element to the div with the class `main` which was created at the start.
2. The function uses jQuery to find each `csid` element in the XML response. Each `csid` value is passed to the `addChangesetData` function to load the details for that changeset.

Add the HTML for a Changeset

For each changeset, we make a REST call to get its details – in this tutorial we only display the id and the comment.

```

...
var addChangesetData = function(repoName, csid, repoDiv) {
    AJS.$.ajax({
        url: "/rest-service-fe/revisionData-v1/changeset/" + repoName + "/" + csid,
        type: "GET",
        dataType: "xml",
        success: function(msg){
            console.log(AJS.$(msg).find("comment"));
            var comment = AJS.$(msg).find("comment").text();
            repoDiv.append("<div class='changeset'>" + csid + ":" + comment + "</div>");
        },
        error: reportError
    });
};

...

```

Step 7. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).



Congratulations, that's it

Your gadget is complete. Have a chocolate!

Plugin Gadget Tutorial - Writing a Gadget that Displays the Days Left in a Version



Level of experience: Beginner

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'beginner' level, so you can follow it even if you have never developed a plugin or gadget before.

The source code of the plugin used in this tutorial is available in our [Subversion repository](#)

On this page:

- [Overview](#)
- [Step 1. Create the Plugin Project](#)
- [Step 2. Add Plugin Name etc to Plugin descriptor](#)
- [Step 3. Create the Gadget Spec](#)
- [Step 4. Customise the Plugin Descriptor and Maven POM](#)
- [Step 5. Make Resources Available to your Gadget](#)
- [Step 6. Add the Gadget to a Dashboard for Testing](#)
- [Step 7. Create the Config Mode for the Gadget](#)
- [Step 8. Adding some CSS to our Gadget](#)
- [Step 9. Use Javascript to get the Versions into the Gadget](#)
- [Step 10. Build, Install and Run the Plugin](#)
- [Step 11. Writing Unit Tests](#)
- [Step 12. Test your Updates on your Dashboard](#)

Overview

In this tutorial, we're going to create a new Atlassian gadget for JIRA, bundle it inside a plugin, use a REST resource to provide it with data, and have the gadget talk to the resource. This gadget will display the days left before a given version is scheduled to be released.

Your gadget will be a 'plugin' gadget. That means that it will be embedded within an Atlassian plugin. The plugin will consist of the following parts:

- Gadget spec file to hold the gadget's XML and JavaScript
- Java classes implementing the REST resource the gadget will use
- Plugin descriptor to enable the plugin module in JIRA

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

If you are interested, you can [compare standalone gadgets and gadgets embedded in plugins](#).

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

We'll be using the Atlassian Plugin SDK throughout the tutorial, so make sure you have it installed and working as described here. To check that you're ready to go, try the `atlas-version` command. You should see output like the following:

```
ATLAS Version:      3.0.4
ATLAS Home:        /Users/tchan/Products/atlassian-plugin-sdk-3.0.4
ATLAS Scripts:     /Users/tchan/Products/atlassian-plugin-sdk-3.0.4/bin
ATLAS Maven Home:  /Users/tchan/Products/atlassian-plugin-sdk-3.0.4/apache-maven
-----
Executing: /Users/tchan/Products/atlassian-plugin-sdk-3.0.4/apache-maven/bin/mvn --version
Apache Maven 2.1.0 (r755702; 2009-03-19 06:10:27+1100)
Java version: 1.6.0_15
Java home: /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home
Default locale: en_US, platform encoding: MacRoman
OS name: "mac os x" version: "10.6.2" arch: "x86_64" Family: "mac"
```

When prompted, create a JIRA plugin with the following specifications:

```
Define value for groupId: : com.atlassian.plugins.tutorial
Define value for artifactId: : jira-gadget-tutorial.plugin
Define value for version: 1.0-SNAPSHOT: :
Define value for package: com.atlassian.plugins.tutorial: :
```

Step 2. Add Plugin Name etc to Plugin descriptor

Edit the plugin descriptor at `src/main/resources/atlassian-plugin.xml` to give your plugin a unique key, name, description and vendor, as shown below.

Here is the `atlassian-plugin.xml` for your plugin:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}"
plugins-version="2">
    <plugin-info>
        <description>${project.description}</description>
        <version>${project.version}</version>
        <vendor name="${project.organization.name}" url="${project.organization.url}" />
    </plugin-info>
</atlassian-plugin>
```

Step 3. Create the Gadget Spec

First we will create the gadget spec file. This is `src/main/resources/days-left-gadget.xml`:

```

<?xml version="1.0" encoding="UTF-8" ?>
<Module>
    <ModulePrefs title="__MSG_gadget.days.left.title__"
                 directory_title="__MSG_gadget.days.left.title__"
                 description="__MSG_gadget.days.left.description__">
        <Require feature="dynamic-height"/>
        <Require feature="oauthpopup"/>
        <Require feature="setprefs"/>
        <Require feature="setttitle"/>
        <Require feature="views"/>
        <Optional feature="atlassian.util"/>
        <Optional feature="gadget-directory">
            <Param name="categories">
                JIRA
            </Param>
        </Optional>

        #oauth
        #supportedLocales("gadget.common,gadget.days.left")
    </ModulePrefs>
    <UserPref name="isConfigured" datatype="hidden" default_value="false"/>
    <UserPref name="projectId" datatype="hidden"/>
    <UserPref name="version" datatype="hidden" default_value="auto"/>
    <Content type="html">
        <![CDATA[
            <!--We will be adding code here soon to create out gadget-->
        ]]>
    </Content>
</Module>

```

You should recognise the `<ModulePrefs>` section as the metadata container for the gadget: title, directory title, description and so on. The `<Content>` section contains the HTML and/or JavaScript that drive the gadget's behaviour. We have left it blank here while we look more closely at `<ModulePrefs>`.

Notice the `<Optional>` feature. This 'gadget-directory' feature specifies that the gadget is for JIRA and should be placed in the 'JIRA' category in the gadget directory browser. Without this, it is much harder to find and use gadgets from the directory browser.

Now we will add an internationalisation file under `src/main/resources/i18n/i18n.properties`:

```

#days left in iteration gadget
gadget.days.left.title = Days Left
gadget.days.left.subtitle= Days Left: {0} - {1}
gadget.days.left.description=Displays the days remaining in specified project iteration
gadget.days.left.daysAgo = Days Ago
gadget.days.left.today = Today!
gadget.days.left.daysRemaining = Days Remaining
gadget.days.left.autoOption = Next Release Due (auto)
gadget.days.left.noReleaseDate = None
gadget.days.left.noVersionWarning = Selected project has no unreleased versions
gadget.days.left.noReleaseDatesWarning = Selected project has no versions with future release
dates.
gadget.days.left.configTitle = Days Left
gadget.days.left.releaseDate= Release Date

```

Step 4. Customise the Plugin Descriptor and Maven POM

Now we need to edit the plugin descriptor at `src/main/resources/atlassian-plugin.xml` to give our plugin a unique key, and some meta information about this plugin.

A gadget is a module in `atlassian-plugin.xml`.

For our plugin, we will start with a module declaration for the gadget spec:

```

<gadget key="test" location="days-left-gadget.xml"/>

```

There are two required properties to note:

- **key** must be unique for all modules in this plugin.
- **location** is the path to the gadget spec file, relative to `src/main/resources`.

Next add the `<resource>` element for the message bundle:

```
<resource type="i18n" location="i18n/i18n" name="i18n" />
```

Next add the `<rest>` element for your rest resource:

```
<rest key="tutorial-gadget-rest-resources" path="/tutorial-gadget" version="1.0">
    <description>Provides the REST resource for the project list.</description>
</rest>
```

Your `atlassian-plugin.xml` should now look as follows:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}"
plugins-version="2">
    <plugin-info>
        <description>${project.description}</description>
        <version>${project.version}</version>
        <vendor name="${project.organization.name}" url="${project.organization.url}" />
    </plugin-info>

    <!--
        Registers the gadget spec as a plugin module. This allows the gadget to
        appear in the gadget directory and also allows administrators to
        disable/enable the gadget.
    -->
    <gadget key="test" location="days-left-gadget.xml"/>

    <!-- Makes the gadget Locale messages available for the gadget's use. -->
    <resource type="i18n" location="i18n/i18n" name="i18n" />

    <!--Automatically finds all JAX-RS resource classes in the plugin andpublishes them.-->
    <rest key="tutorial-gadget-rest-resources" path="/tutorial-gadget" version="1.0">
        <description>Provides the REST resource for the project list.</description>
    </rest>
</atlassian-plugin>
```

Finally, to support the included REST module, update your `pom.xml` file so that it is identical to what is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>com.atlassian.plugin.tutorial</groupId>
    <artifactId>jira-gadget-tutorial-plugin</artifactId>
    <version>1.0-SNAPSHOT</version>

    <organization>
        <name>Example Company</name>
        <url>http://www.example.com/</url>
    </organization>

    <name>jira-gadget-tutorial-plugin</name>
    <description>This is the com.atlassian.plugin.tutorial:jira-gadget-tutorial-plugin plugin for
    Atlassian JIRA.</description>
    <packaging>atlassian-plugin</packaging>

    <dependencies>
        <dependency>
            <groupId>com.atlassian.gadgets</groupId>
            <artifactId>atlassian-gadgets-api</artifactId>
```

```

<version>1.1.5.rc1</version>
</dependency>
<dependency>
    <groupId>com.atlassian.gadgets</groupId>
    <artifactId>atlassian-gadgets-spi</artifactId>
    <version>1.1.5.rc1</version>
</dependency>
<dependency>
    <groupId>com.atlassian.jira</groupId>
    <artifactId>atlassian-jira</artifactId>
    <version>${jira.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.6</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.jira</groupId>
    <artifactId>jira-func-tests</artifactId>
    <version>${jira.version}</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>javax.ws.rs</groupId>
    <artifactId>jsr311-api</artifactId>
    <version>1.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.plugins.rest</groupId>
    <artifactId>atlassian-rest-common</artifactId>
    <version>1.1.0.beta6</version>
    <type>jar</type>
</dependency>
<dependency>
    <groupId>com.atlassian.jira</groupId>
    <artifactId>jira-rest-plugin</artifactId>
    <version>${jira.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.plugins.rest</groupId>
    <artifactId>atlassian-rest-common</artifactId>
    <version>1.0.2</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>2.3</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.sal</groupId>
    <artifactId>sal-api</artifactId>
    <version>2.1.beta4</version>
</dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>com.atlassian.maven.plugins</groupId>
            <artifactId>maven-jira-plugin</artifactId>

```

```
<version>3.0.4</version>
<extensions>true</extensions>
<configuration>
    <productVersion>${jira.version}</productVersion>
    <productDataVersion>${jira.data.version}</productDataVersion>
</configuration>
</plugin>
</plugin>
<artifactId>maven-compiler-plugin</artifactId>
<configuration>
    <source>1.5</source>
    <target>1.5</target>
</configuration>
</plugin>
</plugins>
</build>

<properties>
    <jira.version>4.0.1</jira.version>
    <jira.data.version>4.0</jira.data.version>
```

```
</properties>  
  
</project>
```

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Step 5. Make Resources Available to your Gadget

For this gadget we will need to write a REST resource which retrieves information about all the versions of the project which the user selects in the config mode of the gadget we are creating.

Create a new Java file called `DaysLeftInVersionResource.java` in the following location:
`/src/main/java/com/atlassian/plugin/tutorial`

Your `DaysLeftInVersionResource.java` should be identical to the code below:

```
package com.atlassian.plugin.tutorial;  
  
import com.atlassian.jira.bc.issue.search.SearchService;  
import com.atlassian.jira.project.Project;  
import com.atlassian.jira.project.version.Version;  
import com.atlassian.jira.project.version.VersionManager;  
import com.atlassian.jira.security.JiraAuthenticationContext;  
import com.atlassian.jira.util.velocity.VelocityRequestContextFactory;  
import com.atlassian.jira.web.util.OutlookDate;  
import com.atlassian.plugins.rest.common.security.AnonymousAllowed;  
import org.apache.commons.lang.builder.EqualsBuilder;  
import org.apache.commons.lang.builder.HashCodeBuilder;  
import org.apache.commons.lang.builder.ToStringBuilder;  
import org.apache.commons.lang.builder.ToStringStyle;  
  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.Date;  
import java.util.List;  
import javax.ws.rs.GET;  
import javax.ws.rs.Path;  
import javax.ws.rs.Produces;  
import javax.ws.rs.QueryParam;  
import javax.ws.rs.core.MediaType;  
import javax.ws.rs.core.Response;  
import javax.xml.bind.annotation.XmlElement;  
import javax.xml.bind.annotation.XmlRootElement;  
  
import static com.atlassian.jira.rest.v1.util.CacheControl.NO_CACHE;  
  
/**  
 * REST endpoint for days left in iteration gadget.  
 *  
 * @since v4.0  
 */  
@Path ("/days-left-in-iteration")  
@AnonymousAllowed  
@Produces ({ MediaType.APPLICATION_JSON })  
public class DaysLeftInVersionResource  
{  
  
    static final int MILLISECONDS_IN_SEC = 1000;
```

```

static final int SECONDS_IN_MIN = 60;
static final int MINUTES_IN_DAY = 60;
static final int HOURS_IN_DAY = 24;
private static final ToStringStyle TO_STRING_STYLE = ToStringStyle.SHORT_PREFIX_STYLE;

private final VersionManager versionManager;
private final JiraAuthenticationContext authenticationContext;
private final SearchService searchService;
private final VelocityRequestContextFactory velocityRequestContextFactory;

public DaysLeftInVersionResource(final SearchService searchService, final
JiraAuthenticationContext authenticationContext, final VelocityRequestContextFactory
velocityRequestContextFactory, final VersionManager versionManager)
{
    this.searchService = searchService;
    this.authenticationContext = authenticationContext;
    this.velocityRequestContextFactory = velocityRequestContextFactory;
    this.versionManager = versionManager;
}

@GET
@Path ("/getVersions")

public Response getVersionsForProject(@QueryParam ("projectId") String projectIdString)
{
    Long projectId = Long.valueOf(projectIdString.substring("project-".length()));
    List<Version> versions = getVersionList(projectId);

    final OutlookDate outlookDate = authenticationContext.getOutlookDate();
    long daysRemaining;
    List<VersionInfo> versionList = new ArrayList<VersionInfo>();

    String releaseDate;
    for (Version v : versions){
        releaseDate = formatDate(v.getReleaseDate());
        Project srcProj = v.getProjectObject();
        ProjectInfo targetProj = new ProjectInfo(srcProj.getId(), srcProj.getKey(),
srcProj.getName());
        if(releaseDate == ""){
            daysRemaining = 0;
        }
        else {
            daysRemaining = calculateDaysLeftInVersion(v.getReleaseDate());
        }
        versionList.add(new VersionInfo(v.getId(),v.getName(),
v.getDescription(),releaseDate,targetProj, daysRemaining));
    }

    return Response.ok(new VersionList(versionList)).cacheControl(NoCache).build();
}

public static long calculateDaysLeftInVersion(Date targetDate){
    Date currentDate = new Date(System.currentTimeMillis());
    Date releaseDate = targetDate; //TO DO need to write convert string to date FUNCTION
    long currentTime = currentDate.getTime();
    long targetTime = releaseDate.getTime();

    long remainingTime = targetTime - currentTime; //remaining time in milliseconds
    long hoursRemaining = remainingTime/(MILLISECONDS_IN_SEC* SECONDS_IN_MIN *
MINUTES_IN_DAY);
    long daysRemaining = remainingTime/(MILLISECONDS_IN_SEC* SECONDS_IN_MIN * MINUTES_IN_DAY *
HOURS_IN_DAY); //
    if(hoursRemaining % HOURS_IN_DAY > 0 ) {
        daysRemaining++; //the days remaining includes today should be updated for different
time z
    }
    return daysRemaining;
}

public String formatDate(Date date){
    if(date == null) {
        return "";
    } else {

```

```

        OutlookDate outlookDate = authenticationContext.getOutlookDate();
        return outlookDate.formatDMY(date);
    }
}
public List<Version> getVersionList(Long projectId)
{
    List<Version> versions = new ArrayList<Version>();
    versions.addAll(versionManager.getVersionsUnreleased(projectId, false));

    Collections.sort(versions, new Comparator<Version>()
    {
        public int compare(Version v1, Version v2)
        {
            if(v1.getReleaseDate()== null)
            {
                return 1;
            }
            else if (v2.getReleaseDate() == null)
            {
                return 0;
            }
            else {
                return v1.getReleaseDate().compareTo(v2.getReleaseDate());
            }
        }
    });
    return versions;
}
//CLOVER:OFF

/**
 * The data structure of the days left in iteration
 * <p/>
 * It contains the a collection of versionData about all the versions of a particular project
 */
@XmlRootElement
public static class VersionList
{
    @XmlElement
    Collection<VersionInfo> versionsForProject;

    @SuppressWarnings ({ "UnusedDeclaration", "unused" })
    private VersionList()
    { }

    VersionList(final Collection<VersionInfo> versionsForProject)
    {
        this.versionsForProject = versionsForProject;
    }

    public Collection<VersionInfo> getVersionsForProject()
    {
        return versionsForProject;
    }
}

@XmlRootElement
public static class ProjectInfo
{
    @XmlElement
    private long id;

    @XmlElement
    private String key;

    @XmlElement
    private String name;

    @SuppressWarnings ({ "UnusedDeclaration", "unused" })
    private ProjectInfo()

```

```
{}

ProjectInfo(final long id, String key, String name)
{
    this.id = id;
    this.key = key;
    this.name = name;
}

public long getId()
{
    return id;
}

public String getKey()
{
    return key;
}

public String getName()
{
    return name;
}

@Override
public int hashCode()
{
    return HashCodeBuilder.reflectionHashCode(this);
}

@Override
public boolean equals(final Object o)
{
    return EqualsBuilder.reflectionEquals(this, o);
}

@Override
public String toString()
{
    return ToStringBuilder.reflectionToString(this, TO_STRING_STYLE);
}

@XmlRootElement
public static class VersionInfo
{
    @XmlElement
    private long id;

    @XmlElement
    private String name;

    @XmlElement
    private String description;

    @XmlElement
    private String releaseDate;

    @XmlElement
    private long daysRemaining;

    @XmlElement
    private boolean isOverdue;

    @XmlElement
    private ProjectInfo owningProject;

    @SuppressWarnings ({ "UnusedDeclaration", "unused" })
    private VersionInfo()
    {

        VersionInfo(final long id, final String name, final String description, final String
releaseDate, final ProjectInfo owningProject, final long daysRemaining)
        {
            this.id = id;
        }
    }
}
```

```
        this.name = name;
        this.description = description;
        this.releaseDate = releaseDate;
        this.isOverdue = isOverdue();
        this.owningProject = owningProject;
        this.daysRemaining = daysRemaining;

    }

    public long getId()
    {
        return id;
    }

    public String getName()
    {
        return name;
    }

    public String getDescription()
    {
        return description;
    }

    public String getReleaseDate()
    {
        return releaseDate;
    }

    public long getDaysRemaining()
    {
        return daysRemaining;
    }

    public boolean isOverdue ()
    {
        if (daysRemaining < 0 )
        {
            isOverdue = true;
        }
        else
        {
            isOverdue = false;
        }
        return isOverdue;
    }

    public ProjectInfo getOwningProject()
    {
        return owningProject;
```

```
        }
    }
}
```

To learn more about writing REST resources check out the [tutorial on writing REST services](#).

Step 6. Add the Gadget to a Dashboard for Testing

If you haven't already done so, create a new dashboard to your JIRA instance.

Your gadget can already do something: It can say 'Hello world!'. Test it by adding it to JIRA. You will need a developer or test JIRA instance where you have administrative permission to add gadgets to that instance:

1. Go to a JIRA dashboard that you have created (or create a new one) and click '**Add Gadget**'.
2. The '**Add Gadget**' screen appears, showing the list of gadgets in your directory.
 Your gadget should already appear in the list, because it is added as part of the plugin.
3. Click the '**Add it Now**' button under your gadget to add the gadget to your dashboard.

At this stage you may wish to add two or three projects each containing a few versions in JIRA.

- To learn how to add new projects check out the [video tutorial on adding projects](#).
- To learn how to add new versions to a project check out the [documentation on managing versions](#).

Step 7. Create the Config Mode for the Gadget

We will now add some code in the Content section of our `days-left-gadget.xml` to create the config mode for our gadget.

Add the following code to the content section (inside the section labelled `<![CDATA[DOCSPRINT: <! --Code For Step 7 goes here-->]]>`) of your `days-left-gadget.xml` file

```
#requireResource( "com.atlassian.jira.gadgets:jira-global" )
#includeResources()

<script type="text/javascript">
(function ()
{
    var gadget = AJS.Gadget({
        baseUrl: "__ATLASSIAN_BASE_URL__",
        useOAuth: "/rest/gadget/1.0/currentUser",
        config: {
            descriptor: function(args)
            {
                var gadget = this;
                gadgets.window.setTitle("__MSG_gadget.days.left.configTitle__");
                var projectPicker = AJS.gadget.fields.projectPicker(gadget,
"projectId", args.projectOptions);

                return {

                    theme : function()
                    {
                        if (gadgets.window.getViewportDimensions().width < 450)
                        {
                            return "gdt top-label";
                        }
                        else
                        {
                            return "gdt";
                        }
                    }(),
                    fields: [
                        projectPicker,
                        AJS.gadget.fields.nowConfigured()
                    ]
                };
            },
            args: function()
            {
                return [

```

```
        {
            key: "projectOptions",
            ajaxOptions:
        "/rest/gadget/1.0/filtersAndProjects?showFilters=false"
    }

    ];
    }()
},
view: {
    onResizeAdjustHeight: true,
    enableReload: true,
    template: function (args)
    {
<!-- We will add code here in step 9 -->

    },
    args: [
    {
        key: "versions",
        ajaxOptions: function ()
        {
            return {
                url:
        "/rest/tutorial-gadget/1.0/days-left-in-iteration/getVersions",
                data: {
                    projectId :
gadgets.util.unescapeString(this.getPref("projectId")),
                }
            };
        }
    }
]
}

}
});
```

```
});  
};  
</script>
```

There a few important things to note:

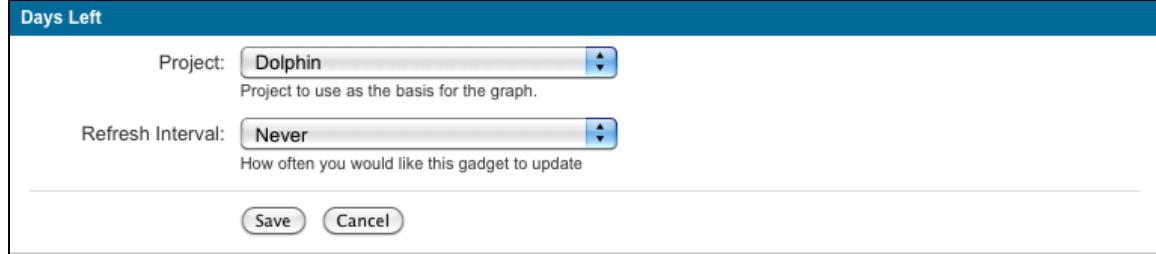
The section below is an Ajax call which retrieves a list of the all your JIRA projects. This list will then appear as the options in the project select field.

```
args: function()  
{  
    return [  
        {  
            key: "projectOptions",  
            ajaxOptions: "/rest/gadget/1.0/filtersAndProjects?showFilters=false"  
        }  
  
    ];  
}()
```

This is the line which makes use of the JSON returned by the above Ajax call and creates the project select field.

```
var projectPicker = AJS.gadget.fields.projectPicker(gadget, "projectId", args.projectOptions);
```

Your gadget should appear as follows on your dashboard:



Step 8. Adding some CSS to our Gadget

We will now add some CSS into our gadget. This will be used by in the view component of our gadget.

Add the following code to the content section of your gadget (`days-left-gadget.xml`) `css code` right after the line:

```
#includeResources()
```

Here is the `css code` to include:

```

<style type="text/css">
    #container {
        padding:15px;
    }
    #no-versions-warning {
        line-height: 1.4;
        font-size: 12px;
    }
    #days-box {
        text-align: center;
    }
    #days-value {
        text-align: center;
        font-size: 5em;
    }
    #days-text {
        padding-bottom: 15px;
    }
    #version-link {
        text-align: center;
    }
    #no-future-versions-warning {
        padding: 15px;
    }
    .view {
        padding:0.5em 1em;
    }
    .overdue {
        color: #cc0000;
    }
    .future-release {
        color: #00cc00;
    }
    .today {
        color: #cc0000;
    }
    #days-text .today {
        font-weight: bold;
    }

    .icon {
        padding-top: 3px;
        padding-right: 3px;
    }
    .disabled {
        color: #C0C0C0;
    }
</style>

```

Step 9. Use Javascript to get the Versions into the Gadget

Now we will create the view component of our gadget.

Add the code below to your days-left-gadget.xml file inside the **template: function (args){ }** section
(In the place where there is a comment that says <!-- We will add code here in Step 9 -->)

```

var versionData = args.versions;
        var currentVersion;
        var gadget = this;
        var baseUrl = AJS.$.ajaxSettings.baseUrl;
        var optionSelected = false;
        var projectVersionList;

        if (!versionData)
        {
            projectVersionList = null;
        }
        else
        {
            projectVersionList = AJS.$(args.versions.versionsForProject);
        }

```

```

        }

        var getContainer = function()
        {
            var container = AJS.$("<div/>").attr('id',
'container').appendTo(gadget.getView().empty());
            return function()
            {
                return container;
            }
        }();
        var hasVersionWithReleaseDate = function(projectVersionList)
{
    var hasReleaseDate = false;
    projectVersionList.each(function()
    {
        if (this.releaseDate != "")
        {
            hasReleaseDate = true;
        }
    });
    return hasReleaseDate;
};
        var setTitle = function(projectVersionList)
{
    if (!projectVersionList ||
!hasVersionWithReleaseDate(projectVersionList))
    {
        gadgets.window.setTitle(gadget.getMsg("gadget.days.left.title"));
    }
    else
    {

        gadgets.window.setTitle(AJS.format("__MSG_gadget.days.left.subtitle__",
currentVersion.owningProject.name, currentVersion.name));

    }
};

        var versionSelector = function(projectVersionList)
{
    var control = AJS.$("<select/>");
    AJS.$("<option/>").attr({id:'next-release-option',
value:'auto'}).text(gadget.getMsg('gadget.days.left.autoOption')).appendTo(control);

    projectVersionList.each(function()
{
        var option = AJS.$("<option/>").attr({ value:  this.id});
        if (this.releaseDate == "")
        {
            option.attr("disabled", "true");
            option.addClass('disabled');
            option.append(this.name + ' - ' +
gadget.getMsg('gadget.days.left.noReleaseDate'));
        }
        else
        {
            option.append(this.name + ' - ' + this.releaseDate);
        }

        if (this.id == gadget.getPref("version"))
        {
            option.attr({selected: "selected"});
            currentVersion = this;
            optionSelected = true;

        }
        control.append(option);
    });
    control.change(function(event)
{
    gadget.savePref("version", AJS.$(this).val());
    gadget.showView(true);
});

```

```

//generate image on side of select bar
AJS.$("#selection").append(AJS.$("<img/>").attr({
    src: baseUrl + "/images/icons/box_16.gif",
    height: 16,
    width: 16,
    title: gadget.getMsg("gadget.roadmap.status.unreleased"),
    alt: gadget.getMsg("gadget.roadmap.status.unreleased"),
    class: "icon"
}));
AJS.$("#selection").append(control);
//try auto select option if no option is selected
if (!optionSelected)
{
    AJS.$('#next-release-option').attr({selected: "selected"});
    currentVersion = projectVersionList[0];
}
};

var daysLeftDisplay = function(projectVersionList, container)
{
    var projectLink = baseUrl + "/browse/" +
currentVersion.owningProject.key
    var versionLink = projectLink + "/fixforversion/" + currentVersion.id

    container.append("<div id ='days-box' />");
    AJS.$("<div/>").attr("id", "days-value").appendTo("#days-box");
    AJS.$("<div/>").attr("id", "days-text").appendTo("#days-box");
    AJS.$("<div/>").attr("id", "version-link").appendTo("#days-box");

    AJS.$("<a/>").attr({
        href: projectLink,
        id: "projectLink"})
        .appendTo('#version-link');

    AJS.$("#version-link").append(" : ");

    AJS.$("<a/>").attr({
        href: versionLink,
        id: "versionLink"})
        .appendTo("#version-link");

    if (hasVersionWithReleaseDate(projectVersionList))
    {
        //if the currentVersion has no release date find the next version
due

        AJS.$("<div/>").attr("id", "days-text").appendTo("#days-box");
        AJS.$("<div/>").attr("id", "version-link").appendTo("#days-box");

AJS.$("#days-value").append(Math.abs(currentVersion.daysRemaining));

        AJS.$('#projectLink').text(currentVersion.owningProject.name);
        AJS.$('#versionLink').text(currentVersion.name);

        AJS.$('<div/>').attr('id',
'release-date').text(gadget.getMsg("gadget.days.left.releaseDate") + " : " +
currentVersion.releaseDate).appendTo('#version-link')

        if (currentVersion.daysRemaining < 0)
        {
            AJS.$('#days-value').addClass('overdue');
            AJS.$('#release-date').addClass('overdue');

AJS.$('#days-text').text(gadget.getMsg("gadget.days.left.daysAgo"))

        }
        else if (currentVersion.daysRemaining == 0)
        {
            AJS.$('#days-value').addClass('today');
            AJS.$('#release-date').addClass('today');

AJS.$('#days-text').addClass('today').text(gadget.getMsg("gadget.days.left.today"))
        }
    }
}

```

```

        else
        {
            AJS.$('#days-value').addClass('future-release');
            AJS.$('#release-date').addClass('future-release');

            AJS.$('#days-text').text(gadget.getMsg("gadget.days.left.daysRemaining"));

        }

    }
else
{
    AJS.$('#days-box').empty();

    var futureVersionsWarning = AJS.$("<div />")
        .attr('id', 'no-future-versions-warning')
        .text(" - " +
gadget.getMsg("gadget.days.left.noReleaseDatesWarning"))
        .appendTo('#days-box');

    AJS.$("<a/>")
        .attr({
            href: projectLink,
            id: "projectLink"})
        .text(currentVersion.owningProject.name)
        .prependTo(futureVersionsWarning);

}

};

if (!projectVersionList)
{
    var noVersionMsg = gadget.getMsg("gadget.days.left.noVersionWarning");
    gadget.getView().empty().append((noVersionMsg));

}
else
{

    var container = getContainer().append("<div id='selection' />");
    versionSelector(projectVersionList);
    daysLeftDisplay(projectVersionList, container);

    setTitle(projectVersionList);
}

```

```
    }  
},
```

The <Content> element in your gadget specification contains the working parts of the gadget. The <Content> element consists of:

- A CDATA declaration, to prevent the XML parser from attempting to parse the gadget content. Include '<! [CDATA[' (without the quotes) at the beginning and '']]>' (without the quotes) at the end of your <Content> element.
- Optional static HTML. When a dashboard renders the gadget, it will render this HTML.
- Optional JavaScript. You can declare JavaScript functions and call them in the usual way. Refer to the [OpenSocial JavaScript API](#) for details of gadget-specific API functions that any OpenSocial gadget container should support.
- Optional CSS style sheets.

Because your gadget is embedded in a plugin, you can use the [Atlassian Gadgets JavaScript Framework](#) in addition to the OpenSocial JavaScript API.

There are a few important things to note:

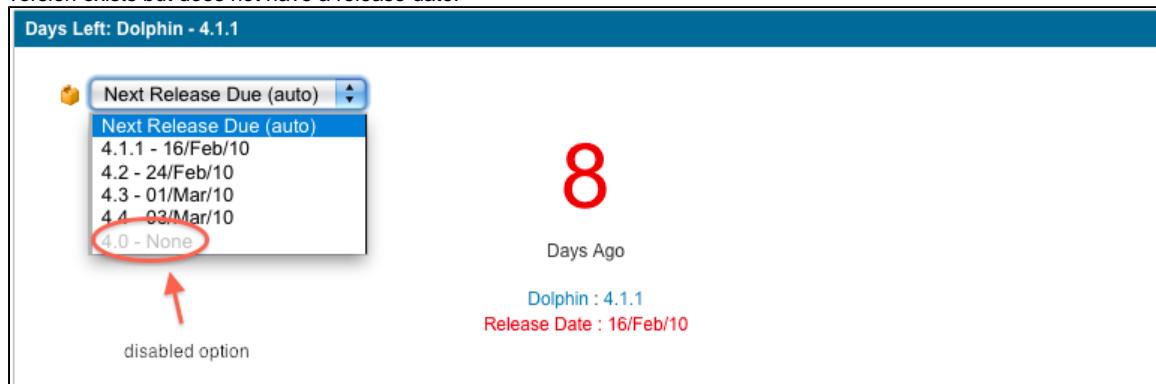
The versionSelector function (shown below) creates the drop down of all the unreleased versions for a specified project:

```
var versionSelector = function(projectVersionList)  
{  
    var control = AJS.$("<select/>");  
    AJS.$("<option/>").attr({id:'next-release-option',  
value:'auto'}).text(gadget.getMsg('gadget.days.left.autoOption')).appendTo(control);  
  
    projectVersionList.each(function()  
    {  
        var option = AJS.$("<option/>").attr({ value: this.id});  
        if (this.releaseDate == "")  
        {  
            option.attr("disabled", "true");  
            option.addClass('disabled');  
            option.append(this.name + ' - ' +  
gadget.getMsg('gadget.days.left.noReleaseDate'));  
        }  
        else  
        {  
            option.append(this.name + ' - ' + this.releaseDate);  
        }  
  
        if (this.id == gadget.getPref("version"))  
        {  
            option.attr({selected: "selected"});  
            currentVersion = this;  
            optionSelected = true;  
        }  
        control.append(option);  
    });  
    control.change(function(event)  
    {  
        gadget.savePref("version", AJS.$(this).val());  
        gadget.showView(true);  
    });  
    //generate image on side of select bar  
    AJS.$("#selection").append(AJS.$("<img/>").attr({  
src: baseUrl + "/images/icons/box_16.gif",  
height: 16,  
width: 16,  
title: gadget.getMsg("gadget.roadmap.status.unreleased"),  
alt: gadget.getMsg("gadget.roadmap.status.unreleased"),  
class: "icon"  
}));  
    AJS.$("#selection").append(control);  
    //try auto select option if no option is selected  
    if (!optionSelected)  
    {  
        AJS.$('#next-release-option').attr({selected: "selected"});  
        currentVersion = projectVersionList[0];  
    }  
};
```

All the parts of the code with AJS.\$ is in fact jQuery which is mainly used to format the appearance of the gadget view mode. Essentially the versionSelector function creates a select box whose options are the versions returned by the Ajax call defined in the following code.

```
args: [
    {
        key: "versions",
        ajaxOptions: function ()
        {
            return {
                url:
                    "/rest/tutorial-gadget/1.0/days-left-in-iteration/getVersions",
                data: {
                    projectId :
                        gadgets.util.unescapeString(this.getPref("projectId")),
                }
            };
        }
    }
]
```

If an unreleased version has no specified release date, it appears as a disabled option in the version select box to show a user that the version exists but does not have a release date.



The **displayDaysLeftGadget** function (as seen below) displays the number of days left in the selected version. The display varies slightly depending on whether or not the version is overdue or yet to be released.

```
var daysLeftDisplay = function(projectVersionList, container)
{
    var projectLink = baseUrl + "/browse/" +
currentVersion.owningProject.key
    var versionLink = projectLink + "/fixforversion/" + currentVersion.id

    container.append("<div id ='days-box' />");
    AJS.$("<div/>").attr("id", "days-value").appendTo("#days-box");
    AJS.$("<div/>").attr("id", "days-text").appendTo("#days-box");
    AJS.$("<div/>").attr("id", "version-link").appendTo("#days-box");

    AJS.$("<a/>").attr({
        href: projectLink,
        id: "projectLink"})
        .appendTo('#version-link');

    AJS.$("#version-link").append(" : ");

    AJS.$("<a/>").attr({
        href: versionLink,
        id: "versionLink"})
        .appendTo("#version-link");

    if (hasVersionWithReleaseDate(projectVersionList))
    {
        //if the currentVersion has no release date find the next version
        due
        AJS.$("<div/>").attr("id", "days-text").appendTo("#days-box");
    }
}
```

```

AJS.$("<div/>").attr("id", "version-link").appendTo("#days-box");

AJS.$("#days-value").append(Math.abs(currentVersion.daysRemaining));

AJS.$('#projectLink').text(currentVersion.owningProject.name);
AJS.$('#versionLink').text(currentVersion.name);

AJS.$('<div/>').attr('id',
'release-date').text(gadget.getMsg("gadget.days.left.releaseDate") + " : " +
currentVersion.releaseDate).appendTo('#version-link')

if (currentVersion.daysRemaining < 0)
{
    AJS.$('#days-value').addClass('overdue');
    AJS.$('#release-date').addClass('overdue');

AJS.$('#days-text').text(gadget.getMsg("gadget.days.left.daysAgo"))

}
else if (currentVersion.daysRemaining == 0)
{
    AJS.$('#days-value').addClass('today');
    AJS.$('#release-date').addClass('today');

AJS.$('#days-text').addClass('today').text(gadget.getMsg("gadget.days.left.today"))
}
else
{
    AJS.$('#days-value').addClass('future-release');
    AJS.$('#release-date').addClass('future-release');

AJS.$('#days-text').text(gadget.getMsg("gadget.days.left.daysRemaining"));

}

}
else
{
    AJS.$('#days-box').empty();

    var futureVersionsWarning = AJS.$("<div />")
        .attr('id', 'no-future-versions-warning')
        .text(" - " +
gadget.getMsg("gadget.days.left.noReleaseDatesWarning"))
        .appendTo('#days-box');

    AJS.$("<a/>")
        .attr({
            href: projectLink,
            id: "projectLink"})
        .text(currentVersion.owningProject.name)
        .prependTo(futureVersionsWarning)
}

```

```
        }
    };
```

Your final days-left-gadget.xml file should look as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
    <ModulePrefs title="__MSG_gadget.days.left.title__"
                 directory_title="__MSG_gadget.days.left.title__"
                 description="__MSG_gadget.days.left.description__">
        <Require feature="dynamic-height"/>
        <Require feature="oauthpopup"/>
        <Require feature="setprefs"/>
        <Require feature="settitle"/>
        <Require feature="views"/>
        <Optional feature="atlassian.util"/>
        #oauth
        #supportedLocales("gadget.common,gadget.days.left")
    </ModulePrefs>
    <UserPref name="isConfigured" datatype="hidden" default_value="false"/>
    <UserPref name="firstTime" datatype="hidden" default_value="true"/>
    <UserPref name="projectId" datatype="hidden"/>
    <UserPref name="version" datatype="hidden" default_value="auto"/>
    <Content type="html">
        <![CDATA[
            #requireResource("com.atlassian.jira.gadgets:jira-global")
            #includeResources()

            <style type="text/css">
                #container {
                    padding:15px;
                }
                #no-versions-warning {
                    line-height: 1.4;
                    font-size: 12px;
                }
                #days-box {
                    text-align: center;
                }
                #days-value {
                    text-align: center;
                    font-size: 5em;
                }
                #days-text {
                    padding-bottom: 15px;
                }
                #version-link {
                    text-align: center;
                }
                #no-future-versions-warning {
                    padding: 15px;
                }
                .view {
                    padding:0.5em 1em;
                }
                .overdue {
                    color: #cc0000;
                }
                .future-release {
                    color: #00cc00;
                }
                .today {
                    color: #cc0000;
                }
                #days-text .today {
                    font-weight: bold;
                }

                .icon {
                    padding-top: 3px;
                }
            </style>
        ]>
    </Content>
</Module>
```

```

        padding-right: 3px;
    }
    .disabled {
        color: #C0C0C0;
    }

```

```
</style>
```

```
<script type="text/javascript">
```

```
    (function ()
    {

        var gadget = AJS.Gadget({
            baseUrl: "__ATLASSIAN_BASE_URL__",
            useOAuth: "/rest/gadget/1.0/currentUser",
            config: {
                descriptor: function(args)
                {

                    var gadget = this;
                    gadgets.window.setTitle("__MSG_gadget.days.left.configTitle__");
                    var projectPicker = AJS.gadget.fields.projectPicker(gadget,
                    "projectId", args.projectOptions);

                    return {

                        theme : function()
                        {
                            if (gadgets.window.getViewportDimensions().width < 450)
                            {
                                return "gdt top-label";
                            }
                            else
                            {
                                return "gdt";
                            }
                        }(),
                        fields: [
                            projectPicker,
                            AJS.gadget.fields.nowConfigured()
                        ]
                    };
                },
                args: function()
                {
                    return [
                        {
                            key: "projectOptions",
                            ajaxOptions:
                            "/rest/gadget/1.0/filtersAndProjects?showFilters=false"
                        },
                        ],
                    ];
                }()
            },
            view: {
                onResizeAdjustHeight: true,
                enableReload: true,
                template: function (args)
                {
                    var versionData = args.versions
                    var currentVersion;
                    var gadget = this;
                    var baseUrl = AJS.$.ajaxSettings.baseUrl;
                    var optionSelected = false;
                    var projectVersionList;

                    if(!versionData) {
                        projectVersionList = null;
                    } else {
                        projectVersionList = AJS.$(args.versions.versionsForProject);
                    }

                    var getContainer = function() {

```

```

        var container = AJS.$("<div/>").attr('id',
'container').appendTo(gadget.getView().empty());
        return function() {
            return container;
        }
    })();
var hasVersionWithReleaseDate = function(projectVersionList) {
    var hasReleaseDate = false;
    projectVersionList.each(function()
    {
        if(this.releaseDate != "") {
            hasReleaseDate = true;
        }
    });
    return hasReleaseDate;
};
var setTitle = function(projectVersionList) {
    if(!projectVersionList || !hasVersionWithReleaseDate(projectVersionList))
    {
        gadgets.window.setTitle(gadget.getMsg("gadget.days.left.title"));
    }
    else
    {
        gadgets.window.setTitle(AJS.format("__MSG_gadget.days.left.subtitle__",
currentVersion.owningProject.name, currentVersion.name));
    }
};

var versionSelector = function(projectVersionList)
{
    var control = AJS.$("<select/>");
    AJS.$("<option/>").attr({id:'next-release-option',
value:'auto'}).text(gadget.getMsg('gadget.days.left.autoOption')).appendTo(control);

    projectVersionList.each(function()
    {
        var option = AJS.$("<option/>").attr({ value: this.id});
        if (this.releaseDate == "")
        {
            option.attr("disabled", "true");
            option.addClass('disabled');
            option.append(this.name + ' - ' +
gadget.getMsg('gadget.days.left.noReleaseDate'));
        }
        else
        {
            option.append(this.name + ' - ' + this.releaseDate);
        }

        if (this.id == gadget.getPref("version"))
        {
            option.attr({selected: "selected"});
            currentVersion = this;
            optionSelected = true;
        }
        control.append(option);
    });
    control.change(function(event)
    {
        gadget.savePref("version", AJS.$(this).val());
        gadget.showView(true);
    });
    //generate image on side of select bar
    AJS.$("#selection").append(AJS.$("<img/>").attr({
        src: baseUrl + "/images/icons/box_16.gif",
        height: 16,
        width: 16,
        title: gadget.getMsg("gadget.roadmap.status.unreleased"),
        alt: gadget.getMsg("gadget.roadmap.status.unreleased"),
        class: "icon"
    });
}

```

```

        });
        AJS.$("#selection").append(control);
        //try auto select option if no option is selected
        if(!optionSelected) {
            AJS.$('#next-release-option').attr({selected: "selected"});
            currentVersion = projectVersionList[0];
        }
    };
    var daysLeftDisplay = function(projectVersionList, container)
    {
        var projectLink = baseUrl + "/browse/" +
currentVersion.owningProject.key
        var versionLink = projectLink + "/fixforversion/" +
currentVersion.id

        container.append("<div id ='days-box' />");
        AJS.$("<div/>").attr("id", "days-value").appendTo("#days-box");
        AJS.$("<div/>").attr("id", "days-text").appendTo("#days-box");
        AJS.$("<div/>").attr("id", "version-link").appendTo("#days-box");

        AJS.$("<a/>").attr({
            href: projectLink,
            id: "projectLink"})
        .appendTo('#version-link');

        AJS.$("#version-link").append(" : ");

        AJS.$("<a/>").attr({
            href: versionLink,
            id: "versionLink"})
        .appendTo("#version-link");

        if(hasVersionWithReleaseDate(projectVersionList)) {
            //if the currentVersion has no release date find the next
version due

            AJS.$("<div/>").attr("id", "days-text").appendTo("#days-box");
            AJS.$("<div/>").attr("id",
"version-link").appendTo("#days-box");

            AJS.$("#days-value").append(Math.abs(currentVersion.daysRemaining));

            AJS.$('#projectLink').text(currentVersion.owningProject.name);
            AJS.$('#versionLink').text(currentVersion.name);

            AJS.$('<div/>').attr('id',
'release-date').text(gadget.getMsg("gadget.days.left.releaseDate") + " : " +
currentVersion.releaseDate).appendTo('#version-link')

            if (currentVersion.daysRemaining < 0)
            {
                AJS.$('#days-value').addClass('overdue');
                AJS.$('#release-date').addClass('overdue');

AJS.$('#days-text').text(gadget.getMsg("gadget.days.left.daysAgo"))

            }
            else if (currentVersion.daysRemaining == 0 )
            {
                AJS.$('#days-value').addClass('today');
                AJS.$('#release-date').addClass('today');

AJS.$('#days-text').addClass('today').text(gadget.getMsg("gadget.days.left.today"))
            }
            else
            {
                AJS.$('#days-value').addClass('future-release');
                AJS.$('#release-date').addClass('future-release');

AJS.$('#days-text').text(gadget.getMsg("gadget.days.left.daysRemaining")));
            }
        }
    }
}

```

```

        }

    }
    else {
        AJS.$('#days-box').empty();

        var futureVersionsWarning = AJS.$("<div />")
            .attr('id', 'no-future-versions-warning')
            .text(" - " +
gadget.getMsg("gadget.days.left.noReleaseDatesWarning"))
            .appendTo('#days-box');

        AJS.$("<a/>")
            .attr({
                href: projectLink,
                id:"projectLink"})
            .text(currentVersion.owningProject.name)
            .prependTo(futureVersionsWarning)

    }
}

if(!projectVersionList)
{
    var noVersionMsg =
gadget.getMsg("gadget.days.left.noVersionWarning");
    gadget.getView().empty().append((noVersionMsg));

}
else
{

    var container = getContainer().append("<div id='selection' />");
    versionSelector(projectVersionList);
    daysLeftDisplay(projectVersionList, container);

    setTitle(projectVersionList);
}

},
args: [
{
    key: "versions",
    ajaxOptions: function ()
    {
        return {
            url:
"/rest/tutorial-gadget/1.0/days-left-in-iteration/getVersions",
            data: {
                projectId :
gadgets.util.unescapeString(this.getPref("projectId")),
            }
        };
    }
}
]
}

});
})();
})();

```

</script>

```
]]>  
</Content>  
</Module>
```

Step 10. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Step 11. Writing Unit Tests

To learn more about writing unit tests check out the [tutorial on writing unit tests for your plugin](#).

Step 12. Test your Updates on your Dashboard

Below are some screenshots of the gadget we have just created in this tutorial. Your final result should look similar to the screenshots shown below.

If the selected version has not been released and is overdue the gadget shows the number of days the version is overdue in red. The text below the number is "Days Ago".



If the selected version is due today then the number displayed then the number of days is displayed in red and the text below the number is "Today!".



If the selected version is due in the future the number of days remaining is displayed in green and the text below the number is "Days To Go".

The screenshot shows a gadget titled "Days Left: Dolphin - 4.4". It displays a large green number "7" indicating days remaining. Below it, the text "Days Remaining" is shown. At the bottom, it says "Dolphin : 4.4" and "Release Date : 03/Mar/10".

If the selected project has no unreleased versions the gadget will appear as follows.

The screenshot shows a gadget titled "Days Left". The main content area contains the text "Selected project has no unreleased versions".

Finally if the selected project has no versions with release dates specified then the gadget will display the following.

The screenshot shows a gadget titled "Days Left". It displays a message: "Dragonfly - Selected project has no versions with future release dates." A green success message at the bottom right says "Congratulations, you have completed this tutorial."

RELATED TOPICS

Packaging your Gadget as an Atlassian Plugin

Plugin Tutorial - Adding a custom action to Confluence

Level of experience: Beginner to Intermediate

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'beginner' level, so you can follow it even if you have never developed a plugin before.

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).

On this page:

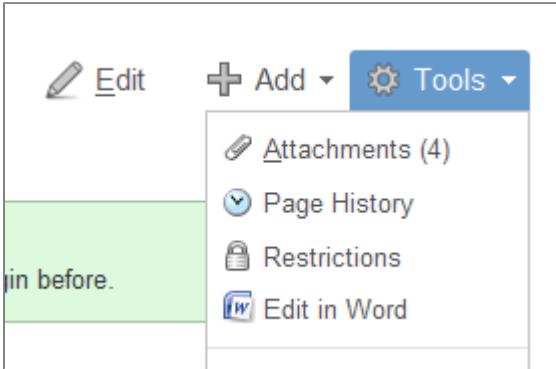
- Overview
- Step 1. Create the Plugin Project
- Step 2. Add Plugin Metadata to the POM
- Step 3. Register the Plugin Modules in the Plugin Descriptor
 - Fix the name parameter of the `atlassian-plugin` element
 - Add the Web Item Module
 - Add the XWork-Webwork Module
- Step 4. Write your Java Class
 - Set up your Java `AddDraftLabelAction` class which extends the `ConfluenceActionSupport` class
 - Add the `execute` method to your `AddDraftLabelAction` class

- Step 5. Build, Install and Run the Plugin
- Step 6. Modifying your source code from here

Overview

This tutorial shows you how to add a custom action to Confluence, which can be accessed via the 'Tools' menu when viewing a Confluence page or blog post. When you choose the "Add 'draft' label" custom action from the Tools menu, a 'draft' label will be added to the page.

Screenshot: The Confluence 'Tools' menu



In order to do this, you will create a Confluence plugin, which will consist of the following components:

- Plugin descriptor to enable the plugin module in Confluence.
- Resources for adding a web item in Confluence and defining what this action does (via Confluence Web Item and XWork-Webwork modules).
- A Java class that encapsulates the plugin logic (which implements the Confluence PageAware interface and injects the LabelManager dependency).

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.



Before you proceed:

To get a better understanding of how to create Java code that interacts with Confluence, you might wish to [download the Confluence source code](#) (available to commercial customers and developers) and examine some of the source code first. The source code should provide many clues on how Java classes, methods, interfaces and dependency injections are implemented in Confluence.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
 - Create and install your plugin from a template
1. Follow the instructions in the above SDK documents to set up your plugin development environment.
 2. Follow the instructions in the above SDK documents to run `atlas-create-confluence-plugin`. This will create your skeleton plugin. You will be prompted for some information to identify your plugin. Enter the following information:
 - **group-id**: com.atlassian.plugins.tutorial
 - **artifact-id**: AddDraftLabelAction
 - **version**: 1.0
 - **package**: com.atlassian.plugins.tutorial

Step 2. Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
<organization>
  <name>Example Company</name>
  <url>http://www.example.com/</url>
</organization>
```

3. Update the <name> element:

```
<name>Add 'draft' label</name>
```

4. Update the <description> element:

```
<description>This plugin adds the label 'draft' to the page, via an "Add 'draft' label" action from the Tools menu.</description>
```

5. Save the file.



Using an older version of the Atlassian Plugin SDK that runs against Confluence 3.0.1 by default?

If so, you will need to add the following instructions element (and its content) within the build/plugins/plugin/configuration element of the pom.xml file:

```
<instructions>
    <Import-Package>
        com.atlassian.confluence.*;version="3.0.1"
    </Import-Package>
</instructions>
```

If you do not, you may end up with an error when you attempt to run the atlas-run command (in step 5 below).

Step 3. Register the Plugin Modules in the Plugin Descriptor

Next, you will need add the required plugin modules to your plugin descriptor at src/main/resources/atlassian-plugin.xml. The plugin descriptor is an XML file that identifies the plugin to Confluence and defines the functionality that the plugin requires.

You must now register the plugin module in your plugin descriptor, atlassian-plugin.xml.

Fix the name parameter of the atlassian-plugin element



This sub-step is required due to a bug in versions of the Atlassian Plugin SDK kit up to 3.0.4. However, from version 3.1 of this kit you can skip this sub-step.

If the name parameter in the atlassian-plugin is \${project.artifactID}, change it to \${project.name} so that it uses the <name> element you defined in the previous step.

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.name}" plugins-version="2">
```

Add the Web Item Module

First, you will need to add the Confluence [Web Item module](#) that defines the location of your custom action in the Confluence interface. In this case, we're adding this action to the 'primary' section of the 'Tools' menu on a Confluence page or blog post).

For more information, please refer to the [diagram](#) on the [Web UI Module](#) page.

The Web Item module also defines the link for the custom action and its syntax. The link itself triggers the action, which is defined in a separate XWork-Webwork module.

```
<web-item name="add-draft-label-action-web-ui" key="add-draft-label-action-web-ui" section="system.content.action/primary" weight="10">
    <description key="item.add-draft-label-action-web-ui.link.desc">Adds the "Add 'draft' label" action into the Tools menu.</description>
    <label key="Add 'draft' label"/>
    <link
        linkId="add-draft-label-action">/plugins/add-draft-label/add-label.action?pageId=$page.id</link>
</web-item>
```

Add the XWork-Webwork Module

Finally, we need to define the XWork-Webwork module. This module defines:

- Each segment of the link in the Web Item module definition (above), including the link's 'action' component.
 - What Java class (or classes) are used to encapsulate the logic and functionality that this action performs.
- i** Note from the XML code below that you will only be using one Java class, called AddDraftLabelAction.
- The 'result' of the action, that is, what happens after an object of the Java class has executed successfully. In this case, if the class executes successfully, reload the current page.

```
<xwork name="add-draft-label-action-xwork" key="add-draft-label-action-xwork">
    <description key="item.add-draft-label-action-xwork.link.desc">Defines what the "Add
'draft' label" action does.</description>
    <package name="add-draft-label" extends="default" namespace="/plugins/add-draft-label">
        <default-interceptor-ref name="defaultStack"/>
        <action name="add-label" class="com.atlassian.plugins.tutorial.AddDraftLabelAction">
            <result name="success" type="redirect">${page.urlPath}</result>
        </action>
    </package>
</xwork>
```

Now you are ready to move onto writing some Java code that makes your plugin 'do' the action.

Step 4. Write your Java Class

Next, we will need to encapsulate what these modules do in Confluence, inside a Java class called AddDraftLabelAction.

Set up your Java AddDraftLabelAction class which extends the ConfluenceActionSupport class

Modify the ExampleMacro.java archetype to the name AddDraftLabelAction, which will extend the ConfluenceConfluenceActionSupport class.



If you're using a sophisticated IDE like Eclipse or IDEA, renaming or 'refactoring' the class name, should also change the name of your Java class file from ExampleMacro.java to AddDraftLabelAction.java. Your Java class name must match its file name.

Your AddDraftLabelAction class will also need to:

- Implement the Confluence PageAware interface, and
- Inject the LabelManager dependency.

Some sophisticated IDEs will allow you to automatically implement the following methods associated with the PageAware interface:

- AbstractPage
- setPage
- isPageRequired
- isLatestVersionRequired
- isViewPermissionRequired

However, you will need to modify these methods so that the current page is returned. Once you have done this, here is what your Java code should look like so far:

```
package com.atlassian.plugins.tutorial;

import com.atlassian.confluence.core.ConfluenceActionSupport;
import com.atlassian.confluence.labels.LabelManager;
import com.atlassian.confluence.pages.AbstractPage;
import com.atlassian.confluence.pages.actions.PageAware;

/**
 * This Confluence action adds the label 'draft' to the page or blog post when a user selects it
 * from the
 * 'Tools' menu in Confluence. Refer to the 'atlassian-plugin.xml' file for details on how this
 * action is
 * implemented in Confluence.
 */
public class AddDraftLabelAction extends ConfluenceActionSupport implements PageAware
{
    private AbstractPage page;
    private LabelManager labelManager;

    /**
     * Implementation of PageAware
     */
    public AbstractPage getPage()
    {
        return page;
    }

    /**
     * Implementation of PageAware
     */
    public void setPage(AbstractPage page)
    {
        this.page = page;
    }

    /**
     * Implementation of PageAware:
     * Returning 'true' ensures that the
     * page is set before the action commences.
     */
    public boolean isPageRequired()
    {
        return true;
    }

    /**
     * Implementation of PageAware:
     * Returning 'true' ensures that the
     * current version of the page is used.
     */
    public boolean isLatestVersionRequired()
    {
        return true;
    }

    /**
     * Implementation of PageAware:
     * Returning 'true' ensures that the user
     * requires page view permissions.
     */
    public boolean isViewPermissionRequired()
    {
        return true;
    }

    /**
     * Dependency-injection of the Confluence LabelManager.
     */
    public void setLabelManager(LabelManager labelManager)
    {
        this.labelManager = labelManager;
    }
}
```

Add the execute method to your AddDraftLabelAction class

Finally, you will need to add the crucial piece of logic to your custom action. This adds the label 'draft' to your Confluence page or blog post. To do this, add the following code to your `AddDraftLabelAction` class.

```
public String execute()
{
    // page is already retrieved by Confluence's PageAwareInterceptor
    // labelManager is injected by Confluence -- see setLabelManager() below
    Label label = new Label("draft");
    labelManager.addLabel(page, label);
    return "success";
}
```

Note that this method must return the string "success", which is interpreted by the `result` element defined in your XWork-Webwork module (above).



You will also need to add the following `import` statement near the top of your Java class file, which many sophisticated IDEs should do for you:

```
import com.atlassian.confluence.labels.Label;
```

Hence, your `AddDraftLabelAction.java` file should look like the following code, once it has been refactored in your IDE.

```
package com.atlassian.plugins.tutorial;

import com.atlassian.confluence.core.ConfluenceActionSupport;
import com.atlassian.confluence.labels.Label;
import com.atlassian.confluence.labels.LabelManager;
import com.atlassian.confluence.pages.AbstractPage;
import com.atlassian.confluence.pages.actions.PageAware;

/**
 * This Confluence action adds the label 'draft' to the page or blog post when a user selects it
 * from the
 * 'Tools' menu in Confluence. Refer to the 'atlassian-plugin.xml' file for details on how this
 * action is
 * implemented in Confluence.
 */
public class AddDraftLabelAction extends ConfluenceActionSupport implements PageAware
{
    private AbstractPage page;
    private LabelManager labelManager;

    /**
     * Implementation of PageAware
     */
    public AbstractPage getPage()
    {
        return page;
    }

    /**
     * Implementation of PageAware
     */
    public void setPage(AbstractPage page)
    {
        this.page = page;
    }

    /**
     * Implementation of PageAware:
     * Returning 'true' ensures that the
     * page is set before the action commences.
     */
    public boolean isPageRequired()
    {
        return true;
    }
}
```

```
/**  
 * Implementation of PageAware:  
 * Returning 'true' ensures that the  
 * current version of the page is used.  
 */  
public boolean isLatestVersionRequired()  
{  
    return true;  
}  
  
/**  
 * Implementation of PageAware:  
 * Returning 'true' ensures that the user  
 * requires page view permissions.  
 */  
public boolean isViewPermissionRequired()  
{  
    return true;  
}  
  
/**  
 * Dependency-injection of the Confluence LabelManager.  
 */  
public void setLabelManager(LabelManager labelManager)  
{  
    this.labelManager = labelManager;  
}  
  
public String execute()  
{  
    // page is already retrieved by Confluence's PageAwareInterceptor  
    // labelManager is injected by Confluence -- see setLabelManager() below  
    Label label = new Label("draft");  
    labelManager.addLabel(page, label);
```

```

        return "success";
    }
}

```

Step 5. Build, Install and Run the Plugin

Run the following commands to build and install your plugin, so that you can test your code.

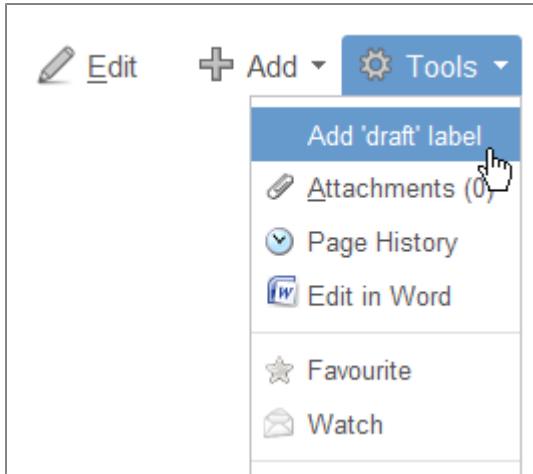
- Open a command window and go to the plugin root folder (where the `pom.xml` is located).
- If you have not already started the application, do that now by running `atlas-run`.
- When prompted, open up a new browser window and enter the URL, which is typically:
`http://localhost:1990/confluence`
- Log in to Confluence with the username of `admin` and password of `admin` too.
- Once the Confluence dashboard has loaded, check that the plugin has been successfully installed in Confluence. To do this:
 1. Visit the Confluence Administration console, by clicking '**Browse**' -> '**Confluence Admin**' and choose '**Plugins**' in the left navigation panel.
 2. Ensure that the '**Add 'draft' label**' plugin is visible and selected. If it has installed successfully, your Confluence Administration console should look similar to the screenshot below.

Screenshot: Confluence Administration console

| Plugins | Add 'draft' label | | | | |
|--|---|-------------------------------|-------------------------|------------------------------|-------------------------|
| Add 'draft' label All modules enabled. | Vendor: Atlassian Plugin Version: 1.0-SNAPSHOT <p>This plugin adds the label 'draft' to the page, via an "Add 'draft' label" action from the Tools menu.</p> <p><input type="checkbox"/> Disable plugin</p> <table border="1"> <tr> <td>add-draft-label-action-web-ui</td> <td>Disable</td> </tr> <tr> <td>add-draft-label-action-xwork</td> <td>Disable</td> </tr> </table> | add-draft-label-action-web-ui | Disable | add-draft-label-action-xwork | Disable |
| add-draft-label-action-web-ui | Disable | | | | |
| add-draft-label-action-xwork | Disable | | | | |
| Advanced Macros All modules enabled. | | | | | |
| Apache Felix Web Management Console No modules enabled. | | | | | |
| Atlassian UI Plugin All modules enabled. | | | | | |
| Attachment Extractors All modules enabled. | | | | | |
| Basic Macros All modules enabled. | | | | | |

- Visit any Confluence page and choose '**Add 'draft' label**' from the '**Tools**' menu.

Screenshot: Confluence 'Tools' menu with your new "Add 'draft' label" menu item



The label 'draft' will be added to your Confluence page:

Screenshot: Draft label added to your page

A screenshot of the Confluence 'Labels' page. It shows a list of labels: 'homepage', 'start here', and 'draft'. The 'draft' label is underlined, indicating it has been recently added.

Congratulations, that's it
Have a chocolate!

Step 6. Modifying your source code from here

- From that point onwards, you can use the command line interface – run `atlas-cli` in another command window to start the CLI.
- When you see the message, `Waiting for commands`, you can run `pi` to compile, package and install the updated plugin.
- The full instructions are in the [SDK guide](#).

Plugin Tutorial - Adding a JQL Function to JIRA

On this page:

- Overview
 - Structure of a JQL Query
 - Structure of a Clause
 - Functions as Operands
 - Roll your Own
- Step 1. Create the Plugin Project
- Step 2. Register the Plugin Module in the Plugin Descriptor
- Step 3. Write Java Classes
- Step 4. Build, Install and Run the Plugin
- Using your New Function

Overview

This tutorial shows you how to add a JQL function to JIRA. We assume that you are already familiar with JQL Advanced Searching. Please also take a look at the list of [functions shipped with JIRA](#).

You will create a JQL function plugin. As with all plugins, your plugin will consist of the following components:

- Java classes encapsulating the plugin logic
- Plugin descriptor to enable the plugin module in JIRA

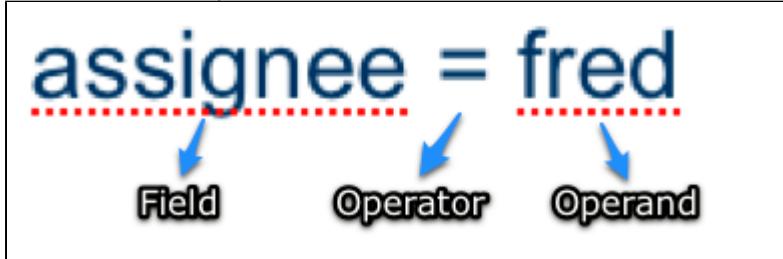
All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Structure of a JQL Query

A JQL query is made of multiple clauses joined together (AND/OR) to form more complex queries.

Structure of a Clause

A clause consists of 3 parts:



- **Field** – The field on which to filter, such as assignee, reporter, create date, status.
- **Operator** – The kind of filter we want to apply, such as =, !=, >, ~, in, not in.
- **Operand** – The values to restrict the filter by, such as psmith, 10/02/09, Open, now(), currentUser(), memberOf("jira-dev").

Functions as Operands

As you can see, operands can be literals (like a username), and can also be functions that return:

- Multiple values that can work with IN and NOT IN.
- Single values that work with equals, not equals, greater than, less than, and so on.

These functions are actually plugin points that offer a powerful way to add functionality to the JIRA search.

Take a look at the [up-to-date list of functions that ship with JIRA](#). The following functions shipped with JIRA 4.0:

- `cascadeOption()` – Allows searching of both values or a single value of a cascade.
- `currentUser()` – Returns the current user.
- `issueHistory()` – Returns all issue keys of previously looked at issues (max 50).
- `linkedIssues()` – Returns all keys of linked issues.
- `membersOf()` – Returns all members of a group.
- `now()` – Returns the current time.
- `releasedVersions()` – Returns all released versions.
- `standardIssueTypes()` – Returns all standard (non-subtask) issue types.
- `subtaskIssueTypes()` – Returns all subtask issue types.
- `unreleasedVersions()` – Returns all unreleased versions.
- `votedIssues()` – Returns issue keys for all issues the current user has voted for.
- `watchedIssues()` – Returns issue keys for all issues the current user is watching.

Roll your Own

We will now show you what is needed to write a very basic function. You will create a function that enables you to restrict the search to projects that you have recently visited. This would be useful in systems that have a number of projects, where you only care about a few.



See the detailed guide to the JQL function plugin module, covering more advanced topics such as parameter passing and operand sanitisation.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

Step 2. Register the Plugin Module in the Plugin Descriptor

Now you need to edit the plugin descriptor at `src/main/resources/atlassian-plugin.xml` to register the plugin module. Add the following plugin module to the XML file:

```
<jql-function key="recent-projects" name="Recent Project Function"
  class="com.atlassian.example.jira.jqlfunc.RecentProjectFunction">
  <!--The name of the function-->
  <fname>recentProjects</fname>
  <description>Provides a JQL function to restrict the search to recently visited
  projects.</description>
  <!--Whether this function returns a list or a single value-->
  <list>true</list>
</jql-function>
```

Step 3. Write Java Classes

Now you are ready to move onto writing some code to make your plugin do something. Stub out the class `RecentProjectFunction`. The easiest way to get started is extend `AbstractJqlFunction`, which looks after a lot of the basic functionality such as function name, initialisation and basic validation.

```
package com.atlassian.jira.plugin.jql.function;

import com.atlassian.jira.JiraDataType;
import com.atlassian.jira.JiraDataTypes;
import com.atlassian.jira.jql.operand.QueryLiteral;
import com.atlassian.jira.jql.query.QueryCreationContext;
import com.atlassian.jira.util.MessageSet;
import com.atlassian.jira.util.NotNull;
import com.atlassian.query.clause.TerminalClause;
import com.atlassian.query.operand.FunctionOperand;
import com.opensymphony.user.User;

import java.util.List;

/**
 * A handler for the "recentHistory" function. This function will return all the projects within
the user's history.
 */
public class RecentProjectFunction extends AbstractJqlFunction
{
    public List<QueryLiteral> getValues(@NotNull QueryCreationContext queryCreationContext, @NotNull
FunctionOperand operand, @NotNull TerminalClause terminalClause)
    {
        return null;
    }

    /**
     * This method validates the passed in args. In this case the function accepts no args, so let's
validate that were none.
     */
    public MessageSet validate(User searcher, @NotNull FunctionOperand operand, @NotNull
TerminalClause terminalClause)
    {
        return validateNumberOfArgs(operand, 0);
    }

    /**
     * This method returns the min number of args the function takes. In this case - 0
     */
    public int getMinimumNumberOfExpectedArguments()
    {
        return 0;
    }

    /**
     * This method needs to return the type of objects the function deals with. In this case -
Projects
     */
    public JiraDataType getDataType()
    {
        return JiraDataTypes.PROJECT;
    }
}
```

Some of these methods are easy to complete:

- `getMinimumNumberOfExpectedArguments()` – This function takes no parameters so we just return 0.
- `getDataType()` – This function only deals with Projects so we return `JiraDataTypes.PROJECT`.
- `validate(...)` – Since our function takes no args, the only validation we need to do is validate that 0 args were passed in.

`getValues(...)`

In here we need to return a list of `QueryLiterals` that represent the list of IDs of projects recently visited. Function classes are injectable, so we just need to inject the `UserProjectHistoryManager` and get the list of project IDs from that, then convert them to `QueryLiterals`:

```

/**
 * A handler for the "recentHistory" function. This function will return all the projects within
the user's history.
*
* @since v4.0
*/
public class RecentProjectFunction extends AbstractJqlFunction
{
    private static final Logger log = Logger.getLogger(RecentProjectFunction.class);

    private final UserProjectHistoryManager userProjectHistoryManager;

    public RecentProjectFunction(UserProjectHistoryManager userProjectHistoryManager)
    {
        this.userProjectHistoryManager = userProjectHistoryManager;
    }

    public List<QueryLiteral> getValues(@NotNull QueryCreationContext queryCreationContext, @NotNull
FunctionOperand operand, @NotNull TerminalClause terminalClause)
    {
        final List<QueryLiteral> literals = new LinkedList<QueryLiteral>();

        /*
         * We do not need to do a security check here as the search will do the security checks for us.
         */
        final List<UserHistoryItem> projects =
userProjectHistoryManager.getProjectHistoryWithoutPermissionChecks(queryCreationContext.getUser());
for (final UserHistoryItem userHistoryItem : projects)
{
    final String value = userHistoryItem.getEntityId();

    try
    {
        literals.add(new QueryLiteral(operand, Long.parseLong(value)));
    }
    catch (NumberFormatException e)
    {
        log.warn(String.format("User history returned a non numeric project ID '%s'.", value));
    }
}

        return literals;
    }

    ...
}

```

Step 4. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Using your New Function

To use this function in a query:

```
project in recentProjects()
```



The JQL autocomplete should now offer your new function in its project completion suggestions.

RELATED TOPICS

[Advanced Searching](#)
[Advanced Plugin Development](#)
[JQL Function Plugin Module](#)
[JIRA 4.0 Release Notes](#)

Plugin Tutorial - Adding your own Menu Items to Confluence



Level of experience: Beginner

This tutorial's level of difficulty is at 'beginner' level, so you can follow it even if you have never developed a plugin before. Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'.

Source Code

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- [Source Code](#)
- [Overview](#)
- [Step 1. Create the Plugin Project](#)
- [Step 2. Edit the atlassian-plugin.xml file](#)
 - Choosing a menu location for the new item
 - Choosing a graphic to appear in the menu
 - Controlling when the web item is displayed
 - Defining a custom label to appear in the Menu UI
 - Adding new resource files and internationalisation
- [Step 4. Build, Install and Run the Plugin](#)

Overview

This tutorial shows you how to use a Web UI module in your plugin, to insert a menu option into the Confluence web interface. The Web UI module can also insert links, tabs and sections of links.

In order to do this, you will create a very simple Confluence plugin. Your plugin will consist of the following components:

- Resources for display of the plugin UI
- Plugin descriptor to enable the plugin module in Confluence.

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.



A more sophisticated plugin would also contain Java classes encapsulating the plugin logic. In this case however, we are simply concentrating on only adding a Web UI Module to Confluence — we'll put an existing Confluence action in place for the button to activate.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- [Set up your development environment](#)
- [Create and install your plugin from a template](#)

You'll be prompted to enter two values, the **group ID** and **artifact ID**. For this tutorial, we used the following values:

- Group ID: **com.atlassian.tutorial**
- Artifact ID: **webitemz**

Step 2. Edit the atlassian-plugin.xml file

In this step, you must now register the plugin module in your plugin descriptor, `atlassian-plugin.xml`. In other words, add the following code to your `atlassian-plugin.xml` file between the `<atlassian-plugin>` tags, but below the `<plugin-info>` tag group.

```

<web-item key="spacelogo" name="Space Logo" section="system.content.add/space" weight="40">
    <label key="webitemz.new.item" />
    <link>/spaces/configurespacelogo.action?key=$space.key</link>
    <icon height="16" width="16">
        <link>/images/icons/logo_add_16.gif</link>
    </icon>
    <condition
class="com.atlassian.confluence.plugin.descriptor.web.conditions.NotPersonalSpaceCondition"/>
</web-item>

```

Let's break down that XML code. In this example we are adding an item to the **Tools** menu in Confluence, which will open the **Customise Space Logo** page when it's clicked.

To do this, we've done a number of things.

Choosing a menu location for the new item

In the code above, this line involves four attributes:

```
<web-item key="spacelogo" name="Space Logo" section="system.content.add/space" weight="40">
```

- The first, **key="spacelogo"** is setting an internal name for the new item.
- The second, **name="Space Logo"** is another internal name for the new item.
- The third, **section="system.content.add/space"** defines the menu location for the item, in the top part of the **Add Menu**. See [more information](#).
- The fourth, **weight** is a variable for controlling presentation, which you can leave at the default value.

Choosing a graphic to appear in the menu

In the code above, this part:

```

<icon height="16" width="16">
    <link>/images/icons/logo_add_16.gif</link>
</icon>

```

Defines a small graphic icon that will appear in the menu. **Height** and **width** are for controlling presentation of the graphic, while **link** points to the location of the graphic file (in this case we are using a graphic that ships with Confluence).

Controlling when the web item is displayed

Examining this line of code:

```

<condition
class="com.atlassian.confluence.plugin.descriptor.web.conditions.NotPersonalSpaceCondition"/>

```

This causes the web item to only be displayed in spaces that are not personal spaces. It does this by using a '**web condition**'. Web conditions control the display of web items; more information is available in the [reference documentation](#) for Confluence plugin development.

Defining a custom label to appear in the Menu UI

In the code above, this line: `<label key="webitemz.new.item" />` defines the label for the button text that will appear in the menu.

Adding new resource files and internationalisation

We will want to specify a text label to display in our Confluence menu item. You could hard-code this information into your `atlassian-plugin.xml` file, but by adding it in a new resource file, we can make our plugin compatible with internationalisation. To do so, simply add a new file in the resources directory of your plugin (called `message.properties`) and enter one line of code into it:

```
webitemz.new.item=New Option!
```

Additionally, we will need to reference this resource file in our `atlassian-plugin.xml` file. To do this, add this line of code above the `<web item>` code block:

```
<resource type="i18n" name="i18n" location="message" />
```

This will access your `message.properties` file and retrieve the text for our button label. The result of this is that the text we have specified here; 'New Option!' appears as the title text of our new button in the **Add** menu.

If you would like to know more about internationalisation, see our [Confluence documentation on the topic](#).

Step 4. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

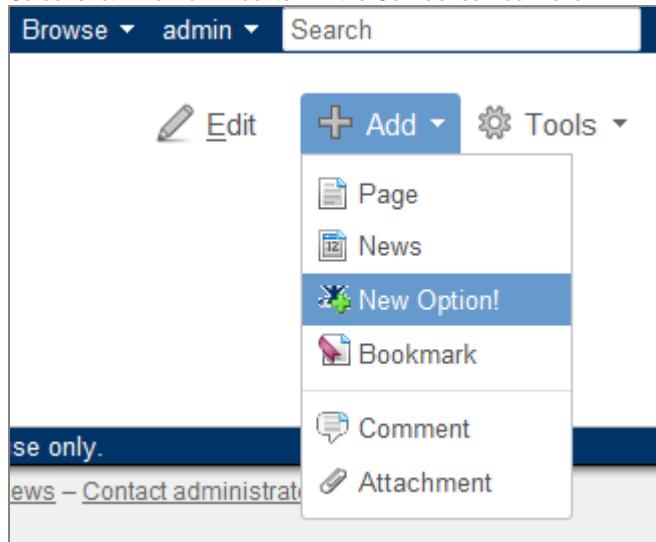
The **atlas-run** command will compile the plugin project and then launch a local instance of Confluence.

Once Confluence has loaded, access the local instance with this URL:

`http://localhost:1990/confluence/`

Login with username "admin" and password "admin", then create a page in the default space. On accessing the '**Add Menu**', you will be able to see the new web item that you've added inside the menu. Clicking it will launch the '**Customise Space Logo**' screen.

Screenshot: The New Web Item in the Confluence Add Menu



That's all that's required to add a menu item in a Confluence plugin. To extend this project, you could build new functionality for your button to activate.

Now you are ready to move onto writing some Java code to achieve that. It's not necessary, for the purposes of this tutorial though.

If you wanted to move on to writing Java to make new functionality to go with your web item, you can create an action with the [XWork-WebWork Module](#).



Congratulations, you have completed this tutorial.

Plugin Tutorial - Adding your own Menu Items to JIRA



Level of experience: Beginner

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'beginner' level, so you can follow it even if you have never developed a plugin before.

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).

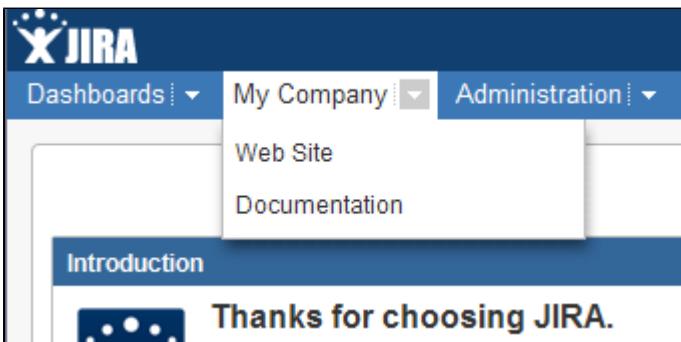
On this page:

- Overview
- Step 1. Create the Plugin Project
- Step 2. Add Plugin Metadata to the POM
- Step 3. Register the Plugin Modules in the Plugin Descriptor
- Step 4. Build, Install and Run the Plugin
- Installing the Plugin into your Company JIRA Site

Overview

This tutorial shows you how to write a simple plugin that adds a new tab (section) to JIRA's top-level menu, with a number of menu items in the section. The menu items link to specific web pages.

Screenshot: What your new 'My Company' menu item will look like



You can use this sort of plugin to add links from JIRA to other locations in your environment, such as your company's intranet or your Confluence site.

In order to do this, you will create a JIRA plugin consisting of the following components:

- A plugin descriptor to enable the plugin module in JIRA.
- The required plugin modules that define the new menu section and menu items.

i Your plugin does not need any Java code, because the plugin modules provided by the Atlassian Plugin Framework provide all the functionality required in this simple plugin.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

1. Follow the instructions in the above SDK documents to set up your plugin development environment.
2. Follow the instructions in the above SDK documents to run `atlas-create-jira-plugin`. This will create your skeleton plugin.

You will be prompted for some information to identify your plugin. Enter the following information:

- **group-id:** com.atlassian.plugins.tutorial
- **artifact-id:** jira-menu-items
- **version:** 1.0
- **package:** com.atlassian.plugins.tutorial

3. Follow the instructions in the above SDK documents to run `atlas-run`.
4. When the script finishes, open your new local instance of JIRA in your browser at <http://localhost:2990/jira>.
5. Your plugin skeleton is already installed and running in JIRA. Take a look at it now, by following these steps:
 - Click the downward-pointing arrow next to 'Administration' in JIRA's top menu bar.
 - Select 'Plugins' from the dropdown list.
 - The 'Current Plugins' page opens. Search for 'jira-menu-items' and click the link.
 - The details of your basic plugin appear, looking like this:

Screenshot: New plugin skeleton in JIRA

The screenshot shows the JIRA Administration interface with the 'Current Plugins' page selected. The left sidebar contains navigation links for Project, Users, Groups & Roles, and Global Settings. The main content area displays a list of installed plugins under 'Installed Plugins'. One plugin, 'jira-menu-items', is highlighted. Its details are shown in a right-hand panel: Description (This is the com.atlassian.plugins.tutorial:jira-menu-items plugin for Atlassian JIRA), Vendor (Example Company), Plugin Version (1.0), JIRA Version (0.0), Installation Mode (Managed), and a 'Disable plugin' checkbox. A search bar at the bottom is set to 'jira-n'.

Step 2. Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
<organization>
    <name>Example Company</name>
    <url>http://www.example.com/</url>
</organization>
```

3. Update the `<description>` element:

```
<description>This plugin adds a new section and items to JIRA's menu</description>
```

4. Save the file.

Step 3. Register the Plugin Modules in the Plugin Descriptor

Next you will add the required plugin modules to your plugin descriptor at `src/main/resources/atlassian-plugin.xml`. The plugin descriptor is an XML file that identifies the plugin to JIRA and defines the functionality that the plugin requires.

Here's a basic plugin descriptor:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}">
    <plugins-version>2</plugins-version>
    <plugin-info>
        <description>${project.description}</description>
        <version>${project.version}</version>
        <vendor name="${project.organization.name}" url="${project.organization.url}" />
    </plugin-info>
</atlassian-plugin>
```

Note that some of the information from the POM is transferred to the plugin descriptor using variable names such as \${project.artifactId}.

You will need the following plugin modules:

- A [web section](#) to define the new tab (section) in your JIRA top navigation bar.
- A number of [web items](#) to add the links and menu items to the new section.

Follow these instructions to add the new section and menu items:

1. Edit your plugin descriptor at `src/main/resources/atlassian-plugin.xml`.
2. Add the following web section into your plugin descriptor. This defines the top-level section where we will put all our menu items and links:

```
<web-section key="my_links_section" name="My Links Main Section" location="my_links_link"
    weight="10"/>
```

 The `weight` attribute determines the order in which web items appear. Items are displayed top to bottom or left to right in order of ascending weight. The 'lightest' weight is displayed first, the 'heaviest' weights sink to the bottom. The weights for most applications' system sections start from 100, and the weights for the links generally start from 10. The weight is incremented by 10 for each in sequence so that there is ample space to insert your own sections and links.

3. Add the following web item. This defines the link on the top-level section, with text 'My Company'. We have pointed the link at the company website:

```
<web-item key="my_links_link" name="Link on My Links Main Section"
    section="system.top.navigation.bar" weight="47">
    <label>My Company</label>
    <link linkId="my_links_link">http://www.atlassian.com</link>
</web-item>
```

 The `linkId` is optional, and provides an XML id for the link being generated.

4. Add another web item, to create a single menu item with text 'Web Site' that links to your company website.

```
<web-item key="website_link" name="Company Web Site"
    section="my_links_link/my_links_section" weight="10">
    <label>Web Site</label>
    <link linkId="website_link">http://www.atlassian.com</link>
</web-item>
```

5. Add another web item, to create a menu item with text 'Documentation' that links to your documentation website.

```
<web-item key="documentation_link" name="Documentation Web Site"
    section="my_links_link/my_links_section" weight="10">
    <label>Documentation</label>
    <link linkId="documentation_link">http://confluence.atlassian.com</link>
</web-item>
```

6. Save the file.

Here's what your `atlassian-plugin.xml` file looks like now:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}" plugins-version="2">
    <plugin-info>
        <description>${project.description}</description>
        <version>${project.version}</version>
        <vendor name="${project.organization.name}" url="${project.organization.url}" />
    </plugin-info>

    <web-section key="my_links_section" name="My Links Main Section" location="my_links_link" weight="10"/>

        <web-item key="my_links_link" name="Link on My Links Main Section" section="system.top.navigation.bar" weight="47">
            <label>My Company</label>
            <link linkId="my_links_link">http://www.atlassian.com</link>
        </web-item>

        <web-item key="website_link" name="Company Web Site" section="my_links_link/my_links_section" weight="10">
            <label>Web Site</label>
            <link linkId="website_link">http://www.atlassian.com</link>
        </web-item>

        <web-item key="documentation_link" name="Documentation Web Site" section="my_links_link/my_links_section" weight="10">
            <label>Documentation</label>
            <link linkId="documentation_link">http://confluence.atlassian.com</link>
        </web-item>
    </atlassian-plugin>
```

Step 4. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Here's a screenshot showing the new '**My Company**' section in JIRA's top menu bar. The dropdown menu shows the two items expected, '**Web Site**' and '**Documentation**'.

The screenshot also shows the plugin as listed in the 'Plugins' section of JIRA's Administration Console.

[Screenshot: New menu items in JIRA](#)

The screenshot shows the JIRA Administration interface with the 'View Plugins' page selected. The left sidebar shows various administration categories like Project, Users, and Global Settings. The main content area displays a list of installed plugins, with 'jira-menu-items' highlighted. This plugin's details are shown on the right, including its description, vendor, version, and installation mode. A table lists its modules and their disable links.

| Module | Action |
|---|--------------------------------|
| My Links Main Section (my_links_section) | Disable module |
| Link on My Links Main Section (my_links_link) | Disable module |
| Company Web Site (website_link) | Disable module |
| Documentation Web Site (documentation_link) | Disable module |

Congratulations, that's it
Have a chocolate!

Installing the Plugin into your Company JIRA Site

When you have finished developing and testing your plugin, you can install it into your company's JIRA instance, so that the new menu items are available for other people to use.



Make sure your plugin is fully tested

The above is a basic tutorial on getting your plugin working in your local development version of JIRA. Please make sure that you have tested the plugin fully before installing it onto your production JIRA site.

Your plugin JAR file is at `target\jira-menu-items-1.0.jar`. You can install this plugin JAR into a JIRA site by following the instructions in the [JIRA Administrator's Guide](#). On that page, follow the instructions for a 'Version 2' (OSGi) plugin, because that is the plugin type that you have created by using the SDK.

RELATED TOPICS

[Web Fragments](#)
[Web Section Plugin Module](#)
[Web Item Plugin Module](#)

Plugin Tutorial - Creating a JIRA Report



Level of experience: Advanced

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'advanced' level. If you have never developed a plugin before, we advise you to try a beginner tutorial first.

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- Overview
- Getting Started
 - Step 1. Create the Plugin Project
 - Step 2. Add Plugin Metadata to the POM
- Example 1 - Extending an Existing System Report
 - Step 1. Register the first Plugin Module in the Plugin Descriptor
 - Step 2. Extend the existing Java class
 - Step 3. Extend the existing view template
 - Step 4. Build, Install and Run the Plugin
- Example 2 - Issue Creation Report
 - Step 1. Register the Plugin Module in the Plugin Descriptor
 - Step 2. Add the Java class for the report calculation
 - Step 3. Add the view template
 - Step 4. Add i18n messages
 - Step 5. Build, Install and Run the Plugin

Overview

This tutorial shows you how to extend an existing JIRA system report in the first example and then how to create your own JIRA custom report from scratch in the second example.

In order to do this, you will create a JIRA plugin. As with all plugins, your plugin will consist of the following components:

- Java classes encapsulating the plugin logic
- Resources for display of the plugin UI
- Plugin descriptor to enable the plugin module in JIRA

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Getting Started

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

When you create the plugin using the above SDK, you will be prompted for some information to identify your plugin. Enter the following information:

- **group-id**: com.atlassian.plugins.tutorial
- **artifact-id**: jira-report-plugin
- **version**: 1.0
- **package**: com.atlassian.plugins.tutorial.jira.report

Step 2. Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
/pom.xml

<organization>
    <name>Example Company</name>
    <url>http://www.example.com/</url>
</organization>
```

3. Update the <description> element:

```
/pom.xml  
<description>This plugin provides additional JIRA reports</description>
```

4. Save the file.

Example 1 - Extending an Existing System Report

In this example, an existing system report - 'Single Level Group By Report' - is extended to display the assignee and the last updated time of the issue. The logic and view templates from the JIRA source code are slightly modified in order to achieve this extension.

Before:



The screenshot shows a JIRA report titled "Report: Single Level Group By Report Extended". The filter is set to "All Issues". A user named "admin" is logged in. The report lists two issues: "TST-1 UNRESOLVED Test" and "TST-2 UNRESOLVED Foo". In the top right corner, there is a progress bar labeled "Progress: 0 of 2 issues have been resolved".

After:



The screenshot shows the same JIRA report after modifications. The list of issues remains the same, but the last updated information is now displayed next to each issue. For "TST-1 UNRESOLVED Test", it says "admin Yesterday 3:59 PM". For "TST-2 UNRESOLVED Foo", it says "admin Yesterday 4:00 PM".

For this example you are not going to create everything from scratch as we focus on chaning the output of an existing report. You can just checkout the source code from [SVN](#) or create the following files and copy the contents from the provided links:

- src/main/resources/templates/groupreport/single-groupby-reportextended.vm
- src/main/resources/com/atlassian/plugins/tutorial/jira/report/singlelevelgroup_report.properties
- src/main/java/com/atlassian/plugins/tutorial/jira/report/example/singlelevelgroup/SingleLevelGroupByReportExtended.java

Step 1. Register the first Plugin Module in the Plugin Descriptor

Now you will add the required plugin modules to your plugin descriptor at `src/main/resources/atlassian-plugin.xml`. The plugin descriptor is an XML file that identifies the plugin to JIRA and defines the functionality that the plugin requires.

Here's a basic plugin descriptor:

```
/atlassian-plugin.xml  
  
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}"  
plugins-version="2">  
    <plugin-info>  
        <description>${project.description}</description>  
        <version>${project.version}</version>  
        <vendor name="${project.organization.name}" url="${project.organization.url}" />  
    </plugin-info>  
</atlassian-plugin>
```

Note that some of the information from the POM is transferred to the plugin descriptor using variable names such as `${project.artifactId}`.

You will need to add the following plugin module:

/atlassian-plugin.xml

```
<report key="singlelevelgroupbyextended" name="Example: Group By Report Extended"
class="com.atlassian.plugins.tutorial.jira.report.SingleLevelGroupByReportExtended">
    <description key="report.singlelevelgroupby.description">i18n description</description>
    <resource type="velocity" name="view"
location="templates/groupreport/single-groupby-report-extended.vm" />
    <resource type="i18n" name="i18n"
location="com.atlassian.plugins.tutorial.jira.report.singlelevelgroup_report" />
    <label key="report.singlelevelgroupby.label.extended" />
<properties>
    <property>
        <key>filterid</key>
        <name>report.singlelevelgroupby.filterId</name>
        <description>report.singlelevelgroupby.filterId.description</description>
        <type>select</type>
        <values class="com.atlassian.jira.portal.SearchRequestValuesGenerator" />
    </property>
    <property>
        <key>mapper</key>
        <name>report.singlelevelgroupby.mapper</name>
        <description>report.singlelevelgroupby.mapper.description</description>
        <type>select</type>
        <values class="com.atlassian.jira.issue.statistics.FilterStatisticsValuesGenerator" />
    </property>
</properties>
</report>
```

This plugin module was just copied from JIRA as we are only going to modify the output of the report in this example. A complete explanation of the plugin module can be found in our plugin module documentation.

Now you are ready to move onto writing some code to make your report plugin output the assignee and the last updated time of the issue.

Step 2. Extend the existing Java class

The logic for the report is encapsulated in the class `SingleLevelGroupByReportExtended` (a slightly modified version of the original `SingleLevelGroupByReport` class within JIRA). The view template requires a manager to correctly display the last updated time for the issue - so we need pass the `OutLookDateManager`'s `OutlookDate` object to the velocity template by adding it to the parameter map:

/src/main/java/com/atlassian/plugins/tutorial/jira/report/SingleLevelGroupByReportExtended.java

```
try
{
    startingParams = EasyMap.build(
        "action", action,
        "statsGroup", getOptions(request, authenticationContext.getUser(), mapper),
        "searchRequest", request,
        "mapperType", mapperName,
        "customFieldManager", customFieldManager,
        "fieldVisibility", new FieldVisibilityBean(),
        "searchService", searchService,
        "portlet", this);

    // Add the OutlookDate instance to the parameter map for Velocity
    startingParams.put("outlookDate",
        outlookDateManager.getOutlookDate(authenticationContext.getLocale()));

    return descriptor.getHtml("view", startingParams);
}
```

We also had to change the constructor of `SingleLevelGroupByReportExtended` to get an `OutLookDateManager` injected.

The last updated time can now appear in the correct date/time format as configured within JIRA.

Step 3. Extend the existing view template

The report view template is also edited from the original to display the assignee and the last updated time for the issue. The modifications were made in two places in the template where we loop over the issue result set:

/src/main/resources/templates/groupreport/single-groupby-report-extended.vm

```
#foreach ($issue in $issues)
    <tr>
        <td width="5%">&nbsp;</td>
        #issueLineItem ($issue)
        <td nowrap>
            #if($issue.getAssignee())
                $issue.getAssignee().getFullName()
            #else
                $i18n.getText('common.concepts.unassigned')
            #end</td>
        <td nowrap>$outlookDate.format($issue.getUpdated())</td>
    </tr>
#end
```

You also need to adjust the colspan for the various cells to account for the two additional columns.

Step 4. Build, Install and Run the Plugin

Run the following commands to build and install your plugin, so that you can test your code.

- Open a command window and go to the plugin root folder (where the pom.xml is located).
- If you have not already started the application, do that now by running atlas-run.

Wait until JIRA is started up and then go back to the browser. The plugin has been installed into the application, and you can test your changes.

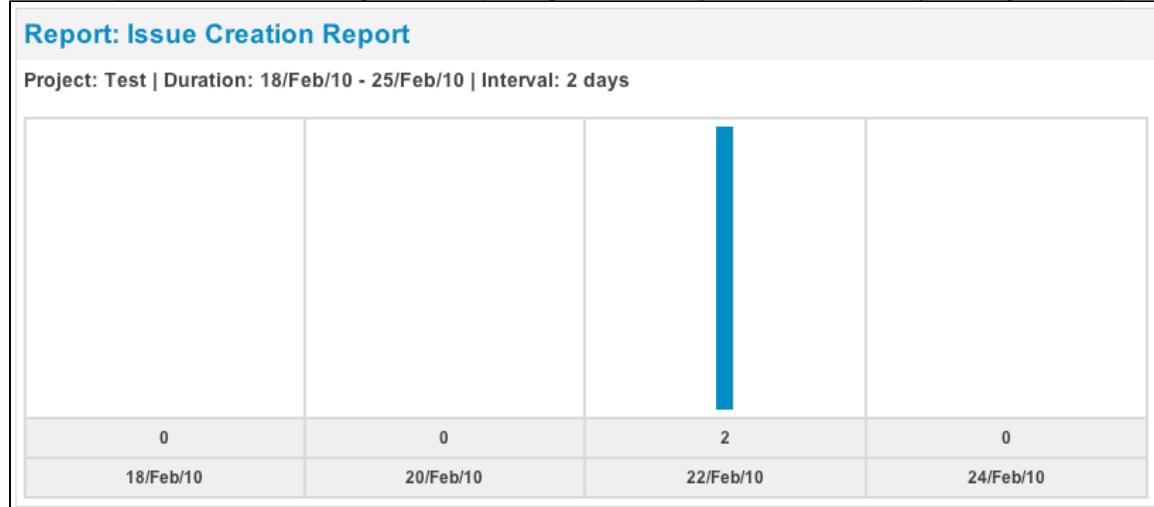
Before you can actually try the report, make sure you've created a project containing some issues and that you've create a filter that has a non empty result. You can then create the report by selecting it from the "Reports" dropdown on the 'Browse Project' page.

Example 2 - Issue Creation Report

In this example, a custom report is coded to display a histogram of issues created over a specified time. The report will collate all issues created within a specific project over the specified duration, subdivided by a configurable time interval.

You can again just [checkout the source code from SVN](#) or follow the tutorial to create the code from scratch.

The final report should look something like this, depending on the data in your instance and how you configured the report:



Step 1. Register the Plugin Module in the Plugin Descriptor

You will need to add another plugin module to your atlassian-plugin.xml to register the second report:

/atlassian-plugin.xml

```
<!-- An 'Issue Creation' Report - displays a histogram of issue create within a specified project over a specified time -->
<report key="issuecreationreport" name="Example: Issue Creation Report"
class="com.atlassian.plugins.tutorial.jira.report.CreationReport">
    <description key="report.issuecreation.description">i18n description</description>
    <label key="report.issuecreation.label" />

    <resource type="velocity" name="view"
location="templates/creationreport/issuecreation-report.vm" />
    <resource type="i18n" name="i18n"
location="com.atlassian.plugins.tutorial.jira.report.issuecreation_report" />

    <properties>
        <property>
            <key>projectid</key>
            <name>report.issuecreation.projectid.name</name>
            <description>report.issuecreation.projectid.description</description>
            <type>select</type>
            <values class="com.atlassian.jira.portal.ProjectValuesGenerator"/>
        </property>
        <property>
            <key>startDate</key>
            <name>report.issuecreation.startdate</name>
            <description>report.issuecreation.startdate.description</description>
            <type>date</type>
        </property>
        <property>
            <key>endDate</key>
            <name>report.issuecreation.enddate</name>
            <description>report.issuecreation.enddate.description</description>
            <type>date</type>
        </property>
        <property>
            <key>interval</key>
            <name>report.issuecreation.interval</name>
            <description>report.issuecreation.interval.description</description>
            <type>long</type>
            <default>3</default>
        </property>
    </properties>
</report>
```

The report module element defines that the logic is encapsulated in the class `CreationReport`. The name and description properties and the location of the i18n files are also specified.

In this case, the report requires four parameters to correctly display the data and are specified as follows:

- `projectid` - a select field specifying the project on which the report will focus. The possible project options available are retrieved through JIRA's `ProjectValuesGenerator`.
- `startDate` - a date field specifying the start of the time period for the report.
- `endDate` - a date field specifying the end of the time period for the report.
- `interval` - a numeric field specifying the time interval used to divide the overall time period (this is the histogram interval).

The plugin system will construct a suitable report configuration screen - allowing the user to specify the above parameters.

Further information:

- [Report Plugin Module](#)
- [Configuring Plugins with Object Configurable Parameters](#)

Step 2. Add the Java class for the report calculation

Add the following class to your project:

/src/main/java/com/atlassian/plugins/tutorial/jira/report/CreationReport.java

```
package com.atlassian.plugins.tutorial.jira.report;

import com.atlassian.core.util.DateUtils;
import com.atlassian.jira.issue.search.SearchException;
```

```

import com.atlassian.jira.issue.search.SearchProvider;
import com.atlassian.jira.jql.builder.JqlQueryBuilder;
import com.atlassian.jira.plugin.report.impl.AbstractReport;
import com.atlassian.jira.project.ProjectManager;
import com.atlassian.jira.util.I18nHelper;
import com.atlassian.jira.util.ParameterUtils;
import com.atlassian.jira.web.action.ProjectActionSupport;
import com.atlassian.jira.web.bean.I18nBean;
import com.atlassian.jira.web.util.OutlookDate;
import com.atlassian.jira.web.util.OutlookDateManager;
import com.atlassian.query.Query;
import com.opensymphony.user.User;

import org.apache.log4j.Logger;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Generate a histogram displaying number of issues opened in a specified period.
 * The time period is divided by the specified value for the histogram display.
 */
public class CreationReport extends AbstractReport
{
    private static final Logger log = Logger.getLogger(CreationReport.class);

    // The max height for each bar in the histogram
    private static final int MAX_HEIGHT = 400;
    // Default interval value
    private Long DEFAULT_INTERVAL = new Long(7);

    // The highest issue count encountered in a search
    private long maxCount = 0;
    // A collection of issue open counts
    private Collection<Long> openIssueCounts = new ArrayList<Long>();
    // A collection of interval start dates - correlating with the openIssueCount collection.
    private Collection<Date> dates = new ArrayList<Date>();

    private final SearchProvider searchProvider;
    private final OutlookDateManager outlookDateManager;
    private final ProjectManager projectManager;

    public CreationReport(SearchProvider searchProvider, OutlookDateManager outlookDateManager,
    ProjectManager projectManager)
    {
        this.searchProvider = searchProvider;
        this.outlookDateManager = outlookDateManager;
        this.projectManager = projectManager;
    }

    // Generate the report
    public String generateReportHtml(ProjectActionSupport action, Map params) throws Exception
    {
        User remoteUser = action.getRemoteUser();
        I18nHelper i18nBean = new I18nBean(remoteUser);

        // Retrieve the project parameter
        Long projectId = ParameterUtils.getLongParam(params, "projectid");
        // Retrieve the start and end dates and the time interval specified by the user
        Date startDate = ParameterUtils.getDateParam(params, "startDate", i18nBean.getLocale());
        Date endDate = ParameterUtils.getDateParam(params, "endDate", i18nBean.getLocale());
        Long interval = ParameterUtils.getLongParam(params, "interval");

        // Ensure that the interval is valid
        if (interval == null || interval.longValue() <= 0)
        {
            interval = DEFAULT_INTERVAL;
            log.error(action.getText("report.issuecreation.default.interval"));
        }

        getIssueCount(startDate, endDate, interval, remoteUser, projectId);
    }
}

```

```

List<Number> normalCount = new ArrayList<Number>();

// Normalise the counts for the max height
if (maxCount != MAX_HEIGHT && maxCount > 0)
{
    for (Long asLong : openIssueCounts)
    {
        Float floatValue = new Float((asLong.floatValue() / maxCount) * MAX_HEIGHT);
        // Round it back to an integer
        Integer newValue = new Integer(floatValue.intValue());

        normalCount.add(newValue);
    }
}

if (maxCount < 0)
    action.addErrorMessage(action.getText("report.issuecreation.error"));

// Pass the issues to the velocity template
Map<String, Object> velocityParams = new HashMap<String, Object>();
velocityParams.put("startDate", startDate);
velocityParams.put("endDate", endDate);
velocityParams.put("openCount", openIssueCounts);
velocityParams.put("normalisedCount", normalCount);
velocityParams.put("dates", dates);
velocityParams.put("maxHeight", new Integer(MAX_HEIGHT));
velocityParams.put("outlookDate",
outlookDateManager.getOutlookDate(i18nBean.getLocale()));
velocityParams.put("projectName", projectManager.getProjectObj(projectId).getName());
velocityParams.put("interval", interval);

return descriptor.getHtml("view", velocityParams);
}

// Retrieve the issues opened during the time period specified.
private long getOpenIssueCount(User remoteUser, Date startDate, Date endDate, Long projectId)
throws SearchException
{
    JqlQueryBuilder queryBuilder = JqlQueryBuilder.newBuilder();
    Query query = queryBuilder.where().createdBetween(startDate,
endDate).and().project(projectId).buildQuery();

    return searchProvider.searchCount(query, remoteUser);
}

private void getIssueCount(Date startDate, Date endDate, Long interval, User remoteUser, Long
projectId) throws SearchException
{
    // Calculate the interval value in milliseconds
    long intervalValue = interval.longValue() * DateUtils.DAY_MILLIS;
    Date newStartDate;
    long count = 0;

    // Split the specified time period by the interval value
    while (startDate.before(endDate))
    {
        newStartDate = new Date(startDate.getTime() + intervalValue);

        // Retrieve the issues opened within the time interval
        if (newStartDate.after(endDate))
            count = getOpenIssueCount(remoteUser, startDate, endDate, projectId);
        else
            count = getOpenIssueCount(remoteUser, startDate, newStartDate, projectId);

        // Store the highest count for normalisation of results
        if (maxCount < count)
            maxCount = count;

        // Store the count and the start date for this period
        openIssueCounts.add(new Long(count));
        dates.add(startDate);

        // Move start date to next period
        startDate = newStartDate;
    }
}

```

```
        }

    }

    // Validate the parameters set by the user.
    public void validate(ProjectActionSupport action, Map params)
    {
        User remoteUser = action.getRemoteUser();
        I18nHelper i18nBean = new I18nBean(remoteUser);

        Date startDate = ParameterUtils.getDateParam(params, "startDate", i18nBean.getLocale());
        Date endDate = ParameterUtils.getDateParam(params, "endDate", i18nBean.getLocale());
        Long interval = ParameterUtils.getLongParam(params, "interval");
        Long projectId = ParameterUtils.getLongParam(params, "projectid");

        OutlookDate outlookDate = outlookDateManager.getOutlookDate(i18nBean.getLocale());

        if (startDate == null || !outlookDate.isDatePickerDate(outlookDate.formatDMY(startDate)))
            action.addError("startDate",
                action.getText("report.issuecreation.startdate.required"));

        if (endDate == null || !outlookDate.isDatePickerDate(outlookDate.formatDMY(endDate)))
            action.addError("endDate", action.getText("report.issuecreation.enddate.required"));

        if (interval == null || interval.longValue() <= 0)
            action.addError("interval", action.getText("report.issuecreation.interval.invalid"));

        if (projectId == null)
            action.addError("projectid",
                action.getText("report.issuecreation.projectid.invalid"));

        // The end date must be after the start date
        if (startDate != null && endDate != null && endDate.before(startDate))
        {
            action.addError("endDate", action.getText("report.issuecreation.before.startdate"));
        }
    }
}
```

```

        }
    }
}

```

The `CreationReport` class retrieves the parameters as specified by the user. Next, the relevant issue counts are retrieved from the system for the specified time over the specified time interval from the specified project. The issue counts are normalised in order to produce a balanced histogram. Finally, the relevant details are passed to the velocity template.

Step 3. Add the view template

The view template displays the histogram constructed from the data passed from the class `CreationReport`:

```

/src/main/resources/templates/creationreport/issuecreation-report.vm

<div style="padding: 5px">
    <!-- Display the report configuration -->
    <h4>
        $i18n.getText('report.issuecreation.project'): $projectName |
        $i18n.getText('report.issuecreation.duration'): $outlookDate.formatDMY($startDate) -
        $outlookDate.formatDMY($endDate) |
        $i18n.getText('report.issuecreation.interval'): $interval
        $i18n.getText('report.issuecreation.interval.days')
    </h4>
    <br />
    <table style="width: 100%; border: 0; background-color: lightgrey">
        <!-- Create a row to display the bars-->
        <tr valign="bottom" style="background-color: white; padding: 1px">
            #foreach ($normalCount in $normalisedCount)
                <td height="$maxHeight" align="center">
                    #if ($normalCount == 0)
                        &nbsp;
                    #else
                        
                    #end
                </td>
            #end
        </tr>
        <!-- Have one row for the issue count -->
        <tr style="background-color: #eee; padding: 1px">
            #foreach ($count in $openCount)
                <td align="center"><b>$count</b></td>
            #end
        </tr>
        <!-- And one row to display the date -->
        <tr style="background-color: #eee; padding: 1px">
            #foreach ($date in $dates)
                <td align="center"><b>$outlookDate.formatDMY($date)</b></td>
            #end
        </tr>
    </table>
</div>

```

Add this under `src/main/resources/templates/creationreport/issuecreation-report.vm` to your project.

Step 4. Add i18n messages

The following messages provide the labels needed for the report configuration screen and the actual report. Create a property file with the following content at `src/main/resources/com/atlassian/plugins/tutorial/jira/report/issuecreation_report.properties`:

src/main/resources/com/atlassian/plugins/tutorial/jira/report/issuecreation_report.properties

```
report.issuecreation.label = Issue Creation Report
report.issuecreation.name = Issue Creation Report
report.issuecreation.projectid.name = Project
report.issuecreation.projectid.description = Select the project to display report on.
report.issuecreation.description = Report displaying a histogram of issues opened over a specified period.
report.issuecreation.startdate = Start Date
report.issuecreation.startdate.description = Graph all issues created after this date.
report.issuecreation.enddate = End Date
report.issuecreation.enddate.description = Graph all issues created before this date.
report.issuecreation.interval = Interval
report.issuecreation.interval.days = days
report.issuecreation.interval.description = Specify the interval (in days) for the report.
report.issuecreation.startdate.required = A valid "Start Date" is required to generate this report.
report.issuecreation.enddate.required = A valid "End Date" is required to generate this report.
report.issuecreation.interval.invalid = The interval must be a number greater than 0.
report.issuecreation.before.startdate = The "End Date" must be after the "Start Date".
report.issuecreation.error = Error occurred generating Issue Creation Report.
report.issuecreation.projectid.invalid = Please select a valid project.
report.issuecreation.default.interval = The interval specified is invalid - using default interval.
report.issuecreation.duration = Duration
report.issuecreation.project = Project
```

Step 5. Build, Install and Run the Plugin

Run the following commands to build and install your plugin, so that you can test your code.

- Open a command window and go to the plugin root folder (where the pom.xml is located).
- If you have not already started the application, do that now by running atlas-run.

Wait until JIRA is started up and then go back to the browser. The plugin has been installed into the application, and you can test your changes.

Before you can actually try the report, again, make sure you've created a project containing some issues. You can then create the report by selecting it from the "Reports" dropdown on the 'Browse Project' page.

Plugin Tutorial - Defining a Pluggable Service in a Confluence Plugin



Level of experience: Advanced

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'advanced' level. If you have never developed a plugin before, we advise you to try a beginner tutorial first.

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#). You will find two projects within this directory:

| | |
|------------------------------|--|
| confluence-reporting-service | Contains the service component and the module type definitions |
| system-config-report | A plugin which defines a report |

On this page:

- Overview
- Required Knowledge
- Plugin Components
- Report Service Plugin
 - New Module Type
 - Report Service
 - Report Web Interface
 - Build and install
- System Reporting Plugin
 - pom.xml setup
 - Creating the Report
 - Build and Install into Confluence
- Conclusion

Overview

This tutorial describes how to define a plugin which has a pluggable service, exposed as a component, where plugins are comprised of a [custom plugin module type](#). This covers several items:

- Defining a Module Type
- Defining a Service that allows implementations of the Module Type to be listed
- Implement an instance of the Module Type

The example we are implementing is a reporting service that provides a way for other plugins to provide reports to be viewed through a single interface.

Required Knowledge

To complete this tutorial, you must already understand the basics of Java development: classes, interfaces, methods, how to use the compiler, and so on. You should also understand:

- How to create an Atlassian Confluence plugin project using the [Atlassian Plugin SDK](#).
- How to compile and install your project within Confluence.

As this is an advanced plugin tutorial we will only highlight or discuss the important parts of the implementation. You should also review the source code for the two example plugins while reading this tutorial.

Plugin Components

This tutorial contains two components:

- **Reporting Service** (`confluence-reporting-service`) - which defines a custom `report-descriptor` plugin module and displays reports
- **System Information Report** (`system-config-report`) - which implements the `report-descriptor` plugin module to display a simple report about the Confluence system

Report Service Plugin

The Report Service Plugin contains an user interface for viewing the registered reporters and for triggering reports. It also contains the new plugin module type that report implementing plugins can use to identify their reports.

New Module Type

We are defining a new module type called `report` in this plugin which allows other plugins specify reports that they are able to produce. A report consists of several properties, used to identify each report, and a method which causes the report to be generated and the results returned. Each plugin that wishes to generate reports must implement the following `Report` interface for each report they wish to export:

```
public interface Report {  
    public String generateReport();  
    public String getName();  
    public String getDescription();  
    public String getKey();  
}
```

At this point it is possible to consider an alternative solution to the one we demonstrate here: rather than define a custom module type other plugins could explicitly and programmatically publish their `Report` types in a similar manner to the [Observer Pattern design pattern](#). The difference between these approaches will be commented on at the end of the tutorial.

In order that plugins can expose their `Report` types we define a new XML element to be used in the `atlassian-plugin.xml`, this is achieved by extending the `AbstractModuleDescriptor`

```
public final class ReportDescriptor extends AbstractModuleDescriptor<Report> {  
    public Report getModule() {  
        return ((AutowireCapablePlugin) getPlugin()).autowire(getModuleClass());  
    }  
}
```

and adding the new module type to the `atlassian-plugin.xml` so that other plugins can use the `<report>` element

```
<module-type key="report" class="com.adaptavist.tutorials.reportingexample.core.ReportDescriptor"  
name="Report Module Descriptor">  
    <description>Module Descriptor for 'report' module type.</description>  
</module-type>
```

Report Service

So that this plugin and others can query and/or generate the defined reports we implement a simple service component, ReportService, which is exposed publicly in the plugin xml file:

```
<component key="report-service"
  class="com.adaptavist.tutorials.reportingexample.core.DefaultReportService" name="Report Service"
  public="true">
  <interface>com.adaptavist.tutorials.reportingexample.core.ReportService</interface>
</component>
```

The DefaultReportService implementation of the ReportService interface uses the PluginAccessor to access all the report plugin modules:

```
public List<Report> getAllReports() {
    List<ReportDescriptor> reportDescriptors =
    pluginAccessor.getEnabledModuleDescriptorsByClass(ReportDescriptor.class);
    List<Report> reportTypes = new ArrayList<Report>(reportDescriptors.size());
    for(ReportDescriptor descriptor : reportDescriptors) {
        reportTypes.add(descriptor.getModule());
    }
    return reportTypes;
}
```

this means that at each invocation the list of available plugins is rebuilt, a type of dynamic lookup which is in part what allows this plugin to be "pluggable".

Report Web Interface

The admin report interface is created using a standard web work action, and web-ui module to provide a link in the Confluence admin console. If you have created a plugin with a custom action previously there should be nothing unfamiliar in the setup, if not then there are several tutorials which will easily guide you through this process.

The UI performs two basic tasks: it displays all the available reports, and displays the output of any one report. In both cases the Action simply invokes a method on the ReportService to find the appropriate Report(s). Because the actions are given a path underneath /admin default Confluence interceptors ensure that only admin users have access to the reports.

Build and install

Once these three components have been set up, you should be able to build and install the plugin in Confluence. When you access the report menu, either via the plugin configure link or the link on the admin sidebar, no reports will be listed as there are no plugins that have a report module installed yet.

System Reporting Plugin

- Implements the Report and returns the system information (similar as to Confluence Admin > System Info page) formatted as a simple (undecorated) HTML page.

pom.xml setup

As we are going to extend the Report interface provided in the Report Service plugin we will need to add it as a dependency to our maven project file. Before you can do this we need to install the service plugin into our local maven repository.

From where the service plugin is checked out type:

```
mvn install
```

or (if using the plugin sdk)

```
atlas-mvn install
```

to install into your local repository.



This will need to be repeated anytime there is an update to the report service plugin.

Now we can add the dependency to our pom.xml for the system reporting plugin:

```

<dependency>
    <groupId>com.adaptavist.tutorials.reportingexample</groupId>
    <artifactId>confluence-reporting-service</artifactId>
    <version>1.0-SNAPSHOT</version>
    <scope>provided</scope>
</dependency>

```

Creating the Report

To create a new custom report we need to create a class that implements the `Report` interface. Our example report `SystemInformationReport` provides simple information about the Confluence system it is installed on.

```

public String generateReport() {
    return "Global Spaces: " + systemInformationService.getUsageInfo().getGlobalSpaces() +
        "<br/>Pages:" + systemInformationService.getUsageInfo().getCurrentContent();
}

```

Once the report is implemented we need to add it to our `atlassian-plugin.xml` config. We use the new `report` module to define our report:

```

<report
    key="systeminfo-report-descriptor"
    name="System Information Report"
    class="com.adaptavist.tutorial.reporting.example.SystemInformationReport" />

```

Build and Install into Confluence

Because the system information report plugin depends on the module defined in the reporting service plugin it must be installed into Confluence first.



During development we have found that updating the plugin that defines a new modules type (e.g. `confluence-reporting-service`) won't re-register all the implementing modules. Uninstalling and reinstalling the implementing modules will normally fix it, or a full system restart. This does slow down development somewhat.

Conclusion

While this example is kept very simple it is possible to see how it could be extended in to an advanced reporting framework where plugins could export reports covering usage, configuration, or be able to output in different formats (XML, PDF, Word) or collate several reports in to one. Alternatively this mechanism of providing a easily extensible plugin can be applied to other situations.

As mentioned earlier in the tutorial this is not the only possible solution to the problem; the `DefaultReportService` performs a dynamic lookup each time all or one of the `Report` instances must be fetched which as the number of installed plugins grows, or the number of defined reports, increases in overhead. Implementing the Observer pattern may also help alleviate this, but that is out of scope for this tutorial.

Plugin Tutorial - Scheduling Events via SAL



Level of experience: Advanced

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'advanced' level. If you have never developed a plugin before, we advise you to try a beginner tutorial first.

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can:

- check out the source code: `svn co https://svn.atlassian.com/svn/public/contrib/tutorials/scheduled-events/trunk scheduler-tutorial`
- or browse the sources online

On this page:

- Overview
- Required Knowledge
- Step 1. Create the Plugin Project
- Step 2. Add the required Maven dependencies
- Step 3. Import the SAL scheduler component to the Plugin Descriptor
- Step 4. Write the Background Task
- Step 5. Write the Component that Schedules the Task

- Step 6. Add the Component to atlassian-plugin.xml
- Step 7. Have a Beer and Put Your Feet Up!
- Step 8. Extend the Component Interface
- Step 9. Add a WebWork Action
- Step 10. Register the WebWork Action in atlassian-plugin.xml
- Step 11. Add the Velocity Template
- Step 12. Start JIRA
 - Screenshots

Overview

This tutorial shows you how to schedule Java tasks in your plugin that run in the background at regular intervals. To this end we will use the cross-product `PluginScheduler` component from `SAL` (Shared Access Layer).

Scheduled background tasks can be useful in many situations where relatively expensive tasks, or housekeeping works needs to be run periodically. In this tutorial our periodic background task will be a search on Twitter that runs every 5 seconds and stores the results of the most recent search in memory (for the sake of this tutorial we'll pretend that a Twitter search is a very expensive task).

To make the tutorial a little interesting, the invisible background search task is accompanied by a JIRA administration page that renders the latest search result and offers the user the ability to change both the search query and the interval period. This also implements the necessary plumbing to unschedule and reschedule events.

In order to do all this, our plugin will consist of the following components:

- Java classes encapsulating the plugin logic (a webwork 1 action and the SAL scheduled event)
- A velocity template for the admin page that renders the Twitter search results
- A web item to add a link to the administration context menu
- An internationalisation resource bundle

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Required Knowledge

To complete this tutorial, you must already understand the basics of Java development: classes, interfaces, methods, how to use the compiler, and so on. You should also understand:

- how to create an Atlassian plugin project using the [Atlassian Plugin SDK](#)
- how to open the plugin project in your IDE
- how to compile your project and create a JAR file using Maven.

This tutorial will teach you:

- how to use a SAL (Shared Access Library) component in a plugin
- how to tap into the plugin framework's lifecycle system
- how to use a webwork action with a velocity template
- how to create and use internationalisation resource bundles
- how to add a Web Item

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

We'll be using the Atlassian Plugin SDK throughout the tutorial, so make sure you have it installed and working as described here. To check that you're ready to go, try the `atlas-version` command; you should see output like the following:

```
$ atlas-version

ATLAS Version: 3.0.4
ATLAS Home: /Users/administrator/usr/atlassian-plugin-sdk-3.0.4
ATLAS Scripts: /Users/administrator/usr/atlassian-plugin-sdk-3.0.4/bin
ATLAS Maven Home: /Users/administrator/usr/atlassian-plugin-sdk-3.0.4/apache-maven
-----
Executing: /Users/administrator/usr/atlassian-plugin-sdk-3.0.4/apache-maven/bin/mvn --version
Apache Maven 2.1.0 (r755702; 2009-03-19 06:10:27+1100)
Java version: 1.6.0_15
Java home: /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home
Default locale: en_US, platform encoding: MacRoman
OS name: "mac os x" version: "10.6" arch: "x86_64" Family: "mac"
$
```

Then create a new JIRA plugin by running the `atlas-create-jira-plugin` command and filling in appropriate values for the plugin's groupId and artifactId when prompted.

Read below for an example of this:

```
$ atlas-create-jira-plugin
Executing: /Users/administrator/usr/atlassian-plugin-sdk-3.0.4/apache-maven/bin/mvn
com.atlassian.maven.plugins:maven-jira-plugin:3.0.4:create
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building Maven Default Project
[INFO]   task-segment: [com.atlassian.maven.plugins:maven-jira-plugin:3.0.4:create]
(aggregator-style)
[INFO] -----
[INFO] [jira:create]
[INFO] Setting property: classpath.resource.loader.class =>
'org.codehaus.plexus.velocity.ContextClassLoaderResourceLoader'.
[INFO] Setting property: velocimacro.messages.on => 'false'.
[INFO] Setting property: resource.loader => 'classpath'.
[INFO] Setting property: resource.manager.logwhenfound => 'false'.
[INFO] [archetype:generate]
[INFO] Generating project in Interactive mode
[INFO] Archetype repository missing. Using the one from
[com.atlassian.maven.archetypes:jira-plugin-archetype:5 -> https://maven.atlassian.com/public]
found in catalog internal
Define value for groupId: : com.atlassian.example
Define value for artifactId: : scheduling
Define value for version: 1.0-SNAPSHOT: :
Define value for package: com.atlassian.example: : com.atlassian.example.scheduling
Confirm properties configuration:
groupId: com.atlassian.example
artifactId: scheduling
version: 1.0-SNAPSHOT
package: com.atlassian.example.scheduling
Y: :
[INFO] -----
[INFO] Using following parameters for creating OldArchetype: jira-plugin-archetype:3.0.4
[INFO] -----
[INFO] Parameter: groupId, Value: com.atlassian.example
[INFO] Parameter: packageName, Value: com.atlassian.example.scheduling
[INFO] Parameter: package, Value: com.atlassian.example.scheduling
[INFO] Parameter: artifactId, Value: scheduling
[INFO] Parameter: basedir, Value: /private/tmp
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] ***** End of debug info from resources from generated POM
*****
[INFO] OldArchetype created in dir: /private/tmp/scheduling
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 1 minute 1 second
[INFO] Finished at: Mon Feb 22 18:13:41 EST 2010
[INFO] Final Memory: 42M/252M
[INFO] -----
$
```

Step 2. Add the required Maven dependencies

In this tutorial we are using both SAL and the open source Java Twitter library [twitter4j](#). Add both to the `pom.xml` file:

```

<dependencies>
...
    <dependency>
        <groupId>net.homeip.yusuke</groupId>
        <artifactId>twitter4j</artifactId>
        <version>2.0.10</version>
    </dependency>
    <dependency>
        <groupId>com.atlassian.sal</groupId>
        <artifactId>sal-api</artifactId>
        <version>2.0.0</version>
        <scope>provided</scope>
    </dependency>
...
</dependencies>

```

Step 3. Import the SAL scheduler component to the Plugin Descriptor

For the plugin framework to be able to inject the SAL `PluginScheduler`, we need to import the component explicitly in `atlassian-plugin.xml`, so add the following element:

```

<component-import key="pluginScheduler">
    <description>SAL Scheduler</description>
    <interface>com.atlassian.sal.api.scheduling.PluginScheduler</interface>
</component-import>

```

Step 4. Write the Background Task

Now let's write a component that gets the SAL `PluginScheduler` injected and then registers the periodic background task at startup.

First implement the job itself, which must be a public class that implements `com.atlassian.sal.api.scheduling.PluginJob`:

```

package com.atlassian.example.scheduling;

import com.atlassian.sal.api.scheduling.PluginJob;
import org.apache.log4j.Logger;
import twitter4j.Query;
import twitter4j.Twitter;
import twitter4j.TwitterException;

import java.util.Date;
import java.util.Map;

public class TwitterQueryTask implements PluginJob {

    private final Logger logger = Logger.getLogger(TwitterQueryTask.class);

    /**
     * Executes this job.
     *
     * @param jobDataMap any data the job needs to execute. Changes to this data will be
     * remembered between executions.
     */
    public void execute(Map<String, Object> jobDataMap) {

        final TwitterMonitorImpl monitor =
        (TwitterMonitorImpl)jobDataMap.get(TwitterMonitorImpl.KEY);
        assert monitor != null;
        try {
            final Twitter twitter = new Twitter();
            monitor.setTweets(twitter.search(new Query(monitor.getQuery())).getTweets());
            monitor.setLastRun(new Date());
        } catch (TwitterException te) {
            logger.error("Error talking to Twitter: " + te.getMessage(), te);
        }
    }
}

```

Note the map that is passed to our `execute()` method at runtime by the central scheduler. It provides us with a means of communicating with the job.

The way the scheduler works is that when we register a job, we pass the scheduler the *class name* of our task, rather than a concrete instance, while it is the scheduler that will instantiate our class. This has a number of consequences, namely that it must have a default, public constructor and that we'll need to use the `jobDataMap` if we want to provide the job with runtime configuration.

When passing data to your job using the `jobDataMap`, use unique string keys to identify them. In our implementation we store a reference to our `TwitterMonitorImpl` plugin component which is responsible to scheduling our task and also receives the Twitter search results. We store this reference using the key `TwitterMonitorImpl.KEY`, which we'll declare in our component in the next section.

Finally see how we use the `twitter4j` library that allows us to do a public, anonymous search with only 2 lines of code.

Step 5. Write the Component that Schedules the Task

This is the class that we register as a plugin component in `atlassian-plugin.xml`. It gets instantiated by the plugin framework at application startup and is responsible for registering our job. It also stores the Twitter search results and is accessible the webwork action we will add later.

```

package com.atlassian.example.scheduling;

import com.atlassian.sal.api.lifecycle.LifecycleAware;
import com.atlassian.sal.api.scheduling.PluginScheduler;
import org.apache.log4j.Logger;
import twitter4j.Tweet;

import java.util.Date;
import java.util.HashMap;
import java.util.List;

public class TwitterMonitorImpl implements TwitterMonitor, LifecycleAware {

    /* package */ static final String KEY = TwitterMonitorImpl.class.getName() + ":instance";
    private static final String JOB_NAME = TwitterMonitorImpl.class.getName() + ":job";

    private final Logger logger = Logger.getLogger(TwitterMonitorImpl.class);
    private final PluginScheduler pluginScheduler; // provided by SAL

    private String query = "Atlassian"; // default Twitter search
    private long interval = 5000L; // default job interval (5 sec)
    private List<Tweet> tweets; // results of the last search
    private Date lastRun = null; // time when the last search returned

    public TwitterMonitorImpl(PluginScheduler pluginScheduler) {
        this.pluginScheduler = pluginScheduler;
    }

    // declared by LifecycleAware
    public void onStart() {
        reschedule(query, interval);
    }

    public void reschedule(String query, long interval) {
        this.query = query;
        this.interval = interval;

        pluginScheduler.scheduleJob(
            JOB_NAME, // unique name of the job
            TwitterQueryTask.class, // class of the job
            new HashMap<String, Object>() {{
                put(KEY, TwitterMonitorImpl.this);
            }}, // data that needs to be passed to the job
            new Date(), // the time the job is to start
            interval); // interval between repeats, in milliseconds
        logger.info(String.format("Twitter search task scheduled to run every %dms", interval));
    }

    public String getQuery() {
        return query;
    }

    /* package */ void setTweets(List<Tweet> tweets) {
        this.tweets = tweets;
    }

    /* package */ void setLastRun(Date lastRun) {
        this.lastRun = lastRun;
    }
}

```

Notice how we implement SAL's `com.atlassian.sal.api.lifecycle.LifecycleAware` interface and use its `onStart()` method to register the job.

It is critical that we do not attempt to (un)register any jobs in our component's constructor, as the scheduler (and indeed SAL itself) may not yet be fully initialised by the time your constructor is called. Therefore, **always** implement `com.atlassian.sal.api.lifecycle.LifecycleAware` and register your job in `onStart()`.

As with every plugin component, we create an interface that we use when sharing our component with other plugin modules:

```
package com.atlassian.example.scheduling;

public interface TwitterMonitor {

    public void reschedule(String query, long interval);
}
```

Step 6. Add the Component to atlassian-plugin.xml

```
...
<component key="schedulerComponent"
class="com.atlassian.example.scheduling.TwitterMonitorImpl"
    system="true" public="true">
    <description>The plugin component that schedules the Twitter search.</description>
    <interface>com.atlassian.sal.api.lifecycle.LifecycleAware</interface>
    <interface>com.atlassian.example.scheduling.TwitterMonitor</interface>
</component>
...
```

Notice the explicit declaration of the `com.atlassian.sal.api.lifecycle.LifecycleAware` interface and how our component is declared public, so that SAL's lifecycle manager can access it.

Step 7. Have a Beer and Put Your Feet Up!

At this point you should have your scheduled event working, complete with Twitter search! Start JIRA, connect your debugger and place a breakpoint in your component's constructor, its `reschedule()` method and your job's `execute()` method and watch it go.



The Plugin SDK facilitates quick and easy deployment and debugging. To run our plugin in JIRA in debug mode, simply run:

```
$ atlas-debug
```

or explicitly using the Maven target:

```
$ mvn jira:debug
```



If you're satisfied with this intermediate result, go forth and schedule your events. If you want more, stick around and see how the remainder of this tutorial adds a Web Item, a webwork action, velocity template and internationalisation support to make things a bit more interactive and interesting.



Up to this point our plugin uses no product specific features or API's, and will therefore run in every Atlassian product, not just JIRA.

Step 8. Extend the Component Interface

In order to display the tweets in an admin page, we'll need to add some methods to the `TwitterMonitor` interface.

This is necessary because we'll have the `TwitterMonitorImpl` component injected into our webwork action and these extra methods in the interface will allow the action to communicate with the component and retrieve the latest search results and interval period.

```

package com.atlassian.example.scheduling;

import twitter4j.Tweet;

import java.util.Date;
import java.util.List;

public interface TwitterMonitor {

    public String getQuery();
    public long getInterval();
    public List<Tweet> getTweets();
    public Date getLastRun();
    public void reschedule(String query, long interval);
}

```

And implement them in TwitterMonitorImpl:

```

...
public class TwitterMonitorImpl implements TwitterMonitor, LifecycleAware {

    ...
    public long getInterval() {
        return interval;
    }

    public Date getLastRun() {
        return lastRun;
    }

    public List<Tweet> getTweets() {
        return tweets;
    }
}
...

```

Step 9. Add a WebWork Action

In the remainder of the tutorial we shall limit ourselves to JIRA and we'll create page in the administration section to display the Twitter search results. We'll also allow the user to change the search query and the search interval.

First let's implement the webwork action:

```

package com.atlassian.example.scheduling;

import com.atlassian.jira.web.action.JiraWebActionSupport;
import twitter4j.Tweet;

import java.util.Date;
import java.util.List;

public class SchedulerAction extends JiraWebActionSupport {

    private final TwitterMonitor twitterMonitor;
    private String query;
    private long interval;

    public SchedulerAction(TwitterMonitor twitterMonitor) {
        this.twitterMonitor = twitterMonitor;
        this.query = twitterMonitor.getQuery();
        this.interval = twitterMonitor.getInterval();
    }

    @Override
    protected String doExecute() throws Exception {
        return SUCCESS;
    }

    public String doReschedule() {
        twitterMonitor.reschedule(query, interval);
        return getRedirect("TwitterScheduler!default.jspa");
    }

    public List<Tweet> getTweets() {
        return twitterMonitor.getTweets();
    }

    public String getQuery() {
        return query;
    }

    public void setQuery(String query) {
        this.query = query;
    }

    public long getInterval() {
        return interval;
    }

    public void setInterval(long interval) {
        this.interval = interval;
    }

    public Date getLastRun() {
        return twitterMonitor.getLastRun();
    }
}

```

We have two entry methods into this action: the `doExecute()` method that has no side effects and merely provides access to the current search results and there's the `doReschedule()` method that we'll use when the user wants to change the interval or the search query, which causes the background job to be canceled and recreated.

Note that after a reschedule action, we won't render a page, but instead we'll redirect the browser back to the read-only action of `doExecute()` to avoid exposing the reschedule URL in the browser, as that would continuously reschedule our job every time the user hits the browser's reload button.

Step 10. Register the WebWork Action in `atlassian-plugin.xml`

We'll register the webwork action in `atlassian-plugin.xml` and also add a Web Item to add a link to the context menu of the JIRA administration section that will link to our new page:

```

...
<resource type="i18n" name="i18n"
location="com.atlassian.example.scheduling.TwitterSchedulerBundle"/>

<web-item key="schedulerActionLink" section="system.admin/system"
    i18n-name-key="com.atlassian.example.scheduling.adminLink"
    name="Scheduled Twitter Search" weight="1">
    <label key="com.atlassian.example.scheduling.adminLink"/>
    <link linkId="schedulerActionLink"/>/secure/admin/TwitterScheduler.jspa</link>
</web-item>

<webwork1 key="schedulerAction" name="SAL Scheduler Example">
    <actions>
        <action name="com.atlassian.example.scheduling.SchedulerAction"
            alias="TwitterScheduler">
            <view name="success"/>/templates/scheduler.vm</view>
            <view name="input"/>/templates/scheduler.vm</view>
        </action>
    </actions>
</webwork1>
...

```

Notice how we also add an i18n resource bundle for internationalisation support, so we can render different string depending on the user's locale.

Always using i18n is a good habit, even if you only provide one language bundle. However, if you can't be bothered at this stage, feel free to omit the bundle declaration and just hardcode your text.

- Go here For the [contents of the language bundle we use](#).
- [Click here](#) for more info on internationalisation support for plugins.

Step 11. Add the Velocity Template

Finally we'll add the `src/main/resources/templates/scheduler.vm` velocity template that renders the page. The snippet below only focuses on the interesting bits while omitting most of the layout. [The full template is in subversion](#).

```

    ...
    <form method="post" action="TwitterScheduler!reschedule.jspa">
        <p>
            <table>
                <tr>
                    <td>$i18n.getText("com.atlassian.example.scheduling.queryCell")</td>
                    <td><input type="text" name="query" value="$query"></td>
                </tr>
                <tr>
                    <td>$i18n.getText("com.atlassian.example.scheduling.intervalCell")</td>
                    <td><input type="text" name="interval" value="$interval"></td>
                </tr>
                <tr>
                    <td colspan="2"><input type="submit" value="$i18n.getText("com.atlassian.example.scheduling.applyButton")"></td>
                </tr>
            </table>
        </p>
    </form>
    ...
    <table class="jiraform maxWidth">
        <thead class="jiraformheader">
            <tr>
                <th>
                    ...
                <th>$i18n.getText("com.atlassian.example.scheduling.result.header.from")</th>
                <th>$i18n.getText("com.atlassian.example.scheduling.result.header.tweet")</th>
                <th>$i18n.getText("com.atlassian.example.scheduling.result.header.date")</th>
            </tr>
        </thead>
        <tbody id="tweets">
            #foreach ( $tweet in $tweets )
            <tr>
                <td></td>
                <td>$tweet.fromUser</td>
                <td>$tweet.text</td>
                <td>$tweet.createdAt</td>
            </tr>
            #end
        </tbody>
    </table>
    <div style="text-align: center;">$i18n.getText("com.atlassian.example.scheduling.lastRun")<br>$lastRun</div>
    ...

```

Step 12. Start JIRA

That concludes all code for our tutorial, so let's start it up and check it out:

```
$ mvn jira:run
```

Screenshots



Plugin Tutorial - Writing event listeners with the atlassian-event library



Level of experience: Intermediate

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'intermediate' level. If you have never developed a plugin before, you may find this one a bit difficult.

The source code of the plugin used in this tutorial is available on [BitBucket](#). You can check out the source code [here](#).

On this page:

- [Overview](#)
- [Required Knowledge](#)
- [Step 1. Create the Plugin Project](#)
- [Step 2. Add Plugin Metadata to the POM](#)
- [Step 3. Register the Plugin Module in the Plugin Descriptor](#)
- [Step 4. Write Java Classes](#)
 - Create the listener class
 - Inject the EventPublisher
 - Add the @EventListener annotation
 - Add the event handling code
 - Coordinate event registration with the plugin lifecycle
- [Step 5. Build, Install and Run the Plugin](#)
- [Notes](#)

Overview

JIRA has supported a [simple listener API](#) for a long time, but it has several problems:

- Difficult installation/setup
- Impossible (as of JIRA 4.0) to share code between listeners and plugins
- Complex configuration

Fortunately, it is possible to use the [atlassian-event library](#) to implement the same functionality as a legacy listener (see [JIRA-specific Atlassian Events](#)). This tutorial will show you how.

In order to do this, you will create a JIRA plugin. As with all plugins, your plugin will consist of the following components:

- Java classes encapsulating the plugin logic
- Plugin descriptor to enable the plugin module in JIRA

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Required Knowledge

To complete this tutorial, you must already understand the basics of Java development: classes, interfaces, methods, how to use the compiler, and so on. You should also understand:

- How to create an Atlassian plugin project using the [Atlassian Plugin SDK](#).
- How to open the plugin project in your IDE.
- How to compile your project and create a JAR file using Maven.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

When you create the plugin using the above SDK, you will be prompted for some information to identify your plugin. Enter the following information:

- **group-id:** com.example.plugins.tutorial
- **artifact-id:** new-listener-plugin
- **version:** 1.0
- **package:** com.example.plugins.tutorial

Step 2. Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
<organization>
    <name>Example Company</name>
    <url>http://www.example.com/</url>
</organization>
```

3. Update the `<description>` element:

```
<description>This plugin implements a simple issue event listener for JIRA using atlassian-event.</description>
```

4. Save the file.

Step 3. Register the Plugin Module in the Plugin Descriptor

Now you will add the required plugin modules to your plugin descriptor at `src/main/resources/atlassian-plugin.xml`. The plugin descriptor is an XML file that identifies the plugin to JIRA and defines the functionality that the plugin requires.

Here's a basic plugin descriptor:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}"
    plugins-version="2">
    <plugin-info>
        <description>${project.description}</description>
        <version>${project.version}</version>
        <vendor name="${project.organization.name}" url="${project.organization.url}" />
    </plugin-info>
</atlassian-plugin>
```

Note that some of the information from the POM is transferred to the plugin descriptor using variable names such as `${project.artifactId}`.

You will need the following plugin modules:

```
<component-import key="eventPublisher" interface="com.atlassian.event.api.EventPublisher"/>
<component key="eventListener" class="com.example.tutorial.plugins.IssueCreatedResolvedListener">
    <description>Class that processes the incoming JIRA issue events.</description>
</component>
```

The first line imports the `EventPublisher` instance used to register for events, while the second line instantiates the listener class we'll write. To be eligible for dependency injection, your classes must be registered as `<component>` plugin modules.

Add this code to your `atlassian-plugin.xml`.

Now you are ready to move onto writing some code to make your plugin do something.

Step 4. Write Java Classes

The recipe for creating an event listener is very simple:

- Inject the `EventPublisher` implementation into your class
- Add an `@EventListener` annotation to the method(s) that should receive events
- Handle events when they come in

To demonstrate this, we'll create a listener that will log a notification whenever an issue is created or resolved. It's easy to imagine how this same approach could be used to send notifications, say to email or IRC, but we'll leave those as exercises for the reader.

Create the listener class

Create a class called `IssueCreatedResolvedListener`:

```
package com.example.tutorial.plugins;

public class IssueCreatedResolvedListener {
```

Inject the EventPublisher

In `atlassian-plugin.xml`, we added a `<component-import>` for the `EventPublisher`, the entry point to the atlassian-event library. The `EventPublisher` handles publication of events and registration of event listeners. In this case, we're going to register our `IssueCreatedResolvedListener` class, so we will have the plugin system inject an `EventPublisher` into our plugin through the constructor:

```
package com.example.tutorial.plugins;

import com.atlassian.event.api.EventPublisher;

public class IssueCreatedResolvedListener {

    public IssueCreatedEventListener(EventPublisher eventPublisher) {
        eventPublisher.register(this);      // Demonstration only -- don't do this in real code!
    }

}
```

While this works for a demonstration, it isn't what you want to do in your actual plugin. We'll explain why in a moment.

Add the @EventListener annotation

The actual event handling is done by methods that are annotated with `@EventListener`. Such methods take a parameter corresponding to the event they wish to handle. JIRA provides `UserEvent` and `IssueEvent` events; we'll use the `IssueEvent` for this illustration.

```
package com.example.tutorial.plugins;

import com.atlassian.event.api.EventListener;
import com.atlassian.event.api.EventPublisher;
import com.atlassian.jira.event.issue.IssueEvent;

public class IssueCreatedResolvedListener {

    public IssueCreatedEventListener(EventPublisher eventPublisher) {
        eventPublisher.register(this);      // Demonstration only -- don't do this in real code!
    }

    @EventListener
    public void onIssueEvent(IssueEvent issueEvent) {
        // ...
    }
}
```

`@EventListener` can be applied to any public method.

Add the event handling code

Let's add some logging to prove that our listener works correctly.

```

package com.example.tutorial.plugins;

import com.atlassian.event.api.EventListener;
import com.atlassian.event.api.EventPublisher;
import com.atlassian.jira.event.issue.IssueEvent;
import com.atlassian.jira.event.type.EventType;
import com.atlassian.jira.issue.Issue;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class IssueCreatedResolvedListener {

    private static final Logger log = LoggerFactory.getLogger(IssueCreatedResolvedListener.class);

    public IssueCreatedEventListerner(EventPublisher eventPublisher) {
        eventPublisher.register(this); // Demonstration only -- don't do this in real code!
    }

    @EventListener
    public void onIssueEvent(IssueEvent issueEvent) {
        Long eventId = issueEvent.getEventTypeId();
        Issue issue = issueEvent.getIssue();

        // if it's an event we're interested in, log it
        if (eventId.equals(EventType.ISSUE_CREATED_ID)) {
            log.info("Issue {} has been created at {}.", issue.getKey(), issue.getCreated());
        } else if (eventId.equals(EventType.ISSUE_RESOLVED_ID)) {
            log.info("Issue {} has been resolved at {}.", issue.getKey(),
                    issue.getResolutionDate());
        }
    }
}

```

By default, the JIRA log4j implementation doesn't know anything about our plugin's logger, and since our package name is `com.example.tutorial.plugins`, we don't inherit any of the `com.atlassian` hierarchical loggers. We'll need to use some custom `log4j.properties` for our testing. To do this in the SDK, we add our own `log4j.properties` and put it somewhere the SDK can find it, say `src/aps`. Then we configure it in the POM:

```

<build>
    <plugins>
        <plugin>
            <groupId>com.atlassian.maven.plugins</groupId>
            <artifactId>maven-jira-plugin</artifactId>
            <version>3.3</version>
            <extensions>true</extensions>
            <configuration>
                <productVersion>${jira.version}</productVersion>
                <productDataVersion>${jira.data.version}</productDataVersion>
                <log4jProperties>src/aps/log4j.properties</log4jProperties> <!-- configure our own
logging -->
            </configuration>
        </plugin>
    </plugins>
</build>

```

Now when we start JIRA with `atlas-run`, our logger's output will be visible on the console. (We've omitted the log4j configuration from this page, but you can find it with [the rest of the plugin source on BitBucket](#).)

Coordinate event registration with the plugin lifecycle

The code so far makes two naive assumptions about the plugin lifecycle: the constructor will only ever be called once; the plugin will only shut down when the system itself shuts down. Neither is true. Plugins may expect to be enabled or disabled at any time, and since our plugin registers with an external service, this needs to be accounted for.

In the plugin system itself, Atlassian plugins are implemented as Spring dynamic modules, and the `atlassian-plugin.xml` is transformed into a Spring XML bean configuration before it is actually loaded by the product. In most cases, plugin developers don't need to worry about this. In our case, we do. As a `<component>`, `IssueCreatedResolvedListener` will become a Spring bean, so we can apply the Spring interfaces `InitializingBean` and `DisposableBean`. Spring guarantees that the bean – in this case, our listener class – will have a chance to get its act together before it's put into active service, or retired from it.

```

package com.example.tutorial.plugins;

import com.atlassian.event.api.EventListener;
import com.atlassian.event.api.EventPublisher;
import com.atlassian.jira.event.issue.IssueEvent;
import com.atlassian.jira.event.type.EventType;
import com.atlassian.jira.issue.Issue;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.DisposableBean;
import org.springframework.beans.factory.InitializingBean;

/**
 * Simple JIRA listener using the atlassian-event library and demonstrating
 * plugin lifecycle integration.
 */
public class IssueCreatedResolvedListener implements InitializingBean, DisposableBean {

    private static final Logger log = LoggerFactory.getLogger(IssueCreatedResolvedListener.class);

    private final EventPublisher eventPublisher;

    /**
     * Constructor.
     * @param eventPublisher injected {@code EventPublisher} implementation.
     */
    public IssueCreatedResolvedListener(EventPublisher eventPublisher) {
        this.eventPublisher = eventPublisher;
    }

    /**
     * Called when the plugin has been enabled.
     * @throws Exception
     */
    @Override
    public void afterPropertiesSet() throws Exception {
        // register ourselves with the EventPublisher
        eventPublisher.register(this);
    }

    /**
     * Called when the plugin is being disabled or removed.
     * @throws Exception
     */
    @Override
    public void destroy() throws Exception {
        // unregister ourselves with the EventPublisher
        eventPublisher.unregister(this);
    }

    /**
     * Receives any {@code IssueEvent}s sent by JIRA.
     * @param issueEvent the IssueEvent passed to us
     */
    @EventListener
    public void onIssueEvent(IssueEvent issueEvent) {
        Long eventId = issueEvent.getEventTypeId();
        Issue issue = issueEvent.getIssue();

        // if it's an event we're interested in, log it
        if (eventId.equals(EventType.ISSUE_CREATED_ID)) {
            log.info("Issue {} has been created at {}.", issue.getKey(), issue.getCreated());
        } else if (eventId.equals(EventType.ISSUE_RESOLVED_ID)) {
            log.info("Issue {} has been resolved at {}.", issue.getKey(),
                    issue.getResolutionDate());
        }
    }
}

```

Here, we've moved the `EventPublisher` reference to a member variable and bound the registration to the `afterPropertiesSet()`/`destroy()` methods, which come from `InitializingBean` and `DisposableBean` respectively. This guarantees the correct behavior even when our plugin is disabled and later reenabled, perhaps by an administrator using the [Universal Plugin Manager](#).

Step 5. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Notes

While writing a listener this way has the advantage of working inside a full-fledged plugin, it doesn't support the properties feature of legacy JIRA listeners. Properties are used to configure a listener; for example, a listener that forwarded issue events to IRC might have properties for the IRC server, username and password to use. `atlassian-event` listeners don't automatically integrate with the JIRA listener configuration screen. However, you can write your own configuration screen that can configure not only your listener but any other information your users might need to input. We suggest the tutorial:

[Writing an Admin Configuration Screen](#)



Congratulations, that's it

Have a chocolate!

Plugin Tutorial - Writing a Confluence Macro that Uses JSON



Level of experience: Intermediate

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'intermediate' level. If you have never developed a plugin before, you may find this one a bit difficult.



Looking for Confluence 4.0 macro help?

You can find tutorials for building Confluence 4.0 macros [\[here\]](#).

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- Overview
- Required Knowledge
- Step 1. Create the Plugin Project
- Step 2. Add Plugin Metadata to the POM
- Step 3. Register the Plugin Module in the Plugin Descriptor
 - Add the Web Resource Module
 - Add the Macro Module
- Step 4. Write Code
 - Write the Java class
 - Write a velocity template
 - Write javascript
- Step 5. Build, Install and Run the Plugin
- Step 6. Write Unit and Integration Tests

Overview

This tutorial shows you how Confluence macros can create `JSON` objects to exchange data that can be processed with javascript in the browser.

In order to do this, you will create a Confluence macro plugin that renders a table listing the next seven days starting from any given date. i.e. a macro that turns wiki markup like this:

```
{examplemacro:date=04/03/2010}
```

into this:

| DAY | DATE |
|-----------|-------------|
| Thursday | 04 Mar 2010 |
| Friday | 05 Mar 2010 |
| Saturday | 06 Mar 2010 |
| Sunday | 07 Mar 2010 |
| Monday | 08 Mar 2010 |
| Tuesday | 09 Mar 2010 |
| Wednesday | 10 Mar 2010 |

In itself, not a very useful macro, but hopefully useful for showing you how to create, exchange and use data in JSON.

As with all plugins, your plugin will consist of the following components:

- Java classes encapsulating the plugin logic
- Resources for display of the plugin UI
- Javascript for client side processing of JSON
- Plugin descriptor to enable the plugin module in Confluence

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Required Knowledge

To complete this tutorial, you must already understand the basics of Java development: classes, interfaces, methods, how to use the compiler, and so on. You should also understand:

- How to create an Atlassian plugin project using the [Atlassian Plugin SDK](#).
- How to open the plugin project in your IDE.
- How to compile your project and create a JAR file using Maven.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

When you create the plugin using the above SDK, you will be prompted for some information to identify your plugin. Here's how this tutorial plugin is configured, but you can use more meaningful information:

- **group-id**: com.some.id
- **artifact-id**: examplemacro
- **version**: 1.0
- **package**: com.some.id

Step 2. Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
<organization>
    <name>Example Company</name>
    <url>http://www.example.com/</url>
</organization>
```

3. Update the `<description>` element:

```
<description>This plugin illustrates how to use JSON in a macro</description>
```

4. Save the file.

Step 3. Register the Plugin Module in the Plugin Descriptor

Now you will add the required plugin modules to your plugin descriptor at `src/main/resources/atlassian-plugin.xml`. The plugin descriptor is an XML file that identifies the plugin to Confluence and defines the functionality that the plugin requires.

Here's a basic plugin descriptor:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}">
  plugins-version="2">
    <plugin-info>
      <description>${project.description}</description>
      <version>${project.version}</version>
      <vendor name="${project.organization.name}" url="${project.organization.url}" />
    </plugin-info>
  </atlassian-plugin>
```

Note that some of the information from the POM is transferred to the plugin descriptor using variable names such as `${project.artifactId}`.

You will need the following plugin modules:

Add the Web Resource Module

First, you will need to add a [Web Resource module](#) to define the downloadable resources for the macro. In this case, we're adding a velocity template called `examplemacro.vm` that will create the HTML for our macro as well as an `examplemacro.js` file that contains the macro's javascript.

```
<web-resource key="resources" name="exampleMacro resources">
  <resource type="velocity" name="template" location="examplemacro.vm"/>
  <resource type="download" name="examplemacro.js" location="examplemacro.js"/>
</web-resource>
```

Add the Macro Module

Secondly add a [Macro module](#) for your plugin. For this tutorial we only need to define a name for the macro and a single parameter "date".

```
<macro name="examplemacro" key="my-macro">
  <parameters>
    <parameter name="date" type="string"></parameter>
  </parameters>
</macro>
```

Now you are ready to move onto writing some code to make your plugin do something.

Step 4. Write Code

In this plugin, we are going to allow users to specify a date in Confluence, which the plugin will use to determine the seven days starting from that date, determine the days of the week on which those dates fall and then display a table containing both the days and the dates in Confluence. The macro is going to store the day and date data in JSON and use javascript to create a table and populate it with the JSON data. To do this, you will write a Java class, a velocity template and some javascript.

Write the Java class

Add the basics

```

public class ExampleMacro extends BaseMacro
{
    public boolean isInline()
    {
        return true;
    }

    public boolean hasBody()
    {
        return false;
    }

    public RenderMode getBodyRenderMode()
    {
        return RenderMode.NO_RENDER;
    }

    public String execute(Map params, String body, RenderContext renderContext) throws MacroException
    {
        // coming up
    }
}

```

For this example our macro will be rendered inline, contain no body and so have no body render mode.

Handle Dates

Define date formats

```

public class ExampleMacro extends BaseMacro
{
    DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    DateFormat dayFormat = new SimpleDateFormat("EEEE");
}

```

Add two date formats: One for formatting dates to dd/MM/yy format and one for formatting dates into the days of the week. We'll use these to format the data displayed to the user.

Determine the start date

```

public String execute(Map params, String body, RenderContext renderContext) throws MacroException
{
    Date startDate = getStartDateFromParams(params);
}

private Date getStartDateFromParams(Map params)
{
    Date startDate = new Date();
    if (params.size() > 1)
    {
        try
        {
            startDate = dateFormat.parse((String) params.get("date"));
        }
        catch (ParseException e)
        {
            e.printStackTrace();
        }
    }
    return startDate;
}

```

When a macro is invoked, it's the execute() method that gets called so that's where we'll be doing the heavy lifting. Confluence passes any macro parameters as Strings in the params Map to execute(), so here we are look for the date parameter and use its value to construct a startDate date object via the getStartDateFromParams() helper method called from execute(). Note that initializing the startDate object like this;

```
Date startDate = new Date();
```

sets the startDate to the current date, so we can use that as our default date if none is specified as a parameter. Also note that the params Map always contains a RAW_PARAMS_KEY key containing the raw parameter string, hence only parsing params when params.size is greater than 1.

Calculate the dates and days for all seven days

```
public String execute(Map params, String body, RenderContext renderContext) throws MacroException
{
    Date startDate = getStartDateFromParams(params);
    for (int i=0; i < 7; i++)
    {
        Calendar cal = Calendar.getInstance();
        cal.setTime(startDate);
        cal.add(Calendar.DATE, i );
        Date date = cal.getTime();
        String day = dateFormat.format(date);
    }
}
```

Here we're adding a for loop that takes the start date and loops through seven times, incrementing the date by one day each time. On each loop the date is formatted to the name of the day and stored in the day variable. Whilst this loop creates the dates and days we need it overwrites them on each loop so we need to extend this to store these values which we'll use do in JSON.

Store the dates and days in a JSON object

```
public String execute(Map params, String body, RenderContext renderContext) throws MacroException
{
    Date startDate = getStartDateFromParams(params);
    JSONObject dayDateJsonObject = new JSONObject();
    JSONArray dayDateJsonArray = new JSONArray();

    for (int i=0; i < 7; i++)
    {
        Calendar cal = Calendar.getInstance();
        cal.setTime(startDate);
        cal.add(Calendar.DATE, i );
        Date date = cal.getTime();
        String day = dateFormat.format(date);
        dayDateJsonArray.add(nextDayDateJsonObject(day, date));
    }

    dayDateJsonObject.setProperty("daysdates", dayDateJsonArray);
}

private JSONObject nextDayDateJsonObject(String day, Date date)
{
    JSONObject nextDayDateJsonObject = new JSONObject();
    nextDayDateJsonObject.setProperty("date", date);
    nextDayDateJsonObject.setProperty("day", day);
    return nextDayDateJsonObject;
}
}
```

Here we've extended execute() and added the nextDayDateJsonObject() method to create some JSON structures. Note that Confluence provides classes for easy creation of JSON objects and JSON arrays, so all there is to do is;

- Create a JSON object containing two name/value pairs (one for the date and one for the day). This is what nextDayDateJsonObject() does.
- Call nextDayDateJsonObject() in the for loop of execute() so that a JSON object is created for each of the seven dates.
- Add each of these objects to a JSON array named dayDateJsonArray.
- Add the JSON array filled with all seven JSON objects, as the value to the only property of another JSON object, dayDateJsonObject.

The end result being a single JSON object, dayDateJsonObject, that contains an array of seven JSON objects, each of which is a set of day and date name/value pairs.

Output the result

```

public String execute(Map params, String body, RenderContext renderContext) throws MacroException
{
    Date startDate = getStartDateFromParams(params);
    JSONObject dayDateJsonObject = new JSONObject();
    JSONArray dayDateJSONArray = new JSONArray();

    for (int i=0; i < 7; i++)
    {
        Calendar cal = Calendar.getInstance();
        cal.setTime(startDate);
        cal.add(Calendar.DATE, i );
        Date date = cal.getTime();
        String day = dateFormat.format(date);
        dayDateJSONArray.add(nextDayDateJsonObject(day, date));
    }

    dayDateJsonObject.setProperty("daysdates", dayDateJSONArray);

    VelocityContext contextMap = new VelocityContext(MacroUtils.defaultVelocityContext());
    if(RenderContext.DISPLAY.equals(renderContext.getOutputType()))
    {
        contextMap.put(DAY_DATE_JSON, GeneralUtil.urlEncode(dayDateJsonObject.serialize()));
    }
    return VelocityUtils.getRenderedTemplate(TEMPLATE, contextMap);
}

```

Probably you'd want to make your JSON available via REST, but REST would require its own tutorial, so to keep this simple, this uses a velocity template to render our output as markup. To do this:

```
contextMap.put(DAY_DATE_JSON, GeneralUtil.urlEncode(dayDateJsonObject.serialize()));
```

1. Serializes the dayDateJsonObject JSON object into a string
2. URL encodes it
3. Puts it into a contextMap

This is then rendered into a String containing markup according to the velocity template we'll create next.

The completed class

[Expand to view entire Java class](#)

```

package com.some.id;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Map;
import com.atlassian.confluence.json.JSONArray;
import com.atlassian.confluence.json.JSONObject;
import com.atlassian.confluence.renderer.radeox.macros.MacroUtils;
import com.atlassian.confluence.util.GeneralUtil;
import com.atlassian.confluence.util.velocity.VelocityUtils;
import com.atlassian.renderer.RenderContext;
import com.atlassian.renderer.v2.macro.BaseMacro;
import com.atlassian.renderer.v2.macro.MacroException;
import com.atlassian.renderer.v2.RenderMode;
import org.apache.velocity.VelocityContext;

public class ExampleMacro extends BaseMacro
{
    DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    DateFormat dayFormat = new SimpleDateFormat("EEEE");
    private static final String DAY_DATE_JSON = "dayDateJson";
    private static final String TEMPLATE = "examplemacro.vm";

    public boolean isInline()
    {
        return true;
    }
}

```

```

    }

    public boolean hasBody()
    {
        return false;
    }

    public RenderMode getBodyRenderMode()
    {
        return RenderMode.NO_RENDER;
    }

    public String execute(Map params, String body, RenderContext renderContext) throws
MacroException
    {
        Date startDate = getStartDateFromParams(params);
        JSONObject dayDateJsonObject = new JSONObject();
        JSONArray dayDateJSONArray = new JSONArray();

        for (int i=0; i < 7; i++)
        {
            Calendar cal = Calendar.getInstance();
            cal.setTime(startDate);
            cal.add(Calendar.DATE, i );
            Date date = cal.getTime();
            String day = dateFormat.format(date);
            dayDateJSONArray.add(nextDayDateJsonObject(day, date));
        }

        dayDateJsonObject.setProperty("daysdates", dayDateJSONArray);

        VelocityContext contextMap = new VelocityContext(MacroUtils.defaultVelocityContext());
        if(RenderContext.DISPLAY.equals(renderContext.getOutputType()))
        {
            contextMap.put(DAY_DATE_JSON, GeneralUtil.urlEncode(dayDateJsonObject.serialize()));
        }

        return VelocityUtils.getRenderedTemplate(TEMPLATE, contextMap);
    }

    private Date getStartDateFromParams(Map params)
    {
        Date startDate = new Date();
        if (params.size() > 1)
        {
            try
            {
                startDate = dateFormat.parse((String) params.get("date"));
            }
            catch (ParseException e)
            {
                e.printStackTrace();
            }
        }
        return startDate;
    }

    private JSONObject nextDayDateJsonObject(String day, Date date)
    {
        JSONObject nextDayDateJsonObject = new JSONObject();
        nextDayDateJsonObject.setProperty("date",date);
        nextDayDateJsonObject.setProperty("day", day);
    }
}

```

```

        return nextDayDateJsonObject;
    }
}

```

Write a velocity template

```

<!--
#requireResource( "confluence.web.resources:jquery" )
#requireResource( "com.some.id.examplemacro:resources" )
-->

    <fieldset class="parameters hidden">
        <input type="hidden" class="dayDates" value="$dayDateJson">
    </fieldset>


```

Here we insert both the javascript for this macro as well as the jQuery javascript framework, included with Confluence, which our javascript depends on.

Then we define the markup for our macro which consists of a single `` element, inside which we have a hidden `<input>` to which we assign the output of our macro's `execute()` method i.e. our url encoded serialized JSON object string.

Write javascript

```

jQuery(function ($) {
    var initExampleMacro = function ()
    {
        $(".exampleMacro").each(function()
        {
            var dayDates = $(this).find("input.dayDates").val();
            var json;

            // determine if the browser has native JSON parser support & create JSON object
            if (typeof (JSON) !== 'undefined' && typeof (JSON.parse) === 'function')
            {
                json = JSON.parse(decodeURIComponent(dayDates).replace(/\+/g, '\u00a0'));
            } else {
                json = eval('(' + decodeURIComponent(dayDates).replace(/\+/g, '\u00a0') + ')');
            }

            // create table
            var html = "<table border=\"1\"><tr><th>DAY</th><th>DATE</th></tr>";
            for (var i=0; i<7; i++)
            {
                html = html + "<tr><td>" + json.daysdates[i].day + "</td><td>" +
                json.daysdates[i].date + "</td></tr>";
            }
            html = html + "</table>";
            $(this).html(html);
        });
    };

    $(document).ready(function()
    {
        initExampleMacro();
    });
});

```

This javascript uses jQuery to initialise each element belonging to the `exampleMacro` class once the page is ready. This entails:

1. Retrieving the url encoded, serialized JSON String from the hidden `<input>` element.
2. Creating a JSON object from that string using the browser's native JSON parser if it has one (more secure) or `eval()` if it doesn't, replacing any spaces with non-breaking spaces in the process.
3. Creating HTML for a table with a row for each of the seven dates and two columns for the dates and days, populating each cell with the relevant values from the JSON object
4. Inserting the HTML into the page.

Note: You should use velocity to create HTML instead of javascript like this. This is just an example to illustrate how to use values from JSON objects.

Step 5. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Step 6. Write Unit and Integration Tests

Always! Refer to the following tutorials on how to:

- Write unit tests for your plugin
- Write integration tests for your plugin



Congratulations, that's it

Have a chocolate!

Plugin Tutorial - Writing a Confluence Theme



Level of experience: Beginner

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'beginner' level, so you can follow it even if you have never developed a plugin before.



The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- Overview
- Required Knowledge
- Step 1. Create the Plugin Project
- Step 2. Add Plugin Metadata to the POM
- Step 3. Register the Plugin Module in the Plugin Descriptor
 - Create the Theme by Adding the Plugin Module to the XML File
- Step 4. Build, Install and Run the Plugin
- Step 5. Enable the Theme in Confluence
- Step 6. Customise the Theme
 - Add a Theme Style
 - Add a Colour Scheme
 - Add a Custom Layout

Overview

This tutorial shows you how to create a theme for Confluence.

In order to do this, you will create a Confluence plugin. As with most plugins, your plugin will consist of the following components:

- Plugin descriptor to enable the plugin module in Confluence
- Resources for display of the plugin UI

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Required Knowledge

To complete this tutorial, you should have some basic understanding of web development. Knowing the template language [Velocity](#) is beneficial, but not an absolute requirement. You should also understand:

- How to create an Atlassian plugin project using the [Atlassian Plugin SDK](#).
- How to open the plugin project in your editor or IDE.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

When you create the plugin using the above SDK, you will be prompted for some information to identify your plugin. Enter the following information:

- **group-id**: com.example.plugins.tutorial
- **artifact-id**: simpletheme
- **version**: 1.0
- **package**: com.example.plugins.tutorial

Step 2. Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
<organization>
    <name>Example Company</name>
    <url>http://www.example.com/</url>
</organization>
```

3. Update the `<description>` element:

```
<description>A simple example theme</description>
```

4. Save the file.

Step 3. Register the Plugin Module in the Plugin Descriptor

Now you will add the required plugin modules to your plugin descriptor at `src/main/resources/atlassian-plugin.xml`. The plugin descriptor is an XML file that identifies the plugin to Confluence and defines the functionality that the plugin requires.

Here's a basic plugin descriptor:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}"
plugins-version="2">
    <plugin-info>
        <description>${project.description}</description>
        <version>${project.version}</version>
        <vendor name="${project.organization.name}" url="${project.organization.url}" />
    </plugin-info>
</atlassian-plugin>
```

Note that some of the information from the POM is transferred to the plugin descriptor using variable names such as `${project.artifactId}`.

Create the Theme by Adding the Plugin Module to the XML File

In this plugin, you want to create a simple theme for Confluence. To do this, you will write a theme module that provides the basis for the Confluence theme.

```
<theme key="simpletheme" name="Simple Theme" class="com.atlassian.confluence.themes.BasicTheme">
    <description key="A simple example theme"/>
    <param name="includeClassicStyles" value="false"/>
    <resource type="download" name="default-theme.css" location="/includes/css/default-theme.css">
        <param name="source" value="webContext"/>
    </resource>
</theme>
```

Here we created a simple theme module with the unique key **simpletheme**, and the optional name **Simple Theme**. The theme module also requires a Java class that provides some basic functionality. But don't worry, for this tutorial, and in fact for most themes, it is enough to simply use the **BasicTheme** class that ships with Confluence. Just copy the class attribute from the example above.

We also provided a description in our example, which will be displayed on the Themes administration screen.

For backwards compatibility, Confluence also includes its classic styles by default. Since we are writing a new theme we don't care about these styles and choose to remove them using the `includeClassicStyles` parameter.

Lastly we don't want to start from scratch. Let's base our new theme on the default Confluence style and include the default-theme CSS. We simply need to define a theme resource for this and point it to the existing CSS file within Confluence. Again, just copy the example above, and don't scratch your head wondering where the location comes from. You would have to look at the Confluence source code to find it.

Step 4. Build, Install and Run the Plugin

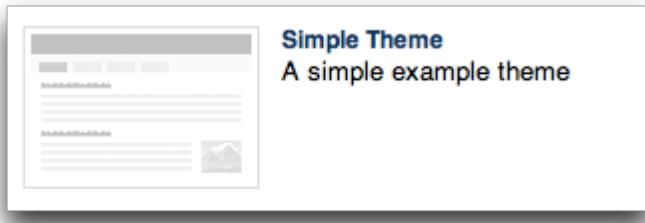
Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Step 5. Enable the Theme in Confluence

After installing the plugin, we can now enable the theme via the Confluence Administration Console. Click on the theme link in the left-hand navigation and you should see the **Simple Theme** option in the list of themes. Select it via the radio button and confirm.



Don't be surprised if nothing has changed. So far we have created a basic theme that looks exactly like the default Confluence theme. Now we are ready to make some customisations.

Step 6. Customise the Theme

Add a Theme Style

We want to change the style of the header bar. We can do this with some custom CSS styles that we can include with the theme. Similar to the default theme's styles, we will create a resource within the theme tag to specify where to find the CSS file. But this time the file is located within our plugin.

```
<resource type="download" name="theme.css" location="simpletheme/theme.css" />
```

Again the resource is of type "download" and we will name the resource simply **theme.css**. We will place the actual CSS file in a folder called **simpletheme** within the resources directory to ensure it doesn't conflict with other plugins. Note that the name of the resource and the actual file name don't have to be the same.

See the documentation on web resource modules for more details about web resources.



All resources are defined relative to the resources directory within the plugin folder. Do not use a leading "/". In our example theme the folder structure looks like this:

```
simpletheme-plugin/src/main/resources
```

Let's open the CSS file we've created in the `simpletheme` folder and add some fancy CSS.

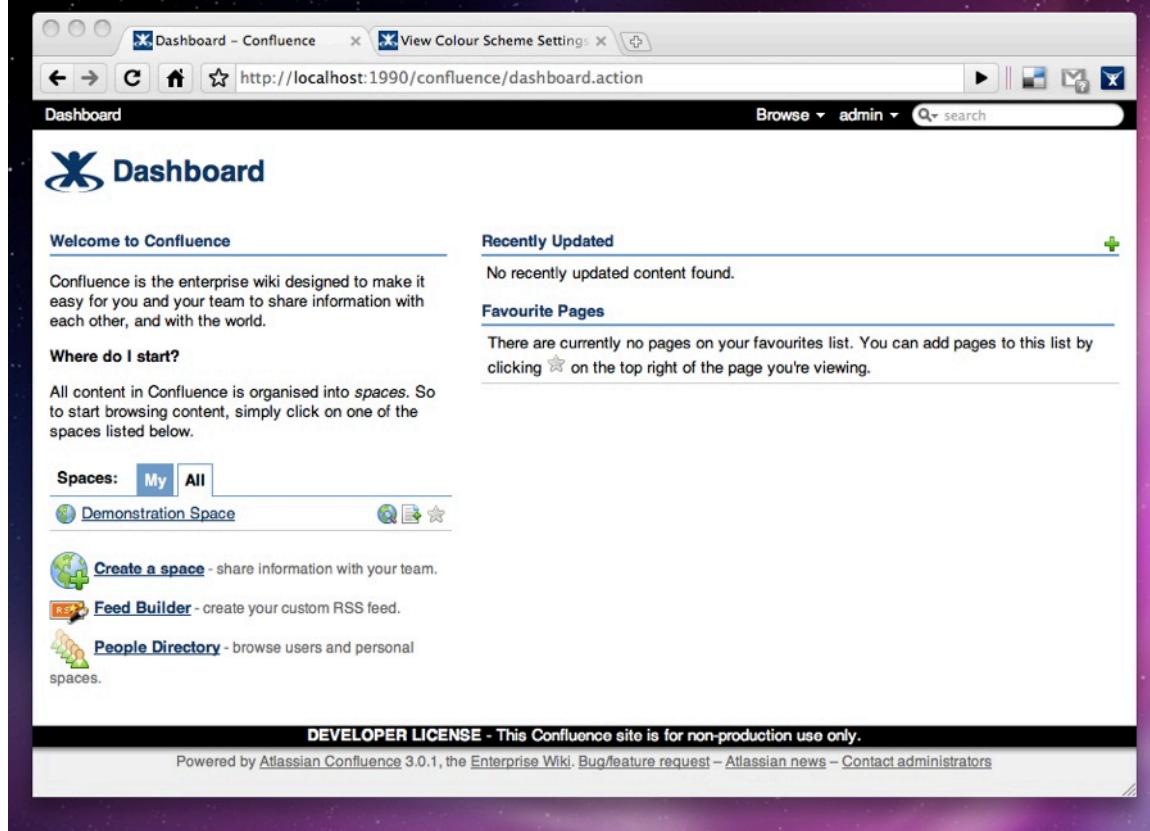
```
#header, #footer p.license{  
    background-color: #000;  
}
```

Ok, it's not that fancy. But you get the gist. The above CSS style will change the color of the header and the footer from the default blue to

black.

Now install the plugin into Confluence again. If you are using the CLI, this is as simple as running `pi` in the command window.

Your Dashboard should now look like the picture below with a black header bar. If it does not, ensure that the simple theme is installed and selected in the Confluence Administration Console.



Add a Colour Scheme

You may choose to provide your theme with a pre-defined colour scheme that users will be able to select under Space Administration. This pre-defined colour scheme will take precedence if no custom user colour scheme is defined. To define a theme's colour scheme, you need to add a **colour scheme module** to the `atlassian-plugin.xml` file and link to it in the **theme module**. For example:

```
<!-- Colour scheme module -->
<colour-scheme key="earth-colours" name="Brown and Red Earth Colours"
class="com.atlassian.confluence.themes.BaseColourScheme">
    <colour key="property.style.topbarcolour" value="#440000"/>
    <colour key="property.style.spacenamecolour" value="#999999"/>
    <colour key="property.style.headingtextcolour" value="#663300"/>
    <colour key="property.style.linkcolour" value="#663300"/>
    <colour key="property.style.bordercolour" value="#440000"/>
    <colour key="property.style.navbgcolour" value="#663300"/>
    <colour key="property.style.navtextcolour" value="#ffffff"/>
    <colour key="property.style.navselectedbgcolour" value="#440000"/>
    <colour key="property.style.navselectedtextcolour" value="#ffffff"/>
</colour-scheme>
```

We have defined some new colours. Now we will have to add them to the theme. Within the theme module tag we add another tag to link the colour scheme:

```
...
<colour-scheme key="${project.groupId}.${project.artifactId}:earth-colours"/>
...
```

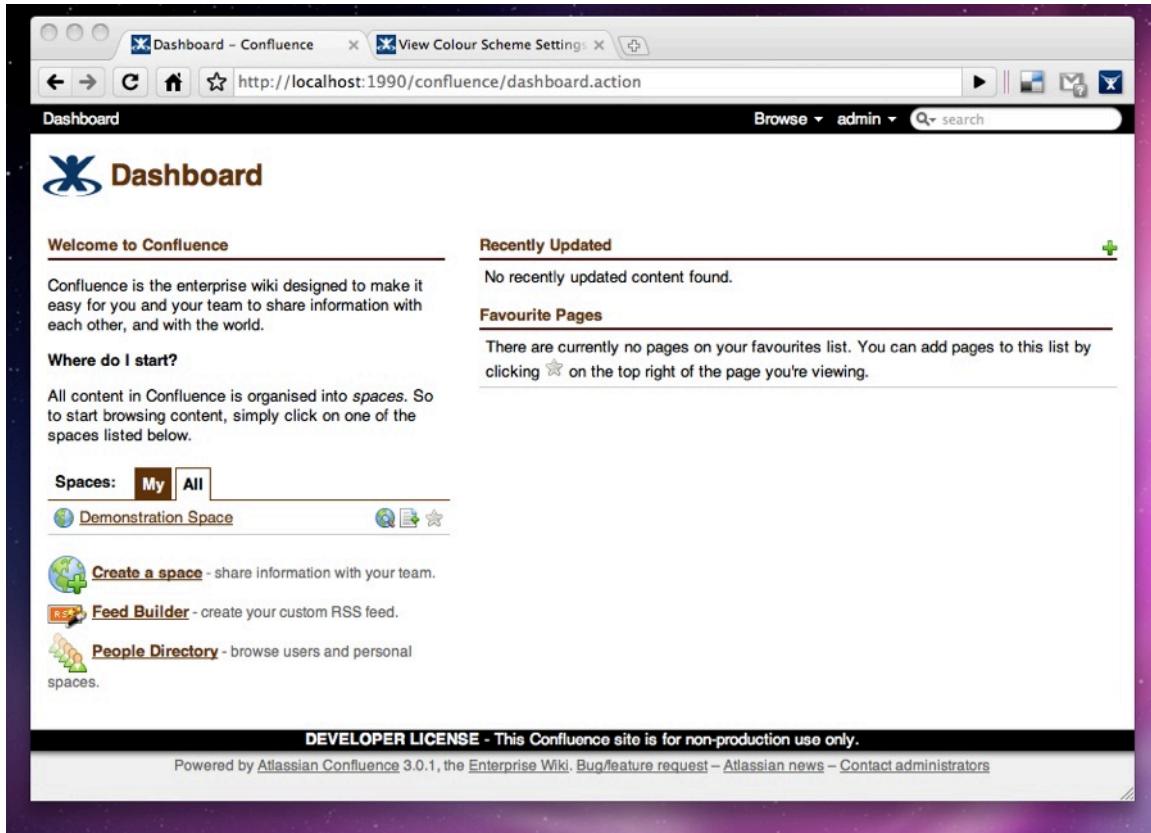
To link to the colour scheme module we will have to use the module key (earth-colours) and prefix it with the full plugin key.



Note how we are using variables for the project **groupId** and the **artifactId**. We recommend that you always use these variables. They are automatically replaced when you build the plugin with the SDK.

Now install the plugin into Confluence again. If you are using the CLI, this is as simple as running `pi` in the command window.

Looking at the Dashboard again, you will see the colours of headings, links and tabs have changed to brown and red earth colours. Note that you might have to select the theme colour scheme in the Confluence Administration Console before the change becomes visible.



Add a Custom Layout

The layout of a theme can be customised using the above discussed theme stylesheets. However, in order to add new elements and functionality to a theme you will have to customise the theme decorators using the **layout module**.



Decorators often change between major versions of Confluence. If you customise a decorator in a theme you will have to maintain the customisation going forward to be compatible with future versions of Confluence

Once again we will have to define a new module in our plugin descriptor and link it within the theme module tag. Let's start by defining the layout module:

```
<layout key="main" name="Main Decorator" class="com.atlassian.confluence.themes.VelocityDecorator"  
overrides="/decorators/main.vmd">  
    <resource type="velocity" name="decorator" location="simpletheme/main.vmd" />  
</layout>
```

We want to override the main decorator in Confluence which is located under `/decorators/main.vmd`. We also have to specify a Java class, but just as above, we don't worry about changing it and simply use the default provided. We also have to define a resource which we will use to override the default decorator. Unlike the CSS files in other resources we have defined so far, this time the resource is of type **velocity**. Now we will copy the actual `main.vmd` file from the Confluence source (located under: `confluence/conf-webapp/src/main/webapp/decorators`) and place it into our `simpletheme` folder within the resources directory.

At this stage, we have the base layout without any changes. Let's add a global navigation to our theme. To do this we want to find the `<div>` with the id **header** and add our new navigation above it.

```

...
<div id="navigation">
<ul>
    <li><a href="http://atlassian.com">Atlassian</a></li>
    <li><a href="http://confluence.atlassian.com">Documentation</a></li>
    <li><a href="http://jira.atlassian.com">Bugs & Feature Requests</a></li>
</ul>
</div>
## CONTENT DIV BEGINS
<div id="header">
...

```

Here we simply added a hard-coded list of links to some important websites. But it won't look pretty without any additional styling. Let's add a couple more styles to our **theme.css** file.

```

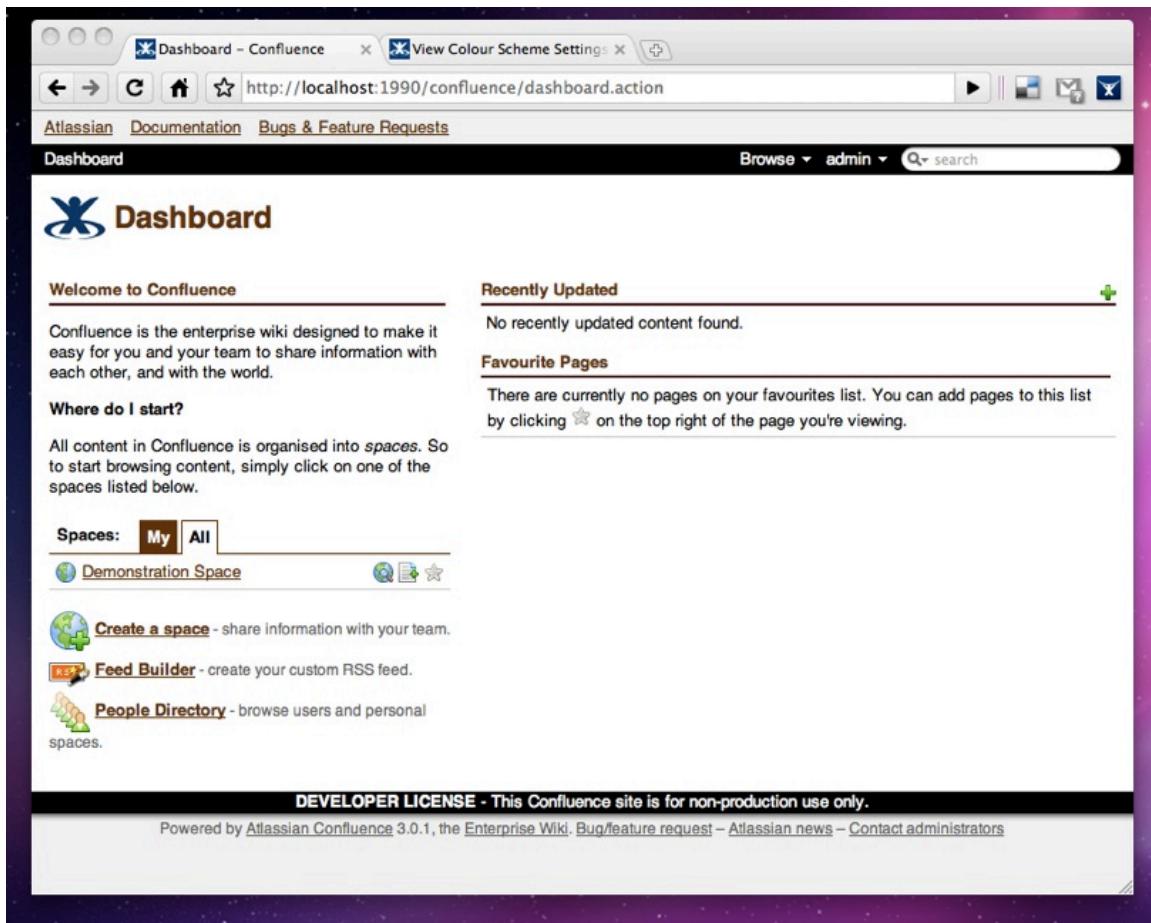
#navigation ul{
    margin: 5px 0px;
    padding: 0px;
}
#navigation li{
    list-style: none;
    display: inline;
    padding-left: 10px;
}

```

This will do.

Now install the plugin into Confluence again. If you are using the CLI, this is as simple as running `pi` in the command window.

You should see a list of links above the header when looking at the Dashboard.



Congratulations, that's it

You have just written your first Confluence theme. Use more complicated stylesheets or further customise the layouts to make Confluence look the way you like it.

Plugin tutorial - Writing an Admin Configuration Screen



Level of experience: Advanced

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'advanced' level. If you have never developed a plugin before, we advise you to try a beginner tutorial first.

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- Overview
- Create the Plugin Project
- Add Plugin Metadata to the POM
- Render the form
 - UI servlet
 - Necessary changes to `atlassian-plugin.xml`
 - Velocity template
 - Start with the basics
 - Add some decoration
 - Style it
 - I18n it
- JavaScript and REST
 - Populating the form
 - Preparation
 - JavaScript for GETting the configuration
 - Serving the configuration from a REST resource
 - Updating the configuration
 - JavaScript for PUTting an updated configuration
 - Updating the configuration with PUT
 - Wiring things together
- Menu items
- Build, Install and Run the Plugin

Overview

This tutorial shows you how to create an admin UI that can be used in any Atlassian application.

In order to do this, you will create a RefApp plugin. (See our page about the [Atlassian RefApp](#).) As with all plugins, your plugin will consist of the following components:

- Java classes encapsulating the plugin logic
- Resources for display of the plugin UI
- Plugin descriptor to enable the plugin module in Atlassian applications

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

We'll be making extensive use of [SAL](#), which exports a bunch of services that can be used for persistence, user authorization, and other common tasks in a way that will work in any Atlassian application.

For rendering the form, we'll use the [Atlassian Template Renderer](#). ATR is a plugin that provides services to other plugins that allow them to render templates, typically in Velocity.

For styling our forms to make them look good and for communicating with the server via ajax, we'll be using [AUI](#).

To create our REST resources, which our JavaScript will communicate with, we'll use the [REST module](#).

Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

When you create the plugin using the above SDK, you will be prompted for some information to identify your plugin. Enter the following information:

- **group-id**: com.atlassian.plugins.tutorial
- **artifact-id**: xproduct-admin-ui-plugin
- **version**: 1.0
- **package**: com.atlassian.plugins.tutorial

Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
<organization>
    <name>Example Company</name>
    <url>http://www.example.com/</url>
</organization>
```

3. Update the `<description>` element:

```
<description>This plugin has an admin UI that can be used in any Atlassian
product.</description>
```

4. Save the file.

Render the form

We'll start by getting the admin form rendering. It won't do anything yet, but it's always nice to start with something we can actually see and add to it as you go.

UI servlet

This is the servlet that will be responsible for rendering our simple administration form.

We'll start with a bare skeleton.

```
package com.atlassian.plugins.tutorial.xproductadminui;

import javax.servlet.http.HttpServlet;

import com.atlassian.sal.api.auth.LoginUriProvider;
import com.atlassian.sal.api.user.UserManager;
import com.atlassian.template.renderer.TemplateRenderer;

public class AdminServlet extends HttpServlet
{
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    IOException, ServletException
    {
    }
}
```

The first thing we want to add is a check that the user is an admin. We'll use the SAL `UserManager` to do that. But before we can do that we'll need to add a dependency on the `sal-api` to our `pom.xml`. Find the `dependencies` section and add the following dependency.

```
<dependency>
    <groupId>com.atlassian.sal</groupId>
    <artifactId>sal-api</artifactId>
    <version>2.0.16</version>
    <scope>provided</scope>
</dependency>
```

Now that SAL is part of our build environment, we can proceed.

```

package com.atlassian.plugins.tutorial.xproductadminui;

import javax.servlet.http.HttpServlet;

import com.atlassian.sal.api.user.UserManager;

public class AdminServlet extends HttpServlet
{
    private final UserManager userManager;

    public AdminServlet(UserManager userManager)
    {
        this.userManager = userManager;
    }

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException
    {
        String username = userManager.getRemoteUsername(request);
        if (username != null && !userManager.isSystemAdmin(username))
        {
            redirectToLogin(request, response);
            return;
        }
    }
}

```

We've added a dependency on SAL's `UserManager`, and we use that in our `doGet` method to check that the user is logged in - i.e. the `username` is not `null` - and that the user is an administrator. If they are not logged in or are not an administrator, we call a helper method to redirect them to the application login page. The `redirectToLogin` helper method is below.

```

public class AdminServlet extends HttpServlet
{
    private final UserManager userManager;
    private final LoginUriProvider loginUriProvider;

    public AdminServlet(UserManager userManager, LoginUriProvider loginUriProvider)
    {
        this.userManager = userManager;
        this.loginUriProvider = loginUriProvider;
    }

    // doGet ellided for brevity

    private void redirectToLogin(HttpServletRequest request, HttpServletResponse response) throws
IOException
    {
        response.sendRedirect(loginUriProvider.getLoginUri(getUri(request)).toASCIIString());
    }

    private URI getUri(HttpServletRequest request)
    {
        StringBuffer builder = request.getRequestURL();
        if (request.getQueryString() != null)
        {
            builder.append("?");
            builder.append(request.getQueryString());
        }
        return URI.create(builder.toString());
    }
}

```

The `redirectToLogin` method uses another service provided by SAL, the `LoginUriProvider`. This service allows us to construct a URI to the applications login page and provide it with a URI parameter that will be where the user is redirected to after they log in. We pass it the current URI so that the user is redirected back to our admin UI when they are properly authenticated. We also added the `LoginUriProvider` to our constructor so that it will be injected by the plugin system.

Now, we're ready to render our form. As mentioned in the [Overview](#), we'll use the Atlassian Template Renderer for that. Once again, before we can write the code, we need to tell Maven that we will be using it. Find the `dependencies` section in the `pom.xml` file and add the following dependency

```

<dependency>
    <groupId>com.atlassian.templaterenderer</groupId>
    <artifactId>atlassian-template-renderer-api</artifactId>
    <version>1.1.1</version>
    <scope>provided</scope>
</dependency>

```

Now let's get to rendering!

```

public class AdminServlet extends HttpServlet
{
    private final UserManager userManager;
    private final TemplateRenderer renderer;
    private final LoginUriProvider loginUriProvider;

    public AdminServlet(UserManager userManager, LoginUriProvider loginUriProvider,
TemplateRenderer renderer)
    {
        this.userManager = userManager;
        this.loginUriProvider = loginUriProvider;
        this.renderer = renderer;
    }

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException
    {
        String username = userManager.getRemoteUsername(request);
        if (username != null && !userManager.isSystemAdmin(username))
        {
            redirectToLogin(request, response);
            return;
        }

        response.setContentType("text/html;charset=utf-8");
        renderer.render("admin.vm", response.getWriter());
    }

    // redirectToLogin ellided for brevity
}

```

We've added a dependency on a TemplateRenderer to our constructor. This will be the renderer we use to display our form. In the doGet method, after doing our authorization check, we set the response content type to text/html and then we make the simple call to the templateRenderer telling it to render our admin.vm template, which we will create shortly.

Necessary changes to atlassian-plugin.xml

Before we go any further, let's tell the plugin system about our servlet and its dependencies. We need to add the following modules to our atlassian-plugin.xml file.

```

<component-import key="userManager" interface="com.atlassian.sal.api.user.UserManager" />
<component-import key="loginUriProvider" interface="com.atlassian.sal.api.auth.LoginUriProvider"
/>
<component-import key="renderer"
interface="com.atlassian.templaterenderer.velocity.one.six.VelocityTemplateRenderer" />

<servlet key="admin-servlet" class="com.atlassian.plugins.tutorial.xproductadminui.AdminServlet">
    <url-pattern>/xproduct/admin</url-pattern>
</servlet>

```

With that done, we can startup the RefApp by running the atlas-run command. If we visit <http://localhost:5990/refapp/plugins/servlet/xproduct/admin> we'll see the output of our servlet. But there won't be much to see except a big exception saying that the admin.vm file couldn't be found. So let's create it.

Velocity template

Start with the basics

Start off by creating a dead simple page with a form on it

```

<html>
  <head>
    <title>XProduct Admin</title>
  </head>
  <body>
    <form id="admin">
      <div>
        <label for="name">Name</label>
        <input type="text" id="name" name="name" />
      </div>
      <div>
        <label for="time">Time</label>
        <input type="text" id="time" name="time" />
      </div>
      <div>
        <input type="submit" value="Save" />
      </div>
    </form>
  </body>
</html>

```

That gets us...

Name:

Time:

Save

something. But it's something that is completely undecorated. The application header, footer and menus are gone and the form itself is, well, ugly. Let's fix that.

Add some decoration

To get the application to decorate the page, we'll add a `meta` tag in the `head` element, after the title.

```
<meta name="decorator" content="atl.admin">
```

This tells the application that it needs to use the "atl.admin" decorator around the body of this page. Now it looks like something the application would actually display...



but the form itself still looks terrible. AUI to the rescue!

Style it

First we need to include AUI in our header. Just add the following line after the `meta` tag.

```
$webResourceManager.requireResource( "com.atlassian.auiplugin:ajs" )
```

What is `$webResourceManager` and where does it come from? It's the `WebResourceManager` from the plugin system. And calling `requireResource` will include all the resources for the web-resource module plus all of its dependencies.

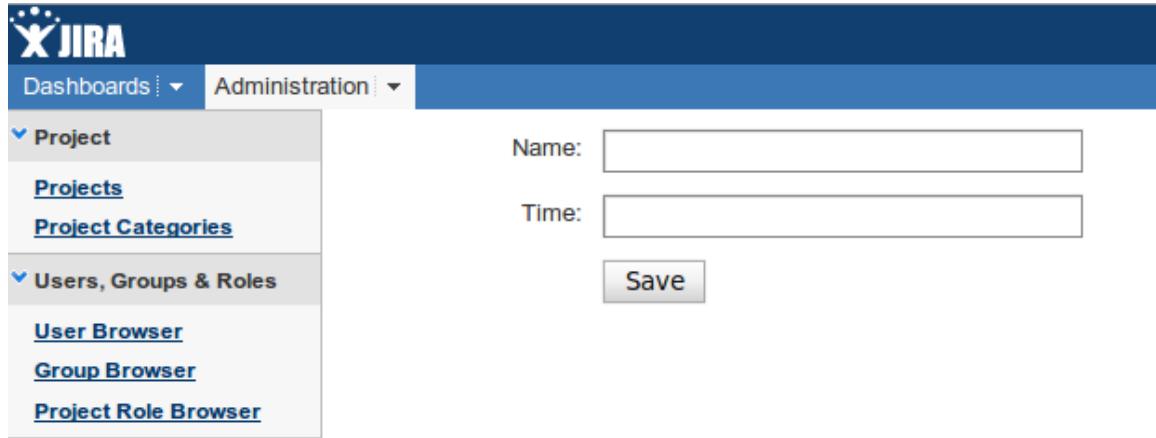
But where did the `WebResourceManager` come from? It was automatically added to the Velocity context by the `TemplateRenderer`. It

makes a few things that are commonly useful available, like the `WebResourceManager`. We'll see another one that is included automatically below, when we get to internationalizing our template.

Next, we need to change our form a little bit, adding class attributes.

```
<form id="admin" class="aui">
  <div>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" class="text">
  </div>
  <div>
    <label for="time">Time:</label>
    <input type="text" id="time" name="time" class="text">
  </div>
  <div>
    <input type="submit" value="Save" class="button">
  </div>
</form>
```

The changes were small. We added `class="aui"` to the `form` element, `class="text"` to each of our text input boxes, and `class="button"` to our submit button. And for such little work we get...



a nice looking form in any Atlassian application that is consistent with the application look and feel!

I18n it

We can also add internationalization pretty easily. First, we'll add a `resource` module specifying our i18n properties file in `atlassian-plugin.xml`

```
<resource type="i18n" name="i18n" location="com.atlassian.tutorial.xproductadminui.i18n" />
```

Next, we create the properties file with the text that appears in the form in `src/main/resources/com/atlassian/tutorial/xproductadminui/i18n.properties` with the following properties defined.

```
xproduct.admin.label=XProduct Admin
xproduct.admin.name.label=Name
xproduct.admin.time.label=Time
xproduct.admin.save.label=Save
```

Now, we replace the hard-coded text values in our template with lookups for the internationalized text.

```

<html>
  <head>
    <title>$i18n.getText("xproduct.admin.label")</title>
    <meta name="decorator" content="atl.admin" />
    $webResourceManager.requireResource("com.atlassian.auiplugin:ajs")
  </head>
  <body>
    <form id="admin" class="aui">
      <div>
        <label for="name">$i18n.getText("xproduct.admin.name.label")</label>
        <input type="text" id="name" name="name" class="text">
      </div>
      <div>
        <label for="time">$i18n.getText("xproduct.admin.time.label")</label>
        <input type="text" id="time" name="time" class="text">
      </div>
      <div>
        <input type="submit" value="$i18n.getText("xproduct.admin.save.label")" class="button">
      </div>
    </form>
  </body>
</html>

```

\$i18n is an instance of SAL's [I18nResolver](#), which is automatically provided by the [TemplateRenderer](#) just like the [WebResourceManager](#). We use it to resolve our message keys to actual text. Now, all that is left to do for full internationalization is getting our properties file translated.

We have our basic UI. It's time to make it do something!

JavaScript and REST

Our form is pretty simple. It doesn't include any data and our servlet doesn't support POSTs for updates. To display data in the form and update the configuration, we'll instead be using JavaScript to make ajax requests to a REST resource.

Populating the form

To populate the form we will make a GET request to a resource that will return the configuration as JSON.

Preparation

Before we can start writing our JavaScript in earnest there are a few things we need to do. We need to know the applications base URL for making our ajax requests. We also need to tweak our template a little bit so that it will include the JavaScript that we're writing in addition to the AUI stuff.

What's my address?

To find out what the application base URL is, we'll use SAL's [ApplicationProperties](#). Before we can use it we need to import it, so we'll need to add

```

<component-import key="applicationProperties"
  interface="com.atlassian.sal.api.ApplicationProperties" />

```

to the `atlassian-plugin.xml` file. We'll want to be able to use the `ApplicationProperties` object from our template, so we should also add a `template-content-item`

```

<template-context-item key="applicationPropertiesContextItem"
  component-ref="applicationProperties"
  context-key="applicationProperties" name="Application Properties Context Item"/>

```

`template-context-item`s allow us to put objects into our Velocity or other renders context so that we can use them in our templates. As an alternative, we could pass the `ApplicationProperties` component into our servlet, which is where we'll do the rendering from, and pass it into the rendering context when we call the `render` method. This just makes that automatic, and if we were using the renderer in multiple places, makes the `ApplicationProperties` object available in all contexts without a bunch of duplicated code.

Now we're ready to add the base URL to our template so that it can be looked up from JavaScript. Add the following `meta` element to the template head.

```
<meta name="application-base-url" content="$applicationProperties.getBaseUrl()">
```

We use the `ApplicationProperties.getBaseUrl()` method to find the application base URL.

Add our JavaScript to the template

The recommended way to add JavaScript to pages is to create a web-resource module in your `atlassian-plugin.xml` file. Using this method, you can also specify dependencies on other web-resource modules, like AUI. Add the following to the `atlassian-plugin.xml` file.

```
<web-resource name="Admin Web Resources" key="resources">
    <dependency>com.atlassian.auiplugin:ajs</dependency>
    <resource type="download" name="admin.js" location="admin.js"/>
</web-resource>
```

Now edit the `admin.vm` template, and change the line that uses the `$webResourceManager.requireResource` from

```
$webResourceManager.requireResource("com.atlassian.auiplugin:ajs")
```

to

```
$webResourceManager.requireResource("com.atlassian.plugins.tutorial.xproduct-admin-ui-plugin:resource")
```

Now we can write some JavaScript!

JavaScript for GETting the configuration

Populating our form is pretty simple. AUI includes jQuery. We'll make use of jQuery's `ajax` method for communicating with our REST resources. We'll start out by making a GET request and setting the values of the input fields. Create the file `src/main/resources/admin.js` and add the following

```
AJS.toInit(function() {
    var baseUrl = AJS.$("meta[name='application-base-url']").attr("content");

    function populateForm() {
        AJS.$().ajax({
            url: baseUrl + "/rest/xproduct-admin/1.0/",
            dataType: "json",
            success: function(config) {
                AJS.$("#name").attr("value", config.name);
                AJS.$("#time").attr("value", config.time);
            }
        });
    }

    populateForm();
});
```

This is pretty straight forward JavaScript if you are familiar with jQuery. `AJS.toInit` is equivalent to using `jQuery(function() { ... })` or `jQuery(document).ready(function() { ... })` to have some JavaScript executed when the document is ready. Our first task is to find the application base URL in the document that we added previously. Having done that, we can go on to query the server to find the current config. Once we have the config, we can populate our data.

Next, we'll write the REST resource that will serve this data.

Serving the configuration from a REST resource

The JavaScript that we've written makes a request to a configuration resource for populating our form. Since we haven't done it yet, we need to create this resource. Before we can go and write the code for that, we need to add a few more dependencies in Maven. The [Atlassian REST module](#) uses Jersey, an implementation of the [JAX-RS API](#). We need to add a dependency on the JAX-RS API.

We'll also be using JAXB to serialize our configuration object for us, so that we don't have to manually construct JSON. So we need to add a dependency for that as well.

So open up the `pom.xml` file and find the `dependencies` section and add the following dependency elements.

```

<dependency>
    <groupId>javax.ws.rs</groupId>
    <artifactId>jsr311-api</artifactId>
    <version>1.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
</dependency>

```

Now we can write our REST resource class.

Retrieving the configuration with GET

Lets start with a simple class that our dependencies get injected into.

```

package com.atlassian.plugins.tutorial.xproductadminui;

import javax.ws.rs.Path;

import com.atlassian.sal.api.pluginsettings.PluginSettingsFactory;
import com.atlassian.sal.api.transaction.TransactionTemplate;
import com.atlassian.sal.api.user.UserManager;

@Path("/")
public class ConfigResource
{
    private final UserManager userManager;
    private final PluginSettingsFactory pluginSettingsFactory;
    private final TransactionTemplate transactionTemplate;

    public ConfigResource(UserManager userManager, PluginSettingsFactory pluginSettingsFactory,
    TransactionTemplate transactionTemplate)
    {
        this.userManager = userManager;
        this.pluginSettingsFactory = pluginSettingsFactory;
        this.transactionTemplate = transactionTemplate;
    }
}

```

To populate the form, we'll do a GET request to this resource that returns the data as JSON. We'll let JAXB serialize our configuration object for us, to make life easier. First, lets add an inner class that encapsulates our configuration data.

```

@XmlRootElement
public static final class Config
{
    @XmlElement private String name;
    @XmlElement private int time;

    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public int getTime()
    {
        return time;
    }

    public void setTime(int time)
    {
        this.time = time;
    }
}

```

Above we've created a class that will encapsulate a few properties, a String and an int. The `@XmlRootElement` and `@XmlElement` annotations are JAXB annotations. The `@XmlRootElement` annotation declares that instances of this type can be serialized to XML, and with a little magic in the REST module, to JSON as well. The `@XmlElement` annotations declare that the attributes should be treated as XML elements or JSON object properties. If we were to serialize an instance of this class to XML it would look like

```

<config>
  <name>Charlie</name>
  <time>5</time>
</config>

```

And in JSON it would look like

```
{
  "name": "Charlie",
  "time": 5
}
```

Now let's implement the GET method, which returns a Response with an instance of the `Config` type we've just created as the entity.

```

@GET
@Produces(MediaType.TEXT_HTML)
public Response get(@Context HttpServletRequest request)
{
    String username = userManager.getRemoteUsername(request);
    if (username != null && !userManager.isSystemAdmin(username))
    {
        return Response.status(Status.UNAUTHORIZED).build();
    }

    return Response.ok(transactionTemplate.execute(new TransactionCallback()
    {
        public Object doInTransaction()
        {
            PluginSettings settings = pluginSettingsFactory.createGlobalSettings();
            Config config = new Config();
            config.setName((String) settings.get(Config.class.getName() + ".name"));

            String time = (String) settings.get(Config.class.getName() + ".time");
            if (time != null)
            {
                config.setTime(Integer.parseInt(time));
            }
            return config;
        }
    })).build();
}

```

This method first checks if the user is logged in by checking if the `username` is `null`. If the user is logged in, it checks that the user is a system admin. If the user is not logged in or is not a system admin, the method returns a `Response` which will redirect the user to the application login page.

Next, we construct our `Response` with an entity that is returned from some call to a funky `transactionTemplate.execute` method, providing it an instance of `TransactionCallback`. If you've used Spring's `TransactionTemplate`, this should look familiar to you. For everyone spoiled by AOP, the summary is that we are accessing data that is coming from a database - well, it's not in Fisheye or Crucible, but work with me here! To ensure that reads and writes don't clash and give us inconsistent data, we need to protect ourselves anytime we access `PluginSettings` data. The `TransactionTemplate` frees us from needing to know the applications specific transaction creation and usage semantics. If we didn't use the `TransactionTemplate` we'd need code similar to the following

```

Transaction tx = // go out and find or create transaction in application specific way
tx.start();
try
{
    // access plugin settings and do other work
    tx.commit();
}
catch (Exception e)
{
    tx.rollback();
}

```

But the `TransactionTemplate` takes care of managing the transaction for us, calling our `TransactionCallback` to perform the real work. The return value of the `TransactionTemplate.execute` method is the return value of calling the `TransactionCallback.doInTransaction` method.

Inside our `TransactionCallback.doInTransaction` method, we use the `PluginSettingsFactory` to create a reference to the global, application wide settings data. We need to use the global settings object because we're configuring a plugin. If we were storing data associated with a project (in JIRA, Bamboo, Fisheye or Crucible) or a space (in Confluence), we'd instead use the `PluginSettingsFactory.createSettingsForKey` method, where the key is the project or space the configuration data should be associated with.

Once we have a reference to a `PluginSettings` object, things are pretty easy. `PluginSettings` can be thought of as a persistent map. Just use `get` to retrieve data and `put` to add or update values.



Namespace your keys!

Because this is **global** application configuration data, it is important that you do some kind of namespacing with your keys. Whether you use your plugin key, a class name or something else entirely is up to you. Just make sure it is unique enough that conflicts with other configuration data won't occur.

Now we can move onto allowing the admin to update the configuration data.

Updating the configuration

Now that our form is populated with our current configuration, we want to allow the admin to update the configuration.

JavaScript for PUTting an updated configuration

Add the following function after the `populateForm` function in `admin.js`.

```
function updateConfig() {
    AJS.$.ajax({
        url: baseUrl + "/rest/xproduct-admin/1.0/",
        type: "PUT",
        contentType: "application/json",
        data: '{ "name": "' + AJS.$("#name").attr("value") + '", "time": ' +
AJS.$("#time").attr("value") + ' }',
        processData: false
    });
}
```

This defines the function we'll call to update the config on the server. It is very minimal. We haven't included error checking and we really should be escaping quotes and other special characters when creating our data, but that is left as an exercise for the reader.

Now, we just need to hook that up to our form. After the call to `populateForm()` add

```
AJS.$("#admin").submit(function(e) {
    e.preventDefault();
    updateConfig();
});
```

Now, whenever the form is submitted, instead of the normal, default action of POSTing the form data, our `updateConfig()` method will be called and our updated configuration PUT to the configuration resource.

Updating the configuration with PUT

To handle the PUT requests from the client, add the following new method to `ConfigResource`

```
@PUT
@Consumes(MediaType.APPLICATION_JSON)
public Response put(final Config config, @Context HttpServletRequest request)
{
    String username = userManager.getRemoteUsername(request);
    if (username != null && !userManager.isSystemAdmin(username))
    {
        return Response.status(Status.UNAUTHORIZED).build();
    }

    transactionTemplate.execute(new TransactionCallback()
    {
        public Object doInTransaction()
        {
            PluginSettings pluginSettings = pluginSettingsFactory.createGlobalSettings();
            pluginSettings.put(Config.class.getName() + ".name", config.getName());
            pluginSettings.put(Config.class.getName() + ".time",
Integer.toString(config.getTime()));
            return null;
        }
    });
}

return Response.noContent().build();
}
```

We declare via the `@Consumes` annotation that we expect a request with a `application/json` body. What we're expecting is JSON in the same form as our `Config` object that we created above is serialized to so that the REST module can deserialize to a `Config` object, which it passes into our `put` method.

Next, we do the authentication and authorization check to make sure no one can modify the configuration except admins. Then, we wrap our modifications in a `TransactionCallback` we pass to the `transactionTemplate`, just like we did when we were retrieving the configuration. Since `PluginSettings` is just a persistent map, we just put the new configuration values into it and we're done.

Finally, we return `204 No Content` to tell the client the update succeeded and that there is nothing else to be said.

Wiring things together

As the last step we need to add the necessary modules to the `atlassian-plugin.xml` file to make what we've done so far work. We have used a few new services from SAL that we need to import. We also need to add the `rest` module so the plugin system knows we have a REST resource and what path they should be available under. To do all this add the following lines to the `atlassian-plugin.xml` file.

```
<component-import key="pluginSettingsFactory"
interface="com.atlassian.sal.api.pluginsettings.PluginSettingsFactory" />
<component-import key="transactionTemplate"
interface="com.atlassian.sal.api.transaction.TransactionTemplate" />

<rest key="rest" path="/xproduct-admin" version="1.0">
    <description>Provides REST resources for the admin UI.</description>
</rest>
```

Menu items

The last thing we need to do is make it easy for the admin to find our configuration administration UI. To do that we'll add `Web items` that will add a link in the application administration menus.

`Web items`. Unfortunately, not every application uses the same layout for their menus, so we need to add a specific entry for each. The `application` attribute will come to our rescue (mostly)!

To add a `web-item` in the "Global Settings" menu in JIRA's administration area, add the following `web-item` element.

```
<web-item key="jira-menu-item" name="XProduct Admin" section="system.admin/globalsettings"
weight="10" application="jira">
    <description>Link to xproduct-admin page.</description>
    <label key="xproduct.admin.label" />
    <link linkId="xproduct-admin-link">/plugins/servlet/xproduct/admin</link>
</web-item>
```



To add a `web-item` in the "Global Settings" menu in Fisheye's administration area, add the following `web-item` element.

```
<web-item key="fe-menu-item" name="XProduct Admin" section="system.admin/global" weight="10"
application="fisheye">
    <description>Link to xproduct-admin page.</description>
    <label key="XProduct Admin" />
    <link linkId="xproduct-admin-link">/plugins/servlet/xproduct/admin</link>
</web-item>
```

Looking carefully, you'll notice that the `label key` attribute isn't really a key. This is because Fisheye doesn't do any internationalization yet, so the value of the `key` attribute will be the text that is displayed.



To add a `web-item` in the "Plugins" menu in Bamboo's administration area, add the following `web-item` element.

```

<web-item key="bamboo-menu-item" name="XProduct Admin" section="system.admin/plugins" weight="10"
application="bamboo">
    <description>Link to xproduct-admin page.</description>
    <label key="XProduct Admin" />
    <link linkId="xproduct-admin-link">/plugins/servlet/xproduct/admin</link>
</web-item>

```

Bamboo has the same limitation as Fisheye, so again the `label` element's `key` attribute is the textual value we want to display.



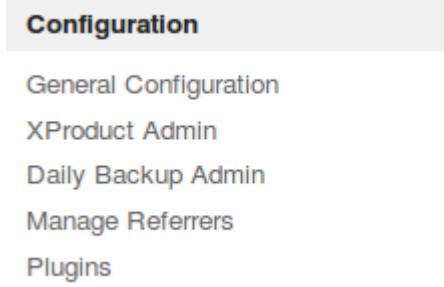
To add a `web-item` in the "Configuration" menu in Confluence's administration area, add the following `web-item` element.

```

<web-item key="conf-menu-item" name="XProduct Admin" section="system.admin/configuration"
weight="10">
    <description>Link to xproduct-admin page.</description>
    <label key="xproduct.admin.label" />
    <link linkId="xproduct-admin-link">/plugins/servlet/xproduct/admin</link>
</web-item>

```

Astute readers will notice the suspicious lack of the `application` tag on that `web-item` element. Well, in Confluence, if you add the `application` attribute, your `web-item` won't show up at all. We'll leave it off here, which is safe because none of the other Atlassian applications have a `system.admin/configuration` section - at least, at the time of this writing they don't.



Finally, let's add a `web-item` to the "General" menu in the RefApp's administration area.

```

<web-item key="refapp-menu-item" name="XProduct Admin" section="system.admin/general" weight="10"
application="refapp">
    <description>Link to xproduct-admin page.</description>
    <label key="xproduct.admin.label" />
    <link linkId="xproduct-admin-link">/plugins/servlet/xproduct/admin</link>
</web-item>

```



And that's it! We're done! Lets run it!

Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:

- Open another command window and go to the plugin root folder.
- Run `atlas-cli` in the second command window to start the CLI.
- When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

The above instructions will work for testing your plugin in the RefApp. To see this working in JIRA, Confluence, FishEye and Bamboo, follow the above steps but where it says to run `atlas-run`, instead use one of the following commands:

- For JIRA

```
atlas-run --product jira --version 4.0.2
```

- For Confluence

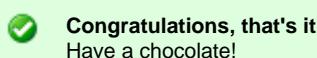
```
atlas-run --product confluence --version 3.1
```

- For FishEye

```
atlas-run --product fecru --version 2.1.4-460
```

- For Bamboo

```
atlas-run --product bamboo --version 2.5
```



RELATED TOPICS

[Using the Atlassian RefApp](#)

Plugin Tutorial - Writing Gadgets for JIRA

The source code of the plugin used in this tutorial is available on the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- Overview
- Step 1. Create the Plugin Project
- Step 2. Create the Gadget Spec
- Step 3. Customise the Plugin Descriptor and Maven POM
- Step 4. Set Up a REST Resource
- Step 5. Use JavaScript to Get the Projects into the Gadget
- Step 6. Build, Install and Run the Plugin
- Step 7. Write Unit and Integration Tests

Overview

In this tutorial, we're going to create a new Atlassian gadget for JIRA, bundle it inside a plugin, use a REST resource to provide it with data, and have the gadget talk to the resource.

Atlassian is [implementing the OpenSocial specification](#) in all of its applications. The gadget we'll write here is actually an OpenSocial gadget with some special sauce for JIRA. We'll assume a passing familiarity with gadgets, but in case they're new to you, there's a [space devoted to gadget authoring](#).

As was said earlier, we will create a gadget plugin. As with all plugins, our plugin will consist of the following components:

- Gadget spec file to hold the gadget's XML and JavaScript
- Java classes implementing the REST resource the gadget will use
- Resources for display of the plugin UI
- Plugin descriptor to enable the plugin module in JIRA

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Step 1. Create the Plugin Project

First we need to set up our development environment and create our plugin project and source directories.

We'll be using the Atlassian Plugin SDK throughout the tutorial, so make sure you have it installed and working as described [here](#). To check that you're ready to go, try the `atlas-version` command; you should see output like the following:

```
$ atlas-version

ATLAS Version: 3.0-beta4
ATLAS Home: /home/user/jlib/atlassian-plugin-sdk-3.0-beta4
ATLAS Scripts: /home/user/jlib/atlassian-plugin-sdk-3.0-beta4/bin
ATLAS Maven Home: /home/user/jlib/atlassian-plugin-sdk-3.0-beta4/apache-maven
-----
Executing: /home/user/jlib/atlassian-plugin-sdk-3.0-beta4/apache-maven/bin/mvn --version
Apache Maven 2.1.0 (r755702; 2009-03-18 11:10:27-0800)
Java version: 1.6.0_16
Java home: /usr/lib/jvm/java-6-sun-1.6.0.16/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux" version: "2.6.28-15-generic" arch: "i386" Family: "unix"
```

Create a new JIRA plugin with the specified values:

```
$ atlas-create-jira-plugin
Executing: /home/user/jlib/atlassian-plugin-sdk-3.0-beta4/apache-maven/bin/mvn
com.atlassian.maven.plugins:maven-jira-plugin:3.0-beta4:create
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building Maven Default Project
[INFO]   task-segment: [com.atlassian.maven.plugins:maven-jira-plugin:3.0-beta4:create]
(aggregate-style)
[INFO] -----
[INFO] [jira:create]
[INFO] Setting property: classpath.resource.loader.class =>
'org.codehaus.plexus.velocity.ContextClassLoaderResourceLoader'.
[INFO] Setting property: velocimacro.messages.on => 'false'.
[INFO] Setting property: resource.loader => 'classpath'.
[INFO] Setting property: resource.manager.logwhenfound => 'false'.
[INFO] [archetype:generate]
[INFO] Generating project in Interactive mode
[INFO] Archetype repository missing. Using the one from
[com.atlassian.maven.archetypes:jira-plugin-archetype:5 -> https://maven.atlassian.com/public]
found in catalog internal
Define value for groupId: : com.atlassian.plugins.tutorial
Define value for artifactId: : jira-gadget-tutorial-plugin
Define value for version: 1.0-SNAPSHOT: : 1.0-SNAPSHOT
Define value for package: com.atlassian.plugins.tutorial: : com.atlassian.plugins.tutorial
Confirm properties configuration:
groupId: com.atlassian.plugins.tutorial
artifactId: jira-gadget-tutorial-plugin
version: 1.0-SNAPSHOT
package: com.atlassian.plugins.tutorial
Y: :
[INFO] -----
[INFO] Using following parameters for creating OldArchetype: jira-plugin-archetype:3.0-beta4
[INFO] -----
[INFO] Parameter: groupId, Value: com.atlassian.plugins.tutorial
[INFO] Parameter: packageName, Value: com.atlassian.plugins.tutorial
[INFO] Parameter: package, Value: com.atlassian.plugins.tutorial
[INFO] Parameter: artifactId, Value: jira-gadget-tutorial-plugin
[INFO] Parameter: basedir, Value: /home/user/tmp
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] ***** End of debug info from resources from generated POM *****
[INFO] OldArchetype created in dir: /home/user/tmp/jira-gadget-tutorial-plugin
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 26 seconds
[INFO] Finished at: Wed Sep 16 16:33:01 PDT 2009
[INFO] Final Memory: 25M/53M
[INFO] -----
```

See our guides:

- Setting up your development tools
- Getting started with developing your plugin

Step 2. Create the Gadget Spec



For complete information on making a gadget spec, see the Atlassian Gadgets documentation.

First we will create the gadget spec file. This is `src/main/resources/gadget.xml`:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
    <ModulePrefs title="__MSG_gadget.title__" directory_title="__MSG_gadget.title__"
        description="__MSG_gadget.description__">
        <Optional feature="gadget-directory">
            <Param name="categories">
                JIRA
            </Param>
        </Optional>
        <Locale
            messages="__ATLASSIAN_BASE_URL__/download/resources/jira-gadget-tutorial-plugin/i18n/ALL_ALL.xml"/>
    </ModulePrefs>
    <Content type="html" view="profile">
        <!-- omitted for now -->
    </Content>
</Module>
```

You should recognise the `<ModulePrefs>` section as the metadata container for the gadget: title, directory title, description and so on. The `<Content>` section contains the HTML and/or JavaScript that drive the gadget's behaviour. It's been left blank here while we look more closely at `<ModulePrefs>`.

There are a few important things to notice:

- The `__MSG_gadget.title__` and `__MSG_gadget.description__` properties are substituted at runtime by the gadget's message bundles, specified in `<Locale>` elements. The bundle is at `src/main/resources/i18n/ALL_ALL.xml`.
- The `<Locale>` element tells the gadget where to find the message bundles. The `messages` attribute takes a URL. We'll show how to expose the message bundle in the next step.
- Finally, notice the `<Optional>` feature. This 'gadget-directory' feature specifies that the gadget is for JIRA and should be placed in the 'JIRA' category in the gadget directory browser. Without this, it is much harder to find and use gadgets from the directory browser.

Here's the XML message bundle itself, under `src/main/resources/i18n/ALL_ALL.xml`:

Step 3. Customise the Plugin Descriptor and Maven POM

Now we need to edit the plugin descriptor at `src/main/resources/atlassian-plugin.xml` to give our plugin a unique key, and some meta information about this plugin.

For our plugin, we will start with a module declaration for the gadget spec:

```
<gadget key="tutorial-gadget" name="JIRA Tutorial Gadget" location="gadget.xml"/>
```

A gadget is a module in `atlassian-plugin.xml`, like any other JIRA module. There are three required properties to note:

- **key** must be unique for all modules in this plugin
- **name** is the name under which the plugin module will be displayed in the JIRA administration console
- **location** is the path to the gadget spec file, relative to `src/main/resources`

Next, we'll add the `<resource>` for the message bundle:

```
<resource type="download" name="i18n/ALL_ALL.xml" location="i18n/ALL_ALL.xml">
    <param name="content-type" value="text/xml; charset=UTF-8"/>
</resource>
```



Building URLs to message bundles

At gadget render time, `__ATLASSIAN_BASE_URL__` will be automatically substituted with JIRA's configured base URL. The rest of the download URL is made from the `atlassian-plugin.xml` plugin key (in this case `jira-gadget-tutorial-plugin`) and the `<resource>`'s key value (in this case `i18n/ALL_ALL.xml`).

Below is the `atlassian-plugin.xml` for our plugin:

Finally, to support the included REST module, add the following dependencies to `pom.xml` in the `<dependencies>` element:

```
<dependency>
    <groupId>com.atlassian.jira</groupId>
    <artifactId>atlassian-jira</artifactId>
    <version>${jira.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.ws.rs</groupId>
    <artifactId>jsr311-api</artifactId>
    <version>1.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.plugins.rest</groupId>
    <artifactId>atlassian-rest-common</artifactId>
    <version>1.0.2</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>2.3</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.atlassian.sal</groupId>
    <artifactId>sal-api</artifactId>
    <version>2.1.beta4</version>
</dependency>
```

Now we are ready to write some code to make our gadget do something.

Step 4. Set Up a REST Resource

Our gadget will be consuming a REST resource to show how to get dynamic data into a gadget. A full discussion of how to write a REST resource is beyond the scope of this tutorial, but there is another tutorial for doing so: [Writing REST services](#). We'll use the resource developed in that tutorial in our plugin.

We encourage you to study the [REST tutorial](#) for a complete explanation of writing REST resources in JIRA.

Step 5. Use JavaScript to Get the Projects into the Gadget

We will now develop JavaScript to call the REST resource and display the information in the gadget. At this point, you are strongly encouraged to read the documentation on [using the Atlassian Gadgets JavaScript Framework](#), as it will help you understand the code we're about to write.

Returning to `src/main/resources/gadget.xml`, let's look at the `<Content>` section:

```

<Content type="html" view="profile">
    <![CDATA[
        #requireResource( "com.atlassian.jira.gadgets:common" )
        #includeResources()

        <h1>Hello from the Atlassian Gadget Tutorial!</h1>

        <script type="text/javascript">
            (function () {
                var gadget = AJS.Gadget({
                    baseUrl: "__ATLASSIAN_BASE_URL__",
                    view: {}
                });
            })();
        </script>
    ]]>
</Content>

```

Here, take note of the following:

- `#requireResource()`/`#includeResources()`, described in detail in [the Atlassian Gadgets documentation](#), is used here to bring in the JIRA Gadget JavaScript framework. The directive `#requireResource` identifies a `<web-resource>` module inside a plugin, while `#includeResources` writes out the HTML tags for the resource in place. `#includeResources` is smart enough to write `<style>` tags for a CSS web resource, `<script>` for a JavaScript resource, and so on.



Isn't that WebResourceManager?

If you've written Confluence or JIRA plugins before, you may recognise these directives as methods on the [WebResourceManager](#).

- `var gadget = AJS.Gadget` constructs a Gadget object. This object is from the JavaScript framework and is documented in [the Atlassian Gadgets documentation](#). In this section, we specify the bare minimum to get a working gadget.
 - A gadget must know its base URL to operate properly. This is provided with the same `__ATLASSIAN_BASE_URL__` system property that was used in the `<Locale>` element inside `<ModulePrefs>`.
 - The `view` object allows us to customise how the gadget is displayed on the page. It is documented in detail in [the Atlassian Gadgets documentation](#).

We'll use the JavaScript framework's built-in support for Ajax to get the data we need from the resource. The code looks like this:

```

(function () {
    var gadget = AJS.Gadget({
        baseUrl: "__ATLASSIAN_BASE_URL__",
        view: {
            template: function(args) {
                var gadget = this;
            },
            args: [
                {
                    key: "projects",
                    ajaxOptions: function() {
                        return {
                            url: "/rest/tutorial-gadget/1.0/projects"
                        };
                    }
                }
            ]
        }
    })();
}
))();

```

The `view` object's most important property is `template`, which is a function that generates the gadget's HTML. How you generate the HTML is up to you, but the most common approach is to use [jQuery's manipulation API](#). Also, the gadget object itself is available as `this` inside `template()`, allowing use of the [Gadget object API](#). `template()` takes a parameter `args`, which gives access to data made available in the `view`'s `args` array.

The `args` array is used to specify what data the gadget needs to use. Each member of the `args` array is a spec for remote data retrieval. A member object's `key` is the name we'll use to refer to the member after it's been passed to `template()`. `ajaxOptions` is an object whose properties are [jQuery Ajax options](#). In this simple case, where we're making only a GET request to a REST service, the `url` property is all that's required.

Finally, we can write the code that will take the returned data and render it:

```
(function () {
    var gadget = AJS.Gadget({
        baseUrl: "__ATLASSIAN_BASE_URL__",
        view: {
            template: function(args) {
                var gadget = this;

                var projectList = AJS.$("<ul/>");

                AJS.$(args.projectData.projects).each(function() {
                    projectList.append(
                        AJS.$("<li/>").append(
                            AJS.$("<a/>").attr({
                                target: "_parent",
                                title: gadgets.util.escapeString(this.description),
                                href: "__ATLASSIAN_BASE_URL__" + "/browse/" + this.key
                            }).text(this.name)
                        )
                );
            });
        },
        gadget.getView().html(projectList);
    },
    args: [
        {
            key: "projectData",
            ajaxOptions: function() {
                return {
                    url: "/rest/tutorial-gadget/1.0/projects.json"
                };
            }
        }
    ]
});
});
```



Be careful of side effects!

Be careful of side effects!
Whenever the gadget is reloaded or resized on the page, it will be re-rendered, and `template()` will be called. Make sure that doing so has no side effects.

Step 6. Build, Install and Run the Plugin

That's it. We now need to test our code. We open a command window and go to the plugin root folder (where the `pom.xml` is located). Run `atlas-cli` to start the CLI. When we see a message, `Waiting for commands`, we can run `pi` to compile, package and install the plugin.

Now we go back to the browser. The updated plugin has been installed into the application, and we can test our changes. If you don't already have a dashboard created, do so.

- Click the Add Gadget menu option in the upper right.
 - Find the gadget called "Test JIRA Tutorial Gadget" in the list. This is our gadget. Add it to the dashboard.
 - You should see a list of projects viewable by all users display in the gadget.



No projects appearing?

Make sure you have at least one project in JIRA that is viewable by all users.

Step 7. Write Unit and Integration Tests



Work in progress

RELATED TOPICS

Developing your Plugin using the Atlassian Plugin SDK

Gadget Development

Plugin Tutorial - Writing Integration Tests for your JIRA plugin



Level of experience: Intermediate

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'intermediate' level. If you have never developed a plugin before, you may find this one a bit difficult.

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- [Overview](#)
- [Step 1. Create the Plugin Project](#)
- [Step 2. Add Plugin Metadata to the POM](#)
- [Step 3. Register the Plugin Module in the Plugin Descriptor](#)
- [Step 4. Write Java Classes](#)
- [Step 5. Build, Install and Run the Plugin](#)
- [Step 6. Write Unit and Integration Tests](#)

Overview

This tutorial shows you how to write integration tests for a plugin. This tutorial demonstrates how to write such tests for both UI elements provided by the plugin itself, as well as the host application.

In order to do this, you will create a JIRA plugin. As with all plugins, your plugin will consist of the following components:

- Java classes encapsulating the plugin logic
- Resources for display of the plugin UI
- Plugin descriptor to enable the plugin module in JIRA.

All these components will be contained within a single JAR file. Each component is discussed further in the examples below.

There are various definitions available for what [integration testing](#) actually means. For the purposes of this tutorial, integration testing is defined as running a set of automated tests against the web-UI of your plugin, using HTTP requests and responses to assert that the web-UI is exhibiting the correct behaviour.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- [Set up your development environment](#)
- [Create and install your plugin from a template](#)

When you create the plugin using the above SDK, you will be prompted for some information to identify your plugin. Enter the following information:

- **group-id:** com.atlassian.plugins.tutorial
- **artifact-id:** jira-integration-tests-tutorial
- **version:** 1.0
- **package:** com.atlassian.plugins.tutorial

Step 2. Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
<organization>
<name>Example Company</name>
<url>http://www.example.com/</url>
</organization>
```

3. Update the `<description>` element:

```
<description>This plugin demonstrates how to write integration tests for a JIRA
plugin.</description>
```

4. Save the file.

Step 3. Register the Plugin Module in the Plugin Descriptor

Now you will add the required plugin modules to your plugin descriptor at `src/main/resources/atlassian-plugin.xml`. The plugin descriptor is an XML file that identifies the plugin to JIRA and defines the functionality that the plugin requires.

Here's a basic plugin descriptor:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}">
  plugins-version="2"
  <plugin-info>
    <description>${project.description}</description>
    <version>${project.version}</version>
    <vendor name="${project.organization.name}" url="${project.organization.url}" />
  </plugin-info>
</atlassian-plugin>
```

Note that some of the information from the POM is transferred to the plugin descriptor using variable names such as `${project.artifactId}`.

You will need the following plugin module:

atlassian-plugin.xml

The above code snippet introduces a webwork action into this plugin. This is not strictly necessary to write integration tests, but it adds a good example to write integration tests for.

Now you are ready to move onto writing some code to make your plugin do something.

Step 4. Write Java Classes

In this plugin, you want to write integration tests for some plugin points defined by your plugin.

To do this, you will write a webwork action that simply displays the projects in which the currently logged in user has the permission provided as a parameter. So for example, a user will be able to show all the projects that he/she has the `Create Issue` permission in. The webwork action will also perform some error checks to ensure that only logged in users can query this information and that the permission submitted as a parameter to the action is a valid JIRA permission.

First we need to construct the webwork action with a JIRA PermissionManager which will be used to check what projects a user has a given permission in:

ShowUsersProjectPermissionAction.java

When a user submits a request to this action, we'll have to do some validation:

ShowUsersProjectPermissionAction.java

If validation passed, the `doExecute()` method is called to retrieve the information we're after:

ShowUsersProjectPermissionAction.java

Finally, all webwork actions need a corresponding view:

showprojects.vm

Step 5. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

You should be able to visit <http://localhost:2990/jira/secure>ShowUsersProjectPermissionAction.jspa?permission=create> to see a list of all the projects that you have the `Create Issue` permission in. This may require you to create some projects first.

Step 6. Write Unit and Integration Tests

Congratulations! You've made it to the meaty part of this tutorial. We'll only focus on integration tests since there's nothing to unit test here, really.

Before we jump into writing an integration test, here's a bit of background about how integration tests work in JIRA. JIRA's integration tests are driven by [jWebUnit](#) (albeit a fairly ancient version: 1.2). jWebUnit's purpose is essentially to simulate a browser, making HTTP requests to your web application and making assertions about the state returned in responses by your application. It provides a number of built-in methods to make this easy such as `clickLink("linkid")` and `assertTextPresent("Sample Text")`.

JIRA built an integration testing framework on top of this to make certain actions in JIRA easier. Historically, integration tests would be written by extending `JiraWebTest` which would provide all of these utility methods to your integration test. This has now been superseded by a new integration testing framework for JIRA which provides much better documentation and is more flexible and extensible. Please have a read of the framework's [API docs](#) for more details about what's possible (make sure to checkout all the interfaces in the main `package` as they contain all the useful utility methods available to you).

Before we start writing an integration test, you'll have to create a `localtest.properties` file in your `src/test/resources` folder:

localtest.properties

JIRA's integration test framework uses this file to figure out which URL your JIRA instance is running under and where to find XML backup files your tests can import to set up the test data. At some stage in the [near future](#), this file will be generated automatically.

Once you have your `localtest.properties` file in place, you can begin writing your actual integration test case in `src/test/java/it`:

ShowUsersProjectPermissionTest.java

To write an integration test using JIRA's integration test framework, your test class has to extend `FuncTestCase`. This class provides a number of helpers such as `administration`, `text` and `navigation` to make integration testing easier. Note that tests not in the `it` directory are treated as unit tests by the `atlas-integration-test` command later on, which means that JIRA won't be automatically started and all the "unit tests" will fail.

Before you can start testing, we need to set up JIRA with some data that we can use to test. This is best done by importing an XML backup before a test runs:

ShowUsersProjectPermissionTest.java

This method uses the `administration` helper to restore the `TestShowUsersProjectPermission.xml` backup in JIRA.



While you can use the helper classes provided by `FuncTestCase` to create test data, this is generally not recommended. For example, creating a large number of issues this way can be very slow. It's better to manually create your test data first, which can then be backed up to XML and imported in your test.

Now we have JIRA setup with some data and we can start writing our first test. Let's test the `webwork` action we wrote earlier:

ShowUsersProjectPermissionTest.java

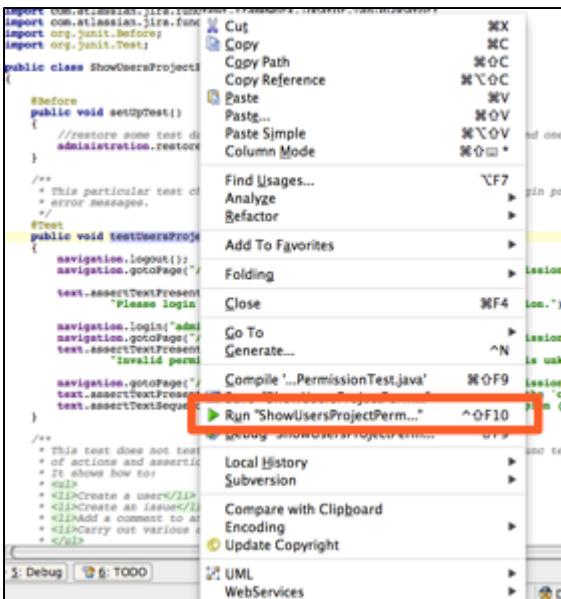
This particular test navigates to our `webwork` action by going to a specific URL and asserts that the correct text has been returned. This test does not use `jWebUnit` directly, but instead uses helpers provided by JIRA's integration testing framework. One particular pattern that is very useful is the `text.assertTextPresent(SomeLocator)` pattern. Quite often a web-page will contain the same text in several different places on a web page (for example the name of the logged in user may appear in the page header as well as in the reporter field of an issue) and it's necessary to only run an assertion in a particular part of a page. Locators are a way of limiting what part of the page an assertion is being run over. This is one of the most powerful parts of the integration testing framework and it is highly recommended to have a look at the different kinds of `locators` available to you.

Finally, let's demonstrate some more advanced ways one can write an integration test:

ShowUsersProjectPermissionTest.java

This test demonstrates some more advanced things that are possible with JIRA's integration testing framework. Line 82 also demonstrates how to use `jWebUnit` directly via the `tester` variable, should this ever be necessary.

To run this test you can simply run `atlas-integration-test` from the command line. This will build your plugin, start up JIRA and run all tests. To run individual tests, you should also be able to start up JIRA with `atlas-run` and execute your test from your IDE. In IntelliJ, you can simply right click on a method and click on `Run '<TESTNAME>'`:



If there's any test failures, the build will fail:

```
Tests run: 2, Failures: 1, Errors: 0, Skipped: 0

[ INFO] -----
[ERROR] BUILD ERROR
[INFO] -----
[INFO] Unable to execute mojo

Embedded error: There are test failures.

Please refer to
/Users/andreas/projects/atlassian/jira-integration-tests-tutorial/target/jira/tomcat6x/surefire-repo
for the individual test results.
```

The directory mentioned in the build log will contain a text file for each test class in your integration test suite with more details about what went wrong:

```
-----
Test set: it.ShowUsersProjectPermissionTest
-----
Tests run: 2, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 72.129 sec <<< FAILURE!
testUsersProjectPermissions(it.ShowUsersProjectPermissionTest)  Time elapsed: 40.25 sec <<<
FAILURE!
junit.framework.AssertionFailedError: The text 'The user admin h the 'create' permission in the
following projects' could not be found via locator WebPageLocator : 1 node(s) - 'Welcome - Your
Company JIRA var conte...'
        at junit.framework.Assert.fail(Assert.java:47)
        at
com.atlassian.jira.functest.framework.assertions.TextAssertionsImpl.assertTextPresentImpl(TextAssertion
at
com.atlassian.jira.functest.framework.assertions.TextAssertionsImpl.assertTextPresent(TextAssertionsI
at
it.ShowUsersProjectPermissionTest.testUsersProjectPermissions(ShowUsersProjectPermissionTest.java:39)
```

In this case I misspelled the text "The user admin **h** the 'create' permission in the following projects" on line 39 of the ShowUsersProjectPermissionTest.



Congratulations, that's it

You should now be able to write integration tests for JIRA. Oh and don't forget to have a chocolate!

RELATED TOPICS

[Plugin Testing Resources and Discussion](#)

Plugin Tutorial - Writing Macros for Confluence



Looking for Confluence 4.0 macro help?

You can find tutorials for building Confluence 4.0 macros [here](#).

The source code of the plugin used in this tutorial is available on the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- Overview
- Step 1. Create the Plugin Project
- Step 2. Customise the Plugin Descriptor
- Step 3. Write Java Classes
- Step 4. Register the Plugin Module in the Plugin Descriptor
- Step 5. Build, Install and Run the Plugin
- Step 6. Write Unit and Integration Tests

Overview

In this tutorial, we'll show how to write a Confluence macro.

In order to do this, we will create a macro plugin. As with all plugins, our plugin will consist of the following components:

- Java classes encapsulating the plugin logic
- Resources for display of the plugin UI
- Plugin descriptor to enable the plugin module in Confluence

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Step 1. Create the Plugin Project

First we need to set up our development environment and create our plugin project and source directories. See our guides:

- [Setting up your development tools](#)
- [Getting started with developing your plugin](#)

Step 2. Customise the Plugin Descriptor

Now we need to edit the plugin descriptor at `src/main/resources/atlassian-plugin.xml` to give our plugin a unique key, and some meta information about this plugin.

For our plugin, we will use the `macro` plugin module.

Below is the `atlassian-plugin.xml` for our plugin:

Now we are ready to move on to writing the macro implementation.

Step 3. Write Java Classes

A macro is a Java class that implements the `com.atlassian.renderer.v2.macro.Macro` interface, which lives in a renderer module shared between Atlassian applications. To simplify our task, we'll extend a convenience class that Confluence provides for macro authors, `BaseMacro`.



All about the Macro interface

There is an [explanation of this interface](#), which we encourage you to read at the end of this tutorial. We'll focus here only on the parts directly relevant to our needs.

The crucial method in `BaseMacro` is `execute()`:

```
String execute(Map<String, String> params, String body, RenderContext context);
```

At render time, macros are invoked through this method. The parameters are:

- **params** - a `Map` of macro parameter names and values. See the [Confluence macro documentation](#) for details on how to pass parameters to macros.
- **body** - the text inside the macro opening and closing tags. Macros that don't have a body (like ours) don't require closing tags, so this parameter will be `null` in such cases.
- **context** - the `RenderContext` object supplies information about the renderer being used to render the current macro. Some

macros can take advantage of this information, but ours will not need to.

Let's see the `TutorialMacro` class as a whole:

```
public class TutorialMacro extends BaseMacro
{
    // constructor and instance variables omitted

    public boolean isInline()
    {
        return false;
    }

    public boolean hasBody()
    {
        return false;
    }

    public RenderMode getBodyRenderMode()
    {
        return RenderMode.NO_RENDER;
    }

    public String execute(Map params, String body, RenderContext renderContext)
        throws MacroException
    {
        // body coming up
    }
}
```

Here we see all four methods that a macro is required to implement. The new ones are:

- `isInline()` - if the macro output should stand apart from surrounding text (like, for example, the built-in `{info}` macro), this should return `false`. If the output should be interleaved into the surrounding text (like a HTML `<a>` tag, because usually occurs within some other tag like `<p>` or `<div>`), this should return `true`. Since our macro output will be a block of standalone HTML, we return `false`.
- `hasBody()` - if the macro expects a body text, this should return `true`. Ours does not, so it returns `false`.
- `getBodyRenderMode()` - specifies what rendering, if any, should be applied to the text returned from `execute()`. If a macro generates wiki markup, returning `RenderMode.ALL` or `RenderMode.INLINE` would be appropriate. Since our macro will render directly to HTML through a Velocity template, we don't want the renderer to mess with it, so we return `RenderMode.NO_RENDER`.

With the skeleton of the macro in place, we can implement the actual behaviour. We'll do something simple – greet the user by name, examine the available spaces and display a page title:

Again, `execute()` returns a `String`, which is just the macro-generated text. Like good software engineers, we'll separate our controller from our view and delegate the HTML rendering to a Velocity template, passing a model object from the former to the latter.



Are you sure you need a macro plugin?

While a very simple macro can assemble the `String` directly in the `execute()` method, such things are probably better implemented as [user macros](#), which don't require you to write a plugin and can be edited by users from inside Confluence.

In summary, this code creates a Velocity context and places several things into it, including the value of the `greeting` parameter (we make one ourselves if none is supplied), the total number of spaces in the installation, a randomly chosen space, and the title and creator of that space's home page. The `return` statement gets the rendered Velocity template and sends it back. Since our implementation of `getBodyRenderMode()` returns `RenderMode.NO_RENDER`, the HTML from Velocity will be inserted into the page with no further processing.

Let's look at the Velocity template, `src/main/resources/templates/tutorial-macro.vm`:

The tokens that start with `$` are Velocity references. They correspond to the objects inserted into the context by `TutorialMacro`. At render time, Velocity will replace these references with the corresponding values.



Be careful of context parameter names

By default, Velocity doesn't distinguish between references in the context that have no value and those that don't exist at all. If your rendered template has stuff like `$myVariable`, make sure you've spelled the reference exactly the same in both the template and the macro class.

Step 4. Register the Plugin Module in the Plugin Descriptor

Now that the macro class is written, we must now register the plugin module in our plugin descriptor, `atlassian-plugin.xml`.

The `<macro>` element is the heart of the configuration. It has several components:

- There are three required attributes, all of which must be unique across the Confluence installation:
 - `name` is used for the macro's markup in Confluence. Users will invoke the macro by writing `{name}`; in this case, our macro is used by typing `{tutorial-macro}`.
 - `class` is the name of the Java class that implements the `Macro` interface. We'll look at this interface more closely in a moment.
 - `key` is the identifier of this plugin module across the entire plugin. We recommend using the same value as `name`.
- The `<description>` element will be displayed next to the macro plugin module in the plugin administration panel.
- A macro plugin can use several `<resource>` types, but one you should always have is a help resource. Here, we've told the plugin module that the help for this macro is available in a Velocity template. The location is relative to `src/main/resources` in the plugin source tree. The `help-section` parameter specifies which part of the Confluence notation guide to put the help in; `confluence` maps to "Confluence Content", which is a good default if you're not sure where it should go.
- `category` defines which category the macro should live in within the [Confluence macro browser](#).
- `parameters` defines the parameters the macro can take. While they're not enforced by the renderer, the macro browser uses this to generate help text for our users, so it's always good to provide them.



The macro browser does a lot for you!

Please make time to read [this page](#). You'll learn how to make your macros user-friendly, attractive, and reliable.

Step 5. Build, Install and Run the Plugin

That's it. We now need to test our code. We open a command window and go to the plugin root folder (where the `pom.xml` is located). Run `atlas-cli` to start the CLI. When we see a message, `Waiting for commands`, we can run `pi` to compile, package and install the plugin.

Edit a page in your Confluence instance and insert the macro tag:

```
{tutorial-macro}
```

Save or preview the page. You should see the rendered output.

Step 6. Write Unit and Integration Tests



Work in progress

RELATED TOPICS

[Developing your Plugin using the Atlassian Plugin SDK](#)
[Confluence Macro Plugin Module](#)
[Writing Macros](#)
[Including Javascript and CSS resources](#)

Plugin Tutorial - Writing REST Services

The source code of the plugin used in this tutorial is available on the Atlassian public source repository. You can check out the source code [here](#).

On this page:

- Overview
- Step 1. Create the Plugin Project
- Step 2. Customise the Plugin Descriptor and Maven POM
 - Add the REST Module to the Plugin Descriptor
 - Add Dependencies to the POM
- Step 3. Write Java Classes
 - The Message Resource
 - The Message
- Step 4. Build, Install and Run the Plugin
- Step 5. Write Unit and Integration Tests

Overview

Atlassian REST services are implemented with [Jersey](#), which is the reference implementation of the [JSR-311](#) standard. We'll see some of Jersey's compelling features in this tutorial, where we want to show how to implement and deploy a REST service using the [Atlassian REST plugin module](#), then we will show you some extra features that are available in JIRA and Confluence.

In order to do this, we will create a REST service plugin which will contain some REST resources, then a JIRA and Confluence specific plugin.

Each plugin will contain:

- Java classes encapsulating the plugin logic (in this case, implementing the resource)
- Plugin descriptor to enable the plugin module in JIRA and Confluence

Each component is further discussed in the examples below.

The first plugin we will develop is designed as an example of the REST functionality available, the other plugins will tie these ideas into the specific applications and show you how to access JIRA and Confluence specific resources.

Step 1. Create the Plugin Project

First we need to set up our development environment and create our plugin project and source directories. See our guides:

- [Setting up your Plugin Development Environment](#)
- [Developing your Plugin using the Atlassian Plugin SDK](#)

It's expected you have followed these guides first and have a working (but probably empty) plugin.

Step 2. Customise the Plugin Descriptor and Maven POM

Add the REST Module to the Plugin Descriptor

We need to add a module type to our plugin to define the REST resource. Adding this module will result in a scan of all your classes for Jersey annotations.

complete `atlassian-plugins.xml`

After deployment the rest resources found in this plugin will be accessible at: `BASE_URL/rest/tutorial-rest/1.0/`, where `BASE_URL` is your application base URL. See the [documentation](#) for more details.

Add Dependencies to the POM

To properly configure your rest resources you will need some extra dependencies in your project.

complete `pom.xml`

These dependencies provide the annotations and other classes you will use to write your REST resources.

Step 3. Write Java Classes

Our sample plugin will provide some simple resources to display i18n resource messages. This is in fact a rather contrived example to show the REST functionality is available with introducing any application specific concepts. For details on how to tie these ideas in to JIRA or Confluence see below.



Atlassian uses the different names **Resource** and **Service** to distinguish between methods that return an entity, like getting a page (and generally not changing it), and a method that performs some sort of calculation, like a search.

The Message Resource

[link](#) `MessageResource.java`

This class introduces a number of Annotations:

| Annotation | Source | Scope | Description |
|--------------------|--------|-----------------|---|
| <code>@Path</code> | Path | Class or Method | Identifies the URI path that a resource class or class method will serve requests for. <code>@Path</code> annotations on methods are relative to the <code>@Path</code> annotation on the enclosing class. |
| <code>@GET</code> | GET | Method | Identifies that the class method will handle requests for a <code>GET</code> http message. Having more than one <code>@GET</code> annotation on a method in a class will fail unless <code>@Path</code> annotations are used to distinguish them. Other annotations are available such as <code>DELETE</code> , <code>POST</code> , <code>HEAD</code> , etc. See: javax.ws.rs |

| | | | |
|--------------------------|------------|------------------|--|
| @AnonymousAllowed | | Method | Identifies that the method can be called without supplying user credentials. If this annotation did not exist then a session would need to be established with your application or you would need to pass in <code>os_username</code> and <code>os_password</code> parameters. |
| @Produces | Produces | Method | It specifies the content types the method may return. If this annotation is not present, the method may return <i>any</i> content type; if it is present, the method must return one of the types specified. |
| @PathParam | PathParam | Method Parameter | It maps a method variable to an element in the <code>@Path</code> . |
| @QueryParam | QueryParam | Method Parameter | It maps a method variable to a query parameter. |

The Message

The message is used to encapsulate all the data that is returned to the client. JAXB Annotations are used to map this class to XML and JSON formats.

link [Message.java](#)

| Annotation | Source | Scope | Description |
|-------------------------|---------------------------|-------------------|---|
| @XmlRootElement | <code>XmlElement</code> | Class or enum | Maps a class or an enum type to an XML element. |
| @XmlAccessorType | <code>XmlAttribute</code> | Class or enum | Controls whether fields or properties are serialised by default |
| @XmlAttribute | <code>XmlElement</code> | Property or field | Maps a property or field to an XML Attribute |
| @XmlElement | <code>XmlElement</code> | Property or field | Maps a property or field to an XML Element |

More information on these annotations (and others available) can be found in the [JAX documentation](#).

Step 4. Build, Install and Run the Plugin

That's it, well almost. We first need to make sure our plugin works. We will be using the [Atlassian Plugin SDK Documentation](#) to run our plugin.

First, start the reference application with our plugin enabled by executing the script `atlas-run`. If you have the Atlassian SDK and the plugin installed correctly you will see the reference application start up.

```
$ atlas-run
Executing: /home/user/atlassian-plugin-sdk-3.0.4/apache-maven/bin/mvn
com.atlassian.maven.plugins:maven-amps-dispatcher-plugin:3.0.4:run
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building REST Service Plugin
[INFO]   task-segment: [com.atlassian.maven.plugins:maven-amps-dispatcher-plugin:3.0.4:run]
...
[INFO] [talledLocalContainer] Tomcat 6.x started on port [5990]
[INFO] refapp started successfully and available at http://localhost:5990/refapp
[INFO] Type CTRL-C to exit
```

This will start the reference application at <http://localhost:5990/refapp>. (See our page about the [Atlassian RefApp](#).) Our REST service will be deployed under <http://localhost:5990/refapp/rest/tutorial-rest/1.0/>, but going to this URL will produce a http 404 error (Not Found) because we don't have any resource mapped to the root.

Have a look how these example URLs are mapped to the class methods.

| URL | Class Method |
|---|---------------------------------|
| http://localhost:5990/refapp/rest/tutorial-rest/1.0/message | <code>getMessage(null)</code> |
| http://localhost:5990/refapp/rest/tutorial-rest/1.0/message?key=test | <code>getMessage("test")</code> |
| http://localhost:5990/refapp/rest/tutorial-rest/1.0/message/test | <code>getMessage("test")</code> |



When should I use PathParams or QueryParams?

Since you can map information from the request into the method in different ways, you may be wondering when to use `PathParams` over `QueryParams`. Here are some guidelines for you to think about.

Query parameters usually mean the resource will not be cached by a proxy or your browser. So if you are passing in an `id` to find some information about some sort of entity, then user a path parameter.

On the other hand, if your input is something that might be change with your data (like a user's full name), then you should use a query parameter. Your aim should be to provide a single authoritative URL for your resources so your clients can predict and cache its results. If a user got married and you used their name in the URL then it might change and caches and links would now fail.

Step 5. Write Unit and Integration Tests



Work in progress

// TODO

Jonathan Doklovic has written a great tutorial on the Sysbliss blog: [Testing REST plugins](#)

Accessing Confluence Resources

Accessing JIRA Resources

// TODO Insert details from Ben's Project Resources example

RELATED TOPICS

[Using the Atlassian RefApp](#)

[Developing your Plugin using the Atlassian Plugin SDK](#)

[Atlassian REST API Design Guidelines version 1](#)

[REST Plugin Module](#)

Plugin Tutorial - Writing Unit Tests for your Plugin

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code here:

- <http://svn.atlassian.com/svn/public/contrib/tutorials/tutorial-unit-testing-plugin/trunk>

On this page:

- Overview
- Required Knowledge
- Step 1. Create the example plugin project
- Step 2. Create a Java class for the macro unit test
- Step 3. Add a dependency on Mockito to pom.xml
- Step 4. Write the unit test code
- Step 5. Run the tests in your IDE
- Step 6. Run the tests with Maven
- Step 7. Adding another test
- Step 8. Run all the tests again
- Where to next?

Overview

This tutorial shows you how to start writing unit tests for your plugin.

Why would you want to write unit tests for a plugin? Here are a few possible reasons. Not all of them apply to all projects.

- Unit testing can find bugs before you release your code.
- Once you have a suite of tests, you can quickly verify that changes to your plugin have not broken existing functionality.
- Unit testing helps enforce modularity in your code by promoting [separation of concerns](#).

As an example of how to get started with unit testing, you will use the example Confluence plugin which is generated by the [Atlassian Plugin SDK](#) and add unit tests to that.

Required Knowledge

To complete this tutorial, you must already understand the basics of Java development: classes, interfaces, methods, how to use the compiler, and so on. You should also understand:

- how to create an Atlassian plugin project using the [Atlassian Plugin SDK](#)
- how to open the plugin project in your IDE
- how to compile your project and create a JAR file using Maven
- how to check out a project using Subversion.

This tutorial will teach you:

- how to write a unit test – a Java class which will automatically test your code
- how to run your unit tests with your IDE or with Maven
- how to use Mockito to create mock objects for testing
- how to use Mockito annotations to create several mock objects with less code.

Step 1. Create the example plugin project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

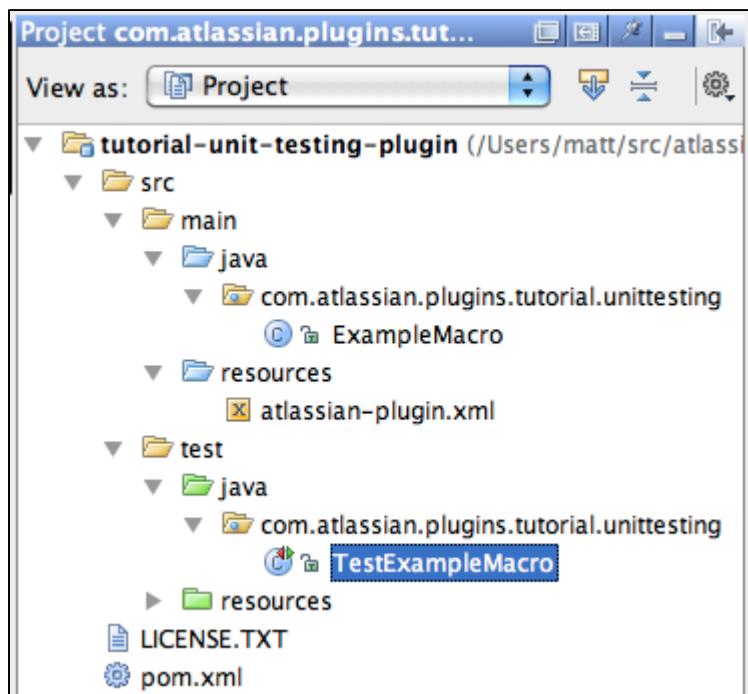
For this tutorial, we will use the following values when creating the plugin with `atlas-create-confluence-plugin`:

- group ID - com.atlassian.plugins.tutorial.unittesting
- artifact ID - tutorial-unittesting-plugin
- version - 1.0-SNAPSHOT
- package - com.atlassian.plugins.tutorial.unittesting

Step 2. Create a Java class for the macro unit test

In a Maven 2 project, your production Java files are stored in `src/main/java`, in a directory for each package. Your test classes will be stored separately in `src/test/java`.

Your project structure should look like this:



Project structure with unit tests in `src/test/java`

The test project already includes a unit test:

`src/test/java/com/atlassian/plugins/tutorial/unittesting/TestExampleMacro.java`. Open this file up, and you should see the following code – a skeleton of a unit test.

```

package com.atlassian.plugins.tutorial.unittesting;

import junit.framework.TestCase;

/**
 * Testing {@link com.atlassian.plugins.tutorial.unittesting.ExampleMacro}
 */
public class TestExampleMacro extends TestCase
{
    public void testBasic()
    {
        // add test here...
    }
}

```

This class shows the fundamentals of every unit test written with JUnit:

- line 1 - the class is in the same package as the class it is testing
- line 8 - the class name is the name of the class it is testing (`ExampleMacro` in this case) with 'Test' at the start
- line 8 - the class extends `junit.framework.TestCase`
- line 10 - each test method is `public void` and its name starts with 'test'.

Step 3. Add a dependency on Mockito to pom.xml

In the `pom.xml` file at the root of your project, you'll find a `<dependencies>` tag which includes all the dependencies of your project. We're going to add a dependency on [Mockito](#) to the POM for use in our unit tests.

Add lines 3-8 to the end of your existing `<dependencies>` tag.

```

<dependencies>
    ...
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-all</artifactId>
        <version>1.8.0</version>
        <scope>test</scope>
    </dependency>
</dependencies>

```

Once you've updated the dependencies in `pom.xml`, you'll need to refresh your Maven project in your IDE for the dependency changes to take effect. How to do this varies from IDE to IDE.

Step 4. Write the unit test code

Below is an example unit test which contains one test method: `testOutputContainsExistingPages`. The code is quite heavily commented (much more than you would need normally), and detailed explanation of the important parts comes after the code sample.

```

package com.atlassian.plugins.tutorial.unittesting;

import com.atlassian.confluence.pages.Page;
import com.atlassian.confluence.pages.PageManager;
import com.atlassian.confluence.renderer.PageContext;
import com.atlassian.confluence.spaces.SpaceManager;
import junit.framework.TestCase;

import java.util.Arrays;
import java.util.HashMap;

import static org.mockito.Mockito.*;

/**
 * Testing {@link com.atlassian.plugins.tutorial.unittesting.ExampleMacro}
 */
public class TestExampleMacro extends TestCase
{
    public void testOutputIncludesRecentPages() throws Exception
    {
        // create test page
        Page page = new Page();
        page.setTitle("Page title");

        // set up mocks
        PageManager pageManager = mock(PageManager.class);
        SpaceManager spaceManager = mock(SpaceManager.class);

        // set up stub method to return our test page
        when(pageManager.getRecentlyAddedPages(55, "DS")).thenReturn(Arrays.asList(page));

        // create the macro
        ExampleMacro exampleMacro = new ExampleMacro(pageManager, spaceManager);

        // verify that the output contains the page title
        String output = exampleMacro.execute(new HashMap(), "", new PageContext(page));
        assertTrue("Output should contain page title but was: " + output,
                   output.contains(page.getTitle()));
    }
}

```

Test structure

First, on line 19, we renamed our test to be more descriptive: `testOutputIncludesRecentPages`.

Lines 22-23 create a test `Page` object, representing a Confluence page. This page is configured with the minimum details necessary for testing: just a page title.

Line 12 is a static import statement for Mockito, which is used to create *mock objects* to use in place of the Confluence classes for testing. See the section below for more information about mock objects.

Line 33 creates a new instance of your `ExampleMacro` class, then line 36 executes it and captures the output.

The most important line of the test is the assertion on lines 37-38. This verifies that the output includes the title of our test page. If this assertion passes, we can be sure that this aspect of the behaviour of our macro is correct.

```

assertTrue("Output should contain page title but was: " + output,
           output.contains(page.getTitle()));

```

JUnit provides a variety of assertion methods to all subclasses of `TestCase`. The three most common are `assertEquals`, `assertTrue` and `assertFalse`. These methods check the condition implied by their name and throw a particular kind of exception if the condition is not true. JUnit checks whether that exception is thrown and marks the test as failed if it is.

Using mock objects

This test uses Mockito to create a *mock object* – a customisable replacement for another class – to make testing easier. Mockito mock objects are created from an interface. Lines 26-27 use `Mockito.mock()` to create a mock instance of the `PageManager` and `SpaceManager` interfaces.

```
// set up mocks
PageManager pageManager = mock(PageManager.class);
SpaceManager spaceManager = mock(SpaceManager.class);
```

With a Mockito mock, any method call on the object will do nothing and return the default value for the method's return type. So, a method on the interface that returns a String will do nothing and return null, if the method is not customised by the test.

Mock objects can be granted additional behaviour by *stubbing* a method with `Mockito.when()`. Line 30 creates a *stub method call* on the `pageManager` mock object. When the `pageManager.getRecentlyAddedPages()` is called – with exactly the same arguments as on line 30: 55, "DS" – the stub method provided by Mockito will return a list with our test page in it.

```
// set up stub method to return our test page
when(pageManager.getRecentlyAddedPages(55, "DS")).thenReturn(Arrays.asList(page));
```

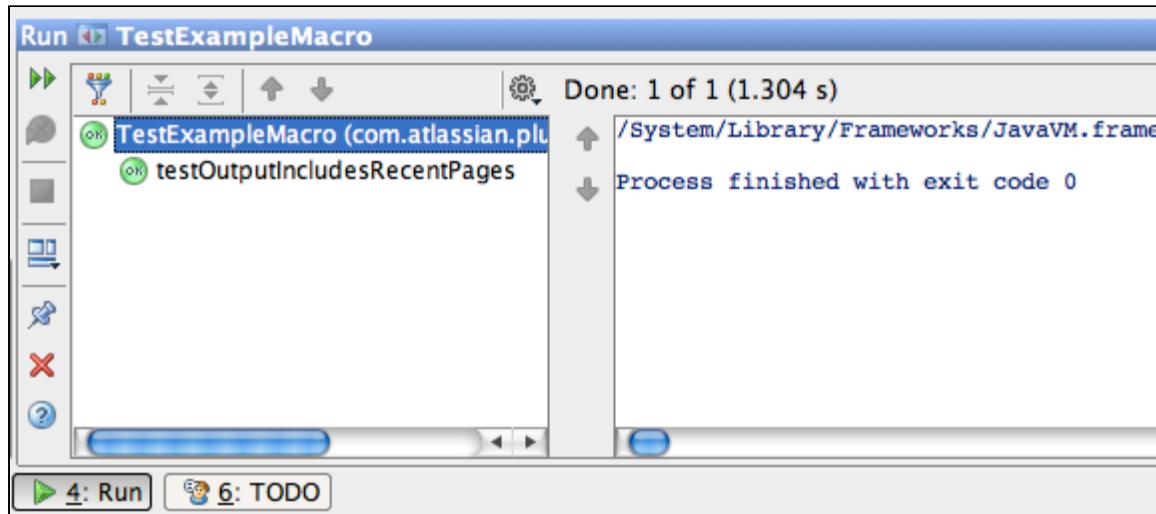
The parameters to our stub method are hard-coded here because the same parameters are also hard-coded in the example implementation code. If we wanted to return the same list regardless of the parameter values, we could use `Mockito.anyInt()` and `Mockito.anyString()` instead:

```
when(pageManager.getRecentlyAddedPages(anyInt(), anyString())).thenReturn(Arrays.asList(page));
```

See the [Mockito documentation](#) for more examples of how to use stubbing and verification with Mockito.

Step 5. Run the tests in your IDE

In Eclipse and IDEA, you can right-click on the name of your test class in your source file and select 'Run As > JUnit Test' or just 'Run TestExampleMacro'. This will run the test case and give you a panel with information about the tests like shown below.



If you have a problem running the test, make sure you copied the code above exactly. In particular, make sure:

- your test class extends `junit.framework.TestCase`
- your test method starts with 'test' and is `public void`
- you have included the static import for `org.mockito.Mockito.*`.

Step 6. Run the tests with Maven

To run the project tests with Maven, at the command line you can run `atlas-mvn test` or `atlas-unit-test`. You should see test results in the output like the following, somewhere near the middle of the build output:

```
-----
T E S T S
-----
Running com.atlassian.plugins.tutorial.unittesting.TestExampleMacro
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.413 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

If any of your tests fail, the build will abort at this step. This lets you use `atlas-install-plugin` or `atlas-release`, confident that the installation or release will only if the tests pass.

Now we're going to start the next stage of the tutorial, which adds another unit test and uses Java annotations to automatically generate mock objects. To start working with the code at this point, you can check it out using Subversion from here:

- <http://svn.atlassian.com/svn/public/contrib/tutorials/tutorial-unit-testing-plugin/tags/single-test>

Step 7. Adding another test

With the addition of a second test, it become necessary to split some duplicated code out into fields and a `setUp()` and `tearDown()` method. But first, we'll see what the new test looks like with all the duplicated code still in it.

```
public void testOutputIncludesCurrentUser() throws Exception
{
    // create test user
    DefaultUser user = new DefaultUser("test");
    user.setFullName("Test User");

    // set up mocks
    PageManager pageManager = mock(PageManager.class);
    SpaceManager spaceManager = mock(SpaceManager.class);

    // create the macro
    ExampleMacro exampleMacro = new ExampleMacro(pageManager, spaceManager);

    // set current user to test user
    AuthenticatedUserThreadLocal.setUser(user);
    try
    {
        // verify that the output contains the current user
        String output = exampleMacro.execute(new HashMap(), "", new PageContext());
        assertTrue("Output should contain current user but was: " + output,
                   output.contains(user.getFullName()));
    }
    finally
    {
        // reset current user
        AuthenticatedUserThreadLocal.setUser(null);
    }
}
```

Here again, the most important part of this test code is the assertion (lines 63-64), which verifies that the current user is included in the output if it is set. In Confluence, the current user is stored in `AuthenticatedUserThreadLocal`, so we create a test user and set it there before the test executes (line 58). In the `finally` block, we set the current user back to `null` so any subsequent tests won't be affected by us changing the state of `AuthenticatedUserThreadLocal`.

Below is the complete test class code after duplication has been removed and the mock objects are created with annotations.

```

package com.atlassian.plugins.tutorial.unittesting;

import com.atlassian.confluence.pages.Page;
import com.atlassian.confluence.pages.PageManager;
import com.atlassian.confluence.renderer.PageContext;
import com.atlassian.confluence.spaces.SpaceManager;
import com.atlassian.confluence.user.AuthenticatedUserThreadLocal;
import com.atlassian.user.impl.DefaultUser;
import junit.framework.TestCase;

import java.util.Arrays;
import java.util.HashMap;

import static org.mockito.Mockito.*;
import static org.mockito.MockitoAnnotations.*;

/**
 * Testing {@link com.atlassian.plugins.tutorial.unittesting.ExampleMacro}
 */
public class TestExampleMacro extends TestCase
{
    @Mock private PageManager pageManager;
    @Mock private SpaceManager spaceManager;

    private ExampleMacro exampleMacro;

    @Override
    protected void setUp() throws Exception
    {
        super.setUp();
        initMocks(this);
        exampleMacro = new ExampleMacro(pageManager, spaceManager);
    }

    public void testOutputIncludesRecentPages() throws Exception
    {
        Page testPage = new Page();
        testPage.setTitle("Page title");

        when(pageManager.getRecentlyAddedPages(55, "DS")).thenReturn(Arrays.asList(testPage));

        String output = exampleMacro.execute(new HashMap(), "", new PageContext(testPage));
        assertTrue("Output should contain page title but was: " + output,
                   output.contains(testPage.getTitle()));
    }

    public void testOutputIncludesCurrentUser() throws Exception
    {
        DefaultUser testUser = new DefaultUser("test");
        testUser.setFullName("Test User");

        // set current user to test user
        AuthenticatedUserThreadLocal.setUser(testUser);
        try
        {
            String output = exampleMacro.execute(new HashMap(), "", new PageContext());
            assertTrue("Output should contain current user but was: " + output,
                       output.contains(testUser.getFullName()));
        }
        finally
        {
            // reset current user
            AuthenticatedUserThreadLocal.setUser(null);
        }
    }
}

```

JUnit `setUp()` and `tearDown()` methods

JUnit convention is that the `setUp()` method is run before each test in a class, and the `tearDown()` is run after each test. You put common initialisation code in these methods when you override them.

In the case of this test class, we override the `setUp()` to initialise our mock objects and the `exampleMacro` object which will be used for

testing.

Using Mockito annotations to construct mock objects

This test uses the `MockitoAnnotations.Mock` annotation (lines 22-23) to indicate that the `pageManager` and `spaceManager` fields should be initialised with mock objects of their respective types: `PageManager` and `SpaceManager`. However, the annotation does nothing by itself. You need to call the code that inspects them and initialises them in your test's `setUp()` method.

Line 31 calls `MockitoAnnotations.initMocks()` to initialise all the fields annotated with `@Mock` in the current test object.

```
@Override  
protected void setUp() throws Exception  
{  
    super.setUp();  
    initMocks(this);  
    exampleMacro = new ExampleMacro(pageManager, spaceManager);  
}
```

Note that we call `super.setUp()` at the top of this method (line 30). This is not strictly necessary, but it is good practice in case we later decide to change the parent class of our test.

Ensuring the independence of unit tests

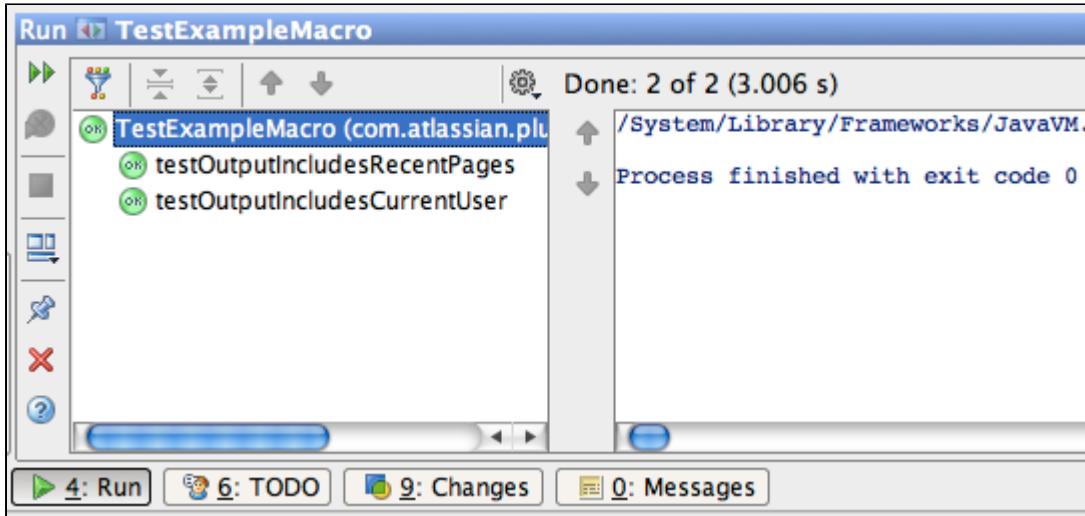
Once the mock objects and our test macro is set up, the tests proceed pretty much as normal. The only unusual feature of our new test is the `try ... finally` block. This is present to ensure that all our tests are as independent from each other as possible. Because the test is setting the current user by changing a shared object, `AuthenticatedUserThreadLocal`, it is also the responsibility of this test to set it back to the default value afterwards. We use a `finally` block to ensure that the user thread-local is *always* reset, even if the test itself fails (lines 60-64).

The main reason to ensure the independence of tests is because JUnit tests can be run in any order. In fact, the order varies across different Java versions. If your tests are not independent from each other, and they run them in a different order on a CI server, you may see unusual build failures which are difficult to debug locally.

The other reason to keep your tests independent of each other is to do with failure. If one test fails, you want to avoid cascading that failure through the remainder of the tests.

Step 8. Run all the tests again

Run the tests in your IDE again to make sure they all pass.



Two tests passing in IDEA

Run the tests at the command line with `atlas-mvn test` or `atlas-unit-test` to check that they pass there too.

```
-----  
TESTS  
-----  
Running com.atlassian.plugins.tutorial.unittesting.TestExampleMacro  
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.512 sec  
  
Results :  
  
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
```

To see the source code for the final version of this tutorial, you can check it using Subversion from this URL:

- <https://svn.atlassian.com/svn/public/contrib/tutorials/tutorial-unit-testing-plugin/trunk>

Where to next?

From here, you may wish to read:

- Plugin Testing Resources and Discussion
- Plugin Tutorial - Writing Integration Tests for your JIRA plugin
- Other tutorials

Standalone Gadget Tutorial - Writing a JQL Gadget



Level of experience: Beginner

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'beginner' level, so you can follow it even if you have never developed a plugin before.

On this page:

- Overview
- Step 1. Create the Gadget Specification
- Step 2. Update the Gadget Title and Other Descriptive Details
- Step 3. Make the Gadget Available on a Server
- Step 4. Add the Gadget to a Confluence Page for Testing
- Step 5. Make the Gadget Do Something Useful
- Step 6. Update the Gadget on the Server
- Step 7. Configure OAuth
- Step 8. Test Working Gadget on Confluence Page
- Step 9. Make the Gadget Look Better

Overview

This tutorial shows you how to write a gadget that will load data from a JIRA JQL Query into the gadget window and allow the user to click on an item to load the issues content in their browser. For simplicity this tutorial we will pull data from a local installation of JIRA (<http://localhost:8070>) into a local installation of Confluence (<http://localhost:8080/confluence>) and using a gadget.xml file coming from a different web server, in this case they are being served by our [SVN server](#). To adapt the tutorial to your environment simply change out the server name ("localhost") and ports ("8070" and "8080") as appropriate.

The reason this tutorial uses local installations of both Confluence and JIRA is because this gadget requires that you configure Confluence as an OAUTH consumer for JIRA. Therefore this tutorial requires that the reader be an Administrator on both Confluence and JIRA. For information on installing [Confluence](#) or [JIRA](#) please refer to the install documentation.

Your gadget will be a 'standalone' gadget. That means that it will consist entirely of HTML, CSS and Javascript, all contained within the gadget XML specification. There is no plugin involved, you can place this gadget xml file on any web server eliminating the need to create a plugin to host your gadget. If you are interested, you can [compare standalone gadgets and gadgets embedded in plugins](#).

This gadget assumes you have an understanding of how OAUTH works and follows the "[Sample Gadget Walkthrough](#)" from Google.

Step 1. Create the Gadget Specification

1. Copy the following code to create the basic XML file that will become your gadget specification:

```
-----
```

2. Paste the code into a text editor and save the file as `jql-gadget.xml`

Step 2. Update the Gadget Title and Other Descriptive Details

1. Update the values of the following attributes in the `<ModulePrefs>` element of your gadget specification:

| | |
|---------------------------|--|
| <code>title</code> | Enter 'JQL Gadget'. |
| <code>description</code> | Enter 'This gadget pulls content from a JIRA JQL Query and displays the results in the gadget window'. |
| <code>author</code> | Enter your own name. |
| <code>author_email</code> | Enter your own email address or remove this attribute. |

- Update the value of the categories parameter of the gadget-directory feature to 'JIRA'. This controls which category your gadget appears in within the JIRA gadget directory. NOTE: Valid values are "JIRA", "Confluence", "FishEye", "Crucible", "Crowd", "Clover", "Bamboo", "Admin", "Charts", "External Content", and "Other".

The resulting XML should look something like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="JQL Gadget"
    description="This gadget pulls content from an arbitrary JIRA JQL Query and displays the results
    in the gadget window"
    author="Atlassian"
    author_email="sales@atlassian.com">
    <Optional feature="gadget-directory">
      <Param name="categories">JIRA</Param>
    </Optional>
  </ModulePrefs>
  <Content type="html">
    <![CDATA[
      Hello world!
    ]]>
  </Content>
</Module>
```

Step 3. Make the Gadget Available on a Server

Because you are developing a standalone gadget, you can host your gadget specification on any server that will make it available to a **gadget container** such as a Confluence page or the JIRA Dashboard. You can host the gadget xml file on any webserver at your disposal. If you want to use the same XML that I used you can follow along using the hosted files from our [SVN server](#). NOTE: These files assume that you have JIRA running at <http://localhost:8070> and Confluence running at <http://localhost:8080/confluence>.

Each file is saved after each step: `jql-gadget-step1.xml`, `jql-gadget-step2.xml`, etc.

Step 4. Add the Gadget to a Confluence Page for Testing

Your gadget can already do something: It can say 'Hello world!'. Test it by adding it to a Confluence page.

- Go to Confluence Administration and click '**External Gadgets**'.
- The '**External Gadgets**' screen appears, showing the list of gadgets available in the Macro Browser.
- Enter the Gadget Specification URL and click '**Add**'.



- Edit a Confluence Page that you want to add your gadget to.

- Click the '**Insert/Edit Macro**' button.
- Enter 'JQL' in the search box



- Highlight your gadget and click the '**Add**' button to add the wiki markup for the gadget macro to the wiki page.



Step 5. Make the Gadget Do Something Useful

Now you will write the JavaScript and HTML code to retrieve data from JIRA so we can display the information in the gadget on a Confluence page.

The `<Content>` element in your gadget specification contains the working parts of the gadget. The `<Content>` element consists of:

- A CDATA declaration, to prevent the XML parser from attempting to parse the gadget content. Include '`<! [CDATA[`' (without the quotes) at the beginning and '`]]>`' (without the quotes) at the end of your `<Content>` element.
- Optional static HTML. When a dashboard renders the gadget, it will render this HTML.
- Optional JavaScript. You can declare JavaScript functions and call them in the usual way. Refer to the [OpenSocial JavaScript API](#) for details of gadget-specific API functions that any OpenSocial gadget container should support.
- Optional CSS style sheets.

In this tutorial you will need a JavaScript function that makes a REST call to JIRA to retrieve data. Out of the box JIRA provides a rich [Gadget Javascript Framework](#) that eases development of gadgets as plugins. We are going to take advantage of these helper javascript files from the remote server to illustrate how easy it can be to get a customized gadget running without touching any plugin code.

To start continue filling out the `ModulePrefs` to include a screenshot and thumbnail for the gadget - these are optional - but will make the gadget look better in the gadget directory.

| | |
|------------|---|
| screenshot | http://svn.atlassian.com/svn/public/contrib/tutorials/jira-standalone-jql-gadget/static/images/jql-gadget-screenshot.png |
| thumbnail | http://svn.atlassian.com/svn/public/contrib/tutorials/jira-standalone-jql-gadget/static/images/jql-gadget-thumb.png |

We also need to add the settings to enable this gadget for OAUTH by adding this XML in the `ModulePrefs` tag

```

<Require feature="oauthpopup" />
<OAuth>
  <Service>
    <Access url="http://localhost:8070/plugins/servlet/oauth/access-token" method="POST" />
    <Request url="http://localhost:8070/plugins/servlet/oauth/request-token" method="POST" />
    <Authorization
url="http://localhost:8070/plugins/servlet/oauth/authorize?oauth_callback=http%3A%2F%2Foauth.gmodules
/>
  </Service>
</OAuth>

```

Now change the `<content>` of the gadget to look like this:

```

<Content type="html">
  <![CDATA[
  <!-- shindig oauth popup handling code -->
  <script
src="http://svn.atlassian.com/svn/public/contrib/tutorials/jira-standalone-jql-gadget/static/js/popup
<div id="main" style="display: none">
  </div>

  <div id="approval" style="display: none">
    Personalize this gadget</a>
  </div>

  <div id="waiting" style="display: none">
    Please click
    <a href="#" id="approvaledone">I've approved access</a>
    once you've approved access to your data.
  </div>

```

```

<script type="text/javascript">
    // Display UI depending on OAuth access state of the gadget (see <divs> above).
    // If user hasn't approved access to data, provide a "Personalize this gadget" link
    // that contains the oauthApprovalUrl returned from makeRequest.
    //
    // If the user has opened the popup window but hasn't yet approved access, display
    // text prompting the user to confirm that s/he approved access to data. The user
    // may not ever need to click this link, if the gadget is able to automatically
    // detect when the user has approved access, but showing the link gives users
    // an option to fetch their data even if the automatic detection fails.
    //
    // When the user confirms access, the fetchData() function is invoked again to
    // obtain and display the user's data.
    function showOneSection(toshow) {
        var sections = [ 'main', 'approval', 'waiting' ];
        for (var i=0; i < sections.length; ++i) {
            var s = sections[i];
            var el = document.getElementById(s);
            if (s === toshow) {
                el.style.display = "block";
            } else {
                el.style.display = "none";
            }
        }
    }

    // Process returned JSON feed to display data.
    function showResults(result) {
        showOneSection('main');

        var titleElement = document.createElement('div');
        var nameNode = document.createTextNode(result.description);
        document.getElementById("main").appendChild(nameNode);
        document.getElementById("main").appendChild(document.createElement("br"));

        var bodyNode = document.createElement("div");
        bodyNode.innerHTML = result.table;
        document.getElementById("main").appendChild(bodyNode);
    }

    // Invoke makeRequest() to fetch data from the service provider endpoint.
    // Depending on the results of makeRequest, decide which version of the UI
    // to ask showOneSection() to display. If user has approved access to his
    // or her data, display data.
    // If the user hasn't approved access yet, response.oauthApprovalUrl contains a
    // URL that includes a Google-supplied request token. This is presented in the
    // gadget as a link that the user clicks to begin the approval process.
    function fetchData() {
        var params = {};
        url = "http://localhost:8070/rest/gadget/1.0/issueTable/jql?jql=";
        url = url + escape("assignee = currentUser() AND resolution = unresolved ORDER BY priority
DESC, created ASC");

        params[gadgets.io.RequestParameters.CONTENT_TYPE] = gadgets.io.ContentType.JSON;
        params[gadgets.io.RequestParameters.AUTHORIZATION] = gadgets.io.AuthorizationType.OAUTH;
        params[gadgets.io.RequestParameters.METHOD] = gadgets.io.MethodType.GET;

        gadgets.io.makeRequest(url, function (response) {
            if (response.oauthApprovalUrl) {
                // Create the popup handler. The onOpen function is called when the user
                // opens the popup window. The onClose function is called when the popup
                // window is closed.
                var popup = shindig.oauth.popup({
                    destination: response.oauthApprovalUrl,
                    windowOptions: null,
                    onOpen: function() { showOneSection('waiting'); },
                    onClose: function() { fetchData(); }
                });
                // Use the popup handler to attach onclick handlers to UI elements. The
                // createOpenerOnClick() function returns an onclick handler to open the
                // popup window. The createApprovedOnClick function returns an onclick
                // handler that will close the popup window and attempt to fetch the user's
                // data again.
            }
        });
    }

```

```
var personalize = document.getElementById('personalize');
personalize.onclick = popup.createOpenerOnClick();
var approvaledone = document.getElementById('approvaledone');
approvaledone.onclick = popup.createApprovedOnClick();
showOneSection('approval');
} else if (response.data) {
    showOneSection('main');
    showResults(response.data);
} else {
    // The response.oauthError and response.oauthErrorText values may help debug
    // problems with your gadget.
    var main = document.getElementById('main');
    var err = document.createTextNode('OAuth error: ' +
        response.oauthError + ': ' + response.oauthErrorText);
    main.appendChild(err);
    showOneSection('main');
}
}, params);
}
// Call fetchData() when gadget loads.
gadgets.util.registerOnLoadHandler(fetchData);
</script>
```

```
] ]>  
</Content>
```

Check out the resulting XML in `jq1-gadget-step3.xml` .

Step 6. Update the Gadget on the Server

1. Follow the steps from 'Step 4' above to add this version of the gadget to your confluence instance and add it to the page for testing.
 2. If you make changes to the xml file you may need to restart Confluence to pick up the changes.

1

The gadget caching issue

The gadget caching issue
Confluence will cache the gadget specification, with the result that your updates will not appear in Confluence until the cache has timed out or you restart Confluence. This caching problem affects other containers too, including iGoogle. There are workarounds, but these are too complex for this tutorial. Here's a short summary of the workarounds, in case you find them useful:

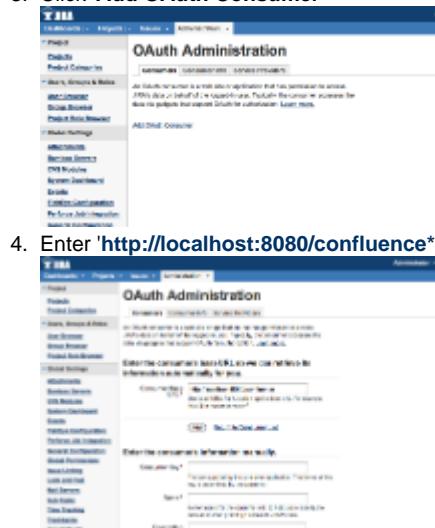
- In Confluence, you need to restart Confluence or run it in development mode.
 - In iGoogle, you can use the Google 'My Gadgets' gadget, which allows you to turn off caching for specific gadgets. See our [\[tips on using the iGoogle development environment\]](#).

When you test the gadget you might see an error that says '**OAuth error: consumer_key_unknown**'. This indicates that OAuth has not been configured between JIRA and Confluence. We will do this in the next step.

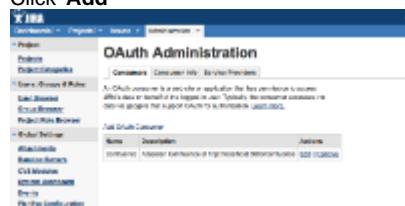


Step 7. Configure OAuth

1. To configure OAuth log in to JIRA as an administrator.
 2. Scroll down to the '**System**' section of the Administration options and click '**OAuth**'.
 3. Click '**Add OAuth Consumer**'.



- ## 5 Click 'Add'



6. If you click into the new consumer you will see that it has been configured with the secret key that allows OAuth requests between the two servers.

The screenshot shows the 'OAuth Administration' screen in Jira. A consumer entry for 'Issues Assigned to Me' is selected. The consumer key is 'flapjack-for-confluence' and the consumer secret is 'PRX-1'. The consumer description is 'Issues Assigned to Me'. The consumer URL is 'http://localhost:8080/confluence/plugins/gadgets/flapjack_for_confluence_gadget.html'. The consumer is listed under the 'Available Consumers' section.

Step 8. Test Working Gadget on Confluence Page

After configuring OAuth you should now be able to authorise the gadget for use by Confluence.

1. Reload the page that contains the gadget.
2. Click the 'Authorise this Gadget' link

The screenshot shows a Confluence page titled 'Issues Assigned to Me'. At the top, there is a blue header bar with the title. Below the header, there is a large white box containing the text 'Authorise this gadget'. At the bottom of this box, there are two buttons: 'Add Labels' and 'Add Comment'.

3. Click the 'Approve Access' button

The screenshot shows a 'Request for Access' dialog box. It contains the text: 'The gadget, "JQL Gadget", is trying to access your information.' and 'The gadget describes itself as: This gadget pulls content from a JQL Query and displays the results in the gadget window.' At the bottom of the dialog, there are two buttons: 'Approve Access' (highlighted) and 'Deny Access'.

4. View the list of issues

The screenshot shows a Confluence page titled 'Issues Assigned to Me'. At the top, there is a blue header bar with the title. Below the header, there is a table with one row. The table has columns for 'T' (Ticket icon), 'Key' (PRX-1), 'Summary' (My First Bug), and 'Pr' (Priority icon). At the bottom of the table, there are two buttons: 'Add Labels' and 'Add Comment'.

Step 9. Make the Gadget Look Better

Since the gadget is just HTML, Javascript and CSS you can change the look and feel of the gadget by adding some CSS to the file. A lot more can be done than this, but adding just a few classes makes the gadget look much better.

```

<style>
    #issuetable {margin:0 0 0.25em;width:100%;}
    #issuetable .issuetype {width:1%;}
    td.colHeaderLink {
        font-family:Arial,Helvetica,sans-serif;
        font-size:12px;
        white-space:nowrap;
    }
    .grid, .bordered {
        background-color:#FCFCFC;
        border:1px solid #D2D2D2;
        border-collapse:collapse;
        margin-bottom:0.5em;
        margin-top:0.5em;
    }
    .grid th, .grid td {
        border:1px solid #D2D2D2;
        padding:4px;
        vertical-align:top;
    }

```

</style>

Check out the resulting XML in `jq1-gadget-step4.xml`.

The screenshot shows a Jira gadget titled "Issues Assigned to Me". It displays a single issue: PRX-1, My First Bug. The issue has a priority of P1. The gadget interface includes a toolbar with icons for edit, delete, and refresh.



Congratulations, that's it

Your gadget is complete. Have a chocolate!

Next Steps

Now that you have created a working gadget, you may like to look at some more advanced topics:

- Internationalising your Gadget
- Packaging your Gadget as an Atlassian Plugin

Tutorial Requests



We're taking requests!

Over the years we've assembled a bunch of tutorials, ranging from [JIRA gadgets](#) and [Confluence macros](#) to [plugin unit testing](#) and [writing REST services](#). We've recorded "technical track" presentations from Summit and AtlasCamp, including some important fundamentals for

plugin developers like OSGi "gotchas", and several of us have blogged on advanced topics like writing Atlassian plugins in Scala and writing expressive tests using Hamcrest.

There's always room for more, though, and we're always looking for new topics to cover next. **So we'd like your input!** When you were writing your first plugins, or if you're just writing your first plugin now, what issues have you faced that we didn't have a good answer for? What documentation have you tried searching for, that you just couldn't find? What questions have you had that the forums weren't able to answer? Let us know here, by commenting below.

Thanks very much!
The Atlassian Developer Relations Team

Tutorial Templates

These are templates for various types of tutorial. Please feel free to copy a template to a new page and base your own tutorial on it.



Feel free to depart from the template

You can start with the template and then change the sections in your own tutorial when the template is no longer useful. The templates serve 2 purposes. The first is to make it easier for people to get started in writing a tutorial. The second is to give all the tutorials the same overall look and feel, so that the reader gets some sense of familiarity and comfort. But the minute the template stops being useful, anyone should feel free to dump or adapt the bits that are not useful.

Plugin tutorial templates:

- [Template for Plugin Tutorials](#)

Gadget tutorial templates:

- [Template for Plugin Gadget Tutorial](#)
- [Template for Standalone Gadget Tutorial](#)

Template for Plugin Tutorials



Please ignore this page unless you want to write a plugin tutorial

This page is just a template for tutorial developers to use when writing a new tutorial. The real tutorials are listed here: [Tutorials](#).

(To the tutorial author: This is a template. You can copy the content of this page and use it to create your own plugin tutorial. When you have finished, delete all sections that start with 'To the tutorial author' from your copy of the page. Otherwise the page will self-destruct when you finish reading it.)

(To the tutorial author: LEAVE THE DRAFT NOTICE (below) IN PLACE UNTIL THE TUTORIAL HAS PASSED REVIEW.)



Draft and under construction

This tutorial is a DRAFT and is under construction. (To the tutorial author: Please leave this warning in place until your tutorial has been reviewed and tested.)

(To the tutorial author: CHOOSE ONE OF THE 3 LEVELS BELOW (beginner, intermediate or advanced), LEAVE THAT ONE IN PLACE AND DELETE THE OTHER TWO.)



Level of experience: Beginner

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'beginner' level, so you can follow it even if you have never developed a plugin before.



Level of experience: Intermediate

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'intermediate' level. If you have never developed a plugin before, you may find this one a bit difficult.



Level of experience: Advanced

Our tutorials are classified as 'beginner', 'intermediate' and 'advanced'. This one is at 'advanced' level. If you have never developed a plugin before, we advise you to try a beginner tutorial first.

The source code of the plugin used in this tutorial is available in the Atlassian public source repository. You can check out the source code [here](#). (To the tutorial author: GIVE THE CORRECT LOCATION OF THE SOURCE CODE.)

On this page:

- Overview
- Required Knowledge
- Step 1. Create the Plugin Project
- Step 2. Add Plugin Metadata to the POM
- Step 3. Register the Plugin Module in the Plugin Descriptor
- Step 4. Write Java Classes
- Step 5. Build, Install and Run the Plugin
- Step 6. Write Unit and Integration Tests

Overview

This tutorial shows you how to (To the tutorial author: TELL THEM THE PURPOSE OF THE PLUGIN THAT WE WILL CREATE IN THIS TUTORIAL)

In order to do this, you will create a XXXXXXXXXXXX (To the tutorial author: GIVE THE TARGET APPLICATION e.g. JIRA) plugin. As with all plugins, your plugin will consist of the following components:

- Java classes encapsulating the plugin logic
- Resources for display of the plugin UI (To the tutorial author: REMOVE THIS LINE IF NOT APPLICABLE FOR THIS TUTORIAL)
- Plugin descriptor to enable the plugin module in (To the tutorial author: GIVE THE TARGET APPLICATION e.g. JIRA)

All these components will be contained within a single JAR file. Each component is further discussed in the examples below.

Required Knowledge

(To the tutorial author: EXPLAIN WHAT THE PLUGIN AUTHOR NEEDS TO KNOW BEFORE THEY CAN START THIS TUTORIAL. BELOW ARE SOME EXAMPLE WORDS. PLEASE ADAPT THEM TO FIT YOUR TUTORIAL.)

To complete this tutorial, you must already understand the basics of Java development: classes, interfaces, methods, how to use the compiler, and so on. You should also understand:

- How to create an Atlassian plugin project using the Atlassian Plugin SDK.
- How to open the plugin project in your IDE.
- How to compile your project and create a JAR file using Maven.

Step 1. Create the Plugin Project

First you need to set up your development environment and create your plugin project and source directories. Follow the instructions in our guides:

- Set up your development environment
- Create and install your plugin from a template

When you create the plugin using the above SDK, you will be prompted for some information to identify your plugin. Enter the following information:

- **group-id:** com.example.plugins.tutorial
- **artifact-id:** XXXXXXXXXXXX (To the tutorial author: GIVE THE ARTIFACT ID HERE)
- **version:** 1.0
- **package:** com.example.plugins.tutorial

Step 2. Add Plugin Metadata to the POM

Now you need to edit your POM (Project Object Model definition file) to add some metadata about your plugin and your company or organisation.

1. Edit the `pom.xml` file in the root folder of your plugin.
2. Add your company or organisation name and your website to the `<organization>` element:

```
<organization>
    <name>Example Company</name>
    <url>http://www.example.com/</url>
</organization>
```

3. Update the `<description>` element:

```
<description>This plugin XXXXXXXXXXXX</description>
```

(To the tutorial author: ENTER A SHORT DESCRIPTION ABOVE)

4. Save the file.

Step 3. Register the Plugin Module in the Plugin Descriptor

Now you will add the required plugin modules to your plugin descriptor at `src/main/resources/atlassian-plugin.xml`. The plugin descriptor is an XML file that identifies the plugin to XXXXXXXXXXXX (To the tutorial author: ADD THE APPLICATION NAME e.g. JIRA) and defines the functionality that the plugin requires.

Here's a basic plugin descriptor:

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" name="${project.artifactId}">
  plugins-version="2">
    <plugin-info>
      <description>${project.description}</description>
      <version>${project.version}</version>
      <vendor name="${project.organization.name}" url="${project.organization.url}" />
    </plugin-info>
  </atlassian-plugin>
```

Note that some of the information from the POM is transferred to the plugin descriptor using variable names such as `${project.artifactId}`.

You will need the following plugin modules:

(To the tutorial author: ADD THE MODULE TYPE(S) THAT WILL GO IN THE PLUGIN DESCRIPTOR)

Now you are ready to move onto writing some code to make your plugin do something.

Step 4. Write Java Classes

In this plugin, you want to (To the tutorial author: TELL THEM WHAT WE WANT OUR PLUGIN TO DO)

To do this, you will write a (To the tutorial author: TELL THEM WHAT MODULE TYPE WE WILL ADD).

(To the tutorial author: GIVE CODE SAMPLES AND NARRATIVE HERE.)

Step 5. Build, Install and Run the Plugin

Follow these steps to build and install your plugin, so that you can test your code:

- If you have not already started the application, start it now:
 - Open a command window and go to the plugin root folder (where the `pom.xml` is located).
 - Run `atlas-run`.
- From this point onwards, you can use the command line interface (CLI) as follows:
 - Open another command window and go to the plugin root folder.
 - Run `atlas-cli` in the second command window to start the CLI.
 - When you see the message 'Waiting for commands', run `pi` to compile, package and install the updated plugin.
- Go back to the browser. The updated plugin has been installed into the application, and you can test your changes.

The full instructions are in the [SDK guide](#).

Step 6. Write Unit and Integration Tests

(To the tutorial author: GIVE CODE SAMPLES AND NARRATIVE FOR UNIT AND INTEGRATION TESTS HERE.)



Congratulations, that's it
Have a chocolate!

Getting Involved in the Atlassian Developer Network

These are some of the ways you can get involved in the Atlassian Developer Network. We would love to have your contributions!

Join the Forum / Mailing List

Subscribe to our [blogs](#) or [newsletter](#), follow us on [Twitter](#) or [Facebook](#), learn on [Answers](#) (learn more [here](#)), or create a [my.atlassian.com](#) account and edit your Email Preferences.

Join the IRC channel

Use your favorite IRC client to join. Wikipedia has a [whole list](#) of clients to choose from. Internally we use [Colloquy](#) on Macs and [mIRC](#) on Windows computers. Additionally some of the regular chat clients are able to connect to IRC. Once you decided on a client, you will need to following details to join the IRC channel:

- Server: <http://freenode.net>
- Channel: #atlassiandev

If you have a question specifically for Atlassian developers, for best results, ask your question, and if you don't get a reply immediately, stay logged in for at least 24 hours, then come back and check for a reply. Most Atlassian developers are in Sydney, and are only monitoring the channel during Sydney business hours. Additionally, not all Atlassian developers stay constantly signed in to IRC, so if you ask it during Sydney business hours you've also got a much higher chance of getting a timely answer from the right developer.

Share your Plugin

You've spent long hours getting your plugin *just right*. Why not share it with the rest of the Atlassian Community? Here's how you can [release your plugin](#) to the Atlassian Community.

Submit a patch

Have you found a bug in an existing plugin? Something not quite right? Is there an itch you want to scratch? Many of our plugins are open-source and welcome contributions. With closed-source projects, including the products themselves, you are still welcome to submit patches for us to review. Here's [how to submit a patch](#).

Join an existing project

If you're really excited about a plugin, jump right in! Contact the plugin's sponsor and discuss how you might be able to collaborate.

RELATED TOPICS

Welcome to the Atlassian Developer Network

DevNet Discussion

- [IRC Chat Transcripts](#)
- [Plugin Testing Resources and Discussion](#)

IRC Chat Transcripts

This page contains IRC chat transcripts from the public IRC channel:

| | |
|---------|------------------|
| Server | irc.freenode.net |
| Channel | #atlassiandev |

- [atlassiandev_log-2010-01-16](#)
- [atlassiandev_log-2010-01-17](#)
- [atlassiandev_log-2010-01-18](#)
- [atlassiandev_log-2010-01-19](#)
- [atlassiandev_log-2010-01-20](#)
- [atlassiandev_log-2010-01-21](#)
- [atlassiandev_log-2010-01-22](#)
- [atlassiandev_log-2010-01-23](#)
- [atlassiandev_log-2010-01-24](#)
- [atlassiandev_log-2010-01-25](#)
- [atlassiandev_log-2010-01-26](#)
- [atlassiandev_log-2010-01-27](#)
- [atlassiandev_log-2010-01-28](#)
- [atlassiandev_log-2010-01-29](#)
- [atlassiandev_log-2010-01-30](#)
- [atlassiandev_log-2010-01-31](#)
- [atlassiandev_log-2010-02-01](#)
- [atlassiandev_log-2010-02-02](#)
- [atlassiandev_log-2010-02-03](#)
- [atlassiandev_log-2010-02-04](#)
- [atlassiandev_log-2010-02-05](#)
- [atlassiandev_log-2010-02-06](#)
- [atlassiandev_log-2010-02-07](#)
- [atlassiandev_log-2010-02-08](#)
- [atlassiandev_log-2010-02-09](#)
- [atlassiandev_log-2010-02-10](#)
- [atlassiandev_log-2010-02-11](#)
- [atlassiandev_log-2010-02-12](#)
- [atlassiandev_log-2010-02-13](#)
- [atlassiandev_log-2010-02-14](#)
- [atlassiandev_log-2010-02-15](#)

- atlassiandev_log-2010-02-16
- atlassiandev_log-2010-02-17
- atlassiandev_log-2010-02-18
- atlassiandev_log-2010-02-19
- atlassiandev_log-2010-02-20
- atlassiandev_log-2010-02-21
- atlassiandev_log-2010-02-22
- atlassiandev_log-2010-02-23
- atlassiandev_log-2010-02-24
- atlassiandev_log-2010-02-25
- atlassiandev_log-2010-02-26
- atlassiandev_log-2010-02-27
- atlassiandev_log-2010-02-28
- atlassiandev_log-2010-03-01
- atlassiandev_log-2010-03-02
- atlassiandev_log-2010-03-03
- atlassiandev_log-2010-03-04
- atlassiandev_log-2010-03-05
- atlassiandev_log-2010-03-06
- atlassiandev_log-2010-03-07
- atlassiandev_log-2010-03-08
- atlassiandev_log-2010-03-09
- atlassiandev_log-2010-03-10
- atlassiandev_log-2010-03-11
- atlassiandev_log-2010-03-12
- atlassiandev_log-2010-03-13
- atlassiandev_log-2010-03-14
- atlassiandev_log-2010-03-15
- atlassiandev_log-2010-03-16
- atlassiandev_log-2010-03-17
- atlassiandev_log-2010-03-18
- atlassiandev_log-2010-03-19
- atlassiandev_log-2010-03-20
- atlassiandev_log-2010-03-21
- atlassiandev_log-2010-03-22
- atlassiandev_log-2010-03-23
- atlassiandev_log-2010-03-24
- atlassiandev_log-2010-03-25
- atlassiandev_log-2010-03-26
- atlassiandev_log-2010-03-27
- atlassiandev_log-2010-03-28
- atlassiandev_log-2010-03-29
- atlassiandev_log-2010-03-30
- atlassiandev_log-2010-03-31
- atlassiandev_log-2010-04-01
- atlassiandev_log-2010-04-02
- atlassiandev_log-2010-04-03
- atlassiandev_log-2010-04-04
- atlassiandev_log-2010-04-05
- atlassiandev_log-2010-04-06
- atlassiandev_log-2010-04-07
- atlassiandev_log-2010-04-08
- atlassiandev_log-2010-04-09
- atlassiandev_log-2010-04-10
- atlassiandev_log-2010-04-11
- atlassiandev_log-2010-04-12
- atlassiandev_log-2010-04-13
- atlassiandev_log-2010-04-14

atlassiandev_log-2010-01-16

[17:09:49 EST(+1100)] * atlbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
 [17:09:49 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myrall on 2010-01-13 13:10:53 EST(+1100)
 [18:15:00 EST(+1100)] * atlbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
 [18:15:00 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myrall on 2010-01-13 13:10:53 EST(+1100)
 [18:17:13 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
 [18:17:13 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myrall on 2010-01-13 13:10:53 EST(+1100)
 [20:40:32 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
 [20:40:32 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myrall on 2010-01-13 13:10:53 EST(+1100)
 [20:57:31 EST(+1100)] * sleberry (n=sleberry@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
 [21:57:22 EST(+1100)] * mrdon_ (n=dbrown@ppp121-45.static.internode.on.net) has left #atlassiandev
 [21:57:26 EST(+1100)] * mrdon_ (n=dbrown@ppp121-45.static.internode.on.net) has joined #atlassiandev

atlassiandev_log-2010-01-17

[10:28:04 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
 [10:28:04 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myrall on 2010-01-13 13:10:53 EST(+1100)

[11:05:55 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[21:30:31 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[21:30:31 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myrall on 2010-01-13 13:10:53 EST(+1100)

atlassiandev_log-2010-01-18

[07:37:32 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[07:42:41 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[08:31:04 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[09:27:40 EST(+1100)] <jenssschumacher> Thanks for the bot Don!
[09:27:51 EST(+1100)] * jenssschumacher (n=jens@59.167.164.33) has left #atlassiandev
[09:32:10 EST(+1100)] * jedi (i=mike@li55-224.members.linode.com) has joined #atlassiandev
[09:51:48 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
[09:54:11 EST(+1100)] <mryall> mrdon_- where are you hosting it?
[09:54:17 EST(+1100)] <mrdon> just my box for now
[09:54:25 EST(+1100)] <mrdon> need to work with IT to get a real server
[09:54:28 EST(+1100)] <mryall> mm

[09:54:32 EST(+1100)] <mryall> good luck with that 😊

[09:54:43 EST(+1100)] <mrdon> or I just may stick it on atlassian27 😊
[10:00:18 EST(+1100)] * pwyatt (n=pwyatt@59.167.164.33) has joined #atlassiandev
[10:07:17 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[10:08:33 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has joined #atlassiandev
[10:13:29 EST(+1100)] * mjensen (n=mjensen@59.167.164.33) has joined #atlassiandev
[10:13:31 EST(+1100)] <dezwart> This channel is now devoid of conversation since mrdon is now logging this channel for public viewing.
[10:13:49 EST(+1100)] <pwyatt> Concur.
[10:14:02 EST(+1100)] <Carlfish> Anything you say can and will be recorded by The Google forever.
[10:14:35 EST(+1100)] <Carlfish> Sort of puts a damper on non-essential conversation.
[10:14:42 EST(+1100)] <dezwart> All your base, are belong to us.
[10:14:49 EST(+1100)] <dezwart> Oooh
[10:14:57 EST(+1100)] <dezwart> XSS via IRC.
[10:15:08 EST(+1100)] <pwyatt> I'm going to construct some additional pylons, that might fix it.
[10:18:24 EST(+1100)] <mrdon> I can take the bot offline too
[10:18:38 EST(+1100)] <mrdon> just thought it would make it easier to follow stuff when offline
[10:18:55 EST(+1100)] <mrdon> can confluence create a robots.txt file?
[10:19:17 EST(+1100)] <mryall> no, you can put one in the webapp, though
[10:30:09 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[11:11:03 EST(+1100)] * nmuldoon2 (n=Nicholas@59.167.164.33) has joined #atlassiandev
[11:20:41 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[11:21:03 EST(+1100)] * nmuldoon2 (n=Nicholas@59.167.164.33) has joined #atlassiandev
[11:21:10 EST(+1100)] * mjensen (n=mjensen@59.167.164.33) has left #atlassiandev
[11:41:51 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has joined #atlassiandev
[11:44:29 EST(+1100)] * dchui (n=dchui@202.169.29.34) has joined #atlassiandev
[12:00:52 EST(+1100)] * jenssschumacher (n=jens@59.167.164.33) has joined #atlassiandev
[12:23:17 EST(+1100)] <jazzy> you never know which user(s) in here could be recording and publishing transcripts, this is an open IRC channel
[12:23:33 EST(+1100)] <jazzy> like jed said, permanent record should be assumed by default
[12:25:07 EST(+1100)] * ryant (n=ryant@59.167.164.33) has joined #atlassiandev
[12:25:58 EST(+1100)] * ryant (n=ryant@59.167.164.33) has joined #atlassiandev
[12:26:04 EST(+1100)] <jazzy> i know people that record logs from every IRC channel they are in and publish stats of them to their webservers, including random quotes etc, I don't know why they do that but they do
[12:27:34 EST(+1100)] <Carlfish> A lot of people auto-log everything on IRC, but there's a difference between ad hoc private logging, log stats, and fully published logs.
[12:33:22 EST(+1100)] <jazzy> nevertheless, we're a corporate company with competitors, there's nothing stopping evil competitor X from coming in this chat room and collecting dirt on us, we must be guarded in what we say anyway
[12:46:18 EST(+1100)] <Carlfish> Obvious point is obvious. My only concern is the potential effect on the kind of casual discussions that keep IRC channels alive.
[12:46:24 EST(+1100)] * Carlfish shrugs.
[12:50:09 EST(+1100)] <mryall> yeah, I'd tend to leave logging disabled until we actually need it for something
[12:50:16 EST(+1100)] <mryall> or at least have the logs protected somehow
[13:28:15 EST(+1100)] <pwyatt> It's not that I'd say anything that I don't want repeated, more that if this is officially published by us then it should be official discussion, not casual conversation. Which is sad.
[13:41:23 EST(+1100)] <jenssschumacher> How about we keep the logs private on CAC for the time being. However, I would like to keep a log for Atlassian. We can always decide later to publish it if we determine that they contain useful information.
[13:41:53 EST(+1100)] <dezwart> I suspect this is all academic as this is on an open IRC network.
[15:08:31 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[15:56:18 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[15:57:19 EST(+1100)] * nmuldoon1 (n=Nicholas@59.167.164.33) has joined #atlassiandev
[17:17:27 EST(+1100)] * nmuldoon (n=Nicholas@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:17:34 EST(+1100)] * dezwart_(n=pdzwart@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:17:34 EST(+1100)] * mryall_(n=mryall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:17:35 EST(+1100)] * Carlfish_(n=cmiller@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:17:36 EST(+1100)] * ckiehl1 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:17:50 EST(+1100)] * ryant_(n=ryant@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:17:50 EST(+1100)] * mrdon_(n=mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:17:50 EST(+1100)] * sleberrir_(n=sleberri@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:18:24 EST(+1100)] * pwyatt1 (n=pwyatt@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:18:38 EST(+1100)] * dezwart_(n=pdzwart@ATL146140-1.gw.connect.com.au) has left #atlassiandev
[17:22:16 EST(+1100)] * ChanServ sets mode +o mryall

```
[17:25:00 EST(+1100)] * Carlfish_ (n=cmiller@59.167.164.33) has joined #atlassiandev
[17:25:08 EST(+1100)] * nmuldoon2 (n=Nicholas@59.167.164.33) has joined #atlassiandev
[17:25:09 EST(+1100)] * ckiehl (n=Adium@59.167.164.33) has joined #atlassiandev
[17:25:13 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[17:25:23 EST(+1100)] * sleberrigaud (n=sleberri@59.167.164.33) has joined #atlassiandev
[17:25:24 EST(+1100)] * ryant__ (n=ryant@59.167.164.33) has joined #atlassiandev
[17:26:55 EST(+1100)] * pwyatt (n=pwyatt@59.167.164.33) has joined #atlassiandev
[17:34:12 EST(+1100)] * mryall_ (n=mryall@59.167.164.33) has joined #atlassiandev
[17:34:13 EST(+1100)] * ChanServ sets mode +o mryall_
[17:44:17 EST(+1100)] * jenssschumacher (n=jens@59.167.164.33) has joined #atlassiandev
[18:09:24 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[21:37:38 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
```

atlassiandev_log-2010-01-19

```
[00:18:54 EST(+1100)] * japaya (n=jasper@d51530A0F.static.telenet.be) has joined #atlassiandev
[00:19:54 EST(+1100)] <japaya> Hello, I'm developing a confluence macro plugin but I'm having problems with including my own .jar files
[00:20:46 EST(+1100)] <japaya> i have added them to my include path but now I keep receiving an unhandled exception type
[00:21:38 EST(+1100)] <japaya> is there a good tutorial/help on how to include custom .jar packages into your own plugins?
[00:38:45 EST(+1100)] <mrdon> is this plugin installed dynamically?
[00:38:56 EST(+1100)] <mrdon> i.e. is it deployed via WEB-INF/lib or via the web UI?
[00:41:02 EST(+1100)] <japaya> I have deployed the plugin using the SDK and maven
[00:41:18 EST(+1100)] <japaya> so everything happens automatically
[00:43:25 EST(+1100)] <mrdon> so then the dependency should already be in META-INF/lib
[00:43:36 EST(+1100)] <mrdon> run 'jar -tf target/myjar.jar'
[00:43:55 EST(+1100)] <mrdon> and you should find the dependent jar in the META-INF/lib directory inside the jar
[00:50:44 EST(+1100)] <mrdon> anyway, if using the sdk, if you mark your dependency as a compile-time dependency, it'll be included in the jar automatically
[00:57:50 EST(+1100)] <japaya> so I probably have to define that dependency in pom.xml?
[04:48:10 EST(+1100)] * tmoore (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[04:48:11 EST(+1100)] * ChanServ sets mode +o tmoore
[05:31:46 EST(+1100)] * jnolen (n=Adium@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[05:33:22 EST(+1100)] * jnolen1 (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:33:23 EST(+1100)] * tmoore1 (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[06:23:03 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[06:23:12 EST(+1100)] * ChanServ sets mode +o bspeakmon
[06:29:25 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[06:31:10 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[06:51:07 EST(+1100)] * ChanServ sets mode +o tmoore
[06:56:30 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[06:58:39 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[08:31:22 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[08:31:29 EST(+1100)] * ChanServ sets mode +o bspeakmon
[08:57:37 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[09:05:59 EST(+1100)] <tmoore> sam you there?
[09:06:05 EST(+1100)] <sleberrigaud> yep
[09:06:13 EST(+1100)] <tmoore> I just came across this: https://studio.atlassian.com/browse/REST-51
[09:06:33 EST(+1100)] <tmoore> I've seen some vague references to PUT and DELETE not being supported in all browsers, but can't find any concrete information
[09:06:48 EST(+1100)] <tmoore> for example, the docs for jQuery.ajax recommend only using GET & POST
[09:07:50 EST(+1100)] <sleberrigaud> don't remember which one though
[09:08:02 EST(+1100)] <sleberrigaud> and I don't think they found any issues with it
[09:08:10 EST(+1100)] <tmoore> I've also seen vague references to some firewalls blocking other methods, but again, have no idea how widespread that is
[09:08:22 EST(+1100)] <tmoore> yeah, if I could just avoid the whole issue, that would be ideal
[09:09:46 EST(+1100)] <jedi> I know in Rails they emulate PUT and DELETE for their REST stuff with a parametrised POST
[09:11:33 EST(+1100)] <tmoore> how so?
[09:11:53 EST(+1100)] <tmoore> oh, just saw jedi's comment
[09:12:05 EST(+1100)] <tmoore> yeah, that's an issue, but the suggested solution I'm talking about uses an HTTP header
[09:12:18 EST(+1100)] <tmoore> which couldn't be exploited by an CSRF attack
[09:13:03 EST(+1100)] <tmoore> Sam, while I agree in principle, it could cause a big support load if all of a sudden all of the dashboard AJAX started to fail in deployments with crappy firewall rules
[09:13:32 EST(+1100)] <tmoore> after my experience with the gadget loopback situation I'm wary of creating a new one :-
[09:13:39 EST(+1100)] <sleberrigaud> 😊
[09:14:45 EST(+1100)] <tmoore> there's no way to configure the REST plugin to add in a post filter from within my plugin, is there?
[09:14:48 EST(+1100)] <sleberrigaud> I've never heard about issues with PUT and DELETE, might be worth checking out if/how products use them at the moment and understand if they have been an issue so far.
[09:15:44 EST(+1100)] <tmoore> wait can I add init-params in my <rest> module?
[09:15:49 EST(+1100)] <tmoore> how does that work?
[09:16:13 EST(+1100)] <tmoore> I can't add a post filter there?
[09:38:45 EST(+1100)] <mryall> the outbound web proxy we used at Mac Bank used to block some kinds of HTTP requests (e.g. CONNECT)
[09:39:13 EST(+1100)] <mryall> I'm not sure if it operated by only allowing GET and POST, or by specifically denying CONNECT
[09:39:15 EST(+1100)] <mryall> but it's a real risk
[09:54:07 EST(+1100)] <tmoore> yeah, and I imagine a terrible failure mode
[09:54:25 EST(+1100)] <tmoore> stuff would probably just mysteriously not work
[09:54:37 EST(+1100)] <tmoore> losing all of your changes in the process
[09:58:41 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
[09:59:28 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
```

[10:01:20 EST(+1100)] <mryall> yup
[10:02:42 EST(+1100)] <mryall> there's no way of knowing what kind of status code a proxy would reply if it doesn't support a particular method
[10:03:55 EST(+1100)] <mryall> it should probably be 405 Method Not Allowed, but I imagine it could also reply with several of the other 400 codes
[10:04:04 EST(+1100)] <mryall> not permitted, forbidden, etc.
[10:06:05 EST(+1100)] <tmoore> yeah, and regardless of what it returns, whatever you just did would get lost (unless we retried using POST, but in that case, why not just do that in the first place?)
[10:15:39 EST(+1100)] <mrdrone> there is a problem with PUT and DELETE when used via XMLHttpRequest in some browsers
[10:15:53 EST(+1100)] <mrdrone> some versions of Safari, iirc
[10:16:06 EST(+1100)] <tmoore> ah OK... not the IE6 usual suspect
[10:16:12 EST(+1100)] <mrdrone> GWT, for example, restricts you to POST and GET for that reason
[10:16:14 EST(+1100)] <tmoore> do we still support those versions of Safari?
[10:16:20 EST(+1100)] <mrdrone> not sure
[10:16:29 EST(+1100)] <mrdrone> I found out about it while digging through GWT source code
[10:17:07 EST(+1100)] <tmoore> so I think we're in general agreement that we need to support faking PUT and DELETE through POSTs
[10:17:17 EST(+1100)] <tmoore> I think it would be nice if the REST module type handled that automatically
[10:17:25 EST(+1100)] <tmoore> but we can trial the solution in AG
[10:18:20 EST(+1100)] <mryall> there's some header used by Google apparently, according to Raible:
http://raibledesigns.com/rd/entry/building_scalable_reliable_and_secure
[10:18:34 EST(+1100)] <mryall> "The biggest problem with REST is firewalls. Many firewalls don't allow PUT or DELETE. Google fixes this by adding a header that specifies a method override."
[10:19:20 EST(+1100)] <tmoore> yes, there seems to be a consensus forming around X-HTTP-Method-Override
[10:19:42 EST(+1100)] <mryall> yup: <http://code.google.com/apis/gdata/docs/2.0/basics.html#UpdatingEntry>
[10:19:59 EST(+1100)] <tmoore> Jersey can support it by installing an optional filter
[10:20:11 EST(+1100)] <tmoore> it shouldn't, because there's no way to specify the header on a form
[10:20:27 EST(+1100)] <sleberrigaud> yeah, I was thinking that as well talking to mryall
[10:20:56 EST(+1100)] <tmoore> anyway, I'm going to add that into AG now using a @HeaderParam
[10:21:11 EST(+1100)] <sleberrigaud> so I think it should be fine, as long as we're wary enough of not accepting POSTs with url-form-encoded parameters
[10:21:16 EST(+1100)] <tmoore> yeah
[10:21:28 EST(+1100)] <sleberrigaud> and have idem potent GET
[10:21:39 EST(+1100)] <tmoore> I'll put you on the review, if you don't mind
[10:22:27 EST(+1100)] <sleberrigaud> sure
[10:30:24 EST(+1100)] * mryall (n=mryall@59.167.164.33) has joined #atlassiandev
[10:30:25 EST(+1100)] * ChanServ sets mode +o mryall
[10:31:29 EST(+1100)] <mryall> hmm.. drag and drop of channels in Colloquy doesn't work so well 
[10:37:00 EST(+1100)] <mrdrone> use limechat and forget about messing with channel tabs
[10:49:45 EST(+1100)] * dchui (n=dchui@202.169.29.34) has joined #atlassiandev
[11:19:36 EST(+1100)] <dchui> It is possible for atlasbot to update chat logs in minor edit mode? I'm watching the space and I'm getting a lot of notifications
[11:20:13 EST(+1100)] <mrdrone> ah right
[11:20:15 EST(+1100)] <mrdrone> good idea
[11:20:53 EST(+1100)] <tmoore> Sam: REST module doesn't export Jettison packages?
[11:21:06 EST(+1100)] <mrdrone> actually, how do you do that?
[11:21:19 EST(+1100)] <mrdrone> it interacts with confluence via that swizzle thing
[11:21:27 EST(+1100)] <mrdrone> don't see a way to set a change as minor
[11:22:22 EST(+1100)] <tmoore> what's the recommended way of dealing with JSON if I don't want to muck w/ JAXB? Take a String and parse it with a library of my choice?
[11:23:02 EST(+1100)] <mrdrone> I guess
[11:23:12 EST(+1100)] <mrdrone> jettison directly, jsonobject from json.org
[11:23:14 EST(+1100)] <mryall> mrdrone: <http://jira.atlassian.com/browse/CONF-5725>
[11:23:31 EST(+1100)] <mryall> (resolved in 2.10, but that issue has the parameters you need to use)
[11:23:33 EST(+1100)] <mrdrone> or there is a helper I found in the rest plugin that makes interacting w/ jaxb directly easier
[11:23:46 EST(+1100)] <sleberrigaud> yes there is a helper in REST
[11:23:49 EST(+1100)] <sleberrigaud> JIRA uses it
[11:23:59 EST(+1100)] <sleberrigaud> but it needs JAXB annotations etc.
[11:24:40 EST(+1100)] <mrdrone> mryall: ah hashtables, how I haven't missed thee
[11:24:48 EST(+1100)] <mryall> hehe
[11:24:51 EST(+1100)] <mryall> Apache XML-RPC FTW
[11:24:57 EST(+1100)] <mrdrone> indeed
[11:25:06 EST(+1100)] <mrdrone> one of the first apache projects I forked just to use proper collections
[11:25:18 EST(+1100)] <mryall> g
[11:25:30 EST(+1100)] <mrdrone> yeah, so this swizzle thing doesn't support that property I guess
[11:25:41 EST(+1100)] <mrdrone> it is code written in 2006
[11:26:38 EST(+1100)] <mryall> oh, it's a Java client
[11:26:39 EST(+1100)] <mryall> ?
[11:26:42 EST(+1100)] <mryall> what operations do you use?
[11:26:55 EST(+1100)] <mrdrone> nuts and no newer versions of sizzle
[11:27:11 EST(+1100)] <mrdrone> I guess swizzle is an api on top of the xmlrpc stuff
[11:27:13 EST(+1100)] <tmoore> sam: what I'd really like is to just be able to declare a parameter with an annotation like @FormParam, but parsed from a JSON entity instead
[11:27:45 EST(+1100)] <tmoore> or alternately, get the whole thing a Map
[11:27:57 EST(+1100)] <tmoore> (or array/list as the case may be)
[11:28:11 EST(+1100)] <mrdrone> goddamn it, I'd like to go one week w/o forking an open source library...
[11:28:20 EST(+1100)] <jedi> lol
[11:29:33 EST(+1100)] <sleberrigaud> tim: well if you use @FormParam you'd be vulnerable to XSRF wouldn't you? The way to that in REST is with a JAXB object that gets decoded for you.
[11:29:49 EST(+1100)] <sleberrigaud> mrdrone: do you actually go one day without forking an open source lib?

[11:30:54 EST(+1100)] <mrdon_> not as often as I'd like 😞
[11:31:01 EST(+1100)] <sleberrigaud> tim: what are you trying to achieve?
[11:31:01 EST(+1100)] <mrdon_> another reason I love github
[11:36:41 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[11:36:41 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myall on 2010-01-13 13:10:53 EST(+1100)
[11:37:19 EST(+1100)] <mrdon_> dchui: ok, should be fixed
[11:37:53 EST(+1100)] <dchui> thank you so much!
[11:40:11 EST(+1100)] <mrdon_> looks like it updated the page...did you get an email?
[11:41:00 EST(+1100)] <tmoore> sam: I have two resources here, both take JSON payloads
[11:41:23 EST(+1100)] <tmoore> one is just a single parameter {"color:"color1"}... seems silly to have to create a JAXB wrapper object just for that
[11:41:32 EST(+1100)] <tmoore> the other takes an array with arbitrary keys
[11:41:36 EST(+1100)] <tmoore> arbitrary
[11:41:45 EST(+1100)] <tmoore> sorry, I mean an object
[11:41:57 EST(+1100)] <sleberrigaud> so it is a map
[11:42:01 EST(+1100)] <tmoore> so like {"up_foo": "x", "up_bar", "y"} yes
[11:42:12 EST(+1100)] <tmoore> can I just get a Map injected? I didn't see any indication that would work
[11:42:38 EST(+1100)] <tmoore> so when I said like @FormParam, what I mean is not actually form params, but something that worked the same way for fields of a JSON object
[11:45:37 EST(+1100)] <tmoore> right and there isn't a way to define them within a plugin, is there?
[11:46:14 EST(+1100)] <mryall> mrdon_: does it buffer the messages before updating Confluence?
[11:46:36 EST(+1100)] <mryall> there are occasional perf problems with pages with hundreds of versions
[11:46:49 EST(+1100)] <sleberrigaud> also on the Map thing with JAXB there is this issue that has been raised:
<https://studio.atlassian.com/browse/REST-109>

[11:47:31 EST(+1100)] <mrdon_> mryall: yes, it should buffer two ways - max 64 messages and max 1 hour
[11:47:41 EST(+1100)] <tmoore> yeah that sounds good Sam
[11:48:05 EST(+1100)] <mrdon_> mryall: configurable, of course
[11:48:22 EST(+1100)] <tmoore> JAX-RS offers a lot of flexibility, but a lot of it isn't yet available through the REST plugin module type
[11:48:34 EST(+1100)] <tmoore> I know nobody is working on it full time, though
[11:48:39 EST(+1100)] <mryall> mrdon_: nice
[11:49:02 EST(+1100)] <mrdon_> I would worry about swapping out the json jaxb provider and how that would break existing apps

[11:49:05 EST(+1100)] <mryall> we may want to bump up the message count if the channel gets busy 😊
[11:49:46 EST(+1100)] <sleberrigaud> mrdon: how do you get AO to create the DB schema for you with your plugin?
[11:49:50 EST(+1100)] <tmoore> maybe I'm making a mistake by trying to avoid JAXB
[11:50:02 EST(+1100)] <mrdon_> sleberrigaud: just call ActiveObjects.migrate()
[11:50:11 EST(+1100)] <mrdon_> with your domain interfaces
[11:50:16 EST(+1100)] <mrdon_> s/domain/entity/
[11:50:48 EST(+1100)] <mrdon_> mryall: yep, easy to do
[11:58:48 EST(+1100)] <sleberrigaud> mrdon: is there some kind of lifecycle module I can use to do that?
[11:59:43 EST(+1100)] <mrdon_> hmm
[11:59:50 EST(+1100)] <mrdon_> you could just do it when you get injected with the object
[12:00:05 EST(+1100)] <mrdon_> or you could implement LifecycleAware and override onStart()
[12:00:19 EST(+1100)] <mrdon_> I would just do it in the constructor
[12:01:20 EST(+1100)] <sleberrigaud> mrdon: still getting notification about logs update to CAC BTW
[12:08:57 EST(+1100)] <mrdon_> nuts, looks like I will have to fork the library
[12:09:14 EST(+1100)] <mrdon_> was hoping it was just another property on the page object I could get around by subclassing
[12:09:23 EST(+1100)] <mrdon_> but it is a new options map, which the API doesn't support
[12:18:47 EST(+1100)] <mrdon_> mryall: love how this new options param isn't documented:
<http://confluence.atlassian.com/display/CONFDEV/Remote+API+Specification#RemoteAPISpecification-Pages>

[12:18:59 EST(+1100)] <mryall> yeah, want to fix it up? 😊
[12:19:01 EST(+1100)] <mrdon_> ok, so maybe love is a strong word... 😊
[12:19:33 EST(+1100)] <mrdon_> is it only on updatePage?
[12:19:36 EST(+1100)] <mrdon_> or storePage as well?
[12:20:05 EST(+1100)] <mryall> only on updatePage
[12:20:13 EST(+1100)] <mryall> you can't have a "minor edit" when you're creating a page
[12:20:15 EST(+1100)] <mryall> or a change comment
[12:20:19 EST(+1100)] <mrdon_> ta
[12:25:08 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[12:25:08 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myall on 2010-01-13 13:10:53 EST(+1100)
[12:25:28 EST(+1100)] <mrdon_> ok, maybe fixed....
[12:33:13 EST(+1100)] <mrdon_> and just for mryall, the docs are updated
[12:36:29 EST(+1100)] <mrdon_> heh, looks like the buffer flush timeout is in seconds, not minutes
[12:36:34 EST(+1100)] <mrdon_> so it flushes every minute
[12:36:35 EST(+1100)] <mrdon_> will fix that
[12:37:17 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[12:37:17 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myall on 2010-01-13 13:10:53 EST(+1100)
[12:38:43 EST(+1100)] <mrdon_> so, can anyone confirm notifications aren't sent anymore?
[12:48:03 EST(+1100)] <dchui> i stopped getting notifications since a few minutes ago
[12:49:59 EST(+1100)] <mrdon_> cool
[12:50:14 EST(+1100)] <mrdon_> that could be cause I changed the flush setting though...
[12:50:19 EST(+1100)] <mrdon_> I guess we'll know in an hour
[12:55:19 EST(+1100)] <jazzy> mrdon_: that atlasbot message, how does that get sent?
[12:56:20 EST(+1100)] <mrdon_> the notice?

[12:57:30 EST(+1100)] <mrdon_> via pircbot's sendNotice() method 😊
[13:34:00 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[13:34:00 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by myall on 2010-01-13 13:10:53 EST(+1100)

[14:21:00 EST(+1100)] <dchui> so i think it worked - i'm no longer getting chat log edit notifications
[14:26:12 EST(+1100)] <mrdon> great
[14:50:53 EST(+1100)] <mryall> would be nice if it remembered who it had sent the message to, by nick
[14:50:59 EST(+1100)] <mryall> and only sent the notice once per day 😊
[15:21:34 EST(+1100)] * pwyatt (n=pwyatt@59.167.164.33) has left #atlassiandev
[15:41:28 EST(+1100)] * sleberrigaud (n=sleberri@59.167.164.33) has joined #atlassiandev
[16:00:00 EST(+1100)] <mrdon> could do that...though isn't it just a notice in our server logs?
[16:00:42 EST(+1100)] <mrdon> also, I should point out I updated the conf docs, being the change I sought, so it's your turn 😊
[16:00:48 EST(+1100)] <mryall> hehe
[16:20:41 EST(+1100)] * nmuldoon1 (n=Nicholas@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:20:47 EST(+1100)] * Carlfish_ (n=cmiller@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:20:50 EST(+1100)] * mryall_ (n=mryall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:20:50 EST(+1100)] * ChanServ sets mode +o mryall_
[16:21:04 EST(+1100)] * sleberri_ (n=sleberri@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:30:42 EST(+1100)] * ckiehl1 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:32:44 EST(+1100)] * mrdon__ (n=mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:34:07 EST(+1100)] * Carlfish_ (n=cmiller@59.167.164.33) has joined #atlassiandev
[16:34:12 EST(+1100)] * ckiehl (n=Adium@59.167.164.33) has joined #atlassiandev
[16:34:13 EST(+1100)] * mryall__ (n=mryall@59.167.164.33) has joined #atlassiandev
[16:34:13 EST(+1100)] * nmuldoon2 (n=Nicholas@59.167.164.33) has joined #atlassiandev
[16:34:14 EST(+1100)] * ChanServ sets mode +o mryall_
[16:34:28 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[16:34:30 EST(+1100)] * sleberrigaud (n=sleberri@59.167.164.33) has joined #atlassiandev
[17:05:40 EST(+1100)] <jazzy> bye everyone
[17:06:21 EST(+1100)] * jazzy thinks you all need a more reliable internet connection
[17:06:51 EST(+1100)] * jedi agree
[17:07:19 EST(+1100)] <dchui> i think there was a net split
[17:07:58 EST(+1100)] <jazzy> a netsplit would show a netsplit message and everyone leaving at once
[17:08:22 EST(+1100)] <jazzy> this is all the clients connecting from the sydney office timing out
[17:08:49 EST(+1100)] <dchui> most of them are dups (those nicks ending with _)
[17:09:04 EST(+1100)] <dchui> for instance, mrdon__ has quit, but mrdon is still here
[17:09:13 EST(+1100)] <mrdon> yep
[17:09:16 EST(+1100)] <mrdon> gotta love the netsplits
[17:29:04 EST(+1100)] <mryall> jazzy: you're not next door at the moment?
[17:40:47 EST(+1100)] <jedi> jazzy and i are connected from other servers, using irssi in a screen session
[17:58:06 EST(+1100)] <jazzy> i'm connected from san jose, ca
[17:58:39 EST(+1100)] <jazzy> it means a wee bit of log on typing in chat, but you get used to it
[17:59:18 EST(+1100)] <jazzy> when i was 1337er i used to run irssi from my linksys wrt54g wireless router
[18:01:06 EST(+1100)] <jazzy> but i got tired of maintaining a machine that only had 4mb disk space for day to day irc use, and didn't like having to choose between having nmap, tcpdump, and qos tools installed, or having irssi
[18:01:12 EST(+1100)] <jazzy> i chose nmap, tcpdump and qos tools
[18:01:50 EST(+1100)] <jedi> i'm connected from a linode in fremont, ca
[18:02:15 EST(+1100)] <jedi> slightly better than a WRT54G but not by much lol
[18:02:27 EST(+1100)] <jazzy> lol
[18:02:36 EST(+1100)] <jazzy> mines actually a solaris zone
[18:03:15 EST(+1100)] <jazzy> seemed like a good idea at a time... chance to learn yet another unix, but admin it so rarely that i have to relearn everything every time i want to do something
[18:04:38 EST(+1100)] <jazzy> i installed a package management system on it, but i can never remember which one, where its installed, or what command i'm supposed to use to run it
[18:05:40 EST(+1100)] <jazzy> man... my typing as been bad today, typos all over the place, missing words etc
[18:05:54 EST(+1100)] <jazzy> that typo in "has" was not intentional!!
[18:07:33 EST(+1100)] <jedi> lol
[18:07:47 EST(+1100)] <jedi> my brain typos when reading so that i don't notice other people's typos
[18:08:00 EST(+1100)] <jedi> then i have to scan their text really carefully after they point it out to find it
[18:08:06 EST(+1100)] <jedi> it's like Where's Wally
[18:08:18 EST(+1100)] <jazzy> lol
[18:13:46 EST(+1100)] * Carlfish_ (n=cmiller@59.167.164.33) has joined #atlassiandev
[18:16:29 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[21:16:28 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[21:16:28 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green' set by mryall on 2010-01-13 13:10:53 EST(+1100)
[22:29:27 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev

atlassiandev_log-2010-01-20

[00:21:00 EST(+1100)] * bspeakmon_ (n=bspeakmo@dsl092-186-178.sfo1.dsl.speakeeasy.net) has joined #atlassiandev
[00:22:12 EST(+1100)] * bspeakmon_ (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[03:18:58 EST(+1100)] * jdoklovic (n=jdoklovi@66.187.202.110) has joined #atlassiandev
[03:19:25 EST(+1100)] <jdoklovic> anyone know what the partner discount is?
[04:37:52 EST(+1100)] * tmoore (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[04:37:52 EST(+1100)] * ChanServ sets mode +o tmoore
[04:44:50 EST(+1100)] <tmoore> is atlasbot supposed to be sending me a msg individually as well as to the channel?
[04:45:15 EST(+1100)] <tmoore> I guess it makes sense, when I join, but maybe should name the channel
[04:45:26 EST(+1100)] <tmoore> alternately...
[04:45:35 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green This channel is being logged with transcripts available at http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#039; set by tmoore on 2010-01-20 04:45:35 EST(+1100)
[04:45:39 EST(+1100)] <tmoore> oops
[04:45:52 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green | This channel is being logged with transcripts available at http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#039; set by tmoore on 2010-01-20 04:45:52 EST(+1100)

[05:01:25 EST(+1100)] <jdoklovic> PStudio is painful!
[05:03:25 EST(+1100)] <tmoore> slow you mean?
[05:04:57 EST(+1100)] <jdoklovic> yeah, really really slow. Also, sometime when i click links (i.e. issues link) from various pages, it takes me to a pre-4.0 JIRA login screen which doesn't work.
[05:05:22 EST(+1100)] <jdoklovic> if i simply hit the dashborad url manually, i'm still logged in
[05:12:33 EST(+1100)] <jdoklovic> @Tim need advice on a unit test.....
[05:13:52 EST(+1100)] <jdoklovic> i need to write a test for the ZipUtils class in AMPS.... the method under test takes a File (target zip file), a Folder (source dir) and a prefix (zip entry prefix)
[05:14:19 EST(+1100)] <jdoklovic> basically creates a zip containing the source folder with all paths prefixed.
[05:15:11 EST(+1100)] <jdoklovic> not sure the best way to test this.... i guess add a test dir in the test/resources folder and try to zip it to the build dir and then make assertions on the ZipEntry objects?
[05:18:41 EST(+1100)] <jdoklovic> i would just copy the tests for the unzip method provided by atlassian and reverse them, however.... there aren't any 😊

[05:19:21 EST(+1100)] * ChanServ sets mode +o bspeakmon
[05:19:54 EST(+1100)] <tmoore> Yeah, I think a test dir in test/resources is good, but I might zip it to a temp dir
[05:20:47 EST(+1100)] <tmoore> You don't really know what directory someone is going to run the tests from, and you don't want to clutter up the working dir too much
[05:21:20 EST(+1100)] <tmoore> if you use JUnit 4.7, it has a built-in utility for creating temp dirs and deleting them when the test finishes: <http://www.infoq.com/news/2009/07/junit-4.7-rules>
[05:22:48 EST(+1100)] <jdoklovic> hmmm. AMPS is now at junit 4.5 Do you think I'll break something if i just bump it up to 4.7 ?
[05:24:48 EST(+1100)] * jnolen (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:27:41 EST(+1100)] <jdoklovic> also have a maven problem that stops me from running the full test suite.... for some reason the bamboo-amps-plugin isn't seeing the bamboo.version prop set 2 parents up, and so it can't find "shitty-bamboo-plugin"
[05:36:31 EST(+1100)] <tmoore> is AMPS using junit or junit-dep?
[05:56:43 EST(+1100)] <tmoore> One issue is that they stopped deploying junit-dep to Maven for some reason :-/br/>[05:57:00 EST(+1100)] <tmoore> and the junit artifact can cause conflicts with Hamcrest
[05:59:12 EST(+1100)] <tmoore> not sure about the bamboo.version thing
[06:01:18 EST(+1100)] <jdoklovic>
<dependency><groupId>junit</groupId><artifactId>junit</artifactId><version>4.5</version><scope>test</scope></dependency>
[06:05:30 EST(+1100)] <tmoore> yeah I bet you can bump that to 4.7 😊

[06:05:50 EST(+1100)] <tmoore> anyway, you can always try it and see what happens 😊
[06:10:55 EST(+1100)] <jdoklovic> also, what's the best way to create a File with the path to test/resources ?
[06:18:01 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[06:49:23 EST(+1100)] <tmoore> hmm can you give it an InputStream or does it have to be a file?
[07:59:25 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[08:13:15 EST(+1100)] * bug (n=bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[08:14:01 EST(+1100)] <bug> thx tmoore for jogging my memory that this was up and running ...
[08:14:20 EST(+1100)] <bug> quick question: is there a mechanism to set a specific comment during a transition?
[08:14:57 EST(+1100)] <bug> i can go write a quick postfunction, but i was hoping that maybe the CreateCommentFunction might be able to take an argument in the workflow XML
[08:15:00 EST(+1100)] <bug> or something similar
[08:17:17 EST(+1100)] <bug> or possibly i'm just missing a built in postfunction?
[08:43:46 EST(+1100)] <tmoore> sorry, no idea bug
[08:43:53 EST(+1100)] <tmoore> maybe a jira dev will show up later
[08:44:56 EST(+1100)] * sleberrigaud (n=sleberri@59.167.164.33) has joined #atlassiandev
[08:45:41 EST(+1100)] <bug> tmoore, np
[08:45:42 EST(+1100)] <bug> thx
[09:10:11 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:27:41 EST(+1100)] <jdoklovic> @tim just decided to scrap the test/resources idea an now the tests create the expected source folder in the temp directory.
[09:28:34 EST(+1100)] <jdoklovic> it's more better that way
[09:56:59 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[09:59:01 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
[10:12:46 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[10:12:46 EST(+1100)] * Topic is Channel favourite colour: Lime Green | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#039;> set by tmoore on 2010-01-20 04:46:07 EST(+1100)
[10:25:34 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[11:10:41 EST(+1100)] * dchui (n=dchui@202.169.29.34) has joined #atlassiandev
[11:50:44 EST(+1100)] * bug (n=bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[11:53:03 EST(+1100)] * bug_ (n=bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev 😊

[12:16:47 EST(+1100)] <mryall> mrdon: haha, Confluence remote API specification "last edited by #atlasbot" 😊
[12:16:49 EST(+1100)] <mryall> <http://confluence.atlassian.com/display/CONFDEV/Remote+API+Specification>
[12:19:23 EST(+1100)] <mrdon> doh
[12:27:03 EST(+1100)] * bug (n=bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:34:43 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[16:24:59 EST(+1100)] * Carlfish_ (n=cmiller@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:25:12 EST(+1100)] * nmuldoon (n=Nicholas@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:25:15 EST(+1100)] * ckiehl1 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:25:15 EST(+1100)] * mryall_ (n=mryall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:25:16 EST(+1100)] * ChanServ sets mode +o mryall_ 😊
[16:25:22 EST(+1100)] * mrdon_ (n=mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:25:29 EST(+1100)] * sleberri_ (n=sleberri@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:46:48 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[16:46:52 EST(+1100)] * ckiehl (n=Adium@59.167.164.33) has joined #atlassiandev
[16:46:57 EST(+1100)] * Carlfish_ (n=cmiller@59.167.164.33) has joined #atlassiandev
[16:46:57 EST(+1100)] * mryall_ (n=mryall@59.167.164.33) has joined #atlassiandev
[16:46:58 EST(+1100)] * ChanServ sets mode +o mryall_ 😊
[16:47:10 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev

[16:53:56 EST(+1100)] * nmuldoon1 (n=Nicholas@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:53:59 EST(+1100)] * ckiehl2 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[16:54:21 EST(+1100)] * mrdon_ (n=mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:03:34 EST(+1100)] * Carlfish_ (n=cmiller@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:04:21 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[17:04:22 EST(+1100)] * ChanServ sets mode +o myrall
[19:08:49 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:49:54 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[20:53:00 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev

atlassiandev_log-2010-01-21

[23:30:26 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:30:26 EST(+1100)] * mrdon_ (n=dbrown@ppp121-45.static.internode.on.net) has joined #atlassiandev
[23:30:26 EST(+1100)] * ckiehl2 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:30:26 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[23:30:26 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[23:30:26 EST(+1100)] * jedi (i=mike@li55-224.members.linode.com) has joined #atlassiandev
[23:42:36 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:44:09 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[23:44:09 EST(+1100)] * mrdon_ (n=dbrown@ppp121-45.static.internode.on.net) has joined #atlassiandev
[23:44:09 EST(+1100)] * ckiehl2 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:44:09 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[23:44:09 EST(+1100)] * jedi (i=mike@li55-224.members.linode.com) has joined #atlassiandev
[23:51:08 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[23:51:08 EST(+1100)] * jedi (i=mike@li55-224.members.linode.com) has joined #atlassiandev
[23:51:08 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[23:51:08 EST(+1100)] * ckiehl2 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:51:08 EST(+1100)] * mrdon_ (n=dbrown@ppp121-45.static.internode.on.net) has joined #atlassiandev
[23:51:08 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:53:44 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:15:07 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:15:07 EST(+1100)] * jedi (i=mike@li55-224.members.linode.com) has joined #atlassiandev
[00:15:07 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[00:15:07 EST(+1100)] * ckiehl2 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:15:07 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[00:17:10 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:35:31 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[00:35:31 EST(+1100)] * ckiehl2 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:35:31 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[00:35:31 EST(+1100)] * jedi (i=mike@li55-224.members.linode.com) has joined #atlassiandev
[00:39:17 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[01:10:38 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:02:54 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:26:33 EST(+1100)] * jdoklovic (n=jdoklovi@66.187.202.110) has joined #atlassiandev
[02:29:09 EST(+1100)] <jdoklovic> anyone tried nimble OSGi? <http://osgi.dzone.com/articles/more-nimble-osgi>
[02:57:39 EST(+1100)] * bug_ (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[04:14:38 EST(+1100)] <jdoklovic> @Ben I'm trying to build AMPS so i can re-submit my home-zip patch....
[04:15:02 EST(+1100)] <jdoklovic> when i run install on maven-refapp-plugin, i get: [INFO] Missing:
[04:15:03 EST(+1100)] <jdoklovic> -----
[04:15:03 EST(+1100)] <jdoklovic> 1) com.atlassian.maven.plugins:maven-refapp-plugin:jar:3.1-m3-SNAPSHOT
[04:15:23 EST(+1100)] <jdoklovic> WTF?
[04:18:33 EST(+1100)] <jdoklovic> nevermind, had to disable the "shitty" profile
[05:01:33 EST(+1100)] <jdoklovic> i was told to open a new review for the amps patch i'm submitting.... here's what studio tells me: You don't have permission to create reviews in any project.
[05:18:10 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[05:31:12 EST(+1100)] <bspeakmon> life is just hard in studio-land
[05:32:06 EST(+1100)] * superemily (n=superemily@c-24-5-41-0.hsd1.ca.comcast.net) has joined #atlassiandev

[05:32:14 EST(+1100)] <superemily> sup homies. 
[05:32:20 EST(+1100)] <bspeakmon> emiry!
[05:33:18 EST(+1100)] <superemily> hai!
[05:44:55 EST(+1100)] * cmclaughlin (n=cmclaugh@adsl-76-192-49-189.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[09:04:37 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:11:36 EST(+1100)] * nmuldoon (n=Nicholas@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[09:13:02 EST(+1100)] * ckiehl (n=Adium@59.167.164.33) has joined #atlassiandev
[09:13:10 EST(+1100)] * nmuldoon1 (n=Nicholas@59.167.164.33) has joined #atlassiandev
[09:13:11 EST(+1100)] * myrall_ (n=myrall@59.167.164.33) has joined #atlassiandev
[09:13:11 EST(+1100)] * ChanServ sets mode +o myrall_ 
[09:24:29 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[09:27:04 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
[09:27:15 EST(+1100)] <Carlfish> Morning
[09:59:03 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev

[09:59:56 EST(+1100)] <superemily> morrrning 
[10:01:30 EST(+1100)] <Carlfish> Hey emily!
[10:05:25 EST(+1100)] <mrdon> tday
[10:05:26 EST(+1100)] <mrdon> gday even
[10:06:31 EST(+1100)] <superemily> tday is a gday.
[10:35:39 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev

[10:57:27 EST(+1100)] * myrall (n=matt@124-170-160-238.dyn.iinet.net.au) has joined #atlassiandev
[10:57:28 EST(+1100)] * ChanServ sets mode +o myrall
[10:58:52 EST(+1100)] * myrall (n=matt@124-170-160-238.dyn.iinet.net.au) has joined #atlassiandev
[10:58:53 EST(+1100)] * ChanServ sets mode +o myrall
[10:59:34 EST(+1100)] <myrall> hah.. created a new plugin with the SDK at the shell
[10:59:55 EST(+1100)] <myrall> then tried to open it up with my 'idea' shortcut, and IDEA just wasn't responding
[11:00:00 EST(+1100)] * jensschumacher_ (n=anonymou@59.167.164.33) has joined #atlassiandev
[11:00:05 EST(+1100)] <myrall> I killed IDEA, restarted it, same problem
[11:00:26 EST(+1100)] <myrall> then I had another look at the console – I'd built the new plugin and was launching idea on my work computer, not my laptop 😊
[11:00:40 EST(+1100)] <myrall> ssh FAIL
[11:17:10 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[11:23:26 EST(+1100)] * dchui (n=dchui@202.169.29.34) has joined #atlassiandev
[11:27:56 EST(+1100)] <myrall> holy moley... confluence-webapp-3.1.war is 105 MB 😞
[11:28:17 EST(+1100)] <Carlfish> We're thinking of selling it by weight.
[11:28:20 EST(+1100)] <myrall> did we put the internet in there, or something 😊
[11:28:47 EST(+1100)] <Carlfish> !bacon 😊
[11:29:02 EST(+1100)] <myrall> Atlassian is going to have to start paying for my bandwidth if we continue making downloads that big...
[11:29:27 EST(+1100)] <Carlfish> Matt: You're working from home?
[11:29:36 EST(+1100)] <myrall> yeah, see email 😊
[11:29:41 EST(+1100)] <Carlfish> I don't read email.
[11:29:47 EST(+1100)] <myrall> I know 😊
[11:30:18 EST(+1100)] <Carlfish> Just FYI, I stole the copy of Effective Java from your desk and lent it to Penny, because it was the simplest way to explain why inheritance is evil.
[11:30:31 EST(+1100)] <myrall> np
[11:30:37 EST(+1100)] <myrall> I borrowed it from Peggy
[11:30:47 EST(+1100)] <myrall> so I'll just have to let her know that it's moved on 😊
[11:31:22 EST(+1100)] <myrall> acc tests passing on import now?
[11:31:57 EST(+1100)] <Carlfish> Something weird's happening with page parent/child thingies. That's the one test left failing.
[11:32:03 EST(+1100)] <Carlfish> Once that's fixed, we're golden.
[11:32:43 EST(+1100)] <myrall> nice
[11:33:38 EST(+1100)] <Carlfish> Confluence. Now with 100% less ReverseDataBinder
[11:34:04 EST(+1100)] <myrall> so you're actually going to delete it? that would be sweet
[11:34:36 EST(+1100)] <Carlfish> Sadly no. I want to keep it around just in case there are some antediluvian backups the new code can't handle.
[11:34:52 EST(+1100)] <myrall> too bad
[11:34:59 EST(+1100)] <myrall> maybe we can nix it in 3.3
[11:35:05 EST(+1100)] <Carlfish> Fingers crossed.
[11:35:35 EST(+1100)] <myrall> I still have to merge my superuser perm changes over to trunk too
[11:35:41 EST(+1100)] <myrall> should try to get that into m3
[11:39:40 EST(+1100)] * superemily (n=superemi@c-24-5-41-0.hsd1.ca.comcast.net) has joined #atlassiandev
[11:44:53 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has joined #atlassiandev
[12:00:20 EST(+1100)] * nmuldoon (n=Nicholas@59.167.164.33) has joined #atlassiandev
[12:15:27 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[12:36:43 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[12:36:43 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has joined #atlassiandev
[12:36:43 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[13:40:29 EST(+1100)] <dezwart> I like cheese!
[13:43:22 EST(+1100)] <dezwart> mrdon: will atlasbot log private messages?
[13:43:25 EST(+1100)] <dezwart> to it
[13:43:28 EST(+1100)] <dezwart> obviously
[13:43:31 EST(+1100)] <mrdon> nope
[13:43:34 EST(+1100)] <dezwart> dang
[13:43:37 EST(+1100)] <mrdon> don't think so anyway
[13:43:48 EST(+1100)] <mrdon> I plan to use that for other things in the future
[13:43:57 EST(+1100)] <dezwart> factoids?
[13:44:01 EST(+1100)] <mrdon> like with the asfinfra bot, you can register your jira username/password
[13:44:05 EST(+1100)] <mrdon> then create issues in the channel
[13:44:10 EST(+1100)] <dezwart> that would be useful
[13:44:33 EST(+1100)] <dezwart> even being able to create an issue via PRIVMSG would be handy.
[13:44:33 EST(+1100)] <mrdon> infrabot will do commenting, updating, creation, and even searching
[13:44:37 EST(+1100)] <dezwart> For the lazy
[13:44:40 EST(+1100)] <mrdon> yep
[13:44:43 EST(+1100)] <mrdon> still need the pw though
[13:44:50 EST(+1100)] <mrdon> which is why the registration would be required
[13:45:04 EST(+1100)] <mrdon> unfortunately, this would be a security concern, which I don't see how to really mitigate
[13:45:13 EST(+1100)] <dezwart> Does freenode support SSL connections?
[13:45:30 EST(+1100)] <mrdon> don't think so, but there is also a concern in how it is stored
[13:45:42 EST(+1100)] <mrdon> cause if it goes on a contegix server, anyone can access it
[13:45:46 EST(+1100)] <mrdon> or anyone w/ sudo anyway
[13:45:51 EST(+1100)] <mrdon> which may not be that big of a deal
[13:46:00 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[13:46:07 EST(+1100)] <mrdon> we could base64 the file just to keep it away from casual or accidental viewing
[13:46:27 EST(+1100)] <dezwart> What about an authentication token
[13:46:33 EST(+1100)] <dezwart> that could be delivered by JIRA

[13:46:37 EST(+1100)] <dezwart> and then registered with the bot
[13:47:24 EST(+1100)] <mrdon> that would be a bit better
[13:47:27 EST(+1100)] <mrdon> but require jira changes
[13:49:19 EST(+1100)] * dezwart hmms to himself
[13:49:30 EST(+1100)] <dezwart> We have the technology, we will rebuilt him/her/it
[14:03:28 EST(+1100)] <jazzy> mrdon: why can't you make your bot support trusted apps?
[14:03:44 EST(+1100)] <mrdon> that did occur to me...
[14:03:53 EST(+1100)] <mrdon> but it seemed that might be even more of a security hole...
[14:04:02 EST(+1100)] <mrdon> I guess I could store mappings on the bot side
[14:04:11 EST(+1100)] <jazzy> yeah...
[14:04:13 EST(+1100)] <mrdon> then have a whitelist of users
[14:04:14 EST(+1100)] <jazzy> how about 3LO?
[14:04:21 EST(+1100)] <mrdon> no web interface
[14:04:26 EST(+1100)] <mrdon> so you'd only have 2lo
[14:04:31 EST(+1100)] <jazzy> give your bot a web interface
[14:05:03 EST(+1100)] <mrdon> but irc isn't a web interface
[14:05:19 EST(+1100)] <mrdon> I guess I could force you go through the web interface the first time to establish a token
[14:05:24 EST(+1100)] <mrdon> then use it from then on...
[14:05:26 EST(+1100)] <jazzy> so, you'd have to go to a web interface once, and from then on, it'd use that token
[14:05:31 EST(+1100)] <mrdon> jinks
[14:05:57 EST(+1100)] <jazzy> the other thing to worry about is whether the user is the first user that gave their credentials
[14:06:12 EST(+1100)] <jazzy> on freenode you could use NickServe to verify a users identity
[14:07:06 EST(+1100)] <jazzy> you'd probably wanna store when they registered their nick though, cos after 60 days of inactivity, a freenode admin may deregister the nick so someone else can use it
[14:08:16 EST(+1100)] <dezwart> jazzy: that is how i lost rasputin
[14:09:25 EST(+1100)] <jazzy> alternatively, why don't you simply make users sign in with your bot every session? the bot could then store the password in memory, and would also store when you connected, and would check how long you've been connected every time you make the call
[14:09:46 EST(+1100)] <jazzy> irssi users just have to configure their client to automatically send the message to your bot when they sign on
[14:16:12 EST(+1100)] <mrdon> true
[14:16:15 EST(+1100)] <mrdon> that would work
[14:16:30 EST(+1100)] <mrdon> that is a bit of a risk as what if atlasbot wasn't online
[14:16:31 EST(+1100)] <mrdon> but another was
[14:16:42 EST(+1100)] <mrdon> but probably the easiest idea so far
[15:04:50 EST(+1100)] * ryant_ (n=ryant@59.167.164.33) has joined #atlassiandev
[15:25:19 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[15:38:36 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[15:39:28 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[15:42:11 EST(+1100)] * sleberri_ (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[15:49:45 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[15:50:09 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[16:25:57 EST(+1100)] <Carlfish> Tests are green. Fish is happy.
[17:05:39 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[17:40:56 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[17:49:25 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:41:28 EST(+1100)] * myrall (n=matt@124-170-160-238.dyn.iinet.net.au) has joined #atlassiandev
[18:41:28 EST(+1100)] * ChanServ sets mode +o myrall
[19:34:23 EST(+1100)] * dchui (n=dchui@202.169.29.34) has joined #atlassiandev
[19:36:02 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[19:42:05 EST(+1100)] * kalamon (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[19:44:34 EST(+1100)] * kalamon (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[19:45:53 EST(+1100)] <kalamon> test
[19:51:15 EST(+1100)] * sleberri_ (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[19:52:53 EST(+1100)] * kalamon (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[19:53:11 EST(+1100)] * kalamon (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[19:54:10 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[19:55:02 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:27:52 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:28:58 EST(+1100)] * sleberri_ (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:39:55 EST(+1100)] * kalamon (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[21:06:23 EST(+1100)] * kalamon_ (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[21:42:24 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[22:13:36 EST(+1100)] * myrall (n=matt@124-170-160-238.dyn.iinet.net.au) has joined #atlassiandev
[22:13:36 EST(+1100)] * ChanServ sets mode +o myrall

atlassiandev_log-2010-01-22

[00:25:59 EST(+1100)] * jensschumacher (n=anonymou@203-158-61-66.dyn.iinet.net.au) has joined #atlassiandev
[00:30:52 EST(+1100)] * superemily (n=superemily@c-24-5-41-0.hsd1.ca.comcast.net) has joined #atlassiandev
[00:30:52 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:30:52 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[00:30:52 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has joined #atlassiandev
[00:32:34 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[00:32:34 EST(+1100)] * jedi (i=mike@li55-224.members.linode.com) has joined #atlassiandev
[00:34:23 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[00:34:23 EST(+1100)] * ckiehl (n=Adium@59.167.164.33) has joined #atlassiandev
[01:35:54 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[01:35:54 EST(+1100)] * ckiehl (n=Adium@59.167.164.33) has joined #atlassiandev
[01:35:54 EST(+1100)] * kalamon__ (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[01:35:54 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev

[01:35:54 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has joined #atlassiandev
[01:35:54 EST(+1100)] * superemily (n=superemi@c-24-5-41-0.hsd1.ca.comcast.net) has joined #atlassiandev
[01:35:54 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:50:00 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[01:50:00 EST(+1100)] * jedi (i=mike@li55-224.members.linode.com) has joined #atlassiandev
[03:17:09 EST(+1100)] * jdoklovic (n=jdokovi@66.187.202.110) has joined #atlassiandev
[05:12:38 EST(+1100)] * tmoore (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:12:39 EST(+1100)] * ChanServ sets mode +o tmoore
[05:40:08 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[06:59:17 EST(+1100)] * jnolen (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[07:15:26 EST(+1100)] * Carlfish (n=cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[07:21:32 EST(+1100)] <jdoklovic> can anyone help with a cargo-maven2 problem I'm having?
[07:22:01 EST(+1100)] <bspeakmon> don would be the guy to ask
[07:35:20 EST(+1100)] * sleberrigaud (n=sleberri@59.167.164.33) has joined #atlassiandev
[08:16:43 EST(+1100)] * mm_ (n=mm@chello089079130073.chello.pl) has joined #atlassiandev
[08:17:28 EST(+1100)] * kalamon_home (n=mm@chello089079130073.chello.pl) has joined #atlassiandev
[08:17:50 EST(+1100)] * kalamon_home (n=mm@chello089079130073.chello.pl) has joined #atlassiandev
[08:18:38 EST(+1100)] * kalamon_home (n=mm@chello089079130073.chello.pl) has joined #atlassiandev
[08:21:08 EST(+1100)] <jdoklovic> so i'm running my own container and added the atlassian-plugins and the atlassian-plugins-servlet. I create a simple servlet plugin and deployed it, but when it runs, I get: java.lang.NoClassDefFoundError:
javax/servlet/http/HttpServletRequest
[08:21:08 EST(+1100)] <jdoklovic> com.sysbliss.plugins.servlet.TestServlet doGet(TestServlet.java:20)
[08:21:19 EST(+1100)] <jdoklovic> anyone know what would cause this?
[08:21:45 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[08:21:57 EST(+1100)] <jdoklovic> am i missing a jar or something? some OSGI'ified version of servlet-api maybe?
[08:24:09 EST(+1100)] <sleberrigaud> jdoklovic: which application?
[08:24:18 EST(+1100)] <jdoklovic> my own
[08:24:50 EST(+1100)] <sleberrigaud> did you specify a list of packages to export from the host (bundle 0)?
[08:25:00 EST(+1100)] <jdoklovic> everything boots up fine and obviously the plugin framework is working cause my servlet is running, just getting the classdef error
[08:25:21 EST(+1100)] <jdoklovic> not sure where to specify package exports
[08:25:38 EST(+1100)] <sleberrigaud> if you set the log level to debug on com.atlassian.plugin you should be able to see what you're exporting
[08:28:12 EST(+1100)] <sleberrigaud> what does your javax.servlet import look like in the manifest of your plugin with a servlet?
[08:28:19 EST(+1100)] <sleberrigaud> it is set optional?
[08:30:42 EST(+1100)] <jdoklovic> 1 sec
[08:36:20 EST(+1100)] <sleberrigaud> you can specify packages to include and exclude from the host application exports using the com.atlassian.plugin.osgi.container.PackageScannerConfiguration, methods are getPackageIncludes and getPackageExcludes
[08:38:14 EST(+1100)] <jdoklovic> I have: <value>javax.*</value> in packageIncludes in my spring context and I see things like:
[FelixOsgiContainerManager] WIRE: 6.0 -> javax.servlet.http -> 0 in my logs
[08:39:11 EST(+1100)] <jdoklovic> I didn't explicitly import the javax.servlet package in my plugin.... didn't think I had to
[08:41:38 EST(+1100)] <sleberrigaud> what's bundle 6, is that your plugin?
[08:42:30 EST(+1100)] <sleberrigaud> because you're writing a servlet you will have to import javax.servlet* in your plugin
[08:47:04 EST(+1100)] <jdoklovic> so I know what's going on, but don't know how to fix it
[08:47:37 EST(+1100)] <jdoklovic> I had this (which I need) in my plugin:
<Import-Package>com.ibsys.kaching.plugins,*;resolution:=optional</Import-Package>
[08:48:21 EST(+1100)] <jdoklovic> with that in there, the transform does not add the javax.servlet to the imports for me. If I remove the custom instruction, javax.servlet gets added to the import list during transform time
[08:51:26 EST(+1100)] <jdoklovic> I guess I have to list all **required** packages and then *
[08:51:37 EST(+1100)] <jdoklovic> as optional
[08:55:54 EST(+1100)] <jdoklovic> yeah, this fixed the issue:
<Import-Package>com.ibsys.kaching.plugins,javax.servlet,javax.servlet.http,*;resolution:=optional</Import-Package>
[08:58:18 EST(+1100)] * bug_ (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:59:39 EST(+1100)] <sleberrigaud> looks good.
[08:59:55 EST(+1100)] <sleberrigaud> ideally all required packages should be listed, yes.
[09:00:58 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[09:03:13 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[09:03:48 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[09:14:00 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[09:36:38 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has left #atlassiandev
[11:11:27 EST(+1100)] <dezwart> hahahaha
[11:11:51 EST(+1100)] <dezwart> jazz: ask dcheney about the snowman

[11:25:22 EST(+1100)] <jazz> I don't see a snowman :
[11:25:32 EST(+1100)] <jazz> I only see ?XC, and the XC is highlighted
[11:25:42 EST(+1100)] <jazz> cos I didn't start screen with -U
[11:26:44 EST(+1100)] * jensschumacher_ (n=anonymou@59.167.164.33) has joined #atlassiandev
[11:33:31 EST(+1100)] * jazz (n=jazzy@208.64.63.145) has joined #atlassiandev
[12:43:15 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[13:02:54 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[13:39:19 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[13:53:01 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[13:54:54 EST(+1100)] * kalamon_ (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[14:43:50 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has joined #atlassiandev
[14:50:30 EST(+1100)] * jensschumacher (n=anonymou@59.167.164.33) has left #atlassiandev
[15:14:49 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[15:47:38 EST(+1100)] * awei (n=awei@59.167.164.33) has joined #atlassiandev
[16:19:39 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:32:59 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[17:43:45 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev

[18:38:37 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:08:13 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:12:58 EST(+1100)] * kalamon_work (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[20:37:14 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[21:03:44 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[21:10:28 EST(+1100)] * jensschumacher (n=anonymou@203-158-61-66.dyn.iinet.net.au) has joined #atlassiandev
[21:11:29 EST(+1100)] * jensschumacher (n=anonymou@203-158-61-66.dyn.iinet.net.au) has left #atlassiandev

atlassiandev_log-2010-01-23

[01:22:34 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[02:02:35 EST(+1100)] * jdokovic (n=jdokovi@66.187.202.110) has joined #atlassiandev
[02:45:04 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[03:39:01 EST(+1100)] * superemily (n=superemi@c-24-5-41-0.hsd1.ca.comcast.net) has joined #atlassiandev
[04:33:15 EST(+1100)] * tmoore (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[04:33:16 EST(+1100)] * ChanServ sets mode +o tmoore
[04:34:20 EST(+1100)] * jnolen (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[06:32:50 EST(+1100)] * bug (n=bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[06:50:18 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[06:56:50 EST(+1100)] * jnolen (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[08:30:22 EST(+1100)] * superemily (n=superemi@c-24-5-41-0.hsd1.ca.comcast.net) has joined #atlassiandev
[09:06:27 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[11:10:53 EST(+1100)] * jnolen (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:51:46 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[13:52:54 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[14:59:53 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[22:43:29 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev

atlassiandev_log-2010-01-24

[05:27:13 EST(+1100)] * superemily (n=superemi@c-24-5-41-0.hsd1.ca.comcast.net) has joined #atlassiandev
[07:42:47 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[08:25:16 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[09:12:09 EST(+1100)] * jnolen (n=Adium@c-98-210-193-114.hsd1.ca.comcast.net) has joined #atlassiandev
[09:12:18 EST(+1100)] * jnolen (n=Adium@c-98-210-193-114.hsd1.ca.comcast.net) has left #atlassiandev
[09:53:07 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[11:04:13 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[12:04:56 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[12:47:09 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[13:03:41 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[13:04:06 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[14:10:14 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[14:31:58 EST(+1100)] * jazzy (n=jazzy@four.entic.net) has joined #atlassiandev
[14:42:11 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[14:42:11 EST(+1100)] * jedi (i=mike@l155-224.members.linode.com) has joined #atlassiandev
[14:42:11 EST(+1100)] * chuck (n=charlie@yourwiki/staff/charlie) has joined #atlassiandev
[14:42:11 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[14:42:11 EST(+1100)] * ckiehl (n=Adium@59.167.164.33) has joined #atlassiandev
[15:40:22 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[16:03:47 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[16:44:08 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:54:13 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:24:55 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[18:44:25 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[19:16:54 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[21:37:34 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev

atlassiandev_log-2010-01-25

[04:44:23 EST(+1100)] * bug (n=bug@adsl-75-62-238-63.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[05:33:02 EST(+1100)] * bug (n=bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[06:34:01 EST(+1100)] * bug (n=bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[08:03:48 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[09:28:18 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[09:56:36 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[10:15:42 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has joined #atlassiandev
[10:30:34 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
[10:30:47 EST(+1100)] <Carlfish> jo
[10:31:44 EST(+1100)] <jazzy> i knew someone would ruin it
[10:32:39 EST(+1100)] <dezwart> ##### #####
[10:32:39 EST(+1100)] <dezwart> ##### #####
[10:32:40 EST(+1100)] <dezwart> ##### #####
[10:32:41 EST(+1100)] <dezwart> ##### #####
[10:32:42 EST(+1100)] <dezwart>

[10:32:44 EST(+1100)] <dezwart>

[10:32:47 EST(+1100)] <dezwart>

#####
[10:32:50 EST(+1100] <dezwart>
#####
[10:32:53 EST(+1100] <dezwart>
#####
[10:32:56 EST(+1100] <dezwart>
#####
[10:32:59 EST(+1100] <dezwart>
#####
[10:33:02 EST(+1100] <dezwart>
#####
[10:33:43 EST(+1100] * **dezwart** (n=pdzwart@59.167.164.33) has joined #atlassiandev
[10:34:05 EST(+1100] <Carlfish> Er...
[10:34:31 EST(+1100] <jazzy> its half full
[10:35:08 EST(+1100] <dezwart> I wonder if all of that was sent to the server befor disconnection.
[10:35:14 EST(+1100] <jedi> i can't help but feel curious to see the rest of it
[10:35:47 EST(+1100] <jazzy> well, it looks to me like a water tank, but its not full, so the server was wrong, no excess flood
[10:36:29 EST(+1100] <dezwart> It was 'Hi' via banner
[10:36:30 EST(+1100] <jazzy> dezwart: we got 12 lines
[10:38:49 EST(+1100] <Carlfish> Normally on IRC servers you're allowed a couple of messages with no delay, but if you post too many too fast they delay them so you're only able to send one every few seconds. Then if too many back up in the queue you get disconnected.
[10:39:34 EST(+1100] <dezwart> did we end up getting the connection limit lifted for our IP?
[10:40:20 EST(+1100] <Carlfish> Wasn't that austnet?
[10:41:02 EST(+1100] <Carlfish> We moved the wow channel here from austnet because of a connection limit, but I can't remember anything about one here.
[10:41:31 EST(+1100] <dezwart> It was about a week ago, no more connections could be made to freenode.

😊

[10:41:42 EST(+1100] <dezwart> Doesn't help that I use more than one 😊
[10:42:16 EST(+1100] <Carlfish> Ah, righto.
[10:58:06 EST(+1100] <dezwart> so, nobody has a +o line
[10:58:18 EST(+1100] <dezwart> Ahh wait
[10:58:29 EST(+1100] <dezwart> mrdon: you can grant yourself one in the case it is needed right?
[11:06:46 EST(+1100] * **bug** (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[11:19:41 EST(+1100] <dezwart> Hello bug
[11:56:34 EST(+1100] * **dchui** (n=dchui@202.169.29.34) has joined #atlassiandev
[13:06:05 EST(+1100] <dezwart> Good afternoon, gentlemen.
[13:13:10 EST(+1100] <Carlfish> Another visitor. Stay a while. Stay forever!
[13:13:38 EST(+1100] <dezwart> Stay awhile & listen.
[14:04:39 EST(+1100] * **jedi** does the Deckard Cain rap (<http://www.youtube.com/watch?v=1tDleRljcDc>)
[14:08:34 EST(+1100] <dezwart> hahahaha
[14:08:38 EST(+1100] <dezwart> Glad someone got it
[15:10:41 EST(+1100] <mrdon> someone need ops?
[15:45:18 EST(+1100] <Carlfish> I dunno. Might be a reasonable idea to op Atlassian staff with reg'd nicks.
[15:46:01 EST(+1100] <Carlfish> Not fussed, though. Freenode seems surprisingly free of the general range of IRC abuses.
[15:55:06 EST(+1100] <mrdon> damn it
[15:55:23 EST(+1100] <dezwart> ?
[15:55:35 EST(+1100] <mrdon> gave charles ops on #struts
[15:56:05 EST(+1100] <mrdon> interesting
[15:56:09 EST(+1100] <mrdon> only four founders a channel
[15:57:32 EST(+1100] <mrdon> well, you both now have ops
[15:57:39 EST(+1100] <mrdon> and charles made the last founder slot
[16:00:24 EST(+1100] * **ChanServ** sets mode +o dezwart
[16:00:28 EST(+1100] * **dezwart** sets mode -o dezwart
[16:00:43 EST(+1100] <dezwart> Merci beaucoup.
[16:01:37 EST(+1100] <dezwart> mrdon: I've heard that you have some published works.
[16:01:50 EST(+1100] <mrdon> yep
[16:03:02 EST(+1100] <dezwart> Maven related I suspect.
[16:03:12 EST(+1100] <mrdon> nope, struts 2 in action
[16:03:15 EST(+1100] <dezwart> ahhhh
[16:03:29 EST(+1100] <mrdon> and some academic papers you've never heard of
[16:04:06 EST(+1100] <dezwart> I suspect for the DoD?
[16:04:10 EST(+1100] <mrdon> yep
[17:39:58 EST(+1100] * **bug** (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:51:18 EST(+1100] * **kalamon** (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[19:12:23 EST(+1100] * **sleberrig** (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:20:03 EST(+1100] * **sleberrig** (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:52:22 EST(+1100] * **sebr** (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[20:54:12 EST(+1100] * **jensschumacher** (n=jensschu@203-158-61-66.dyn.iinet.net.au) has joined #atlassiandev
[21:12:16 EST(+1100] * **Carlfish** (n=cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[23:00:42 EST(+1100] * **ckiehl1** (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:00:53 EST(+1100] * **dezwart** (n=pdzwart@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:01:05 EST(+1100] * **mrdon_** (n=mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[23:04:03 EST(+1100] * **ckiehl** (n=Adium@59.167.164.33) has joined #atlassiandev
[23:04:31 EST(+1100] * **mrdon** (n=mrdon@59.167.164.33) has joined #atlassiandev
[23:05:28 EST(+1100] * **dezwart** (n=pdzwart@59.167.164.33) has joined #atlassiandev
[23:08:54 EST(+1100] * **sebr** (n=sruiz@amarok/developer/sebr) has joined #atlassiandev

atlassiandev_log-2010-01-26

[00:51:04 EST(+1100] * **bug** (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

[01:58:02 EST(+1100)] * jdokovic (n=jdokovi@66.187.202.72) has joined #atlassiandev
[05:21:23 EST(+1100)] * bspeakmon (n=bspeakmo@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:21:23 EST(+1100)] * ChanServ sets mode +o bspeakmon
[05:25:21 EST(+1100)] * tmoore (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:25:22 EST(+1100)] * ChanServ sets mode +o tmoore
[05:34:10 EST(+1100)] * jnolen (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[07:47:44 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[09:57:17 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[10:13:47 EST(+1100)] * sebr_ (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[10:16:13 EST(+1100)] * Carlfish (n=cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[10:36:14 EST(+1100)] * sebr_ (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[11:20:03 EST(+1100)] * dchui (n=dchui@202.169.29.34) has joined #atlassiandev
[11:23:42 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[11:39:53 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[14:45:56 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[15:00:45 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[15:51:59 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:18:26 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:25:29 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[21:53:35 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev

atlassiandev_log-2010-01-27

[01:29:38 EST(+1100)] * bug_ (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:37:55 EST(+1100)] * jdokovic (n=jdokovi@66.187.202.72) has joined #atlassiandev
[04:00:08 EST(+1100)] * jdokovic (n=jdokovi@66.187.202.72) has joined #atlassiandev
[04:56:44 EST(+1100)] * tmoore (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[04:56:44 EST(+1100)] * ChanServ sets mode +o tmoore
[05:09:52 EST(+1100)] * jnolen (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:26:06 EST(+1100)] * jnolen (n=Adium@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:28:47 EST(+1100)] * jnolen (n=jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[06:05:36 EST(+1100)] * bug_ (n=bug@69.38.223.210) has joined #atlassiandev
[06:31:18 EST(+1100)] * bug (n=bug@69.38.223.210) has joined #atlassiandev
[07:29:42 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[08:19:45 EST(+1100)] * sleberrigaud (n=sleberri@59.167.164.33) has joined #atlassiandev
[08:53:00 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[09:51:30 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[09:51:30 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#039;> set by tmoore on 2010-01-20 04:46:07 EST(+1100)
[09:55:06 EST(+1100)] <dezwart> thanks atlasbot; I'll keep http://www.austlii.edu.au/au/legis/nsw/consol_act/wsa2005245/ in mind
[10:01:20 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
[10:01:26 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
[10:07:22 EST(+1100)] * sebr- (n=sruiz@59.167.164.33) has joined #atlassiandev
[10:07:51 EST(+1100)] <jdokovic> is it possible to make an osgi plugin non-singleton? I have a servlet that looks up plugins via module descriptors and uses them to do a transform based on the request.
[10:08:43 EST(+1100)] <jdokovic> i would think i need unique instances of the plugin(s) to ensure i'm thread-safe, correct?
[10:08:58 EST(+1100)] <jazzy_> do you mean an osgi plugin component, or the plugin itself?
[10:09:45 EST(+1100)] <jdokovic> i'm adding a new module-type. I need instances of that module-type to be thread-safe within the context of a servlet request
[10:10:55 EST(+1100)] <jazzy_> that's up to your module descriptor, when you implement the getModule() method, you can return a new instance each time, or a cached instance
[10:12:33 EST(+1100)] <jdokovic> i'm using: return (VastTransformer)((AutowireCapablePlugin) plugin).autowire(getModuleClass());
[10:12:54 EST(+1100)] <jazzy_> that should return a new instance each time
[10:16:17 EST(+1100)] <jdokovic> are you sure?
[10:16:54 EST(+1100)] <jdokovic> if descriptor.init gets called when the plugin is loaded, and all I do is return plugin, isn't that the same instance?
[10:18:17 EST(+1100)] <jdokovic> or is it the autowire that's creating the new instance?
[10:33:25 EST(+1100)] <jazzy_> autowire is creating the new instance
[10:33:34 EST(+1100)] <jazzy_> the module descriptor itself is a singleton
[10:33:38 EST(+1100)] <jazzy_> oh... he left
[10:37:30 EST(+1100)] <tmoore> hey, good thing we're logging the channel, eh?
[10:38:51 EST(+1100)] <jazzy_> indeed
[10:54:56 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[11:01:41 EST(+1100)] * bug (n=bug@74.5.120.152) has joined #atlassiandev
[11:02:09 EST(+1100)] * sebr_ (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[11:05:21 EST(+1100)] * bug_ (n=bug@74.5.120.152) has joined #atlassiandev
[11:08:29 EST(+1100)] * bug (n=bug@74.5.120.152) has joined #atlassiandev
[11:11:34 EST(+1100)] * dchui (n=dchui@202.169.29.34) has joined #atlassiandev
[11:12:50 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[11:15:05 EST(+1100)] * bug_ (n=bug@74.5.120.152) has joined #atlassiandev
[11:17:31 EST(+1100)] * bug (n=bug@74.5.120.152) has joined #atlassiandev
[11:23:22 EST(+1100)] * bug (n=bug@74.5.120.152) has joined #atlassiandev
[11:25:04 EST(+1100)] * sebr_ (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[11:26:21 EST(+1100)] * sebr_ (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[11:29:33 EST(+1100)] * Carlfish_ (n=cmiller@59.167.164.33) has joined #atlassiandev
[11:34:20 EST(+1100)] * bug (n=bug@74.5.120.152) has joined #atlassiandev
[11:56:14 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[12:48:57 EST(+1100)] * kalamon (n=kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[14:10:04 EST(+1100)] * dezwart_ (n=pdzwart@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[14:10:13 EST(+1100)] * ckiehl1 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev

[14:10:28 EST(+1100)] * mrdon_ (n=mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[14:10:56 EST(+1100)] * ckiehl (n=Adium@59.167.164.33) has joined #atlassiandev
[14:11:22 EST(+1100)] * ckiehl1 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[14:11:24 EST(+1100)] * dezwart (n=pdzwart@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[15:00:04 EST(+1100)] * myrall (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[15:00:04 EST(+1100)] * ChanServ sets mode +o myrall
[15:01:56 EST(+1100)] * sleberrigaud (n=sleberri@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[15:06:51 EST(+1100)] * JustinK (n=justin@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[15:43:36 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[16:02:56 EST(+1100)] * bug (n=bug@63.161.118.42) has joined #atlassiandev
[16:07:55 EST(+1100)] * bug (n=bug@63.161.118.42) has joined #atlassiandev
[16:19:36 EST(+1100)] * bug (n=bug@63.161.118.42) has joined #atlassiandev
[18:15:42 EST(+1100)] * atlasbot (n=PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[18:15:42 EST(+1100)] * Topic is 'Channel favourite colour: Lime Green | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts'>; set by tmoore on 2010-01-20 04:46:07 EST(+1100)
[18:16:41 EST(+1100)] <dchui> nice - i like join/quit notifications
[18:20:37 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[19:04:13 EST(+1100)] * jmimi (n=mohammad@81.29.242.4) has joined #atlassiandev
[19:22:17 EST(+1100)] * skrebs (n=shannon@119.12.210.76) has joined #atlassiandev
[19:35:54 EST(+1100)] * me0wster (n=chatzill@202.169.29.34) has joined #atlassiandev
[19:42:54 EST(+1100)] * avdd (n=avdd@ppp121-44-60-105.lns20.syd6.internode.on.net) has joined #atlassiandev
[19:50:33 EST(+1100)] * avdd (n=avdd@ppp121-44-60-105.lns20.syd6.internode.on.net) has left #atlassiandev
[19:53:17 EST(+1100)] * azw (n=a@202.169.29.34) has joined #atlassiandev
[19:55:04 EST(+1100)] * ssmith (n=ssmith@ppp121-45-190-182.lns6.syd7.internode.on.net) has joined #atlassiandev
[20:59:00 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[21:02:49 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[21:26:45 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[22:04:59 EST(+1100)] * jonmort (n=jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[22:12:59 EST(+1100)] * maier (n=maier@cpe-065-190-158-038.nc.res.rr.com) has joined #atlassiandev
[22:58:17 EST(+1100)] * tgdavies (n=tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev

atlassiandev_log-2010-01-28

[23:20:06 EST(+1100)] * skeinei (n=stefan@dslb-084-057-170-068.pools.arcor-ip.net) has joined #atlassiandev
[23:42:03 EST(+1100)] * skeinei (n=stefan@dslb-084-057-170-068.pools.arcor-ip.net) has left #atlassiandev
[23:42:32 EST(+1100)] * skeinei (n=stefan@dslb-084-057-170-068.pools.arcor-ip.net) has joined #atlassiandev
[23:42:40 EST(+1100)] * skeinei (n=stefan@dslb-084-057-170-068.pools.arcor-ip.net) has left #atlassiandev
[23:43:43 EST(+1100)] * skeinei (n=stefan@dslb-084-057-170-068.pools.arcor-ip.net) has joined #atlassiandev
[00:05:35 EST(+1100)] * dezwart_ (n=pdzwart@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:05:44 EST(+1100)] * myrall_ (n=myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:05:44 EST(+1100)] * ChanServ sets mode +o myrall_
[00:05:48 EST(+1100)] * ckiehl1 (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:05:52 EST(+1100)] * mrdon_ (n=mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[00:07:24 EST(+1100)] * ckiehl2 (n=Adium@59.167.164.33) has joined #atlassiandev
[00:07:49 EST(+1100)] * myrall_ (n=myrall@59.167.164.33) has joined #atlassiandev
[00:07:50 EST(+1100)] * dezwart_ (n=pdzwart@59.167.164.33) has joined #atlassiandev
[00:07:52 EST(+1100)] * ChanServ sets mode +o myrall_
[00:17:44 EST(+1100)] * whalejy (n=whaley@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[00:17:57 EST(+1100)] <whaley> o/
[00:19:45 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlassiandev
[01:36:17 EST(+1100)] * joevano (n=joevano@bzflag/developer/JoeVano) has joined #atlassiandev
[01:56:00 EST(+1100)] * jdoklovic (n=jdoklovi@66.187.202.72) has joined #atlassiandev
[01:56:04 EST(+1100)] * sohail (n=sohail@unaffiliated/sohail) has joined #atlassiandev

[01:56:13 EST(+1100)] <sohail> hi 😊
[01:57:14 EST(+1100)] <jdoklovic> @jazzy.... thanks for the info yesterday.... saw your reply in the confluence logs
[02:02:15 EST(+1100)] * jhickman (n=user@24-217-158-0.dhcp.stls.mo.charter.com) has joined #atlassiandev
[02:24:10 EST(+1100)] * sohail1 (n=sohail@CPE001bfc8b793b-CM000a73a081a5.cpe.net.cable.rogers.com) has joined #atlassiandev
[04:29:13 EST(+1100)] * tmoore (n=tmoore@207.105.30.91) has joined #atlassiandev
[04:29:13 EST(+1100)] * ChanServ sets mode +o tmoore
[04:36:40 EST(+1100)] * tmoore (n=tmoore@207.105.30.91) has left #atlassiandev
[04:36:42 EST(+1100)] * tmoore (n=tmoore@207.105.30.91) has joined #atlassiandev
[04:36:42 EST(+1100)] * ChanServ sets mode +o tmoore
[05:21:13 EST(+1100)] * bowman (n=foo@195.46.44.113.dynamic.cablesurf.de) has joined #atlassiandev
[06:30:18 EST(+1100)] * azw (n=a@202.169.29.34) has joined #atlassiandev
[07:18:16 EST(+1100)] * sohail (n=sohail@CPE001bfc8b793b-CM000a73a081a5.cpe.net.cable.rogers.com) has joined #atlassiandev
[07:29:03 EST(+1100)] * skrebs (n=shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[07:50:40 EST(+1100)] * Jilliano1 (n=jdressle@69.11.216.98) has joined #atlassiandev
[07:54:54 EST(+1100)] * whalejy (n=whaley@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[08:07:07 EST(+1100)] * sleberrigaud (n=sleberri@59.167.164.33) has joined #atlassiandev
[08:07:39 EST(+1100)] * skrebs_ (n=shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[08:12:30 EST(+1100)] * tmoore (n=tmoore@207.105.30.91) has joined #atlassiandev
[08:12:30 EST(+1100)] * ChanServ sets mode +o tmoore
[08:26:12 EST(+1100)] * tgdavies (n=tomd@174.8.70.115.static.exetel.com.au) has joined #atlassiandev
[08:36:12 EST(+1100)] * tgdavies (n=tomd@174.8.70.115.static.exetel.com.au) has joined #atlassiandev
[08:37:18 EST(+1100)] * pleschev (n=pleschev@59.167.164.33) has joined #atlassiandev
[08:58:42 EST(+1100)] * tgdavies (n=tomd@59.167.164.33) has joined #atlassiandev
[09:00:23 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[09:04:53 EST(+1100)] <dhardiker> !8ball is there an 8ball yet?

[09:05:00 EST(+1100)] <dhardiker> ... I guess not 😕

[10:01:58 EST(+1100)] * **Carlfish** (n=cmiller@59.167.164.33) has joined #atlassiandev

[10:05:42 EST(+1100)] * **jдумай** (n=jдумай@59.167.164.33) has joined #atlassiandev

[10:08:18 EST(+1100)] * **jдумай** (n=jдумай@59.167.164.33) has joined #atlassiandev

[10:09:17 EST(+1100)] * **sebr** (n=sruiz@amarok/developer/sebr) has joined #atlassiandev

[10:11:26 EST(+1100)] * **skrebs_** (n=shannon@119.12.210.76) has joined #atlassiandev

[10:26:46 EST(+1100)] * **sebr_** (n=sruiz@amarok/developer/sebr) has joined #atlassiandev

[10:34:33 EST(+1100)] * **skrebs_** (n=shannon@121.91.0.225) has joined #atlassiandev

[10:36:44 EST(+1100)] <Carlfish> Dudes.

[10:37:17 EST(+1100)] <dezwart> wassup?

[10:37:23 EST(+1100)] <mrdon> g'day

[10:37:25 EST(+1100)] <ckiehl2> sweets

[10:37:53 EST(+1100)] <mrdon> I really expected this place to be bouncing around talking about how cool the ipad is

[10:38:10 EST(+1100)] <dezwart> the what?

[10:38:10 EST(+1100)] <mrdon> instead, I get nothing

[10:38:17 EST(+1100)] <mrdon> the new apple tablet

[10:38:22 EST(+1100)] <dezwart> woo

[10:38:34 EST(+1100)] <mrdon> basically iphone XL

[10:38:44 EST(+1100)] <dezwart> Are we going to support our products on it? 😊

[10:39:04 EST(+1100)] <mrdon> I think we are finally going in the opposite direction in that regard 😊

[10:39:11 EST(+1100)] <dezwart> Yes, thankfully.

[10:41:42 EST(+1100)] <tmoore> Apple released a tablet?

[10:41:52 EST(+1100)] <tmoore> I hadn't heard

[10:42:50 EST(+1100)] <dezwart> <http://www.apple.com/ipad/>

[10:43:44 EST(+1100)] <jazzy> ipad got a mention in #bacon

[10:43:56 EST(+1100)] <tmoore> does it come with a free unicorn?

[10:44:13 EST(+1100)] <dezwart> Yes, shrinkwrapped however.

[10:44:19 EST(+1100)] <jazzy> yes, but you have to go to www.cornify.com

[10:44:23 EST(+1100)] <dezwart> Flat-pack IKEA unicorn.

[10:44:51 EST(+1100)] <bsspeakmon> some assembly required.

[10:45:36 EST(+1100)] <jazzy> the twitterverse is bouncing with ipads, but the only place on IRC that i've found thats bouncing is the channel i hang out in on the ozorg network

[10:46:12 EST(+1100)] <tmoore> No GPS. Less space than an HP/Microsoft Slate. Lame

[10:46:42 EST(+1100)] <jazzy> it does have gps

[10:46:59 EST(+1100)] <tmoore> oh, then sign me up

[10:47:07 EST(+1100)] <tmoore> someone told me it didn't

[10:47:13 EST(+1100)] <jazzy> and a compass, in case you're lost in the desert and all you've got is an ipad

[10:47:15 EST(+1100)] <bsspeakmon> no multitasking

[10:47:25 EST(+1100)] <bsspeakmon> you only run one app at a time

[10:47:30 EST(+1100)] <Carlfish> I want one.

[10:47:34 EST(+1100)] <Carlfish> Then again that's a given.

[10:47:45 EST(+1100)] <skrebs_> usb?

[10:47:47 EST(+1100)] <jazzy> by multitasking, what they really mean is no window manager

[10:48:04 EST(+1100)] <tmoore> I've been in an "Enterprise OpenSocial meeting" all day. What I really need is to be hit by an iBrick on the iHead.

[10:48:08 EST(+1100)] <jazzy> i'm sure its more than capable of multitasking...

[10:48:15 EST(+1100)] <Carlfish> I assume they also mean the same thing as the iPhone. When you switch apps, the previous app stops running.

[10:48:45 EST(+1100)] <Carlfish> Unless you're one of the magical blessed Apple apps that's allowed to break the rules.

[10:48:53 EST(+1100)] * **sohail** (n=sohail@CPE001bfc8b793b-CM000a73a081a5.cpe.net.cable.rogers.com) has joined #atlassiandev

[10:49:08 EST(+1100)] <jazzy> i would have thought they could have put enough RAM in the ipad to support it

[10:49:26 EST(+1100)] <jazzy> or is it battery life they're worried about?

[10:49:31 EST(+1100)] <Carlfish> I think what most people are ignoring in the "it's just a big iPhone" commentary is that everyone who's tried to do regular general purpose computing on a tablet has failed to make a dent in the market because you just end up with a laptop with a bad form factor.

[10:49:44 EST(+1100)] <dezwart> Carlfish: +1

[10:49:58 EST(+1100)] <Carlfish> So maybe "It's a big handheld" is a better angle than "it's a keyboardless laptop"

[10:50:04 EST(+1100)] <Carlfish> Time will tell.

[10:50:17 EST(+1100)] <dezwart> It's a handheld for the BFG.

[10:50:27 EST(+1100)] * **sebr** (n=sruiz@amarok/developer/sebr) has joined #atlassiandev

[10:50:32 EST(+1100)] <Carlfish> I just hope Blizzard ports WoW to it. 😊

[10:51:22 EST(+1100)] * **dezwart** tries very hard not to say anything

[10:52:34 EST(+1100)] * **skrebs_** (n=shannon@119.12.210.76) has joined #atlassiandev

[10:52:56 EST(+1100)] <Carlfish> Aside from mass storage, it does pretty much everything my mother needs in a computer.

[10:54:15 EST(+1100)] <jazzy> does it run microsoft word?

[10:54:45 EST(+1100)] <Carlfish> Not running Word is a feature.

[10:55:06 EST(+1100)] <Carlfish> s/computer/home computer

[10:55:13 EST(+1100)] <jazzy> ...that will turn the majority of people off immediately

[10:59:35 EST(+1100)] * **dchui** (n=dchui@202.169.29.34) has joined #atlassiandev

[11:15:30 EST(+1100)] * **deadheart** (n=chatzill@59.167.164.33) has joined #atlassiandev

[11:16:10 EST(+1100)] <rackley> anyone here?

[11:16:21 EST(+1100)] <dezwart> no

[11:17:00 EST(+1100)] <dezwart> you should change your nick to lawson

[11:17:31 EST(+1100)] <lawson> how do I write a macro?

[11:18:20 EST(+1100)] <dezwart> Could you give us some more information?

[11:37:23 EST(+1100)] <Carlfish> Someone is trying to sell the Sydney poker mailing-list a russian bride.

[12:03:10 EST(+1100)] * **sebr** (n=sruiz@amarok/developer/sebr) has joined #atlassiandev

[12:55:07 EST(+1100)] * **rburton-** (n=rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev

[12:55:10 EST(+1100)] <rburton-> How's it going
[12:55:15 EST(+1100)] * rburton- looks for folks he knows
[12:55:26 EST(+1100)] <rburton-> hey mrdon, whaley
[12:55:30 EST(+1100)] <rburton-> justin_, stepka?
[12:55:37 EST(+1100)] <whaley> rburton-: o/
[12:55:41 EST(+1100)] <rburton-> 😊
[12:55:53 EST(+1100)] * rburton- running Bamboo at home for my company project
[12:55:53 EST(+1100)] <rburton-> 😊
[12:56:22 EST(+1100)] <whaley> outside of Confluence, I haven't used Atlassian stuff since the middle of '09 😕
[12:56:33 EST(+1100)] <rburton-> What?
[12:56:38 EST(+1100)] <whaley> one project I'm on has a confluence wiki at least
[12:56:40 EST(+1100)] <rburton-> Mike use to come online all the time on efnet
[12:56:43 EST(+1100)] <rburton-> back in the day
[12:57:14 EST(+1100)] <rburton-> Since this chat is being index. <http://SmartCodellc.com> rocks 😊
[12:58:55 EST(+1100)] <justin_> @rburton, sorry mate. I am an Australian version. I don't think stepka hangs out on irc.
[12:59:13 EST(+1100)] <rburton-> ah I got his IM anyhow and cell phone num
[12:59:17 EST(+1100)] <rburton-> thought you might have been him
[12:59:20 EST(+1100)] <rburton-> How's it going
[12:59:45 EST(+1100)] <justin_> fine, I think everyone is busy watching the keynote.
[12:59:58 EST(+1100)] <rburton-> which of Apple
[13:00:47 EST(+1100)] <justin_> I don't think there is any other kind 😊
[13:01:01 EST(+1100)] <rburton-> 😊 Well don't expect it to last long after Steve goes 😊
[13:01:07 EST(+1100)] <rburton-> I'm not impressed with the ipad
[13:01:31 EST(+1100)] <rburton-> sort of let down by it.
[13:11:33 EST(+1100)] * lawson_ (n=chatzill@59.167.164.33) has joined #atlasiandev
[13:32:08 EST(+1100)] <jazzy> good to see people are finding their way in here, in spite of the server in the devblog post being called "[#034;](http://freenode.net)
[13:47:03 EST(+1100)] <Carlfish> Haha
[13:57:43 EST(+1100)] * roberto_ (n=roberto@S010600090f525567.vc.shawcable.net) has joined #atlasiandev
[14:31:27 EST(+1100)] * jazzy wonders if raising an issue on stac for the fridge having no soft drink in it will go down well
[14:33:49 EST(+1100)] * rodogu (n=Adium@S010600090f525567.vc.shawcable.net) has joined #atlasiandev
[14:34:32 EST(+1100)] <rodogu> Got adium working 😊
[15:07:43 EST(+1100)] * skrebs_ (n=shannon@121.91.0.225) has joined #atlasiandev
[15:13:52 EST(+1100)] * Tarka (n=ssmith@59.167.164.33) has joined #atlasiandev
[15:26:50 EST(+1100)] <rburton-> jazzy, folks know IRC know how to get around 😊
[15:32:20 EST(+1100)] <jazzy> indeed they do... but those that don't might have difficulties
[16:15:02 EST(+1100)] * skrebs_ (n=shannon@121.91.0.225) has joined #atlasiandev
[16:19:33 EST(+1100)] * kalamon (n=kalamon@chello089074130182.chello.pl) has joined #atlasiandev
[16:37:36 EST(+1100)] * tmoore (n=tmoore@c-76-21-32-65.hsd1.ca.comcast.net) has joined #atlasiandev
[16:37:36 EST(+1100)] * ChanServ sets mode +o tmoore
[16:44:06 EST(+1100)] <jazzy> anyone seen eric?
[16:45:54 EST(+1100)] * pleschev_ (n=pleschev@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:45:55 EST(+1100)] * myrall_ (n=mryall@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:45:55 EST(+1100)] * ChanServ sets mode +o myrall_
[16:45:58 EST(+1100)] * ckiehl (n=Adium@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:46:02 EST(+1100)] * i386_workies (n=jdumay@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:46:04 EST(+1100)] * dezwart_ (n=pdzwart@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:46:18 EST(+1100)] * mrdon_ (n=mrdon@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:46:23 EST(+1100)] * tarka_ (n=ssmith@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:46:31 EST(+1100)] * lawson_ (n=chatzill@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:46:50 EST(+1100)] * tg davies_ (n=tomd@ATL146140-1.gw.connect.com.au) has joined #atlasiandev
[16:47:17 EST(+1100)] * i386 (n=jdumay@59.167.164.33) has joined #atlasiandev
[16:48:07 EST(+1100)] * tg davies_ (n=tomd@59.167.164.33) has joined #atlasiandev
[16:48:10 EST(+1100)] * Tarka (n=ssmith@59.167.164.33) has joined #atlasiandev
[16:48:14 EST(+1100)] * lawson_ (n=chatzill@59.167.164.33) has joined #atlasiandev
[16:48:16 EST(+1100)] * myrall_ (n=mryall@59.167.164.33) has joined #atlasiandev
[16:48:16 EST(+1100)] * ChanServ sets mode +o myrall_
[16:48:18 EST(+1100)] * dezwart_ (n=pdzwart@59.167.164.33) has joined #atlasiandev
[16:48:32 EST(+1100)] * mrdon (n=mrdon@59.167.164.33) has joined #atlasiandev
[16:48:33 EST(+1100)] * ckiehl1 (n=Adium@59.167.164.33) has joined #atlasiandev
[16:48:36 EST(+1100)] <dezwart> That is going to look good in the log.
[16:48:38 EST(+1100)] * pleschev (n=pleschev@59.167.164.33) has joined #atlasiandev
[16:57:05 EST(+1100)] <rburton-> Hmm
[16:57:14 EST(+1100)] <rburton-> There needs to be a better build system than maven.
[17:06:53 EST(+1100)] <jazzy> what are you trolling for? 😊
[17:16:18 EST(+1100)] <rburton-> 😊 I'm not just thinking
[17:16:22 EST(+1100)] <rburton-> I use it a lot
[17:45:08 EST(+1100)] <jazzy> maven tries to take too much control, what you need is the dependency and reactor mechanisms maven provides, coupled with an easy to learn/use DSL so unconventional things can be done simply
[17:47:04 EST(+1100)] <jazzy> its so hard to know whats happening because of that structure, what you need to see in your pom is "this happens then this happens then this happens"
[17:48:43 EST(+1100)] <jazzy> and all this needs to still be compatible with a convention over configuration method

```
[17:49:42 EST(+1100)] * dezwart removes the soap box from under jazzy
[17:49:57 EST(+1100)] * tgdavies (n=tomd@174.8.70.115.static.exetel.com.au) has joined #atlassiandev
[18:23:05 EST(+1100)] * tgdavies (n=tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[18:53:02 EST(+1100)] * mustafay (n=m@193.255.135.1) has joined #atlassiandev
[19:21:02 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[20:00:54 EST(+1100)] * jonmort (n=jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[21:40:41 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[22:28:43 EST(+1100)] * skeleinei (n=stefan@dslb-084-057-170-068.pools.arcor-ip.net) has joined #atlassiandev
[22:46:52 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
```

atlassiandev_log-2010-01-29

```
[00:56:17 EST(+1100)] * i386|laptop (n=jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[00:56:26 EST(+1100)] <i386|laptop> ho!
[00:58:54 EST(+1100)] * puskar (n=puskar@ool-45782058.dyn.optonline.net) has joined #atlassiandev
[00:59:42 EST(+1100)] <azw> ho!
[01:20:09 EST(+1100)] * bug (n=bug@1400hosta2.starwoodbroadband.com) has joined #atlassiandev
[01:32:54 EST(+1100)] * MartinCleaver (n=martincl@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[01:33:30 EST(+1100)] * sohail (n=sohail@76-10-161-135.dsl.teksavvy.com) has joined #atlassiandev
[02:17:50 EST(+1100)] * rodogu (n=Adium@d154-5-174-17.bchisia.telus.net) has joined #atlassiandev
[02:25:44 EST(+1100)] * bug (n=bug@1400hosta2.starwoodbroadband.com) has joined #atlassiandev
[02:30:04 EST(+1100)] * rodogu1 (n=Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[02:36:58 EST(+1100)] * sohail (n=sohail@76-10-161-135.dsl.teksavvy.com) has joined #atlassiandev
[02:37:34 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[02:55:08 EST(+1100)] * bug (n=bug@1400hosta2.starwoodbroadband.com) has joined #atlassiandev
[03:20:42 EST(+1100)] * Broady (n=b@unaffiliated/broady) has joined #atlassiandev
[03:20:56 EST(+1100)] <Broady> yo.
[03:23:58 EST(+1100)] * MartinCleaver waves from Toronto, Canada
[03:24:32 EST(+1100)] * Broady waves from Insomniaville, Sydney, Australia
[03:25:07 EST(+1100)] * MartinCleaver got up at 4:24am this morning, gave up on sleeping
[03:39:35 EST(+1100)] * cemerick (n=la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev

[03:39:51 EST(+1100)] <cemerick> Broady: hey, long time no chat 😊
[03:41:14 EST(+1100)] * jdoklovic (n=jdoklovi@66.187.202.72) has joined #atlassiandev
[03:41:51 EST(+1100)] <jdoklovic> does anyone know how to put a host component in the context of ALL osgi bundles?
[03:43:12 EST(+1100)] <jdoklovic> i can set plugin:available="true" on the bean, but i think that still requires plugins to do a component-import. I just want some host components to be available without a component-import
[04:02:09 EST(+1100)] * sohail (n=sohail@76-10-161-135.dsl.teksavvy.com) has joined #atlassiandev
[04:04:19 EST(+1100)] <jdoklovic> nevermind.... my plugin was trying to use a concrete class. switched it to the interface of the host component and it works fine.
[04:18:12 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[04:24:40 EST(+1100)] * puskar (n=puskar@ool-45782058.dyn.optonline.net) has joined #atlassiandev
[04:32:11 EST(+1100)] * sohail (n=sohail@unaffiliated/sohail) has left #atlassiandev
[04:39:37 EST(+1100)] * bug (n=bug@63.161.118.42) has joined #atlassiandev
[05:17:48 EST(+1100)] * tmoore (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:17:48 EST(+1100)] * ChanServ sets mode +o tmoore
[05:58:17 EST(+1100)] * puskar (n=puskar@ool-45782058.dyn.optonline.net) has joined #atlassiandev
[06:12:16 EST(+1100)] * jnolen_ (n=jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[06:12:57 EST(+1100)] * jnolen_ (n=jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[06:14:56 EST(+1100)] * jnolen_laptop (n=jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[06:49:11 EST(+1100)] * kalamon_home (n=mm@chello089079130073.chello.pl) has joined #atlassiandev
[06:51:39 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[06:55:56 EST(+1100)] * bug (n=bug@74.5.120.31) has joined #atlassiandev
[07:08:44 EST(+1100)] * AJC_Z0 (n=nAJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[07:13:22 EST(+1100)] * jnolen_ (n=jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[07:14:43 EST(+1100)] * bug (n=bug@74.5.120.31) has joined #atlassiandev
[07:18:00 EST(+1100)] * bug (n=bug@74.5.120.31) has joined #atlassiandev
[07:18:28 EST(+1100)] * bug (n=bug@74.5.120.31) has left #atlassiandev
[07:38:58 EST(+1100)] <dhardiker> word to the wise ... dont create a servlet filter plugin module with a constructor injected component that's not installed yet - you need to remove the plugin from the db and restart to recover
[07:39:48 EST(+1100)] <dhardiker> the plugin subsystem should probably just log the fatal death of the filter and disable the module/plugin ...
at least then it can be uninstalled 😊
[07:40:03 EST(+1100)] * skrebs (n=shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[07:40:28 EST(+1100)] <dhardiker> the servlet filter needs to catch /* or another pattern that include the install/uninstall plugin actions
[07:46:41 EST(+1100)] <rodogu1> Hi there! I am having a problem with a hugh thread when invoking one of my confluence plugin's action... I did a thread dump and identified the hung thread before even getting into my plugin's code
[07:46:45 EST(+1100)] <rodogu1> Thread[http-8087-15,5,main]
[07:46:45 EST(+1100)] <rodogu1> java.util.HashMap.get(HashMap.java:329)
[07:46:45 EST(+1100)] <rodogu1> com.opensymphony.xwork.util.OgnlUtil.compile(OgnlUtil.java:192)
[07:46:45 EST(+1100)] <rodogu1> com.opensymphony.xwork.util.OgnlValueStack.findValue(OgnlValueStack.java:141)
[07:46:45 EST(+1100)] <rodogu1>
com.opensymphony.webwork.views.velocity.WebWorkVelocityContext.internalGet(WebWorkVelocityContext.java:72)
[07:46:46 EST(+1100)] <rodogu1> org.apache.velocity.context.AbstractContext.get(AbstractContext.java:193)
[07:46:48 EST(+1100)] <rodogu1> org.apache.velocity.context.InternalContextAdapterImpl.get(InternalContextAdapterImpl.java:286)
[07:46:50 EST(+1100)] <rodogu1> org.apache.velocity.runtime.directive.Foreach.render(Foreach.java:330)
[07:46:52 EST(+1100)] <rodogu1> org.apache.velocity.runtime.parser.node.ASTDDirective.render(ASTDirective.java:175)
[07:46:54 EST(+1100)] <rodogu1> org.apache.velocity.runtime.parser.node.SimpleNode.render(SimpleNode.java:336)
[07:46:56 EST(+1100)] <rodogu1> any ideas?
[07:46:58 EST(+1100)] <rodogu1> s/hugh/hung/
```

[07:47:49 EST(+1100)] <dhardiker> is it reproducible?

[07:49:23 EST(+1100)] <rodogu1> yep, it happens all the time on any page... but in a production server 😕 I could add some debugging info but I am afraid the bug would go away...

[07:49:37 EST(+1100)] <rodogu1> ... cannot reproduce in other server 😕

[07:49:59 EST(+1100)] <dhardiker> does it still happen after an appserver restart?

[07:50:27 EST(+1100)] <rodogu1> heh heh... I could try that... but what if it just goes away?

[07:50:44 EST(+1100)] <dhardiker> Im not sure you can get any additional information without a debugger

[07:50:44 EST(+1100)] <dhardiker> if so, can you attach a debugger to log out the problem

[07:51:11 EST(+1100)] <dhardiker> have you done multiple thread dumps about 30 seconds apart?

[07:51:27 EST(+1100)] <dhardiker> or run a CPU profile?

[07:51:27 EST(+1100)] <rodogu1> now, what would my debug could do? as I've said, my code doesn't even gets invoked (it doesn't looks like, from the thread dump)

[07:51:57 EST(+1100)] <dhardiker> is the CPU at 100%?

[07:51:57 EST(+1100)] <rodogu1> I've run the dump several times and the thread remains there

[07:52:03 EST(+1100)] * rodogu1 checks

[07:52:18 EST(+1100)] <dhardiker> (it would be if it was spinning over a hashmap forever)

[07:52:22 EST(+1100)] <tmoore> that thread dump doesn't look like a deadlock, it looks like it could be slow code or an infinite loop

[07:52:40 EST(+1100)] <dhardiker> my bet would be on an infinite loop

[07:53:20 EST(+1100)] <dhardiker> it would have to be a hell of a map / ognl expression parsing to take minutes

[07:53:43 EST(+1100)] <dhardiker> in the multiple thread dumps, is it staying within HashMap.get() at the top?

[07:53:47 EST(+1100)] <dhardiker> and if so, the same line?

[07:53:52 EST(+1100)] <rodogu1> heh heh... can't tell, the sever on which is running has 32 cpus 😊

[07:54:40 EST(+1100)] <rodogu1> yep, same line;

[07:54:41 EST(+1100)] <rodogu1> java.util.HashMap.get(HashMap.java:329)

[07:54:41 EST(+1100)] <rodogu1> com.opensymphony.xwork.util.OgnlUtil.compile(OgnlUtil.java:192)

[07:54:41 EST(+1100)] <rodogu1> com.opensymphony.xwork.util.OgnlValueStack.findValue(OgnlValueStack.java:141)

[07:54:41 EST(+1100)] <rodogu1> com.opensymphony.webwork.views.velocity.WebWorkVelocityContext.internalGet(WebWorkVelocityContext.java:72)

[07:55:11 EST(+1100)] <rodogu1> is HashMap thread-safe?

[07:55:16 EST(+1100)] <dhardiker> no

[07:55:21 EST(+1100)] <dhardiker> ConcurrentHashMap is

[07:55:36 EST(+1100)] <rodogu1> The confluence server itself is running ok... (not slow)

[07:55:36 EST(+1100)] <dhardiker> HashMap isn't synchronized

[07:55:45 EST(+1100)] <tmoore> this line I think is worth noticing:

[07:55:46 EST(+1100)] <tmoore> org.apache.velocity.runtime.directive.Foreach.render(Foreach.java:330)

[07:55:46 EST(+1100)] <tmoore> 12:48

[07:55:49 EST(+1100)] <tmoore> what is it iterating over?

[07:55:54 EST(+1100)] <rodogu1> it is hanging when compiling a vm template I guess

[07:56:55 EST(+1100)] <rodogu1> good point...

[07:57:05 EST(+1100)] * rodogu1 checks the .vm file

[07:58:07 EST(+1100)] <dhardiker> it's possible you have a corrupted list in your HashMap, but it's also possible that the state in your application is causing an endless loop (or a very large amount of work)

[07:58:34 EST(+1100)] <dhardiker> in the latter, a restart of Confluence should cause the latter problem to persist

[07:59:03 EST(+1100)] <dhardiker> if it's a memory-state problem then the restart should clear it, at least until it reoccurs

[07:59:22 EST(+1100)] <rodogu1> I do have two nested #foreach loops... I do not believe the lists are too long, but will look into it...

[08:00:29 EST(+1100)] <dhardiker> do you have a test system with a replica of the data/state?

[08:00:31 EST(+1100)] <rodogu1> @tim, @dan do you know if that portion is actually when rendering the page or compiling it?

[08:00:39 EST(+1100)] * tgdavies (n=tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev

[08:01:09 EST(+1100)] * rodogu1 doesn't make to make any assumption with com.opensymphony.xwork.util.OgnlUtil.compile 😊

[08:01:33 EST(+1100)] <dhardiker> that's the OGNL expression compiler

[08:01:46 EST(+1100)] <dhardiker> so \${req.contextPath} would run through that when parsing the page

[08:03:12 EST(+1100)] <rodogu1> OK, it makes sense now: my plugin's code is being executed all the way, the actions response is what invokes the .vm file, that's why I don't see my code in the thread stack trace... which doesn't mean I'm off the hook 😊

[08:03:42 EST(+1100)] <dhardiker> you're action is still on the OGNL value stack though (as "action" inventively ... it's also the root)

[08:03:58 EST(+1100)] <dhardiker> so call backs into your action methods from the vm will still run through your code 😊

[08:04:09 EST(+1100)] <dhardiker> and you can still tell velocity to tie itself up in infinite loops 😊

[08:04:41 EST(+1100)] * rodogu1 goes back to have a closer look to his .vm file and data model

[08:04:48 EST(+1100)] <rodogu1> thanx guys!

[08:06:43 EST(+1100)] <dhardiker> if you're a high level of concurrency through a HashMap without any locking, I'd check that a restart (or otherwise reinitialise the backing of the Map) doesn't just clear the error

[08:07:01 EST(+1100)] <rodogu1> heh heh, at least the issue is not recursive 😊 so far, no stack overflow

[08:29:41 EST(+1100)] * bug (n=bug@74.5.120.31) has joined #atlassiandev

[08:33:59 EST(+1100)] * bug (n=bug@74.5.120.31) has joined #atlassiandev

[09:05:19 EST(+1100)] * Topic is Apple juice, for HALF PRICE | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#039;> set by bspeakmon on 2010-01-29 09:05:19 EST(+1100)

[09:07:52 EST(+1100)] <rodogu1> Ok, this is weird. If I have a plain #foreach look to traverse a list, it hangs:

[09:07:54 EST(+1100)] <rodogu1> #foreach (\$activityReportEntry in \$activityReportEntries)

[09:07:54 EST(+1100)] <rodogu1> stuff

[09:07:54 EST(+1100)] <rodogu1> #end

[09:08:03 EST(+1100)] <rodogu1> but I have replace it to gets, it works:

[09:08:12 EST(+1100)] <rodogu1> #set (\$account = \$activityReportEntries.size() - 1)

[09:08:12 EST(+1100)] <rodogu1> #foreach (\$ix in [0..\$account])

[09:08:12 EST(+1100)] <rodogu1> #set (\$activityReportEntry = \$activityReportEntries.get(\$ix))

[09:08:12 EST(+1100)] <rodogu1> stuff

[09:08:12 EST(+1100)] <rodogu1> #end

[09:09:11 EST(+1100)] <rodogu1> I cannot reproduce in a dev environment (same confluence version and data)... right before finishing my action, I do traverse the list, and there is no problem
[09:12:47 EST(+1100)] <tmoore> i wonder if something is adding to the list on each iter?
[09:13:07 EST(+1100)] <tmoore> hmm that would probably throw an exception though
[09:18:19 EST(+1100)] <rodogu1> nop... it a plain getting
[09:18:30 EST(+1100)] <rodogu1> s/it/it's/
[09:19:22 EST(+1100)] <rodogu1> Not sure if it is an infinite loop OR it is just hanging
[09:19:50 EST(+1100)] * sebr (n=seb@amarok/developer/sebr) has joined #atlassiandev
[09:21:21 EST(+1100)] * rodogu1 tries to find out if the loop is actually happening
[09:23:17 EST(+1100)] <tmoore> how much more useful would java stack traces be if it showed the value of params in each frame?
[09:23:22 EST(+1100)] <tmoore> much more
[09:27:21 EST(+1100)] <rodogu1> uh?
[09:27:57 EST(+1100)] * Carlfish (n=cmiller@59.167.164.33) has joined #atlassiandev
[09:28:19 EST(+1100)] <Carlfish> Hey hey
[09:28:34 EST(+1100)] <rodogu1> The loop never happens... it seems to hang before: in the loop I am accessing a fake variable, then I am logging it (in my action's getter) but it is not invoked
[09:29:35 EST(+1100)] <tmoore> oh yeah, how about that
[09:29:49 EST(+1100)] <rodogu1> ... but I use my hacked #foreach , it get's invoked :/-
[09:29:53 EST(+1100)] <tmoore> do you have an existing item in the context called activityReportEntry?
[09:30:14 EST(+1100)] <tmoore> looks like the line it's on is trying to save any existing value
[09:30:51 EST(+1100)] <tmoore> maybe try changing the name to something else & see what happens
[09:31:14 EST(+1100)] * rodogu1 checks
[09:31:49 EST(+1100)] <rodogu1> no, but will try renaming just for the heck of it...
[09:32:37 EST(+1100)] * rodogu1 renames to 'activityReportEntryCheese'
[09:35:00 EST(+1100)] <rodogu1> bingo?
[09:36:51 EST(+1100)] <Carlfish> I like cheese.
[09:37:20 EST(+1100)] <bspeakmon> this is an apple juice channel

[09:39:01 EST(+1100)] <rodogu1> unfortunately the cheese factor didn't work this time... problem remains even after renaming variable 😕
[09:42:11 EST(+1100)] <Carlfish> I didn't see the code, but it's not one of those annoying "Velocity can't handle null" problems?
[09:42:50 EST(+1100)] <Broady> Carlfish: evidently that annoying velocity 'feature' can be fixed with a config option
[09:43:12 EST(+1100)] <Broady> but the world would probably explode
[09:43:13 EST(+1100)] <Carlfish> Not in the version of velocity we started with.
[09:43:15 EST(+1100)] <Carlfish> Yeah.
[09:43:38 EST(+1100)] <Carlfish> We'd have to go through an awful lot of code that assumes the original behaviour.
[09:45:09 EST(+1100)] <Carlfish> Remember, we only upgraded Velocity pretty recently for Oysta's anti-xss stuff.

[09:45:11 EST(+1100)] <rodogu1> none of the entries in the list is null 😕
[09:50:12 EST(+1100)] <Carlfish> Hmm. Then I can only suggest reversing the polarity.
[09:50:38 EST(+1100)] <rodogu1> nice one

[09:51:38 EST(+1100)] <rodogu1> server is foobar with too many stalled threads 😕 the perils of debugging on a production server ... restarting
[09:55:54 EST(+1100)] * rodogu1 wonders if he should rename variable to activityReportEntryAppleJuice
[09:56:18 EST(+1100)] <bspeakmon> only if you want it to work
[09:57:23 EST(+1100)] <Broady> cheesyPantsActivityReportFactory
[10:04:02 EST(+1100)] <AJC_Z0> Where's the right place to ask about RPMs for Jira, Confluence, etc.? My (limited) research so far has turned up nothing significant
[10:05:30 EST(+1100)] * skrebs_ (n=shannon@119.12.210.76) has joined #atlassiandev
[10:08:11 EST(+1100)] <rodogu1> RPMs? I guess here's the right location, but AFAIK, there are not RPMs
[10:09:15 EST(+1100)] <AJC_Z0> Thanks. It's tricky to search, given the necessity to include "jira" or "confluence" in the search terms
[10:10:33 EST(+1100)] <Carlfish> There are no rpms that I know of. One of our support engineers made some deba a while back, but that's about it.
[10:11:20 EST(+1100)] <bspeakmon> it's unlikely there will ever be supported RPMs

[10:12:36 EST(+1100)] <mrdon> however, we certainly take donations 😊
[10:12:38 EST(+1100)] <AJC_Z0> Is supporting packaging formats for supported platforms really that much effort, given the many advantages to both provider and user?
[10:12:58 EST(+1100)] * sieberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[10:13:08 EST(+1100)] <mrdon> AJC_Z0: getting the product installed: easy...handling upgrades with who knows what database, who knows what app server: hard
[10:13:10 EST(+1100)] <rodogu1> everything is working after server restart... no need for Cheese or AppleJuice suffix nor cheesyPants prefix... My guess is somehow the velocity engine got foobar :/-
[10:13:12 EST(+1100)] <AJC_Z0> That sounded a lot more argumentative than intended
[10:13:20 EST(+1100)] <bspeakmon> heh
[10:13:43 EST(+1100)] <bspeakmon> it actually is trickier than it sounds
[10:14:15 EST(+1100)] <mrdon> one of the big problems is how many of the files in the webapp need to be modified by hand
[10:14:21 EST(+1100)] <mrdon> making upgrades rather tricky
[10:14:45 EST(+1100)] <mrdon> *nix apps separate config from binaries
[10:14:51 EST(+1100)] <mrdon> we throw it all into one big ball of wax
[10:15:39 EST(+1100)] <mryall> well, that's the way Java webapps are designed: you have automatic and easy access to files contained within the webapp
[10:15:48 EST(+1100)] <mryall> other files out on the file system somewhere, not necessarily
[10:15:58 EST(+1100)] <AJC_Z0> I'm painfully familiar with the difficulties of packaging (though less so with RPM than others), so I don't doubt it
[10:16:41 EST(+1100)] <bspeakmon> the real kicker, though, is that there is very little demand for it from our users
[10:17:20 EST(+1100)] <mrdon> mryall: though I would ammend that to the way "poor java web apps are designed"
[10:17:23 EST(+1100)] <AJC_Z0> but even Red Hat can't be bothered to try to help admins merge changed files in their own OS (i.e. .rpmnew)
[10:17:24 EST(+1100)] <mryall> bspeakmon: I think it's hard to say that. Certainly upgrade issues are high on the list of problems for many customers.

[10:17:45 EST(+1100)] <bspeakmon> sure, but wrapping it in a packager just adds another layer of issues
[10:17:53 EST(+1100)] <mrdon> now that said, as I've said before, our webapps shouldn't be wars, they should be full applications that include the app server
[10:17:53 EST(+1100)] <bspeakmon> because java web stuff is "special" 

[10:17:57 EST(+1100)] <mrdon> but that's a rant for another day
[10:18:21 EST(+1100)] <mryall> mrdon: it's really about the servlet APIs, that make accessing the web app easy, and an attempt by JEE administrators to sandbox the apps
[10:18:26 EST(+1100)] <Carlfish> mrdon: That's probably something the Atlassian of today is in a better position to pull off than the Atlassian of five years ago.
[10:18:51 EST(+1100)] <mryall> s/the web app/the web app files/
[10:19:11 EST(+1100)] <mryall> I don't think sandboxing is bad, per se
[10:19:23 EST(+1100)] <mrdon> mryall: no, but you could do that just as well with a bundled app server
[10:19:26 EST(+1100)] <mrdon> perhaps better, in fact
[10:19:34 EST(+1100)] <mryall> having unix apps write and read files all over the disk can often be a real PITA
[10:19:46 EST(+1100)] <mrdon> as now, our apps depend on all sorts of weird environment variables that make debugging and upgrading a PITA
[10:20:03 EST(+1100)] <jazzy> its great for backups
[10:20:09 EST(+1100)] <mrdon> well, if certain patterns are followed, it can be really awesome
[10:20:17 EST(+1100)] <mrdon> the problem is when apps want to invent their own patterns
[10:20:18 EST(+1100)] <mryall> bundling the app server doesn't fix the database though
[10:20:31 EST(+1100)] <mrdon> no, unfortunately it doesn't
[10:20:37 EST(+1100)] <mryall> we've had standalone for ages - it doesn't automatically upgrade itself
[10:20:48 EST(+1100)] <AJC_Z0> bear in mind it doesn't have to one fits-all-sizes package for any given "platform", so jira-4.0.1-mysql-sunJDK16-blue-flibble.rpm would be fine
[10:20:49 EST(+1100)] <jazzy> to backup the data of a computer, just backup /var, and you can be confident that you have all the data from every app that you do or don't know about
[10:20:50 EST(+1100)] <mrdon> right, though we never really tried
[10:20:54 EST(+1100)] <jazzy> configuration, backup /etc
[10:20:59 EST(+1100)] <mrdon> php apps, I've found, usually have the best upgrade path
[10:21:08 EST(+1100)] <mryall> jazzy: /home/?
[10:21:19 EST(+1100)] <mrdon> unzip the new app into the directory, then the web GUI walks you through the configuration
[10:21:22 EST(+1100)] <mrdon> soo easy
[10:22:20 EST(+1100)] <bspeakmon> the downside, of course, is that you're then using php 

[10:22:26 EST(+1100)] <mrdon> bah, php rocks 
[10:22:31 EST(+1100)] <mrdon> there's a function for everything!
[10:22:35 EST(+1100)] <mryall> haha
[10:23:24 EST(+1100)] <jazzy> /home is for users files, on most web servers these aren't that important
[10:23:29 EST(+1100)] <tmoore> the solution of course is to go hosted-only for everything
[10:24:25 EST(+1100)] <mrdon> you have to admit - php apps are probably the most hackable apps out there
[10:24:26 EST(+1100)] <Broady> i <3 mysql_real_escape_string
[10:24:34 EST(+1100)] <mrdon> so easy to tweak
[10:24:35 EST(+1100)] <mryall> actually, my favourite PHP function is array()
[10:24:43 EST(+1100)] <mrdon> managing those tweaks across upgrades, of course, is a different story
[10:25:08 EST(+1100)] <mryall> it's just hilarious that constructing an array is a function call rather than a syntactic construct
[10:25:22 EST(+1100)] <mrdon> at least they are consistent
[10:25:25 EST(+1100)] <mryall> I can't think of any other language with that "feature"
[10:25:34 EST(+1100)] <mryall> except if you count (list ...)
[10:25:56 EST(+1100)] <jazzy> Arrays.asList(o1, o2, o3) 

[10:26:12 EST(+1100)] <mrdon> touche 
[10:28:05 EST(+1100)] <Broady> scala
[10:28:45 EST(+1100)] <Broady> Array(1, 2, 3) or List(1, 2, 3)
[10:31:13 EST(+1100)] <jazzy> Eiffel: create my_array.with_capacity(10, 0)
[10:32:20 EST(+1100)] <jazzy> no way of initialising it with values in it
[10:35:30 EST(+1100)] <mryall> jazzy: you're constructing a list there, not an array
[10:35:35 EST(+1100)] <Carlfish> [NSArray arrayWithObjects: @ "One", @ "Two", nil];
[10:36:02 EST(+1100)] <mryall> oh yeah, ObjC wins hands down 
[10:36:24 EST(+1100)] <Carlfish>
http://developer.apple.com/mac/library/documentation/Cocoa/Reference/Foundation/Classes/NSArray_Class/NSArray.html
[10:36:43 EST(+1100)] <jazzy> an array list, which, when talking cross languages, can be considered an array
[10:37:20 EST(+1100)] <mryall> yeah, but there's a simpler syntax for arrays in Java: Object[] objects = { o1, o2, o3 };
[10:37:54 EST(+1100)] <Carlfish> But arrays in Java are evil and broken.
[10:38:09 EST(+1100)] <mryall> Carlfish: so is the List interface :/
[10:38:09 EST(+1100)] <Carlfish> And only there to keep C refugees happy.
[10:38:12 EST(+1100)] <jazzy> besides, how often are arrays actually used in java, in comparison to ArrayList's? ArrayList is the type in java that fulfills the general requirement for arrays in programming
[10:38:33 EST(+1100)] <bspeakmon> sun would like you to forget that java has arrays
[10:38:44 EST(+1100)] <bspeakmon> they're just a filthy rumor
[10:42:38 EST(+1100)] <jazzy> i would like you to forget that java has arrays
[10:42:38 EST(+1100)] <Carlfish> These are not the datastructures you are looking for. You can go about your business.
[10:42:38 EST(+1100)] <mryall> maybe it's because I know C, but I still think of arrays as a fixed-length consecutive memory allocation
[10:42:38 EST(+1100)] <Carlfish> Move along.
[10:42:38 EST(+1100)] <bspeakmon> they are, in C
[10:42:38 EST(+1100)] <lawson> I think they are in Java
[10:42:38 EST(+1100)] <mryall> they are in Java too, right?
[10:42:38 EST(+1100)] <lawson> system.arraycopy
[10:42:38 EST(+1100)] <lawson> just does a C memcpy
[10:42:38 EST(+1100)] <lawson> I think

[10:42:38 EST(+1100)] <mryall> anyway, my point was that there's no syntax for array construction in PHP, where as most languages that don't make your eyes bleed (i.e. Objective-C) do have such a syntax
[10:42:38 EST(+1100)] <jazzy> but, where you'd use an array in PHP, wouldn't you use a List in Java?
[10:42:38 EST(+1100)] <jazzy> in which case, Java doesn't have an equivalent
[10:42:38 EST(+1100)] <Carlfish> Yeah. System.arraycopy is native code.
[10:42:38 EST(+1100)] <Carlfish> Er, arraycopy. Hooray for things written before they came up with standards.
[10:42:56 EST(+1100)] <jazzy> like array.length
[10:42:57 EST(+1100)] <mryall> jazzy: depends; there's no fixed size array structure in PHP
[10:43:19 EST(+1100)] <Carlfish> Yeah, and Java's lack of syntax for easily creating lists and maps can be just as equally derided.
[10:43:24 EST(+1100)] <mryall> oh definitely
[10:43:33 EST(+1100)] <lawson> I think they did it that way because it's the fastest way to copy arrays
[10:43:39 EST(+1100)] <Carlfish> Having to use Arrays.asList just to create a list in a one-liner.
[10:43:40 EST(+1100)] <tmoore> even better "An array in PHP is actually an ordered map"
[10:43:46 EST(+1100)] <mryall> if you could assign to lists with the same syntax as arrays, I think I would like Java much much more
[10:43:53 EST(+1100)] <tmoore> from the docs <http://www.php.net/manual/en/language.types.array.php>
[10:43:57 EST(+1100)] <Carlfish> Or Collections.singletonList. Haha.
[10:44:23 EST(+1100)] <Carlfish> Fun things to do with Java code: pass Collections.singletonList or Arrays.asList results into random APIs that expect a list, and see what blows up.
[10:44:23 EST(+1100)] <jazzy> an object in python is just a map with a particular key pointing to its class
[10:44:25 EST(+1100)] <mryall> yeah, and now with static imports you're going to see thousands of naked asList() calls :/
[10:44:39 EST(+1100)] <Carlfish> Or, for that matter, pass in an immutable list.
[10:45:23 EST(+1100)] <jazzy> the class, in turn, is also a map
[10:45:37 EST(+1100)] <mryall> reminds me of Perl, in a bad way
[10:46:16 EST(+1100)] <Carlfish> It's possible to be reminded of Perl in a good way?
[10:46:33 EST(+1100)] <mryall> "a method is just a sub", "a class is just a module", "self is just the first argument in @_ ... umm, okay
[10:46:43 EST(+1100)] <jazzy> its great, you can override a method on an object by replacing it in its map.
[10:46:47 EST(+1100)] <Carlfish> In other news: all programming languages suck.
[10:46:54 EST(+1100)] <mryall> yeah, first class regular expressions in JS remind me of Perl in a good way
[10:47:00 EST(+1100)] <bspeakmon> "we kinda hacked OO on top of perl. SHUT UP, IT'S AWESOME"
[10:47:01 EST(+1100)] <Carlfish> (Except that cool one you've never really used)
[10:47:21 EST(+1100)] <jazzy> you mean lisp?
[10:47:32 EST(+1100)] <Carlfish> That'd be it!
[10:48:15 EST(+1100)] <jazzy> i never want to try it, i'm scared my disillusion that there is at least one cool language will prove false
[10:49:35 EST(+1100)] <jazzy> obligatory xkcd:
[10:49:37 EST(+1100)] <jazzy> <http://xkcd.com/224/>
[10:51:24 EST(+1100)] <mryall> yeah, the jury is still out for me on Scheme
[10:51:30 EST(+1100)] <mryall> need to write some serious project in it
[10:51:42 EST(+1100)] <Carlfish> Rewrite Confluence in scheme. Add a pony.
[10:51:42 EST(+1100)] <jazzy> i had a chance to use scheme
[10:51:44 EST(+1100)] <mryall> if only you could write iPhone/iPad applications in Scheme
[10:51:49 EST(+1100)] <jazzy> i used python instead
[10:52:14 EST(+1100)] <jazzy> turned out to be a lot easier to write a gimp plugin in a language i already knew
[10:56:37 EST(+1100)] * jdumay (n=jdumay@59.167.164.33) has joined #atlassiandev
[10:59:04 EST(+1100)] <Carlfish> Wow. The source for the Java System class makes me a lot less guilty about some of the hacky code I've written.
[10:59:24 EST(+1100)] <Carlfish> public final static InputStream in = nullInputStream();
[10:59:30 EST(+1100)] <Carlfish> Where nullInputStream is
[10:59:40 EST(+1100)] <Carlfish> private static InputStream nullInputStream() throws NullPointerException {
[10:59:40 EST(+1100)] <Carlfish> if (currentTimeMillis() > 0) {
[10:59:40 EST(+1100)] <Carlfish> return null;
[10:59:40 EST(+1100)] <Carlfish> }
[10:59:41 EST(+1100)] <Carlfish> throw new NullPointerException();
[10:59:41 EST(+1100)] <Carlfish> }
[10:59:55 EST(+1100)] <Broady> what the
[11:00:11 EST(+1100)] <mryall> wtf?
[11:00:16 EST(+1100)] <lawson> lol
[11:00:17 EST(+1100)] <mryall> that's nuts
[11:00:31 EST(+1100)] <bspeakmon> that's almost as bad as some of the stuff in confluence
[11:00:52 EST(+1100)] <mryall> I don't think we have functionality predicated on the time of day
[11:00:59 EST(+1100)] <mryall> except maybe the blog post publishing date
[11:01:35 EST(+1100)] <Carlfish> That URLEncoder hack got removed in 3.1, I believe.
[11:03:42 EST(+1100)] <bspeakmon> the htmlunit tests (not us, I know, but still a pain) have timebombs in them
[11:03:54 EST(+1100)] * MartinCleaver (n=martincl@206-248-161-74.dsl.teksavy.com) has joined #atlassiandev
[11:07:28 EST(+1100)] * MartinCleaver (n=martincl@206-248-161-74.dsl.teksavy.com) has joined #atlassiandev
[11:21:21 EST(+1100)] <jazzy> is that the one that runs assertions of the java code to ensure that the copyright year matches the current year?
[11:24:53 EST(+1100)] <jazzy> Carlfish: the comment above nullInputStream() makes it make sense
[11:25:14 EST(+1100)] <Carlfish> Well, yes. But it's still a weird-ass hack. 
[11:46:27 EST(+1100)] <tmoore> jazzy: yes, that's the one
[11:46:38 EST(+1100)] <tmoore> I filed a bug arguing that they should remove it and they refused
[11:47:04 EST(+1100)] <jazzy> crazy
[11:47:11 EST(+1100)] <tmoore> their reasoning was something like "why don't you just use the code from HEAD?"
[11:48:48 EST(+1100)] <jazzy> did you ask them what's the point in tagging their repository if the code in the tag will be unusable in a year?
[11:49:09 EST(+1100)] * pleschев (n=pleschев@59.167.164.33) has left #atlassiandev
[11:49:27 EST(+1100)] <tmoore> "Building previous 'snapshots' should not be encouraged, and tests of any
[11:49:28 EST(+1100)] <tmoore> 'release' are guaranteed to pass, so they can be easily escaped by 'mvn
[11:49:28 EST(+1100)] <tmoore> -Dmaven.skip.test=true"
[11:49:31 EST(+1100)] * pleschев (n=pleschев@59.167.164.33) has joined #atlassiandev
[11:50:18 EST(+1100)] <tmoore> here's the issue if you're curious
https://sourceforge.net/tracker/?func=detail&atid=448266&aid=2927131&group_id=47038

[11:51:31 EST(+1100)] <tmoore> IIRC, there are also other timezone bugs in the tests that cause tests to fail when run in Australia
[11:51:53 EST(+1100)] <tmoore> or anywhere else where the current time is past midnight UTC
[12:44:33 EST(+1100)] * puskar (n=puskar@ool-45782058.dyn.optonline.net) has joined #atlassiandev
[13:53:04 EST(+1100)] <Carlfish> Anyone awake in here who's down in the Confluence area? 

[13:53:19 EST(+1100)] <Carlfish> I'm upstairs and too lazy to check for myself if I broke the build. 
[13:54:49 EST(+1100)] <Broady> should totally have a bamboo/IRC bot
[13:55:04 EST(+1100)] <Broady> oh wait, i forgot jabber is the future
[13:56:17 EST(+1100)] <Carlfish> Wired: Twitter. Tired: Jabber. Expired: IRC
[13:57:30 EST(+1100)] <Carlfish> But publishing our build statuses on a public IRC channel that gets logged on a public website might be stretching even "Open Company No Bullshit" a bit far.
[13:58:03 EST(+1100)] <Carlfish> Although possibly not as bad as publishing our checkin logs.
[13:58:11 EST(+1100)] <skrebs> we'd all know if the wagon wheels had fallen off....
[13:58:25 EST(+1100)] <Carlfish> The donkey cart is still plodding along!
[14:00:14 EST(+1100)] * myrall is in love with Mockito
[14:00:40 EST(+1100)] <myrall> such beautiful defaults, and the verification process makes total sense
[14:01:05 EST(+1100)] <myrall> it took me several hundred tests before I arrived at this opinion, incidentally

[14:01:12 EST(+1100)] <Broady> Carlfish: i use twitter via tircd  it's good.
[14:01:31 EST(+1100)] <Carlfish> Mockito is teh pwnage.
[14:01:36 EST(+1100)] <Broady> isn't the private irc still going? that's what i was referring to
[14:01:42 EST(+1100)] <jazzy> twitter can't be compared to jabber... 140 characters per message and rate limited polling, it's impossible to hold a meaningful conversation on twitter
[14:02:56 EST(+1100)] <Carlfish> Yep, I broke the build.
[14:03:01 EST(+1100)] <Carlfish> Stupid Java.
[14:04:05 EST(+1100)] <jazzy> the hardest thing about learning mockito is learning to trust it... compared to every other mock framework i've used (i've used jmock 1 and 2, easymock and mock objects), it makes testing **too** easy, such that it leads me to mistrust it
[14:05:09 EST(+1100)] <jazzy> i feel like i should mock something to return an empty list... just to make sure... cos you never know, mockito might change its mind
[14:11:08 EST(+1100)] <Carlfish> Charles Miller committed 106635 to Confluence - Stupid @Override. Stupid Charles.
[14:11:42 EST(+1100)] <jazzy> i get that one all the time
[14:12:36 EST(+1100)] * ckiehl1 (n=Adium@59.167.164.33) has left #atlassiandev
[14:14:19 EST(+1100)] <myrall> yeah, it's hard to be confident when converting existing tests that you can just delete all the unnecessary stubs
[14:14:45 EST(+1100)] <myrall> it's especially difficult in Confluence that overloads null to mean anonymous user, among other things
[14:15:04 EST(+1100)] <myrall> you need to check that your test hasn't changed behaviour by removing the stubs
[14:19:51 EST(+1100)] <jdumay> myrall: I shudder when I come across easymock tests now that ive started using mockito
[14:24:19 EST(+1100)] <jazzy> jdumay: if that makes you shudder, don't ever look at mockobjects tests
[14:33:56 EST(+1100)] <Broady>
<http://www.news.com.au/technology/iphone-fans-linked-to-stockholm-syndrome/story-e6frfro0-1225811048192>
[14:34:04 EST(+1100)] <Broady> stick taht in your ipad
[14:46:58 EST(+1100)] <jazzy> my iphone sucks and i'm not afraid to admit it
[14:47:06 EST(+1100)] <jazzy> but its not as bad as all the other crap out there
[15:54:23 EST(+1100)] * bug (n=bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:00:17 EST(+1100)] * tmoore (n=tmoore@c-76-21-32-65.hsd1.ca.comcast.net) has joined #atlassiandev
[16:00:17 EST(+1100)] * ChanServ sets mode +o tmoore
[16:02:31 EST(+1100)] * tg davies (n=tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[16:26:18 EST(+1100)] * dchui (n=dchui@202.169.29.34) has joined #atlassiandev
[16:33:50 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has left #atlassiandev
[16:37:13 EST(+1100)] * dezwart (n=pdzwart@59.167.164.33) has joined #atlassiandev
[16:41:13 EST(+1100)] <lawson> my iphone kicks ass. I can call anyone in the world, surf the web while I'm taking a dump. Play games in meetings. Seriously, wtf else could you ask for??? X-Ray vision???

[16:42:30 EST(+1100)] <jazzy> x-ray vision would certainly be an improvement
[16:44:25 EST(+1100)] <lawson> ya that would be awesome
[16:44:46 EST(+1100)] <lawson> or the ability to record my thoughts and the thoughts of other people
[16:44:52 EST(+1100)] <lawson> but only my iphone
[16:45:04 EST(+1100)] <lawson> nobody else's iphone could have that ability
[16:46:52 EST(+1100)] * rburton- (n=rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[16:46:53 EST(+1100)] <rburton-> re
[17:26:47 EST(+1100)] * mattw_watson (n=mattw_wa@59.167.164.33) has joined #atlassiandev
[18:27:42 EST(+1100)] * skrebs (n=shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[18:33:48 EST(+1100)] * jdumay (n=jdumay@59.167.164.33) has joined #atlassiandev
[19:05:25 EST(+1100)] * sleberrig (n=sleberr@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[19:05:35 EST(+1100)] * tg davies_ (n=tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[20:43:55 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[21:05:48 EST(+1100)] * tg davies (n=tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[21:17:35 EST(+1100)] * i386 (n=jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[21:45:14 EST(+1100)] * sleberrig (n=sleberr@220.233.30.4) has joined #atlassiandev
[22:38:07 EST(+1100)] * dhardiker (n=dhardike@host86-145-166-104.range86-145.btcentralplus.com) has joined #atlassiandev
[23:03:50 EST(+1100)] * cemerick (n=la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-01-30

[23:19:09 EST(+1100)] * skrebs (n=shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[23:31:40 EST(+1100)] * i386 (n=jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[00:15:08 EST(+1100)] * skeleinei (n=stefan@84.57.170.68) has joined #atlassiandev
[00:21:29 EST(+1100)] * jonmort (n=jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[00:51:51 EST(+1100)] * whaleyl (n=whaleyl@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[00:57:14 EST(+1100)] * MartinCleaver (n=martincl@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[01:08:20 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev

[01:21:58 EST(+1100)] * leonardinius (n=leonidms@80.233.159.254) has joined #atlassiandev
[01:27:32 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[01:32:43 EST(+1100)] * jdoklovic (n=jdoklovi@66.187.202.72) has joined #atlassiandev
[01:34:59 EST(+1100)] * lmaslovs (n=leonidms@80.233.159.254) has joined #atlassiandev
[01:35:51 EST(+1100)] <jdoklovic> would it be possible to enable V2 plugins to specify a folder in the app's home directory to look for templates at runtime? This would allow making changes to report templates and such without having to re-install the plugin and restart the server
[01:35:54 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[01:36:33 EST(+1100)] <jdoklovic> sucks to tell my sysadmins they have to restart jira once again because someone wants another field displayed in a report
[01:37:43 EST(+1100)] <jdoklovic> would also allow customers to customize the templates if they wanted. should be easy to make the plugin use homeLocator to find/write templates from a jar... not sure how to make the templates load though
[01:44:22 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[02:08:35 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[04:30:54 EST(+1100)] <lmaslovs> jdoklovic - actually this coul be done (at least for e-mail templates to be stored in database)
[04:31:05 EST(+1100)] <lmaslovs> extra coding is required though
[04:46:09 EST(+1100)] <jdoklovic> @lmaslovs - I was thinking the templates that get jar'ed up in a plugin. Would be nice for a plugin to be able to write them to the home dir and have the system find them as overrides for what's in the jar
[05:04:14 EST(+1100)] * bug (n=bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[05:06:16 EST(+1100)] * tmoore (n=tmoore@216-75-233-106.static.wiline.com) has joined #atlassiandev
[05:06:16 EST(+1100)] * ChanServ sets mode +o tmoore
[05:12:13 EST(+1100)] <tmoore> jdoklovic... just caught up with the logs
[05:12:28 EST(+1100)] <tmoore> I'd imagine that it's possible, but I don't know off the top of my head... I'll dig around
[05:13:47 EST(+1100)] <tmoore> what kind of plugin is this?
[05:13:54 EST(+1100)] <tmoore> oh report
[05:28:39 EST(+1100)] <tmoore> ok looks like your best bet would be to remove the Velocity resources inside your plugin module entirely, then in the generateReportHtml/Excel methods, instead of calling descriptor.getHtml(...) you would want to use VelocityManager directly (so inject that into your report class)
[06:08:33 EST(+1100)] <jdoklovic> hmmm
[06:13:39 EST(+1100)] <jdoklovic> @tmoore, so use VM.getBody(homeTemplateDir, mytemplate, context) ?
[06:15:50 EST(+1100)] * dhardiker (n=dhardike@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[06:16:54 EST(+1100)] <tmoore> yeah I think so
[06:17:16 EST(+1100)] <tmoore> that's basically what descriptor.getHtml does internally
[06:26:21 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[06:46:41 EST(+1100)] * User283 (n=User@cpe-76-95-137-244.socal.res.rr.com) has joined #atlassiandev
[07:12:46 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[07:21:23 EST(+1100)] * sleberri_ (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[07:29:41 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[07:39:51 EST(+1100)] * puskar (n=puskar@PUSKAR.ES.ITS.NYU.EDU) has joined #atlassiandev
[08:14:25 EST(+1100)] * skrebs (n=shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[08:17:02 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:18:41 EST(+1100)] * bug_ (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:23:56 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:15:17 EST(+1100)] * bug (n=bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:32:22 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[11:30:58 EST(+1100)] * cemerick (n=la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[11:43:42 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[11:45:48 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[12:24:02 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[12:39:40 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[12:39:54 EST(+1100)] * rburton- (n=rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[12:56:59 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[13:20:37 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[13:38:55 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[13:46:37 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[13:47:58 EST(+1100)] * bug_ (n=bug@63.175.33.80) has joined #atlassiandev
[13:49:12 EST(+1100)] * jnolen (n=jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:49:23 EST(+1100)] * tg davies (n=tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[13:49:28 EST(+1100)] * justin_ (n=justin@59.167.164.33) has joined #atlassiandev
[13:50:13 EST(+1100)] * bug_ (n=bug@63.175.33.80) has joined #atlassiandev
[14:00:42 EST(+1100)] * cemerick (n=la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[14:07:27 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[14:13:39 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[14:17:10 EST(+1100)] * bug_ (n=bug@63.175.33.80) has joined #atlassiandev
[14:28:34 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[14:37:46 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[14:48:28 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[15:11:49 EST(+1100)] * bug_ (n=bug@63.175.33.80) has joined #atlassiandev
[15:52:34 EST(+1100)] * sebr (n=sruiz@amarok/developer/sebr) has joined #atlassiandev
[16:09:05 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[16:42:13 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[16:46:23 EST(+1100)] * bug_ (n=bug@63.175.33.80) has joined #atlassiandev
[16:49:27 EST(+1100)] * bug (n=bug@63.175.33.80) has joined #atlassiandev
[16:58:29 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[17:28:39 EST(+1100)] * bug_ (n=bug@63.175.33.80) has joined #atlassiandev
[17:33:55 EST(+1100)] * sleberrig (n=sleberri@4.30.233.220.static.exetel.com.au) has joined #atlassiandev
[18:42:35 EST(+1100)] * sebr- (n=sruiz@60-241-117-149.static.tpgi.com.au) has joined #atlassiandev
[19:31:43 EST(+1100)] * atlasbot (~PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[19:31:43 EST(+1100)] * Topic is 'Apple juice, for HALF PRICE | This channel is being logged with transcripts available at http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#039; set by bspeakmon on 2010-01-29 09:06:35 EST(+1100)

[19:31:47 EST(+1100)] * justin (~justin@59.167.164.33) has joined #atlassiandev
[19:31:48 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[19:31:48 EST(+1100)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[19:33:23 EST(+1100)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[19:34:40 EST(+1100)] * chuck (~charlie@yourwiki/staff/charlie) has joined #atlassiandev
[19:37:38 EST(+1100)] * mustafa (~m@193.255.135.1) has joined #atlassiandev
[19:37:53 EST(+1100)] * lawson (~chatzilla@59.167.164.33) has joined #atlassiandev
[19:39:36 EST(+1100)] * jedi (mike@li55-224.members.linode.com) has joined #atlassiandev
[19:52:38 EST(+1100)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[19:55:01 EST(+1100)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[20:00:55 EST(+1100)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[20:52:07 EST(+1100)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[20:53:51 EST(+1100)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[21:44:56 EST(+1100)] * i386 (~jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[23:02:31 EST(+1100)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev

atlassiandev_log-2010-01-31

[23:46:10 EST(+1100)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[00:11:56 EST(+1100)] * i386 (~jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[00:47:31 EST(+1100)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[00:52:17 EST(+1100)] * whaleys (~whaleys@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[03:15:58 EST(+1100)] * lawson_ (~chatzilla@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[03:16:28 EST(+1100)] * justin_ (~justin@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[03:16:53 EST(+1100)] * mrdon_ (~mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[03:17:46 EST(+1100)] * lawson (~chatzilla@59.167.164.33) has joined #atlassiandev
[03:17:46 EST(+1100)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[03:17:57 EST(+1100)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[05:06:07 EST(+1100)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[06:20:18 EST(+1100)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[06:28:12 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[06:45:55 EST(+1100)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[06:49:42 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[07:07:50 EST(+1100)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[08:17:31 EST(+1100)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[08:48:54 EST(+1100)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:09:23 EST(+1100)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[09:11:36 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:18:24 EST(+1100)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:28:15 EST(+1100)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[09:29:14 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:33:57 EST(+1100)] * jonmort (~jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[09:44:59 EST(+1100)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:51:49 EST(+1100)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:55:34 EST(+1100)] * whaleys (~whaleys@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[09:58:54 EST(+1100)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[09:58:54 EST(+1100)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[09:58:54 EST(+1100)] * i386 (~jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[09:58:54 EST(+1100)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[09:58:54 EST(+1100)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[09:58:54 EST(+1100)] * chuck (~charlie@yourwiki/staff/charlie) has joined #atlassiandev
[09:58:54 EST(+1100)] * jedi (mike@li55-224.members.linode.com) has joined #atlassiandev
[09:58:54 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:58:54 EST(+1100)] * mustafa (~m@193.255.135.1) has joined #atlassiandev
[09:58:54 EST(+1100)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[09:58:54 EST(+1100)] * lawson (~chatzilla@59.167.164.33) has joined #atlassiandev
[09:59:14 EST(+1100)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[10:03:12 EST(+1100)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[10:13:01 EST(+1100)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[10:58:15 EST(+1100)] * bug (~bug@adsl-76-211-229-44.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[11:27:25 EST(+1100)] * bug (~bug@adsl-76-211-229-44.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[11:29:25 EST(+1100)] * bug (~bug@adsl-76-211-229-44.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[13:36:46 EST(+1100)] * rburton (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[13:51:10 EST(+1100)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[14:51:00 EST(+1100)] * jensschumacher (~anonymous@124.171.58.93) has joined #atlassiandev
[15:17:16 EST(+1100)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:19:35 EST(+1100)] * Carlfish (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[16:51:50 EST(+1100)] * tg davies (~tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[19:17:33 EST(+1100)] * tg davies (~tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[19:21:09 EST(+1100)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[19:43:02 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[21:07:56 EST(+1100)] * lawson_ (~chatzilla@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[21:08:12 EST(+1100)] * mrdon_ (~mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[21:08:19 EST(+1100)] * justin_ (~justin@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[21:09:41 EST(+1100)] * lawson (~chatzilla@59.167.164.33) has joined #atlassiandev
[21:09:59 EST(+1100)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[21:10:06 EST(+1100)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[21:38:24 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[22:48:07 EST(+1100)] * jensschumacher (~anonymous@124.171.58.93) has joined #atlassiandev

atlassiandev_log-2010-02-01

[00:17:29 EST(+1100)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[01:20:25 EST(+1100)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavy.com) has joined #atlassiandev
[01:51:29 EST(+1100)] * jonmort (~jonmort@ccp1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[02:09:54 EST(+1100)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:42:27 EST(+1100)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavy.com) has joined #atlassiandev
[03:35:30 EST(+1100)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[04:12:59 EST(+1100)] * whaley (~whaley@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[04:27:21 EST(+1100)] * _5cats (~jd@2002:48cf:2dae:0:225:ff:fe4d:44bc) has joined #atlassiandev
[04:41:34 EST(+1100)] * lawson_ (~chatzilla@59.167.164.33) has joined #atlassiandev
[05:06:03 EST(+1100)] * cemerick (~la_mer@64.241.37.140) has joined #atlassiandev
[05:49:25 EST(+1100)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[05:49:51 EST(+1100)] * bug_ (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[06:07:05 EST(+1100)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[06:07:05 EST(+1100)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[06:11:16 EST(+1100)] * lawson (~chatzilla@59.167.164.33) has joined #atlassiandev
[07:22:41 EST(+1100)] * tgdavies (~tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[07:38:35 EST(+1100)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[07:58:58 EST(+1100)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavy.com) has joined #atlassiandev
[07:59:22 EST(+1100)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[08:16:56 EST(+1100)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[08:24:20 EST(+1100)] * tgdavies (~tomd@174.8.70.115.static.exetel.com.au) has joined #atlassiandev
[08:53:32 EST(+1100)] * myrall (~myrall@59.167.164.33) has joined #atlassiandev
[08:53:33 EST(+1100)] * ChanServ sets mode +o myrall
[08:53:45 EST(+1100)] <myrall> morning folks
[08:54:26 EST(+1100)] <dezwart> doffs hat to myrall
[10:22:45 EST(+1100)] * atlasbot (~PircBot@ppp121-45.static.internode.on.net) has joined #atlassiandev
[10:22:45 EST(+1100)] * Topic is 'Apple juice, for HALF PRICE | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#>; set by bspeakmon on 2010-01-29 09:06:35 EST(+1100)
[10:51:17 EST(+1100)] <mryall> mrdon: I've made a minor change to Atlasbot to fix some colours in the transcripts
[10:51:20 EST(+1100)] <mryall> how do I bounce him?
[10:51:52 EST(+1100)] <mrdon> well, until sysadmins give me a server, there only way is to wait till I get home :/
[10:52:09 EST(+1100)] <mryall> oh
[10:52:17 EST(+1100)] <mryall> hrm
[10:53:54 EST(+1100)] <Broady> what did you think of pircbot mryall
[10:53:56 EST(+1100)] <Broady> mrdon: **
[10:54:12 EST(+1100)] <mryall> what's pircbot?
[10:54:43 EST(+1100)] <Broady> isn't that what atlasbot is running on?
[10:54:54 EST(+1100)] <Broady> java library for irc bots
[10:56:12 EST(+1100)] <mryall> ah yes
[10:56:13 EST(+1100)] <mryall> it is
[10:56:15 EST(+1100)] <mrdon> I love pircbot
[10:56:16 EST(+1100)] <mryall> I haven't dug that far yet
[10:56:22 EST(+1100)] <mrdon> was my first bot framework
[10:56:35 EST(+1100)] <mrdon> and yes, I think atlasbot uses it as well
[10:57:33 EST(+1100)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[10:57:50 EST(+1100)] <Broady> i was looking for a irc bot that would do hot loadable plugins, that'd be cool
[10:58:10 EST(+1100)] <Broady> then started writing one with osgi then brain exploded
[11:04:02 EST(+1100)] <mryall> you could use atlassian-plugins
[11:04:16 EST(+1100)] <mryall> Don has tidied it up quite a bit
[11:04:49 EST(+1100)] <mryall> <https://studio.atlassian.com/wiki/display/PLUG/Atlassian+Plugins+Framework>
[11:05:01 EST(+1100)] <mryall> not sure whether that URL is public yet, but you probably still have access
[11:07:37 EST(+1100)] <Broady> yup, public
[11:08:15 EST(+1100)] <Broady> ill give it a whirl, could be something exciting to break up days of windows server administration
[11:57:23 EST(+1100)] <jazzy> Broady: have you finished at csc?
[12:06:37 EST(+1100)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev
[12:21:42 EST(+1100)] <rburton-> re
[12:21:57 EST(+1100)] <mrdon> rburton-: fancy meeting you here
[12:22:09 EST(+1100)] <rburton-> How's it going Don
[12:22:29 EST(+1100)] <mrdon> pretty good, how's life?
[12:22:51 EST(+1100)] <rburton-> Going very well, working on a personal project which I plan to market in a month and working as a consultant
[12:22:56 EST(+1100)] <rburton-> Over all, very well
[12:23:03 EST(+1100)] <mrdon> nice
[12:23:09 EST(+1100)] <rburton-> I'll be in San Fran for google IO if you're going to be in town.
[12:23:15 EST(+1100)] <mrdon> probably not
[12:23:21 EST(+1100)] <mrdon> have a baby on the way, so am needed at home
[12:23:30 EST(+1100)] <rburton-> Congrats!
[12:23:33 EST(+1100)] <rburton-> Second one no?
[12:23:36 EST(+1100)] <mrdon> yep
[12:23:40 EST(+1100)] <rburton-> Very nice 
[12:23:43 EST(+1100)] <rburton-> Boy/Girl?
[12:23:49 EST(+1100)] <mrdon> girl
[12:23:55 EST(+1100)] <rburton->  very nice
[12:24:02 EST(+1100)] <rburton-> how are things with you?
[12:24:22 EST(+1100)] <mrdon> pretty good, nothing really new though
[12:24:24 EST(+1100)] <rburton-> I've picked up a copy of Bamboo for my project (10 user lic)

[12:24:26 EST(+1100)] <mrdon> other than a bit of gwt stuff lately
[12:24:50 EST(+1100)] <rburton> How do you like GWT? I found it to be a little too 'noisy' with all of the interfaces etc.. but that was with an older version.
[12:25:02 EST(+1100)] <mrdon> yeah, still is quite verbose coming from a javascript world
[12:25:09 EST(+1100)] <mrdon> but good tools make a world of difference
[12:25:15 EST(+1100)] <mrdon> though I still find it sad they are necessary
[12:25:19 EST(+1100)] <rburton> Using IDEA 9x?
[12:25:24 EST(+1100)] * rburton- noids
[12:25:26 EST(+1100)] <mrdon> 9.0.2 EAP
[12:25:34 EST(+1100)] <mrdon> as it finally supports uibinder
[12:25:36 EST(+1100)] <mrdon> ...kinda
[12:25:47 EST(+1100)] <rburton> I'm on 9.0.1
[12:26:09 EST(+1100)] <rburton> Bought a copy Friday and got he lic yesterday.
[12:26:09 EST(+1100)] <mrdon> personally, I wonder if I would like wicket more
[12:26:20 EST(+1100)] <rburton> I honestly don't like wicket
[12:26:30 EST(+1100)] <mrdon> but I haven't done a lot of client-side stuff, so I'm withholding judgement
[12:26:43 EST(+1100)] <rburton> I do like the HTML presentation layer and how the AJAX stuff is done though
[12:27:20 EST(+1100)] <rburton> I hope someone replaces HTML honestly
[12:27:29 EST(+1100)] <mrdon> not likely



[12:27:32 EST(+1100)] <rburton> I know 😞
[12:27:38 EST(+1100)] <mrdon> html 5 is looking nice though
[12:28:03 EST(+1100)] <rburton> I agree they've done some nice things to it
[12:28:10 EST(+1100)] <mrdon> and being able to put most of the presentation logic on the client, including form workflow, with services on the backend is a nice arch
[12:28:15 EST(+1100)] <rburton> Sadly I find it to be simular to JPA vs. Hlbernate
[12:28:20 EST(+1100)] <rburton> they're lacking
[12:29:00 EST(+1100)] <rburton> They're going in the right direction, but they could have done a better job. I guess I'm concern that it'll be that way for another 10 years before an update comes out
[12:29:45 EST(+1100)] <rburton> mrdon, Are you guys looking at writing tools to allow clients to manage multiple projects using JIRA, Bamboo, etc. from a single interface?
[12:29:48 EST(+1100)] <rburton> like jira.com



[12:29:51 EST(+1100)] <mrdon> we are about to drop support for IE 6, so I'm counting my blessings 😊
[12:29:58 EST(+1100)] <mrdon> you mean ala studio?
[12:30:03 EST(+1100)] <rburton> Yes
[12:30:06 EST(+1100)] <rburton> IE6--
[12:30:15 EST(+1100)] <mrdon> right, so I was the lead of that project for its first year and a bit
[12:30:36 EST(+1100)] <mrdon> and yes, dev is ongoing
[12:30:41 EST(+1100)] <mrdon> what tools did you have in mind?
[12:31:08 EST(+1100)] <rburton> My client has Jira, Bamboo, etc. and when a new project comes onboard, they need to go into each of those tools and create a project there etc.
[12:31:17 EST(+1100)] <rburton> Mind you, they love the Atlassian products
[12:31:18 EST(+1100)] <mrdon> ah, so studio behind the firewall



[12:31:30 EST(+1100)] <mrdon> the much-talked about, never started project 😞
[12:31:33 EST(+1100)] <rburton> Right
[12:31:58 EST(+1100)] <mrdon> well, some good steps happening now...am starting on applinks 3.0, which will help tie our apps together in a supported way
[12:32:03 EST(+1100)] <rburton> I think the tools integrate well, now the next step I see is to bring the automation and management to the clients personal setup
[12:32:03 EST(+1100)] <mrdon> and a common event bus
[12:32:13 EST(+1100)] <mrdon> so once you have those, you can start building things on top of that
[12:32:22 EST(+1100)] <mrdon> gwt might give us a common UI framework across products
[12:32:28 EST(+1100)] <mrdon> which has been another stumbling block
[12:32:32 EST(+1100)] <rburton> So events are emitted and other systems can subscribe etc.?
[12:32:47 EST(+1100)] <mrdon> well, intravm for now
[12:32:57 EST(+1100)] <mrdon> but there is a web hook plugin called notifier or something
[12:33:03 EST(+1100)] <rburton> My client was tickled pink when they saw a svn commit can link things together to JIRA, etc..
[12:33:10 EST(+1100)] <rburton> That's nice
[12:33:17 EST(+1100)] <mrdon> yeah, that's the stuff we are working on
[12:33:19 EST(+1100)] <mrdon> better integration
[12:33:34 EST(+1100)] <rburton> It would be interested to see product to product integration through events outside of the vm
[12:34:06 EST(+1100)] <rburton> Where a change in a Jira ticket could trigger a JiraTicketChangeEvent (or whatever) and systems that subscribe could be notified
[12:34:06 EST(+1100)] <mrdon> yeah, we almost went down that way for studio at the start
[12:34:11 EST(+1100)] <mrdon> via jms
[12:34:22 EST(+1100)] <mrdon> we'll see if a form of it gets resurrected
[12:34:45 EST(+1100)] <rburton> I think that'll open the door for plug-ins to help move integration forward
[12:35:01 EST(+1100)] <rburton> There's a lot of 'enterprise' tooling that can come from that
[12:35:06 EST(+1100)] <mrdon> yep



[12:35:41 EST(+1100)] <rburton> Justin is back in the states playing lots of hockey 😊
[12:36:22 EST(+1100)] <rburton> I'm working on an interesting project for application configuration management
[12:36:43 EST(+1100)] <rburton> should be useful for large companies who want a solution for managing application configuration
[12:36:59 EST(+1100)] <rburton> trying to provide a centralized form that allows for realtime changes and impact analysis etc.
[12:37:04 EST(+1100)] <mrdon> cool
[12:37:33 EST(+1100)] <rburton> I don't get the iPad I think its a complete waste
[12:38:30 EST(+1100)] <mrdon> no need for it myself
[12:38:31 EST(+1100)] <mryall> waste of what?
[12:38:37 EST(+1100)] <rburton> Time and money

[12:38:43 EST(+1100)] <rburton-> It's a huge ipod touch 😊

[12:39:19 EST(+1100)] <mryall> I guess it's lucky you won't be wasting either on it, then 😊

[12:39:20 EST(+1100)] <rburton-> Even the icons at the bottom have the same number of icons the iphone has pretty sad

[12:39:32 EST(+1100)] <mryall> yeah, the app screen looks a bit bare with only a four-by-four grid

[12:39:57 EST(+1100)] <mryall> I wonder how you're going to access files that you create with Keynote, etc.

[12:40:13 EST(+1100)] <rburton-> special software 😊 or the Apple protocol

[12:40:16 EST(+1100)] <mryall> you have to have some way to email them to people, etc.

[12:41:08 EST(+1100)] <mryall> I think it will appeal mostly to people who aren't programmers or IT people

[12:41:32 EST(+1100)] <mryall> people who struggle with using computers at the moment

[12:41:33 EST(+1100)] <rburton-> I think it's going to flop unless they do some upgrades soon

[12:41:51 EST(+1100)] <mryall> oh? you mean more functionality?

[12:41:58 EST(+1100)] <rburton-> all around

[12:42:06 EST(+1100)] <rburton-> it's too basic compared to the ipod touch

[12:42:10 EST(+1100)] <rburton-> I don't see the benfit

[12:42:23 EST(+1100)] <rburton-> except a large screen

[12:48:24 EST(+1100)] <mryall> I thought it was the same as a touch, pretty much? how is it more basic?

[12:49:53 EST(+1100)] <rburton-> What I mean by the statement is, I expected a lot more than what the iPod touch gives. I really mean 'more basic than what I expected compared to the touch'

[12:50:16 EST(+1100)] <mryall> right

[12:50:24 EST(+1100)] <mryall> it's a fair bit bigger and faster, so you'd expect more, right?

[12:50:51 EST(+1100)] <mryall> some kind of multitasking would be a good start, I think

[12:51:02 EST(+1100)] <mryall> it's pretty irritating to have to quit one app to use another

[12:51:08 EST(+1100)] <rburton-> Exactly something so basic like that

[12:52:51 EST(+1100)] <Broady> jazzy: yup, now im at cca (coke amatil)

[12:53:13 EST(+1100)] <mryall> Broady: do they have free drinks in the fridge there? 😊

[12:53:25 EST(+1100)] <Broady> mryall: yahuh. so if you ever want some free beverages, drop by 😊

[12:53:32 EST(+1100)] <mryall> sweet 😊

[12:53:47 EST(+1100)] <Broady> oh wait. theres plenty of drink at atlassian. hahaha

[12:54:01 EST(+1100)] <mryall> yeah, we need to get some more of the diet stuff though

[12:54:09 EST(+1100)] <mryall> too much sugar makes matt something something

[12:54:27 EST(+1100)] <Broady> actually people mostly drink water here, don't see much other than mt franklin being consumed

[12:55:56 EST(+1100)] <mryall> heh, interesting

[13:15:14 EST(+1100)] * cemerick (-la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlasiandev

[13:25:21 EST(+1100)] <jazzy> speaking of coke amatil, i was recently in PNG, i bought a can of solo, it was made by coke amatil PNG

[13:26:48 EST(+1100)] <jazzy> which i thought is odd because the solo trademark is owned by schweppes

[13:28:35 EST(+1100)] <jedi> yeah, normally coke has Lift

[13:29:22 EST(+1100)] <jazzy> on the can it said it was made by CCA under license of schweppes

[13:29:55 EST(+1100)] <mryall> speaking of soft drinks, someone finished all the diet coke in the fridge and didn't replenish it

[13:30:23 EST(+1100)] * mryall looks at Carlfish (notably absent)

[13:30:41 EST(+1100)] <jazzy> i thought the fridge replenished itself? 😊

[13:42:43 EST(+1100)] <mryall> apparently not 😊

[13:48:02 EST(+1100)] * bug (-bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlasiandev

[13:50:10 EST(+1100)] <rburton-> VERY STRANGE

[13:50:16 EST(+1100)] <rburton-> Are people hacking gmail accounts?

[13:50:34 EST(+1100)] <rburton-> Someone created a new account using my gmail account and I just changed the password and claimed the new account 😊

[13:54:02 EST(+1100)] * MartinCleaver (-martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlasiandev

[14:52:35 EST(+1100)] * cemerick (-la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlasiandev

[15:39:34 EST(+1100)] <Broady> jazzy: yep, schweppes is distributed by CCA in nz and some other places

[15:53:49 EST(+1100)] <justin_> jazzy: CCA is a bottler, CocaCola owns a minor stake in CCA.

[17:02:37 EST(+1100)] * tg davies (-tomd@59.167.164.33) has joined #atlasiandev

[17:04:33 EST(+1100)] * dezwart (-pdzwart@59.167.164.33) has left #atlasiandev

[17:52:55 EST(+1100)] * _hen (-hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlasiandev

[18:11:50 EST(+1100)] * bug (-bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlasiandev

[18:47:19 EST(+1100)] * skleinei (-stefan@dsb-084-057-136-200.pools.arcor-ip.net) has joined #atlasiandev

[19:37:31 EST(+1100)] * _hen (-hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlasiandev

[19:37:58 EST(+1100)] * tg davies (-tomd@225.38.70.115.static.exetel.com.au) has joined #atlasiandev

[19:40:10 EST(+1100)] * jonmort (-jonmort@ccp1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlasiandev

[20:04:30 EST(+1100)] * lmaslovs (-leonidms@80.233.159.254) has joined #atlasiandev

[21:09:03 EST(+1100)] * trochej (trochej@noches.pl) has joined #atlasiandev

[21:29:56 EST(+1100)] <skleinei> jdoklovic: you can load resources from the development dir by setting a system property like this:
-Dplugin.resource.directories=/Users/stefan/Development/Projects/eddie/trunk/src/main/resources

[21:30:22 EST(+1100)] <skleinei> (this was somewhere in DEVNET, but I can't find the page anymore)

[21:34:43 EST(+1100)] * kalamon (-kalamon@chello089074130182.chello.pl) has joined #atlasiandev

[21:37:57 EST(+1100)] <skleinei> just found it:
<http://confluence.atlassian.com/display/PLUGINFRAMEWORK23/Instant+Loading+of+Plugin+Resources>

[21:46:55 EST(+1100)] <trochej> Hi, how general and nonspecific are we allowed to be here?

[22:16:39 EST(+1100)] * cemerick (-la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlasiandev

[22:17:39 EST(+1100)] * sebr (-sruiz@amarok/developer/sebr) has joined #atlasiandev

[22:26:39 EST(+1100)] * i386 (-jdumay@ppp201-143.static.internode.on.net) has joined #atlasiandev

atlasiandev_log-2010-02-02

[23:18:08 EST(+1100)] * cemerick (-la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlasiandev
[00:23:45 EST(+1100)] * rburton- (-rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlasiandev

[00:51:31 EST(+1100)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavy.com) has joined #atlassiandev
[01:36:45 EST(+1100)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[01:41:06 EST(+1100)] * jdoklovic (~jdoklovic@66.187.202.66) has joined #atlassiandev
[03:11:08 EST(+1100)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[03:12:58 EST(+1100)] * AJC_Z0 (~nnAJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[04:49:39 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[06:01:54 EST(+1100)] <rodogus> hey, any idea on when the 3.2 EAP is going to start?
[06:02:10 EST(+1100)] <rodogus> (Confluence 3.2 EAP, that is)
[06:18:59 EST(+1100)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[06:23:17 EST(+1100)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[06:23:18 EST(+1100)] * ChanServ sets mode +o bspeakmon
[07:16:00 EST(+1100)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[07:33:02 EST(+1100)] * whaleys (~whaleys@i35-137.members.linode.com) has joined #atlassiandev
[07:33:26 EST(+1100)] * whaleys (~whaleys@i35-137.members.linode.com) has joined #atlassiandev
[07:34:47 EST(+1100)] * whaleys (~whaleys@i35-137.members.linode.com) has joined #atlassiandev
[08:42:08 EST(+1100)] * tg davies (~tomd@59.167.164.33) has joined #atlassiandev
[08:46:48 EST(+1100)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:47:30 EST(+1100)] * sleberrigaud (~sleberrigaud@59.167.164.33) has joined #atlassiandev
[09:12:05 EST(+1100)] <mryall> when we have a milestone with some decent work in it
[09:13:02 EST(+1100)] <mryall> I think probably within two or three weeks we'll release 3.2-m3 publicly
[09:13:30 EST(+1100)] <mryall> if you want to mess around with it without a proper distribution, the earlier milestone WARs go straight to the Maven repo
[09:55:38 EST(+1100)] * jnolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[10:17:37 EST(+1100)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev
[10:22:46 EST(+1100)] * jdumay (~jdumay@59.167.164.33) has joined #atlassiandev
[10:31:03 EST(+1100)] * jnolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[10:31:44 EST(+1100)] * tg davies (~tomd@59.167.164.33) has joined #atlassiandev
[10:31:53 EST(+1100)] * sleberrigaud (~sleberrigaud@59.167.164.33) has joined #atlassiandev
[10:43:17 EST(+1100)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[10:59:10 EST(+1100)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[11:04:34 EST(+1100)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[11:17:09 EST(+1100)] * ckiehl (~Adium@59.167.164.33) has joined #atlassiandev
[11:54:16 EST(+1100)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[12:27:05 EST(+1100)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[13:18:54 EST(+1100)] * mustafa (~m@193.255.135.1) has joined #atlassiandev
[13:57:33 EST(+1100)] <rburton-> re
[13:57:40 EST(+1100)] <rburton-> Anyone here get git + bamboo working?
[13:57:49 EST(+1100)] <rburton-> github is the hosting provider for git
[14:05:36 EST(+1100)] <jensschumacher> Haven't tried it... but there is a Bamboo Git plugin: <http://slnc.me/bamboo-git-plugin/>
[14:13:44 EST(+1100)] * mustafa (~m@193.255.135.1) has joined #atlassiandev
[14:32:15 EST(+1100)] <rburton-> mrdon, ping
[14:36:14 EST(+1100)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[14:40:40 EST(+1100)] <mryall> he's not at his desk currently, rburton-, anything someone else could help with?
[15:03:53 EST(+1100)] <rburton-> I know he's looked at git with Bamboo I wanted to see if there's any good information for setting it up on windows

[15:04:15 EST(+1100)] <rburton-> Stepka should be sitting next to you guys, but noo he had to creep back to SF
[15:15:02 EST(+1100)] * pramod (~pramod@59.167.164.33) has joined #atlassiandev
[15:30:25 EST(+1100)] * pramod (~pramod@59.167.164.33) has left #atlassiandev
[15:32:01 EST(+1100)] * testatlasbot (~pramod@59.167.164.33) has joined #atlassiandev
[15:34:50 EST(+1100)] * pramod (~pramod@59.167.164.33) has joined #atlassiandev
[15:47:25 EST(+1100)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[17:06:37 EST(+1100)] * pramod (~pramod@59.167.164.33) has left #atlassiandev
[18:28:48 EST(+1100)] * skeinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[18:52:26 EST(+1100)] <trochej> Hi
[18:53:34 EST(+1100)] <trochej> I have a jira-charting-plugin-1.5.jar for Jira 4.0.1. When I build jira with this plugin in edit-webapp/WEB-INF/lib/ Jira complains that OSGi plugins cannot be deployed that way. What would be a proper way of including it in my instance of Jira?
[19:04:19 EST(+1100)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[19:12:57 EST(+1100)] * tg davies (~tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[19:13:01 EST(+1100)] <trochej> Also, would there be any suggestions, why, if I deploy bamboo 2.5.1 on the same tomcat 5.5.28 that Jira 4.0.1 and configure Jira to connect to this bamboo, after restarting tomcat Jira hangs just after connecting to database
[16:59:00 CST(-0600)] * atlasbot (~PircBot@63-246-22-217.contegix.com) has joined #atlassiandev
[16:59:00 CST(-0600)] * Topic is 'Apple juice, for HALF PRICE | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#>; set by bspeakmon on 2010-01-28 16:06:35 CST(-0600)

[16:59:38 CST(-0600)] <pramod> atlasbot is now running from a contegix server
[17:00:08 CST(-0600)] <pramod> I'll add a page on EAC to provide some doc
[17:07:04 CST(-0600)] <dezwart_> pramod: chrooted I hope.
[17:29:11 CST(-0600)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[17:32:04 CST(-0600)] <pramod> it's java
[17:33:23 CST(-0600)] * pramod (~pramod@59.167.164.33) has left #atlassiandev
[17:34:28 CST(-0600)] <Broady> famous last words
[17:45:16 CST(-0600)] * MartinCleaver (~martincle@66-207-222-14.beanfield.net) has joined #atlassiandev
[18:03:23 CST(-0600)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[18:25:02 CST(-0600)] * atlasbot (~PircBot@63-246-22-217.contegix.com) has joined #atlassiandev
[18:25:02 CST(-0600)] * Topic is 'Apple juice, for HALF PRICE | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#>; set by bspeakmon on 2010-01-28 16:06:35 CST(-0600)

[18:26:14 CST(-0600)] <rburton-> re
[18:50:51 CST(-0600)] <rburton-> Im obtaining "Missing class: com/atlassian/bamboo/repository/WebRepositoryEnabledRepository" for a plug-in, isn't that a Bamboo class and it should be in version 2.5 right?
[18:54:36 CST(-0600)] <jazzy> there were some changes around web repositories in bamboo 2.5, it's possible that that interface is now

gone, and the plugin needs to be updated

[18:55:03 CST(-0600)] <jazzy> your best bet is raising a support request

[18:55:08 CST(-0600)] <jazzy> what plugin is it?

[18:55:18 CST(-0600)] <rburton-> git based plug-in

[18:55:31 CST(-0600)] <rburton-> I have the code checked out, if I can find out what class replaces it, I'll create a patch

[18:56:24 CST(-0600)] <jazzy> gimme a tick

[18:56:33 CST(-0600)] <rburton-> tick?

[18:56:48 CST(-0600)] <jazzy> gimme a second, i'm looking up the code

[18:56:53 CST(-0600)] <rburton-> k

[18:58:16 CST(-0600)] <jazzy> yes, it has definitely been deleted

[18:58:57 CST(-0600)] <rburton-> Ok, I wonder what the replacement is

[18:59:05 CST(-0600)] * rburton- looks at how the code uses it

[18:59:29 CST(-0600)] <rburton-> Ah they implement WebRepositoryEnabledRepository

[19:00:38 CST(-0600)] <jazzy> the proper fix is to write a new web repository plugin (this is a new plugin point to allow for different web repository apps)

[19:01:02 CST(-0600)] <jazzy> but to make the plugin work for now, just remove the interface and the implemented methods from the class

[19:01:45 CST(-0600)] <rburton-> What's the right fix? Is there an interface to implement or has the API changed a great deal?

[19:02:30 CST(-0600)] <jazzy> it used to be the responsibility of the repository plugin to handle how links to the web interface to the repository were generated

[19:02:47 CST(-0600)] <rburton-> so now it's no longer required by the plug-in?

[19:02:54 CST(-0600)] <jazzy> no

[19:04:57 CST(-0600)] <jazzy> unfortunately there's no docs yet for implementing the new plugin point for web repositories, but I think a web repository viewer plugin has to implement the com.atlassian.bamboo.webrepository.WebRepositoryViewer interface, and declared in the plugin descriptor using <webRepositoryViewer ...>

[19:07:03 CST(-0600)] <jazzy> but, if you don't need links to changed source files and commits etc rendered, you don't need this, or if you're using FishEye, then when configuring a source repository in a plan, you can select the fisheye web repository viewer

[19:14:50 CST(-0600)] <rburton-> Nice I'll look into this

[19:14:56 CST(-0600)] * rburton- is making changes now

[19:15:04 CST(-0600)] <rburton-> thanks

[19:46:12 CST(-0600)] <jazzy> rburton-: <http://confluence.atlassian.com/display/BAMBOO/Web+Repository+Viewer+Module>

[19:46:19 CST(-0600)] <jazzy> written just then

[19:47:01 CST(-0600)] <jazzy> (not by me)

[19:52:35 CST(-0600)] <rburton-> nice

[19:52:36 CST(-0600)] <rburton-> thanks

[20:55:20 CST(-0600)] <rburton-> Hmm, another API change that's breaking the plug-in.

[20:55:26 CST(-0600)] <rburton-> Wonder where I can find a list of changes

[20:55:57 CST(-0600)] <rburton-> AbstractRepository.WEB_REPO_URL Was removed

[21:40:28 CST(-0600)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev

[22:01:18 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev

[22:08:51 CST(-0600)] <trochej> dezwart_: chroot is not a security device 😊

[22:09:48 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev

[22:29:11 CST(-0600)] <jazzy> rburton-: that makes sense, it's part of the same thing

[22:30:35 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

[22:45:23 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev

[23:49:10 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-02-03

[01:18:29 CST(-0600)] <trochej> e, we have Jira 3.13.2 with Bamboo 2.3.1 in the same Tomcat and it works.

[01:18:42 CST(-0600)] <trochej> frag

[01:18:57 CST(-0600)] <trochej> jazzy As a side note, we have Jira 3.13.2 with Bamboo 2.3.1 in the same Tomcat and it works.

[01:19:31 CST(-0600)] <trochej> I irc from phone. Ircing is not exactly approved in my company.

[01:19:48 CST(-0600)] <trochej> I actually break few rules to get us going.

[01:19:52 CST(-0600)] <trochej> 😊

[01:27:33 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev

[01:36:31 CST(-0600)] <jazzy> i irc from the phone when I'm on the train

[01:38:49 CST(-0600)] <trochej> I irc from the phone when I help my company using communication channels unapproved. 😊

[01:40:00 CST(-0600)] <trochej> And I hate to waste that free 1 GB internet I got from my mobile operator. 😊

[01:40:29 CST(-0600)] <Broady> sorry to hear

[01:43:26 CST(-0600)] <trochej> What, that I can't stand all those unused bytes polluting cyberspace? 😊

[01:43:51 CST(-0600)] <Broady> no, that your company won't let you IRC 😊

[01:45:13 CST(-0600)] <trochej> Heh, they won't let me more. I guess this is because they are root CA for Polish social security system

[01:45:32 CST(-0600)] <trochej> A bit of paranoia 😊

[01:45:42 CST(-0600)] <Broady> ah, there are always ways 😊 i tunnel through a http proxy to my ssh server

[01:46:12 CST(-0600)] <trochej> Broad I thought of it, but it means that there still is a session out

[01:46:41 CST(-0600)] <trochej> I use ssh to an outside host which hosts my irssi in screen anyway 😊

[01:47:11 CST(-0600)] <trochej> And unidentifies ssh session is a no-no.

[01:47:55 CST(-0600)] <trochej> I could forward it on 443, but then I'd get questioned what ciphered page I keep connected to all day

[01:48:26 CST(-0600)] <trochej> So it is easier and less invasive to just use my phone

[01:48:28 CST(-0600)] <trochej> qwerty

[01:48:35 CST(-0600)] <trochej> Can live with that

[01:48:38 CST(-0600)] <trochej> 😊
[01:53:51 CST(-0600)] <trochej> Is there a way to give a Jira user permission to create component version but not give the permission to add users to project?
[01:55:46 CST(-0600)] * i386|laptop (~jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[02:31:18 CST(-0600)] <Broady> trochej: yup i figured that too, but nothing's happened 😊
[02:31:50 CST(-0600)] <Broady> then again, its only one connection, could be comet or something
[02:33:34 CST(-0600)] <trochej> 😊
[02:33:54 CST(-0600)] <trochej> Have a coffee, then. 😊
[02:33:56 CST(-0600)] <trochej> 😊
[02:41:10 CST(-0600)] <jazzy> fortunately, the worst i've ever had at work is needing to use a non standard port for ssh
[02:42:23 CST(-0600)] <trochej> Worst that I had was No Coffee. I quit faster than I applied. 😊
[02:42:31 CST(-0600)] <jazzy> lol
[02:43:57 CST(-0600)] <jazzy> i was once close to accepting a project for asio (australian security intelligence organisation)
[02:44:13 CST(-0600)] <jazzy> i hate to think what i wouldn't have been allowed had i have accepted it 😊
[02:45:26 CST(-0600)] <Broady> does atlassian have a coffee machine yet? 😊
[02:45:40 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[02:45:41 CST(-0600)] <jazzy> lol no
[02:45:49 CST(-0600)] <jazzy> but we got a new super duper fridge yesterday
[02:45:55 CST(-0600)] <Broady> what's going in it?
[02:46:05 CST(-0600)] <jazzy> beer and softdrink
[02:46:23 CST(-0600)] <jazzy> its got two glass doors
[02:46:28 CST(-0600)] <Broady> nice
[02:52:13 CST(-0600)] * i386|laptop (~jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[02:54:53 CST(-0600)] * skeleinei_ (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[02:56:15 CST(-0600)] <trochej> We had three fridges and coffee-grinders at my previous job
[02:57:18 CST(-0600)] <trochej> Can I repeat my question? 😊
[02:57:47 CST(-0600)] <trochej> Is there a way to give a Jira user permission to create component version but not give the permission to add users to project?
[02:57:54 CST(-0600)] <trochej> I did anyway 😊
[02:58:36 CST(-0600)] <trochej> Coffee
[03:05:42 CST(-0600)] * skeleinei_ (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[03:24:28 CST(-0600)] * skeleinei_ (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[03:31:37 CST(-0600)] * skeleinei_ (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[04:19:09 CST(-0600)] * i386|laptop (~jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[04:59:13 CST(-0600)] <trochej> jazzy dezwart If you are interested: JSP-51856
[05:17:38 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[05:59:27 CST(-0600)] * lmaslovs slaps lmaslovs around a bit with a large trout
[06:34:47 CST(-0600)] * trochej has coffee
[06:34:50 CST(-0600)] <trochej> eeeeeeeeeeeeee
[07:16:00 CST(-0600)] * MartinCleaver (~martincline@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[07:58:46 CST(-0600)] * rburton_ (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[08:36:58 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.66) has joined #atlassiandev
[09:34:50 CST(-0600)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[09:54:31 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[09:55:23 CST(-0600)] <pidan> Anybody know how to set up a mysql connector in a JIRA / Confluence plugin pom.xml such that it gets included in the jar?
[10:02:33 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[10:13:48 CST(-0600)] * bug_ (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[10:24:30 CST(-0600)] * jonmort (~jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[11:22:24 CST(-0600)] * jdumay (~jdumay@59.167.164.33) has joined #atlassiandev
[12:27:55 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[13:49:36 CST(-0600)] * jonmort (~jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[14:37:35 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:50:37 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[15:09:37 CST(-0600)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.cox.net) has joined #atlassiandev
[15:13:15 CST(-0600)] * AJC_Z0 (~nnAJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[15:25:06 CST(-0600)] * sleberrigaud (~sleberrig@59.167.164.33) has joined #atlassiandev
[15:25:16 CST(-0600)] * sebr_ (~sruiz@60-241-117-149.static.tpgi.com.au) has joined #atlassiandev
[16:05:25 CST(-0600)] * tg davies (~tomd@59.167.164.33) has joined #atlassiandev
[16:19:11 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[16:19:41 CST(-0600)] * sleberrigaud (~sleberrig@59.167.164.33) has joined #atlassiandev
[16:20:03 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev
[16:53:18 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[17:22:03 CST(-0600)] * jdumay (~jdumay@59.167.164.33) has joined #atlassiandev
[17:43:45 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:46:26 CST(-0600)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[18:04:55 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[18:51:44 CST(-0600)] * rburton_ (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[19:08:40 CST(-0600)] * MartinCleaver (~martincline@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[19:12:45 CST(-0600)] * rodogu1 (~Adium@d154-5-174-17.bchsia.telus.net) has joined #atlassiandev
[19:28:15 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[19:56:11 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[19:56:28 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[20:34:45 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev

[20:58:20 CST(-0600)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[21:09:00 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has left #atlassiandev
[21:12:25 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:47:00 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[23:05:30 CST(-0600)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[23:05:31 CST(-0600)] * ChanServ sets mode +o bspeakmon
[23:57:35 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev

atlassiandev_log-2010-02-04

[00:33:31 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[00:52:29 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:42:20 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[01:54:34 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[01:56:45 CST(-0600)] * Carlfish_ (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[02:04:55 CST(-0600)] * tg davies (~tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[02:28:40 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[02:32:50 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[03:05:49 CST(-0600)] * jonmort (~jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[03:51:56 CST(-0600)] * imaslovs (~leonidms@80.233.159.254) has left #atlassiandev
[03:52:08 CST(-0600)] * imaslovs (~leonidms@80.233.159.254) has joined #atlassiandev
[05:37:26 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:39:52 CST(-0600)] * Martin Cleaver (~martinle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[08:15:36 CST(-0600)] * jonmort (~jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[08:21:47 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:40:30 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.66) has joined #atlassiandev
[09:20:23 CST(-0600)] <jdoklovic> FYI, studio.plugins does not forward you to the page you were on after login. The login page has the forward in the url, but it always takes you to the dashboard
[09:22:23 CST(-0600)] <jdoklovic> grrr. logged in, went to "Administer Project" and added a new version for JWD, then clicked the issues tab and it took me to old-school jira login page that doesn't work
[10:07:03 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[11:09:40 CST(-0600)] * pidan (~chatzilla@207.96.182.162) has joined #atlassiandev
[12:20:53 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[12:45:30 CST(-0600)] * jnolen (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[13:29:46 CST(-0600)] <jnolen> hello?
[13:48:20 CST(-0600)] * hdk (~hiegdk@h209-17-159-3.gtnconnect.net) has joined #atlassiandev
[13:48:43 CST(-0600)] <hdk> Hi all
[13:49:12 CST(-0600)] <hdk> anyone here ever get NTLM working with Jira4/Crowd?
[13:53:40 CST(-0600)] * tg davies (~tomd@225.38.70.115.static.exetel.com.au) has joined #atlassiandev
[14:28:01 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:30:10 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:25:23 CST(-0600)] * imaslovs2 (imaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[15:47:48 CST(-0600)] * JonathanNolen (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[16:09:17 CST(-0600)] * tg davies (~tomd@59.167.164.33) has joined #atlassiandev
[16:37:46 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev
[16:50:07 CST(-0600)] * leonidms (imaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[17:20:31 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev
[18:22:46 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:40:50 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[18:53:29 CST(-0600)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[18:53:41 CST(-0600)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[18:58:07 CST(-0600)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[20:17:40 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[20:28:06 CST(-0600)] * JonathanNolen (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[20:36:33 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev
[20:37:25 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[21:48:56 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[22:00:57 CST(-0600)] * _5cats (~jd@2002:48cf:2dae:0:225:ff:fe4d:44bc) has joined #atlassiandev
[22:06:23 CST(-0600)] * _5cats (~jd@2002:48cf:2dae:0:225:ff:fe4d:44bc) has left #atlassiandev
[22:09:33 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[22:19:49 CST(-0600)] * dchui (~dchui@202.169.29.34) has left #atlassiandev
[22:20:16 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[22:20:22 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:45:47 CST(-0600)] * myrall_ (~myrall@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:45:47 CST(-0600)] * ChanServ sets mode +o myrall_
[22:45:51 CST(-0600)] * ckiehl1 (~Adium@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:45:54 CST(-0600)] * dezwart_ (~pdzwart@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:45:57 CST(-0600)] * jensschumacher_ (~jens@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:46:11 CST(-0600)] * justin_ (~justin@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:46:33 CST(-0600)] * tg davies_ (~tomd@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:46:33 CST(-0600)] * i386_workies (~jdumay@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:46:35 CST(-0600)] * pleshev (~pleshev@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:46:48 CST(-0600)] * Carlfish_ (~cmiller@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:48:42 CST(-0600)] * mrdon (~mrdon@ATL146140-1.gw.connect.com.au) has joined #atlassiandev
[22:49:51 CST(-0600)] * mrdon_ (~mrdon@59.167.164.33) has joined #atlassiandev
[22:49:54 CST(-0600)] * jdumay (~jdumay@59.167.164.33) has joined #atlassiandev
[22:49:58 CST(-0600)] * ckiehl (~Adium@59.167.164.33) has joined #atlassiandev
[22:50:16 CST(-0600)] * myrall_ (~myrall@59.167.164.33) has joined #atlassiandev
[22:50:16 CST(-0600)] * ChanServ sets mode +o myrall_
[22:50:22 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev

[22:50:24 CST(-0600)] * pleschev_ (~pleschev@59.167.164.33) has joined #atlassiandev
[22:53:12 CST(-0600)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[23:01:57 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[23:05:15 CST(-0600)] <dezwart> nuts
[23:14:03 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[23:18:30 CST(-0600)] <jazzy> /nick fail

[23:44:57 CST(-0600)] <Broady> beers @ the bavarian if anyone wants to say hi 😊
[23:49:40 CST(-0600)] <Broady> onclick="alert('\${i18n.getText('fisheyethis.not.delong.here'))}';"
[23:49:41 CST(-0600)] <Broady> lol
[23:53:00 CST(-0600)] <dezwart> Broady: have you seen the fish?
[23:53:26 CST(-0600)] <Broady> nope! when is it activated?
[23:53:53 CST(-0600)] <Broady> need to find a public jira/fisheyey

atlassiandev_log-2010-02-05

[00:12:57 CST(-0600)] <trochej> Coffee
[00:59:01 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev
[01:03:55 CST(-0600)] * dchui (~dchui@202.169.29.34) has left #atlassiandev
[01:04:36 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[02:00:58 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[02:38:33 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[02:59:21 CST(-0600)] * jonmort (~jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[03:07:43 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[03:33:38 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev

[04:42:13 CST(-0600)] <lmaslovs> Some silly question regarding issue searchers. I'm not sure it's the right place to ask though. 😊
[04:42:16 CST(-0600)] <lmaslovs> I need to add a searcher to search against some daat into lucene index. I didn't find any way to define one except for adding is as custom field searcher. Q: Could I implement calculatable field to display some information and provide it with custom field searcher without indexer but with searcher (clauses, input transforemrns and stuff) only?
[04:43:42 CST(-0600)] <lmaslovs> Would be any consequencues / impacts here?
[07:23:08 CST(-0600)] * cemerick (~la_mer@c-75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[08:41:00 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.66) has joined #atlassiandev
[09:55:06 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[09:58:33 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[10:10:03 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[11:12:56 CST(-0600)] <jdoklovic> anyone know the proper bamboo.version and bamboo.data.version for building a 2.5 plugin?
[11:52:51 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[12:33:50 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:14:48 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[14:18:22 CST(-0600)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has joined #atlassiandev
[14:39:04 CST(-0600)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[14:41:06 CST(-0600)] <rodogu> @lmaslovs I would check if this is kosher for plugins v2, I am doing it Confluence + plugins v1 API
[14:42:49 CST(-0600)] * skeleinei (~stefan@dslb-084-057-136-200.pools.arcor-ip.net) has joined #atlassiandev
[14:55:52 CST(-0600)] * IgorMinar (~Adium@nat/sun/x-qygpjekkdtqwtb) has joined #atlassiandev
[15:22:33 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[15:33:20 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:08:50 CST(-0600)] * MartinCleaver (~martincle@216.191.155.62) has joined #atlassiandev
[17:22:45 CST(-0600)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has joined #atlassiandev
[17:22:55 CST(-0600)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has left #atlassiandev
[17:35:48 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[18:21:51 CST(-0600)] * dezwart (~pdzwart@ppp121-44-87-241.lns20.syd6.internode.on.net) has joined #atlassiandev
[18:47:45 CST(-0600)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[19:10:56 CST(-0600)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[19:16:14 CST(-0600)] <rburton-> re
[19:21:44 CST(-0600)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[19:25:21 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:00:28 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[20:20:39 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[21:02:50 CST(-0600)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[21:05:54 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:27:27 CST(-0600)] * bwright (~bwright@CPE-124-188-64-146.facz1.ken.bigpond.net.au) has joined #atlassiandev
[21:27:31 CST(-0600)] <bwright> Hey.

atlassiandev_log-2010-02-06

[02:40:25 CST(-0600)] * leonidms (~lmaslovs@81.198.49.242) has joined #atlassiandev
[02:48:47 CST(-0600)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[07:19:14 CST(-0600)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[07:56:14 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[08:46:49 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[09:06:35 CST(-0600)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[10:17:41 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[10:56:09 CST(-0600)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[11:26:30 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[12:01:36 CST(-0600)] * bug_ (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[12:23:24 CST(-0600)] * Jilliano (~jdressler@c-98-212-125-118.hsd1.il.comcast.net) has joined #atlassiandev
[13:29:00 CST(-0600)] * MartinCleaver_ (~martincle@206-248-161-74.dsl.teksavvy.com) has joined #atlassiandev
[16:26:22 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

```
[16:59:25 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:18:46 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[18:43:41 CST(-0600)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavy.com) has joined #atlassiandev
[21:18:14 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
```

atlassiandev_log-2010-02-07

```
[23:51:59 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[01:41:50 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[04:31:44 CST(-0600)] * MartinCleaver (~martincle@206-248-161-74.dsl.teksavy.com) has joined #atlassiandev
[07:13:09 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[10:14:39 CST(-0600)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[10:27:15 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavy.com) has joined #atlassiandev
[11:49:42 CST(-0600)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[12:51:10 CST(-0600)] * skeinei (~stefan@dslb-084-057-130-205.pools.arcor-ip.net) has joined #atlassiandev
[12:57:53 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:40:42 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:40:05 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[17:06:30 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[17:23:39 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[17:25:34 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[17:26:37 CST(-0600)] * mryall (~mryall@59.167.164.33) has joined #atlassiandev
[17:26:38 CST(-0600)] * ChanServ sets mode +o mryall
[17:54:00 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[18:19:14 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:49:49 CST(-0600)] <mryall> Caused by: java.lang.LinkageError: loader constraints violated when linking org/apache/commons/logging/Log class
[20:50:19 CST(-0600)] <mryall> ^ anyone seen that in a v2 plugin in Confluence? the plugin bundles commons-logging 1.1.1 and Spring MVC (the latter is causing the error)
[20:53:09 CST(-0600)] <mryall> ah, found it
[20:53:42 CST(-0600)] <mryall> was the difference between "org.apache.commons.logging" and "org.apache.commons.logging*" in the BND instructions - one little asterisk :/
[20:54:51 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:14:33 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[21:41:10 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:37:59 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev
[22:47:41 CST(-0600)] <ckiehl> do you really need the bnd instructions anyway?
[22:57:35 CST(-0600)] <mryall> yeah, auto-imports are evil
[22:58:15 CST(-0600)] <mryall> got Velocity working too now - woot
[22:58:37 CST(-0600)] <mryall> cross-product UI here we come...
[22:58:49 CST(-0600)] <mryall> just need to work out why Sitemesh isn't decorating the response....
[23:11:02 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has left #atlassiandev
[23:22:11 CST(-0600)] <ckiehl> what's so bad about the auto imports?
[23:34:46 CST(-0600)] <mryall> it doesn't really work in my case
[23:35:10 CST(-0600)] <mryall> I have a dependency on Velocity that can't be determined from the code
[23:37:22 CST(-0600)] <jazzy> mryall: make sure you are setting Content-Type to text/html, otherwise sitemesh claims ignorance 😊
[23:37:30 CST(-0600)] <mryall> yeah, discovered that 😊
[23:37:41 CST(-0600)] <mryall> there's a filter in Confluence that sets it above the sitemesh filter
[23:38:03 CST(-0600)] <mryall> so the setContentType() in sitemesh's response wrapper never gets called
[23:38:09 CST(-0600)] <mryall> and content doesn't get decorated
[23:38:28 CST(-0600)] <mryall> and Spring MVC's VelocityView never sets the content type, so I have to munge it into the controller :/
[23:39:14 CST(-0600)] <jazzy> why is there a filter setting the content type? how does the filter know?
[23:39:26 CST(-0600)] <mryall> shrug
[23:39:29 CST(-0600)] <mryall> I think it sets an initial value
[23:39:34 CST(-0600)] <mryall> of text/html
[23:39:34 CST(-0600)] <jazzy> ah...
[23:40:06 CST(-0600)] <mryall> com.atlassian.core.filters.encoding.AbstractEncodingFilter#doFilter
[23:40:17 CST(-0600)] <jazzy> i thought you guys were using GWT?
[23:40:32 CST(-0600)] <Broady> lol ^
[23:40:46 CST(-0600)] <mryall> Broady: not so "lol" - we are using it for a dialog in 3.2
[23:41:02 CST(-0600)] <Broady> just one dialog?
[23:41:05 CST(-0600)] <jazzy> lol
[23:41:07 CST(-0600)] <mryall> I had a look at it with Don for a few hours
[23:41:26 CST(-0600)] <mryall> but the amount of hacks required to get basic form mapping and validation is depressing
[23:41:38 CST(-0600)] <mryall> so I spent some time spiking a Spring MVC plugin
[23:41:49 CST(-0600)] <jazzy> i was surprised that it was ever considered for plugins
[23:41:52 CST(-0600)] <mryall> the work won't start for another couple of weeks, so we'll take another look at the GWT stuff then
[23:42:02 CST(-0600)] <Broady> sounds fun
[23:42:07 CST(-0600)] <mryall> jazzy: which one?
[23:42:28 CST(-0600)] <Broady> i'd like to take a proper look at gwt one day, its all hearsay for me currently
[23:42:38 CST(-0600)] <jazzy> not that i've ever touched GWT, i just didn't think it would ever be a nice thing to get into an osgi environment
[23:42:51 CST(-0600)] <mryall> it wasn't that so much
[23:43:15 CST(-0600)] <mryall> it was the compile-time-generated form binding and validation handlers
[23:43:21 CST(-0600)] <mryall> which is just bizarre
[23:43:25 CST(-0600)] <mryall> because GWT doesn't support reflection
[23:43:26 CST(-0600)] <jazzy> what about struts 2?
[23:43:43 CST(-0600)] <mryall> Spring MVC was thought to be easier because the plugin system already includes 80% of it
[23:44:25 CST(-0600)] <jazzy> i would have thought struts 2 would be a more sensible spike because we have a wealth of knowledge on xwork, not to mention Don, whereas as a company we know nothing about spring MVC and its gotchas
```

[23:44:32 CST(-0600)] <mryall> yeah
[23:44:37 CST(-0600)] <mryall> well, I'd just like to use webwork myself
[23:44:43 CST(-0600)] <mryall> 
[23:44:56 CST(-0600)] <jazzy> webwork and struts 2 are half similar
[23:45:02 CST(-0600)] <mryall> extract common logic into non-action classes, and there will only be a slight amount of difference between JIRA and Confluence
[23:45:08 CST(-0600)] <jazzy> the other half, i really struts for
[23:45:15 CST(-0600)] <Broady> wait so confluence will have spring mvc soon?
[23:45:26 CST(-0600)] <mryall> Broady: a plugin might use it, perhaps
[23:45:55 CST(-0600)] <Broady> how about pluggable content types, is that coming soon?
[23:46:10 CST(-0600)] <mryall> um.. not afaik
[23:46:18 CST(-0600)] <Broady> damn
[23:46:58 CST(-0600)] <jazzy> spring mvc is quite different again... i've used it, but i must admit, i got quite overwhelmed, there are something like 13 different built in action types, and very little documentation that said which one was good for what
[23:47:23 CST(-0600)] * mryall sets mode +o jazzy
[23:47:40 CST(-0600)] * mryall sets mode +o Broady
[23:47:49 CST(-0600)] <jazzy> so, the key to ops is admitting a failing in spring mvc?
[23:47:56 CST(-0600)] <jazzy> wait! he doesn't even work here!
[23:48:02 CST(-0600)] <mryall> nah, just figured we might as well have a couple
[23:48:21 CST(-0600)] <mryall> and I wanted to remember how the /mode command worked
[23:48:25 CST(-0600)] <mryall> stuffed it up, the first time
[23:48:44 CST(-0600)] <mryall> you have to use /mode #chan +oo nick1 nick2
[23:48:47 CST(-0600)] <jazzy> i'm connecting from a server with a tier 1 internet connection, you've just granted me ops permanently
[23:48:51 CST(-0600)] <mryall> i.e. two +o's
[23:49:01 CST(-0600)] <jazzy> in irssi, you just go /op nick1 nick2
[23:49:12 CST(-0600)] <Broady> yaaay irssi
[23:49:22 CST(-0600)] <mryall> haha, it's the freenode outages and netsplits that will deop you, not your own connection 
[23:49:29 CST(-0600)] <Broady> i set up an ircd today for my classmates, good lord they talk a lot
[23:50:08 CST(-0600)] <jazzy> my uni friends and i use ozorg

atlassiandev_log-2010-02-08

[23:50:37 CST(-0600)] <Broady> oh yeah, i had to set up my own because they don't know how to use irc properly, or work firewall blocks them
[23:50:43 CST(-0600)] <Broady> so i set up a qwebirc frontend
[23:50:51 CST(-0600)] <jazzy> ah
[23:51:32 CST(-0600)] <Broady> i registered my nick on ozorg like 3 years ago, and never got the confirmation email
[23:51:51 CST(-0600)] <jazzy> we used to have some people needing to use work firewall, so we'd have servers running a web based irc frontend, and then they'd find that domain would be blocked, so we'd get another dyndns domain name... and so on
[23:52:10 CST(-0600)] <Broady> haha, hosting from home? dynamic ip?
[23:52:18 CST(-0600)] <jazzy> yep
[23:52:21 CST(-0600)] <Broady> nice
[23:53:13 CST(-0600)] <jazzy> that was back then, now most of us have proper servers for that sort of thing, and those that don't have static IPs and their own domain name
[23:53:40 CST(-0600)] <Broady> holy crap. employees here get a 45% discount on AFL membership
[23:54:37 CST(-0600)] * Broady is outie
[00:03:16 CST(-0600)] * mryall is getting JIRA up and going
[00:03:25 CST(-0600)] <mryall> I had completely forgotten about the atlassian-idea-plugin
[00:03:54 CST(-0600)] <mryall> I can't believe how long their instructions are, either. These days, we just check out trunk and open the pom.xml in IDEA
[01:53:08 CST(-0600)] * kalamon (-kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[02:07:09 CST(-0600)] * Carlfish_ (-cmiller@59.167.164.33) has joined #atlassiandev
[02:37:45 CST(-0600)] * jonmort (-jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[03:04:17 CST(-0600)] * skrebs_ (-shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[07:15:55 CST(-0600)] * cemerick (-la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[08:26:13 CST(-0600)] * rburton- (-rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[08:48:35 CST(-0600)] * skeleinei (-stefan@dslb-084-057-130-205.pools.arcor-ip.net) has joined #atlassiandev
[09:14:55 CST(-0600)] * skeleinei (-stefan@dslb-084-057-130-205.pools.arcor-ip.net) has joined #atlassiandev
[09:32:23 CST(-0600)] * rodogu (-Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[09:32:38 CST(-0600)] * bug (-bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:32:46 CST(-0600)] * rodogu (-Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[09:41:46 CST(-0600)] * jdoklovic (-jdoklovic@66.187.202.105) has joined #atlassiandev
[12:52:46 CST(-0600)] * bspeakmon (-bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:52:47 CST(-0600)] * ChanServ sets mode +o bspeakmon
[13:44:22 CST(-0600)] * JonathanNolen (-jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:51:55 CST(-0600)] * MartinCleaver (-martinle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[14:32:38 CST(-0600)] * skrebs (-shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:33:52 CST(-0600)] * imaslovs (-leonidms@80.233.159.254) has left #atlassiandev
[14:34:23 CST(-0600)] * imaslovs (-leonidms@80.233.159.254) has joined #atlassiandev
[14:43:15 CST(-0600)] * bug (-bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[14:49:43 CST(-0600)] * bug (-bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[15:00:30 CST(-0600)] * cemerick (-la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[15:07:27 CST(-0600)] * cemerick (-la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[15:10:03 CST(-0600)] * bug (-bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[16:26:38 CST(-0600)] * tmoore (-tmoore@59.167.164.33) has joined #atlassiandev
[16:26:39 CST(-0600)] * ChanServ sets mode +o tmoore
[16:44:12 CST(-0600)] * Carlfish_ (-cmiller@59.167.164.33) has joined #atlassiandev
[16:55:43 CST(-0600)] * bug (-bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev

[17:02:18 CST(-0600)] <mryall> "The Hyper-Text Transfer Protocol (HTTP) is perhaps the most significant protocol used on the Internet today."
[17:02:23 CST(-0600)] <mryall> "perhaps"? 
[17:02:29 CST(-0600)] <mryall> ^ <http://hc.apache.org/httpcomponents-client/index.html>
[17:04:08 CST(-0600)] <mryall> the only other application that could almost rival the significance of the web is email, but that's split between at least three protocols
[17:08:10 CST(-0600)] <Carlfish> Certainly.
[17:08:27 CST(-0600)] <Carlfish> I would say HTTP is more "significant" than SMTP simply because it does so many different things.
[17:08:48 CST(-0600)] <Carlfish> SMTP is a protocol for sending mail. HTTP has grown into an abstraction layer on top of TCP/IP itself.
[17:09:20 CST(-0600)] <Carlfish> When you were writing a new network app, you used to start with the TCP/IP stack and build a protocol on top of it. Now you start with the HTTP stack.
[17:09:39 CST(-0600)] <Carlfish> Much to the disgust of network admins.
[17:10:23 CST(-0600)] <Broady> how about gopher
[17:10:35 CST(-0600)] <Carlfish> I remember gopher.
[17:10:49 CST(-0600)] <mryall> I always wanted a gopher.
[17:10:59 CST(-0600)] <Carlfish> Anyone remember wiretap.spies.com?
[17:11:13 CST(-0600)] <mryall> Um, don't think so.
[17:11:17 CST(-0600)] <mryall> What's that?
[17:11:36 CST(-0600)] <mryall> You might be able to make a case for DNS, actually.
[17:11:56 CST(-0600)] <Carlfish> Ah, neat. It still exists, it just moved.
[17:12:21 CST(-0600)] <Carlfish> It was a gopher archive of "questionable documents". Government leaks, the terrorists handbook, the MIT lockpick guide etc.
[17:15:55 CST(-0600)] <Carlfish> "Wiretap has been in dozens of books (Planet Internet, Cultural Treasures of the Internet, Yellow Pages, etc), and innumerable newspapers, including the New York Times. It is probably the single useful gopher resource remaining on the Internet."
[17:16:10 CST(-0600)] <Broady> s/useful//
[17:18:18 CST(-0600)] <Broady> anarchists cookbook? that was interesting
[17:53:45 CST(-0600)] <mryall> url of the day: <http://fullof.bs/>
[17:55:09 CST(-0600)] <Broady> <http://boo.bs/>
[18:06:22 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:06:22 CST(-0600)] * ChanServ sets mode +o tmoore
[18:20:54 CST(-0600)] <Broady> hmm what did you teach the students... one of my classmates is now saying "javascript isn't a real language, it's getting replaced by gwt"
[18:21:39 CST(-0600)] <Broady> ahhh i think it was a joke
[18:24:50 CST(-0600)] * hdk (~hiegdk@h209-17-159-3.gtconnect.net) has joined #atlassiandev
[18:26:32 CST(-0600)] <hdk> hi all, having some trouble with crowd integration on Jira 4.0.1 (have it working on Jira 4.0). Crowd logs indicate that my login attempt is succesful against the jira application (in crowd) but Jira says username or password is incorrect. Any ideas?
[18:27:00 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:27:00 CST(-0600)] * ChanServ sets mode +o tmoore
[18:27:30 CST(-0600)] * rburton- (~rburton@c-76-124-12-5.hsd1.nj.comcast.net) has joined #atlassiandev
[18:32:38 CST(-0600)] <hdk> could someone at least tell me how to view/log what jira is getting back from crowd?
[18:34:49 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:42:45 CST(-0600)] <mryall> hdk: this is a developer forum, we don't have anyone here for support
[18:43:00 CST(-0600)] <mryall> you're probably best to open a case on <https://support.atlassian.com>
[18:43:24 CST(-0600)] <hdk> ah, thanks... will do
[19:02:58 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[19:02:58 CST(-0600)] * ChanServ sets mode +o tmoore
[19:11:18 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:29:42 CST(-0600)] * MartinCleaver (~martin@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[20:36:19 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:50:29 CST(-0600)] <mryall> someone brought a saw into the office
[21:50:32 CST(-0600)] <mryall> that's a bit random
[21:50:45 CST(-0600)] <mryall> not a power-saw, and old style push-and-pull one
[21:51:10 CST(-0600)] <mryall> "watch while I cut my female coworker in half!"
[21:51:35 CST(-0600)] <mryall> (fortunately, that last bit hasn't happened yet)
[21:51:42 CST(-0600)] <Carlfish> ...yet?
[21:51:55 CST(-0600)] <jedi> well when it does, let us know
[21:52:06 CST(-0600)] <skrebs> 133t hacksaw?
[21:53:20 CST(-0600)] <Carlfish> Maybe they just decided the desks are too big.
[21:53:25 CST(-0600)] <Carlfish> Or that developers don't need legs to work.
[21:54:30 CST(-0600)] * pleschev (~pleschev@59.167.164.33) has left #atlassiandev
[21:55:09 CST(-0600)] <mryall> well, that's probably true
[21:55:11 CST(-0600)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev
[21:55:15 CST(-0600)] <mryall> it would make it harder for them to leave the office
[21:55:38 CST(-0600)] <Carlfish> It would make us more easily stackable, too.
[21:56:23 CST(-0600)] <mryall> you could get them to wear a helmet and fit their leg stubs with a lego pattern for easy stacking
[21:56:32 CST(-0600)] <Carlfish> Stackable legless cloned reptilian developers
[21:56:35 CST(-0600)] <Carlfish> I see a great need.
[21:56:43 CST(-0600)] <Carlfish> (This is the channel that gets logged, right?)
[21:56:48 CST(-0600)] <mryall> yes
[21:56:52 CST(-0600)] <Carlfish> pwn
[21:57:03 CST(-0600)] <mryall> our secret plan is now out
[21:57:05 CST(-0600)] <Carlfish> OUR REPTILE ARMY IS COMING FOR YOU, GOOGLE!
[21:57:17 CST(-0600)] <mryall> just don't let the wheels fall off, Charles
[21:57:27 CST(-0600)] <Carlfish> If we had no legs, we'd need wheels.
[21:58:15 CST(-0600)] <Broady> also, don't the logs get indexed by connie and thus searchable?
[21:58:30 CST(-0600)] <Carlfish> Yes. If someone searches for "Atlassian reptile army" they'll find this log.
[21:58:34 CST(-0600)] <Carlfish> Otherwise we're probably safe.
[21:59:36 CST(-0600)] <Carlfish> I can haz ops?
[21:59:48 CST(-0600)] * Broady sets mode +o Carlfish
[21:59:52 CST(-0600)] <Carlfish> Ta.

[22:12:56 CST(-0600)] <tmoore> Carlfish: you're registered as a founder w/ ChanServ, too, so you should be able to ask it to op you
[22:14:39 CST(-0600)] <Carlfish> Thanks
[22:18:21 CST(-0600)] <mryall> use your founder powers for good, not evil
[22:19:26 CST(-0600)] <mryall> I always find it funny when I put on headphones at work, and the music is still playing from when I started it yesterday
[22:19:46 CST(-0600)] <mryall> not on repeat either, just working its way through my 20.2 day collection
[22:21:13 CST(-0600)] <mryall> starting from Led Zeppelin yesterday, we're now up to Marco Cerletti (pianist), going by artist
[22:27:25 CST(-0600)] <tmoore> you listen to your music alphabetically?
[22:32:13 CST(-0600)] <mryall> occasionally
[22:32:28 CST(-0600)] <mryall> I just scroll through the list of album by artist until I see something I want to listen to
[22:33:11 CST(-0600)] <mryall> it's pretty random in terms of genre, but you still get to hear albums in their entirety
[22:44:50 CST(-0600)] <Carlfish> I once tried to listen to my entire music library alphabetically by song.
[22:44:55 CST(-0600)] <Carlfish> I think I got up to 'e' before I gave up.
[22:51:12 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

atlassiandev_log-2010-02-09

[23:09:05 CST(-0600)] <Broady> doing it by album would be okay
[23:09:38 CST(-0600)] <Broady> by song would be very eclectic
[23:09:41 CST(-0600)] <mryall> yeah, by song name is a bit insane
[23:10:35 CST(-0600)] <mryall> if you have classical music, you'd end up listening to "Piano Concerto in A", "Piano Concerto in B", etc.
[23:24:11 CST(-0600)] <mryall> okay, I think I'm going to have to skip Mariah Carey's Merry Christmas...
[23:41:39 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:00:53 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev
[01:52:14 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[01:58:28 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[02:27:32 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[02:39:24 CST(-0600)] * jonmort (~jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[02:59:17 CST(-0600)] <trochej> Coffee
[05:34:52 CST(-0600)] * skeleini (~stefan@dslb-084-057-130-205.pools.arcor-ip.net) has joined #atlassiandev
[06:26:54 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[06:33:39 CST(-0600)] * MartinCleaver (~martinicle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[07:39:29 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:57:04 CST(-0600)] * jonmort (~jonmort@cpc1-heme9-2-0-cust774.9-1.cable.virginmedia.com) has joined #atlassiandev
[09:37:44 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.105) has joined #atlassiandev
[09:39:08 CST(-0600)] <jdoklovic> never knew jira had decorator and decorator-mapper plugin modules available... freakin' sweet. now I can finally add drag/drop support to reorder versions
[09:47:56 CST(-0600)] * groove (~ohookins@194.29.233.134) has joined #atlassiandev
[09:48:32 CST(-0600)] <groove> hi, i'm trying to find a workaround for JRA-9979
[09:48:57 CST(-0600)] <groove> i just need to find out from someone in the know, if adding a timestamp column to the notificationinstance table will not cause problems
[10:30:22 CST(-0600)] * groove (~ohookins@194.29.233.134) has left #atlassiandev
[12:10:51 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:18:56 CST(-0600)] <jdoklovic> anyone know why a jira servlet-filter plugin wouldn't run?
[12:19:11 CST(-0600)] <jdoklovic> I have: <servlet-filter
[12:19:11 CST(-0600)] <jdoklovic> name="jwdSendRedirectFilter"
[12:19:11 CST(-0600)] <jdoklovic> key="jwdSendRedirectFilter"
[12:19:11 CST(-0600)] <jdoklovic> class="com.sysbliss.jira.plugins.workflow.servlet.JWDSendRedirectFilter"
[12:19:11 CST(-0600)] <jdoklovic> location="before-dispatch"
[12:19:12 CST(-0600)] <jdoklovic> weight="200"
[12:19:14 CST(-0600)] <jdoklovic> <url-pattern>*</url-pattern>
[12:19:16 CST(-0600)] <jdoklovic> </servlet-filter>
[12:19:41 CST(-0600)] <jdoklovic> the filter logs that it runs (among other things), but it's not running at all
[12:19:50 CST(-0600)] <jdoklovic> i.e. nothing in my logs
[12:33:06 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:09:04 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[13:09:34 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[14:26:38 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[14:32:23 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:37:19 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:14:15 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:14:15 CST(-0600)] * ChanServ sets mode +o tmoore
[16:41:52 CST(-0600)] <mryall> jdoklovic: I think you normally use /* to match everything
[16:42:08 CST(-0600)] <mryall> or *.ext
[16:53:58 CST(-0600)] <jazzy> i would agree with mryall, except (this is a guess) *.ext probably won't work, unless somethings change since I last looked at the matching code... one problem with servlet and servlet-filter URL matching is that there is nothing in th Java spec that says exactly how it should be done, there's only a few mailing list posts that describe how some of the current containers do it
[16:55:01 CST(-0600)] <jazzy> definitely for servlets, *.ext matching was never implemented in atlassian plugins, i'm not sure if it was implemented for servlet filters
[16:55:09 CST(-0600)] <mryall> I'm pretty sure prefix and suffix matching are a de facto standard
[16:55:19 CST(-0600)] <mryall> probably a bug that we don't support them
[16:55:47 CST(-0600)] <jazzy> it doesn't make sense to support them for servlets, because servlets only come under /plugins/servlet
[16:56:04 CST(-0600)] <mryall> yeah, I guess so
[16:56:13 CST(-0600)] <mryall> would be a bit weird if another plugin servlet started handling your requests
[16:56:22 CST(-0600)] <mryall> because it matched *.js or *.css or something
[16:56:40 CST(-0600)] <jazzy> using our current mechanism, (a servlet mounted on /plugins/servlet that then delegates to paths under that), there's no way to match *.ext
[16:57:19 CST(-0600)] <jazzy> the code that does it is horrible, i wouldn't be surprised if there's lots of things that tomcat supports that we don't
[17:02:24 CST(-0600)] * lmaslovs (~lmaslovs@80.233.159.254) has joined #atlassiandev

[17:02:51 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[17:03:50 CST(-0600)] <Carlfish> j0
[17:04:10 CST(-0600)] <Carlfish> Hello Atlasbot. Would you like to play a game?
[17:06:29 CST(-0600)] <jazzy> oh i like games
[17:07:51 CST(-0600)] <jedi> Global Thermonuclear Warfare
[17:08:55 CST(-0600)] <bspeakmon> australia will be a bit outgunned in that one
[17:08:55 CST(-0600)] <Carlfish> Top left.
[17:09:48 CST(-0600)] <jazzy> myall: actually, i reckon *.ext must work for servlet filters, because Don said he got a struts 2 plugin working, the most logical way to get it working in plugins 2 would be to use the struts filter dispatcher mapped on *.action, so if it wasn't already implemented, don probably would have implemented it
[17:10:15 CST(-0600)] <mryall> "there's no way to match *.ext"
[17:10:26 CST(-0600)] <mryall> it would only match *.ext underneath the /plugins/servlet namespace
[17:10:28 CST(-0600)] <mryall> that's implied
[17:10:32 CST(-0600)] <jazzy> yes
[17:10:42 CST(-0600)] <mryall> same as setting the pattern to "/foo/" **currently matches /plugins/servlet/foo/**
[17:11:29 CST(-0600)] <mryall> Don may have just stuck the filter on /*
[17:11:40 CST(-0600)] <mryall> dunno
[17:12:16 CST(-0600)] <jazzy> yeah actually, struts can handle that too
[17:23:25 CST(-0600)] * leonidms (imaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[19:09:44 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:50:34 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev
[19:51:08 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev
[19:54:55 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[20:07:09 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:11:12 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:57:20 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[21:40:56 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[22:10:09 CST(-0600)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev

atlassiandev_log-2010-02-10

[23:20:55 CST(-0600)] * justin____ (~justin@59.167.164.33) has joined #atlassiandev
[00:19:57 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[00:57:03 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:39:31 CST(-0600)] * imaslovs (~leonidms@80.233.159.254) has joined #atlassiandev
[01:43:46 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev
[02:08:57 CST(-0600)] <imaslovs> chat logs is actually a nice thing.
[02:09:22 CST(-0600)] <imaslovs> due a time shift - still able to read responsec / catch up interesting things
[03:46:53 CST(-0600)] * groove (~ohookins@194.29.233.136) has joined #atlassiandev
[03:47:22 CST(-0600)] <groove> hi, i'm trying to find a workaround for JRA-9979
[03:47:29 CST(-0600)] <groove> i just need to find out from someone in the know, if adding a timestamp column to the notificationinstance table will not cause problems
[04:01:22 CST(-0600)] <imaslovs> and by problems you mean - sideeffects?
[04:02:21 CST(-0600)] <groove> yes
[04:02:47 CST(-0600)] <groove> i highly doubt it, but if there were any rogue "select * from notificationinstance" queries it might cause problems
[04:02:59 CST(-0600)] <groove> if the number of columns is unexpected
[04:03:25 CST(-0600)] <imaslovs> Isn't this the same stuff as for all the GVs?
[04:03:40 CST(-0600)] <imaslovs> Generic Values ant Entities in the OFBIZ
[04:04:50 CST(-0600)] <groove> you've lost me
[04:06:40 CST(-0600)] <imaslovs> rephrasing: If you are asking whether is/not raw SQL queries performed against this database table? When I don't know (i'm just casual plugin developer)
[04:07:42 CST(-0600)] <imaslovs> However if all the access is done through OFBIZ access layer (Generic Values and Entities) - then it really shouldn't be a problem (or at least is the same stuff for all other places as well)
[04:08:27 CST(-0600)] <imaslovs> PS. Me is just casual plugin developer, so take it with a lot's salt of irony
[04:10:43 CST(-0600)] <imaslovs> maybe I didn't understand a question
[04:21:56 CST(-0600)] <groove> so in other words, there should be no problems
[04:22:14 CST(-0600)] <groove> there won't be unexpected side effects by having an extra column in that table
[04:32:06 CST(-0600)] <imaslovs> I would expect so (in case there is no places with raw jdbc SQL queries, i believe it's not the case)
[04:32:32 CST(-0600)] <imaslovs> Note: jira itself won't be able to get access to those columns via OFBIZ layer
[04:32:49 CST(-0600)] <imaslovs> until you declare extra column in OFBIZ configuration
[04:33:18 CST(-0600)] <groove> good, that is exactly what i want
[04:33:25 CST(-0600)] <imaslovs> (so you'll stick to raw sql on your own side until you extend ofbiz schema definition)
[04:33:54 CST(-0600)] <groove> all i want is for each new row in notificationinstance to be timestamped

[04:34:01 CST(-0600)] <imaslovs> Please take a look into my PS 😊 I'm not extra expert here.
[04:34:04 CST(-0600)] <groove> then i can clear them all out with a cron job later on
[04:48:31 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[05:09:54 CST(-0600)] * i386|laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[06:22:15 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:38:53 CST(-0600)] * groove (~ohookins@194.29.233.136) has left #atlassiandev
[07:40:05 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:30:04 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.105) has joined #atlassiandev
[09:14:34 CST(-0600)] <jdoklovic> fyi, discovered my jira servlet-filter IS running... just not doing anything
[09:15:06 CST(-0600)] <jdoklovic> granted, i'm trying to do something a little strange
[09:15:55 CST(-0600)] <jdoklovic> I created a servlet-filter that wraps the servlet response in a custom wrapper that overrides the sendRedirect method
[09:16:33 CST(-0600)] <jdoklovic> essentially anytime sendRedirect is called, I want to trap it, add a url param to the redirect url and then send the redirect
[09:17:27 CST(-0600)] <jdoklovic> right now, my wrapped sendRedirect is not being called and i think it might have something to do with the

placement of my filter in the chain, but i'm still investigating

[09:28:38 CST(-0600)] <rodogu> Probably a stupid question but, introducing the SAL into confluence 3.2 should not have any impact on existing plugins as it works on top on the existing Confluence API, right?

[09:33:18 CST(-0600)] * MartinCleaver (~marticle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev

[10:18:13 CST(-0600)] * mtheoryx (~mtheoryx@140-182-65-64.ssl-vpn.iupui.edu) has joined #atlassiandev

[10:31:40 CST(-0600)] <mtheoryx> we use jira and confluence enterprise at my uni, and i'm interested in using it personally. I see the "starter" package. What server requirements are there for running these apps?

[10:31:57 CST(-0600)] <mtheoryx> I can't run them on my shared host, but am looking into a vps from slicehost for running these.

[11:15:29 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev

[11:51:16 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev

[12:14:06 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev

[12:16:27 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev

[12:18:05 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev

[14:13:24 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev

[14:22:07 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

[15:40:41 CST(-0600)] * cyano (~cyano@190.187.56.55) has joined #atlassiandev

[15:40:49 CST(-0600)] <cyano> hi

[15:41:12 CST(-0600)] <cyano> anyone can help me?

[15:41:36 CST(-0600)] <cyano> i wanna open jira with Eclipse... but a can't

[15:42:13 CST(-0600)] <cyano> hello?

[15:42:40 CST(-0600)] <cyano> hi?

[15:50:31 CST(-0600)] * cyano (~cyano@190.187.56.55) has left #atlassiandev

[15:51:43 CST(-0600)] * cyano (~cyano@190.187.56.55) has joined #atlassiandev

[15:58:08 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev

[15:58:08 CST(-0600)] * ChanServ sets mode +o tmoore

[16:45:51 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev

[16:49:08 CST(-0600)] * Carlfish (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev

[16:56:50 CST(-0600)] <mryall> rodogu: you mean dropping the SAL JAR into WEB-INF/lib/?

[17:04:02 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev

[17:08:15 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has left #atlassiandev

[17:08:28 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev

[17:11:52 CST(-0600)] <Carlfish> I love deleting code.

[17:13:15 CST(-0600)] <Carlfish> I just found the remnants of a moinmoin importer Mike Cannon-Brookes started writing five years ago that never got past the "two classes and lots of commented out code" stage.

[17:13:22 CST(-0600)] <Carlfish> [Delete]

[17:18:39 CST(-0600)] <mryall> hehe

[18:52:18 CST(-0600)] <rodogu> @mryall: 3.2 M3 is said to have "Migrated our Sal Plugin into Confluence" so want to make sure that's not going to have impact on my plugins 😊

[18:52:42 CST(-0600)] <mryall> that's an interesting way of putting it

[18:53:06 CST(-0600)] <mryall> I think we shipped it in 3.1, so it shouldn't make any difference at all to anyone, anywhere

[18:53:18 CST(-0600)] <mryall> it's just about where the source code lives, in internal Atlassian land

[18:54:14 CST(-0600)] <rodogu> so that's just an abstraction plugin, isn't it? I think somebody talked about it in AtlasCamp

[18:54:21 CST(-0600)] <tmoore> that's right

[18:55:38 CST(-0600)] <rodogu> Got it,

[18:55:45 CST(-0600)] <mryall> if you use it directly, there may have been some API changes, but I don't think there's anything serious

[18:56:01 CST(-0600)] <mryall> like, if we changed the version which is bundled with Confluence

[18:56:19 CST(-0600)] <rodogu> Any details on "Many common modules were upgraded"? Carlfish?

[18:56:39 CST(-0600)] * rodogu is going through the 3.2-m2 release notes

[18:59:04 CST(-0600)] <mryall> rodogu: that's internal stuff that won't affect most plugins

[18:59:18 CST(-0600)] <mryall> if you want an authoritative list, compare the POM files:

[18:59:49 CST(-0600)] <mryall>

<http://maven.atlassian.com/public/com/atlassian/confluence/confluence-project/3.2-m3/confluence-project-3.2-m3.pom>

[19:00:02 CST(-0600)] <mryall>

<http://maven.atlassian.com/public/com/atlassian/confluence/confluence-project/3.1.1/confluence-project-3.1.1.pom> 😊

[19:00:09 CST(-0600)] <rodogu> "most" plugins 😊

[19:00:39 CST(-0600)] <mryall> well, given we don't have a published API, it depends on which parts you pick to use

[19:01:05 CST(-0600)] <mryall> we won't have changed any important Confluence interfaces in any way that breaks plugins

[19:01:25 CST(-0600)] <mryall> but we might have slightly changed a private or protected method in some class in atlassian-gzip-filter

[19:01:29 CST(-0600)] <mryall> for example

[19:01:47 CST(-0600)] <mryall> by "most", I mean "basically all"

[19:01:50 CST(-0600)] <rodogu> Package names changes is what concerns me, but you guys have been very good on that in the last releases

[19:02:24 CST(-0600)] <mryall> I can't recall any significant package name changes in Confluence that haven't had compatibility measures (e.g. com.atlassian.user.cache -> com.atlassian.cache)

[19:02:32 CST(-0600)] <mryall> which ones can you remember?

[19:03:58 CST(-0600)] <rodogu> There was a funny thing with Lucene (Fieldable?) but that was third party

[19:04:29 CST(-0600)] <mryall> ah right

[19:04:46 CST(-0600)] <mryall> did that break search extractors?

[19:05:01 CST(-0600)] * JonathanNolen (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev

[19:05:15 CST(-0600)] <rodogu> Just when trying to keep the same binaries for different versions

[19:06:17 CST(-0600)] <rodogu> (re: Confluence Plugin Nirvana <http://bit.ly/dkrbDi>)

[19:07:18 CST(-0600)] <rodogu> but the window of supported version is narrowing down (at least for my plugins): next version will support only from 2.10, so soon the Search API roller coaster is just going to be a bad memory 😊

[19:07:26 CST(-0600)] <rodogu> s/version/versions/

[19:07:41 CST(-0600)] <mryall> hm

[19:07:48 CST(-0600)] <mryall> yeah

[19:07:54 CST(-0600)] <mryall> I think from 2.8 onwards is a bit easier

[19:10:08 CST(-0600)] <rodogu> I think I found something funny still in 3.0 (ILuceneConnection) but I know, I know 😊
[19:11:18 CST(-0600)] <Broady> iirc that's new in 3.0
[19:12:12 CST(-0600)] <mryall> yeah, I *think* that was backwards-compatible

[19:12:18 CST(-0600)] <mryall> if not, you can blame me 😊
[19:12:45 CST(-0600)] <Broady> sounds like the stuff dave & i worked on
[19:14:17 CST(-0600)] * rodogu searches in his code...
[19:17:20 CST(-0600)] <rodogu> Oh year: ILuceneConnection.SearcherAction.perform() changed from boolean to void (2.10 -> 3.0)
[19:17:28 CST(-0600)] <rodogu> s/year/yeah/

[19:19:05 CST(-0600)] <rodogu> But nothing that cannot be fixed with some reflection magic 😊
[19:20:17 CST(-0600)] <rodogu> Noticed the upgrade of quartz on 3.2-m3, any particular reason? I have had problems with stalled scheduled jobs...
[19:23:09 CST(-0600)] <ckiehl> @rodogu: we were getting some random NPEs with quartz in our tests. That issue was fixed in the new version.
[19:23:45 CST(-0600)] <rodogu> ok, thnx
[20:19:54 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:34:25 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[20:36:11 CST(-0600)] <mryall> rodogu: there was also a signature change on the writer, I think
[20:36:22 CST(-0600)] <mryall> to remove the parameter that allowed you to leak the writer
[20:36:27 CST(-0600)] <mryall> actually, maybe it was on the searcher
[20:36:38 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:38:51 CST(-0600)] <rodogu> k
[21:12:09 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[21:27:09 CST(-0600)] * Ycros (~ycros@gnaw.yi.org) has joined #atlassiandev
[22:36:09 CST(-0600)] * JonathanNolen (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-02-11

[23:12:26 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[23:43:12 CST(-0600)] * JonathanNolen (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[00:18:01 CST(-0600)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[00:19:33 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:44:03 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:02:48 CST(-0600)] * Carlfish (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[02:03:11 CST(-0600)] * ckiehl (~Adium@59.167.164.33) has joined #atlassiandev
[02:25:11 CST(-0600)] * trochej (trochej@noches.pl) has joined #atlassiandev
[04:56:51 CST(-0600)] * jdumayllaptop (~jdumay@ppp201-143.static.internode.on.net) has joined #atlassiandev
[05:52:50 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:36:31 CST(-0600)] * MartinCleaver (~martinicle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[07:03:09 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:25:49 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:52:29 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[08:36:14 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.105) has joined #atlassiandev
[10:21:54 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[11:09:49 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[11:15:52 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:16:23 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:19:05 CST(-0600)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:41:51 CST(-0600)] * blitzbrau (~blitzbrau@cpe-075-178-183-217.nc.res.rr.com) has joined #atlassiandev
[12:42:55 CST(-0600)] * blitzbrau (~blitzbrau@cpe-075-178-183-217.nc.res.rr.com) has left #atlassiandev
[12:53:10 CST(-0600)] <twong> Sigtrap: isn't that when you have a process in a debugger and set a breakpoint?
[13:00:15 CST(-0600)] * justme (~justme@76.73.16.26) has joined #atlassiandev
[13:00:25 CST(-0600)] <justme> hi folks
[13:01:06 CST(-0600)] <Guest76393> quick question: is there any word on when the plugin dev kit for jira 4 can be expected?
[13:03:59 CST(-0600)] <Guest76393> mhmmm, not that much traffic here
[13:06:24 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[13:14:26 CST(-0600)] <twong> I'd respond. but i have no clue :/
[13:14:45 CST(-0600)] <twong> but i'm sure some of the other people in here might know
[13:18:15 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has left #atlassiandev
[13:18:55 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:25:32 CST(-0600)] <bspeakmon> the new plugin SDK supports jira 4 and is ready for use immediately
[13:28:04 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[13:41:20 CST(-0600)] <jdoklovic> anyone know how to get logging working in felix??? I get this on startup of the atlassian plugin system:
2010-02-11 13:39:43,793 WARN [main] [OsgiBundlePlugin] Unable to enable plugin 'slf4j.log4j12-1.5.10'
[13:41:20 CST(-0600)] <jdoklovic> com.atlassian.plugin.PluginException: org.osgi.framework.BundleException: Fragment bundles can not be started.
[13:42:10 CST(-0600)] <jdoklovic> all because of this in the slf4j-log4j12 manifest: Fragment-Host: slf4j.api
[13:48:12 CST(-0600)] <jdoklovic> duh, maybe i should just use: Atlassian Plugins - OSGi logging framework bundle
[14:29:56 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:35:41 CST(-0600)] <Guest76393> @bspeakmon do you have a download-link by any chance? all i found was 3.1.3
[14:43:09 CST(-0600)] <jdoklovic> @Guest76393: http://confluence.atlassian.com/display/DEVNET/Atlassian+Plugin+SDK
[14:51:20 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[15:30:14 CST(-0600)] <bspeakmon> what jdok said
[15:47:27 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[15:47:27 CST(-0600)] * ChanServ sets mode +o tmoore
[15:54:48 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[15:54:48 CST(-0600)] * ChanServ sets mode +o tmoore
[16:16:06 CST(-0600)] <jdoklovic> hooray for me! I just successfully transformed a doubleclick ASX ad to VAST format using a proxy server !

build on top of the "atlassian-webkit". Now I just have to write plugins for all of the other non-vast ad providers!

[16:16:31 CST(-0600)] <bspeakmon> grats 😊
[16:16:45 CST(-0600)] <jdoklovic> just happy since it's my first app built on all this crap
[16:17:21 CST(-0600)] <bspeakmon> any progress is good progress
[16:17:25 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev
[16:17:31 CST(-0600)] <jdoklovic> maven is a cruel mistress
[16:18:30 CST(-0600)] <bspeakmon> oh, btw, jdok, don and I are going to get your amps patch in next week
[16:18:46 CST(-0600)] <bspeakmon> I'll be in sydney the next two weeks, and it's on the list for week one
[16:19:00 CST(-0600)] <jdoklovic> sweet
[16:19:14 CST(-0600)] <jdoklovic> i've been waiting for that.... it's highly useful
[16:19:49 CST(-0600)] <jdoklovic> btw, do you know if there's a way to specify an alternate war to grab in the amps:run mojo?
[16:20:14 CST(-0600)] <bspeakmon> somebody asked for that... dunno if it's been done, but the will is there

[16:20:37 CST(-0600)] <jdoklovic> great, sounds like I may have to provide the way again 😊
[16:21:11 CST(-0600)] <bspeakmon> please do
[16:21:23 CST(-0600)] <bspeakmon> I have way more shit to do than time to do it in
[16:21:46 CST(-0600)] <jdoklovic> fyi, I'll be providing an updated patch for the bamboo extractDependencies fix tomorrow as well
[16:22:09 CST(-0600)] <bspeakmon> noice

[16:22:21 CST(-0600)] <bspeakmon> send 'em in, and if you don't hear from us, bitch until you do 😊
[16:23:05 CST(-0600)] <jdoklovic> i'm sure you'll hear about them since i have to bug jnolen everytime to create a code review for me
[16:23:12 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[16:36:07 CST(-0600)] <mrdon> morning
[16:37:34 CST(-0600)] * jдумай (~jдумай@59.167.164.33) has joined #atlassiandev
[16:43:28 CST(-0600)] * mttheoryx (~mttheoryx@c-68-58-39-37.hsd1.in.comcast.net) has joined #atlassiandev
[16:54:36 CST(-0600)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:55:15 CST(-0600)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:50:26 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[18:04:47 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[18:26:59 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:27:59 CST(-0600)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:32:29 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:32:29 CST(-0600)] * ChanServ sets mode +o tmoore
[18:55:54 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[19:02:11 CST(-0600)] * jensschumacher3 (~jensschum@59.167.164.33) has joined #atlassiandev
[19:11:22 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[19:37:11 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:38:52 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[19:38:55 CST(-0600)] * ChanServ sets mode +o tmoore
[19:58:22 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[20:05:08 CST(-0600)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[20:31:11 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[20:38:51 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[20:40:30 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[21:01:41 CST(-0600)] * bug (~bug@72-254-88-78.client.stsn.net) has joined #atlassiandev
[21:20:19 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[21:28:42 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[22:51:17 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev

atlassiandev_log-2010-02-12

[23:51:31 CST(-0600)] * bug (~bug@72-254-60-56.client.stsn.net) has joined #atlassiandev
[00:45:22 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[00:51:18 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has left #atlassiandev
[00:51:25 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[01:19:18 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[01:31:08 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[03:36:20 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[03:43:06 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[04:01:23 CST(-0600)] * gmcDonald (~gmcDonald@apache/committer/gmcDonald) has joined #atlassiandev
[04:29:04 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[04:30:07 CST(-0600)] * sebr (~sruiz@60-241-117-149.static.tpgi.com.au) has joined #atlassiandev
[04:34:14 CST(-0600)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[04:38:46 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[04:39:36 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[04:55:06 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[06:00:58 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:37:04 CST(-0600)] * i386|laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[07:29:14 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:31:19 CST(-0600)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[07:38:19 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[08:16:39 CST(-0600)] * bug (~bug@72-254-87-166.client.stsn.net) has joined #atlassiandev
[09:00:05 CST(-0600)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[09:00:05 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:23:25 CST(-0600)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[09:34:46 CST(-0600)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[09:36:35 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.105) has joined #atlassiandev
[10:18:13 CST(-0600)] <lmaslovs> Hi guys! Does anybody have a clue how to enable jRebel support for JIRA plugins v2? (to speedup redeployment process with custom field modules)

[10:24:15 CST(-0600)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[10:45:30 CST(-0600)] * AJC_Z0 (~nnAJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[10:57:12 CST(-0600)] <jdoklovic> @lmaslovs I don't think you can. jRebel does class instrumentation and uses classloader extensions. In plugins V2 (aka osgi) you have so many classloaders it's crazy. One thing to check though is jRebel included support for glassfish (and supposedly it's embedded felix osgi container) in version 2.1
[10:58:11 CST(-0600)] <jdoklovic> you might find something in the docs that might help, but i have a feeling you'll spend a week getting it working. Why not just use amps and the pi command?
[10:59:45 CST(-0600)] <jdoklovic> the other thing to keep in mind is that when using amps, only java code needs to be redeployed (using pi). If you're making velocity template changes or something, you should see them immediately in the running app
[11:08:04 CST(-0600)] <lmaslovs> atlas-cli , pi works perfectly fine
[11:08:05 CST(-0600)] * leonidms (~lmaslovs@80.233.159.254) has joined #atlassiandev
[11:08:25 CST(-0600)] <lmaslovs> until you are making something like custom field plugin
[11:08:48 CST(-0600)] <lmaslovs> which still requires the whole JIRA to restart to redeploy plugin
[11:09:23 CST(-0600)] <lmaslovs> whole restart cycle is up to 7-10 minutes for me
[11:09:30 CST(-0600)] <lmaslovs> it's very annoying
[11:13:01 CST(-0600)] * JonathanNolen_ (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[11:31:58 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:45:14 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[11:55:48 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.105) has joined #atlassiandev
[12:40:56 CST(-0600)] <jdoklovic> @lmas sounds like your saying JIRA scans for customfield plugins at boot time and then caches them. can anyone verify this?
[12:48:22 CST(-0600)] <bspeakmon> hm
[12:49:01 CST(-0600)] <bspeakmon> I don't think custom fields are treated any differently from the other modules
[12:55:34 CST(-0600)] <bspeakmon> lmaslovs: file a bug at <https://studio.atlassian.com/browse/AMPS> and I'll take a look
[13:05:35 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[13:08:05 CST(-0600)] * bug_ (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[13:25:49 CST(-0600)] <bspeakmon> jdoklovic: do me a favor? complete <https://studio.atlassian.com/source/cru/CR-AMPS-43> and I'll get it in
[13:40:01 CST(-0600)] * MartinCleaver (~martincle@66-207-222-14.beanfield.net) has joined #atlassiandev
[13:43:11 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[13:43:20 CST(-0600)] <leonidms> i don't believe it's a bug
[13:43:55 CST(-0600)] <jdoklovic> @ben completed
[13:44:07 CST(-0600)] <leonidms> as far i understood - those modules marked as @RestartRequired - just require the whole application to restart
[13:44:27 CST(-0600)] <leonidms> so .. no support for reloadable custom fields at the moment
[13:44:40 CST(-0600)] <leonidms> should i still file it as a bug?
[13:45:04 CST(-0600)] <bspeakmon> file it as a jira bug, since it's not the sdk's fault
[13:45:46 CST(-0600)] <jdoklovic> is your plugin using @RestartRequired or is it some atlassian base class?
[13:47:02 CST(-0600)] <leonidms> @jdoklovic - no... i use atlassian-plugin.xml to define JIRA custom field and custom field searcher
[13:47:26 CST(-0600)] <leonidms> it happens to be that those elements are transformed at plugin module descriptors in the runtime ...
[13:47:37 CST(-0600)] <leonidms> abd they are marked by that annotation
[13:48:17 CST(-0600)] <leonidms> the sdk does not reload plugins that contains modules with that kind of annotation
[13:48:27 CST(-0600)] <bspeakmon> at the least it should log that
[13:48:36 CST(-0600)] <leonidms> @bspeakmon - ok, i'll file that
[13:48:37 CST(-0600)] <bspeakmon> (assuming the sdk realizes that
[13:48:38 CST(-0600)] <leonidms> mmm
[13:48:38 CST(-0600)] <bspeakmon> ()
[13:48:51 CST(-0600)] <leonidms> i think it says something about that ...
[13:48:53 CST(-0600)] <bspeakmon> either the sdk or plugin system should log it
[13:48:58 CST(-0600)] <leonidms> something like - could not reinstall
[13:49:23 CST(-0600)] <leonidms> it's logged but i don;t know a man who understand that it's telling 
[13:49:49 CST(-0600)] <leonidms> here are 3-5 questions about it in the JIRA dev forum 
[13:50:14 CST(-0600)] <bspeakmon> what's the log msg? (is it in one of the forum posts?)
[13:50:28 CST(-0600)] <leonidms> i'll see ... 1 min
[13:52:10 CST(-0600)] <leonidms> <http://forums.atlassian.com/thread.jspa?messageID=257328570> for example
[13:52:38 CST(-0600)] <leonidms> WARNING talledLocalContainer 10-Jan-2010 11:02:07
com.atlassian.plugin.loaders.ScanningPluginLoader addFoundPlugins
[13:52:39 CST(-0600)] <leonidms> WARNING talledLocalContainer INFO: No plugins found to be installed
[13:52:53 CST(-0600)] <leonidms> no one has a clue that does it mean 
[13:53:09 CST(-0600)] <leonidms> i ended up with 2-3h debugging session to find that
[13:53:53 CST(-0600)] <leonidms> should i still file it? as JIRA issue?
[13:55:41 CST(-0600)] <bspeakmon> hm.
[13:56:07 CST(-0600)] <bspeakmon> there's two problems: jira can't reload custom fields, and the plugin system doesn't explain why a plugin couldn't be installed in that case
[13:56:34 CST(-0600)] <leonidms> exactly
[13:56:34 CST(-0600)] <bspeakmon> so file two issues, one in jira and the other in plugins (<https://studio.atlassian.com/browse/PLUG>)
[13:56:48 CST(-0600)] <bspeakmon> or I can do the plugins one if you want. 
[13:57:11 CST(-0600)] <bspeakmon> willing to buy the argument that plugin system bugs are our fault, not yours 
[13:57:22 CST(-0600)] <bspeakmon> s/fault/problem/
[13:57:44 CST(-0600)] <leonidms> fair enough... thanks a lot .. i believe all the jira plugin developers will appreciate that
[14:00:35 CST(-0600)] <bspeakmon> no worries
[14:00:43 CST(-0600)] <bspeakmon> wondering if it belongs in the SDK known issues as well
[14:02:48 CST(-0600)] <leonidms> no mention in the official documentation
[14:11:54 CST(-0600)] <leonidms> @bspeakmon done, PLUG-532
[14:16:23 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[17:06:45 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[17:17:15 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[17:51:48 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev

[18:13:25 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:29:26 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:20:05 CST(-0600)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:47:18 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[20:04:11 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[20:28:29 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:38:47 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[20:54:13 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[21:49:06 CST(-0600)] * snerd (~snerdlet@fedora/robk) has joined #atlassiandev
[21:49:13 CST(-0600)] <snerd> howdy people
[21:49:22 CST(-0600)] <snerd> is the plugin repository under maintenance?
[21:49:39 CST(-0600)] <snerd> I'm getting a pki error when I go to refresh the plugin cache
[21:49:43 CST(-0600)] <snerd> clean install of confluence 3.1.1
[21:50:12 CST(-0600)] <snerd> darn spherical earth
[21:58:30 CST(-0600)] <snerd> huh, plugins.atlassian.com - looks like that change has broken the out-of-the-box plugin repository manager
[21:58:33 CST(-0600)] <snerd> that's kind of sucky
[21:58:33 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[22:00:02 CST(-0600)] <snerd> well, bum
[22:47:15 CST(-0600)] * tmoore (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[22:47:15 CST(-0600)] * ChanServ sets mode +o tmoore
[22:59:37 CST(-0600)] * eroussel (~eroussel@dsl-145-204.aei.ca) has joined #atlassiandev
[23:00:47 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-02-13

[23:38:05 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[23:41:59 CST(-0600)] * tmoore (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[23:41:59 CST(-0600)] * ChanServ sets mode +o tmoore
[23:53:20 CST(-0600)] * tmoore (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[23:53:20 CST(-0600)] * ChanServ sets mode +o tmoore
[00:00:56 CST(-0600)] <snerd> plugin repository is down in case no-one has noticed
[00:05:34 CST(-0600)] <Broady> up but slow for me
[00:06:33 CST(-0600)] <Broady> \$ time curl -I <https://plugins.atlassian.com/plugin/home> 2>/dev/null | grep 200
[00:06:36 CST(-0600)] <Broady> HTTP/1.1 200 OK
[00:06:38 CST(-0600)] <Broady> real 0m4.600s
[00:22:07 CST(-0600)] <snerd> Error downloading:
<http://confluence.atlassian.com/plugin-repository/proxy.action?profile=confluence&decorator=none&buildNumber=1724&repoC>
[00:22:07 CST(-0600)] <snerd> sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
[00:22:18 CST(-0600)] <snerd> vanilla 3.1.1 install, straight out of the box
[00:23:14 CST(-0600)] <snerd> pki error?
[00:25:46 CST(-0600)] * tmoore (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[00:25:47 CST(-0600)] * ChanServ sets mode +o tmoore
[00:48:15 CST(-0600)] * tmoore (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[00:48:15 CST(-0600)] * ChanServ sets mode +o tmoore
[01:11:53 CST(-0600)] * tmoore1 (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[01:14:23 CST(-0600)] * ChanServ sets mode +o tmoore
[02:38:46 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[03:32:46 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[03:33:38 CST(-0600)] * leonidms test
[04:09:09 CST(-0600)] * tmoore (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[04:09:09 CST(-0600)] * ChanServ sets mode +o tmoore
[04:47:13 CST(-0600)] * tmoore (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[04:47:13 CST(-0600)] * ChanServ sets mode +o tmoore
[04:55:01 CST(-0600)] <snerd> bah 
[05:15:07 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[05:30:00 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[05:39:31 CST(-0600)] * tmoore (~tmoore@122-149-238-170.static.dsl.dodo.com.au) has joined #atlassiandev
[05:39:32 CST(-0600)] * ChanServ sets mode +o tmoore
[06:25:53 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[06:31:56 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[06:34:53 CST(-0600)] * roxcorp (~user@5ad3ea58.bb.sky.com) has joined #atlassiandev
[06:58:02 CST(-0600)] * i386|laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[08:25:47 CST(-0600)] * leonidms (lmaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[10:11:45 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[10:14:08 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[10:26:02 CST(-0600)] * mtheoryx (~mtheoryx@c-68-58-39-37.hsd1.in.comcast.net) has joined #atlassiandev
[11:22:24 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[11:39:25 CST(-0600)] * bug (~bug@adsl-75-62-236-192.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[12:19:04 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[13:33:46 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:09:12 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:21:45 CST(-0600)] * Carlfish (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[16:22:38 CST(-0600)] * Carlfish (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[16:38:54 CST(-0600)] * bug (~bug@adsl-76-192-51-53.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[16:51:30 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[17:43:56 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[17:46:13 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

```
[18:17:07 CST(-0600)] * i386|laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[19:34:56 CST(-0600)] * jensschumacher (~jens@59.167.164.33) has joined #atlassiandev
[19:37:09 CST(-0600)] * i386|laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[20:40:55 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:44:33 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:47:30 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
```

atlassiandev_log-2010-02-14

```
[00:23:55 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:51:34 CST(-0600)] * Carlfish (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[03:26:20 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[06:02:57 CST(-0600)] * leonidms (imaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[07:26:18 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:45:28 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[08:45:43 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:33:03 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[09:51:14 CST(-0600)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[09:51:18 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[10:05:41 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[11:09:03 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[12:13:41 CST(-0600)] * leonidms (imaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[12:28:32 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[12:37:16 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[12:50:39 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[12:56:21 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[14:14:44 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[16:44:25 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[16:44:31 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[17:01:43 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[17:41:37 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[18:42:12 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[18:44:43 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:46:21 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[19:00:58 CST(-0600)] * skehrs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[20:15:36 CST(-0600)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[20:22:04 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[20:22:16 CST(-0600)] * ChanServ sets mode +o bspeakmon
[20:29:16 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[20:39:56 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[20:41:26 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[20:51:58 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:42:23 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
```

atlassiandev_log-2010-02-15

```
[23:11:24 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[00:05:32 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[00:21:46 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[00:27:22 CST(-0600)] * leonidms (imaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[00:55:33 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[01:25:16 CST(-0600)] * dezwart (~pdzwart@ppp121-44-108-216.ins20.syd6.internode.on.net) has joined #atlassiandev
[01:37:41 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[02:43:28 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btccentralplus.com) has joined #atlassiandev
[04:47:22 CST(-0600)] <snerd> hrm, busy channel
[06:08:17 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:04:37 CST(-0600)] * slooe (~ecyr@2002:59ee:d6b::59ee:d6b) has joined #atlassiandev
[07:05:31 CST(-0600)] <slooe> http://xpango.com.pl/
[07:12:21 CST(-0600)] * slooe (~ecyr@2002:59ee:d6b::59ee:d6b) has left #atlassiandev
[07:29:01 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:41:20 CST(-0600)] <Ycros> that looks dodgy
[07:43:55 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[08:47:28 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:49:15 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[09:04:48 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:33:35 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[09:40:33 CST(-0600)] * leonidms (imaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[09:54:32 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[10:36:22 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[10:42:37 CST(-0600)] * jkr801 (~jkr801@dsl093-228-008.slc1.dsl.speakeasy.net) has joined #atlassiandev
[11:27:10 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[13:50:46 CST(-0600)] * leonidms (~imaslovs@80.233.159.254) has joined #atlassiandev
[13:55:20 CST(-0600)] * leonidms (imaslovs@balticom-187-40.balticom.lv) has joined #atlassiandev
[14:13:02 CST(-0600)] * bug (~bug@65-112-21-194.dia.static.qwest.net) has joined #atlassiandev
[14:15:33 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btccentralplus.com) has joined #atlassiandev
[14:37:21 CST(-0600)] * skehrs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:06:06 CST(-0600)] * bug (~bug@65-112-21-194.dia.static.qwest.net) has joined #atlassiandev
[15:08:17 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[16:11:22 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
```

[16:11:39 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[16:15:56 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:15:56 CST(-0600)] * ChanServ sets mode +o tmoore
[16:28:21 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[16:28:21 CST(-0600)] * ChanServ sets mode +o bspeakmon
[16:36:58 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[16:45:02 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[16:45:24 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[17:19:49 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[17:20:25 CST(-0600)] * Carlfish (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[17:21:48 CST(-0600)] * jkr801 (~jkr801@c-67-164-200-10.hsd1.ut.comcast.net) has joined #atlassiandev
[17:24:32 CST(-0600)] * ChanServ sets mode +o bspeakmon
[17:38:20 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[18:02:42 CST(-0600)] * bug (~bug@75-101-91-24.dsl2dy.lmi.net) has joined #atlassiandev
[18:19:44 CST(-0600)] * ChanServ sets mode +o Carlfish
[18:20:08 CST(-0600)] * Topic is 'LF32M. pst nerdscore and achievements | This channel is being logged with transcripts available at http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#O39; set by Carlfish on 2010-02-15 18:20:08 CST(-0600)
[18:24:48 CST(-0600)] * bug (~bug@75-101-91-24.dsl2dy.lmi.net) has joined #atlassiandev
[18:48:24 CST(-0600)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[20:04:47 CST(-0600)] * gmcnold (~gmcnold@apache/committer/gmcnold) has left #atlassiandev
[20:10:09 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[20:22:18 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:25:18 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[20:25:19 CST(-0600)] * ChanServ sets mode +o bspeakmon
[21:15:03 CST(-0600)] <Carlfish> Quiet day.
[21:18:22 CST(-0600)] <jdumay> tis governor 
[21:18:43 CST(-0600)] <i386> struggling with some elusive functional test 
[21:19:17 CST(-0600)] * Carlfish is downloading the internet after having trashed his .m2 directory. Whee.
[21:19:29 CST(-0600)] <i386> oh I trashed ~/Library on friday
[21:19:33 CST(-0600)] <i386> by accident
[21:19:47 CST(-0600)] <bspeakmon> ssshh
[21:19:48 CST(-0600)] <bspeakmon> holiday in the US today
[21:19:57 CST(-0600)] <i386> Setting up IDEA, email, adium, etc is a PITA
[21:20:04 CST(-0600)] <i386> bspeakmon: what holiday?
[21:20:16 CST(-0600)] <bspeakmon> President's Day
[21:20:25 CST(-0600)] <i386> but hes a nazi!!! 
[21:20:27 CST(-0600)] <i386> 
[21:20:28 CST(-0600)] <bspeakmon> it's over now, of course
[21:20:36 CST(-0600)] <bspeakmon> eh?
[21:20:52 CST(-0600)] <Carlfish> i386: is there some kind of Godwin's law related to maven?
[21:21:27 CST(-0600)] <i386> Carlfish: yes, when someone pulls the "Its the Maven Way" card in the conversation
[21:21:45 CST(-0600)] <i386> bspeakmon: referring to teabaggers
[21:22:06 CST(-0600)] <bspeakmon> ah
[21:22:25 CST(-0600)] <bspeakmon> I thought the preferred nomenclature was "commie bastard" with them
[21:22:26 CST(-0600)] <Carlfish> Let's get some maven trash talk into the IRC logs.
[21:23:07 CST(-0600)] <i386> bspeakmon: btw, enjoy your new health care system
[21:23:14 CST(-0600)] <i386> ours has been awesome thus far
[21:25:07 CST(-0600)] <i386> Carlfish: sir, if you would start please
[21:25:11 CST(-0600)] <Carlfish> Well, when we're not reclassifying patients randomly to make our waiting list numbers look better.
[21:25:26 CST(-0600)] <i386> Carlfish: this is true
[21:26:24 CST(-0600)] <i386> although if your sick and need to see a GP you can get an appointment on the same day
[21:27:03 CST(-0600)] <Carlfish> Don't get me wrong. I like our system better than the alternatives.
[21:27:20 CST(-0600)] <i386> and its free*
[21:27:20 CST(-0600)] <Carlfish> It's just when your brother reports on what's wrong with the industry for a living, you tend to get a skewed view of it. 
[21:27:33 CST(-0600)] <i386> agreed
[21:27:41 CST(-0600)] <i386> ahh yes
[21:27:46 CST(-0600)] <i386> I always forget about your bro
[21:27:59 CST(-0600)] <i386> Carlfish: so im going to buy n ssd today
[21:28:02 CST(-0600)] <i386> an*
[21:28:27 CST(-0600)] <Carlfish> I did my bonus splurge on the weekend. I'm waiting for my letter of thanks from Steve Jobs.
[21:28:57 CST(-0600)] <Carlfish> 27" quad-core iMac. Whee.
[21:29:06 CST(-0600)] <Broady> nice work
[21:29:12 CST(-0600)] <Broady> core i7?
[21:29:39 CST(-0600)] <Carlfish> i5
[21:29:46 CST(-0600)] <Broady> also... can someone fill me in about refinedwiki? is it just me or is it a very lightly modded confluence
[21:29:47 CST(-0600)] <i386> HOT
[21:29:52 CST(-0600)] <i386> when does it get delivered ?
[21:30:07 CST(-0600)] <Carlfish> i386: Three days ago.
[21:30:17 CST(-0600)] <i386> is it nice?
[21:30:21 CST(-0600)] <Carlfish> I walked into the Apple Store, pointed and said "I want that one."
[21:30:28 CST(-0600)] <i386> :O
[21:30:29 CST(-0600)] <Carlfish> Very.
[21:30:31 CST(-0600)] <i386> want
[21:30:38 CST(-0600)] <i386> how much did you end up paying for it?
[21:30:52 CST(-0600)] <Carlfish> AU\$2.6k
[21:31:02 CST(-0600)] <Broady> <http://images.quizfarm.com/1105227604andy.jpg>
[21:31:09 CST(-0600)] <Carlfish> I got applecare for nine bucks because they were price-matching Dick Smiths.

[21:31:37 CST(-0600)] <Carlfish> It doesn't make maven any faster, but IDEA is a lot more pleasant.
[21:31:49 CST(-0600)] <Carlfish> And, more importantly, World of Warcraft. Obviously.
[21:50:51 CST(-0600)] <bspeakmon> you'll want a top of the line machine for wow
[21:51:12 CST(-0600)] <Carlfish> Absolutely.
[21:51:30 CST(-0600)] <bspeakmon> also, I will hear exactly zero complaints from aussies about health care
[21:51:56 CST(-0600)] <Carlfish> We complain about the implementation, not the architecture.
[21:53:13 CST(-0600)] <Carlfish> There was a funny thread on reddit a while back about australian healthcare. There were heaps of comments from Aussies saying "Yeah, our system is great you guys should stop knocking it", then about three pages down one comment from an American saying "That system could never work!"
[21:53:39 CST(-0600)] <bspeakmon> if there's one thing we know, it's how to make shitty healthcare
[21:53:46 CST(-0600)] <bspeakmon> and bombs
[21:53:50 CST(-0600)] <bspeakmon> we fucking rule at bombs
[22:01:36 CST(-0600)] <tmoore> interesting <http://java.dzone.com/news/git-forgoes-backward>
[22:03:46 CST(-0600)] <tmoore> actually, the headline overstates it
[22:17:10 CST(-0600)] <i386> oh idea
[22:17:28 CST(-0600)] <i386> How I love you when you freeze on me
[22:17:49 CST(-0600)] <Broady> i thought you had an idea to share with us

[22:17:49 CST(-0600)] <Broady> 😊 😕

[22:18:02 CST(-0600)] <i386> nope 😕
[22:18:15 CST(-0600)] <i386> All my ideas are property of Atlassian PTY LTD
[22:18:18 CST(-0600)] <i386> and are not for sharing
[22:18:41 CST(-0600)] <bspeakmon> take that, open company
[22:28:46 CST(-0600)] <i386> lol
[22:31:39 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:52:54 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[22:53:03 CST(-0600)] * ChanServ sets mode +o bspeakmon
[22:55:09 CST(-0600)] <bspeakmon> dear makers of star trek movies: stop using the goddamn Romulans as villains.
[23:04:50 CST(-0600)] <jedi> bring back Q
[23:05:57 CST(-0600)] <Carlfish> Or give him a brother, so it could be QQ.
[23:06:26 CST(-0600)] <jedi> lol!
[23:07:11 CST(-0600)] <Carlfish> Problem is they don't really have that many villains to work with.
[23:07:27 CST(-0600)] <bspeakmon> how about the klingons? remember those?
[23:07:32 CST(-0600)] <Carlfish> The Klingons were rendered boring, the Borg is a one-note.
[23:08:05 CST(-0600)] <Carlfish> Q is fun but not really villain-y.
[23:12:13 CST(-0600)] <i386> and dont get me started with that Cheers in Space Deep Space Nine
[23:12:15 CST(-0600)] <i386> that was balls
[23:14:49 CST(-0600)] * leonardinius (~leonidms@80.233.159.254) has joined #atlassiandev
[23:15:37 CST(-0600)] <jedi> Enterprise was pretty good
[23:16:32 CST(-0600)] <Carlfish> Enterprise shat all over continuity.
[23:17:43 CST(-0600)] <jedi> i'm sure it was just a similar parallel universe
[23:18:18 CST(-0600)] <Carlfish> "Hey look! Ferengi! Ignore the fact that they were introduced as totally new in TNG, we're too lazy to come up with new races and we've still got a truck full of ear-sex jokes."
[23:18:30 CST(-0600)] <Carlfish> ---- NERRRRRRRRRRRRRRRRRR
[23:18:44 CST(-0600)] <i386> Dear computer shop who im going to fork a lot of money out to: try deleting email from your inbox so you can receive mail and orders, Love me
[23:19:51 CST(-0600)] <i386> and their line is busy
[23:20:01 CST(-0600)] <i386> Hell is other people
[23:22:05 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[23:22:06 CST(-0600)] * ChanServ sets mode +o bspeakmon
[23:22:48 CST(-0600)] <bspeakmon> the wireless here sucks
[23:22:52 CST(-0600)] <bspeakmon> drops every 20 minutes
[23:22:59 CST(-0600)] <i386> bspeakmon: which part of the office are you in?
[23:23:07 CST(-0600)] <bspeakmon> sitting with conf guys
[23:23:17 CST(-0600)] <bspeakmon> but was acting up in integration land yesterday
[23:23:29 CST(-0600)] <i386> bspeakmon: wonder over to IT and ask for a ethernet port
[23:23:32 CST(-0600)] <i386> wander *
[23:24:01 CST(-0600)] <bspeakmon> ethernet? what is this, 1997?

[23:24:50 CST(-0600)] <i386> or you could bitch on EAC about it 😊
[23:25:07 CST(-0600)] <bspeakmon> it IS the sydney way
[23:25:21 CST(-0600)] <bspeakmon> I'm learning fast
[23:26:07 CST(-0600)] <i386> haha
[23:28:22 CST(-0600)] <bspeakmon> HAHA, fixed it
[23:28:27 CST(-0600)] <bspeakmon> conquered groovy AND maven
[23:32:50 CST(-0600)] <bspeakmon> why don't maven plugins fail more often on config errors?
[23:37:20 CST(-0600)] <i386> why does water feel wet?
[23:41:32 CST(-0600)] <bspeakmon> because it wants to fuck with you?
[23:43:56 CST(-0600)] <bspeakmon> oh shit, I'm in the southern hemisphere
[23:48:06 CST(-0600)] <mrdon> Carlfish: and don't get me started on how they did Klingons
[23:48:23 CST(-0600)] <mrdon> I was at a trek convention and some writer or whatever was there
[23:48:23 CST(-0600)] <Carlfish> Who's been doing Klingons?
[23:48:42 CST(-0600)] <mrdon> they said they can do what they want in enterprise because of some time event happened in the first episode
[23:48:48 CST(-0600)] <mrdon> so from then on, it is a new universe
[23:48:57 CST(-0600)] <mrdon> what Enterprise did with Klingons
[23:49:00 CST(-0600)] <mrdon> and their forehead
[23:49:11 CST(-0600)] <mrdon> worf said those were because of some event he wouldn't talk about
[23:50:44 CST(-0600)] <bspeakmon> all the ST fans are coming out of the woodwork now
[23:50:57 CST(-0600)] <bspeakmon> when did ST become so uncool?

[23:51:05 CST(-0600)] <bspeakmon> (I mean, uncool among nerds)
[23:51:05 CST(-0600)] <mrdon> wait - it was ever cool?
[23:51:22 CST(-0600)] <mrdon> nerds just aren't what they used to be
[23:52:18 CST(-0600)] <Carlfish> Star Trek has always been cringeworthy.
[23:52:38 CST(-0600)] <bspeakmon> there was a distinct hierarchy: Xena fans were looked down on by ST fans, who were looked down on by B5 fans, who were looked down on by hard sci-fi readers
[23:53:06 CST(-0600)] <mrdon> Carlfish: no where near as much as star wars /ducks
[23:53:10 CST(-0600)] <bspeakmon> then at some point the SG / BSG fanbases got mixed in 😕
[23:53:18 CST(-0600)] <bspeakmon> and SW too, I forgot 😕
[23:53:37 CST(-0600)] <bspeakmon> with the result being that ST got shoved way down the list of shit you geek out about in front of others
[23:53:37 CST(-0600)] <mrdon> also BSG > SG
[23:53:41 CST(-0600)] <bspeakmon> truth
[23:53:53 CST(-0600)] <Carlfish> mrdon: Indeed. http://www.salon.com/ent/movies/feature/1999/06/15/brin_main/
[23:56:30 CST(-0600)] <Carlfish> Did everyone watch that long video review of Phantom Menace?
[23:57:05 CST(-0600)] <mrdon> nope
[23:57:10 CST(-0600)] <Carlfish>
<http://www.slashfilm.com/2009/12/17/watch-this-70-minute-video-review-of-star-wars-the-phantom-menace/>
[23:57:11 CST(-0600)] <mrdon> I try to forget that movie

atlassiandev_log-2010-02-16

[23:57:18 CST(-0600)] <mrdon> and I'm a huge star wars book fan
[23:57:21 CST(-0600)] <Carlfish> The voice is annoying but you get used to it.
[23:57:56 CST(-0600)] <bspeakmon> the review is epic
[23:58:02 CST(-0600)] <bspeakmon> I'm a big fan of his ST reviews too
[23:58:10 CST(-0600)] <Carlfish> Yeah, I discovered those afterwards.
[23:58:38 CST(-0600)] <bspeakmon> what's wrong with your faaaaace?
[00:08:08 CST(-0600)] <bspeakmon> also, sydney, where should I get dinner?
[00:08:25 CST(-0600)] <Carlfish> What are you looking for?
[00:09:14 CST(-0600)] <bspeakmon> something good a foreigner wouldn't necessarily notice
[00:10:17 CST(-0600)] <Carlfish> You could go into Newtown and wander down King St. until something caught your eye.
[00:10:59 CST(-0600)] * bspeakmon fumbles for the map. King St is near, but this Newtown thing is weird and strange.
[00:12:45 CST(-0600)] <bspeakmon> ah, I see
[00:12:49 CST(-0600)] <bspeakmon> but crap, gotta run
[00:12:51 CST(-0600)] <bspeakmon> bbl
[05:09:20 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:18:32 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[06:44:41 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[07:29:21 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:02:46 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[08:54:18 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.102) has joined #atlassiandev
[10:29:47 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[11:04:25 CST(-0600)] * theblackbox (~sammo@unaffiliated/theblackbox) has joined #atlassiandev
[11:05:51 CST(-0600)] <theblackbox> hey all, sorry if this is OT or FAQ (I can't find it if it is) but I've just set up JIRA and edited necessary files for integration with mysql, but I can't seem to shut jira down
[11:06:17 CST(-0600)] <theblackbox> I was wondering if there was a way to force shutdown rather than relying on bin/shutdown.sh
[11:07:07 CST(-0600)] <theblackbox> I get "Error occurred during initialization of VM Could not reserve enough space for object heap"
[11:13:11 CST(-0600)] <JonathanNolen> Hey – that definitely sounds like something that support can help you with. I'd filed a ticket at <http://support.atlassian.com/>. This channel is intended for people developing Atlassian plugins to help each other.
[11:14:19 CST(-0600)] <theblackbox> yeah, I figured I just thought it might be dev know how as it looks like the JVM playing up
[11:14:43 CST(-0600)] <theblackbox> thanks anyway JonathanNolen
[11:15:18 CST(-0600)] <JonathanNolen> well, the first thing to try would be giving it more memory. But if that doesn't work, you'll probably need support's help.
[11:24:02 CST(-0600)] <theblackbox> but giving it more memory would require "rebooting" bit of a catch 22 =S
[11:24:10 CST(-0600)] <theblackbox> following up a support req now
[11:43:20 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:51:34 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[12:59:09 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[14:23:08 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[14:31:39 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:34:27 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:39:04 CST(-0600)] * JonathanNolen_ (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[15:45:58 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:12:29 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:12:29 CST(-0600)] * ChanServ sets mode +o tmoore
[16:13:47 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:45:23 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[16:45:23 CST(-0600)] * ChanServ sets mode +o bspeakmon
[16:52:34 CST(-0600)] * Timothee (~Timothee@207.47.4.66.static.nextweb.net) has joined #atlassiandev
[16:52:46 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:04:33 CST(-0600)] * skrebs_ (~shannon@121.91.169.147) has joined #atlassiandev
[17:23:36 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[17:28:42 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:30:25 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[17:30:26 CST(-0600)] * ChanServ sets mode +o bspeakmon
[17:48:45 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:00:53 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:02:52 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:02:52 CST(-0600)] * ChanServ sets mode +o tmoore

[18:32:54 CST(-0600)] * mryall (~mryall@59.167.164.33) has joined #atlasiandev
[18:32:55 CST(-0600)] * ChanServ sets mode +o mryall
[18:43:56 CST(-0600)] <mryall> morning folks 

[18:47:01 CST(-0600)] <i386> morning mryall 
[18:47:23 CST(-0600)] * skrebs (~shannon@121.91.169.147) has joined #atlasiandev
[19:27:52 CST(-0600)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlasiandev
[19:42:57 CST(-0600)] <Broady> mryall: yo
[19:43:03 CST(-0600)] <Broady> (belated yo)
[19:43:07 CST(-0600)] <mryall> heh
[19:44:04 CST(-0600)] <mryall> discovered today that OSUser and atl-user authentication uses String.getBytes() to hash your password
[19:44:09 CST(-0600)] <mryall> joys of working with legacy code...
[19:44:33 CST(-0600)] <mryall> platform-encoding-specific legacy code, that is
[19:45:04 CST(-0600)] <Broady> hmmm, has it ever been an issue?
[19:47:14 CST(-0600)] <mryall> no, just came across it while migrating the code to Crowd
[19:47:46 CST(-0600)] <mryall> fortunately, Crowd has encoding-switching logic in its LDAP directory, so you can specify the encoding on each hash
[19:48:09 CST(-0600)] <mryall> so we can migrate people away from the legacy encoding to a more consistent one with a proper salt too
[19:48:18 CST(-0600)] <Broady> sounds good

[19:48:31 CST(-0600)] <mryall> yeah, it's just work 
[19:48:41 CST(-0600)] <Broady> how would you handle upgrading to a new scheme?
[19:48:53 CST(-0600)] <Broady> it'd be much easier to store passwords in plain text
[19:49:35 CST(-0600)] <Broady> /lunch
[20:27:55 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlasiandev
[20:27:56 CST(-0600)] * ChanServ sets mode +o bspeakmon
[20:36:15 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlasiandev
[20:36:15 CST(-0600)] * ChanServ sets mode +o bspeakmon
[20:44:00 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlasiandev
[20:44:01 CST(-0600)] * ChanServ sets mode +o bspeakmon
[21:01:03 CST(-0600)] * theblackbox (~sammo@unaffiliated/theblackbox) has joined #atlasiandev
[21:04:55 CST(-0600)] * JonathanNolen_ (~jnolen@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlasiandev
[21:12:57 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlasiandev
[22:18:22 CST(-0600)] <mryall> hehe, plaintext for the win
[22:19:20 CST(-0600)] <Broady> i got a bit of a shock the other day when i logged into a domain name provider's mgmt console, bloody password staring me straight in the face
[22:20:42 CST(-0600)] <mryall> mm

[22:20:54 CST(-0600)] <mryall> makes you check your shoulder, eh? 
[22:39:46 CST(-0600)] * justin_ (~justin@59.167.164.33) has joined #atlasiandev
[22:59:17 CST(-0600)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlasiandev

atlasiandev_log-2010-02-17

[00:13:15 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlasiandev
[00:54:07 CST(-0600)] * dezwart (~pdzwart@ppp121-44-84-124.lns20.syd6.internode.on.net) has joined #atlasiandev
[00:57:33 CST(-0600)] * dezwart (~pdzwart@ppp121-44-84-124.lns20.syd6.internode.on.net) has left #atlasiandev
[02:08:15 CST(-0600)] * skrebs (~shannon@121.91.169.147) has joined #atlasiandev
[03:52:51 CST(-0600)] <snerd> it's all very confusing really
[06:11:04 CST(-0600)] * skrebs_ (~shannon@121.91.169.147) has joined #atlasiandev
[06:24:33 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlasiandev
[06:38:52 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlasiandev
[08:14:13 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlasiandev
[09:10:08 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.102) has joined #atlasiandev
[09:35:18 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.102) has joined #atlasiandev
[09:35:39 CST(-0600)] <jdoklovic> anyone know if JIRA includes velocity tools?
[09:35:45 CST(-0600)] <jdoklovic> and which ones?
[10:46:17 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlasiandev
[11:38:44 CST(-0600)] <jdoklovic> anyone know why mvn deploy tries to deploy a plugin .obr file 2 times in the same build???

[12:10:34 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlasiandev
[12:40:32 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlasiandev
[13:10:51 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlasiandev
[13:34:07 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlasiandev
[13:36:25 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlasiandev
[13:56:12 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlasiandev
[14:53:14 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlasiandev
[15:02:00 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlasiandev
[15:39:33 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlasiandev
[16:26:10 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlasiandev
[16:26:10 CST(-0600)] * ChanServ sets mode +o tmoore
[16:56:09 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlasiandev
[16:56:10 CST(-0600)] * ChanServ sets mode +o bspeakmon
[16:56:52 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlasiandev
[16:57:32 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlasiandev
[16:57:42 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlasiandev
[17:07:26 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlasiandev
[17:07:34 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlasiandev
[17:10:01 CST(-0600)] <mrdon> morning all
[17:14:51 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlasiandev
[17:22:36 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlasiandev

[17:36:09 CST(-0600)] <mryall> morning
[17:40:12 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[17:58:51 CST(-0600)] <mryall> mrdon: do we have obr support in atl-plugins now?
[17:59:04 CST(-0600)] <mryall> see jdoklovic's question above
[18:01:14 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:02:18 CST(-0600)] <mrdon> the obr support has been in amps since almost the beginning
[18:02:27 CST(-0600)] <mrdon> however, it is up to the UPM to have server-side support
[18:02:46 CST(-0600)] <mrdon> the server-side support in AMPS is from the crappy pdkinstall-plugin we put into every app
[18:05:14 CST(-0600)] <mryall> right
[18:05:21 CST(-0600)] <bspeakmon> what does it mean to have server-side support? putting a bundle repo in the product?
[18:05:23 CST(-0600)] <mryall> so there's no app support for OBR installation yet?
[18:06:31 CST(-0600)] <mryall> do we need a new PluginArtifact implementation, or do you think the OBR should be expanded prior to installation??
[18:06:58 CST(-0600)] <mrdon> bspeakmon: it means having the plugin upload allow an .obr file and know how to process it via the felix obr service then install the appropriate plugins
[18:07:03 CST(-0600)] <mrdon> you can see more in the pdkinstall code
[18:07:22 CST(-0600)] <bspeakmon> but where does it get any dependent plugins?
[18:07:26 CST(-0600)] <mrdon> mryall: the obr files is expanded via the pdkinstall plugin, w/ the help of the felix obr service
[18:07:30 CST(-0600)] <mrdon> they are bundled in the obr file
[18:07:46 CST(-0600)] <mrdon> obr file contains the plugin, dependencies, and an xml file describing the package dependencies
[18:07:55 CST(-0600)] <bspeakmon> I'm thinking of the case where someone uploads a bundle with deps that it expects the product to provide
[18:08:03 CST(-0600)] <mryall> so from the atl-plugins API point of view, they get installed separately?
[18:08:07 CST(-0600)] <mrdon> correct
[18:08:22 CST(-0600)] <mryall> I think there might be some confusion around the term OBR, though
[18:08:27 CST(-0600)] <mryall> <http://stackoverflow.com/questions/260256/does-any-one-use-osgis-obr>
[18:08:36 CST(-0600)] <mryall> ^ this seems to refer to a central repo for bundles, a la Maven
[18:08:59 CST(-0600)] <mrdon> <https://labs.atlassian.com/svn/PDKI/trunk/src/main/java/com/atlassian/pdkinstall/ObrPluginTypeInstaller.java>
[18:09:00 CST(-0600)] <bspeakmon> right, that's what I'm also asking
[18:09:09 CST(-0600)] <mrdon> yes, you can do that, but I'd prefer not to
[18:09:19 CST(-0600)] <mrdon> the obr file basically is a local repo
[18:09:24 CST(-0600)] <mrdon> sorry, the obr zip 

[18:09:34 CST(-0600)] <mrdon> not to be confused with the obr xml file inside the obr zip
[18:09:48 CST(-0600)] <bspeakmon> but an obr CAN depend on stuff it doesn't contain
[18:09:56 CST(-0600)] <bspeakmon> like, for example, com.atlassian crap
[18:10:08 CST(-0600)] <mrdon> well, you pass the repo(s) to the obr service
[18:10:16 CST(-0600)] <mrdon> so whether they are remote or local, up to you
[18:10:47 CST(-0600)] <mryall> so the UPM will allow configuring the remote lookups for Felix - proxy settings, etc.?
[18:11:07 CST(-0600)] <mryall> is there an HTTP retrieval service in SAL yet?
[18:11:13 CST(-0600)] <mrdon> I'd rather not
[18:11:16 CST(-0600)] <mryall> we should probably push the Confluence one in there
[18:11:25 CST(-0600)] <mryall> it's fairly abstract, supports trusted apps, etc.
[18:11:28 CST(-0600)] <mrdon> I like depending on these .obr files, which already contain their dependencies
[18:11:34 CST(-0600)] <mrdon> that way, no network access required
[18:11:37 CST(-0600)] <mryall> yeah
[18:11:44 CST(-0600)] <mryall> network access at start-up time would be sucky
[18:11:47 CST(-0600)] <bspeakmon> so. an atlassian plugin in OBR form should be expected to contain everything it needs and not rely on anything else being made available (except com.atlassian)?
[18:11:55 CST(-0600)] <mrdon> and the http retrieval service in sal is basically a copy/paste from confluence with some changes
[18:12:01 CST(-0600)] <mryall> oh it is?
[18:12:03 CST(-0600)] <mryall> nice
[18:12:08 CST(-0600)] <mryall> we should change our implementation to delegate to it
[18:12:11 CST(-0600)] <mrdon> bspeakmon: yes, it contains everything it needs
[18:12:36 CST(-0600)] <mrdon> mryall: that might be tricky since it is an osgi service
[18:12:46 CST(-0600)] <mrdon> and the api jar is deployed as a bundle
[18:13:08 CST(-0600)] <mryall> right - compile-time dependencies would be tricky
[18:13:20 CST(-0600)] <bspeakmon> all this naturally presupposes a world where the products have been fully OSGi-ized, with packages being exported from the bundled plugins so installable plugins can get to them
[18:13:22 CST(-0600)] <mryall> it should be a common library which both SAL and Confluence use, then
[18:13:31 CST(-0600)] <mrdon> basically you'd put the interfaces in core, then the impl as a plugin, then a bridge to lookup the impl
[18:13:51 CST(-0600)] <mryall> bspeakmon: I think the main problem it is solving is to bundle a bunch of plugins together - e.g. gadgets, oauth, dashboards
[18:14:13 CST(-0600)] <mryall> it isn't supposed to solve plugins which have external dependencies that can't be bundled with it
[18:14:21 CST(-0600)] <mrdon> bspeakmon: if you mean it only works with osgi plugins that have package deps on each other, then yes
[18:14:22 CST(-0600)] <mryall> as far as I understand, anyway
[18:14:23 CST(-0600)] <bspeakmon> that's fine
[18:14:31 CST(-0600)] <mrdon> though you can do a bundle import to force that dep
[18:15:03 CST(-0600)] <mrdon> the code in amps is pretty simple - find all plugins marked as provided and put them in the zip
[18:15:10 CST(-0600)] <bspeakmon> yes, I saw that
[18:16:07 CST(-0600)] <bspeakmon> so we're going to require authors to write
<Import-Package>com.atlassian.jira*>;version="4.0.2"<!-- part of doing a jira plugin?-->
[18:17:29 CST(-0600)] <mryall> it's automatically generated if you don't provide that instruction, so you're not required to do it
[18:17:29 CST(-0600)] <mrdon> this method does presuppose they are generating their manifest at compile time, however, the imports could be autogenerated
[18:18:38 CST(-0600)] <bspeakmon> hm
[18:18:40 CST(-0600)] <mrdon> correct
[18:18:54 CST(-0600)] <mrdon> we could give a warning saying their plugin couldn't be used with obr or something
[18:19:21 CST(-0600)] <mrdon> but it usually a special type of plugin meant to be depended upon
[18:21:39 CST(-0600)] <bspeakmon> this is going to be damn hard to explain to developers, much less write correctly
[18:25:32 CST(-0600)] <bspeakmon> there's already resistance to plugins 2

[18:29:27 CST(-0600)] <mrdon> well, most plugins don't depend on other plugins that aren't bundled
[18:29:41 CST(-0600)] <mrdon> and if so, having to learn some osgi should be expected
[18:32:11 CST(-0600)] * dchui (<mailto:dchui@202.169.29.34>) has joined #atlassiandev
[18:32:17 CST(-0600)] <bspeakmon> lots of plugins depend on stuff they think is in the product. not just com.atlassian, but commons-lang, google-collect, webwork, etc.
[18:36:03 CST(-0600)] <bspeakmon> and we change that stuff from version to version, without notice
[18:36:12 CST(-0600)] <bspeakmon> it's de facto public API
[18:36:57 CST(-0600)] <mryall> most of that stuff never changes, does it?
[18:37:08 CST(-0600)] <mryall> I don't think we've bumped a webwork version since I've been here in Confluence
[18:37:25 CST(-0600)] <mryall> and the Apache libraries are almost always entirely backwards compatible
[18:37:59 CST(-0600)] <mryall> I think plugin authors would be more concerned about changes to com.atlassian.core, com.atlassian.user, etc.
[18:39:47 CST(-0600)] <bspeakmon> but you don't know
[18:39:49 CST(-0600)] <bspeakmon> no one does
[18:40:05 CST(-0600)] <bspeakmon> yes, the com.atlassian stuff changes more often, and that's where the most pain comes from
[18:41:13 CST(-0600)] <bspeakmon> but when you get people using, say, TextUtils, who don't bundle webwork in their plugin because they know conf uses it – well, that's bad
[18:42:21 CST(-0600)] <bspeakmon> so we can document and export those packages from each project, or we can make them all private-package and export only com.atlassian

[18:42:30 CST(-0600)] <bspeakmon> both solutions have pain 😊
[18:46:52 CST(-0600)] <bspeakmon> I'd rather just make all the third party stuff private, if it weren't for the problem that it would break approximately 93.8% of plugins, including 100% of the good ones
[18:47:22 CST(-0600)] <mrdon> wrt to library versions, we have never done checks before
[18:47:29 CST(-0600)] <mrdon> you just drop it all into web-inf/lib and hope for the best
[18:47:34 CST(-0600)] <mrdon> we still have that now with plugins 2
[18:47:53 CST(-0600)] <mrdon> only now you have the option to specify what versions you want and have the install fail if those conditions won't be met
[18:48:06 CST(-0600)] <mrdon> I don't see how we can do better than that
[18:49:31 CST(-0600)] <bspeakmon> but are the products exporting the package and version of the libraries being used?
[18:49:39 CST(-0600)] <bspeakmon> (I'm asking, I honestly don't know)
[18:50:21 CST(-0600)] <mrdon> yes
[18:50:23 CST(-0600)] <mrdon> for the most part
[18:50:29 CST(-0600)] <mrdon> jira is really good about it
[18:50:42 CST(-0600)] <mrdon> confluence not as much, though the version guessing algorithm usually gets it right
[18:52:14 CST(-0600)] <bspeakmon> it would alleviate a lot of pain if we just sat down, got them right, and then published them on CAC
[18:54:40 CST(-0600)] <bspeakmon> along with the "here's how you could import the product's stuff, but it's subject to change without notice and you should really bundle your own dealie!"
[18:54:41 CST(-0600)] <bspeakmon> giggles was worried about a situation where five plugins all bundle google-collections and suck up lots of classloader memory, but I reassured him that nobody besides us gives a shit about google-collections 😊
[18:58:30 CST(-0600)] <mrdon> heh
[18:58:37 CST(-0600)] <mrdon> but even still, there are a lot of dups in our plugins
[18:58:44 CST(-0600)] <mrdon> well, maybe a lot is strong, but a few
[18:59:07 CST(-0600)] <mrdon> and they shouldn't need to bundle their own as long as they specify version ranges
[18:59:25 CST(-0600)] <mrdon> and yes, it would be awesome to, at the very least, get the upm to show host components and host packages
[18:59:36 CST(-0600)] <mrdon> publishing to CAC somehow automated would rock
[19:00:10 CST(-0600)] <bspeakmon> agreed. but if they specify a version range the product doesn't have anymore...
[19:00:10 CST(-0600)] <bspeakmon> install fails
[19:00:10 CST(-0600)] <bspeakmon> which is correct
[19:00:10 CST(-0600)] <bspeakmon> but then they have to figure out and bundle it, or upgrade to whatever the product is now using
[19:00:21 CST(-0600)] <mrdon> yep
[19:00:36 CST(-0600)] <mrdon> if their lib is a bundle, they can use obr to optionally install it
[19:00:43 CST(-0600)] <mrdon> best of both worlds
[19:01:07 CST(-0600)] <bspeakmon> we all know 94.2% of libs won't be bundles 😊
[19:01:15 CST(-0600)] <mrdon> well, there is that 😊
[19:02:15 CST(-0600)] <bspeakmon> so where do the products declare versions of their packages? something in the plugins implementation?
[19:02:29 CST(-0600)] <bspeakmon> (because I'll probably wind up doing this myself 😊)
[19:03:06 CST(-0600)] * twong (<mailto:twong@216-75-233-106.static.wiline.com>) has joined #atlassiandev
[19:08:54 CST(-0600)] <mryall> bspeakmon: it's in the package scanner config
[19:10:07 CST(-0600)] <bspeakmon> and if I put explicit exports in there, it's not going to break anything?
[19:16:07 CST(-0600)] <mrdon> well, I haven't seen a case where you want to export a package not in the classloader
[19:16:11 CST(-0600)] <mrdon> however, you can customize versions
[19:16:24 CST(-0600)] <mrdon> also, at least currently, package scanner config is done anywhere the product likes
[19:16:28 CST(-0600)] <mrdon> fisheye does it in their java code
[19:16:31 CST(-0600)] <mrdon> confluence xml
[19:16:35 CST(-0600)] <mrdon> jira java code as well
[19:16:46 CST(-0600)] <mrdon> this is definitely something that needs improving
[19:18:21 CST(-0600)] <bspeakmon> hm
[19:18:21 CST(-0600)] <bspeakmon> maybe I'm saying the wrong thing
[19:18:23 CST(-0600)] <bspeakmon> when the product is running, they all have packages that plugins can import, along with versions of them. that stuff is configured somewhere
[19:19:24 CST(-0600)] <mrdon> yep, packagescanner
[19:19:33 CST(-0600)] <mrdon> but how they configure it is up to the product
[19:21:08 CST(-0600)] <mryall> well, the packages and versions depend highly on the product
[19:21:20 CST(-0600)] <mryall> unless you can add it some way declaratively to the POM

[19:21:36 CST(-0600)] <mryall> there's no "logical" place for it, imho
[19:40:41 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:41:24 CST(-0600)] * skrebs_ (~shannon@119.12.31.218) has joined #atlassiandev
[19:45:31 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:15:17 CST(-0600)] * skrebs_ (~shannon@119.12.31.218) has joined #atlassiandev
[21:00:27 CST(-0600)] <mrdon> mryall: well, a couple thoughts on pkgsScanner config
[21:00:36 CST(-0600)] <mrdon> 1. It should be in a text file, probably outside other config
[21:00:53 CST(-0600)] <mryall> okay, like in a manifest format?
[21:01:02 CST(-0600)] <mrdon> 2. it should be possible to add exports and versions somehow, like via a sysprop or something
[21:01:13 CST(-0600)] <mrdon> I guess, only manifest files suck
[21:01:16 CST(-0600)] <mryall> yeah
[21:01:17 CST(-0600)] <mrdon> but yes, it would work
[21:01:22 CST(-0600)] <mryall> "add exports"
[21:01:24 CST(-0600)] <mryall> what would that mean?
[21:01:38 CST(-0600)] <mrdon> well, think of a plugin that has a version 1 plugin component
[21:01:45 CST(-0600)] <mrdon> usually via some low-level integration thing
[21:01:53 CST(-0600)] <mrdon> no way to export those packages unless they match an existing pattern
[21:02:08 CST(-0600)] <mrdon> I guess if the scanner txt file was available, they could provide instructions on manually overriding that
[21:02:17 CST(-0600)] <mrdon> or you could support some system property or whatever
[21:02:37 CST(-0600)] <mryall> yeah, but it kind of no longer becomes a plugin then
[21:02:42 CST(-0600)] <mrdon> exactly
[21:02:43 CST(-0600)] <mryall> if you have to modify the system config to use it
[21:02:53 CST(-0600)] <mryall> that's kind of lame
[21:02:56 CST(-0600)] <mrdon> you could also say that you concat all package-scanner.txt files
[21:03:02 CST(-0600)] <mrdon> then a plugin could simply put that file in its jar
[21:03:07 CST(-0600)] <mryall> hmm ok
[21:03:07 CST(-0600)] <mryall> why .txt instead of .xml?
[21:03:16 CST(-0600)] <mrdon> cause my fingers were closer to txt
[21:03:18 CST(-0600)] <mryall> it has some internal structure that might suit XML better
[21:03:23 CST(-0600)] <mryall> ok
[21:03:24 CST(-0600)] <mrdon> the other involves my right hand
[21:03:27 CST(-0600)] <mryall> hehe

[21:03:46 CST(-0600)] <mryall> both hands on the keyboard, don! 😊
[21:03:52 CST(-0600)] <mrdon> I need to drink!
[21:04:00 CST(-0600)] <mrdon> but yeah, the more I think about it, I like this idea

[21:04:01 CST(-0600)] <mryall> geez, it's only 2.30 😊
[21:04:10 CST(-0600)] <mrdon> then the defaults could be in WEB-INF/classes or the app jar
[21:04:12 CST(-0600)] <mryall> 2 o'clock, even
[21:04:20 CST(-0600)] <mrdon> but static plugins could override as needed
[21:04:28 CST(-0600)] <mryall> append, you mean?
[21:04:32 CST(-0600)] <mrdon> and yes, it is early, but it is hot as well, so need my water
[21:04:42 CST(-0600)] <mrdon> speaking of, what happened to all the water glasses?
[21:04:48 CST(-0600)] <mrdon> and, append
[21:04:50 CST(-0600)] <mrdon> what did I say?
[21:04:53 CST(-0600)] <mryall> "override"
[21:04:55 CST(-0600)] <mrdon> append/merge, whatever
[21:04:56 CST(-0600)] <mryall> ^ scary
[21:05:19 CST(-0600)] <mrdon> well, I could see a plugin wanting to override say a package version
[21:05:24 CST(-0600)] <mryall> I don't see a big problem with Confluence having it in our Spring config, to be honest
[21:05:27 CST(-0600)] <mrdon> no, that doesn't make sense
[21:05:28 CST(-0600)] <mrdon> nm
[21:05:35 CST(-0600)] <mryall> it's just the Java code that's annoying
[21:05:47 CST(-0600)] <mrdon> that's fine, but still a static plugin needs to be able to influence the config
[21:05:50 CST(-0600)] <mryall> yeah
[21:05:57 CST(-0600)] <mryall> DynamicPackageScannerConfiguration
[21:06:04 CST(-0600)] <mrdon> yep
[21:06:09 CST(-0600)] <mrdon> may sneak that into 2.5
[21:07:25 CST(-0600)] <bspeakmon> lots of glasses in the dishwasher
[21:10:26 CST(-0600)] <mryall> I lucked out the other day, and found one of the old massive glasses we had in the old office
[21:10:32 CST(-0600)] <mryall> never going to let that one go now
[21:10:49 CST(-0600)] <mryall> it's hold like 500 mL or something - awesome
[21:10:57 CST(-0600)] <bspeakmon> ooo
[21:11:04 CST(-0600)] <bspeakmon> watch your desk, that one may vanish on you
[21:11:33 CST(-0600)] <bspeakmon> I miss our SF pint glasses

[21:13:28 CST(-0600)] <Broady> come work for CCA. fridge full of 600 ml water bottles 😊
[21:14:01 CST(-0600)] <Broady> wait what am i saying... disregard

[21:14:37 CST(-0600)] <bspeakmon> 😊
[21:15:08 CST(-0600)] <Broady> llol mryall "tough as old nails"
[21:15:35 CST(-0600)] <Broady> tough as old boots x tough as nails?
[21:16:21 CST(-0600)] <bspeakmon> tough as old nails used to hold together tough old boots?
[21:50:00 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:04:15 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-02-18

[23:10:57 CST(-0600)] <mryall> aren't old nails tough?

[23:11:08 CST(-0600)] <mryall> I sure wouldn't want to take one on
[23:13:14 CST(-0600)] <Broady> they might be rusty
[23:34:24 CST(-0600)] <mryall> yeah, exactly
[23:34:27 CST(-0600)] <mryall> high risk of tetanus
[23:36:04 CST(-0600)] <Broady> i wonder if work would give me a day off to go racing
[23:36:06 CST(-0600)] <Broady> hehe
[23:36:18 CST(-0600)] <Broady> probably not, i got a weird look even when i asked to see my honours supervisor
[00:05:09 CST(-0600)] <mryall> what kind of racing?
[00:08:14 CST(-0600)] <Broady> nothing serious, just fun days at the circuit
[00:08:22 CST(-0600)] <Broady> the cheapie ones that run during the week aren't timed

[00:08:55 CST(-0600)] <Broady> oh, car... ive taken my mx5 a couple times, and want to MORE 😊 it's awesome fun
[00:15:10 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:10:46 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[02:21:02 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[02:28:27 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[02:50:15 CST(-0600)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[05:43:59 CST(-0600)] * i386laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[06:30:07 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[06:37:47 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:32:52 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[07:57:11 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[08:39:32 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.102) has joined #atlassiandev
[11:09:05 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:21:32 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[13:40:40 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[14:01:37 CST(-0600)] <jdoklovic> can i say again how much i hate studio.plugins !!
[14:41:13 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:22:31 CST(-0600)] * bug (~bug@12.155.186.6) has joined #atlassiandev
[15:34:27 CST(-0600)] <mrdon> jdoklovic: why?
[15:36:12 CST(-0600)] <jdoklovic> cause it randomly takes me to a non-functioning login screen even though i'm logged in
[15:37:00 CST(-0600)] <jdoklovic> such as: i login, click on a project, click to an issue, click on the issues tab to get back to the issues list and BAM login
[15:37:42 CST(-0600)] <jdoklovic> it's almost unusable. everytime i get one of those, I simply hit the dashboard again (which now it knows i'm still logged in) and navigate back to where i was
[15:38:33 CST(-0600)] * bug (~bug@12.155.186.6) has joined #atlassiandev
[15:46:36 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[15:49:19 CST(-0600)] <mrdon> hmm...definitely a bug
[15:49:26 CST(-0600)] <mrdon> did you file an issue or support request?

[15:49:32 CST(-0600)] <mrdon> or yell at jonathan? 😊
[15:49:40 CST(-0600)] <mrdon> or I guess Jens it is now
[16:08:18 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:08:18 CST(-0600)] * ChanServ sets mode +o tmoore
[16:09:52 CST(-0600)] * i386laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[16:12:42 CST(-0600)] * i386laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[16:20:59 CST(-0600)] * bug (~bug@12.155.186.6) has joined #atlassiandev
[16:58:09 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[16:58:10 CST(-0600)] * ChanServ sets mode +o bspeakmon
[17:10:23 CST(-0600)] * dchui (~dchui@61.59.49.60.klj02-home.tm.net.my) has joined #atlassiandev
[17:11:54 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[17:34:57 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:35:43 CST(-0600)] <tmoore> do we document the packages exported by JIRA 4 anywhere?
[18:00:53 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:10:46 CST(-0600)] * JonathanNolen_ (~jnolen@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[18:13:12 CST(-0600)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:23:41 CST(-0600)] <jazzy> document? what's that?
[18:35:11 CST(-0600)] <tmoore> right
[18:35:30 CST(-0600)] <tmoore> I think I knew the answer before I posted the question :-P
[19:13:27 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:09:33 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:45:33 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[20:46:48 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:48:07 CST(-0600)] * skrebs_ (~shannon@119.12.31.218) has joined #atlassiandev
[20:51:29 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[20:51:29 CST(-0600)] * ChanServ sets mode +o bspeakmon
[21:07:23 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[21:11:27 CST(-0600)] <mryall> hehe
[21:11:33 CST(-0600)] <mryall> they're documented in the source code, of course
[21:11:50 CST(-0600)] <mryall> PackageScannerConfigurationFactory ftw
[21:14:41 CST(-0600)] <mrdon> we need a standard page you can view to see the results of the scan
[21:14:46 CST(-0600)] <mrdon> something that will be in upm, hopefully
[21:18:08 CST(-0600)] <bspeakmon> a what now?
[21:18:21 CST(-0600)] <mrdon> confluence has one
[21:18:31 CST(-0600)] <mryall> only for components, iirc
[21:18:36 CST(-0600)] <mrdon> a url you can visit to see all the package and versions of packages exported by the system
[21:18:43 CST(-0600)] <mrdon> no, I think there is another...
[21:18:47 CST(-0600)] <mrdon> lemme check...
[21:18:51 CST(-0600)] <mryall> /admin/pluginexports.action
[21:20:18 CST(-0600)] <bspeakmon> same thing the web console does, except built in to the product?
[21:20:24 CST(-0600)] <tmoore> oh nice

[21:21:47 CST(-0600)] <mrdon> yeah, I really miss that and plugin-bundles.jsp when I'm in other apps
[21:22:07 CST(-0600)] <mrdon> maybe we standardise the osgi console, whatever, I just want something
[21:22:26 CST(-0600)] <bspeakmon> write a cross-product plugin 

[21:22:51 CST(-0600)] <mrdon> like the upm
[21:23:12 CST(-0600)] <bspeakmon> sure
[21:23:22 CST(-0600)] <bspeakmon> plus the upm may actually get finished one of these days
[21:23:30 CST(-0600)] <bspeakmon> AND then the sdk wouldn't need to worry about the web console
[21:23:37 CST(-0600)] <mrdon> yep
[21:23:41 CST(-0600)] <bspeakmon> ...in a year or so.
[21:36:02 CST(-0600)] * i386 (~jdmay@59.167.164.33) has joined #atlassiandev
[21:50:43 CST(-0600)] * skrebs (~shannon@119.12.31.218) has joined #atlassiandev

atlassiandev_log-2010-02-19

[01:22:51 CST(-0600)] * Carlfish_ (~cmiller@59.167.164.33) has joined #atlassiandev
[01:52:21 CST(-0600)] * skrebs_ (~shannon@119.12.31.218) has joined #atlassiandev
[02:26:10 CST(-0600)] * lmaslovs (~leonidms@80.233.159.254) has joined #atlassiandev
[02:29:24 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btcentralplus.com) has joined #atlassiandev
[03:09:51 CST(-0600)] <trochej> Hi, is there a simple way to run Linux Bamboo 2.5 standalone as given user from system startup scripts?
[04:18:44 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[04:29:34 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[05:02:14 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[05:13:51 CST(-0600)] * skrebs (~shannon@119.12.31.218) has joined #atlassiandev
[05:21:06 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[06:08:05 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[06:24:02 CST(-0600)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[06:50:37 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:16:18 CST(-0600)] <trochej> Found a solution: a wrapper calling Bamboo/bamboo.sh actions via runuser - bamboo -c
[07:59:39 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btcentralplus.com) has joined #atlassiandev
[08:19:55 CST(-0600)] * theblackbox (~sammo@unaffiliated/theblackbox) has joined #atlassiandev
[08:40:34 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.102) has joined #atlassiandev
[08:56:57 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:47:25 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:54:19 CST(-0600)] * Jilliano (~jdressler@69.11.216.98) has joined #atlassiandev
[11:12:55 CST(-0600)] * i386|laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[12:04:11 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[12:37:41 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:53:59 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[14:20:00 CST(-0600)] * cloder_ (~cloder@76-217-151-154.lightspeed.irvnca.sbcglobal.net) has joined #atlassiandev

[14:20:02 CST(-0600)] <cloder_> hi 
[15:35:17 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[15:39:26 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:51:26 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[17:11:50 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:51:36 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[19:00:06 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:34:00 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[20:45:17 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:14:25 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:36:17 CST(-0600)] * i386|laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev

atlassiandev_log-2010-02-20

[03:34:16 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btcentralplus.com) has joined #atlassiandev
[03:45:27 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[06:40:33 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[07:31:55 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[12:47:38 CST(-0600)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[12:51:54 CST(-0600)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[14:20:48 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:19:04 CST(-0600)] * cloder_ (~cloder@76-217-151-154.lightspeed.irvnca.sbcglobal.net) has joined #atlassiandev
[16:25:56 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[17:08:13 CST(-0600)] * mm_ (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[17:32:45 CST(-0600)] * bug (~bug@adsl-76-200-186-146.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[17:54:02 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[18:44:34 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[21:03:35 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[21:46:54 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:38:19 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-02-21

[08:33:02 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[09:00:47 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[09:20:15 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[10:45:59 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev

[10:47:18 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[12:38:10 CST(-0600)] * bug_ (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[13:01:03 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[13:03:09 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[13:58:13 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btcentralplus.com) has joined #atlassiandev
[15:32:02 CST(-0600)] * bug (~bug@adsl-76-200-186-146.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[16:00:26 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[16:00:32 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[16:17:37 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[16:17:48 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[16:36:17 CST(-0600)] <mryall> morning, people
[16:37:23 CST(-0600)] <Broady> yawning
[16:40:25 CST(-0600)] <mryall> yeah, might be time for a cuppa
[16:49:22 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:49:22 CST(-0600)] * ChanServ sets mode +o tmoore
[16:56:36 CST(-0600)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[17:31:18 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:41:50 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[17:49:40 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:49:40 CST(-0600)] * ChanServ sets mode +o tmoore
[18:07:21 CST(-0600)] * mryall_ (~matt@59.167.164.33) has joined #atlassiandev
[18:07:22 CST(-0600)] * ChanServ sets mode +o mryall_
[18:08:24 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[18:08:24 CST(-0600)] * ChanServ sets mode +o bspeakmon
[18:08:27 CST(-0600)] * furgin (~mjensen@59.167.164.33) has joined #atlassiandev
[18:09:14 CST(-0600)] * mryall killed his clone
[18:09:15 CST(-0600)] <mryall> muahaha
[18:09:24 CST(-0600)] <bspeakmon> I think I saw that movie
[18:10:12 CST(-0600)] * andreas_knecht (~andreas_k@59.167.164.33) has joined #atlassiandev
[18:10:22 CST(-0600)] <bspeakmon> holy shit, an andreas sighting

[18:10:27 CST(-0600)] <andreas_knecht>
[18:10:41 CST(-0600)] * mryall splinters
[18:10:46 CST(-0600)] * mjensen (~mjensen@59.167.164.33) has left #atlassiandev
[18:11:39 CST(-0600)] <Carlfish> Does it count if you get andreas to type California?
[18:11:48 CST(-0600)] <andreas_knecht> No
[18:12:22 CST(-0600)] <Carlfish> Spoilsport.
[18:34:13 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:56:22 CST(-0600)] <mryall> argh, did it again
[18:56:37 CST(-0600)] <mryall> logged in to my work desktop via SSH, created a new project at the terminal
[18:56:43 CST(-0600)] <mryall> then wondered why I can't find it on my laptop
[18:56:45 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[18:56:46 CST(-0600)] * ChanServ sets mode +o bspeakmon
[18:58:10 CST(-0600)] <Broady> @mryall customise your PS1
[18:58:33 CST(-0600)] <mryall> I have: mryall@draught plugins\$
[18:58:44 CST(-0600)] <mryall> I need to change the colour of the hostname or something
[18:58:54 CST(-0600)] <mryall> make home blue and work green, or something
[18:59:27 CST(-0600)] <Broady> you should be able to recognise if its a remote session too
[18:59:41 CST(-0600)] <Broady> (i.e. not show the hostname if its local)
[19:04:17 CST(-0600)] <mryall> hmm
[19:04:22 CST(-0600)] <mryall> that would be nice for local
[19:04:27 CST(-0600)] <mryall> any idea how to do that?
[19:05:48 CST(-0600)] <mryall> looks like people tend to use if [-n "\$SSH_CLIENT"];
[19:06:14 CST(-0600)] <Broady> yup
[19:06:20 CST(-0600)] <Broady> SSH2_CLIENT too
[19:07:01 CST(-0600)] <Broady> http://www.debian-administration.org/article/Fancy_Bash_Prompts
[19:07:05 CST(-0600)] <Broady> complete overkill but pretty cool
[19:07:24 CST(-0600)] <bspeakmon> ooo
[19:34:37 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:42:07 CST(-0600)] <bspeakmon> basically all the attributes in
http://confluence.atlassian.com/display/PLUGINFRAMEWORK/_Content%20for%20Component%20Plugin%20Module, since they're the same in all jira 4 plugins
[19:44:54 CST(-0600)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[20:01:15 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[20:01:16 CST(-0600)] * ChanServ sets mode +o bspeakmon
[20:07:05 CST(-0600)] <i386>
http://laughingsquid.com/why-you-shouldnt-drive-behind-a-jet-engine/?utm_source=feedburner&utm_medium=feed&utm_campaign=
)[20:14:15 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[20:14:16 CST(-0600)] * ChanServ sets mode +o bspeakmon
[20:15:18 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[20:37:55 CST(-0600)] <Broady> 13:37 <artr> hey Broady explain the cheese thing with confluence?
[20:37:56 CST(-0600)] <Broady> lol
[20:38:59 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:39:19 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[20:40:34 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[20:40:34 CST(-0600)] * ChanServ sets mode +o tmoore
[20:42:12 CST(-0600)] <mryall> bspeakmon: was that to the right channel?
[21:02:06 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:15:31 CST(-0600)] <mryall> Broady: I like cheese
[21:31:30 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev

[21:31:31 CST(-0600)] * ChanServ sets mode +o bspeakmon
[21:37:38 CST(-0600)] <Broady> mryall: that was my answer pretty much... "charles likes cheese"
[21:37:58 CST(-0600)] <mryall> hehe
[21:49:16 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-02-22

[23:32:41 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[23:32:41 CST(-0600)] * ChanServ sets mode +o tmoore
[00:23:04 CST(-0600)] * mryall_ (~matt@59.167.164.33) has joined #atlassiandev
[00:23:04 CST(-0600)] * ChanServ sets mode +o mryall_
[01:02:38 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btcentralplus.com) has joined #atlassiandev
[02:30:28 CST(-0600)] * imaslovs (~leonidms@80.233.159.254) has left #atlassiandev
[02:31:37 CST(-0600)] * imaslovs (~leonidms@80.233.159.254) has joined #atlassiandev
[05:00:27 CST(-0600)] * MartinCleaver (~martinle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[07:08:15 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:11:03 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:50:15 CST(-0600)] * MartinCleaver (~martinle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[08:18:18 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btcentralplus.com) has joined #atlassiandev
[08:21:19 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:58:33 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.85) has joined #atlassiandev
[10:02:37 CST(-0600)] * theblackerbox (~sammo@92.27.217.117) has joined #atlassiandev
[10:09:06 CST(-0600)] * cloder_ (~cloder@76-217-151-154.lightspeed.irvnca.sbcglobal.net) has joined #atlassiandev
[11:06:23 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[11:07:12 CST(-0600)] <phuff> I can't get fisheye to run using the sdk, because of a dependency issue that looks very similar to this:
<http://jira.atlassian.com/browse/FE-2087>
[11:07:41 CST(-0600)] <phuff> I'm trying to follow this tutorial
<http://confluence.atlassian.com/display/FECRUDEV/FishEye+Twitter+Integration+Plugin+Tutorial>
[11:08:18 CST(-0600)] <phuff> anybody seen that before?
[11:20:28 CST(-0600)] * jonmort_ (~jonmort@host86-141-202-101.range86-141.btcentralplus.com) has joined #atlassiandev
[11:28:09 CST(-0600)] * MartinCleaver (~martinle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[12:21:14 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[13:10:31 CST(-0600)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[15:25:23 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[15:47:28 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[15:56:36 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[16:29:11 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[16:29:12 CST(-0600)] * ChanServ sets mode +o bspeakmon
[16:33:28 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:33:28 CST(-0600)] * ChanServ sets mode +o tmoore
[16:53:17 CST(-0600)] * mryall (~matt@59.167.164.33) has joined #atlassiandev
[16:53:17 CST(-0600)] * ChanServ sets mode +o mryall
[16:54:12 CST(-0600)] <mryall> morning all
[16:55:14 CST(-0600)] <mrdron> howdy
[17:15:27 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[17:15:27 CST(-0600)] * ChanServ sets mode +o bspeakmon
[17:36:36 CST(-0600)] * justin_ (~justin@59.167.164.33) has joined #atlassiandev
[17:40:41 CST(-0600)] * bug (~bug@65-112-21-194.dia.static.qwest.net) has joined #atlassiandev
[17:49:12 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[17:50:00 CST(-0600)] * Carlfish (~cmiller@59.167.164.33) has joined #atlassiandev
[18:09:16 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[18:24:01 CST(-0600)] * MartinCleaver (~martinle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[18:36:49 CST(-0600)] * MartinCleaver (~martinle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[19:34:46 CST(-0600)] * MartinCleaver (~martinle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[21:13:04 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[21:13:04 CST(-0600)] * ChanServ sets mode +o tmoore
[21:24:43 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[21:24:43 CST(-0600)] * ChanServ sets mode +o tmoore
[21:28:25 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:43:58 CST(-0600)] <mryall> added a CAC shortcut link for Confluence API documentation
[22:44:00 CST(-0600)] <mryall> example:
[22:44:02 CST(-0600)] <mryall> [PageManager|com/atlassian/confluence/pages/PageManager@confapi]
[22:44:21 CST(-0600)] <mryall> ^ becomes
<http://docs.atlassian.com/atlassian-confluence/latest/com/atlassian/confluence/pages/PageManager.html>
[22:53:26 CST(-0600)] <Broady> mmm nice, how do you do that?
[22:55:50 CST(-0600)] <Broady> <http://confluence.atlassian.com/display/DOC/Configuring+Shortcut+Links>
[22:55:51 CST(-0600)] <Broady> goddit
[23:02:11 CST(-0600)] <tmoore> mryall: what's the right way to parse JSON in a Confluence plugin?
[23:02:22 CST(-0600)] <tmoore> the javadocs for com.atlassian.json.parser are kind of confusing
[23:02:30 CST(-0600)] <tmoore> is that package deprecated?
[23:02:48 CST(-0600)] <mryall> "parse JSON" ?
[23:02:53 CST(-0600)] <mryall> sent from the client side?
[23:03:00 CST(-0600)] <mryall> we have a library but it's deprecated
[23:03:05 CST(-0600)] <mryall> you should bundle your own, probably
[23:03:23 CST(-0600)] <mryall> can you send it back in request parameters instead?

atlassiandev_log-2010-02-23

[23:04:18 CST(-0600)] <tmoore> well in a test class actually

[23:04:45 CST(-0600)] <tmoore> I could use Jettison
[23:04:54 CST(-0600)] <tmoore> just wasn't sure of the status of the built in stuff
[23:07:01 CST(-0600)] <mryall> it's probably alright to use the existing stuff in a test
[23:07:07 CST(-0600)] <mryall> I think we use it in our acceptance tests
[23:07:12 CST(-0600)] <mryall> so it's not likely to disappear overnight
[23:08:20 CST(-0600)] <mryall> ooo nice - the new code macro has a 'startline' parameter for sub-code snippets
[23:10:43 CST(-0600)] <mryall> sorry, firstline
[23:13:40 CST(-0600)] <bspeakmon> yeah, it does ranges
[23:13:49 CST(-0600)] <bspeakmon> we had our intern hack that in for our mutual pleasure
[23:15:18 CST(-0600)] <Broady> silly interns
[23:31:32 CST(-0600)] <Broady> heh. i just got code complete for free off yoink.com
[23:41:37 CST(-0600)] <mryall> nice
[23:41:41 CST(-0600)] <mryall> which edition?
[23:43:55 CST(-0600)] <Broady> i think its the 1st
[23:44:02 CST(-0600)] <Broady> 16:40 < F> i feel like food now
[23:44:05 CST(-0600)] <Broady> oops
[23:44:10 CST(-0600)] <Broady> <http://www.yoink.com/items/582/>
[23:51:48 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[23:58:53 CST(-0600)] <mryall> hehe
[23:58:55 CST(-0600)] <mryall> wrong paste buffer
[23:59:52 CST(-0600)] <Broady> indeed.. hehe
[00:00:08 CST(-0600)] <Broady> im an impulsive clicker/selector, and putty copies text on selection
[00:03:10 CST(-0600)] <mryall> ah yes 😊

[00:03:15 CST(-0600)] <mryall> you use Putty for IRC? 😊
[00:03:23 CST(-0600)] <mryall> HTTP tunnelling out of the office?
[00:03:27 CST(-0600)] <Broady> hehe. yeah
[00:03:37 CST(-0600)] <mryall> nice.. those were the days
[00:03:42 CST(-0600)] <Broady> authenticated http proxy -> ssh on port 443 on my vps -> screen -> irssi
[00:03:58 CST(-0600)] <mryall> hehe
[00:04:05 CST(-0600)] <mryall> yoink is a great idea for a service
[00:04:13 CST(-0600)] <mryall> but I wonder how they make money?
[00:04:27 CST(-0600)] <Broady> yeah, well... the idea isn't new, cf freecycle, reuseit etc
[00:04:34 CST(-0600)] <Broady> but freecycle is an evil enterprise
[00:09:31 CST(-0600)] <mryall> oh ok
[00:09:34 CST(-0600)] <mryall> haven't come across them before
[01:06:50 CST(-0600)] * jonmort (~jonmort@host86-141-202-101.range86-141.btcentralplus.com) has joined #atlassiandev
[01:27:05 CST(-0600)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[01:27:41 CST(-0600)] <nyerup> Hi all.
[01:29:17 CST(-0600)] <nyerup> I'm in the process of setting up Jira for my company, and I'm somewhat eager to get it to authenticate against an existing user database (Postgres).
[01:30:28 CST(-0600)] <nyerup> Googling this stuff tends to bring me lots of information about **other** projects (which coincidentally use Jira as bugtracker) and their 3rd party authentication mechanisms, but not much on Jira itself.
[01:30:46 CST(-0600)] <nyerup> Are any of you able to point me in the right direction?
[02:14:08 CST(-0600)] * Carlfish (~cmiller@epiphany.home.pastiche.org) has joined #atlassiandev
[03:44:09 CST(-0600)] <lmaslovs> @nyerup - Is importing that userbase into JIRA's dedicated SQL server a valuable option?
[03:45:02 CST(-0600)] <lmaslovs> You could write (groovy) periodically run service for this
[03:45:30 CST(-0600)] <lmaslovs> I'm not of any ready-use solutions
[03:45:36 CST(-0600)] <lmaslovs> *not aware
[03:46:12 CST(-0600)] <lmaslovs> Alternatively you could write your own authentication component - like the LDAP one
[03:47:16 CST(-0600)] <lmaslovs> o, me wrong .. take a look into
<http://www.opensymphony.com/osuser/api/com/opensymphony/user/provider/jdbc/JDBCCredentialsProvider.html>

[04:21:41 CST(-0600)] <nyerup> lmaslovs: One of my developers actually just found that one. 😊
[04:21:59 CST(-0600)] <nyerup> lmaslovs: Syncing the userbase is mainly my plan B. 😊

[04:22:02 CST(-0600)] <nyerup> .. or C. 😊
[04:42:32 CST(-0600)] <nyerup> lmaslovs: Do you know if it's documented how to replace the implementation of CredentialsProvider provided with the Jira installation with this one?
[04:43:12 CST(-0600)] <nyerup> When the developer send me the link, I thought this was an abstract class for me to extend, but it actually looks implemented. 😊
[05:05:47 CST(-0600)] * lmaslovs_ (~leonidms@80.233.159.254) has joined #atlassiandev
[05:06:11 CST(-0600)] * lmaslovs_ (~leonidms@80.233.159.254) has left #atlassiandev
[05:06:14 CST(-0600)] * lmaslovs_ (~leonidms@80.233.159.254) has joined #atlassiandev
[05:07:08 CST(-0600)] <lmaslovs_> nyerup not Sure...
[05:07:28 CST(-0600)] <lmaslovs_> You could google.com for osuser.xml and related stuff
[05:07:56 CST(-0600)] <lmaslovs_> You definitely will find LDAP configuration samples
[05:07:56 CST(-0600)] <nyerup> lmaslovs_: Alright - I'll do that. Thanks.

[05:08:17 CST(-0600)] <lmaslovs_> You are welcome 😊
[05:45:41 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[05:53:44 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[06:15:39 CST(-0600)] * MartinCleaver (~martinicle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[09:30:27 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.85) has joined #atlassiandev
[11:46:54 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:20:37 CST(-0600)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:00:18 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:17:35 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:32:10 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.85) has joined #atlassiandev
[15:48:25 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.85) has joined #atlassiandev

```
[15:54:26 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:37:44 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:37:44 CST(-0600)] * ChanServ sets mode +o tmoore
[16:47:52 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:47:52 CST(-0600)] * ChanServ sets mode +o tmoore
[16:58:03 CST(-0600)] * tmoore1 (~tmoore@59.167.164.33) has joined #atlassiandev
[17:30:21 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:30:21 CST(-0600)] * ChanServ sets mode +o tmoore
[18:11:50 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:11:50 CST(-0600)] * ChanServ sets mode +o tmoore
[18:15:53 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[18:27:17 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:37:12 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:46:30 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[21:03:09 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:11:49 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[22:11:49 CST(-0600)] * ChanServ sets mode +o tmoore
[22:45:02 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
```

atlassiandev_log-2010-02-24

```
[01:11:27 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[02:01:51 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[02:04:01 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[02:04:55 CST(-0600)] * dchui (~dchui@202.169.29.34) has left #atlassiandev
[02:06:24 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[05:59:00 CST(-0600)] * lmaslovs (~leonidms@80.233.159.254) has joined #atlassiandev
[05:59:56 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:04:37 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:40:47 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[08:39:00 CST(-0600)] * MartinCleaver (~martincle@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[09:25:51 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[09:33:21 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[09:42:33 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.85) has joined #atlassiandev
[11:18:36 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:28:30 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[11:33:21 CST(-0600)] <jdoklovic> anyone knoe if the source for jira-func-tests is available somewhere?
[11:57:54 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:09:42 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:52:38 CST(-0600)] * puskar (~puskar@PUSKAR.ES.ITS.NYU.EDU) has joined #atlassiandev
[13:38:52 CST(-0600)] * kalamon_home (~mm@chello089079130073.chello.pl) has joined #atlassiandev
[13:39:25 CST(-0600)] <kalamon_home> what's the best free irc client for windows?
[13:39:37 CST(-0600)] <kalamon_home> I am using XChat 2 but it blows
[15:14:48 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[15:18:39 CST(-0600)] <phuff> Anybody around?
[15:19:29 CST(-0600)] <phuff> I'm trying to find the Crucible project for an arbitrary repository
[15:19:41 CST(-0600)] <phuff> But it looks like I can only find a linkage between the project
[15:19:59 CST(-0600)] <phuff> 's default repository and the project, not between any repository and the project, using the APIs in a plugging.
[15:20:01 CST(-0600)] <phuff> plugin
[15:55:02 CST(-0600)] <mrdon> a project can have multiple repositories
[15:55:11 CST(-0600)] <mrdon> though I think one will be defined as the default
[15:57:14 CST(-0600)] <phuff> mrdon: Yeah, but I can't figure out how to tell what project a repository belongs to
[15:57:21 CST(-0600)] <phuff> If it's not the default repository
[15:57:24 CST(-0600)] <mrdon> well, it can belong to multiple
[15:57:42 CST(-0600)] <mrdon> can't you get all projects and iterate over their repos?
[15:57:47 CST(-0600)] <phuff> No
[15:57:51 CST(-0600)] <phuff> You can't get a list of repos
[15:57:56 CST(-0600)] <phuff> and by project
[15:57:58 CST(-0600)] <phuff> or vice versa
[15:58:27 CST(-0600)] <mrdon> there is no project.getRepos()?
[15:58:41 CST(-0600)] <phuff> Not that I can see.
[15:58:46 CST(-0600)] <phuff> there's just getDefaultRepo
[15:58:50 CST(-0600)] <mrdon> odd
[15:58:57 CST(-0600)] <mrdon> since that is a fundamental relationship
[15:59:01 CST(-0600)] <mrdon> since projects aren't just for crucible
[15:59:08 CST(-0600)] <mrdon> there is such a thing as fisheye projects
[16:00:51 CST(-0600)] <phuff> Yeah
[16:39:23 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:39:23 CST(-0600)] * ChanServ sets mode +o tmoore
[16:53:17 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[16:56:05 CST(-0600)] * bspeakmon (~bspeakmon@59.167.164.33) has joined #atlassiandev
[16:56:06 CST(-0600)] * ChanServ sets mode +o bspeakmon
[17:23:56 CST(-0600)] * myrall (~myrall@59.167.164.33) has joined #atlassiandev
[17:23:57 CST(-0600)] * ChanServ sets mode +o myrall
[17:33:24 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[17:48:39 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:48:39 CST(-0600)] * ChanServ sets mode +o tmoore
[17:49:18 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:49:18 CST(-0600)] * ChanServ sets mode +o tmoore
[18:48:57 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
```

[20:32:27 CST(-0600)] * Broady (-b@unaffiliated/broady) has joined #atlassiandev
[21:18:26 CST(-0600)] * kalamon (-kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[22:06:42 CST(-0600)] * cemerick (-la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[22:53:54 CST(-0600)] * tmoore (-tmoore@59.167.164.33) has joined #atlassiandev
[22:53:58 CST(-0600)] * ChanServ sets mode +o tmoore

atlassiandev_log-2010-02-25

[23:10:34 CST(-0600)] * jus (-justin@59.167.164.33) has joined #atlassiandev
[23:17:43 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[23:31:25 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[01:17:24 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[02:22:01 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[04:34:14 CST(-0600)] * kalamon (-kalamon@adq232.neoplus.adsl.tpgnet.pl) has joined #atlassiandev
[04:56:14 CST(-0600)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[06:13:29 CST(-0600)] * Ycros (~ycros@gnaw.yi.org) has joined #atlassiandev
[06:15:08 CST(-0600)] * jedi (mike@li55-224.members.linode.com) has joined #atlassiandev
[06:18:59 CST(-0600)] * i386 (~jdumay@59.167.164.33) has joined #atlassiandev
[07:20:13 CST(-0600)] * cemerick (-la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[07:23:56 CST(-0600)] * MartinCleaver (~martincline@76-10-164-167.dsl.teksavvy.com) has joined #atlassiandev
[08:16:10 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[08:40:25 CST(-0600)] * jdokovic (~jdokovic@66.187.202.85) has joined #atlassiandev
[11:09:20 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:49:03 CST(-0600)] * bug (~bug@ads1-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:57:50 CST(-0600)] * tmoore (-tmoore@59.167.164.33) has joined #atlassiandev
[15:57:50 CST(-0600)] * ChanServ sets mode +o tmoore
[16:16:25 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[16:19:00 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[16:31:48 CST(-0600)] <phuff> Anybody around?
[16:31:57 CST(-0600)] <tmoore> yup
[16:32:02 CST(-0600)] <phuff> Sweet.
[16:32:08 CST(-0600)] <phuff> I'm writing a fecru plugin...
[16:32:16 CST(-0600)] <phuff> And trying to save some settings.
[16:32:26 CST(-0600)] <phuff> But it'll only accept per repo-keys for settings.
[16:32:46 CST(-0600)] <phuff> Is there a good place to go to talk to somebody about that kind of problem?
[16:34:16 CST(-0600)] <phuff> Maybe not?
[16:38:04 CST(-0600)] <jdokovic> does fecru use bandana? if so you could just store settings there
[16:38:29 CST(-0600)] <phuff> I haven't seen bandana... I've seen a lot of references to SAL for storing settings.
[16:38:32 CST(-0600)] <tmoore> the fecru team recommends using PluginSettings from SAL
[16:38:34 CST(-0600)] <tmoore> yeah
[16:38:51 CST(-0600)] <phuff> But the PluginSettings per plugin key off repo ids not projects
[16:38:57 CST(-0600)] <phuff> The problem is I've got one project with like 40 repos
[16:39:01 CST(-0600)] <tmoore> ah I see
[16:39:05 CST(-0600)] <phuff> And I don't want to store the same setting 40 times
[16:39:30 CST(-0600)] <phuff> I also can't figure out a list of what projects a repo is associated with from the spi.RepoServices api
[16:39:42 CST(-0600)] <phuff> So I was trying to store things by project to try and get around that, I think.
[16:40:09 CST(-0600)] <tmoore> yeah looks like you can't pass in a project key
[16:40:24 CST(-0600)] <tmoore> but you could make a global setting with the project key in your setting name
[16:41:14 CST(-0600)] <phuff> Yeah
[16:41:19 CST(-0600)] <phuff> I thought about doing that
[16:41:36 CST(-0600)] <phuff> tmoore: Any ideas on how to find repos in a project all the projects for a repo?
[16:41:43 CST(-0600)] <phuff> tmoore: I can only figure out how to find the defaultRepo for a project
[16:42:00 CST(-0600)] <tmoore> that I'm not sure about
[16:42:27 CST(-0600)] <phuff> right now I'm thinking about just having the user configure it for my plugin (trying to beef up the auto review creator)
[16:43:12 CST(-0600)] <phuff> Thanks for your help, tmoore, I really appreciate it.
[16:43:46 CST(-0600)] <tmoore> sebr: do you have any idea about finding the repos for a project?
[16:44:10 CST(-0600)] <phuff> Or the projects for a repo...

[16:45:33 CST(-0600)] <tmoore> no problem, phuff 
[17:18:57 CST(-0600)] <sebr> tmoore: sorry dude, didn't see your ping
[17:19:42 CST(-0600)] <sebr> phuff: still need some help?
[17:19:49 CST(-0600)] <phuff> Sure.

[17:19:50 CST(-0600)] <phuff> 
[17:20:05 CST(-0600)] <phuff> I'm trying to write some auto review creator plugins
[17:20:14 CST(-0600)] <sebr> You know that a few already exist?
[17:20:25 CST(-0600)] <phuff> Yeah
[17:20:28 CST(-0600)] <phuff> I'm modifying one
[17:20:32 CST(-0600)] <sebr> ok
[17:20:32 CST(-0600)] <phuff> The example one
[17:21:06 CST(-0600)] <sebr> And what do you need help with?
[17:21:21 CST(-0600)] <phuff> but it figures out what project to put the review in by matching the repo and the project's default repo
[17:21:41 CST(-0600)] <phuff> Is there anyway to tell what projects a repo is part of and submit a review to that project?
[17:21:51 CST(-0600)] <sebr> Let me check
[17:21:56 CST(-0600)] <phuff> (We have a one project to many repos situation...)
[17:22:08 CST(-0600)] <phuff> (So we need to find the project for a repo, essentially)
[17:23:10 CST(-0600)] <sebr> You mean you want to find which projects have the repository as the default repository?
[17:23:25 CST(-0600)] <sebr> Because 1 repository can be the default for N projects
[17:24:22 CST(-0600)] <phuff> No

[17:24:27 CST(-0600)] <phuff> I mean I have one project that has 30 repos
[17:24:41 CST(-0600)] <phuff> If a commit comes in for a repo that's not the default, how do I tell what project it belongs to?
[17:24:52 CST(-0600)] <phuff> can I tell that?
[17:26:24 CST(-0600)] <sebr> give me a few minutes, i have a meeting and will need to check up on your question
[17:28:03 CST(-0600)] <phuff> Ok
[17:28:06 CST(-0600)] <phuff> no prob, sebr

[17:28:10 CST(-0600)] <phuff> I'll be around 
[17:28:13 CST(-0600)] <phuff> I appreciate your help.
[17:46:33 CST(-0600)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlasiandev
[17:48:33 CST(-0600)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlasiandev

[17:54:45 CST(-0600)] <sebr> phuff: okay, i've had a good look around, and it seems as though that information isn't available in our api 
[17:55:17 CST(-0600)] <phuff> Okay, that's what I had pretty much figured out.
[17:55:31 CST(-0600)] <phuff> So my best bet is to just manually configure a project per repo in my configuration settings?
[17:55:31 CST(-0600)] <sebr> I would suggest that you file a JIRA with that request
[17:55:41 CST(-0600)] <sebr> Yes, I think so

[17:55:47 CST(-0600)] <phuff> I committed to having my plugin done by tomorrow 
[17:55:48 CST(-0600)] <sebr> Sorry for the inconvenience
[17:55:51 CST(-0600)] <phuff> No worries.

[17:55:56 CST(-0600)] <phuff> You can only do what you can do 
[17:56:19 CST(-0600)] <sebr> If you want to file a request, here's our jira instance: jira.atlassian.com/browse/CRUC
[17:56:37 CST(-0600)] <phuff> Is that something that's actually recorded in the system?
[17:58:00 CST(-0600)] <sebr> sorry?
[17:58:28 CST(-0600)] <phuff> Like is there some place in the crucible system that actually has a relationship like that already (this repo belongs to these projects) and it's just not being served up by the API?
[17:58:40 CST(-0600)] <phuff> Or is that something that is entirely a new concept in crucible and I'm just thinking about things the wrong way.
[17:58:40 CST(-0600)] <sebr> Yes, it exists in our internal model
[17:58:47 CST(-0600)] <sebr> we just haven't exposed it
[17:58:49 CST(-0600)] <phuff> Ok
[18:06:44 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlasiandev
[18:06:45 CST(-0600)] * ChanServ sets mode +o tmoore
[20:17:57 CST(-0600)] * sebr (~sruijz@amarok/developer/sebr) has joined #atlasiandev
[20:36:03 CST(-0600)] * MartinCleaver (~martinle@76-10-151-65.dsl.teksavy.com) has joined #atlasiandev
[20:52:14 CST(-0600)] * MartinCleaver (~martinle@76-10-151-65.dsl.teksavy.com) has joined #atlasiandev
[21:23:31 CST(-0600)] <jazzy> the difficulty of supporting both project keys and repo keys in project settings is what if you have a repo and a project that have the same key?
[21:24:02 CST(-0600)] <jazzy> and, you have some sort of cross product plugin that wants to store data against both Crucible projects and FishEye repositories
[21:25:07 CST(-0600)] <jazzy> then you have a problem... so a decision was made very early on to only store data against fisheye repositories, given that fisheye already provided a way to store arbitrary data against a repository, and crucible didn't, it made sense back then
[21:25:12 CST(-0600)] <jazzy> not so much now
[21:45:09 CST(-0600)] <phuff> Ah, that makes sense jazzy.
[21:45:44 CST(-0600)] <phuff> Out of curiosity, how scalable is that storage?
[21:46:21 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlasiandev
[21:47:22 CST(-0600)] <jazzy> lol
[21:47:29 CST(-0600)] <jazzy> have a look at fisheyes config.xml and you tell me
[21:47:49 CST(-0600)] <phuff> Hah
[21:48:00 CST(-0600)] <phuff> Is that the SAL based storage?
[21:48:04 CST(-0600)] <phuff> It's stored in the .xml file?
[21:48:08 CST(-0600)] <jazzy> yep
[21:48:17 CST(-0600)] <phuff> Ahhhh...
[21:48:17 CST(-0600)] <phuff> Ok

[21:48:36 CST(-0600)] <phuff> So, I'm not going to be storing one item per crucible review in there then 
[21:48:39 CST(-0600)] <phuff> Good to know 
[21:48:45 CST(-0600)] <jazzy> the only user of SAL used to only be JIRA Studio, it was a way for us to be able to write and run the same code in all products
[21:48:56 CST(-0600)] <jazzy> so far, storing in config.xml has worked
[21:49:02 CST(-0600)] <jazzy> but it won't in the long term
[21:49:14 CST(-0600)] <jazzy> now that more people are starting to use it
[21:49:29 CST(-0600)] <jazzy> fisheye/crucible will need to add a mechanism for plugins to store data
[21:49:56 CST(-0600)] <jazzy> i would suggest, if you're not writing a cross product plugin, don't use SAL, SAL has quite a number of limitations
[21:50:13 CST(-0600)] <jazzy> though i don't know off the top of my head what other alternatives you have
[21:50:36 CST(-0600)] <phuff> Yeah, I think it's just the SAL stuff
[21:51:08 CST(-0600)] <phuff> My real problem is that I want to create a review per JIRA ticket

[21:51:09 CST(-0600)] <phuff> 
[21:51:22 CST(-0600)] <phuff> But the jira part I might just fake
[21:51:23 CST(-0600)] <phuff> -ish
[21:51:51 CST(-0600)] <phuff> We want to be able to make "Review is closed" a jira ticket workflow condition for moving to a "done" state.
[21:52:23 CST(-0600)] <jazzy> when you say a review per jira ticket, do you mean you want to have something create a review automatically when a particular jira workflow transition happens?
[21:52:38 CST(-0600)] <phuff> No, when you make a commit for ticket FOO-1234, it creates a review
[21:52:45 CST(-0600)] <phuff> And then if you make more commits against FOO-1234 it adds to the review
[21:53:11 CST(-0600)] <phuff> And then when you go to close FOO-1234, it won't move to the done state until the review for FOO-1234 is

closed.

[21:53:18 CST(-0600)] <jazzy> ah ok... on studio we've approached the problem slightly differently

[21:53:36 CST(-0600)] <phuff> I'm definiltey open to suggestions 

[21:54:05 CST(-0600)] <phuff> But that's kind of the workflow we need (to make graphically show whether or not a people are actually doing reviews before closing tickets)

[21:54:16 CST(-0600)] <jazzy> using the fisheye commit commands (

http://blogs.atlassian.com/developer/2009/10/dragon_slayer_supplement_action_issues_with_commit_commands.html)

[21:54:49 CST(-0600)] <jazzy> we have a workflow state called "Send to Review" so you commit with #review in it, and JIRA transitions the issue to that state

[21:55:09 CST(-0600)] <phuff> Ah

[21:55:13 CST(-0600)] <jazzy> then that transition has a post command that creates a review in crucible via RPC

[21:55:20 CST(-0600)] <jazzy> that part of it is currently not working

[21:55:29 CST(-0600)] <jazzy> but i can point you to the source code, it was working when i did it for fedex

[21:55:36 CST(-0600)] <phuff> Sure

[21:55:55 CST(-0600)] <phuff> That'd be great.

[21:56:20 CST(-0600)] <jazzy> the final stage was when the review is closed, the issue should be closed, but i never got to implementing that because i ran out of time, and also the version of crucible that studio is on doesn't have a review completed event

[21:56:24 CST(-0600)] <phuff> By the by, as long as I've got you around... while I'm developing my fisheye plugin, how do I make test commits against atlas-run ?

[21:56:28 CST(-0600)] <jazzy> 2.x does though

[21:56:40 CST(-0600)] <phuff> Yeah, I think we're on the latest 2.x

[21:57:23 CST(-0600)] <jazzy> i'm not that familiar with amps to be honest, we usually do such changes manually using svnkit

[21:57:51 CST(-0600)] <phuff> Ah okay

[21:58:58 CST(-0600)] <jazzy> damn... the jira studio source repository is still private... it will be public again in a week or two

[21:59:05 CST(-0600)] <phuff> no worries.

[21:59:09 CST(-0600)] <phuff> I thikn I can figure it out

[21:59:20 CST(-0600)] <phuff> I greatly appreciate your help, jazzy 

[22:00:55 CST(-0600)] <jazzy> pm me your email and i'll email you the source code for the crucible review creation stuff

[22:02:32 CST(-0600)] <jazzy> eventually this stuff will hopefully make it into the jira fisheye plugin, but don't hold your breath, like i said, we don't even have it working properly in jira studio

[22:03:23 CST(-0600)] <phuff> It should be technically possible, though, right?

[22:03:41 CST(-0600)] <jazzy> absolutely

[22:03:49 CST(-0600)] <jazzy> its just a matter of someone implementing it

[22:09:35 CST(-0600)] <phuff> Got that code.

[22:09:37 CST(-0600)] <phuff> Thanks, Jazzy.

[22:42:09 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

atlassiandev_log-2010-02-26

[04:50:38 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

[05:52:31 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev

[07:46:57 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev

[07:53:16 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev

[08:37:51 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.85) has joined #atlassiandev

[08:56:57 CST(-0600)] <phuff> Can I use the sweet ajax-y components for things like user selection in my plugin's velocity templates?

[09:03:49 CST(-0600)] * AJC_Z0 (~nnAJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev

[09:13:20 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

[09:17:46 CST(-0600)] * AJC_Z0 (~nnAJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev

[12:34:30 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.85) has joined #atlassiandev

[12:35:21 CST(-0600)] <jdoklovic> @jnolen discovered I could point idea remote debugger at jira standalone and debug the source.

[12:35:50 CST(-0600)] <jdoklovic> then i can "monkey patch" the standalone by dropping my classes into web-inf lib

[12:36:04 CST(-0600)] <jdoklovic> a lot easier than trying to get all that maven 1.x crap to work

[13:09:51 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev

[13:13:03 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev

[13:42:45 CST(-0600)] * mdoar (~mdoar@63.204.145.2) has joined #atlassiandev

[14:12:41 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev

[14:26:22 CST(-0600)] <mdoar> Ping

[14:31:47 CST(-0600)] * mdoar (~mdoar@63.204.145.2) has joined #atlassiandev

[14:32:37 CST(-0600)] * mdoar (~mdoar@63.204.145.2) has joined #atlassiandev

[14:46:51 CST(-0600)] <mdoar> ping?

[16:43:08 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

[17:07:39 CST(-0600)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev

[18:22:44 CST(-0600)] <rodogu> hi! anybody tried running intelliJ with JDK 6 64 bits in Snow Leopard? should I bother?

[19:53:18 CST(-0600)] * desultir (~d@44.84.70.115.static.exetel.com.au) has joined #atlassiandev

[19:53:40 CST(-0600)] <desultir> good afternoon!

[19:54:06 CST(-0600)] <desultir> anyone home? 

[20:01:16 CST(-0600)] <desultir> i guess we're all out having fun on a saturday

[20:01:18 CST(-0600)] <desultir> cya

[22:59:36 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

atlassiandev_log-2010-02-27

[23:12:58 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has left #atlassiandev

[23:16:16 CST(-0600)] * skrebs (~shannon@123.200.229.36) has joined #atlassiandev

[23:22:19 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev

[00:52:34 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev

[01:04:12 CST(-0600)] * skrebs (~shannon@123.200.229.36) has joined #atlassiandev
[03:36:59 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[05:50:48 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[06:20:16 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[08:08:39 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[11:57:11 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[15:35:16 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[16:08:32 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[16:35:17 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[17:00:43 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:01:26 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:10:58 CST(-0600)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[18:47:08 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[19:11:05 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[19:33:41 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:11:41 CST(-0600)] <Broady> anyone about? 

[20:11:54 CST(-0600)] <Broady> don't suppose so 
[20:43:27 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:58:26 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:13:31 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:20:56 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:37:13 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-02-28

[23:26:04 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:55:00 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[01:05:48 CST(-0600)] * Ratler (~ratler@lunar-linux/developer/ratler) has joined #atlassiandev
[01:24:36 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[01:44:18 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[02:43:06 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[04:04:59 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[04:09:18 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[04:45:42 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[05:05:31 CST(-0600)] * mryall_ (~matt@81.63.233.220.static.exetel.com.au) has joined #atlassiandev
[05:05:31 CST(-0600)] * ChanServ sets mode +o mryall_
[05:43:37 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[07:01:44 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[10:20:24 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[13:07:56 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[13:26:09 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[13:59:21 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:49:17 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[16:03:58 CST(-0600)] * Carlfish (~cmiller@12-50-239-2.att-inc.com) has joined #atlassiandev
[16:05:42 CST(-0600)] <Carlfish> j0
[16:27:30 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[16:33:48 CST(-0600)] <Broady> morning
[16:34:34 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:34:34 CST(-0600)] * ChanServ sets mode +o tmoore
[16:35:38 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:47:21 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:57:00 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[17:02:19 CST(-0600)] <mryall> morning
[17:02:34 CST(-0600)] <mryall> Hey Carlfish, how's things?
[17:28:23 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:31:13 CST(-0600)] * sebr_ (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[17:39:47 CST(-0600)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[17:40:58 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:45:31 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[18:12:24 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:12:24 CST(-0600)] * ChanServ sets mode +o tmoore
[18:32:23 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[18:33:56 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[18:45:41 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[22:09:29 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:16:53 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[22:23:21 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-01

[23:16:03 CST(-0600)] <Broady> hmm, wonder if it would be nice to be able to run a mailing list via confluence groups
[23:16:45 CST(-0600)] <Broady> well, it **would** be nice, but a good idea?
[23:17:14 CST(-0600)] <Broady> <https://plugins.atlassian.com/plugin/details/263>
[23:18:16 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[23:19:46 CST(-0600)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[00:09:53 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:37:13 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:49:47 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev

[04:44:01 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[05:07:41 CST(-0600)] * R4bid (~R4bid@192.146.11.2) has joined #atlassiandev
[05:12:20 CST(-0600)] <R4bid> hi all, I'm trying to make a JQL query to show all items I commented on
[05:12:35 CST(-0600)] <R4bid> I was hoping 4.x could do it
[05:14:42 CST(-0600)] <R4bid> I'm currently evaluating my company's upgrade from JIRA 3.x to 4.x, my current state is at "it's slower without any benefits", comments are appreciated 😊
[05:19:30 CST(-0600)] <Broady> http://jira.atlassian.com/browse/JRA-17540
[05:19:36 CST(-0600)] <Broady> vote for it!
[05:19:47 CST(-0600)] <Broady> http://jira.atlassian.com/browse/JRA-1648
[05:23:45 CST(-0600)] <R4bid> nice, txr
[07:09:43 CST(-0600)] * cemerick (~la_mer@c-75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:48:41 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:15:11 CST(-0600)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev

[08:37:14 CST(-0600)] <rodogu> just ran my unit + integration tests on Confluence 3.2-m4 and everything worked 😊 Will try Selenium later
[09:05:17 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[09:23:36 CST(-0600)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[10:25:59 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[10:33:44 CST(-0600)] * Carlfish (~cmiller@c-98-235-184-226.hsd1.pa.comcast.net) has joined #atlassiandev
[11:22:00 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:45:32 CST(-0600)] * puskar (~puskar@PUSKAR.ES.ITS.NYU.EDU) has joined #atlassiandev
[12:11:37 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[12:36:11 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.144) has joined #atlassiandev
[12:36:20 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[12:36:29 CST(-0600)] <jdoklovic> @jnolen ping....
[12:49:36 CST(-0600)] * Carlfish (~cmiller@12-50-239-2.att-inc.com) has joined #atlassiandev
[13:38:56 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[14:10:47 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:38:58 CST(-0600)] * bug_ (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[14:51:05 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.144) has joined #atlassiandev
[14:55:30 CST(-0600)] * Carlfish (~cmiller@12-50-239-2.att-inc.com) has joined #atlassiandev
[15:20:15 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[16:03:36 CST(-0600)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:14:03 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:52:44 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:52:44 CST(-0600)] * ChanServ sets mode +o tmoore
[16:55:56 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[17:13:18 CST(-0600)] * JonathanNolen_ (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[17:23:33 CST(-0600)] <rodogu> (Confluence)
[17:25:19 CST(-0600)] <rodogu> oh boy...
[17:26:58 CST(-0600)] <rodogu> ok, found it:
[17:26:59 CST(-0600)] <rodogu> AbstractEntityQueryParser
[17:27:40 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[17:28:49 CST(-0600)] * Carlfish (~cmiller@c-98-235-184-226.hsd1.pa.comcast.net) has joined #atlassiandev
[17:29:01 CST(-0600)] <Carlfish> I like cheese!
[17:35:15 CST(-0600)] * twong_ (~twong@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[17:35:37 CST(-0600)] * JonathanNolen_ (~jnolen@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[17:36:44 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:36:45 CST(-0600)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:56:18 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:56:18 CST(-0600)] * ChanServ sets mode +o tmoore
[18:04:32 CST(-0600)] <mryall> rodogu: no membership queries in atlassian-user, unfortunately
[18:04:44 CST(-0600)] <mryall> they've been deprecated in 3.2, and will be removed (replace with Embedded Crowd) in 3.3
[18:05:16 CST(-0600)] <mryall> for the moment, you have to use the GroupManager.getMembers(Group), or whatever it is
[18:06:39 CST(-0600)] * Carlfish (~cmiller@c-98-235-184-226.hsd1.pa.comcast.net) has joined #atlassiandev
[18:18:35 CST(-0600)] <rodogu> @mryall did it ever worked?
[18:19:17 CST(-0600)] <rodogu> as per the code, it seems it was just good intentions 😊
[18:20:10 CST(-0600)] * Carlfish (~cmiller@c-98-235-184-226.hsd1.pa.comcast.net) has joined #atlassiandev

[18:20:25 CST(-0600)] <rodogu> When I saw those queries I thought it was too good to be true 😊 so yeah, I'll deal with GroupManager membership methods
[18:32:17 CST(-0600)] * sebr (~seb@amarok/developer/sebr) has joined #atlassiandev
[18:41:13 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:12:51 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:33:41 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[19:41:04 CST(-0600)] <mryall> it has never worked, unfortunately
[19:41:11 CST(-0600)] <mryall> I only got around to reviewing that code last month
[20:22:48 CST(-0600)] <Broady> anyone going to ignite tonight?
[20:22:57 CST(-0600)] <Broady> mryall: ?
[20:31:23 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:59:45 CST(-0600)] <tmoore> Broady: where/when is it?
[21:00:51 CST(-0600)] <Broady> tonight
[21:00:56 CST(-0600)] <Broady> erm... let me see. (i should know)
[21:01:08 CST(-0600)] <Broady> oxford art factory
[21:01:14 CST(-0600)] <Broady> 5.30 to start at 6.30
[21:10:02 CST(-0600)] * JonathanNolen_ (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[21:35:42 CST(-0600)] <mryall> erm, ignite?
[21:35:50 CST(-0600)] <mryall> is that one of them user group thingies?
[21:37:26 CST(-0600)] <Broady> kinda like webjam

[21:37:28 CST(-0600)] <Broady> except worse
[21:37:34 CST(-0600)] <Broady> but still good (i think)
[21:37:42 CST(-0600)] <Broady> soren is talking!
[21:38:02 CST(-0600)] <dezwart> Broady: do you have a URL for the proceedings?
[21:38:29 CST(-0600)] <Broady> <http://www.ignitesydney.com/speakers/>
[21:38:39 CST(-0600)] <dezwart> Thanks mate.
[21:38:48 CST(-0600)] <Broady> there are no tix left but i'm sure you could just rock up and talk your way in - also arthur lee (BIT intern) has at least one spare
[21:39:31 CST(-0600)] <dezwart> Nah, I've got D&D on tonight; but having a link to Soren's blog is win.

atlassiandev_log-2010-03-02

[23:29:15 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[23:51:50 CST(-0600)] * JonathanNolen_ (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[00:07:09 CST(-0600)] * atlasbot (~PircBot@63-246-22-217.contegix.com) has joined #atlassiandev
[00:07:09 CST(-0600)] * Topic is 'LF32M. pst nerdscore and achievements | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#039;> set by Carlfish!~cmiller@epiphany.home.pastiche.org on 2010-02-15 18:20:08 CST(-0600)
[00:30:19 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[01:07:04 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:51:58 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[03:58:37 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[06:23:28 CST(-0600)] * MartinCleaver (~martincline@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[08:03:01 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[08:48:39 CST(-0600)] * jonmort (~jonmort@195.27.20.107) has joined #atlassiandev
[11:17:46 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:19:05 CST(-0600)] * MartinCleaver (~martincline@32.60.108.73) has joined #atlassiandev
[12:27:08 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[12:34:09 CST(-0600)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:34:10 CST(-0600)] * ChanServ sets mode +o bspeakmon
[14:33:10 CST(-0600)] * puskar (~puskar@PUSKAR.ES.ITS.NYU.EDU) has joined #atlassiandev
[14:33:38 CST(-0600)] <puskar> does the openldap ldapsearch work with LDAPS or just StartTLS?
[15:00:41 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:33:25 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[15:33:25 CST(-0600)] * ChanServ sets mode +o tmoore
[16:14:43 CST(-0600)] * pharkmillups (~mphilips@206.83.89.3.ptr.us.xo.net) has joined #atlassiandev
[16:19:00 CST(-0600)] <Broady> mryall: ping
[16:19:38 CST(-0600)] * pharkmillups (~mphilips@206.83.89.3.ptr.us.xo.net) has left #atlassiandev
[16:24:00 CST(-0600)] * tmoore1 (~tmoore@59.167.164.33) has joined #atlassiandev
[16:55:49 CST(-0600)] <JonathanNolen> @rodogu - you there?
[17:08:33 CST(-0600)] <mryall> back
[17:08:37 CST(-0600)] <mryall> what's up, Broady?
[17:29:13 CST(-0600)] <rodogu> Here
[18:53:06 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:01:41 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev

atlassiandev_log-2010-03-03

[23:49:42 CST(-0600)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[00:35:39 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:50:56 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[03:16:00 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[03:39:29 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[04:35:18 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[05:45:45 CST(-0600)] * mm_ (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[06:40:16 CST(-0600)] * bspeakmon (~bspeakmon@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[06:40:24 CST(-0600)] * JonathanNolen_ (~jnolen@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[06:45:39 CST(-0600)] * bspeakmon_ (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[08:25:18 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:38:57 CST(-0600)] * cemerick (~la_mer@pool-72-79-222-128.spfdma.east.verizon.net) has joined #atlassiandev
[08:55:29 CST(-0600)] * cemerick (~la_mer@pool-72-79-222-128.spfdma.east.verizon.net) has joined #atlassiandev
[10:08:15 CST(-0600)] * MartinCleaver_ (~martincline@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[10:54:30 CST(-0600)] <MartinCleaver_> Is there a way, in the browser, to remove from text?
[10:55:00 CST(-0600)] <MartinCleaver_> or, for that matter, all colour from text
[11:03:15 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:55:55 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[12:17:43 CST(-0600)] * JonathanNolen_ (~jnolen@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[12:30:19 CST(-0600)] * jonmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[12:54:37 CST(-0600)] * ChanServ sets mode +o bspeakmon
[14:31:25 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:17:07 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:17:07 CST(-0600)] * ChanServ sets mode +o tmoore
[16:49:58 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[17:42:38 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[18:24:35 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:31:50 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[20:34:12 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:34:38 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-04

[23:10:54 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[23:27:34 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[00:08:18 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[02:46:53 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[03:33:04 CST(-0600)] * jmort (~Adium@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[04:16:41 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[04:46:26 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[05:17:57 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:33:16 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[08:02:26 CST(-0600)] * MartinCleaver (~martincline@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[09:07:51 CST(-0600)] * MartinCleaver (~martincline@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[09:48:47 CST(-0600)] * MartinCleaver (~martincline@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[13:16:09 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[14:45:34 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:56:26 CST(-0600)] * IgorMinar (~Adium@nat/sun/x-jphgdrprrgbatrh) has joined #atlassiandev
[15:45:14 CST(-0600)] <IgorMinar> hi there, is here anyone who could find the diff for confluence issue that was fixed recently? I'm debugging an complicated issue that might have been caused by this change
[15:55:33 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[15:55:33 CST(-0600)] * ChanServ sets mode +o tmoore
[16:06:11 CST(-0600)] <bspeakmon> igor: which issue?
[16:06:35 CST(-0600)] <IgorMinar> this one: <http://jira.atlassian.com/browse/CONF-14989>
[16:07:50 CST(-0600)] <IgorMinar> I'm curious whether the DefaultHibernateUser stored in the session was really replaced with detached DefaultUser version
[16:09:01 CST(-0600)] <skrebs> is this related to <http://jira.gliffy.com/browse/GLIFFY-1883> by any chance?
[16:09:13 CST(-0600)] <IgorMinar> yes it is 😊
[16:09:53 CST(-0600)] <IgorMinar> skrebs: are you affected too?
[16:10:18 CST(-0600)] <bspeakmon> I'm checking our fisheye. hope andrew was kind enough to mark the commit 😊
[16:10:34 CST(-0600)] <IgorMinar> bspeakmon: thanks
[16:12:15 CST(-0600)] <bspeakmon> okay, confluence itself didn't change, the fix was in our hibernate 2 fork
[16:12:18 CST(-0600)] <skrebs> I'm not affected, but interested to see the outcome incase we need to update the gliffy code. I help out with the Gliffy Confluence plugin in my spare time.
[16:14:56 CST(-0600)] <IgorMinar> @bspeakmon I see, thanks a lot. I modified our custom authenticator based on the discussion with Matt and Andrew and that seems to have resulted in a new bug that affects gliffy (and gliffy only for some weird reason).
[16:15:42 CST(-0600)] <IgorMinar> I thought that they would make the same change in the default authenticator, but as I suspected when running diffs between 3.0 and 3.1.1 code, they didn't.
[16:15:42 CST(-0600)] <bspeakmon> gimme a sec and I'll find the diffs in there 😊
[16:17:24 CST(-0600)] <bspeakmon> weird
[16:17:49 CST(-0600)] <IgorMinar> @skrebs oh, ok. I still don't understand why only gliffy can trigger the bug even though the bug is in what I suspect is a frequently used confluence api (but occurs only if DefaultUser is put into the http session instead of DefaultHibernateUser)
[16:20:45 CST(-0600)] <bspeakmon> after reading both of the issues I can honestly say that I have no idea wtf is going on 😊
[16:21:22 CST(-0600)] <IgorMinar> based on CONF-14989 I modified our authenticator to store user object in the http session like this:
[16:22:01 CST(-0600)] <IgorMinar> User disconnectedUser = // more complicated but basically comes from:
userAccessor.getUser(username);
[16:24:13 CST(-0600)] <IgorMinar> disconnectedUser.setFullName(userFromDb.getFullName());
[16:24:13 CST(-0600)] <IgorMinar> disconnectedUser.setEmail(userFromDb.getEmail());
[16:24:13 CST(-0600)] <IgorMinar> ...
[16:24:13 CST(-0600)] <IgorMinar> request.getSession()
[16:24:14 CST(-0600)] <IgorMinar> .setAttribute(DefaultAuthenticator.LOGGED_IN_KEY, disconnectedUser);
[16:26:22 CST(-0600)] <IgorMinar> and when gliffy calls permissionManager.hasPermission(user, permission, diagram);
[16:26:52 CST(-0600)] <IgorMinar> and the permission is not cached resulting in a hibernate query
[16:27:33 CST(-0600)] <IgorMinar> it comes down to HibernateGroupManager#getAllGroupsForUser which calls isUserExternal(user)
[16:27:50 CST(-0600)] <IgorMinar> this check looks like: !(user instanceof DefaultHibernateUser)
[16:28:07 CST(-0600)] <IgorMinar> and the user is incorrectly identified as external user
[16:28:36 CST(-0600)] <IgorMinar> this results in an incorrect hql/sql to be generated, which returns 0 groups and that results in permission denied for the user
[16:29:02 CST(-0600)] <IgorMinar> as I said, it's pretty complicated 😊
[16:30:10 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[16:30:14 CST(-0600)] <IgorMinar> I guess the right thing to do is to revert back to using DefaultHibernateUser in the http session for now
[16:30:39 CST(-0600)] <IgorMinar> but I'm puzzled about the fact that only gliffy can trigger this
[16:31:21 CST(-0600)] <IgorMinar> (as far as we know)
[16:32:09 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[16:33:55 CST(-0600)] <bspeakmon> sounds like a myrall question, honestly
[16:34:02 CST(-0600)] <bspeakmon> either him or andrew
[16:35:11 CST(-0600)] <IgorMinar> I'll experiment a bit now that I know that the default authenticator was not modified as discussed with matt and andrew. Thanks for that info
[16:38:08 CST(-0600)] <bspeakmon> well, no commit tied to those issues changes the authenticator.
[16:38:23 CST(-0600)] <bspeakmon> they could have changed something without marking it. I can look at the specific file if you want
[16:39:24 CST(-0600)] <IgorMinar> I compared the diffs between 3.0 and 3.1.1 and didn't see the change I expected either. So I just wanted to make sure that I didn't miss it somehow.
[17:15:41 CST(-0600)] <IgorMinar> I have a fix. Authenticators can't store DefaultUser in the http session. Creating a disconnected DefaultHibernateUser clone and storing that one resolved the issue. Thanks bspeakmon.
[17:16:23 CST(-0600)] <bspeakmon> me? I did nothing 😊

[17:17:29 CST(-0600)] <IgorMinar> heh.. you confirmed that your authenticator was not modified as a result of the confluence issue I mentioned. that was enough to send me the right direction
[17:23:24 CST(-0600)] <bspeakmon> alrighly then
[17:32:47 CST(-0600)] <IgorMinar> I noticed that 3.1.2 source code is not available for download. Can someone please push it there?
[17:37:29 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[17:46:20 CST(-0600)] <bspeakmon> you'll get a faster response if you go through support for that one
[17:47:23 CST(-0600)] <IgorMinar> I'll do that. thnx
[18:02:39 CST(-0600)] <jazzy> morning all
[18:03:08 CST(-0600)] * jazzy is waiting for idea to update its indicies before it will show me a call hierachy
[18:03:55 CST(-0600)] * jazzy installs ubuntu security updates
[18:05:10 CST(-0600)] <tmoore> haha yeah isn't the background index update so awesome? rolls eyes
[18:16:56 CST(-0600)] <mryall> morning
[18:24:46 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:32:37 CST(-0600)] <jazzy> i think i've encountered a memory leak in idea... idea was searching for references to a constructor, and it ran out of memory. so i restarted, and then tried to search for references to that constructor again, and watched as memory usage climbed, until it ran out of memory.
[18:41:59 CST(-0600)] <mryall> sounds like you should raise a bug
[18:43:49 CST(-0600)] <jazzy> for some reason i can't get a heap dump
[18:44:10 CST(-0600)] * jazzy switches to a sun jdk
[18:44:21 CST(-0600)] <bspeakmon> well there's your problem
[18:44:29 CST(-0600)] <jazzy> i'm using OpenJDK
[18:44:39 CST(-0600)] <jazzy> there's no reason why it shouldn't work
[18:44:47 CST(-0600)] <jazzy> as they are the same source code base
[18:45:20 CST(-0600)] <mryall> for the JDK - what about the JVM implementation?
[18:46:07 CST(-0600)] <jazzy> i thought they were the same, Sun released HotSpot under GPL, and OpenJDK is where development is done on it
[18:48:18 CST(-0600)] <jazzy> <http://openjdk.java.net/groups/hotspot/>
[18:48:30 CST(-0600)] <jazzy> hotspot (sun jvm) is part of OpenJDK
[18:51:01 CST(-0600)] <bspeakmon> and yet, all released sun jdk's are pre-openjdk.
[18:51:02 CST(-0600)] <bspeakmon> <http://www.sun.com/software/opensource/java/faq.jsp#b10>
[18:51:47 CST(-0600)] <bspeakmon> meaning, basically, that openjdk is not to be trusted, especially in light of the fact that sun is in no hurry to release an official jdk from it.
[18:53:26 CST(-0600)] <jazzy> well, JDK 7 will be released from it
[18:53:49 CST(-0600)] <bspeakmon> ain't gonna be no trial – er, JDK 7
[18:54:02 CST(-0600)] <bspeakmon> (my prediction, not a confirmed fact)
[18:54:12 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[18:54:13 CST(-0600)] <bspeakmon> point is, if you need to run something on java today, you use a sun jdk.
[18:54:24 CST(-0600)] <bspeakmon> or ibm if you're stuck in websphere land
[18:55:18 CST(-0600)] <jazzy> openjdk was forked from the stable JDK 6 source code... it's not like its the new thing that they've written from scratch
[18:55:53 CST(-0600)] <jazzy> or rather, the JDK trunk became OpenJDK, and JDK 6 stable branch was forked off of it
[18:56:18 CST(-0600)] <jazzy> so OpenJDK has its roots in a working, production JDK 6
[18:56:32 CST(-0600)] <bspeakmon> I know these things
[18:56:51 CST(-0600)] <jazzy> so while i wouldn't use openjdk in production, that doesn't mean it shouldn't work
[18:57:12 CST(-0600)] <bspeakmon> our experience has been that it doesn't work.
[18:57:19 CST(-0600)] <bspeakmon> for servers or clients
[19:02:30 CST(-0600)] <jazzy> the only problem i've found with it in the past is it has difficulty detecting out of memory errors, it will spin for ages doing nothing but garbage collection and then throw an error saying its spending too long doing garbage collection, not that its out of heap space like the sun jdk will
[19:03:30 CST(-0600)] <jazzy> its supposed to throw an error saying its out of heap space when it finds its reclaiming too little memory from garbage collection
[19:03:39 CST(-0600)] <jazzy> but it wasn't doing that
[19:05:20 CST(-0600)] <jazzy> hey cool, my wifes bringin home an OLPC laptop tonight
[19:05:25 CST(-0600)] * jazzy is going to have some fun
[19:08:13 CST(-0600)] <jazzy> sun jdk having the same problem with jmap
[19:18:31 CST(-0600)] <mryall> bug bug bug
[19:19:38 CST(-0600)] <bspeakmon> bug
[19:20:10 CST(-0600)] <jazzy> quiet already, i've reported it
[19:21:39 CST(-0600)] <jazzy> ok, i've managed to get a heap dump... question... are there any problems with sending a heap dump of an IDE loaded with our source code to a competitor?
[19:23:45 CST(-0600)] <skrebs> they could always buy a license if they wanted to read the source code :-/
[19:24:18 CST(-0600)] <jazzy> true, but they could potentially find out about features we're implementing that we haven't released yet
[19:26:37 CST(-0600)] <jazzy> though... they could probably just check JAC for that
[19:32:51 CST(-0600)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has joined #atlassiandev
[19:51:33 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[20:17:58 CST(-0600)] * IgorMinar (~Adium@nat/sun/x-nxbhlnzpknfhzxj) has joined #atlassiandev
[20:54:58 CST(-0600)] <mryall> jazzy: I'd wait until they ask for it - could be they know about the problem already

atlassiandev_log-2010-03-05

[23:48:00 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[00:43:38 CST(-0600)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[02:29:09 CST(-0600)] * jmort (~jonmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[06:24:51 CST(-0600)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[06:24:51 CST(-0600)] * ChanServ sets mode +o tmoore
[06:53:14 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[06:56:28 CST(-0600)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[08:04:46 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[08:21:23 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev

[09:14:30 CST(-0600)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[09:42:44 CST(-0600)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[10:12:05 CST(-0600)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has joined #atlassiandev
[10:58:28 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[11:23:10 CST(-0600)] * IgorMinar (~Adium@nat/sun/x-kezcxapfshdqava) has joined #atlassiandev
[12:59:18 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[13:20:44 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[13:24:32 CST(-0600)] * IgorMinar (~Adium@nat/sun/x-kezcxapfshdqava) has left #atlassiandev
[14:10:13 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:45:43 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[16:15:24 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[16:15:29 CST(-0600)] <phuff> jazz: you around?
[17:06:57 CST(-0600)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:25:49 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:02:35 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[18:42:10 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[18:55:47 CST(-0600)] <phuff> Anybody around?
[19:25:14 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[20:53:16 CST(-0600)] * cemerick (~la_mer@64.30.48.43) has joined #atlassiandev
[20:55:35 CST(-0600)] * cemerick (~la_mer@64.30.48.43) has joined #atlassiandev
[21:20:34 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:48:43 CST(-0600)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev

atlassiandev_log-2010-03-06

[23:46:20 CST(-0600)] * cemerick (~la_mer@64.30.48.43) has joined #atlassiandev
[23:56:19 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[00:02:10 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:03:35 CST(-0600)] * mm_ (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[05:08:04 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[05:20:04 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[06:42:13 CST(-0600)] * cemerick (~la_mer@64.30.48.43) has joined #atlassiandev
[08:39:40 CST(-0600)] * cemerick (~la_mer@64.30.48.43) has joined #atlassiandev
[08:45:08 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[09:19:54 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[10:37:59 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[12:35:24 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[12:52:28 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[13:58:55 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[14:24:04 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[15:12:33 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:33:01 CST(-0600)] * cemerick (~la_mer@64.30.48.43) has joined #atlassiandev
[15:39:49 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[16:17:30 CST(-0600)] * cemerick (~la_mer@64.30.48.43) has joined #atlassiandev
[17:38:32 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[18:56:59 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[20:44:04 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:49:58 CST(-0600)] * cemerick (~la_mer@64.30.48.43) has joined #atlassiandev
[22:12:27 CST(-0600)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[22:12:27 CST(-0600)] * ChanServ sets mode +o tmoore
[22:35:30 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:53:10 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

atlassiandev_log-2010-03-07

[01:16:34 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:02:09 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:04:24 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[11:50:00 CST(-0600)] * bug (~bug@adsl-76-211-225-144.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[12:32:08 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[13:11:39 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[14:32:08 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:12:24 CST(-0600)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:52:26 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[15:52:26 CST(-0600)] * ChanServ sets mode +o tmoore
[16:19:21 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[17:36:52 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[17:56:13 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:25:45 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-08

[23:43:32 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:47:39 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[07:04:35 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:56:29 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[07:59:06 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[08:38:34 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

[09:14:53 CST(-0600)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[09:43:40 CST(-0600)] * MartinCleaver (~martin cleaver@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[09:45:49 CST(-0600)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[09:47:00 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[10:00:23 CST(-0600)] * bug (~bug@adsl-71-135-165-31.dsl.pltn13.pacbell.net) has joined #atlassiandev
[10:04:51 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[10:34:48 CST(-0600)] <phuff> Anybody around?
[10:34:51 CST(-0600)] <phuff> jazzy?
[12:07:01 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[12:26:07 CST(-0600)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:26:07 CST(-0600)] * ChanServ sets mode +o bspeakmon
[13:00:03 CST(-0600)] <phuff> Anybody know if I can use json in a fecru plugin relatively easily?
[14:41:58 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:54:01 CST(-0600)] * bug (~bug@adsl-71-135-166-108.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:08:35 CST(-0600)] * bug (~bug@adsl-71-135-166-108.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:32:43 CST(-0600)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:03:58 CST(-0600)] * pipegeek (~pipegeek@nyc01.limewire.com) has joined #atlassiandev
[16:04:37 CST(-0600)] * jcostello (~jcostello@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:06:49 CST(-0600)] <pipegeek> I'm currently trying to use bamboo with git, and almost succeeding. I'm using the latest stable version of the plugin available at <http://github.com/krosenvold/bgit/>. I'm using a remote build agent.
[16:09:08 CST(-0600)] <pipegeek> The problem I'm encountering is that **sometimes**, triggering a build from the REST api fails. An edu.nyu.cs.javagit.api.JavaGitException is logged, with the message "Error calling git-reset. The git-reset error message: { line1=\[\], line2=[HEAD is now at 70646d8... foo] }"
[16:09:19 CST(-0600)] <pipegeek> It looks an awful lot like the git plugin might be misparsing git output
[16:09:34 CST(-0600)] <pipegeek> does anyone have any experience with the bamboo git plugin?
[16:14:52 CST(-0600)] * bug (~bug@adsl-71-135-166-108.dsl.pltn13.pacbell.net) has joined #atlassiandev
[16:28:08 CST(-0600)] <pipegeek> sadly, upgrading git on the machine is not an option
[16:29:50 CST(-0600)] <bspeakmon> might be support request time
[16:31:47 CST(-0600)] <pipegeek> not sure the git plugin is supported
[16:35:18 CST(-0600)] <bspeakmon> still worth a try.
[16:35:26 CST(-0600)] <bspeakmon> and I'm not sure it isn't supported.
[16:47:15 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[16:50:34 CST(-0600)] <jazzy> hmm... opening gadgets pom.xml as a project in IDEA makes IDEA do silly things, like create a directory called \${project.build.directory} and ask if you want to add it to subversion
[16:51:37 CST(-0600)] <mrdon> pipegeek: yes, I've done some with the git plugin on bamboo
[16:51:46 CST(-0600)] <mrdon> I wrote the first version, then used the new version recently
[16:52:14 CST(-0600)] <mrdon> no options if you can't upgrade git though, afaiak
[16:52:36 CST(-0600)] <mrdon> you could always fork and fix the output parsing
[16:53:42 CST(-0600)] <pipegeek> mrdon: may be what we end up doing
[16:54:01 CST(-0600)] <pipegeek> but you believe that's the problem?
[16:55:41 CST(-0600)] <mrdon> that'd be my guess
[16:59:03 CST(-0600)] * bug (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev

[16:59:11 CST(-0600)] <pipegeek> incidentally, not important at all, but src/main/resources/atlassian-plugin.xml needs updating 
[16:59:18 CST(-0600)] <pipegeek> I'll submit a patch
[17:10:09 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:11:05 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:11:05 CST(-0600)] * ChanServ sets mode +o tmoore
[17:35:44 CST(-0600)] <pipegeek> mrdon: we've got a bet going here.... do you happen to know if the current maintainer, krosenvold, an atlassian employee?
[17:38:30 CST(-0600)] <mrdon> he isn't
[17:38:36 CST(-0600)] <mrdon> at least not that I know of
[17:38:40 CST(-0600)] <pipegeek> fooy
[17:40:27 CST(-0600)] <dezwart> Anyone in here have experience with Javascript + SOAP/XMLRPC?
[17:59:01 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:19:48 CST(-0600)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:33:17 CST(-0600)] <mrdon> xmlrpc yes, soap no
[19:51:07 CST(-0600)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[20:02:28 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:32:31 CST(-0600)] <tmoore> dezwart: my experience with JavaScript + SOAP/XMLRPC is that you don't want to do it :-P
[20:33:06 CST(-0600)] <tmoore> XML-RPC will be easier than SOAP though. You can find some free libraries out there if you Google for it.
[20:48:24 CST(-0600)] * dezwart (~pdzwart@ppp121-44-113-124.lns20.synd6.internode.on.net) has joined #atlassiandev
[20:52:43 CST(-0600)] * dezwart (~pdzwart@ppp121-44-113-124.lns20.synd6.internode.on.net) has left #atlassiandev
[20:56:41 CST(-0600)] <mrdon> I used xml-rpc in client-side js back in the day
[20:56:48 CST(-0600)] <mrdon> circa '02 or so
[20:56:53 CST(-0600)] <mrdon> was quite happy with it
[21:46:56 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:50:18 CST(-0600)] * JonathanNolen_ (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-03-09

[00:09:22 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:45:28 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[03:38:31 CST(-0600)] * inoX1 (~bethke@212.51.18.54) has joined #atlassiandev
[03:40:08 CST(-0600)] <inoX1> Hey, I am currently trying to create a some soap client for jira but usually I use ant, so maven is a whole new world :S ...however, right now I got stuck because I have no idea where to get the dependencies from the com.atlassian.jira.rpc.soap.client package
[03:44:35 CST(-0600)] * inoX1 (~bethke@212.51.18.54) has left #atlassiandev
[03:44:42 CST(-0600)] * inoX1 (~bethke@212.51.18.54) has joined #atlassiandev
[04:11:55 CST(-0600)] * inoX1 (~bethke@212.51.18.54) has left #atlassiandev

[04:56:32 CST(-0600)] * inoX1 (~bethke@212.51.18.54) has joined #atlassiandev
[04:57:10 CST(-0600)] <inoX1> Is there anyone who can help me with a dependency problem?
[05:45:37 CST(-0600)] * inoX1 (~bethke@212.51.18.54) has joined #atlassiandev
[07:03:20 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[08:49:56 CST(-0600)] * inoX1 (~bethke@212.51.18.54) has joined #atlassiandev
[08:50:21 CST(-0600)] <inoX1> Hey can anyone help me understanding the development on the soap rpc api?
[08:50:52 CST(-0600)] <inoX1> I always used ant and I am somehow lost with maven...
[09:02:54 CST(-0600)] <inoX1> This IRC Channel is like a bad joke....check the log... ...without anyone answering it is just a waste of time for people who really need help.
[09:02:56 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[09:03:57 CST(-0600)] * inoX1 (~bethke@212.51.18.54) has left #atlassiandev
[09:05:10 CST(-0600)] * MartinCleaver (~martincle@66-207-222-14.beanfield.net) has joined #atlassiandev
[09:09:14 CST(-0600)] * trochej (trochej@noches.pl) has joined #atlassiandev
[09:20:33 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.84) has joined #atlassiandev
[09:21:08 CST(-0600)] <jdoklovic> has anyone played with op4j ? looks pretty cool.... http://www.op4j.org
[09:49:01 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[11:08:29 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:43:39 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[11:44:18 CST(-0600)] <jdoklovic> FYI: http://confluence.atlassian.com/display/DEVNET/Plugins+That+Cannot+Be+Reloaded+with+pi
[13:10:35 CST(-0600)] <bspeakmon> if inoX1 had bothered to check the clock, he would have learned that he asked his question while both sydney and sf were asleep.
[13:10:55 CST(-0600)] <bspeakmon> jdoklovic: +1
[14:24:50 CST(-0600)] * jdoklovic (~jdoklovic@71-210-161-132.mpls.qwest.net) has joined #atlassiandev
[14:29:13 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[14:54:48 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.84) has joined #atlassiandev
[15:13:57 CST(-0600)] * MartinCleaver (~martincle@CPE0018f8f3ed5d-CM000a735cc904.cpe.net.cable.rogers.com) has joined #atlassiandev
[15:47:25 CST(-0600)] * MartinCleaver (~martincle@CPE0018f8f3ed5d-CM000a735cc904.cpe.net.cable.rogers.com) has joined #atlassiandev
[16:36:33 CST(-0600)] * MartinCleaver (~martincle@CPE0018f8f3ed5d-CM000a735cc904.cpe.net.cable.rogers.com) has joined #atlassiandev
[16:47:03 CST(-0600)] * MartinCleaver (~martincle@74.198.8.70) has joined #atlassiandev
[17:05:45 CST(-0600)] * MartinCleaver (~martincle@CPE0018f8f3ed5d-CM000a735cc904.cpe.net.cable.rogers.com) has joined #atlassiandev
[17:12:39 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:12:39 CST(-0600)] * ChanServ sets mode +o tmoore
[17:40:39 CST(-0600)] * MartinCleaver (~martincle@CPE0018f8f3ed5d-CM000a735cc904.cpe.net.cable.rogers.com) has joined #atlassiandev
[17:41:36 CST(-0600)] * MartinCleaver (~martincle@CPE0018f8f3ed5d-CM000a735cc904.cpe.net.cable.rogers.com) has left #atlassiandev
[18:07:00 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:11:21 CST(-0600)] <pipegeek> moral of my story is, don't use horrible ancient git with bamboo
[19:53:03 CST(-0600)] * pipegeek (~pipegeek@nyc01.limewire.com) has left #atlassiandev
[20:12:46 CST(-0600)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[20:25:40 CST(-0600)] * tmoore1 (~tmoore@59.167.164.33) has joined #atlassiandev
[20:56:02 CST(-0600)] <dchui> is there a max length for content properties? i'm reading the PropertySet.hbm.xml file and it seems to suggest so (~97KB)
[20:56:35 CST(-0600)] <dchui> I mean PropertysetItem.hbm.xml
[21:50:06 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:05:45 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-10

[00:16:23 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:57:31 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[01:59:29 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[02:17:51 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[03:07:49 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[07:22:46 CST(-0600)] * trochej (trochej@noches.pl) has joined #atlassiandev
[07:49:26 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[08:01:30 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev
[08:17:19 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[08:18:52 CST(-0600)] * phuff (~phuff@c-71-195-248-102.hsd1.ut.comcast.net) has joined #atlassiandev
[08:18:58 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[08:19:35 CST(-0600)] <phuff> Hey.
[08:19:48 CST(-0600)] <phuff> Anybody know if I can use those fancy user selection widgets in Crucible in a plugin?
[08:23:59 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[08:43:43 CST(-0600)] <phuff> Anybody know if I can link a review to a jira ticket programmatically from a plugin?
[08:48:38 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[09:00:27 CST(-0600)] <phuff> Also, anybody know how to test an eventlistener plugin for fecru?
[09:06:30 CST(-0600)] * bug (~bug@64.134.23.118) has joined #atlassiandev
[10:55:18 CST(-0600)] * bug (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev
[11:05:44 CST(-0600)] <phuff> Anybody around that can answer questions?
[11:42:37 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[13:52:24 CST(-0600)] * IgorMinar (~Adium@nat/sun/x-mijhfmdstsrlact) has joined #atlassiandev
[13:56:09 CST(-0600)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[14:48:45 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:00:38 CST(-0600)] * jdoklovic (~jdoklovic@66.187.202.84) has joined #atlassiandev
[15:01:27 CST(-0600)] <jdoklovic> I can get struts2 + sitemesh + velocity working in a really simple webapp, but when i try to switch it to freemarker, it doesn't work..... anyone have any suggestions?
[15:02:47 CST(-0600)] <jdoklovic> just have StrutsPrepare, FreeMarkerPageFilter, and StrutsExecute filters in web.xml and a simple .jsp and

a simple .ftl decorator

[15:03:47 CST(-0600)] <bspeakmon> not a freemarker expert.

[15:09:43 CST(-0600)] <MartinCleaver> me neither

[15:10:34 CST(-0600)] <bspeakmon> doesn't struts 2 have some kind of built-in freemarker thingy?

[15:11:18 CST(-0600)] <jdoklovic> you mean like org.apache.struts2.sitemesh.FreeMarkerPageFilter

[15:12:01 CST(-0600)] <bspeakmon> sure 

[15:12:15 CST(-0600)] <bspeakmon> don would know, he perpetrated struts 2

[15:12:20 CST(-0600)] <jdoklovic> yeah, got that.... followed all the examples, but they don't seem to work

[15:30:19 CST(-0600)] <jdoklovic> in case anyone wants to see my configuration....
<http://stackoverflow.com/questions/2420752/struts2-sitemesh-freemarker-doesnt-work>

[16:05:28 CST(-0600)] * Topic is '!! I THINK THE FREEZER DESERVES A LIGHT AS WELL!! | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts'>; set by bspeakmon on 2010-03-10

[16:05:28 CST(-0600)]

[16:50:46 CST(-0600)] <bspeakmon> skrebs: I can't find the source from the link in your page
<http://confluence.atlassian.com/display/DEVNET/Plugin+Tutorial+-+Defining+a+Pluggable+Service+in+a+Confluence+Plugin>

[16:52:50 CST(-0600)] <skrebs> bspeakmon: updated the link 

[16:53:27 CST(-0600)] <bspeakmon> now that's service

[16:53:28 CST(-0600)] <bspeakmon> thx

[17:09:12 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev

[17:11:08 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev

[17:11:08 CST(-0600)] * ChanServ sets mode +o tmoore

[17:13:30 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev

[17:13:30 CST(-0600)] * ChanServ sets mode +o tmoore

[17:13:47 CST(-0600)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev

[17:57:50 CST(-0600)] * bigmountainben (~Adium@c-24-5-43-22.hsd1.ca.comcast.net) has joined #atlassiandev

[17:59:01 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

[18:37:41 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev

[18:39:12 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev

[20:55:20 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

[21:15:36 CST(-0600)] * bug_ (~bug@75-151-78-9-siba.hfc.comcastbusiness.net) has joined #atlassiandev

[21:16:11 CST(-0600)] * bug_ (~bug@75-151-78-9-siba.hfc.comcastbusiness.net) has joined #atlassiandev

[21:45:13 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

[22:51:17 CST(-0600)] * jdoklovic (~jdoklovic@c-75-72-233-198.hsd1.mn.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-03-11

[23:36:51 CST(-0600)] <jdoklovic> why is there always somewhere between 23-24 users on this chat? how is that possible???

[23:37:38 CST(-0600)] <Broady> why not? 

[23:40:05 CST(-0600)] <jdoklovic> roll call.... mickey?? goofey???

[01:24:31 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev

[02:12:22 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev

[02:49:04 CST(-0600)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

[03:31:34 CST(-0600)] * bigmountainben (~Adium@c-24-5-43-22.hsd1.ca.comcast.net) has joined #atlassiandev

[03:42:56 CST(-0600)] * myrall_ (~matt@81.63.233.220.static.exetel.com.au) has joined #atlassiandev

[03:42:57 CST(-0600)] * ChanServ sets mode +o myrall_

[04:40:36 CST(-0600)] * myrall_ (~matt@81.63.233.220.static.exetel.com.au) has joined #atlassiandev

[04:40:36 CST(-0600)] * ChanServ sets mode +o myrall_

[05:56:37 CST(-0600)] * bug_ (~bug@75-151-78-9-siba.hfc.comcastbusiness.net) has joined #atlassiandev

[06:11:26 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

[08:00:30 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev

[08:25:02 CST(-0600)] * twong (~twong@c-216-75-233-106.static.wiline.com) has joined #atlassiandev

[08:43:16 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev

[09:41:22 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

[12:39:11 CST(-0600)] * bug_ (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev

[12:52:03 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

[13:35:20 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev

[13:41:11 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

[13:44:48 CST(-0600)] * bug_ (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev

[14:18:40 CST(-0600)] * myrall_ (~matt@124-168-150-204.dyn.iinet.net.au) has joined #atlassiandev

[14:18:40 CST(-0600)] * ChanServ sets mode +o myrall_

[14:25:40 CST(-0600)] * bug_ (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev

[14:58:54 CST(-0600)] * sebr (~sruiz@60-241-117-149.static.tpgi.com.au) has joined #atlassiandev

[14:58:57 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev

[16:11:17 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev

[16:22:17 CST(-0600)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev

[16:42:26 CST(-0600)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev

[16:42:26 CST(-0600)] * ChanServ sets mode +o tmoore

[16:52:51 CST(-0600)] <phuff> Anybody around who knows about fisheye/crucible?

[17:00:27 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

[17:05:51 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev

[17:07:31 CST(-0600)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev

[18:07:09 CST(-0600)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev

[18:33:23 CST(-0600)] * bug_ (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev

[20:01:48 CST(-0600)] * bug_ (~bug@65-112-21-194.dia.static.qwest.net) has joined #atlassiandev

[20:14:20 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

[21:21:14 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavy.com) has joined #atlassiandev

[21:52:45 CST(-0600)] * jedi (mike@li55-224.members.linode.com) has joined #atlassiandev

[21:56:39 CST(-0600)] * Ycros (~ycros@gnaw.yi.org) has joined #atlassiandev
[22:00:14 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:01:29 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:01:48 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:11:19 CST(-0600)] <phuff> Anybody that does fisheye dev around?
[22:11:44 CST(-0600)] <phuff> I'm trying to write a plugin and I'm a bit stuck.
[22:27:54 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-12

[23:43:09 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:22:02 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[01:27:33 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[01:59:21 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[02:31:55 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[02:59:46 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[05:18:56 CST(-0600)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[06:00:51 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[06:14:20 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[06:59:30 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[08:16:41 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[08:38:47 CST(-0600)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[08:56:34 CST(-0600)] * bug (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev
[10:54:00 CST(-0600)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:23:16 CST(-0600)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[13:19:13 CST(-0600)] * i386|laptop (~i386@ppp201-143.static.internode.on.net) has joined #atlassiandev
[15:38:01 CST(-0600)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[15:48:31 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:39:53 CST(-0600)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[20:05:14 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:13:04 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-03-13

[03:35:36 CST(-0600)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[04:33:56 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[04:55:56 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[06:58:53 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:15:10 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[08:19:14 CST(-0600)] * MartinCleaver (~martincle@76-10-151-65.dsl.teksavvy.com) has joined #atlassiandev
[09:20:18 CST(-0600)] * MartinCleaver (~martincle@69-165-148-247.dsl.teksavvy.com) has joined #atlassiandev
[09:26:43 CST(-0600)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[13:57:36 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[14:03:16 CST(-0600)] * MartinCleaver (~martincle@69-165-148-247.dsl.teksavvy.com) has joined #atlassiandev
[15:44:51 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:12:53 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[18:06:33 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:13:32 CST(-0600)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:33:48 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:11:39 CST(-0600)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[22:46:12 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-14

[00:42:26 CST(-0600)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[03:24:15 CDT(-0500)] * Ycros (~ycros@gnaw.yi.org) has joined #atlassiandev
[09:22:58 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[09:37:32 CDT(-0500)] * MartinCleaver (~martincle@69-165-148-247.dsl.teksavvy.com) has joined #atlassiandev
[10:29:16 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[10:58:59 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[11:35:17 CDT(-0500)] * miasma1 (~Brad@c-76-24-223-64.hsd1.ma.comcast.net) has joined #atlassiandev
[11:38:02 CDT(-0500)] * miasma1 (~Brad@c-76-24-223-64.hsd1.ma.comcast.net) has left #atlassiandev
[11:45:00 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[12:11:34 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[13:13:39 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[14:21:51 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:49:42 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[16:55:17 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:16:14 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[17:27:25 CDT(-0500)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[17:53:32 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:53:32 CDT(-0500)] * ChanServ sets mode +o tmoore
[19:08:21 CDT(-0500)] * jensschumacher (~anonymous@59.167.164.33) has joined #atlassiandev
[19:22:38 CDT(-0500)] * jensschumacher (~anonymous@59.167.164.33) has left #atlassiandev
[19:46:02 CDT(-0500)] * dchui (~dchui@159.192.48.60.klj02-home.tm.net.my) has joined #atlassiandev

atlassiandev_log-2010-03-15

[01:15:23 CDT(-0500)] * imaslovs (~leonidms@80.233.159.254) has joined #atlassiandev
[01:42:17 CDT(-0500)] * jensschumacher (~anonymous@59.167.164.33) has joined #atlassiandev
[01:42:24 CDT(-0500)] * jensschumacher (~anonymous@59.167.164.33) has left #atlassiandev
[02:03:17 CDT(-0500)] * trochej (trochej@noches.pl) has joined #atlassiandev
[02:29:10 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[02:29:10 CDT(-0500)] * ChanServ sets mode +o tmoore
[02:31:32 CDT(-0500)] * jensschumacher (~anonymous@59.167.164.33) has joined #atlassiandev
[03:30:08 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[03:32:58 CDT(-0500)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[05:19:00 CDT(-0500)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[06:02:57 CDT(-0500)] * jensschumacher (~anonymous@124-168-169-9.dyn.iinet.net.au) has joined #atlassiandev
[06:03:19 CDT(-0500)] * jensschumacher (~anonymous@124-168-169-9.dyn.iinet.net.au) has left #atlassiandev
[08:58:43 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[09:18:17 CDT(-0500)] * bug (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:18:44 CDT(-0500)] * MartinCleaver (~martincle@69-165-148-247.dsl.teksavvy.com) has joined #atlassiandev
[09:37:15 CDT(-0500)] * pharkmillups (~mphilips@206.83.89.3.ptr.us.xo.net) has joined #atlassiandev
[09:59:46 CDT(-0500)] <pharkmillups> morning, all
[10:00:48 CDT(-0500)] <pharkmillups> anyone know the trick to configuring Confluence to allow new users to sign in and post comments but not actually chance content on pages?
[11:59:58 CDT(-0500)] * dranged (~pwhite@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:43:08 CDT(-0500)] * bspeakmon_ (~bspeakmon@c-98-210-37-165.hsd1.ca.comcast.net) has joined #atlassiandev
[12:45:19 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[12:47:31 CDT(-0500)] * ChanServ sets mode +o bspeakmon_
[13:04:12 CDT(-0500)] * pharkmillups_ (~mphilips@206.83.89.3.ptr.us.xo.net) has joined #atlassiandev
[14:15:30 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has left #atlassiandev
[14:39:56 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[15:16:11 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[15:20:02 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[16:09:27 CDT(-0500)] * maleko_ (~maleko@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:49:57 CDT(-0500)] <rodogu> hey, I just ran my Unit + Integrations + Selenium tests on confluence 3.2-beta2 and everything worked out of the box! great job guys! you are helping me reaching plugin nirvana

[17:01:26 CDT(-0500)] <bspeakmon> 
[17:38:47 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:38:47 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:03:35 CDT(-0500)] * bug (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev
[18:16:14 CDT(-0500)] * maleko_ (~maleko@216-75-233-106.static.wiline.com) has left #atlassiandev
[18:31:22 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[20:43:51 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[21:26:29 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[21:26:29 CDT(-0500)] * ChanServ sets mode +o tmoore
[22:02:43 CDT(-0500)] * MartinCleaver (~martincle@69-165-148-247.dsl.teksavvy.com) has joined #atlassiandev

atlassiandev_log-2010-03-16

[00:19:43 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[00:38:47 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:38:20 CDT(-0500)] * jedi (mike@li55-224.members.linode.com) has joined #atlassiandev
[01:38:41 CDT(-0500)] * Ycros (~ycros@gnaw.yi.org) has joined #atlassiandev
[04:51:13 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[06:43:45 CDT(-0500)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:38:33 CDT(-0500)] * jm0rt1 (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[08:33:38 CDT(-0500)] * bigmountainben (~Adium@c-24-5-43-22.hsd1.ca.comcast.net) has joined #atlassiandev
[08:37:32 CDT(-0500)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[08:47:42 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[08:50:51 CDT(-0500)] * MartinCleaver (~martincle@69-165-148-247.dsl.teksavvy.com) has joined #atlassiandev
[09:42:14 CDT(-0500)] * puskar (~puskar@PUSKAR.ES.ITS.NYU.EDU) has joined #atlassiandev
[11:39:20 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:28:16 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:45:59 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[12:48:06 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajcZ0) has joined #atlassiandev
[13:18:06 CDT(-0500)] * bug (~bug@adsl-71-135-168-56.dsl.pltn13.pacbell.net) has joined #atlassiandev
[14:53:34 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:02:36 CDT(-0500)] <rodogu> Hey, any idea what this is all about?
[15:02:37 CDT(-0500)] <rodogu> 2010-03-11 17:15:22,171 ERROR http-8443-7 confluence.status.service.DefaultSystemInformationService getModifications No registry provided.

[15:02:48 CDT(-0500)] <rodogu> re: Confluence 
[16:20:41 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[17:03:07 CDT(-0500)] <mryall> does that happen when you go to the sys info page?
[17:03:24 CDT(-0500)] <mryall> I think it is supposed to check for modifications to your webapp, compared to the version we shipped
[17:03:51 CDT(-0500)] <mryall> and "no registry" means there's no information about the state of the app when it shipped, so it can't check for changes
[17:04:24 CDT(-0500)] <mryall> probably shouldn't be an "ERROR" level warning though, it is really just INFO
[17:16:53 CDT(-0500)] <mryall> no, it should only happen on the system info or 500 page, I believe
[17:17:03 CDT(-0500)] <mryall> definitely worth raising a support case though
[17:17:09 CDT(-0500)] <mryall> they'll get to the bottom of it
[17:43:21 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[18:07:09 CDT(-0500)] * i386 (~jdumay@59.167.164.33) has joined #atlassiandev
[18:13:41 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev

[18:17:07 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:17:07 CDT(-0500)] * ChanServ sets mode +o tmoore
[19:28:03 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[20:08:37 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[21:55:57 CDT(-0500)] * nenolod (quasselcor@atheme/member/pdpc.active.nenolod) has joined #atlassiandev
[21:56:13 CDT(-0500)] <nenolod> oh hey, that is better
[21:56:16 CDT(-0500)] <nenolod> so
[21:56:21 CDT(-0500)] <nenolod> http://confluence.atheme.org/ 😊
[21:56:23 CDT(-0500)] <nenolod> wtf 😊
[22:07:23 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[22:10:33 CDT(-0500)] <rodogu> @mryall, ok, thanx!

atlassiandev_log-2010-03-17

[00:14:38 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[00:38:55 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[01:15:31 CDT(-0500)] * bug (~bug@64.1.210.2.ptr.us.xo.net) has joined #atlassiandev
[01:40:25 CDT(-0500)] * bug (~bug@64.1.210.2.ptr.us.xo.net) has joined #atlassiandev
[03:23:38 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[03:24:25 CDT(-0500)] * jm0rt1 (~jm0rt1@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[04:01:13 CDT(-0500)] * jm0rt1 (~jm0rt1@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[04:18:38 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:37:39 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[07:03:57 CDT(-0500)] * MartinCleaver (~martincle@69-165-148-247.dsl.teksavy.com) has joined #atlassiandev
[08:30:00 CDT(-0500)] * bug (~bug@64.1.210.2.ptr.us.xo.net) has joined #atlassiandev
[08:36:22 CDT(-0500)] * bug (~bug@64.1.210.2.ptr.us.xo.net) has joined #atlassiandev
[08:42:20 CDT(-0500)] * jm0rt1 (~jm0rt1@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[10:22:14 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[11:01:55 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:07:14 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[11:13:22 CDT(-0500)] <phuff> Is there a betterway to reload a plugin on a live fecru besides deleting the osgi cached jar and restarting?
[12:04:12 CDT(-0500)] <bspeakmon> can you just disable/reenable?
[12:07:22 CDT(-0500)] <phuff> I tried that and it didn't work, bspeakmon
[12:07:35 CDT(-0500)] <bspeakmon> brb
[12:10:41 CDT(-0500)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:10:42 CDT(-0500)] * ChanServ sets mode +o bspeakmon
[12:11:18 CDT(-0500)] <bspeakmon> is it, like, a new version with new code, or do you just want it to reinitialize?
[12:11:31 CDT(-0500)] <phuff> A new version with new code
[12:11:36 CDT(-0500)] <phuff> I'm testing against the dev environment
[12:11:40 CDT(-0500)] <bspeakmon> ah
[12:11:42 CDT(-0500)] <bspeakmon> using the sdk?
[12:11:44 CDT(-0500)] <phuff> But to do anything with commit events I need to actually commit
[12:11:52 CDT(-0500)] <phuff> And I can't figure out how to do that on my dev environment
[12:12:01 CDT(-0500)] <bspeakmon> yeah
[12:12:05 CDT(-0500)] <bspeakmon> never tried that one myself.
[12:12:09 CDT(-0500)] <bspeakmon> but I see the problem.
[12:12:11 CDT(-0500)] <phuff> So I'm installing against a live version
[12:12:16 CDT(-0500)] <phuff> well, live-ish version
[12:12:29 CDT(-0500)] <bspeakmon> fisheye guys aren't awake yet
[12:12:36 CDT(-0500)] <phuff> What timezone are they in?
[12:12:45 CDT(-0500)] <phuff> I've tried a few different times during the day to try and find them
[12:12:55 CDT(-0500)] <bspeakmon> AEST
[12:13:23 CDT(-0500)] <bspeakmon> so it's 4:15 in the morning there
[12:13:56 CDT(-0500)] <bspeakmon> failing that, maybe one of the public fisheye plugins does something similar? in the functional tests?
[12:20:20 CDT(-0500)] * bug (~bug@208.66.27.210) has joined #atlassiandev
[14:12:38 CDT(-0500)] * _hen (~hen@12.71.56.2) has joined #atlassiandev
[14:12:44 CDT(-0500)] * _hen (~hen@12.71.56.2) has left #atlassiandev
[15:10:54 CDT(-0500)] * jdoklovic (~jdoklovic@66.187.202.189) has joined #atlassiandev
[15:45:10 CDT(-0500)] * CooPs89 (~chatzilla@h-60-19.A163.priv.bahnhof.se) has joined #atlassiandev
[15:54:01 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[15:54:49 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:39:26 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:28:06 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:28:44 CDT(-0500)] * MartinCleaver (~martincle@216.191.155.62) has joined #atlassiandev
[17:30:29 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[17:50:21 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:50:21 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:06:25 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[19:24:23 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:27:53 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:48:59 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[21:50:34 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:33:34 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev

atlassiandev_log-2010-03-18

[01:36:07 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[02:09:39 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev

[02:14:13 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[02:14:13 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[02:14:13 CDT(-0500)] * bigmountainben (~Adium@c-24-5-43-22.hsd1.ca.comcast.net) has joined #atlassiandev
[02:14:13 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[02:14:13 CDT(-0500)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[02:14:13 CDT(-0500)] * trochej (trochej@noches.pl) has joined #atlassiandev
[02:14:13 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[02:42:20 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[04:00:03 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[04:27:31 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[06:45:54 CDT(-0500)] * dcorbin_work (~dcorbin@unaffiliated/dcorbin) has joined #atlassiandev
[06:47:06 CDT(-0500)] <dcorbin_work> Is there support in jira for defining "dynamic alias" for an integration? For example, I'd like to be able to have an iteration called "current" which is calculated based on the date.
[07:41:48 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:48:27 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[08:27:44 CDT(-0500)] * dcorbin_work (~dcorbin@unaffiliated/dcorbin) has joined #atlassiandev
[08:36:50 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[09:12:25 CDT(-0500)] <rodogu> is https://studio.plugins.atlassian.com down?
[09:14:36 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[09:17:26 CDT(-0500)] <rodogu> not down, but very slow
[09:34:18 CDT(-0500)] * alpaca (~swe@unaffiliated/alpaca) has joined #atlassiandev
[09:37:52 CDT(-0500)] * MartinCleaver dutifully repoints everyone that arrives into #confluence into #atlassiandev
[09:39:19 CDT(-0500)] <alpaca> \o/
[09:45:34 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[10:32:10 CDT(-0500)] * jdoklovic (~jdoklovic@66.187.202.189) has joined #atlassiandev
[10:32:39 CDT(-0500)] <jdoklovic> @ben got an atlassian-plugins/osgi question for you
[12:17:55 CDT(-0500)] <bspeakmon> heh, I'll try
[12:19:48 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[14:46:59 CDT(-0500)] <phuff> So, I'm hacking up a review creation plugin
[14:47:10 CDT(-0500)] <phuff> but adding changesets to a review is sort of semi-silently failing
[14:47:23 CDT(-0500)] <phuff> It just does this: WARN - Rolled back transaction
[15:38:50 CDT(-0500)] <phuff> Ah hah, it silently dies when you try and manipulate an instantiated review and you're not the creator/moderator/author
[15:38:53 CDT(-0500)] <phuff> (Not sure which).
[15:38:58 CDT(-0500)] <phuff> (they're all the same in my scenario)
[15:43:16 CDT(-0500)] * vaix (~jmarquart@EXT-22.rdc.com) has joined #atlassiandev
[15:43:39 CDT(-0500)] <vaix> Is there a way to "include" content from a confluence page in a JIRA description/comment?
[15:44:05 CDT(-0500)] <vaix> I can see a lot of ways to include JIRA content in confluence - but not the other way. (w/ the exception of ont he dashboard via a gadget/portlet)
[15:44:40 CDT(-0500)] <vaix> Even if it was possible to embed a gadget in an issue - I could use the confluence page gadget to link it in....
[16:10:00 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[16:17:51 CDT(-0500)] <bspeakmon> phuff: you may care about the ImpersonationService
[16:19:29 CDT(-0500)] <phuff> Yeah, that's what I had to use to get it to not silently die, bspeakmon
[16:21:30 CDT(-0500)] <bspeakmon> good, wanted to make sure you weren't still stuck on it
[16:22:35 CDT(-0500)] <phuff> Now I just have to figure out how to do some basic java stuff (randomization, etc.) and I'll be good to go.
[16:22:40 CDT(-0500)] <phuff> thanks, for your help bspeakmon
[16:24:10 CDT(-0500)] <bspeakmon> no worries
[17:07:08 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[17:12:33 CDT(-0500)] * IgorMinar (~Adium@nat/sun/x-uamnwheewikbqaau) has joined #atlassiandev
[17:12:35 CDT(-0500)] * dhardiker (~dhardiker@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[17:12:59 CDT(-0500)] <dhardiker> allo allo
[17:13:04 CDT(-0500)] <dhardiker> anyone around?
[17:13:11 CDT(-0500)] <IgorMinar> ehlo 😊
[17:13:57 CDT(-0500)] <IgorMinar> does anyone know how to access felix web console with confluence?
[17:14:10 CDT(-0500)] <IgorMinar> or even telnet console would do 😊
[17:14:16 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[17:15:27 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:15:27 CDT(-0500)] * ChanServ sets mode +o tmoore
[17:15:34 CDT(-0500)] <bspeakmon> plugins/servlet/system/console should get you there
[17:16:11 CDT(-0500)] <IgorMinar> yeah, I found that mentioned at <http://confluence.atlassian.com/display/PLUGINFRAMEWORK/Troubleshooting+a+BundleException>
[17:16:11 CDT(-0500)] <dhardiker> neat trick ... is that blogged anywhere?
[17:16:36 CDT(-0500)] <bspeakmon> no – it doesn't work in all the products yet.
[17:16:51 CDT(-0500)] <dhardiker> lol ok 😊 that explains the page not found
[17:16:59 CDT(-0500)] <IgorMinar> but I was wondering if it works without SDK too? When I access that path on my build, I get 404 😞
[17:17:02 CDT(-0500)] <dhardiker> so I can find it at <http://localhost:8080/plugins/servlet/system/console> it just doesn't work 😞
[17:17:09 CDT(-0500)] <bspeakmon> no, the SDK slips it in during atlas-run/atlas-debug
[17:17:22 CDT(-0500)] <dhardiker> ah I see
[17:22:55 CDT(-0500)] <IgorMinar> do you know how its done in the sdk? I wonder if I could do the same thing in my confluence build
[17:46:20 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[17:47:34 CDT(-0500)] <bspeakmon> it's just a plugin
[17:47:39 CDT(-0500)] <bspeakmon> er, technically an osgi bundle
[17:47:58 CDT(-0500)] <bspeakmon> if you look at the startup logs from the SDK you'll see it being installed
[17:59:23 CDT(-0500)] * jedi (mike@l55-224.members.linode.com) has joined #atlassiandev
[18:04:45 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:06:10 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[18:06:33 CDT(-0500)] * jazzyl_ (~jazzyl@four.entic.net) has joined #atlassiandev

[18:13:35 CDT(-0500)] <IgorMinar> yay! I installed org.osgi.compendium-1.2.0.jar, httpservice-bridge-0.5.3.jar and org.apache.felix.webconsole-1.2.8.jar bundles as if they were plugins and now the console is accessible at /plugins/servlet/system/console/bundles
[18:24:36 CDT(-0500)] * IgorMinar1 (~Adium@nat/sun/x-iwpezzymkqirbt) has joined #atlassiandev
[18:25:15 CDT(-0500)] * Ycros (~ycros@gnaw.yi.org) has joined #atlassiandev
[18:41:26 CDT(-0500)] * nenolod (quasselcor@nenolod.net) has joined #atlassiandev
[18:41:26 CDT(-0500)] * nenolod (quasselcor@atheme/member/pdpc.active.nenolod) has joined #atlassiandev
[18:56:25 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:47:30 CDT(-0500)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has joined #atlassiandev
[21:41:19 CDT(-0500)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has joined #atlassiandev
[21:47:42 CDT(-0500)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has left #atlassiandev

atlassiandev_log-2010-03-19

[01:19:02 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[03:32:04 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[03:59:01 CDT(-0500)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[04:00:09 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btcentralplus.com) has joined #atlassiandev
[04:47:02 CDT(-0500)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[04:59:34 CDT(-0500)] * dhardiker (~dhardiker@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[05:08:36 CDT(-0500)] * frerich (~frerich@85.183.18.204) has joined #atlassiandev
[05:10:09 CDT(-0500)] <frerich> Hi. I've been looking for a particular JIRA Gadget for some time now but all my web search didn't yield any interesting support, so maybe somebody here can shed some light? I'm trying to find out whether there's an "Activity Stream" gadget which doesn't work on a project but on a filter.
[05:10:56 CDT(-0500)] <frerich> I'm considering to write this myself if it doesn't exist already (it seems like a typical "scratch your own itch" case 😊) but I was wondering whether there's maybe a technical reason which makes it very hard to show the activity in issues which match a given filter.
[05:12:13 CDT(-0500)] <frerich> In my particular case, I'd like an "Activity Stream" view on all issues which are assigned to me, so that I can see when other developers tinker with my issues. 😊
[06:29:58 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[06:34:14 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[06:36:03 CDT(-0500)] * dcorbin_work (~dcorbin@unaffiliated/dcorbin) has joined #atlassiandev
[06:43:01 CDT(-0500)] * frerich just found <http://jira.atlassian.com/browse/JRA-19155> - so I'm not the first to think of this. Yay!
[08:05:05 CDT(-0500)] * cemerick (~la_mer@cpe-98-14-141-244.nyc.res.rr.com) has joined #atlassiandev
[08:32:19 CDT(-0500)] * MartinCleaver (~martinle@69-196-163-99.dsl.teksavy.com) has joined #atlassiandev
[08:33:48 CDT(-0500)] * dhardiker_ (~dhardiker@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[08:47:08 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[10:49:33 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[10:58:52 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:24:16 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[11:58:26 CDT(-0500)] * dhardiker (~dhardiker@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[12:39:17 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[13:05:48 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[14:34:08 CDT(-0500)] * jdoklovic (~jdoklovic@66.187.202.189) has joined #atlassiandev
[14:37:38 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[14:44:08 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[14:45:39 CDT(-0500)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has joined #atlassiandev
[15:05:21 CDT(-0500)] * MartinCleaver (~martinle@66-207-222-14.beanfield.net) has joined #atlassiandev
[15:15:05 CDT(-0500)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[15:15:05 CDT(-0500)] * mryall (~mryall@59.167.164.33) has joined #atlassiandev
[15:15:05 CDT(-0500)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[15:15:05 CDT(-0500)] * MartinCleaver (~martinle@66-207-222-14.beanfield.net) has joined #atlassiandev
[15:15:05 CDT(-0500)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has joined #atlassiandev
[15:15:05 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:15:05 CDT(-0500)] * jdoklovic (~jdoklovic@66.187.202.189) has joined #atlassiandev
[15:15:05 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[15:15:05 CDT(-0500)] * dhardiker (~dhardiker@cpc1-warr6-2-0-cust65.1-1.cable.virginmedia.com) has joined #atlassiandev
[15:15:05 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[15:15:05 CDT(-0500)] * dcorbin_work (~dcorbin@unaffiliated/dcorbin) has joined #atlassiandev
[15:15:05 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[15:15:05 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[15:15:05 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[15:15:05 CDT(-0500)] * nenolod (quasselcor@atheme/member/pdpc.active.nenolod) has joined #atlassiandev
[15:15:05 CDT(-0500)] * Ycros (~ycros@gnaw.yi.org) has joined #atlassiandev
[15:15:05 CDT(-0500)] * jazzy_ (~jazzy@four.entic.net) has joined #atlassiandev
[15:15:05 CDT(-0500)] * jedi (mike@li55-224.members.linode.com) has joined #atlassiandev
[15:15:05 CDT(-0500)] * vaix (~jmarquart@EXT-22.rdc.com) has joined #atlassiandev
[15:15:05 CDT(-0500)] * alpaca (~swe@unaffiliated/alpaca) has joined #atlassiandev
[15:15:05 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[15:15:05 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[15:15:05 CDT(-0500)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[15:15:05 CDT(-0500)] * bigmountainben (~Adium@c-24-5-43-22.hsd1.ca.comcast.net) has joined #atlassiandev
[15:15:05 CDT(-0500)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[15:15:05 CDT(-0500)] * trochej (trochej@noches.pl) has joined #atlassiandev
[15:15:30 CDT(-0500)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[15:17:58 CDT(-0500)] * dudehook (~dhooker@static-71-170-11-48.dllstx.dsl-w.verizon.net) has joined #atlassiandev
[15:18:36 CDT(-0500)] <dudehook> I want to set up my JIRA to send notifications to our private IRC server when issues change
[15:18:49 CDT(-0500)] <dudehook> Anyone know if this is possible?

[15:22:22 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[15:29:15 CDT(-0500)] <dudehook> nm, found it
[15:35:22 CDT(-0500)] * jmort (~jmort@host81-156-15-18.range81-156.btccentralplus.com) has joined #atlassiandev
[15:35:26 CDT(-0500)] * IgorMinar (~Adium@c-98-234-176-3.hsd1.ca.comcast.net) has left #atlassiandev
[15:57:29 CDT(-0500)] * MartinCleaver_ (~martincle@66-207-222-14.beanfield.net) has joined #atlassiandev
[16:08:41 CDT(-0500)] * dudehook (~dhooker@static-71-170-11-48.dllstx.dsl-w.verizon.net) has left #atlassiandev
[16:19:29 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[17:00:08 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[17:29:38 CDT(-0500)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:57:19 CDT(-0500)] * sherloc (~sherloc@122.169.132.150) has joined #atlassiandev
[17:57:50 CDT(-0500)] <sherloc> dont think so
[18:00:28 CDT(-0500)] * sherloc (~sherloc@122.169.132.150) has left #atlassiandev
[18:18:13 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:09:26 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[19:14:16 CDT(-0500)] * tome (~tom@c-67-180-211-64.hsd1.ca.comcast.net) has joined #atlassiandev
[19:15:44 CDT(-0500)] <tome> Howdy all, I'm trying to get jira running on linux and it never seems to start up, Coyote keeps dying with
[19:15:46 CDT(-0500)] <tome> SEVERE: StandardServer.await: create[9005]
[19:15:51 CDT(-0500)] <tome> java.net.SocketException: Invalid argument
[19:16:12 CDT(-0500)] <tome> I've tried many ports for shutdown and access port, and nothing seems to get by this. Anyone see this before?
[19:23:24 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[19:24:09 CDT(-0500)] <phuff> Any fecru peeps around?
[19:24:49 CDT(-0500)] <phuff> Gotta head to dinner. But when I add a jiraissuekey to a review, the review creation bombs for some reason. Any thoughts?
[19:25:24 CDT(-0500)] * twong_ (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:28:41 CDT(-0500)] <bspeakmon> it's the weekend in sydney, so no answer is forthcoming 😊
[19:45:12 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:43:58 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:53:37 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[23:07:33 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev

atlassiandev_log-2010-03-20

[23:47:46 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[23:47:46 CDT(-0500)] * ChanServ sets mode +o tmoore
[00:33:28 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:42:22 CDT(-0500)] <alpaca> for the record, my freezer **does** have a light
[01:26:28 CDT(-0500)] <Broady> alpaca: good to know
[01:26:33 CDT(-0500)] <Broady> mine doesn't.
[02:20:54 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[02:20:54 CDT(-0500)] * ChanServ sets mode +o tmoore
[02:31:16 CDT(-0500)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[05:10:12 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[05:10:12 CDT(-0500)] * ChanServ sets mode +o tmoore
[07:23:52 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[07:23:58 CDT(-0500)] * ChanServ sets mode +o tmoore
[07:39:58 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[09:41:32 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[10:28:13 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[10:46:44 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[12:03:11 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[17:15:29 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[21:01:04 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[21:12:50 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[21:14:12 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[21:14:41 CDT(-0500)] <phuff> that was weird
[21:51:41 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev

atlassiandev_log-2010-03-21

[23:24:01 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[02:30:48 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[03:01:29 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[09:10:23 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[12:03:47 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[13:05:19 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[13:26:36 CDT(-0500)] * dhardiker (~dhardiker@80.187.148.54) has joined #atlassiandev
[15:13:43 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[16:11:43 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:17:27 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[17:41:03 CDT(-0500)] * atlasbot (~PircBot@63-246-22-217.contegix.com) has joined #atlassiandev
[17:41:03 CDT(-0500)] * Topic is !!! I THINK THE FREEZER DESERVES A LIGHT AS WELL! !! | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#039;> set by bspeakmon!~bspeakmon@216-75-233-106.static.wiline.com on 2010-03-10 16:05:28 CST(-0600)
[17:42:51 CDT(-0500)] <Broady> ZOMG per is leaving??
[18:08:10 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[18:14:01 CDT(-0500)] * skrebs_ (~shannon@121.91.30.142) has joined #atlassiandev
[18:22:42 CDT(-0500)] * Ycros (~ycros@gnav.yi.org) has joined #atlassiandev

[18:55:05 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlasiandev
[18:56:06 CDT(-0500)] * Sorcy (~ycros@gaw.yi.org) has joined #atlasiandev
[19:32:12 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlasiandev
[19:46:34 CDT(-0500)] <skrebs> Is there any way to install multiple plugins as part of the atlassian SDK commands for example when one plugin depends on another to be installed?
[19:51:41 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlasiandev
[19:51:41 CDT(-0500)] * ChanServ sets mode +o tmoore
[19:52:41 CDT(-0500)] <mrdon> sure
[19:52:54 CDT(-0500)] <mrdon> skrebs: you can run multiple atlas-cli executions, just specify a different port
[19:53:17 CDT(-0500)] <mrdon> or you can configure your pom to ensure the other plugin is always installed when atlas-run is executed
[19:53:48 CDT(-0500)] <skrebs> is there doco on configuring the pom anywhere?
[19:55:26 CDT(-0500)] <mrdon> I know ben has been working on it
[19:55:32 CDT(-0500)] <mrdon> not sure if it is published anywhere yet though
[19:55:59 CDT(-0500)] <mrdon> if using an ide like intelliJ, it should prompt you
[19:56:11 CDT(-0500)] <mrdon> and I think there is some maven command to list the configuration options for a plugin
[22:18:08 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlasiandev
[22:56:56 CDT(-0500)] * skrebs (~shannon@119.12.76.131) has joined #atlasiandev

atlasiandev_log-2010-03-22

[02:56:16 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlasiandev
[04:08:43 CDT(-0500)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlasiandev
[04:50:23 CDT(-0500)] * jmort (~jmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlasiandev
[04:53:14 CDT(-0500)] * skrebs_ (~shannon@119.12.76.131) has joined #atlasiandev
[04:56:37 CDT(-0500)] * jmort1 (~jmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlasiandev
[06:15:08 CDT(-0500)] <dcorbin_work> Is there support in jira for defining "dynamic alias" for an integration? For example, I'd like to be able to have an iteration called "current" which is calculated based on the date.
[06:20:02 CDT(-0500)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlasiandev
[06:28:02 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlasiandev
[08:06:56 CDT(-0500)] * jmort (~jmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlasiandev
[08:35:45 CDT(-0500)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlasiandev
[08:39:22 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlasiandev
[08:57:50 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlasiandev
[08:59:39 CDT(-0500)] * kalamon (~kalamon@adx1.neoplus.adsl.tpnet.pl) has joined #atlasiandev
[09:12:05 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlasiandev
[09:12:26 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlasiandev
[09:19:12 CDT(-0500)] * alpaca_ (~swe@cpe-65-25-30-10.neo.res.rr.com) has joined #atlasiandev
[09:22:46 CDT(-0500)] * jmort1 (~jmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlasiandev
[09:25:36 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlasiandev
[09:34:55 CDT(-0500)] * dcorbin_work (~dcorbin@unaffiliated/dcorbin) has joined #atlasiandev
[10:01:04 CDT(-0500)] * kalamon_ (~kalamon@acv49.neoplus.adsl.tpnet.pl) has joined #atlasiandev
[10:39:38 CDT(-0500)] * jmort (~jmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlasiandev
[12:09:59 CDT(-0500)] * alpaca (~swe@unaffiliated/alpaca) has joined #atlasiandev
[12:22:46 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlasiandev
[12:52:14 CDT(-0500)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlasiandev
[12:52:14 CDT(-0500)] * ChanServ sets mode +o bspeakmon
[13:42:12 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlasiandev
[13:43:33 CDT(-0500)] <phuff> bspeakmon: know anything about Review state in crucible?

[13:43:35 CDT(-0500)] <phuff>
[13:43:45 CDT(-0500)] <phuff> I'll try and catch the crucible guys on in another 5-6 hours, if not.
[13:46:07 CDT(-0500)] <bspeakmon> I know **of** it
[13:46:16 CDT(-0500)] <bspeakmon> what specifically?
[13:46:44 CDT(-0500)] <phuff> Well, I've set a review to closed in the GUI
[13:46:46 CDT(-0500)] <phuff> and it shows up as closed
[13:46:54 CDT(-0500)] <phuff> But when I fetch it's ReviewData the state is "Review"
[13:47:04 CDT(-0500)] <bspeakmon> which version?
[13:47:04 CDT(-0500)] <phuff> It's closed but hasn't been all the way reviewed, so I wonder if that's part of the problem
[13:47:10 CDT(-0500)] <phuff> 2.2 I thin
[13:47:11 CDT(-0500)] <phuff> K
[13:47:14 CDT(-0500)] <bspeakmon> hm
[13:47:14 CDT(-0500)] <alpaca> im having issues with a java heap space error, ive added lines to my startup.sh in tomcat to set these JAVA_OPTS
[13:47:16 CDT(-0500)] <phuff> But I'm using the dev stuff
[13:47:23 CDT(-0500)] <phuff> with atlas-run
[13:48:04 CDT(-0500)] <bspeakmon> and you're making sure to get the reviewdata afresh from a new review object that you get **after** the review is closed?
[13:48:14 CDT(-0500)] <phuff> bspeakmon: Well, it's in a commit handler
[13:48:18 CDT(-0500)] <phuff> And it's happening after close
[13:48:20 CDT(-0500)] <bspeakmon> ah
[13:48:21 CDT(-0500)] <bspeakmon> okay
[13:48:45 CDT(-0500)] <phuff> I wonder if searchService is caching reviewData
[13:48:58 CDT(-0500)] <bspeakmon> alpaca: if you're having a problem running jira or another product on your server, you should open a support case at support.atlassian.com, they can better help
[13:49:03 CDT(-0500)] <bspeakmon> I think it does
[13:49:13 CDT(-0500)] <bspeakmon> I ran into something similar the last time I was in there
[13:49:30 CDT(-0500)] <phuff> or if ReviewData.getState() will return Review if there's still a reviewer outstanding, even if it's been closed.
[13:49:33 CDT(-0500)] <bspeakmon> the services don't return live object views
[13:49:43 CDT(-0500)] <phuff> I'm trying to reopen the review and it's failing
[13:49:44 CDT(-0500)] <bspeakmon> that second part is what I would ask the fecru guys

[13:50:07 CDT(-0500)] <phuff> Maybe I could work around it by loading the review not via the service?
[13:52:22 CDT(-0500)] <phuff> Ok
[13:52:26 CDT(-0500)] <phuff> Thanks, bspeakmon
[13:54:20 CDT(-0500)] <phuff> I also apparently can't like to a jirakey programatically.
[13:56:40 CDT(-0500)] <bspeakmon> do you have applinks configured?
[14:02:18 CDT(-0500)] <phuff> I have jira linking configured
[14:02:26 CDT(-0500)] <bspeakmon> hm

[14:02:30 CDT(-0500)] <bspeakmon> I'm out of ideas then 😊
[14:02:36 CDT(-0500)] <phuff> Yeah
[14:02:38 CDT(-0500)] <phuff> It's weird
[14:02:45 CDT(-0500)] <phuff> When I did jira linking the transaction stopped rolling back
[14:02:48 CDT(-0500)] <phuff> But it's still not linking.
[14:02:57 CDT(-0500)] <bspeakmon> so some exception is being thrown during the link
[14:03:07 CDT(-0500)] <phuff> yeah
[14:03:10 CDT(-0500)] <phuff> Probably
[14:03:20 CDT(-0500)] <phuff> I'll wait until later today and check with the guys in Australia
[14:03:32 CDT(-0500)] <phuff> Is there somebody specific I should look for, bspeakmon?
[14:03:58 CDT(-0500)] <bspeakmon> whoever shows up, really
[14:04:01 CDT(-0500)] <phuff> ok
[14:04:07 CDT(-0500)] <bspeakmon> t davies usually comes in
[14:04:08 CDT(-0500)] <bspeakmon> he's good
[14:04:21 CDT(-0500)] <phuff> Do they get in around 9-10am local time?
[14:04:33 CDT(-0500)] <phuff> or later on?
[14:04:40 CDT(-0500)] <bspeakmon> yes, but they have standup and the morning coffee run (it's an aussie thing), so closer to 11
[14:05:15 CDT(-0500)] <phuff> Ok

[14:05:18 CDT(-0500)] <phuff> That's fine 😊
[14:05:42 CDT(-0500)] <phuff> That's like 7:00-8:00 my time so I just have to remember to come on to freenode while I'll be working in the garden 😊

[14:05:58 CDT(-0500)] <phuff> Thanks for all your help with this, bspeakmon
[14:06:00 CDT(-0500)] <phuff> I really appreciate it.
[14:06:27 CDT(-0500)] <bspeakmon> no worries

[14:31:25 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[14:31:41 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[14:40:35 CDT(-0500)] * nenolod (quasselcor@nenolod.net) has joined #atlassiandev
[14:40:35 CDT(-0500)] * nenolod (quasselcor@atheme/member/pdpc.active.nenolod) has joined #atlassiandev
[15:00:34 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:12:41 CDT(-0500)] * jmort (~jmort@host81-158-19-64.range81-158.btcentralplus.com) has joined #atlassiandev
[15:47:30 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[15:49:21 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[16:12:21 CDT(-0500)] * jmort (~jmort@host81-158-19-64.range81-158.btcentralplus.com) has joined #atlassiandev
[18:04:54 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[18:46:01 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:46:01 CDT(-0500)] * ChanServ sets mode +o tmoore
[20:06:50 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[20:20:17 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:04:40 CDT(-0500)] * skrebs_ (~shannon@119.12.76.131) has joined #atlassiandev
[21:59:50 CDT(-0500)] * skrebs_ (~shannon@119.12.76.131) has joined #atlassiandev
[22:23:08 CDT(-0500)] * skrebs_ (~shannon@119.12.76.131) has joined #atlassiandev

atlassiandev_log-2010-03-23

[23:18:39 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[23:31:30 CDT(-0500)] <phuff> Any fecru devs around?
[23:31:59 CDT(-0500)] <phuff> Or did I miss them again?
[23:33:01 CDT(-0500)] <phuff> darn
[23:54:21 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:00:20 CDT(-0500)] <tmoore> we don't seem to ever have a large fecru contingent in here
[00:00:34 CDT(-0500)] <tmoore> they don't like IRC for some reason :-P

[00:00:37 CDT(-0500)] <phuff> I've noticed 😊
[00:00:57 CDT(-0500)] <phuff> I'm having trouble getting my fecru commit event handler to twiddle crucible review states
[00:01:04 CDT(-0500)] <phuff> and add links
[00:01:27 CDT(-0500)] <phuff> Is there a better place than #atlassiandev for asking questions like that?
[00:01:59 CDT(-0500)] <tmoore> have you tried <http://forums.atlassian.com/forum.jspa?forumID=127> ?
[00:02:05 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[00:02:54 CDT(-0500)] <phuff> no

[00:03:05 CDT(-0500)] <phuff> But I got some help here before, so like a crack addict I kept coming back 😊
[00:06:05 CDT(-0500)] <tmoore> ha
[00:06:10 CDT(-0500)] <tmoore> sometimes sebr jumps on
[00:06:14 CDT(-0500)] <tmoore> i'll see if he's around
[00:06:30 CDT(-0500)] <phuff> Thanks
[00:08:51 CDT(-0500)] * sebr (~seb@59.167.164.33) has joined #atlassiandev
[00:08:57 CDT(-0500)] * sebr (~seb@amarok/developer/sebr) has joined #atlassiandev
[00:09:29 CDT(-0500)] <tmoore> hey sebr, thanks for jumping on
[00:09:33 CDT(-0500)] <sebr> nw
[00:09:35 CDT(-0500)] <sebr> phuff - i hear you have some fecru questions?

[00:09:39 CDT(-0500)] <phuff> Yeah
[00:10:02 CDT(-0500)] <phuff> I've got an overly hacked review-creator plugin I'm working on to do some custom stuff around assigning reviews to people on the right teams

[00:10:10 CDT(-0500)] <phuff> And I've got two problems 😊
[00:10:24 CDT(-0500)] <phuff> #1 I can't change the state of a review without it erroring out
[00:10:33 CDT(-0500)] <phuff> and #2 I can't associate a ticket wit
[00:10:44 CDT(-0500)] <phuff> *associate a jira ticket
[00:11:18 CDT(-0500)] <phuff> Everything else is working great, though.
[00:11:27 CDT(-0500)] <phuff> I've been super impressed with the APIs
[00:11:47 CDT(-0500)] <sebr> and you're certain that your workflow allows you to change the state of the review?
[00:12:02 CDT(-0500)] <sebr> ie, if user A creates, but is not the author/moderator, they may not be able to start the review
[00:12:08 CDT(-0500)] <phuff> I'm using th impersonator to do it as the admin
[00:12:30 CDT(-0500)] <phuff> And I also tried doing it as the review author
[00:12:37 CDT(-0500)] <sebr> and admin user doesn't have su workflow rights though
[00:12:42 CDT(-0500)] <sebr> hm, okay
[00:12:46 CDT(-0500)] <sebr> what does your code look like?
[00:13:11 CDT(-0500)] <phuff> reviewService.changeState(rd.getPermalink(), ReviewService.Action.Reopen.getActionString());
[00:13:30 CDT(-0500)] <phuff> But the thing is, even though the UI shows the review as closed
[00:13:47 CDT(-0500)] <phuff> the state of the review that I get back from the searchService isn't closed.
[00:13:55 CDT(-0500)] <phuff> It's like the searchService is caching the review data or something
[00:14:03 CDT(-0500)] <sebr> wacky
[00:14:11 CDT(-0500)] <phuff> I've outputted the state and it's not closed, even though it says closed in the web ui
[00:14:33 CDT(-0500)] <phuff> Like, I closed the ticket by hand and I'm trying to reopen it programmatically as the rd.getAuthor()
[00:15:32 CDT(-0500)] <phuff> I'm also not able to associate jira tickets
[00:15:51 CDT(-0500)] <phuff> When I joined my dev instance up with our jira it stopped rolling back the transaction when I tried to associate them
[00:15:55 CDT(-0500)] <phuff> But the associations don't take.
[00:16:02 CDT(-0500)] <phuff> Does the searchService do any caching?
[00:16:25 CDT(-0500)] <sebr> shouldn't do
[00:16:37 CDT(-0500)] <phuff> Or should I re-fetch ReviewData for an individual dev when I get it form the search service?
[00:16:53 CDT(-0500)] <sebr> honestly, not sure as i didn't engineer the api
[00:17:54 CDT(-0500)] <phuff> Ok
[00:17:57 CDT(-0500)] <phuff> here's another question: 😊
[00:18:06 CDT(-0500)] <phuff> I can probably just do this the right way 😊
[00:18:14 CDT(-0500)] <phuff> But I closed the review before the reviewers had reviewed it
[00:18:17 CDT(-0500)] <sebr> you're best off to file a support request for these things
[00:18:29 CDT(-0500)] <phuff> Would a review not show up as closed if the reviewers havent' reviewed it maybe?
[00:18:47 CDT(-0500)] <sebr> no, reviews are closed if they are closed – participant completion is irrellevant
[00:19:08 CDT(-0500)] <sebr> although a participant will not be marked as complete for the review iirc
[00:19:34 CDT(-0500)] <phuff> ok
[00:19:52 CDT(-0500)] <phuff> any ideas on the jira ticket linking?
[00:20:06 CDT(-0500)] <sebr> let me check
[00:20:28 CDT(-0500)] <phuff> I'm happy to file support requests...
[00:21:00 CDT(-0500)] <phuff> like I told tmoore I just got some help here and I've been like a crack addict.

[00:21:02 CDT(-0500)] <phuff> 😊
[00:21:24 CDT(-0500)] <sebr> phuff: secret engineer access
[00:21:40 CDT(-0500)] <sebr> when you create a review, if you have sent the jiralIssueKey value then it should be automagically linked
[00:21:55 CDT(-0500)] <phuff> Yeah, it's not doing that...
[00:22:06 CDT(-0500)] <sebr> the issueKey is part of the ReviewData object
[00:22:31 CDT(-0500)] <phuff> reviewData.setJiralIssueKey(jirald);
[00:22:44 CDT(-0500)] <phuff> That's how I'm setting it, as the author
[00:22:45 CDT(-0500)] <sebr> is the jira project mapped to the crucible project?
[00:22:53 CDT(-0500)] <sebr> ie, you can link a jira in the crucible ui?
[00:23:09 CDT(-0500)] <phuff> It suggests the linking based on the comments
[00:23:14 CDT(-0500)] <phuff> And when I click "Link to Jira" it works
[00:23:41 CDT(-0500)] <phuff> But when I do it programatically it doesn't work.
[00:24:38 CDT(-0500)] <phuff> Like I said, until I did it as the review author with Jira linked, it would error out.
[00:25:11 CDT(-0500)] <sebr> what error?
[00:25:27 CDT(-0500)] <phuff> It always errors out silently and says, "transaction rolled back" on the atlas-run stdout
[00:25:39 CDT(-0500)] <phuff> so I don't actually see the exceptions
[00:26:48 CDT(-0500)] <phuff> is there some other place that it logs errors/exceptions that get thrown?
[00:29:07 CDT(-0500)] <sebr> should be in var/logs
[00:29:16 CDT(-0500)] <sebr> \$FISHEYE_INST/var/logs
[00:29:58 CDT(-0500)] <phuff> Ok
[00:29:59 CDT(-0500)] <phuff> Thanks
[00:31:26 CDT(-0500)] <sebr> getting a bit busy here – i suggest that you file support requests
[00:31:31 CDT(-0500)] <phuff> Ok
[00:31:33 CDT(-0500)] <phuff> thanks, sebr
[00:31:35 CDT(-0500)] <phuff> I appreciate your help.
[00:31:57 CDT(-0500)] <sebr> np
[00:33:13 CDT(-0500)] <phuff> sebr: Is there a particular place to submit support requests?
[00:33:55 CDT(-0500)] <phuff> for stuff like this?
[00:34:03 CDT(-0500)] <sebr> <http://support.atlassian.com/>
[00:34:09 CDT(-0500)] <phuff> Ok
[00:34:09 CDT(-0500)] <phuff> Thanks
[00:34:13 CDT(-0500)] <sebr> np
[01:05:15 CDT(-0500)] * skrebs (~shannon@123.200.214.188) has joined #atlassiandev
[01:09:20 CDT(-0500)] * jmort (~jonmort@host81-158-19-64.range81-158.btcentralplus.com) has joined #atlassiandev

[01:48:42 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[02:10:32 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:31:05 CDT(-0500)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[04:25:53 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[04:40:40 CDT(-0500)] * jmort (~jonmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlassiandev
[06:18:28 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavy.com) has joined #atlassiandev
[06:53:08 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:18:49 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[08:47:19 CDT(-0500)] * jmort (~jonmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlassiandev
[08:53:34 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[10:01:33 CDT(-0500)] * whaleys (~whaleys@li35-137.members.linode.com) has joined #atlassiandev
[10:04:32 CDT(-0500)] * MartinCleaver (~martincle@66-207-222-14.beanfield.net) has joined #atlassiandev
[10:55:26 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[11:24:36 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:01:39 CDT(-0500)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:24:29 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[12:27:50 CDT(-0500)] * MartinCleaver (~martincle@66-207-222-14.beanfield.net) has joined #atlassiandev
[12:48:39 CDT(-0500)] * jmort (~jonmort@gmt-049-047.t-mobile.co.uk) has joined #atlassiandev
[12:56:39 CDT(-0500)] <alpaca> does anyone know of an easy way to flag my jira install as not setup yet, ie to re-run the setup? i need to create my database
[12:58:48 CDT(-0500)] <bspeakmon> quickest way would be to remove the jira-home directory, I think
[13:40:24 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[13:48:27 CDT(-0500)] * Carlfish (cmiller@bodhi.pastiche.org) has joined #atlassiandev
[13:48:38 CDT(-0500)] <Carlfish> I like cheese!
[13:49:20 CDT(-0500)] <bspeakmon> MacroException: No page context found
[13:49:45 CDT(-0500)] <bspeakmon> NOW YOU KNOW how it is
[13:49:48 CDT(-0500)] <Carlfish> Spoilsport.
[14:00:10 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[14:04:29 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[14:09:19 CDT(-0500)] <alpaca> bspeakmon: ive tried removing the lock file and removing all the folder contents. every time i start the app it takes a very long time to start, and it always results in giving me the database lock error
[14:13:06 CDT(-0500)] <Carlfish> Sorry to come into this late. Which application, and are you using standalone or your own appserver deployment?
[14:13:29 CDT(-0500)] <alpaca> im using a tomcat6 deployment
[14:13:43 CDT(-0500)] <alpaca> running both jira and confluence
[14:13:54 CDT(-0500)] <Carlfish> Did you put the war in the webapps directory and configure it in the config xml at the same time?
[14:14:02 CDT(-0500)] <Carlfish> That tends to start the app twice.
[14:16:33 CDT(-0500)] <Carlfish> http://fishbowl.pastiche.org/2005/04/01/random_tomcat_annoyance/
[14:19:38 CDT(-0500)] <alpaca> Carlfish: that doesnt seem to be the case :/
[14:55:42 CDT(-0500)] <alpaca> yeah. i fixed my slow loading problem, but JIRA still thinks its locked every time i start it, even after i remove the home directory's contents
[15:53:30 CDT(-0500)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[15:53:36 CDT(-0500)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[15:53:46 CDT(-0500)] * anatoli_ (~anatoli@59.167.164.33) has joined #atlassiandev
[15:56:14 CDT(-0500)] * myrall (~myrall@59.167.164.33) has joined #atlassiandev
[15:56:14 CDT(-0500)] * ChanServ sets mode +o myrall
[16:42:54 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[16:44:11 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:44:11 CDT(-0500)] * ChanServ sets mode +o tmoore
[16:54:36 CDT(-0500)] * bug (~bug@69.4.176.194) has joined #atlassiandev
[17:13:56 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:13:58 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[17:13:58 CDT(-0500)] * ChanServ sets mode +o tmoore
[17:29:52 CDT(-0500)] * MartinCleaver (~martincle@66-207-222-14.beanfield.net) has joined #atlassiandev
[18:23:42 CDT(-0500)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:24:47 CDT(-0500)] * bug (~bug@65-112-21-194.dia.static.qwest.net) has joined #atlassiandev
[18:41:18 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:20:54 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:30:15 CDT(-0500)] * jedi_ (mike@li55-224.members.linode.com) has joined #atlassiandev
[19:31:10 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[19:38:56 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[20:06:40 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[21:24:54 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[21:41:14 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavy.com) has joined #atlassiandev
[21:43:27 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:46:00 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:03:23 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:23:55 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:39:57 CDT(-0500)] * rodogu1 (~Adium@d154-5-174-17.bchsia.telus.net) has joined #atlassiandev
[22:45:57 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev

atlassiandev_log-2010-03-24

[00:57:27 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[04:03:30 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[04:36:25 CDT(-0500)] * jmort (~jonmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlassiandev
[05:40:58 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[05:41:35 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[06:07:05 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[06:31:00 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev

[06:31:00 CDT(-0500)] * ChanServ sets mode +o tmoore
[07:29:31 CDT(-0500)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:52:08 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[08:21:00 CDT(-0500)] * jmort (~jonmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlassiandev
[09:36:12 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[10:27:35 CDT(-0500)] * jmort (~jonmort@host86-173-10-133.range86-173.btcentralplus.com) has joined #atlassiandev
[10:30:30 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[10:34:24 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[11:58:42 CDT(-0500)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:10:43 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[12:15:03 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[12:16:55 CDT(-0500)] * jmort (~jonmort@genkt-057-182.t-mobile.co.uk) has joined #atlassiandev
[12:36:42 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[12:37:06 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[12:54:28 CDT(-0500)] * jmort1 (~jonmort@genkt-056-044.t-mobile.co.uk) has joined #atlassiandev
[13:55:55 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[14:57:32 CDT(-0500)] * pharkmillups (~mphilips@206.83.89.3.ptr.us.xo.net) has joined #atlassiandev
[14:58:07 CDT(-0500)] <pharkmillups> afternoon. any confluence lovers in the house?
[15:14:25 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:30:58 CDT(-0500)] * pharkmillups (~mphilips@206.83.89.3.ptr.us.xo.net) has left #atlassiandev
[15:38:04 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:20:14 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[16:23:36 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:23:36 CDT(-0500)] * ChanServ sets mode +o tmoore
[16:57:35 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[17:09:33 CDT(-0500)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:20:50 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[18:27:54 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[18:29:02 CDT(-0500)] * bspeakmon sets mode +o drosen
[18:29:47 CDT(-0500)] <drosen> oh hell yes
[18:34:26 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[19:12:08 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:12:38 CDT(-0500)] <Carlfish> \$5
[19:32:20 CDT(-0500)] * Sorcy (~ycros@gnaw.yi.org) has joined #atlassiandev
[19:32:20 CDT(-0500)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:32:20 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[19:32:20 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[19:32:20 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[19:32:20 CDT(-0500)] * kalamon_home (~mm@chello087207167045.chello.pl) has joined #atlassiandev
[19:32:20 CDT(-0500)] * mryall (~mryall@59.167.164.33) has joined #atlassiandev
[19:32:20 CDT(-0500)] * whalejy (~whalejy@i35-137.members.liinode.com) has joined #atlassiandev
[19:32:20 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[19:32:20 CDT(-0500)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:32:20 CDT(-0500)] * trochej (trochej@noches.pl) has joined #atlassiandev
[19:32:42 CDT(-0500)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[19:32:54 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:32:54 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:32:54 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[19:32:54 CDT(-0500)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[19:32:54 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[19:55:53 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[19:56:23 CDT(-0500)] * JonathanNolen_ (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[20:38:24 CDT(-0500)] * Topic is 'This rabbit will find your card or die. | This channel is being logged with transcripts available at <http://confluence.atlassian.com/display/DEVNET/IRC+Chat+Transcripts#>; set by bspeakmon on 2010-03-24 20:38:24 CDT(-0500)
[20:39:25 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[21:04:43 CDT(-0500)] * bspeakmon_ (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[21:06:33 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[21:06:33 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[21:06:33 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[21:06:33 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[21:06:33 CDT(-0500)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[21:06:33 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[21:24:38 CDT(-0500)] * AJC_Z0 (~AJCZ0@ip68-105-188-179.dc.dc.cox.net) has joined #atlassiandev
[21:38:15 CDT(-0500)] * Carlfish (cmiller@bodhi.pastiche.org) has joined #atlassiandev
[21:52:07 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[22:00:23 CDT(-0500)] * bspeakmon_ (~bspeakmon@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[22:00:24 CDT(-0500)] * drosen_ (~drosen@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[22:05:42 CDT(-0500)] * bug (~bug@adsl-76-199-101-156.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[22:09:28 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:28:21 CDT(-0500)] * bspeakmon_ (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[22:28:30 CDT(-0500)] * drosen_ (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev

atlassiandev_log-2010-03-25

[23:46:45 CDT(-0500)] * bug (~bug@64.1.210.2.ptr.us.xo.net) has joined #atlassiandev
[00:44:30 CDT(-0500)] * bug (~bug@64.1.210.2.ptr.us.xo.net) has joined #atlassiandev
[00:52:40 CDT(-0500)] * bspeakmon_ (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[03:19:27 CDT(-0500)] * skrebs_ (~shannon@121.91.100.70) has joined #atlassiandev
[04:29:07 CDT(-0500)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[05:10:32 CDT(-0500)] * skrebs (~shannon@121.91.100.70) has joined #atlassiandev

[05:17:43 CDT(-0500)] * cemerick (~la_mer@75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[05:34:16 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[06:47:08 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[08:24:07 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[08:47:09 CDT(-0500)] * bug (~bug@64.1.210.2.ptr.us.xo.net) has joined #atlassiandev
[09:33:44 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:35:39 CDT(-0500)] * jdoklovic (~jdoklovic@66.187.202.84) has joined #atlassiandev
[10:23:58 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[10:54:08 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[11:56:10 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[12:52:04 CDT(-0500)] * ChanServ sets mode +o bspeakmon
[13:10:56 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:10:56 CDT(-0500)] * ChanServ sets mode +o drosen
[13:17:28 CDT(-0500)] * drosen_ (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:21:42 CDT(-0500)] * drosen_ (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:24:41 CDT(-0500)] * ChanServ sets mode +o drosen
[13:34:09 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[14:56:54 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[15:08:24 CDT(-0500)] <jdoklovic> anyone know if there's a way to run the atlassian-plugin bundle transformation at build time instead of waiting for runtime?
[15:10:05 CDT(-0500)] <bspeakmon> you mean the obr phase?
[15:17:49 CDT(-0500)] <phuff> What is the obr for anyway?
[15:18:09 CDT(-0500)] <phuff> I've been installing my plugin by just copying the jar to /var/plugins/user/
[15:18:13 CDT(-0500)] <phuff> Do I need to do something with the obr too?
[15:22:30 CDT(-0500)] <bspeakmon> the obr is a specialized OSGi bundle. basically a jar with OSGi stuff added to the manifest.
[15:22:42 CDT(-0500)] <bspeakmon> the plugin system doesn't use obs yet, but it may one day.
[15:22:46 CDT(-0500)] <phuff> Ok
[15:22:47 CDT(-0500)] <bspeakmon> so you can ignore it for now.
[15:23:03 CDT(-0500)] <phuff> So I'm not screwing something up by just copying the jar over to my production fecru?
[15:23:09 CDT(-0500)] <bspeakmon> no
[15:23:12 CDT(-0500)] <phuff> sweet.

[15:23:20 CDT(-0500)] <phuff> Once again, I greatly appreciate the hand-holding, bspeakmon 😊
[15:23:25 CDT(-0500)] <phuff> This has been my first foray into java 😊

[15:23:25 CDT(-0500)] <bspeakmon> no worries 😊
[15:23:54 CDT(-0500)] <phuff> Hey do you know how fisheye decides when to poll repos?
[15:24:07 CDT(-0500)] <phuff> There are a bunch on our production instance that say "Four hours ago" for the last scanned time.
[15:24:39 CDT(-0500)] <phuff> Where as others are scanned relatively frequently.
[15:25:35 CDT(-0500)] <bspeakmon> it's configurable per repo, I believe
[15:25:42 CDT(-0500)] <phuff> Ok
[15:25:47 CDT(-0500)] <phuff> I couln't find a good place to configure it
[15:25:51 CDT(-0500)] <phuff> I'll just dig deeper.
[15:25:58 CDT(-0500)] <bspeakmon> I know there's something
[15:26:08 CDT(-0500)] <bspeakmon> might be a global setting instead of per repo.
[15:26:09 CDT(-0500)] <bspeakmon> not sure.
[15:26:10 CDT(-0500)] <phuff> The only config option that looked remotely related that I saw was like number of threads
[15:26:13 CDT(-0500)] <phuff> Ok
[15:26:16 CDT(-0500)] <phuff> Thanks, bspeakmon
[15:51:23 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:52:25 CDT(-0500)] * cemerick (~la_mer@64.241.37.140) has joined #atlassiandev
[17:17:49 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[17:17:56 CDT(-0500)] <Broady> anatoli: !!
[17:18:33 CDT(-0500)] <anatoli> @broady ?
[17:18:46 CDT(-0500)] <Broady> sup? it's chris b
[17:22:02 CDT(-0500)] <anatoli> Yeah, I know. I've heard you are starting with us again
[17:22:23 CDT(-0500)] <Broady> waaaaah! where'd you hear that
[17:22:30 CDT(-0500)] <Broady> LOL
[17:28:30 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:28:30 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:15:51 CDT(-0500)] <phuff> Ahhh, there's supposed to be an updater setting for each repo
[18:15:55 CDT(-0500)] <phuff> But there isn't one showing up.
[18:15:57 CDT(-0500)] <phuff> Maybe it's a git thing?
[18:26:06 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:17:49 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[20:21:00 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[20:26:39 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[20:46:54 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:19:54 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[22:19:41 CDT(-0500)] * skrebs_ (~shannon@123.200.231.142) has joined #atlassiandev

atlassiandev_log-2010-03-26

[23:56:17 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[00:15:54 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:16:34 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[02:01:05 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[03:25:38 CDT(-0500)] * nenolod (quasselcor@atheme/member/pdpc.active.nenolod) has joined #atlassiandev
[06:21:03 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev

[07:30:16 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[07:40:04 CDT(-0500)] * cemerick (~la_mer@c-75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[07:55:27 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:47:00 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[11:04:22 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[11:32:36 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[14:17:03 CDT(-0500)] * cemerick (~la_mer@c-75-147-38-122-NewEngland.hfc.comcastbusiness.net) has joined #atlassiandev
[15:11:02 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[17:24:53 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[17:45:17 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:45:17 CDT(-0500)] * ChanServ sets mode +o drosen
[18:12:01 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[18:12:01 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:20:29 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[18:59:29 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:41:53 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-27

[23:55:32 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[02:22:25 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[05:10:13 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[05:13:06 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:18:34 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:26:09 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[10:03:24 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[11:42:42 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[11:51:30 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has left #atlassiandev
[13:02:05 CDT(-0500)] * bug_ (~bug@12.155.186.6) has joined #atlassiandev
[14:02:47 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[16:33:44 CDT(-0500)] * bug_ (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[18:35:45 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[19:04:40 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:15:16 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:23:28 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[21:02:46 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[21:02:46 CDT(-0500)] * ChanServ sets mode +o tmoore
[21:21:05 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[21:40:19 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[21:40:19 CDT(-0500)] * ChanServ sets mode +o tmoore

atlassiandev_log-2010-03-28

[07:11:34 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[07:43:17 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[10:33:43 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[11:45:22 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[11:45:22 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:45:22 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[11:45:22 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[11:45:22 CDT(-0500)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[11:45:22 CDT(-0500)] * Carlfish (cmiller@bodhi.pastiche.org) has joined #atlassiandev
[11:45:22 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[11:45:22 CDT(-0500)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[11:45:22 CDT(-0500)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[11:45:22 CDT(-0500)] * trochej (trochej@noches.pl) has joined #atlassiandev
[11:45:22 CDT(-0500)] * whalej (~whalej@li35-137.members.linode.com) has joined #atlassiandev
[11:45:22 CDT(-0500)] * Ycross (~ycros@gnav.yi.org) has joined #atlassiandev
[11:58:38 CDT(-0500)] * Carlfish (cmiller@bodhi.pastiche.org) has joined #atlassiandev
[11:59:25 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:59:25 CDT(-0500)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[11:59:25 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[11:59:37 CDT(-0500)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[12:01:21 CDT(-0500)] * MartinCleaver_ (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[12:13:24 CDT(-0500)] * Carlfish_ (cmiller@bodhi.pastiche.org) has joined #atlassiandev
[12:16:38 CDT(-0500)] * kalamon (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[12:22:16 CDT(-0500)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[12:31:47 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[12:31:47 CDT(-0500)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[13:27:20 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[14:35:02 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[14:58:41 CDT(-0500)] * whalej (~whalej@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[16:53:01 CDT(-0500)] * mryall (~mryall@59.167.164.33) has joined #atlassiandev
[16:53:01 CDT(-0500)] * ChanServ sets mode +o mryall
[17:28:56 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:28:56 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:17:48 CDT(-0500)] <mryall> morning folks
[18:38:56 CDT(-0500)] <Broady> morning
[18:39:00 CDT(-0500)] <Broady> hey so mryall are you still on the CONF team?

[18:39:04 CDT(-0500)] <mryall> what's up, Chris?
[18:39:18 CDT(-0500)] <Broady> bored @ work. nothing to do really. working on my honours
[18:39:19 CDT(-0500)] <mryall> I'm on secondment to integration, working on embedded Crowd, currently
[18:39:24 CDT(-0500)] <Broady> aha
[18:39:35 CDT(-0500)] <mryall> so yeah, sitting upstairs

[18:40:48 CDT(-0500)] <Broady> shame per is leaving 😕
[18:41:03 CDT(-0500)] <mryall> mmm
[18:41:08 CDT(-0500)] <mryall> it's going to be very weird
[18:41:22 CDT(-0500)] <mryall> there's a bit of a reshuffle going on at the office soon, wrt desks
[18:41:35 CDT(-0500)] <mryall> we've got more office space across the street, so the JIRA and support teams are moving over there
[18:41:41 CDT(-0500)] <Broady> ooooh
[18:41:47 CDT(-0500)] <mryall> I think the Confluence guys might be moving back upstairs again, not sure
[18:41:56 CDT(-0500)] <Broady> d'oh. no more fort confluence
[18:43:52 CDT(-0500)] <tmoore> the JIRA space is ours!
[18:44:14 CDT(-0500)] <mryall> oh, is that what's happening?
[18:44:25 CDT(-0500)] <tmoore> hope so
[18:44:28 CDT(-0500)] <mryall> I've no idea, honestly
[18:44:38 CDT(-0500)] <Broady> who's jira tech lead now?
[18:44:40 CDT(-0500)] <tmoore> well I know Ted wanted to get Studio & Integration together somewhere
[18:44:44 CDT(-0500)] <tmoore> so he doesn't need two desks
[18:44:46 CDT(-0500)] <mryall> just saw Mike and Scott walking around on Friday and heard "mumble mumble Confluence mumble mumble"
[18:45:00 CDT(-0500)] <Broady> mryall: they're probably dropping the product
[18:45:02 CDT(-0500)] <Broady> =p
[18:45:06 CDT(-0500)] <mryall> yeah

[18:45:16 CDT(-0500)] <mryall> was probably "should sell Confluence to Google" or something 😊
[18:45:20 CDT(-0500)] <Broady> haha
[18:45:39 CDT(-0500)] <Broady> i like the completion in 3.2 sehr gut
[18:45:47 CDT(-0500)] <mryall> yeah, it's nice
[18:46:08 CDT(-0500)] <mryall> occasionally glitchy, which is annoying when you're typing something with square brackets
[18:46:28 CDT(-0500)] <mryall> but saves a lot of time opening the link browser
[18:46:44 CDT(-0500)] <Broady> makes the RTE acceptable to use i think
[18:46:44 CDT(-0500)] <mryall> I still want to get something in like PBWorks's link inserter in the sidebar
[18:47:00 CDT(-0500)] <mryall> there should be a quick way of inserting links with the mouse that is always visible
[18:47:24 CDT(-0500)] <Broady> mouse gestures
[18:47:34 CDT(-0500)] <Broady> you just draw the URL with the mouse
[18:47:37 CDT(-0500)] <mryall> at the moment, it's at least three clicks to insert a link with the mouse
[18:47:43 CDT(-0500)] <mryall> which is kind of woeful
[18:47:56 CDT(-0500)] <mryall> and the first tab which opens in the link browser doesn't even let you use your mouse
[18:48:09 CDT(-0500)] <Broady> haha
[18:48:10 CDT(-0500)] <mryall> anyway, I digress
[18:50:11 CDT(-0500)] <mryall> there's a few JIRA tech leads, by the way
[18:50:19 CDT(-0500)] <mryall> and Dylan is still architect until he leaves
[18:50:24 CDT(-0500)] <mryall> which is a couple of months away at least
[18:50:47 CDT(-0500)] <Broady> oh - i meant project manager
[18:51:21 CDT(-0500)] <mryall> Paul is the team lead/project manager
[18:51:29 CDT(-0500)] <mryall> Edwin is product manager
[18:51:32 CDT(-0500)] <Broady> ahhhh
[18:51:53 CDT(-0500)] <mryall> so Anton and Paul just swapped, basically
[18:55:38 CDT(-0500)] <mryall> where's your current placement?
[18:55:48 CDT(-0500)] <Broady> coca cola
[18:56:01 CDT(-0500)] <Broady> i'm 2 months in
[19:13:13 CDT(-0500)] <mryall> cool
[19:15:13 CDT(-0500)] <tmoore> new mac chrome beta
[19:15:18 CDT(-0500)] <tmoore> actually accepts google ssl certificates
[19:15:19 CDT(-0500)] <tmoore> progress!
[19:15:24 CDT(-0500)] <mryall> haha
[19:15:38 CDT(-0500)] <mryall> FF used to warn about them too - do they use wildcard certs?
[19:15:55 CDT(-0500)] <tmoore> no, this looks like it was just a bug in chrome
[19:16:24 CDT(-0500)] <tmoore> <http://code.google.com/p/chromium/issues/detail?id=37722>
[19:19:39 CDT(-0500)] <mryall> ah nice
[19:19:58 CDT(-0500)] <mryall> haha
[19:20:03 CDT(-0500)] <mryall> it didn't even let you access the page before?
[19:20:07 CDT(-0500)] <mryall> that's pretty bad
[19:21:25 CDT(-0500)] <mryall> I wonder if Chrome respects HTTP caching headers properly
[19:21:31 CDT(-0500)] <mryall> should probably test that at some point...
[19:21:40 CDT(-0500)] * dchui (-dchui@202.169.29.34) has joined #atlassiandev
[19:22:18 CDT(-0500)] <tmoore> I never had the problem where I couldn't get in at all
[19:22:27 CDT(-0500)] <tmoore> just had a big red warning I had to click through
[19:22:57 CDT(-0500)] <tmoore> and then in the address bar it puts a strikethrough through the 'https' scheme portion of the URL, highlights it red, and sticks a little yellow warning icon where the padlock normally is
[19:22:59 CDT(-0500)] <tmoore> pretty clever 😊
[19:23:05 CDT(-0500)] <tmoore> when it works
[19:23:13 CDT(-0500)] <tmoore> now the https is green
[19:23:24 CDT(-0500)] <tmoore> OTOH they seem to have removed pinned tabs
[19:23:38 CDT(-0500)] <tmoore> but they added auto-translation of pages in other languages
[19:23:57 CDT(-0500)] * skrebs (-shannon@119.12.88.141) has joined #atlassiandev
[19:24:02 CDT(-0500)] <tmoore> and shift-reload to force refetching everything

[19:24:18 CDT(-0500)] <tmoore> overall, I'm finding lots to like about chrome
[19:32:47 CDT(-0500)] <mryall> yeah, shift-cmd-R not working in Safari is my biggest reason for not developing with it
[19:33:04 CDT(-0500)] <mryall> Chrome dev tools are supposed to be alright too, eh?
[19:34:41 CDT(-0500)] <Broady> i made a chrome extension a couple weeks ago
[19:34:42 CDT(-0500)] <Broady> pretty easy
[19:34:49 CDT(-0500)] <Broady> or do you mean the inspector
[19:35:06 CDT(-0500)] <mryall> yeah, web dev tools
[19:35:10 CDT(-0500)] <Broady> theyre just the webkit ones, so yeah... awesome
[19:35:10 CDT(-0500)] <mryall> Firebug, etc.
[19:35:14 CDT(-0500)] <mryall> oh ok
[19:35:20 CDT(-0500)] <Broady> same as safari
[19:35:23 CDT(-0500)] <mryall> cool
[19:35:46 CDT(-0500)] <mryall> they're sometimes a bit glitchy, but otherwise good
[19:58:31 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:44:06 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[21:36:07 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:31:55 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-29

[00:01:45 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:30:16 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:30:16 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[01:30:16 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[01:30:16 CDT(-0500)] * mryall (~mryall@59.167.164.33) has joined #atlassiandev
[01:30:16 CDT(-0500)] * whalej (~whalej@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[01:30:16 CDT(-0500)] * kalamon_ (~kalamon@chello089074130182.chello.pl) has joined #atlassiandev
[01:30:16 CDT(-0500)] * Broady (~b@unaffiliated/broady) has joined #atlassiandev
[01:30:16 CDT(-0500)] * Carlfish_ (cmiller@bodhi.pastiche.org) has joined #atlassiandev
[01:30:16 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[01:30:16 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[01:30:16 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[01:30:16 CDT(-0500)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[01:30:16 CDT(-0500)] * trochej (trochej@noches.pl) has joined #atlassiandev
[01:30:16 CDT(-0500)] * Ycros (~ycros@gnaw.yi.org) has joined #atlassiandev
[02:41:19 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[02:41:19 CDT(-0500)] * ChanServ sets mode +o tmoore
[03:26:44 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[05:57:57 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[05:57:57 CDT(-0500)] * ChanServ sets mode +o tmoore
[07:58:55 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[08:27:55 CDT(-0500)] * igorsereda (~6dbce0e6@gateway/web/freenode/x-ofcjnoyresblodooc) has joined #atlassiandev
[08:29:08 CDT(-0500)] <igorsereda> hey guys
[08:30:00 CDT(-0500)] <igorsereda> does anyone know what's up with the soap api in the latest jira 4.1 beta?
[08:50:56 CDT(-0500)] * puskar (~puskar@PUSKAR.ES.ITS.NYU.EDU) has joined #atlassiandev
[08:59:56 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:29:34 CDT(-0500)] * AJC_Z0 (~AJCZ0@pdpc/supporter/professional/ajz0) has joined #atlassiandev
[10:19:41 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[11:57:19 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:57:19 CDT(-0500)] * ChanServ sets mode +o drosen
[12:30:18 CDT(-0500)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:30:18 CDT(-0500)] * ChanServ sets mode +o bspeakmon
[13:27:52 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[13:37:26 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[13:42:04 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has left #atlassiandev
[16:38:33 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:38:33 CDT(-0500)] * ChanServ sets mode +o drosen
[16:45:37 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[16:46:52 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:46:52 CDT(-0500)] * ChanServ sets mode +o tmoore
[17:24:32 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[17:43:30 CDT(-0500)] * mryall (~mryall@59.167.164.33) has joined #atlassiandev
[17:43:30 CDT(-0500)] * ChanServ sets mode +o mryall
[17:45:49 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:45:49 CDT(-0500)] * ChanServ sets mode +o tmoore
[19:21:04 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[19:43:58 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:14:25 CDT(-0500)] * skrebs_ (~shannon@121.91.61.81) has joined #atlassiandev
[20:17:31 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[20:32:53 CDT(-0500)] <skrebs> does anyone know if there is any documentation on how to get i18n working in Jira plugins?
[20:35:19 CDT(-0500)] <mrdon> define a <resource type="i18n" ...> element
[20:36:32 CDT(-0500)] <skrebs> hurrm, did try that, will test again
[20:36:51 CDT(-0500)] <mryall> there's some draft doco here:
<http://confluence.atlassian.com/display/DOCSPRINT/Internationalising+your+Plugin>
[20:37:31 CDT(-0500)] <skrebs> @mryall "Page level restrictions have been applied that limit access to this page. "
[21:45:39 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-03-30

[23:51:32 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[02:52:16 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[03:33:55 CDT(-0500)] * skrebs (~shannon@121.91.61.81) has joined #atlassiandev
[04:47:06 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[05:23:03 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[05:47:06 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[05:51:59 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:00:03 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[06:00:03 CDT(-0500)] * ChanServ sets mode +o tmoore
[07:22:24 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[07:30:50 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[07:31:11 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[09:40:15 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[10:23:41 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[12:22:58 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[12:39:47 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[13:18:51 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:18:59 CDT(-0500)] * ChanServ sets mode +o drosen
[13:52:14 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[14:40:17 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:10:46 CDT(-0500)] * arnoldito (~whaley@c-24-127-238-185.hsd1.fl.comcast.net) has joined #atlassiandev
[16:14:55 CDT(-0500)] * bug (~bug@65-112-21-194.dia.static.qwest.net) has joined #atlassiandev
[17:12:19 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[17:22:42 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:24:41 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[17:24:41 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:46:47 CDT(-0500)] * skrebs (~shannon@121.91.61.81) has joined #atlassiandev
[18:54:33 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:17:18 CDT(-0500)] * arnoldito (~whaley@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[20:17:30 CDT(-0500)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[21:32:03 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:49:47 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[22:10:40 CDT(-0500)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev
[23:10:37 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev

atlassiandev_log-2010-03-31

[23:35:10 CDT(-0500)] * bspeakmon_ (~bspeakmon@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[23:35:18 CDT(-0500)] * drosen_ (~drosen@dsl092-186-178.sfo1.dsl.speakeasy.net) has joined #atlassiandev
[23:35:18 CDT(-0500)] * ChanServ sets mode +o drosen_
[23:36:43 CDT(-0500)] * bspeakmon_ (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[23:36:55 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[23:36:55 CDT(-0500)] * ChanServ sets mode +o drosen
[03:09:55 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[06:04:58 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[08:12:29 CDT(-0500)] * nenolod_ (~quassel@nenolod.net) has joined #atlassiandev
[08:12:29 CDT(-0500)] * nenolod_ (~quassel@atheme/member/pdpc.active.nenolod) has joined #atlassiandev
[08:46:25 CDT(-0500)] * whaleys (~whaleys@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[08:58:31 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[11:17:21 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:33:10 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[12:49:46 CDT(-0500)] * ChanServ sets mode +o bspeakmon
[14:59:40 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[14:59:40 CDT(-0500)] * ChanServ sets mode +o tmoore
[15:47:55 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[16:21:32 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[16:42:10 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:42:10 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:04:04 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[19:27:49 CDT(-0500)] * whaleys (~whaleys@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[19:51:25 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[22:34:04 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-04-01

[23:19:49 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[01:16:00 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[04:18:06 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:51:05 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[06:59:08 CDT(-0500)] * dsc (~dcorbin@72.242.104.2) has joined #atlassiandev
[07:00:01 CDT(-0500)] <dsc> When I click on a bug # on the task board, I get a "pseudo dialog" for the bug. Is there anyway to make it just take me to the full-page for the bug?
[07:53:32 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:52:58 CDT(-0500)] * MartinCleaver (~martincle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[10:04:12 CDT(-0500)] * kevindank (>chatzilla@205.232.42.46) has joined #atlassiandev
[10:04:29 CDT(-0500)] <kevindank> Hello, I need some help/have a question about Jira Standard 3.13
[10:05:05 CDT(-0500)] <kevindank> is it possible when a ticket is closed in a specific queue to have a popup/model box display a "Rate my Performance" form
[10:05:18 CDT(-0500)] <kevindank> or basically just in general have a pop display on ticket close

[10:34:17 CDT(-0500)] <kevindank> or a way for it to send an e-mail to reporter after the ticket is closed
[11:19:56 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:27:20 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:56:37 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:56:37 CDT(-0500)] * ChanServ sets mode +o drosen
[12:34:02 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev

[13:54:05 CDT(-0500)] <bug> hey, where's my fourwalls iPhone app? 
[13:54:51 CDT(-0500)] <bug> (apologies for rehashing the joke, assuming its already been hashed)
[13:56:37 CDT(-0500)] <bspeakmon> you have to consent to a subcutaneous RFID chip
[13:58:30 CDT(-0500)] * bug consents
[13:58:48 CDT(-0500)] <bspeakmon> the Atlassian Compliance Squad will be by shortly (for some value of "shortly")
[13:59:00 CDT(-0500)] <bug> but i thought that was just for australian and british citizens?
[13:59:10 CDT(-0500)] <bspeakmon> same thing
[13:59:29 CDT(-0500)] <bug> i guess for this purpose, true
[13:59:42 CDT(-0500)] <bug> in the US, we're just monitored via brainwave from space
[14:00:22 CDT(-0500)] <bug> we'll shortly be launching conspiracytheory.gov
[14:54:16 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[16:25:06 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[16:58:45 CDT(-0500)] * whale (~whale@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[17:04:49 CDT(-0500)] * arnoldito (~whale@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[17:23:36 CDT(-0500)] * MartinCleaver (~marticle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[17:24:18 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[18:11:33 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[18:18:35 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[18:18:35 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:29:10 CDT(-0500)] * skrebs (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[18:54:56 CDT(-0500)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[18:55:37 CDT(-0500)] * skrebs_ (~shannon@220-245-106-10.static.tpgi.com.au) has joined #atlassiandev
[19:35:00 CDT(-0500)] * whale (~whale@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[20:24:53 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[21:12:29 CDT(-0500)] * MartinCleaver (~marticle@69-196-163-99.dsl.teksavvy.com) has joined #atlassiandev
[22:34:00 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev

atlassiandev_log-2010-04-02

[23:21:58 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[23:24:15 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[00:44:33 CDT(-0500)] * kevindank_ (~chatzilla@205.232.42.46) has joined #atlassiandev
[06:31:09 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:16:17 CDT(-0500)] * bug_ (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:52:54 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[08:38:11 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[09:04:18 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[09:54:57 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[10:16:00 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[11:19:21 CDT(-0500)] * whale (~whale@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[11:19:28 CDT(-0500)] * bspeakmon (~bspeakmon@c-98-210-37-165.hsd1.ca.comcast.net) has joined #atlassiandev
[11:19:28 CDT(-0500)] * ChanServ sets mode +o bspeakmon
[11:46:15 CDT(-0500)] <AJC_Z0> Atlassian mentioned as the first example a company which understands motivation - http://www.ted.com/talks/dan_pink_on_motivation.html
[11:46:46 CDT(-0500)] <AJC_Z0> (around the 15:00 mark for the impatient, but it's an interesting talk)
[11:53:21 CDT(-0500)] * whale (~whale@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[12:04:24 CDT(-0500)] <drosen> AJC_Z0: i saw that a while back, very interesting talk
[12:04:50 CDT(-0500)] <drosen> it was sent to me by a former co-worker, which was an example of a company that definitely didn't understand motivation
[13:36:41 CDT(-0500)] * rodogu1 (~Adium@d154-5-174-17.bchsa.telus.net) has joined #atlassiandev
[14:22:46 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[14:44:23 CDT(-0500)] * matthew (~11e221f@gateway/web/freenode/x-vyxahunskmbxlcaf) has joined #atlassiandev
[14:44:53 CDT(-0500)] <Guest77784> hello, looking for help with masking urls
[14:45:25 CDT(-0500)] <Guest77784> we had this working with 2.8 but it seems to have broken with 3.1
[14:46:12 CDT(-0500)] <Guest77784> we have a support contract this is for the Apple iWork Team
[14:48:29 CDT(-0500)] <bspeakmon> you probably want to open a ticket at support.atlassian.com then
[14:48:49 CDT(-0500)] <Guest77784> we did, but nothing custom is supported
[14:48:59 CDT(-0500)] <Guest77784> they recommended we post here
[14:49:23 CDT(-0500)] <bspeakmon> you can also check at forums.atlassian.com
[14:49:29 CDT(-0500)] <bspeakmon> if nobody here has any idea
[14:49:35 CDT(-0500)] <bspeakmon> sydney is on vacation right now
[14:50:16 CDT(-0500)] <Guest77784> we has a simple script that would mask all urls as site.my_server.com/go.html
[14:50:40 CDT(-0500)] <Guest77784> it seems something changed in the more recent releases and not it no longer works
[14:51:49 CDT(-0500)] <Guest77784> the following code was placed into the viewpage.vm
[14:51:50 CDT(-0500)] <Guest77784> <body> <!-- wiki content --> #set (\$externalLinkRegex = "href=\"(https?|s?ftp)://([^\"]+\\\")" #set
(\$externalLinkRedirector = "href=\"http://site.apple.com/go.html?\${1://\$2}"" #pageXHtmlContent.replaceAll(\$externalLinkRegex,
\$externalLinkRedirector) </body>
[14:58:14 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[15:25:16 CDT(-0500)] * whale (~whale@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[16:34:45 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[16:52:56 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:16:01 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev

```
[17:24:19 CDT(-0500)] * twong (~twong@216-75-233-106.static.wiline.com) has joined #atlassiandev
[17:47:11 CDT(-0500)] * bug (~bug@65-112-21-194.dia.static.qwest.net) has joined #atlassiandev
[19:23:15 CDT(-0500)] * whalejy (~whalejy@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[20:23:25 CDT(-0500)] * whalejy (~whalejy@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[21:43:12 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
```

atlassiandev_log-2010-04-03

```
[04:55:01 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[04:55:01 CDT(-0500)] * ChanServ sets mode +o tmoore
[05:26:36 CDT(-0500)] * whalejy (~whalejy@64.241.37.140) has joined #atlassiandev
[06:15:06 CDT(-0500)] * arnoldito (~whalejy@64.241.37.140) has joined #atlassiandev
[09:30:15 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[12:31:51 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:31:46 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[17:08:42 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:00:56 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[20:45:31 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
```

atlassiandev_log-2010-04-04

```
[04:46:24 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[04:46:24 CDT(-0500)] * ChanServ sets mode +o tmoore
[16:01:54 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:43:20 CDT(-0500)] * twong (~twong@c-69-181-215-191.hsd1.ca.comcast.net) has joined #atlassiandev
[17:48:22 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[18:41:44 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[18:41:44 CDT(-0500)] * ChanServ sets mode +o tmoore
[19:20:03 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[19:20:03 CDT(-0500)] * ChanServ sets mode +o tmoore
[19:31:27 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[22:08:05 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:28:35 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
```

atlassiandev_log-2010-04-05

```
[23:16:13 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[00:34:46 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[03:18:57 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[04:12:27 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[06:06:37 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[06:06:37 CDT(-0500)] * ChanServ sets mode +o tmoore
[06:46:34 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[07:06:21 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[08:50:15 CDT(-0500)] * whalejy (~whalejy@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[08:53:45 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[11:43:14 CDT(-0500)] * whalejy (~whalejy@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[12:24:36 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:24:37 CDT(-0500)] * ChanServ sets mode +o drosen
[14:27:11 CDT(-0500)] * whalejy (~whalejy@c-98-224-44-53.hsd1.fl.comcast.net) has joined #atlassiandev
[14:35:30 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:13:19 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[15:32:13 CDT(-0500)] * rodogu (~Adium@S010600090f525567.vc.shawcable.net) has joined #atlassiandev
[16:01:35 CDT(-0500)] * bug (~bug@adsl-71-135-163-159.dsl.pltn13.pacbell.net) has joined #atlassiandev
[16:51:32 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[18:19:17 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:19:17 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:39:01 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
```

atlassiandev_log-2010-04-06

```
[23:36:35 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[01:34:17 CDT(-0500)] <jazzy> rodogu: two years ago, I did some work with Apache xmlrpc client, and from what I remember, the upgrade to version 3 is non trivial, I doubt a drop in will work
[01:35:53 CDT(-0500)] <jazzy> including in your jar files on the other hand... i'll have to check if the confluence plugins 1 classloader is parent first
[01:37:04 CDT(-0500)] <jazzy> it's parent first, so including it in the jar file won't work because you'll still get the old version
[01:37:40 CDT(-0500)] <jazzy> if you use plugins 2, as long as you get your osgi manifests right, you should be able to use the new version with no problems
[05:40:39 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[09:11:56 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[12:42:15 CDT(-0500)] * Carlfish (cmiller@bodhi.pastiche.org) has joined #atlassiandev
[15:06:50 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:28:23 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:28:23 CDT(-0500)] * ChanServ sets mode +o tmoore
[19:10:55 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[19:18:48 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
```

[20:02:57 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[20:17:59 CDT(-0500)] * ed1t (~edited@unaffiliated/ed1t) has joined #atlassiandev
[20:25:24 CDT(-0500)] <ed1t> I am trying to build jira but I'm getting the "Unable to find resource 'com.atlassian.jira:atlassian-jira:jar:4.0.2-SNAPSHOT" error
[20:25:43 CDT(-0500)] <ed1t> Missing: com.atlassian.jira:atlassian-jira:jar:4.0.2-SNAPSHOT
[20:55:22 CDT(-0500)] * ed1t_ (~edited@2002:ad03:7f3c:0:21e:c2ff:feb8:c508) has joined #atlassiandev
[20:55:22 CDT(-0500)] * ed1t_ (~edited@unaffiliated/ed1t) has joined #atlassiandev
[21:03:53 CDT(-0500)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[21:04:01 CDT(-0500)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[21:04:29 CDT(-0500)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[21:26:03 CDT(-0500)] <ed1t> anyone here know how to use my google app engine with my jira?
[21:26:09 CDT(-0500)] <ed1t> app engine email
[21:36:49 CDT(-0500)] * ed1t (~edited@2002:ad03:7f3c:0:21e:c2ff:feb8:c508) has joined #atlassiandev
[21:36:49 CDT(-0500)] * ed1t (~edited@unaffiliated/ed1t) has joined #atlassiandev
[22:20:31 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-04-07

[00:40:39 CDT(-0500)] * jedi (mike@li55-224.members.linode.com) has joined #atlassiandev
[01:45:45 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[04:08:48 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-215.hsd1.ma.comcast.net) has joined #atlassiandev
[07:08:38 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[10:00:30 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[12:34:17 CDT(-0500)] * dstanek (~dstanek@global.ag.com) has joined #atlassiandev
[12:34:39 CDT(-0500)] <dstanek> is it possible to open a window from within a gadget?
[12:35:35 CDT(-0500)] <dstanek> i started developing a gadget on IG and it worked there, once i moved to Jira it stopped working
[12:39:51 CDT(-0500)] <dstanek> hmm...it actually may be my jquery.contextmenu that is not working
[12:57:18 CDT(-0500)] <dstanek> it appears that firefox will do the open only if i have an alert('whatever') on the prior line
[13:42:46 CDT(-0500)] * superemily (~emily@199.199.210.153) has joined #atlassiandev
[13:43:00 CDT(-0500)] <superemily> aha here it is
[13:43:16 CDT(-0500)] * gjoseph (~gjoseph@firewall.magnolia-cms.com) has joined #atlassiandev
[13:51:15 CDT(-0500)] * puskar (~puskar@PUSKAR.ES.ITS.NYU.EDU) has joined #atlassiandev
[13:56:26 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[14:00:09 CDT(-0500)] * superemily (~emily@199.199.210.153) has left #atlassiandev
[14:01:50 CDT(-0500)] * gjoseph (~gjoseph@firewall.magnolia-cms.com) has left #atlassiandev
[14:12:17 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[14:55:52 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[14:59:17 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[16:25:37 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[16:52:04 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[16:59:48 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[16:59:48 CDT(-0500)] * ChanServ sets mode +o tmoore
[17:31:09 CDT(-0500)] * bug (~bug@65-112-21-194.dia.static.qwest.net) has joined #atlassiandev
[18:19:30 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:19:30 CDT(-0500)] * ChanServ sets mode +o tmoore
[18:23:56 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[18:24:14 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[19:03:08 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[19:20:26 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[19:38:13 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[20:09:42 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[21:37:18 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-04-08

[02:54:50 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[02:55:43 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[04:09:10 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[06:33:12 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:50:31 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[10:12:16 CDT(-0500)] * MartinCleaver (~martincle@CPE002369def1ec-CM000e5c6dba64.cpe.net.cable.rogers.com) has joined #atlassiandev
[10:52:13 CDT(-0500)] * MartinCleaver (~martincle@2002:63e7:d408:1234:21e:52ff:fe75:82fc) has joined #atlassiandev
[10:55:04 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[12:08:26 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[12:31:35 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:33:41 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[12:39:41 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[12:56:26 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[12:59:36 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[13:14:21 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[13:21:11 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[13:25:48 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[13:35:14 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[13:59:15 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[14:46:38 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[15:56:09 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[16:12:13 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[16:14:36 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev

[16:24:30 CDT(-0500)] * rodogu1 (~Adium@d154-5-174-17.bchisia.telus.net) has joined #atlassiandev
[16:28:56 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[18:24:16 CDT(-0500)] * anatoli (~anatoli@59.167.164.33) has joined #atlassiandev
[18:32:26 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:04:30 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:09:23 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:13:01 CDT(-0500)] * dchui (~dchui@202.169.29.35) has joined #atlassiandev
[19:37:12 CDT(-0500)] * dchui (~dchui@202.169.29.35) has joined #atlassiandev
[19:46:24 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[20:04:39 CDT(-0500)] * rodogu (~Adium@d154-5-174-17.bchisia.telus.net) has joined #atlassiandev
[20:12:17 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[20:52:08 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[21:05:19 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[22:29:00 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-04-09

[01:59:25 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[07:38:08 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[08:11:25 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[08:38:31 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[08:47:08 CDT(-0500)] * puskar (~puskar@PUSKAR.ES.ITS.NYU.EDU) has joined #atlassiandev
[08:47:38 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[10:13:44 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[10:52:14 CDT(-0500)] * kevindank (~chatzilla@205.232.42.46) has joined #atlassiandev
[10:52:31 CDT(-0500)] <kevindank> Anyone in here write jira plugins?
[11:50:48 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:52:56 CDT(-0500)] * psytek (~psytek@rrcs-72-43-189-138.nyc.biz.rr.com) has joined #atlassiandev
[12:53:22 CDT(-0500)] <psytek> is there a way to require a due date only for a specific user?
[12:53:29 CDT(-0500)] <psytek> in Jira
[14:54:32 CDT(-0500)] * psytek (~psytek@rrcs-72-43-189-138.nyc.biz.rr.com) has left #atlassiandev
[15:17:01 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[17:37:45 CDT(-0500)] * outofrange1 (~moseley@174.47.2.0) has joined #atlassiandev
[17:39:38 CDT(-0500)] <outofrange1> In Jira 4.0.1 is there a way to search for all tickets touched (updated, closed, commented, etc.) by a given user over some time period – say in the last week?
[18:39:39 CDT(-0500)] * outofrange1 (~moseley@174.47.2.0) has left #atlassiandev
[20:08:55 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[21:24:50 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[21:51:43 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:52:13 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-04-10

[23:23:08 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[00:16:56 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[00:45:18 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[08:19:42 CDT(-0500)] * MartinCleaver (~martincle@206-248-156-161.dsl.teksavvy.com) has joined #atlassiandev
[09:23:30 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[12:59:17 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[13:07:11 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[14:31:49 CDT(-0500)] * bug_ (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:15:26 CDT(-0500)] * bug_ (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[17:49:10 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:10:14 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[18:15:22 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[18:39:54 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[19:28:34 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[20:16:01 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[20:52:30 CDT(-0500)] * minge (~minge@adsl-90-10-63.mob.bellsouth.net) has joined #atlassiandev
[20:54:34 CDT(-0500)] <minge> is there a way to get activity stream events from jira, fisheye, etc... sent to an IRC channel?
[21:12:32 CDT(-0500)] * minge (~minge@adsl-90-10-63.mob.bellsouth.net) has left #atlassiandev
[21:31:20 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev
[22:37:13 CDT(-0500)] * _hen (~hen@c-76-121-50-243.hsd1.wa.comcast.net) has joined #atlassiandev

atlassiandev_log-2010-04-11

[00:49:23 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[07:29:10 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[11:58:39 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[12:37:22 CDT(-0500)] * bug_ (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[12:46:43 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[15:15:00 CDT(-0500)] * bug (~bug@c-71-198-76-230.hsd1.ca.comcast.net) has joined #atlassiandev
[16:42:37 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[19:02:27 CDT(-0500)] * jazzy_ (~jazzy@four.entic.net) has joined #atlassiandev
[19:08:40 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[19:08:40 CDT(-0500)] * ChanServ sets mode +o tmoore
[22:17:12 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev

atlassiandev_log-2010-04-12

[23:21:31 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[02:41:39 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[03:23:03 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[03:33:59 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[03:33:59 CDT(-0500)] * ChanServ sets mode +o tmoore
[05:43:02 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[06:47:16 CDT(-0500)] * dcorbin_work (~dcorbin@unaffiliated/dcorbin) has joined #atlassiandev
[06:47:50 CDT(-0500)] <dcorbin_work> On the task board, there's a colored status bar (green, yellow, red). Is that "user configurable" in anyway?
[07:25:31 CDT(-0500)] * tmoore (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[07:25:32 CDT(-0500)] * ChanServ sets mode +o tmoore
[07:39:29 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[07:57:21 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[08:12:03 CDT(-0500)] * MartinCleaver (~martincle@69-196-140-172.dsl.teksavy.com) has joined #atlassiandev
[10:30:22 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[11:53:51 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[11:53:58 CDT(-0500)] * ChanServ sets mode +o drosen
[12:06:38 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:15:10 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[12:19:54 CDT(-0500)] * bspeakmon (~bspeakmon@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:19:55 CDT(-0500)] * ChanServ sets mode +o bspeakmon
[12:25:02 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:48:31 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[12:48:59 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:06:30 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[13:06:30 CDT(-0500)] * ChanServ sets mode +o drosen
[14:31:05 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[15:16:39 CDT(-0500)] * JonathanNolen (~jnolen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[16:32:06 CDT(-0500)] * chuck (~charlie@yourwiki/staff/charlie) has joined #atlassiandev
[18:57:38 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[18:57:38 CDT(-0500)] * ChanServ sets mode +o tmoore
[19:16:23 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[19:32:42 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:00:22 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[21:23:27 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[21:46:49 CDT(-0500)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[23:07:52 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev

atlassiandev_log-2010-04-13

[00:35:26 CDT(-0500)] * jazzy (~jazzy@four.entic.net) has joined #atlassiandev
[00:35:26 CDT(-0500)] * sebr (~sruiz@amarok/developer/sebr) has joined #atlassiandev
[00:35:26 CDT(-0500)] * tmoore (~tmoore@59.167.164.33) has joined #atlassiandev
[00:35:26 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[00:35:26 CDT(-0500)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[00:35:26 CDT(-0500)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev
[00:35:26 CDT(-0500)] * nyerup (irc@jespernyerup.dk) has joined #atlassiandev
[00:35:26 CDT(-0500)] * Ycros (~ycros@gnav.yi.org) has joined #atlassiandev
[00:35:26 CDT(-0500)] * trochej (trochej@noches.pl) has joined #atlassiandev
[00:37:04 CDT(-0500)] * drosen (~drosen@216-75-233-106.static.wiline.com) has joined #atlassiandev
[00:37:05 CDT(-0500)] * mrdon (~mrdon@59.167.164.33) has joined #atlassiandev
[03:00:25 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[03:28:55 CDT(-0500)] * pramod (~pramod@59.167.164.33) has joined #atlassiandev
[03:48:04 CDT(-0500)] * pramod (~pramod@59.167.164.33) has left #atlassiandev
[04:15:21 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[05:47:13 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[05:53:06 CDT(-0500)] * tmoore1 (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[07:58:11 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[11:06:46 CDT(-0500)] * MartinCleaver (~martincle@69-196-140-172.dsl.teksavy.com) has joined #atlassiandev
[11:52:16 CDT(-0500)] * dcorbin_wrk (~dcorbin@72.242.104.2) has joined #atlassiandev
[12:37:50 CDT(-0500)] * bug (~bug@w118.z067104062.sjc-ca.dsl.cnc.net) has joined #atlassiandev
[13:58:57 CDT(-0500)] * jmort (~jmort@93-96-212-135.zone4.bethere.co.uk) has joined #atlassiandev
[16:33:37 CDT(-0500)] * tmoore1 (~tmoore@CPE-58-172-105-250.ezsb2.ken.bigpond.net.au) has joined #atlassiandev
[16:38:07 CDT(-0500)] * bug (~bug@adsl-76-192-234-227.dsl.pltn13.sbcglobal.net) has joined #atlassiandev
[16:41:42 CDT(-0500)] * JonathanNolen (~jnolen@c-67-188-110-99.hsd1.ca.comcast.net) has joined #atlassiandev
[17:58:18 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[18:05:54 CDT(-0500)] <phuff> Hey there.
[18:08:20 CDT(-0500)] <phuff> Can I update a crucible review's description programmatically after the fact?
[18:08:51 CDT(-0500)] <phuff> I don't see anything in crucible.spi.services.ReviewService that will let me do it.
[18:09:36 CDT(-0500)] <phuff> Like, I want to be able to upate the description if I add revisions to it later on.
[18:12:05 CDT(-0500)] <bspeakmon> I'm not sure
[18:21:19 CDT(-0500)] <phuff> ok
[18:30:06 CDT(-0500)] * tmoore1 (~tmoore@59.167.164.33) has joined #atlassiandev
[18:46:04 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[18:52:01 CDT(-0500)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev
[19:04:16 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[19:53:03 CDT(-0500)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev

```
[19:55:27 CDT(-0500)] * Carlfish_ (~cmiller@216-75-233-106.static.wiline.com) has joined #atlassiandev
[19:56:46 CDT(-0500)] * mryall (~mryall@59.167.164.33) has joined #atlassiandev
[19:56:46 CDT(-0500)] * ChanServ sets mode +o mryall
[20:38:24 CDT(-0500)] * sebr (~seb@amarok/developer/sebr) has joined #atlassiandev
[20:40:45 CDT(-0500)] * tmoore1 (~tmoore@59.167.164.33) has joined #atlassiandev
[21:12:31 CDT(-0500)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev
[21:22:07 CDT(-0500)] * jus (~justin@59.167.164.33) has joined #atlassiandev
[21:30:58 CDT(-0500)] * tmoore1 (~tmoore@59.167.164.33) has joined #atlassiandev
[21:59:05 CDT(-0500)] * Carlfish_ (~cmiller@216-75-233-106.static.wiline.com) has joined #atlassiandev
[22:01:12 CDT(-0500)] * bug (~bug@75-151-78-9-sfba.hfc.comcastbusiness.net) has joined #atlassiandev
[22:10:35 CDT(-0500)] * tmoore1 (~tmoore@59.167.164.33) has joined #atlassiandev
[22:21:36 CDT(-0500)] * phuff (~phuff@unaffiliated/phuff) has joined #atlassiandev
[22:56:54 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has joined #atlassiandev
[22:57:09 CDT(-0500)] * dezwart (~pdzwart@59.167.164.33) has left #atlassiandev
[23:02:42 CDT(-0500)] * Carlfish_ (~cmiller@c-76-21-32-65.hsd1.ca.comcast.net) has joined #atlassiandev
```

atlassiandev_log-2010-04-14

```
[23:18:26 CDT(-0500)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev
[23:20:45 CDT(-0500)] * cemerick (~la_mer@c-71-192-209-171.hsd1.ma.comcast.net) has joined #atlassiandev
[00:06:01 CDT(-0500)] * dchui (~dchui@202.169.29.34) has joined #atlassiandev
[01:11:49 CDT(-0500)] * pleschev (~pleschev@59.167.164.33) has joined #atlassiandev
```

Plugin Testing Resources and Discussion

This page has a short URL: <http://j.mp/plugin-testing>

Plugin Testing Topics

- Writing Tests for your Plugin
- Adding Selenium to Functional Tests
- Functional Testing with Multiple Products
- Product Data for Integration Testing
- Plugin Tutorial - Writing Unit Tests for your Plugin
- Plugin Tutorial - Writing Integration Tests for your JIRA plugin
- atlas-unit-test
- atlas-integration-test

External Articles on Plugin Testing

- Integration Testing of Confluence Plugins by Bo Wang
- Confluence plugin integration testing by Roberto Dominguez
- Confluence Plugin Testing by Dan Kigelman
- Unit testing a Confluence Action by Edward Robertshaw

General Testing Articles

- TDD Anti-Patterns by James Carr
- JUnit Anti-patterns by Joe Schmetzer
- JUnit Cookbook by Kent Beck and Erich Gamma
- Introduction to Test Driven Design by Scott W. Ambler
- Mocks Aren't Stubs by Martin Fowler
- Code Unit Test First on the C2 Wiki (the first wiki and one of the original sources of information on agile development)

Tools

- JUnit
- Mockito
- Hamcrest
- Clover
- HtmlUnit
- JWebUnit
- Atlassian Plugin SDK Documentation
- AMPS Plugin for Maven
- Selenium
- maven-itblast-plugin

Other Resources

- Test Driven Development: By Example book by Kent Beck
- Test-driven Development Yahoo! Group
- Tim Moore's AtlasCamp 2009 Presentation on Plugin Testing

Mailing Lists and Support

Forums and Mailing lists exclusively for developers and development questions. Please post all queries to the appropriate Developer Forum. This is a terrific community resource for exchanging knowledge with other plugin developers. You can use the web-based forum or the email-based mailing list, whichever you choose.

| | |
|-----------------------|--------|
| JIRA Developers | Forums |
| Confluence Developers | Forums |
| Bamboo Developers | Forums |
| Crowd Developers | Forums |
| FishEye Developers | Forums |
| Crucible Developers | Forums |

Contributing to the Atlassian Developer Documentation

Would you like to share your hints, tips and techniques for developing with one or more of the Atlassian applications? We welcome your contributions. Have you found a mistake in the documentation, or do you have a small addition that would be so easy to add yourself rather than asking us to do it? You can update the documentation page directly.

To update the [Developer Network](#) or other developer-focused wiki spaces, just sign up for a wiki username then log in and make the change.

If you want to update the user's and administrator's guides, take a look at the full guide to [contributing to the Atlassian documentation](#).

Codegeist V

We've kicked off [Codegeist V](#), and this page contains links to the documentation resources you'll need for the contest.

Speakeasy Extensions

Speakeasy is a new, experimental extension mechanism for Atlassian's products. Speakeasy extensions are designed to be easy and shareable. They are front-end only: no Java code, just web standards (HTML, Javascript, CSS, etc.). The Speakeasy Framework is currently a developer preview; this is the first time it's been shared outside of Atlassian.

- [Speakeasy Overview](#)
- [Speakeeasy Install Guide](#)
- [Speakeeasy Extension Development Guide](#)
- [Speakeeasy Extension Examples](#)
- [Speakeeasy Plugin Development](#)

Join the [SpeakEasy Google Group](#) and give us any feedback you have.

General Plugin Development

In addition to Speakeasy, Atlassian has a powerful plugin platform that enables developers to build almost anything. The [Writing your first plugin](#) doc has all you need to build an Atlassian plugin. And we've got a ton of cookbook-style [Tutorials](#). We've also created a very comprehensive multi-part tutorial about modern plugin architecture, which should be a great way to get started, too. Take a look:

- [Part 1: A New Hope](#)
- [Part 2: The 3-Tiered Architecture Strikes Back](#)
- [Part 3: Return of the jQuery](#)
- [Part 4: The Phantom Mess](#)

You can also take a look at the plugins that have already been developed on the [Atlassian Plugin Exchange](#), as well as checking out all of the parts of our applications you can change:

- [JIRA](#)
- [Confluence](#)
- [FishEye & Crucible](#)
- [Bamboo](#)

ActiveObjects

We're almost finished incorporating a new ORM layer into our products that enables significantly easier, faster and more scalable data access and storage than our old Bandana and PluginSettings APIs. This ORM is called ActiveObjects, based on the open source project initially developed by Daniel Spiewak. ActiveObjects is currently a developer preview that we're just now sharing outside of Atlassian.

- Overview
- Getting started

Give us Feedback on ActiveObjects.

Speakeasy Admin Guide

The following are problems that may occur with Speakeeasy and their solution:

| Problem | Solution |
|---|---|
| User can't access the Speakeeasy page due to a rogue extension | <p>Have the user visit</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> http://THE_SERVER_URL/plugins/servlet/speakeeasy/unsubscribe </div> <p>to be removed from all Speakeeasy extension access lists</p> |
| User is having UI issues that may be due to a Speakeeasy extension | If the user knows the probable extension, they can visit the Speakeeasy page in their user profile and disable it. Otherwise, they can use the unsubscribe URL listed above to automatically unsubscribe from all Speakeeasy extensions |
| Speakeeasy has a bug and one of the extensions is polluting the UI for everyone | First, try to disable the Speakeeasy extension by disabling its plugin in the UPM. If that doesn't work, you can disable Speakeeasy itself also in the UPM. When you re-enable Speakeeasy, all user extension subscriptions will be preserved. |
| After upgrading Speakeeasy, some extensions are missing | The plugin system has a bug that prevents Speakeeasy from being upgraded by simply installing the new version. You have to explicitly uninstall the previous Speakeeasy plugin then install the upgraded jar. |

Speakeeasy Extension Development Guide

Speakeeasy lets you to write a special, simple type of Atlassian Plugin, called an extension. The extension can then be shared with others by allowing them to opt-in, or enable, the extension on a person-by-person basis. This dramatically reduces the risk to server operation and other users, while giving you an opportunity to try out our plugin idea with production data and use it on a daily basis.

- **What can extensions do?**
 - Great, but what can they *NOT* do?
- **Getting started**
 - Fork an existing extension
- **Developing your extension**
 - Using the Web IDE
 - Developing Offline
 - Using IDEA
 - Using git
- **Advanced Topics**
 - Using Web Items
 - Referencing Images in CSS
 - Using CommonJS Modules
 - Contextualizing modules
 - Consuming modules
 - Providing modules
 - Using the Plugin SDK
 - Under the covers

What can extensions do?

Extensions are currently limited to all things client-side:

- JavaScript
- CSS
- Web items/sections/panels
- Images
- HTML

As such, you can do things like:

- Add, modify, or remove UI elements
- Provide shortcuts for sets of operations via XHR automation
- Incorporate external data via consumption of JSONP

Even better, take a look at the [Speakeeasy Extension Examples](#) for ideas.

Great, but what can they NOT do?

Since no server-side code is allowed, your extension cannot:

- Implement arbitrary application plugin modules
- Access product information not exposed via REST, XML-RPC, or SOAP
- Provide new URLs
- Retrieve data from remote servers that don't support JSONP

* You can work around both these issues if you get creative. In the first case, you can hijack an existing URL and pass additional data via query parameters or anchors, then rewriting the DOM. For the second, if the remote server is public, you can use YQL to access that server and repackage the data into JSONP.

Getting started

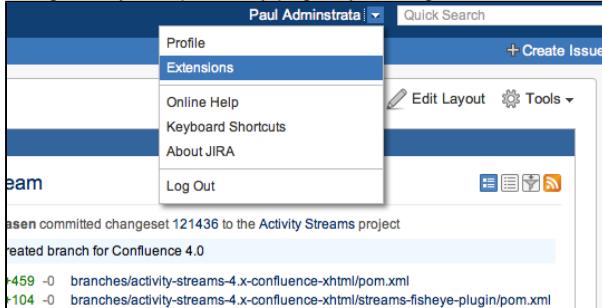
To create your extension, you have two options: fork an existing plugin or start from scratch. For Codegeist all extensions will start from scratch.



To try out your idea, before getting into extension development and packaging, you may find it useful to use Firefox with the Firebug extension. With this, you can try out JavaScript scripts using its multi-line editor, running them on the current page. Once you work out all the kinks, you can follow these instructions to package it as a Speakeasy extension.

When starting from scratch, you can use the extension wizard to create a new conventions-based, or "zip", extension:

1. Install the Speakeeasy plugin on an instance of Confluence or JIRA
2. Navigate to your Speakeeasy page, by clicking "Extensions" in the username dropdown:



1. To start creating your Extension, click on the **+ Install** button, and then click on the "use the wizard" link
2. Fill in the key, name, and description of your new extension
3. Click "submit" and you'll see your new extension in the list
4. Click on "Enable" and refresh the page to see the banner your new extension is displaying

If you click on "Edit", you'll see this plugin have the following files, assuming your extension key is 'myext':

| File | Description | Required |
|--------------------------|---|----------|
| atlassian-extension.json | The manifest for the extension. The only required attributes are "key" and "version". | ✓ |
| screenshot.png | The screenshot to show in the extension list. Referred to in the descriptor. Must be a 175 x 80 pixel gif. | |
| js/myext/main.js | The JavaScript module to execute on the default context of "atl.general". The example puts a banner on the top of every non-admin page with an image. | |
| css/main.css | The CSS file to be displayed whenever the JavaScript is executed. The example hides the second banner that says "Bye". | |
| images/projectavatar.png | An image file that is referenced in the JavaScript | |
| ui/web-items.json | A list of web items. The example is one that puts a "Yahoo" link on the Speakeeasy page | |

Fork an existing extension

If you see an extension you like and want to tweak, or just want to do something similar, you may want to fork an existing extension following these steps:

1. Find an extension that you find useful but want to customise and click "Fork"
2. Type a description of what this fork will do
3. Click "Edit" to modify the extension right in the browser
4. Press "Save" and try it out

Once you get the hang of it, you can create your own extension.

Developing your extension

To develop your extension, you have two options: the web IDE or offline.

Using the Web IDE

Speakeasy has a very simple text editor built into the plugin. With it, you can edit your extension right in the browser, with each save causing the plugin to be rebuilt and upgraded into the product. However, with this editor, you will be unable to delete, add, or rename files.

Developing Offline

If you need to make several changes and don't want to interrupt your users, you can write your extension with just a text editor and a zip tool. These instructions assume working from the terminal, though GUI tools could be used as well.

1. Find your extension on the Speakeeasy page and click "Download"
2. Click the "Download" link to download the artifact
3. Unzip the artifact, whether it is a jar or zip, into a new directory
4. With your text editor, start working on your extension
5. When ready to test, run:

```
zip -r ../my-extension.zip *
```

Make sure you preserve your file extension

6. Visit the Speakeeasy page in your target application and upload your newly created extension zip or jar.

Be aware that any upgrades or live edits to your extension will immediately be seen by all those that enabled your extension.

Using IDEA

If you want to use IDEA and have created a zip extension that doesn't include a "Download as SDK Project" link, you can still use IDEA to modify your extension:

1. Download then unzip the extension to a new directory
2. Create an IDEA project from existing sources
3. Add the "js", "css", "ui", and "images" directories as source directories
4. Now, you should be able to code like normal, using an external script for deployment

Using git

New in 0.10, you can interact with your extension via git as Speakeeasy acts as a git server for each extension. Operations include:

1. `git clone` - Checkout a copy of the extension repository ready for deployment
2. `git pull` - Keep up-to-date with deployed versions as well as bring in changes from forking extensions
3. `git push` - Deploy extension changes from the command-line

During development on a remote server, `git push` can be a useful technique for rapidly deploying locally-developed changes:

```
git commit -a -m "Local changes" && git push
```

Advanced Topics

Using Web Items

Web items are defined in the file `ui/web-items.json` as a list of web item objects. The following properties are allowed on web item objects:

| Name | Description | Required |
|----------|---|----------|
| section | Where the web item will show up | ✓ |
| label | The text to show for the web item | ✓ |
| url | The URL for the link to point to | ✓ |
| cssId | The CSS ID of the link anchor (0.12.2 or later) | |
| cssClass | The CSS class or set of classes for the link anchor | |
| weight | The weight of the web item | |
| tooltip | The text to show for the tool tip, if applicable | |

| | | |
|------|---|--|
| icon | The object describing the icon. Valid properties are: <ul style="list-style-type: none"> • width - The width of the icon • height - The height of the icon • url - The url of the icon | |
|------|---|--|

Referencing Images in CSS



This feature is new in version 0.9

You can reference an image in your CSS by using a special token that will be replaced at runtime: @imagesPath. For example, you could refer to a background image in your CSS like this:

```
#foo {
  background-image: url(@imagesPrefix/myimage.png);
}
```

Using CommonJS Modules

Your first plugin created with the extension wizard contains a JavaScript file that is loaded as a CommonJS module. CommonJS is a set of standards for JavaScript, of which their Module spec is the one used here. A module fundamentally can import other modules and selectively expose properties or functions to other modules. Each module is loaded in its own scope, so you don't need to worry about global variable clashes.

Contextualizing modules

Entry point modules will need to tell Speakeasy on which page they should be included. This is done via the "@context" annotation in the top JavaDoc-style comment on the module. For example:

```
/** 
 * @context dashboard
 */
... module code here ...
```

This example is configured to have its module loaded when on the Confluence dashboard page. These contexts are also known as 'web resource contexts', as they are the same used for scoping web resources such as JavaScript and CSS for normal Atlassian plugins. For more information about contexts, see the following links:

- Contexts in all products
- JIRA contexts
- Confluence contexts

Consuming modules

Most extensions will interact with CommonJS through consuming other modules, usually ones provided by Speakeeasy. The "CommonJS Modules" tab shows those and all other public modules available for consumption.

Modules can be consumed via the "require(moduleId)" syntax. Here is an example of consuming the 'speakeeasy/jquery' module, which exports the 'jQuery' property:

```
var $ = require('speakeeasy/jquery').jQuery;
```

Module ids can be specified as absolute ids or relative to your current module. Therefore, if you had a module 'myext/util' that you wanted to consume from 'myext/main', you could consume it via:

```
var util = require('./util');
```

The right modules will be automatically delivered to your page via the Speakeeasy plugin. Therefore, you don't need to worry whether the module is in your extension or another extension, although the end user will need to have enabled the dependent extension in order to use yours.

Providing modules

Modules are private to your extension by default. If you want to create a 'myext/util', just create a js/myext/util.js file in your extension and consume it from another module as described above. The goal is to make it really easy to decompose your extension into modules, making it easier to maintain and eventually share with others.

If you want to provide your module to other extensions, you can do so by adding the "@public" annotation to your module's JsDoc (the JavaDoc-like comment at the top):

```
/**  
 * This is a public module  
 * @public  
 */  
exports.foo = "Foo";
```

This module will now show up in the "CommonJS Modules" tab and be available for other extensions to import.



It is recommended that any hack you have to do to retrieve data from the application or manipulate the page be extracted into their own modules. This way, the extension is easier to maintain or tweak by forkers as the ugly bits aren't mixed in with the extension logic, and if you decide to share your hack, you already have a shareable module ready to annotate.

Using the Plugin SDK

If your extension is a "jar" extension, so in other words, a normal OSGi plugin that has an `atlassian-plugin.xml`, you can use the [Plugin SDK](#) to develop your extension.

1. Click on the "Download" link by your extension and click "Download as SDK Project"
2. Unzip the SDK Project into a new directory
3. In a terminal window, run:

```
mvn confluence:run
```

Note, replace "confluence" with the desired target product.

4. In another terminal window, run:

```
mvn confluence:cli
```

Again, replace "confluence" with the desired target product.

5. Navigate to <http://localhost:1990/confluence> in your browser. See the [Plugin SDK](#) docs for other product URLs.
6. Login with a username of "admin" and password of "admin"
7. Move your cursor to your username on the top banner and click "Speakeasy". You should see your extension.
8. In your text editor, open files `src/main/resources` and start working on your extension. Most changes should be visible immediately, but others will require you to switch to your terminal window running the cli and enter 'pi'

If your extension is a "zip" extension, you can still use the Plugin SDK to develop your extension, albeit in a more limited fashion:

1. Click on the "Download" link by your extension and click "Download"
2. Unzip the artifact into a new directory
3. Run `atlas-run-standalone` to start the Atlassian product. In this example, we start Confluence 3.5.3:

```
atlas-run-standalone --product confluence --version 3.5.2 --jvmargs  
"-Dplugin.resource.directories=. -Xmx512m -XX:MaxPermSize=128m" --plugins  
com.atlassian.labs:speakeasy-plugin:0.10.2
```

4. Upload or push your extension to your local Confluence running at <http://localhost:1990/confluence> and start developing. JS and CSS changes that don't require a plugin reinstallation can be seen immediately via a reload (thanks to that 'plugin.resource.directories' system prop)

Under the covers

A Speakeasy extension, at a technical level, is an OSGi plugin that contains only Speakeeasy-provided dynamic module descriptors. These include:

- `scoped-web-resources`
- `scoped-web-item`
- `scoped-modules`

The custom module types are necessary to ensure the plugin can be enabled on a per-user basis. Any other module descriptors, or even files considered not appropriate for such a plugin, will result in extension installation rejection. Inappropriate files include JAR files, Java class files, and Spring XML descriptor files. Since Speakeeasy extensions are real plugins, they can be viewed, managed, and uninstalled from the Plugin Management UI in each product.



Keep in mind the idea is the application should work 100% correctly if your extension is not enabled. This means you shouldn't do things like create psuedo Confluence macros that show useful information for your extension users but blank screens or gibberish for all others.

Speakeasy Extension Examples

| Plugin Name | Plugin Description | Source/Binary |
|--|---|---|
| Confluence Panada Plugin | Resolves an annoyance of the SF Integration Team. Correctly replaces any 'panada' user macro with 'bwoskow'. | https://bitbucket.org/bwoskow/confluence-panada-plugin |
| Google Link Test Extension | Test extension that puts a link to "Google" in the admin menu in every product. Includes a tree of files, CSS, JS and images (none of which are used - purely there to test the Speakeeasy IDE) | src / zip jar |
| Image Attachments Extension | Very simple example that adds thumbnails to the Confluence attachments page for image attachments. Great example of a super simple extension. 10 lines of Javascript! Example screenshot . | src / jar |
| Remove Sidebar Extension | Another simple example plugin that is a single CSS file, showing how you can use Speakeeasy to augment the UI using just CSS. Removes Confluence's personal info sidebar from personal spaces. | src / jar |
| JIRA Attachment Improvements Extension | Improves the display of attachments within the JIRA change history (something that has annoyed me for years!). Links attachments to download in change history, shows thumbnails of image attachments and shows indicators if an attachment has been removed in the future. | src / jar |
| JIRA Admin Prototype Extension | A prototype of what the JIRA admin would look like if we made it modal, removed the sidebar, re-used the top menus, made the searchbox a quick nav for admin screens. See Screenshot | src / jar |
| Confluence Thumbnail Galleries | Fixes a pet peeve issue...if you have more than one thumbnail image on a page, you can now view them in a gallery-style slideshow | zip |
| JIRA Splat Plugin | Inspired by an urgent customer request on Twitter . | src / jar |
| JIRA Inline Issue Preview Plugin | Allows inline previews of the issue's description in the issue navigator. | src |
| Confluence BitBucket Repo Hover | Provides more information about BitBucket repositories mentioned on a Confluence page with a BitBucket repository hover. | src |
| Tronfluence | Reskins Confluence with a blue-neon-on-black theme. | src+binary |
| Confluence Active User Plugin | Indicates users without an active account by striking through their username. | src / jar |
| Confluence flamewar detector | Tells you if a page is likely to have a flamewar on it. | src |

Speakeeasy Install Guide

Installation

In order to start building and testing Speakeeasy Extensions, you'll first need to install Speakeeasy on a JIRA or Confluence instance. You can do this via the plugin manager built into JIRA and Confluence. Search for "Speakeeasy Plugin" and click "Install".

Manual Standalone Installation

Run through the following steps to get the Speakeeasy framework installed on a standalone instance:

1. Download and install the most recent Plugin SDK (v3.3).
2. Start JIRA or Confluence using [atlas-run-standalone](#):

```
atlas-run-standalone --product jira
```

or

```
atlas-run-standalone --product confluence
```

and login with username "admin", password "admin".

3. Navigate to **Administration > Plugins > Install** and click the **Upload Plugin** link.
4. Enter [https://maven.atlassian.com/content/repositories/atlassian-public/com/atlassian/labs/speakeeasy-plugin/1.0/speakeeasy-plugin-1.0.jar](https://maven.atlassian.com/content/repositories/atlassian-public/com/atlassian/labs/speakeasy-plugin/1.0/speakeeasy-plugin-1.0.jar) in the "From this URL" field (or download it and enter the local path in the "From my computer" field), and click "Upload."

Configuration

The default Speakeasy configuration restricts who can develop and enable Extensions. You'll need to change these to suit your instance.

1. Navigate to **Administration Dropdown (top right) > Extensions** and click "Edit". If you do not see **Extensions**, refresh the page.
2. Because you're developing on a standalone instance with no sensitive data, untick the box titled 'Restrict to non-Administrators' and click "Save."

Extension Security For Production Servers

Speakeasy Extensions are still experimental, but if you wish to enable Speakeasy on a JIRA or Confluence server that contains non-public information, here are some notes on security implications:

Enabling an extension **allows the Extension author to execute JavaScript as you**. Because of this, by default, Administrators cannot run Extensions. You can enable Administrator access to Extensions, but we recommend only doing so if you *really* trust all your users.

A more sensible configuration might be to allow developers to develop Extensions and internal users to view and enable them. **If your instance allows public signup, it's a bit soon to turn on Speakeasy.**

Next Steps...

It's time to start building a Speakeeasy extension!

Speakeeasy Overview

What is Speakeeasy?

Speakeeasy is a new extension mechanism we're experimenting with at Atlassian. The basic design goals are:

- extensions are social
- extensions are augmentative
- extensions are per user
- extensions are written using simple web technologies

If you're familiar with our normal plugin system, you can think of an extension as a subset of a plugin that has slightly less capabilities but is much faster to write, much simpler to write and quicker to deploy. Note - plugins are still very useful for heavy modifications and full applications, we've learned they're just too heavyweight for simple customisations.

Further to these goals, let me throw a few points out there to clarify.

For admins:

- End users can install, enable and disable their own extensions. (The set of users who can install, and the set of users who can enable extensions will eventually be customisable - hopefully next week - right now, it's everyone - so be wary)
- The augmentative part we're still working on, the broad idea is that if you turn them off, the base product is unchanged. This reduces the "upgrade pain" associated with plugin compatibility somewhat (you can wait for an extension to be made compatible with a new release and still upgrade)
- Plugins are enabled per server, for everyone. This causes various issues with deployability as a rogue or broken plugin can bring down a server, or at least cause it harm. Extensions however are per user. This means a single user can only hurt themselves (they

might get a funny looking interface) and not any other users if they're experimenting or trying a new extension. This again increases the safety for your server. (Note: there is a "disable all" URL that always works - if someone gets themselves into trouble, they can always use that to get out of it)

- Learn more at the [Speakeasy Admin Guide](#)

For developers:

- Simple web technologies? Yup that's right - extensions require no Java, no Maven, no JDK, no JAR. You can create 'em in any text editor. Don't fear - they can still do powerful things! They're written purely in Javascript, CSS, HTML, JSON. They can access REST and XML-RPC resources. They're packaged in a standard ZIP file. (Note - you *can* use AMPS or the Atlassian SDK to write an extension if you're a power developer, but you don't have to - and we encourage you not to - try our new tools, you'll never go back)
- Convention over configuration - the traditional `atlassian-plugin.xml` configuration file doesn't work that well for small plugins, so we have a new convention based system that makes for much simpler extensions. Just put your files in the right directories, a sprinkling of config (often 4 lines) and zip it up.
- Extensions are designed to be shared, forked, copied, modified - and even edited online. Speakeasy already comes with a mini web-based IDE built in, so your power users can fork an extension that someone else has written, tweak it and see their tweaks live. It's quite amazing.

There are still one or two major technical pillars we're building - but we're ready to get some feedback from our user community on what we have so far.

What does an extension look like?

Take a look at the [Speakeeasy Extension Examples](#) to or the [Speakeeasy Extension Development Guide](#).

How do I play with Speakeeasy?

Right now, you must [install the Speakeeasy plugin](#) into your server (JIRA 4.3+ or Confluence 3.5+). Any user can then go to "[Their name] > Speakeeasy" and upload an extension. Other users can then enable that extension, or fork it etc.

Once you have Speakeeasy installed, take a look at the [Speakeeasy Extension Examples](#) to or the [Speakeeasy Extension Development Guide](#) to get started.

Where do I get extensions?

Extensions aren't available on [plugins.atlassian.com](#) yet - but we've created quite a few internally (somewhere north of 40 I'd estimate).

Take a look at the [Speakeeasy Extension Examples](#) to see some extensions we've published.

How do I write my own extensions?

The [Speakeeasy Extension Development Guide](#) has everything you need to build your own extension.

What's the catch?

We *do not* recommend you run Speakeeasy publicly. We're still doing a thorough security review, and there's a few more features we'd like to polish before having a "1.0 final" public release.

That said, if you trust your userbase, or have a small set of knowledgeable users, it's pretty solid already. And you can always turn it off by disabling the Speakeeasy plugin itself. As a reference, we're running it at Atlassian on all our internal instances for the moment - and have been for a while. We have upwards of 30 extensions on some instances, with hundreds of user enablements.

Resources

- [Maven repository](#)
- [Github project page](#)

Speakeeasy Plugin Development

If you so desire to improve the Speakeeasy plugin itself, here are some steps to do so. Note that you don't need to hack on the Speakeeasy plugin to enter the contest.

Take the following steps to install the Speakeeasy plugin:

1. Clone (or fork) the [github speakeeasy-plugin](#) project via:

```
git clone https://github.com/mrdon/speakeeasy-plugin.git
```

2. Start the plugin in the desired product:

- Refapp:

```
mvn refapp:debug
```

- Confluence:

```
mvn refapp:debug -Dproduct=confluence
```

- JIRA:

```
mvn refapp:debug -Dproduct=jira
```

3. Open pom.xml in IDEA

4. Start the cli for rapid deployments:

- Refapp:

```
mvn refapp:cli
```

- Confluence:

```
mvn refapp:cli -Dproduct=confluence
```

- JIRA:

```
mvn refapp:cli -Dproduct=jira
```

5. Visit the user profile page of Speakeasy:

- Refapp:

```
firefox http://localhost:5990/refapp/plugins/servlet/speakeasy/user
```

- Confluence:

```
firefox http://localhost:1990/confluence/plugins/servlet/speakeasy/user
```

- JIRA:

```
firefox http://localhost:2990/jira/plugins/servlet/speakeasy/user
```

To try out extensions, you can create a new one or work on the testing plugin found in src/test/resources. It is deployed via the tpi cli command (stands for 'test plugin install')

Developer Relations State of the Union 2011 - Resources

Universal Plugin Manager

- [Plugin](#)
- [Docs](#)

Active Objects

- [Docs & Getting Started](#)
- [Dev Project](#)

Speakeasy

- [Docs](#)
- [Dev Project](#)

Application Links

- [Docs](#)

Atlassian Plugin SDK

- [Download](#)
- [Docs](#)
- [Dev Project](#)

Developer Email List

- <https://my.atlassian.com/email/index>

Plugin Libraries

Many Atlassian customers have contributed the plugins back to the Atlassian community as open-source plugins. And there are a growing number of plugins provided by commercial vendors. These can all be used to extend and enhance your Atlassian tools. If you're developing a plugin yourself, you can also look at these plugins as examples to help you with your own development.

Plugin developers list their offerings on our add-ons site: <http://plugins.atlassian.com>. You can [visit and browse](#), or search using the form below:

Search the Atlassian Plugin Exchange



[All Products](#) [JIRA](#) [Confluence](#) [Fis](#)
[hEye](#) [Bamboo](#) [Crowd](#) [Crucible](#)

 Search

Plugin (Glossary Entry)

A **plugin** is a bundle of code, resources and configuration files that can be dropped into an Atlassian product to add new functionality or change the behaviour of existing features. Every plugin is made up of one or more **plugin modules**. A single plugin may do many things, while a plugin module represents a single function of the plugin.

RELATED TOPICS

[Glossary - Terms and Definitions]
[Welcome to the Atlassian Developer Network](#)

Admin-Config Wishes

- [Development Kit for Eclipse+Ant](#)
- [NTLM Authentication](#)
- [Plugin Repository Submission Form](#)

Development Kit for Eclipse+Ant

| | |
|----------|---------------------------------|
| Name | Development Kit for Eclipse+Ant |
| Priority | High |

| | |
|--------------------|--|
| Complexity | Medium |
| Author(s) | |
| Description | Create a development Kit for Eclipse+Ant because Maven sucks and no one uses IDEA. |

Description/features

The majority of developers don't use Maven, and I actively avoid it - because it sucks. I for one (and my company) are not going to start using Maven. Ever.

See http://fishbowl.pastiche.org/2007/04/03/an_open_letter_to_maven_2 for comments from Charles Millar.

Also eclipse has by far the biggest share of the IDE market - 67% at the last figure.

To lower the barrier for plugin development, provide a development kit suited to developing with eclipse and ant **without** Maven.

I guess this would be a template project with eclipse .project and .classpath files set up correctly, and a build.xml set up to build and deploy a plugin jar without too much hassle, and all config externalized out to a properties file.

If you really want to have the whole dependencies stuff, then ivy can do that, but really its not essential for most developers.

I have a million ideas for plugins, and writing the code would be a 10 minutes job (reality: 10 hour job), but I just can't be bothered setting up the project and build files.

Usage

This small development kit might contain the needed functionality.

Technical Implementation

Other Resources

EclipseAntDevelopmentKit

Development kit for developing Confluence plugins without Maven

This is a small development kit helping in developing Confluence plugins using Ant and (optional) Eclipse. It contains everything needed to generate a Eclipse project, to compile the plugin and install it to a running Confluence instance (using the Confluence Remote API). It has only been tested against Confluence 3.0 on a Windows development machine so far. It might work for all versions which have the installPlugin method in their remote API. The location of the Confluence libraries is based on the default directory structure of Confluence standalone version.

The development of this kit was driven by the author's needs and might not contain all the functionality needed by others. Feel free to comment or contact the author for additional functionalities.

It has also been tested with existing sources generated with Maven.

Prerequisites

1. Download the development kit
2. Unzip to a directory of your choice
3. Point the environment variable "CONFLUENCE_ANT_HOME" to this directory
4. Add "CONFLUENCE_ANT_HOME/bin" to the Path
5. Point "CONFLUENCE_HOME" to the Confluence standalone installation
6. Create a user library named "Confluence" in Eclipse and add the JARs from CONFLUENCE_HOME/confluence/WEB-INF/lib and CONFLUENCE_HOME/lib
7. Make sure the Remote API is activated in the Confluence installation in which you wish to install the plugin

Usage

Generate a Eclipse project

- Open a terminal / command prompt
- Go to your workspace directory
- type 'create-eclipse-project'
- When asked, enter the values for the Eclipse project name, the Plugin name (as shown e.g. in the admin console) and the Plugin ID (used internally, should be unique; corresponds to Maven artifactId).
- Open Eclipse and use 'File' -> 'Import' -> 'Existing projects into workspace' and import the generated directory.
- Edit/Enter the missing properties in 'build.properties' and 'ant-confluence.properties'

Develop your plugin

- Develop your plugin as you would using Maven

- You can leave the variables in atlassian-plugin.xml, they will be replaced with the correct values (taken from build.properties) during the build process.
- Put all external libraries needed by the plugin in lib. They will be added to the Jar. Libraries added to lib-compile will only be used during the Ant build (useful e.g. for junit.jar).
- If all you want to do is modify an existing plugin, put the existing plugin Jar into the jar directory, add e.g. a translation resource and use the Ant build in the same manner as you would when developing a new plugin.

Deploy your plugin

- The default Ant target is 'compile'. This will build the plugin Jar in the newly generated dist directory.
- You can run unit tests (classes ending with 'Test' in the src/test/java directory) using the 'test' target.
- Install the plugin using the 'install' target.

Leave some feedback

- It would be great if you could tell me if the development kit worked for you, if the instructions were clear enough, etc.

NTLM Authentication

| | |
|-------------|---------------------------------|
| Name | NTLM Authentication |
| Author(s) | |
| Priority | High |
| Description | Write an authenticator for NTLM |

This could use Seraph or maybe Crowd? See <http://jira.atlassian.com/browse/CONF-2326>

Plugin Repository Submission Form

| | |
|-------------|--|
| Name | Plugin Repo Submission Form |
| Author(s) | |
| Priority | Medium |
| Description | A handy web form to auto-generate and commit Plugin Repo metadata based on your input. |

Avail - JIRA

- Agile Charts and Graphs
- JIRA MS Project Integration

Agile Charts and Graphs

| | |
|-------------|---|
| Name | Agile Charts and Graphs |
| Author(s) | |
| Priority | High |
| Description | Various charts and graphs useful for agile teams, (e.g. Burndown, Velocity) |

Description

Ideally, we could combine the other two agile charting plugins and add new capabilities in order to create an official "agile reporting package" for JIRA.

Resources

- Using JIRA for Agile Development
- Laughing Panda JIRA Agile Report Plugins
- Agile Velocity Tracking plugin

JIRA MS Project Integration

| | |
|-------------|-------------------------------------|
| Name | JIRA/MS Project Integration |
| Author(s) | |
| Priority | High |
| Description | 2-way sync between Project and JIRA |

I'd love to see an integration with MS Project able to do the following:

there is a task A in MS-Project with Bob assigned as a ressource. 1-n according issues are created in jira manually and assigned to developer Bob. Bob has worked 4 hours on that issue and updates the according effort in jira. The project manager opens the project plan and retrieves the newest jira data. the effort spent on the jira issue is updated automatically in the according MS-Project task(s) and the progress bar is visible.

if Bob enters a new estimation for remaining work in jira, the information should be updated in MS-Project as well.

AFAIK there was a solution, but it seems to be dead by now: <http://www.mondo.com.br>

Exists

- Completed Specs

Completed Specs

Labelled 'complete'

Confluence

Unable to render macro: No such page "plugin wishlist"

Jira

Unable to render macro: No such page "plugin wishlist"

Bamboo

Unable to render macro: No such page "plugin wishlist"

Crowd

Unable to render macro: No such page "plugin wishlist"

Children of Completed Specs

| Page | Description | Status | Author | Plugin Implementation |
|--------------------------------------|--|----------|--------|--------------------------------|
| BloggerRPC Plugin | Implement the Blogger API (or some other one) so that people can use blog tools (like marsedit or ecto) to blog on Confluence. | Released | | Confluence Blogging RPC Plugin |
| Comments Tab Theme | Move the comments to a separate tab, ala wikipedia. At least two customers interested. | Released | | |
| Confluence Contributors Plugin | List all the authors who have contributed to a page. | Released | | Confluence Contributors Plugin |
| Content Searcher for OpenOffice docs | Search inside OpenOffice and OpenDocument docs | | | |
| Crowd PHP Integration Libraries | Allow PHP apps to use Crowd as their user store and SSO | | | |

| | | | | |
|--|--|------------------------------|--|------------------------------|
| Crowd Ruby Integration Libraries | Allow Ruby and Rails apps to use Crowd for their user store and SSO. | | | |
| Dynamic Announcement Banner | Put noticed into every page of Confluence. | Released | | |
| Google Maps Plugin | A macro to show you a Google map of a particular address. | Released | | |
| Import Visio Diagrams | Import Visio files and display diagrams on the page. | | | |
| IntelliJ Idea Plugin for Jira and Confluence | Ability to refer to Jira issues and Confluence from within IntelliJ IDEA | | | |
| Invite user to space plugin | Automatically send email inviting people to a space. Clicking link in email would create appropriate accounts and permissions | Released | | |
| JIRA Calendar Plugin | Buid calendar functionality into JIRA as a portlet and as an iCal export. | Released | | |
| JIRA Labels | Another awesome JIRA feature would be a labels custom field to allow you to create, index and search issues into arbitrary groups. See JRA-8578 . | Released | | |
| Mail This Page | Mail to User Groups or Individual from a wiki page. | | | Emailing a Page |
| Mark For Review | Allow readers of the wiki to mark content for 'review' by managers. This content may be marked because it's inappropriate, worthy of promotion or simply incomplete. | | | |
| Math Equations Plugin | Allow users to use LaTeX to render pretty math equations. | Released | | |
| NAnt builder | Create a Builder Plugin for NAnt | | | BAMBOO:Builder Plugin Module |
| New Code Macro | Syntax Highlighting for ECMA 262 (eg: JavaScript / ActionScript) based languages CONF-893 | Released | | |
| Popups or Tooltips | Simple javascript popup macro. | | | |
| SVN Commit Acceptance | Allow JIRA to accept or reject commits on certain criteria. | Released | | |
| SVN Project Tab | Add a project tab to the SVN plugin. | Released | | |
| Search Keyword Macro | hint/add keywords to the search index for a page to manipulate search results. | | | |
| Social Bookmarking | A del.icio.us-style social bookmarking feature for both internal and external URLs | Released | | |
| Translation Plugin | Plugin to facilitate distributed, crowd-sourced translations. | Released | | |
| Usage Stats Plugin | Show graphs of site usage | Awaiting developer resources | | |
| Vote-for Custom Field | A custom field for the column view that allows you to vote for items ajax-style. | Released | | |
| WebDav Plugin | Access Confluence over WebDAV. | Released | | |
| Zip Search Extractor | a new Search Extractor for Confluence which is able to index the individual files within a Zip file. | Released | | |

Mail This Page

| | |
|-------------|---|
| Name | Mail This Page |
| Author(s) | |
| Priority | High |
| Description | Mail to User Groups or Individual from a wiki page. |



Complete! Agnes wrote a [Mail Page plugin](#) that ships with Confluence 2.4

Provide a way to mail to User Groups or Individual from a wiki page.

Any user can decide to watch pages or subscribe to RSS feeds from the wiki.

However, there is no way to inform other users that something could be important for them: as the publisher of such important information, you have to open your email system and manually create a mail to all the individuals you want to inform and refer to the page URL.

It would be nice to be able to have a quick link on top of a page that would prepare a mail with the link to the page, and that would allow a publisher to pick User Groups and/or individual users from the wiki user list. The user preparing this alert would be allowed to change the mail content, and to add some relevant information.

Once confirmed, the system would send an email to all the users targeted (as individuals or as part of User Groups); the system would be able to send only one email to a user being part of several User Groups targeted by the alert.

Related

[Emailing a Page](#)

and see also: <http://forums.atlassian.com/thread.jspa?messageID=257218161&>, and Agnes Ro.

JIRA Labels

| Name | JIRA Labels |
|-------------|---|
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Another awesome JIRA feature would be a labels custom field to allow you to create, index and search issues into arbitrary groups. See JRA-8578 . |

Take the [JIRA Labels Plugin](#) and complete it. You'll want to look at the way Labels work in Confluence and mimic that behaviour as much as possible. You should look at the Confluence labels code and lift what you can. Also, check out the Voters and Watchers plugin for an example of an Ajax Issue Navigator field, and look at <http://confluence.atlassian.com/display/DISC/Running+DWR+inside+a+Plugin> to see how to embed DWR inside your plugin.

Requirements:

- Fix [LBL-1](#): Searches for multiple tags only keep the first tag in the search box
- Implement [Ajax interface](#) in both Issue View and Project view of the [custom field](#).
- Implement Tag Cloud/Heatmap view (see Confluence [Screenshot](#)) as a [Project Tab](#) and a [Dashboard Portlet](#).

Screenshots

JIRA Your Company JIRA

[HOME](#) [BROWSE PROJECT](#) [FIND ISSUES](#) [CREATE NEW ISSUE](#) [ADMINISTRATION](#)

Issue Details ([XML](#) | [Word](#))

| | |
|------------------|----------------------------|
| Key: | TST-1 |
| Type: | Bug |
| Status: | Open |
| Priority: | Major |
| Assignee: | Jonathan Nolen |
| Reporter: | Jonathan Nolen |
| Votes: | 2 (View) |
| Watchers: | 0 (View) |

Available Workflow Actions

- [Start Progress](#)
- [Resolve Issue](#)
- [Close Issue](#)

Operations

- [Assign](#) this issue
- [Attach file](#) to this issue
- [Attach screenshot](#) to this issue
- [Clone](#) this issue
- [Comment](#) on this issue
- [Delete](#) this issue
- [Edit](#) this issue
- [Move](#) this issue
- [Voting:](#)
You cannot vote for an issue you have reported.
- [Watching:](#)
You are not watching this issue.
[Watch it](#) to be notified of changes

Test

test bug 1

Created: 04/Aug/06 03:24 PM Updated: Today 01:41 PM

| | |
|---------------------------|------|
| Component/s: | None |
| Affects Version/s: | None |
| Fix Version/s: | None |

Labels: cat ❌ dog ❌ bird ❌

Add Labels

Enter labels to add to this issue: cow

Suggested labels: maven, ant-optional, int optional, junit

[Add](#) [Done](#)

Tip: Looking for a label? Just start typing.

[All](#) [Comments](#) [Change History](#)

There are no comments yet on this issue.

| T | Key | Summary | Assignee | Reporter | Pr | Status | Res | Labels | Voters | Vote | |
|-------------------------------------|-------|--------------------|--|-----------------|----|--------|------------|---------------------|----------------------|------|--|
| <input type="checkbox"/> | TST-2 | Test new feature 1 | Jonathan Nolen | Jonathan Nolen2 | ↑ | Open | UNRESOLVED | Bob | Vote | | |
| <input checked="" type="checkbox"/> | TST-1 | test bug 1 | Add Labels Labels: cat ❌ dog ❌ bird ❌ Enter labels to add to this issue: cow Suggested labels: maven, ant-optional, introduction, ant optional, junit Add Done <i>Tip: Looking for a label? Just start typing.</i> | | | | | | | | |

IntelliJ Idea Plugin for Jira and Confluence

| | |
|-------------------|---|
| Name | IntelliJ Idea Plugin for Jira and Confluence |
| Priority | High |
| Complexity | medium |
| Author(s) | |
| Support | Please submit an IDE Plugin issue under https://studio.atlassian.com |

| | |
|--------------------|--|
| Description | Ability to refer to Jira issues and Confluence from within IntelliJ IDEA |
|--------------------|--|

Description/features

Ability to refer to Jira issues and Confluence from IntelliJ IDEA.

Jira

- Whenever there is a Jira issue in the sources comment, it should be shown as a link, when clicked, then the browser should be opened
- If you want to get fancy, when hovering over the link, it'd show the issue title/description

Confluence

- Each project, module, class could have a page. When right-clicking over either, it should give the option 'see confluence'
- Parsing for comments including confluence style links.i.e.

```
/**  
 * See [DoSomething]  
 */  
public void doSomething() {
```

Usage

Configuration

- Confluence and Jira server information to be configured for each project
- Alternatively, Jira server information could be embedded in the actual source files (i.e. for open-source public access repos)
- alternatively, Jira server info could be configured globally per package name

Technical Implementation

Easy!

1. Go to http://www.jetbrains.com/idea/plugins/plugin_developers.html
2. Get Jira + Confluence remote APIs
3. Develop plugin
4. that's it!



Other Resources

- There is something on Jira DISC:IntelliJ IDEA Plugin but it seems obsolete
- Leo Przybylski's comments on [Using Confluence for professional documentation](#) got me thinking

Dynamic Announcement Banner

| | |
|-------------|--|
| Name | Dynamic Announcement Banner |
| Priority | Medium |
| Status | Released |
| Author(s) | [|
| Description | Put noticed into every page of Confluence. |

Similar to the new JIRA feature appearing in [JIRA 3.4 and 3.4.1 Release Notes](#)(thanks Nick!). It would be excellent if this could be done also in Confluence. So far the only plugin I have been able to muster requires theme injection - however a component, a helper and an action would do. The ability to add items to the Admin menu may also be needed ([CONF-3724](#)).

Invite user to space plugin

| | |
|----------|-----------------------------|
| Name | Invite user to space plugin |
| Priority | Medium |

| | |
|-------------|---|
| Status | Released |
| Author(s) | |
| Description | Automatically send email inviting people to a space. Clicking link in email would create appropriate accounts and permissions |

Popups or Tooltips

| | |
|-------------|--------------------------------|
| Name | Popups or Tooltips |
| Author(s) | |
| Priority | Low |
| Description | Simple javascript popup macro. |

Similar to the footnote plugin (I think), this would implement [overlib](#) in pages. Overlib is a simple, mature JavaScript library created to enhance websites with small popup information boxes (like tooltips) to help visitors around your website. It looks nice and can be quite useful giving hints and context to users.

Or, it may be easier to use scripaculous, since we already have that integrated in Conf.

SVN Project Tab

| | |
|-------------|--------------------------------------|
| Name | SVN Project Tab |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Add a project tab to the SVN plugin. |

This piece of the puzzle will involve building some new views for the SVN information. You'll be adding this feature to the existing [JIRA Subversion Plugin](#) This plugin will require careful attention to memory usage and performance.

Doing the commits table reload via ajax might be nice, but isn't a requirement.

Requirements

- A new Project Tab that shows the most recent 20 commits against that project.
- A dropdown on that same Project Tab that will change the view to the more recent 20 commits against a particular version
- A small bar graph of commits over time (probably the last month).

Unable to render embedded object: File (project-tab.png) not found.

Requirements

These are the requirements for the project tab, taken from the [parent of this page].

A new Project Tab that shows the most recent 20 commits against that project.

New document field

The Lucene documents that are used to index revisions are extended with a new Keyword field: *project*. It holds the project keys for all projects affected by a commit (the project keys are extracted from the issue keys). The new field is then used to look up commits against the current project efficiently in all repositories.

When upgrading to the new version of the plugin, administrators can either delete the old indexes so that new ones are created with the project field, or they can do nothing. In that case, none of the old revisions will ever be shown in the project tab (since the old revision entries do not have the project keyword), but all revisions indexed after the upgrade will have the project key. Eventually, there will be 20 revisions for the project to show anyway.

Note that if several repositories are configured (and used) for a project, the tab will show the 20 most recent commits in each of the repositories.

GUI

The format of the commit entries should be identical to that on the SVN Issue Tab.

Maybe we should have a separate .vm file for the commit table, which we could include from both the project and the issue tab templates?

A dropdown on that same Project Tab that will change the view to the more recent 20 commits against a particular version

The project versions are read from the JIRA VersionManager.

When a particular version is selected, the 20 latest commits for issues logged against that version are shown.

A small bar graph of commits over time (probably the last month).

This requirement will be implemented last, when the other parts are working well.

The graph will be implemented using JFreeChart, which is included in JIRA.

About the searches

I have added two search methods to RevisionIndexer: one that gets commits to an entire project and one that fetches commits for any of a collection of versions. The project search just gets all commits with the project's key (I set up the project keys in the Lucene documents during indexing, so the search should be fast).

The version search is more tricky. For each version (and there may be many of those, e.g., if the user selected "Released Versions") I look up all issues affected or fixed by the version and put them in a HashSet to get rid of duplicates. Then, I construct a BooleanQuery with a clause for each issue (taking care to increase the maximum number of clauses if needed).

I wanted to let Lucene handle as much as possible of the work, so I implemented the queries as sorted on date, descending. Then I just look up the revision entries for the first 20 hits. I suppose this could be optimized slightly by limiting Lucene to return just 20 hits instead of 100, which I believe is the default. How to do that was not immediately obvious, however, and the gain is probably fairly small.

Translation Plugin

| | |
|-------------|---|
| Name | Translation Plugin |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Plugin to facilitate distributed, crowd-sourced translations. |

A Confluence plugin that will offer a better interface for translating properties files in a distributed manner.

Documentation

This is the documentation for the initial release and is designed to get you up and running.

Required Modifications

In order for web-item tabs to work properly, the `confluence/decorators/page.vmd` needs a little help. Find the line that reads:

```
## ATTACHMENTS
#elseif ($mode == "view-attachments" || $mode == "move-attachments")
    $body
#end
```

and replace it with:

```
## ATTACHMENTS
#elseif ($mode == "view-attachments" || $mode == "move-attachments")
    $body

## CATCH ALL
#else
    $body
#endif
```

For more information you can see: [CONF-6853](#)

Also, in order for the i18n keys to resolve properly (in the Confluence breadcrumbs and the web-item) you will need to be running at least Confluence 2.3-DR1.

Usage

Once the plugin has been installed, place a {translate-apps} or a {translate-languages:Application Name} on a page. This will render the list of applications, or the languages for the latest version of an application respectively. A "Translation" tab will also appear.

There are a few things worth knowing:

- Clicking on the tab will *always* take you back to the applications listing of the translation plugin.
- Data is stored centrally inside Bandana and is not relative to page nor space.
- The source locale is set during the creation of an application, and when a version is created the source properties are read in. These settings are immutable.
- Be careful when adding - once an application / version / translation is created, it cannot be deleted.

Common navigation is provided through a dedicated breadcrumb trail that appears whenever in the Translation tab. Links from the macros link directly to their respective areas inside the tab's realm. Where the UI doesn't work intuitively - it will prompt you when it suspects you might be doing something silly, for example: editing a translation, but not saving the contents before clicking Finish. You should find it generally logical as you step from one area to another, with links to the typical areas you may want to cross to from each screen.

The majority of the application uses traditional web workflow techniques, however the edit and complete translation screens both use AJAX techniques for inline reading and saving.

Application List

This is the initial screen upon clicking the tab, and is also displayed inline in a page via the {translate-apps} macro. A table will be displayed with the following information:

- Application name
- Application's source locale
- Number of versions in this application
- If there is at least 1 version, then the following is also shown for the latest version:
 - Latest version
 - Number of languages
 - Progress bar representing the number of completed languages in this version
 - Progress bar representing the average amount of completion over all the languages in this version

There is an add application link at the bottom, and each application links either to the version list or to add version, depending on whether the application has any versions already.

The progress bars are done using CSS and there is a hack in there to get it to layer properly on IE. When you mouse over the bar you should get more detailed information in a tooltip.

Add Application

You will need the following information:

- A unique application name (it should not contain a version number)
- The locale which represents the source language properties for this application's version

This information is immutable and cannot be deleted either. When you have finished you will be returned to the application list where you can create your first version.

Add Version

You will need the following information:

- A unique version number (it's string based and isn't heavily validated - be warned that versions are sorted using String.compare()!)
- The source language properties in the locale specified when the application was added

This information is immutable and cannot be deleted either. When you have finished you will be returned to the version list.

Version List

A table will be displayed with the following information:

- Application Name + Version
- Language Count
- Completed Languages
- Average Completion

The latter 3 fields work as prescribed in the application list, but for the relevant version in each case. There is an add version link at the bottom, as well as a link to download the source language as a properties file. Each version has a link to access the languages for the respective version - this takes you to the languages list.

Languages List

A table will be displayed with the following information:

- Locale
- Completion - represented by a progress bar, working as normal

There is a link for add language, all languages have an edit link, incomplete languages have a complete link and languages with at least one key translated have a download link. The download link retrieves the language's translation as a properties file, much the same way as the version's source language properties can also be downloaded.

If there is a language for another version of this application, which doesn't have a translation for this version - a table will appear below the languages list explaining this. The language will list the locale and the latest version from which translation information is available. Next to each language will be an upgrade link which will show the upgrade language link.

Add Language

You will need the following information:

- Locale to translate to
- The translated language properties (optional - useful for migrating existing translation into the plugin)

The locale is immutable, however the properties can be updated through the edit / complete translation links. The language cannot be deleted once created.

Complete Language

This gives you a nice quick way to get started translating the application version.

You will be presented with a screen that has overview information at the top, a nice big progress bar (dynamically updated) and a translation section will be shown at the bottom. The translation section contains a source key, source value and a space for you to enter the translation for the key. Once you have entered the translation you can click the next button to continue to the next untranslated key - if you prefer you can skip any keys you don't want to translate by using the link provided. You can, of course, finish at any time with the appropriate button - you will automatically finish once the language is fully translated. When finished you will be returned to the language list.

Note: providing an empty translation value and clicking next is the same as clicking skip

Note: this screen uses AJAX to load and save its data - UI controls will be disabled when AJAX communications are in progress

Edit Language

This gives you a large table where you can edit the entire translation on a single screen.

You will be presented with a screen that has overview information at the top, a nice big progress bar (dynamically updated), a search box (for filtering the table below) and a translation table. Once the data has loaded you can click on the translated values to have them turn into an textbox where you can change them (emptying them removes the translation for that key). If you want to save your changes, push enter - if you want to revert them, push escape. The progress bar will update accordingly automatically.

Searching can only be performed when no keys are in edit mode - you will also visually appear loose and changes you've made. If you have pushed enter, the data will have been sent to the server correctly.

Note: changes made by other people updating the translation will not be reflected until you refresh as the data is only loaded from the server once.

Note: this screen uses AJAX to load and save its data - UI controls will be disabled when AJAX communications are in progress

This user interface may wish (in future) to have more work spent on it to increase its responsiveness with large amounts of data and to have it update dynamically / regularly to reflect other people's changes.

Upgrade Language

This takes a translation for a different version and tries to apply the translation to this version where there is key overlap.

The screen shown first describes the process and then lists all the keys which have been removed between the version being translated from and to. This is for informational purposes, but it should explain what has happened to some of the translated keys you have already done.

The next section takes all the keys which exist in both the version being translated from and to, then compares their source language values. If the values are identical then the translation is automatically migrated to the new language for the target version. For any key matches where the value differs a manual upgrade process is required and an upgrade grid is shown with the following information on it:

- Key
- Original Source Value
- Original Translated Value
- New Source Value
- A text box for you to enter the new translated value

Should you leave the textbox empty, that key will be filtered out and be untranslated. It can always be translated later through the edit / complete methods.

Note: you will not be shown any keys added, as this information is of little value at this point

When you have finished, you will be returned to the language list.

Original Specification

An Alternative Approach

I have created a prototype to see how this would all come together, and I've found that using attachments as a means of data storage leads to significant uncertainty when it comes to things like caching & change detection, metadata association and most of all - application versioning, which was ear-marked to use the Confluence versioning.

The resulting code is solid but contains a fair amount of cludge which makes it feel messy to walk through and I can imagine it'll only get worse over time. To be honest, the whole idea of having the data layer free to user meddling isn't a nice one.

The Alternative

As this plugin will be destined to be used on Confluence 2.3, I would like to suggest using Bandana to store the properties - keyed both on the application name, application version & translation revision, language & country codes. The translation revision starts off at 0 and work's it's way up.

If the application named in the macro doesn't exist, it would be created - the contents of the properties file would be initially blank. The master properties could be updated either through a UI (it could allow you to add keys only when editing the master properties) or through uploading a new properties file.

A manager would sit in the middle and control access through to the backend - this would significantly reduce the amount of checking code based needed. It would also provide a very simple way of dealing with application versioning. When a new application version is added, the user selects which existing application version it's based on, and a new carbon copy is born. This can be updated in the ways listed in the paragraph above.

When factoring the amount of time saved, and looking at the additional work needed to give users things like download links to get the properties files - the estimates come to the same values. The major benefit of doing it the alternative was is that you get a cleaner & simpler codebase coupled with increased functional flexibility.

Below I have quoted for the first 3 phases which comes to a total of 19 hours (+/- 10%).

Phase 1 - Create Translation

Estimated 10 hours

A page will hold a series of attachments. One attachment represents the original language: MyApp.properties. There can be multiple properties files per page, if necessary.

A macro called {languages} placed on that page will display a table with all the translations for each properties file based on which files exist: MyApp_fr_fr.properties, MyApp_fr_ca.properties, etc.

The language also compares the number of keys in each attachment and then calculates a percent-complete by comparing to the default properties file. (You could do this naively by counting keys, or more intelligently by actually comparing which keys are there.) You'll display a little bar graph, and (33%, 33/100).

There is a "create new translation" button which uses an XWork action ask the user which app, language and locale and then creates new, appropriately named attachments. The attachments start out blank.

Phase 2 - Complete Translation

Estimated 6 hours

Next to each translation in the languages plugin, there is a button called "Complete translation." This jumps to another XWork action which pulls up a random key from the missing ones of that particular translation. It presents the key, the English (or default) translation, and then a text box for the user to fill in. It also has a completeness bar graph across the top. When you hit finish and hit next, it just brings up another random empty key. Eventually, you can hit "done" and go back to the main page.

There should be two other macros. One that lists all complete keys and another that lists all incomplete keys for a given language.

*Dan Hardiker
To make things technically simpler, I'd like to have Phase 2 and 3 work the same way (both using DWR preferably).*

Phase 3 - Edit Translation

Estimated 3 hours

Another button labeled "Edit translation" shows you a big table containing the key, the Default language, and your language. Each of the "your language" table cells should be click-to-edit, and save the translations ajax-style. It has a search form up the top. Ajax-style filtering would be best here.

Phase 4 - Upgrade Translation

Estimated ? hours (estimation varies too much to predict at this time)

When you upgrade your app, and therefore your properties file, you click the "Upgrade translation" you make a note of the fact that application version has changed in the attachment comment. When the version of the default properties file revs, we also rev all of the other files. We automatically remove any keys that have disappeared, find the new blank keys, and note (somehow?) which keys have changed. Then we recalculate the percent complete, and people start working to translate again. When we present a key that changed in the default language (thus invalidating the previous translation in the target language) we should show four items: previous version english, new version english, previous version french, edit box for new version french.

Resources

- Remember the Milk
- Google Translate

JN Comments on First Delivery

- In translate-apps macro, show empty row with "none added" if there are no rows.
 - **Dan Hardiker** ✓ This was the way it should have worked - the bug has been found, and quashed
- In all tables, make a footer row to contains the commands "add applicaton" etc. and right align them.
 - **Dan Hardiker** ✓ Done on all list screens.
- All the pages need some top margin between the title and the page chrome.
 - **Dan Hardiker** ✓ Done - added 15px margin-top to all the titles.
- Tables should probably be the same width (like 80% or something variable).
 - **Dan Hardiker** ✓ I've set it to 100% width of the standard form div which is 740px in width. I've wrapped the macro's output in the standard form div too.
- "Version Count" -> "Versions"
 - **Dan Hardiker** ✓ Done on all list screens.
- "Language Count" -> "Languages"
 - **Dan Hardiker** ✓ Done on all list screens.
- Put the action links in the "Versions" tab instead of having Actions column?
 - **Dan Hardiker** I'm not sure what you mean about versions tab? Can you be a bit more precise about which action links on which screen you'd like putting where?
 - **Jonathan Nolen** Sorry, should read "Versions column" not tab. I'm suggesting (but not sure) that we should eliminate the action column and simply link the contents of the Versions column – the number, if there are any, or the word "add" if there are zero. what do you think?
 - **Dan Hardiker** ? As you're not sure I've given you both implementations. I've added linking to the number in the Versions column, and changed 0 to add. The Actions column is still there, however that can easily be removed - or the Versions column reverted. Alternatively, you might like how it looks now! Let me know.
- Source locale should read EN_US instead of English, right?
 - **Dan Hardiker** English (United States) has the locale en_US, English has the locale en_.
 - **Dan Hardiker** Do you want me to convert over to the locale; if so just for the source locale on the "list applications" screen?
 - **Dan Hardiker** Do you want me to remove the non-country based locales (so English is invalid without a country specified)?
 - **Jonathan Nolen** No, I understand now. That's fine as is.
 - **Dan Hardiker** ✓ Okay.
- On create application, should we add a link to the application homepage that is used in the first column?
 - **Dan Hardiker** Should we be asking for a homepage URL, or do you have some way of working it out?
 - **Jonathan Nolen** No, we'd ask for it on the create form.
 - **Dan Hardiker** ✓ Okay - added the homepage which validates it as a URL, if optionally entered.
- On new application page: "This is the language of the below entered properties." - what does this refer to?
 - **Dan Hardiker** ✓ This has been removed - it was deprecated when versions were slotted in.
- Should we be able to download partially complete languages? I think so.
 - **Dan Hardiker** ✓ This was the way it should have worked - the bug has been found, and quashed

Text Changes

An application has a series of versions, each of which contains a source language which can be translated into other languages. This form creates a new application with no versions. In the next step, you will create your first version.

Dan Hardiker ✓ Changed on the "add application" screen, and partially changed on the "add version" screen.

You should paste in the source langage properties which will be available for translation. It uses the standard Java property file form of key = value. These properties cannot be changed for this version once they have been set.

Dan Hardiker ✓ Changed on the "add version" screen.

You should paste in the langage properties which are used in the translation. It uses the standard Java property file form of key = value and the keys are matched up against the source for validation. This is optional, and not all keys have to be present - however, all keys entered here must also exist in the source translation.

Dan Hardiker Changed on the "add language" screen.

Upgrade page.

A total of 1 keys have changed going from v1.0 to v1.1. Examine each new or changed property and update the value to match the new source language. Any properties left empty can be completed later.

Dan Hardiker Updated the changed keys section description.

The upgrade process compares the current translated version for this language with the new version of the source language. Where the source language is unchanged, the translated property will be upgraded automatically. Where properties have been deleted from the source language, they will be automatically removed. Where properties have been changed, you will be prompted to enter a new translation. New properties will not be listed and should be translated in the normal manner.

Dan Hardiker Updated the header description (yellow box at the top).

NOTE: why don't we show them the added one as well?

Dan Hardiker Done.

- Change "Original Language in English (United States)" to "Old text in English (United States)", etc.
 - Dan Hardiker Done - I have also converted "Original Translation" to "Old translation" to fit.
- Change "Upgraded Language in English (United States)" to "New text in English (United States)", etc.
 - Dan Hardiker Done - I have also converted "Upgraded Translation" to "New translation" to fit.
- the upgrade process left the bar at 0%, even tho I'd completed the translation.
 - Dan Hardiker I am unable to replicate this - I assume you are talking about the progress bar on the resulting Translations list page, which updates properly for me after an upgrade.
- Is this sentence necessary: "This bar tracks the ratio of translated keys against ones remaining. Once this bar reaches 100% you will be taken back to the list of translations to select another language."
 - Dan Hardiker I assume this is on the "Complete" page as it doesn't appear on the "Upgrade" page ... I can remove it if you like, however it informs the user that when they complete the translation they will be automatically taken to a different place - that behaviour might otherwise catch them by surprise?
Jonathan Nolen I still think we should remove it. It just seems like you're wasting too much space and too many words for something that only occurs very infrequently.

edit page.

Note: Search will not see any changes you've made until you refresh the page. You cannot modify the search keywords while a key is being translated.

Dan Hardiker Done - however I don't like the way it works. The search **should** be able to see the changes made, and it shouldn't make it appear as if the changes have been lost until refreshing. I didn't have time to restructure the JS to have this effect in the time allotted unfortunately.

Dan Hardiker If this is something you think would be worth a couple of hours extra work, that is what it would take to restructure the edit page's JS to cope. Let me know.

Jonathan Nolen yes, i think this is definitely worth a couple more hours.

Click on any translated key to edit it. Press enter to save your change, or escape to cancel.

Dan Hardiker Skipped - see below.

- the text in the yellow box and above the table are redundant. we only need one.

Dan Hardiker As all the other pages have the standard header (yellow) box, I have kept that one.

BloggerRPC Plugin

| | |
|-----------------------|--|
| Name | BloggerRPC Plugin |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Implement the Blogger API (or some other one) so that people can use blog tools (like marsedit or ecto) to blog on Conf. |
| Plugin Implementation | Confluence Blogging RPC Plugin |

Search Keyword Macro

| | |
|-------------|--|
| Name | Search Keyword Macro |
| Author(s) | |
| Priority | Low |
| Description | hint/add keywords to the search index for a page to manipulate search results. |

See [DISC:CONF-1788](#).

SVN Commit Acceptance

| | |
|-------------|---|
| Name | SVN Commit Acceptance |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Allow JIRA to accept or reject commits on certain criteria. |

Phase 1 - Rules

This plugin will let you define certain basic rules that will determine whether a commit will be accepted based on information in JIRA. For example, you could define a rule that no commit will be accepted unless it contains a valid issue key. Or you could set a rule that no commit would be accepted if the issue was not in an "open" state.

Both CVS and SVN have the concept of [pre-commit-hooks](#), which can be used to run arbitrary code at commit time. We would offer a set of these hooks that the admin would configure for this JIRA instance and install on his source code repository. The would run, RPC into JIRA to evaluate the rules, and the allow or disallow a commit.

Inside JIRA, we would need, in addition to the evaluation action, an action that allowed users to configure the rules behaviour. Initially, this would be a site-wide setting, and just have three checkboxes for the three rules below.

Requirements

- A new action to allow you to set the rules
- A new action to run the rules over a commit which would be called from RPC and would return true or false.
- Pre-commit hook scripts (written in Perl? Ruby? Python? batch for Windows?) that would be called from SVN (and CVS) to check with JIRA and allow or disallow the commit based on the outcome of the Rules RPC action.

Initial rules

- Commit must have Issue Key
- Issue must be in UNRESOLVED
- Issue must be assigned to COMMITTER

Phase 2 - More Rules

Later, we want to add more rules and offer a standard set of rules that they can choose from, as well as the option to define their own. I imagine the interface being similar to the [rules UI in Apple Mail](#).

- Issue must have a fix/for version
- Issue must have specific FIX/FOR VERSION
- Issue's fix/for version must be RELEASED|UNRELEASED
- Issue's component must be COMPONENT
- Committer must be in GROUP
- Committer must have PERMISSION

Phase 3 - Workflow Action

The second phase of the project involves the ability to trigger actions from commit message. The user should be able to put keywords into the message that will trigger actions. For example: *NEW_ISSUE*, *RESOLVE* or *CLOSE*. (Chose whatever syntax you think best.) If the action were to fail, should we stop the commit or allow it to continue. (I think stop, but could be argued otherwise. Or it could be a preference.)

- We should support all the available workflow transitions for a given issue/committer.
- **New Issue** this is a little trickier, in that we should return the issuekey from the action and automatically insert that into the commit message before commit.

Phase 1 design sketches

Requirements

| | |
|--------------------|---|
| Java version | 1.4 |
| Base package | com.atlassian.jira.ext.commitacceptance |
| SVN repository URL | https://svn.atlassian.com/svn/public/contrib/jira/jira-commitacceptance-plugin |
| FishEye URL | http://svn.atlassian.com/fisheye/viewrep/public/contrib/jira/jira-commitacceptance-plugin/trunk |
| Docs | Wiki pages at... |

Separation of concerns

1. all business logic is implemented on the serverside
2. the client is just a dummy "delegator" that:
 - a. passes basic info (like commit message) from the SCM system to the server
 - b. receives a boolean flag that specifies whether the commit is accepted (TRUE) or rejected (FALSE)
 - c. accepts/rejects the commit with the particular SCM implementation (SVN or CVS) based on this received value

Components

EvaluateAction

The main serverside component. Basic responsibility:

1. parses issue keys from the log message
2. loads the issues and provides getters for the predicates to access them
3. loads user-defined criteria
4. evaluates a combination of Predicates
 - a. see [org.apache.commons.collections.Predicate](#)) that is built based on the user-defined criteria with AND relation
 - b. there can be multiple issue keys in the commit message. In this case, the criteria must be true for **all** of them.
5. returns the boolean result by writing to the *HttpResponse*

```
public abstract AbstractJiraPredicate {  
    AbstractJiraPredicate(String userName, Set<Issue> issues);  
    String getUserName();  
    Set<Issue> getIssues();  
}  
public HasIssuePredicate extends AbstractJiraPredicate ...  
public IsIssueInStatePredicate extends AbstractJiraPredicate ...  
public IsIssueAssignedToPredicate extends AbstractJiraPredicate ...
```

This is tested by *EvaluateActionTests* test cases with a mock JIRA implementation.

Client

The only client-side component to be installed to the SCM system. The goal to keep everything in the same physical client (a single JAR shipped!) where SCM dependent behaviour is hidden behind interfaces and implementation classes.

1. fires an HTTP POST request passing the committer's username and the log message.
2. receives a single boolean written as text to the response
3. it is implemented as a tiny commandline Java application. (If it's difficult to invoke from the commandline, then helper .bat and .sh will be delivered).
4. we deliver a full guide both for CVS and SVN how to configure the client as a commit hook.

ConfigureAction

The actual UI front-end.

1. gets available statuses and usernames
2. displays a groupbox of 3 checkboxes combined with comboboxes:

| |
|---|
| [x] Commit message must have an issue key |
| [x] Issue must be in \$STATE state |
| [x] Issue must be assigned to \$USERNAME |

3. persists and reloads these settings

Open questions

Q: what syntax to use when defining issue keys in the commit message? Something like this?

This is a fix for [JIRA-123].

A: Ideally, there should be no special syntax – it should just recognize the issue key. There is already code for this in JIRA, b/c we highlight the issue keys in comments, so you should be able to use that.

Q: can the client and the server communicate via a plain vanilla HTTP POST or do you have other requirements?

A: HTTP sounds fine. Although making sure that we can do SSL would be nice, if not too much more work.

Q: where do you want to persist the site-wide user settings (the status of the checkboxes and comboboxes)?

A: I'm not sure yet. We'll probably need to stick a new menu item in the admin page, but that won't be possible until 3.7. For the meantime, just pretend that it is.

Vote-for Custom Field

| | |
|-------------|--|
| Name | Vote-for Custom Field |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | A custom field for the column view that allows you to vote for items ajax-style. |

Mark For Review

| | |
|-------------|--|
| Name | Mark For Review |
| Priority | High |
| Complexity | Low |
| Author(s) | Mike Cannon-Brookes, Jonathan Nolen |
| Description | Allow readers of the wiki to mark content for 'review' by managers. This content may be marked because it's inappropriate, worthy of promotion or simply incomplete. |

Background reading

Some of this functionality can be addressed, albeit in a complex way, by the [Workflow Plugin Prototype](#). In particular, a couple of new workflow triggers are currently being worked on that can change a label on a page or email a changed page to a particular group.

There is also a fairly detailed discussion of a similar idea at the [Approval Macro](#) page. The 'approval' and 'review' concepts are very similar.

There is an implementation of this idea now available at [Mark for Review Plugin](#)

User interface

- User marks content for review
 - Button / icon next to the favourite icon
 - Brings up screen which asks for:
 - type of review (select box)
 - textarea to ask for user's comment
- Review gets saved in bandana for the space including the user who marked the content, their comment, the review level, the date and revision of the review.
- Reviewers get emailed a link to the review

Reviewer interface

Shows a list of all reviews required for a given space.

Reviews can then be deleted after they are acted on (or ignored).

The list of reviews shows a table which links to the user, the page (including revision) being reviewed, date, review level.

Space admin interface

Configuration screen allows you to set up the reviewers for a space. Here you select which users should be reviewers, or delete existing reviewers.

Admin interface

Configuration screen allows you to set up the 'review labels'.

Initial default labels are:

- Inappropriate content
- Should be deleted
- Should be featured
- Incorrect content

Math Equations Plugin

| | |
|-------------|---|
| Name | Math Equations Plugin |
| Priority | Medium |
| Status | Released |
| Author(s) | Stepstone Technologies |
| Description | Allow users to use LaTeX to render pretty math equations. |

Background

One thing that Confluence appears to lack is support for writing and rendering maths equations using the markup language. I know that the Confluence rival [MediaWiki](#) supports the use of [Latex](#). A quick web search didn't come up with any similar plugins for Confluence.

Plug-in Now Available

The [LaTeX](#) Plug-in is now available for download.

Google Maps Plugin

| | |
|-------------|---|
| Name | Google Maps Plugin |
| Priority | Medium |
| Status | Released |
| Author(s) | [|
| Description | A macro to show you a Google map of a particular address. |

A macro to show you a Google map of a particular address. Could be as simple as {gmap:375 Alabama St, San Francisco, CA} which would show you an inline, editable Google map [like this!](#)



Work has started on this. Check out the [Google Maps Plugin](#).

New Code Macro

| | |
|-------------|---|
| Name | New Code Macro |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Syntax Highlighting for ECMA 262 (eg: JavaScript / ActionScript) based languages CONF-893 |

Confluence Contributors Plugin

| | |
|----------|--------------------------------|
| Name | Confluence Contributors Plugin |
| Priority | Medium |

| | |
|------------------------------|--|
| Status | Released |
| Author(s) | |
| Description | List all the authors who have contributed to a page. |
| Plugin Implementation | Confluence Contributors Plugin |

Comments Tab Theme

| | |
|--------------------|--|
| Name | Comments Tab Theme |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Move the comments to a separate tab, ala wikipedia. At least two customers interested. |

JIRA Calendar Plugin

| | |
|--------------------|---|
| Name | [JIRA Calendar] |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Buid calendar functionality into JIRA as a portlet and as an iCal export. |

Social Bookmarking

| | |
|--------------------|--|
| Name | Social Bookmarking |
| Author(s) | Shannon Krebs |
| Priority | High |
| Progress | refer to Confluence Social Bookmarking Plugin |
| Status | Released |
| Description | A del.icio.us-style social bookmarking feature for both internal and external URLs |

Description

Bookmarking is an extremely useful group-behaviour. It allows you to easily collect information from all kinds of sources and disseminate it through a team.

Use-cases

- A. Jim uses the bookmarking system entirely for his own use. He bookmarks things he wants to remember and they are stored in his personal space. He doesn't mind anyone else seeing his bookmarks, so he leaves his space totally open.
- B. Michael the Manager wants to keep up with what his team is reading, so he creates a subscription for bookmarks "by:jim," "by:pam" and "by:dwight."
- C. The Sales team wants to deliberately share bookmarks with each other, so they create a tag "for:sales" and then each member subscribes to that list.
- D. Jim wants to send a funny link to Pam, so he tags it as "for:pam" and Pam reads it in her personal space.
- E. Stanley creates a page about the competition, so he includes a bookmark macro that only displays things tagged "competition."
- F. Jan doesn't want anyone else to see her bookmarks, so she sets her personal space to private. Her bookmarks don't show up in any other lists, no matter what they are tagged.
- G. Pam bookmarks a URL for sales, and Jim comments on the bookmark disagreeing with the article.

Tasks

- Create a bookmark (choose location)
- Tag a bookmark
- Tag a bookmark for: someone/some group
- Hide bookmarks
- Include list of bookmarks in a page (by user, tag, space)
- Subscribe to a list of bookmarks (by user, tag, space)
- Comment on a bookmark

Implementation

Ideally, Social bookmarking would take advantage of the new Confluence Content Types, but unfortunately, that feature isn't available yet. However, I think it should be possible to achieve the same functionality using the existing capabilities of Confluence. Here's how:

View bookmarks

The plugin should include a macro for `bookmarks`. You should be able to request bookmarks for user(s), group(s), spaces(s) or labels(s) (or any combination thereof). The macro will list X requested bookmarks, order by most recent in a list format, containing the linked title, the comments and (maybe) the tags. The `bookmarks` macro should include an RSS link to that feed of bookmarks.

Bookmark macro:

```
{bookmark} -- most recent 15 bookmarks in this space

{bookmark:space=SALES,DEV      -- in SALES OR DEV spaces
 |labels=for:jim,foo,bar -- labeled for:jim, OR foo OR bar
 |user=pam           -- created by Pam
 |max=20            -- show 20 links
 |showLabels        = true|false
 |showDescription   = true|false
 |showAuthor        = true|false
 |showDate          = true|false
 |sort=+date+,author,title
}
```

Bookmark feeds

Just as the `bookmarks` macro should be able to list bookmarks by various filters (users, labels, etc.) we should also publish an RSS feed of the bookmarks, by with those same filter. We should be able to leverage all the existing RSS feedbuilder functionality, but just limit it to pages that we know are bookmarks based on their position as children of the "Bookmarks" page.

Add/delete bookmark

The plugin will need to publish a new action called "add bookmark." This is what the bookmarklet will call when the user wants to save a new bookmark. This action will publish a simple page with five fields: url, title, comment, labels and space. The url and title should be pre-filled from the query parameters that the bookmarklet submits. The comments should allow free-form text entry. The tags field should mimic the tags interface that we already have on pages. It should pull its labels from the same global and personal namespaces that a page would. The space should default to the most recently selected space (or personal space if none), but should also offer any space on which the user has create-page permission.

To delete a bookmark entry, just delete the page.

Bookmark entry will also need to be able to support captcha entry if enabled in the Confluence instance.

Edit bookmark

To edit a bookmark, just edit the contents of the page or change the labels on the page.

Space Tab

Each space should get a new space tab, analogous to pages, that lists all the bookmarks for that space. It can use the same macro as "View Bookmarks" above, but should also include a view of `for:space` bookmarks – that is, bookmarks left in other spaces, but tagged with "for:this_space." This page should also include links to bookmark feeds.

Bookmark storage

Since we don't have content-types yet, we'll need to store bookmarks as pages. The plugin should create a default "Bookmarks" page.

The bookmarks themselves should be saved as child pages of this "Bookmarks" page. Each bookmark should use the bookmark title as the page title. The page content should contain a table with the four pieces of user-submitted data (see above), perhaps surrounded by meta-data-list if that proves useful.

There are five fields that we care about.

- **Title** - Stored as the page title
- **Description** - stored in page metadata
- **Tags** - stored as page labels
- **Author** - stored as page creator
- **Date** - stored as page creation date

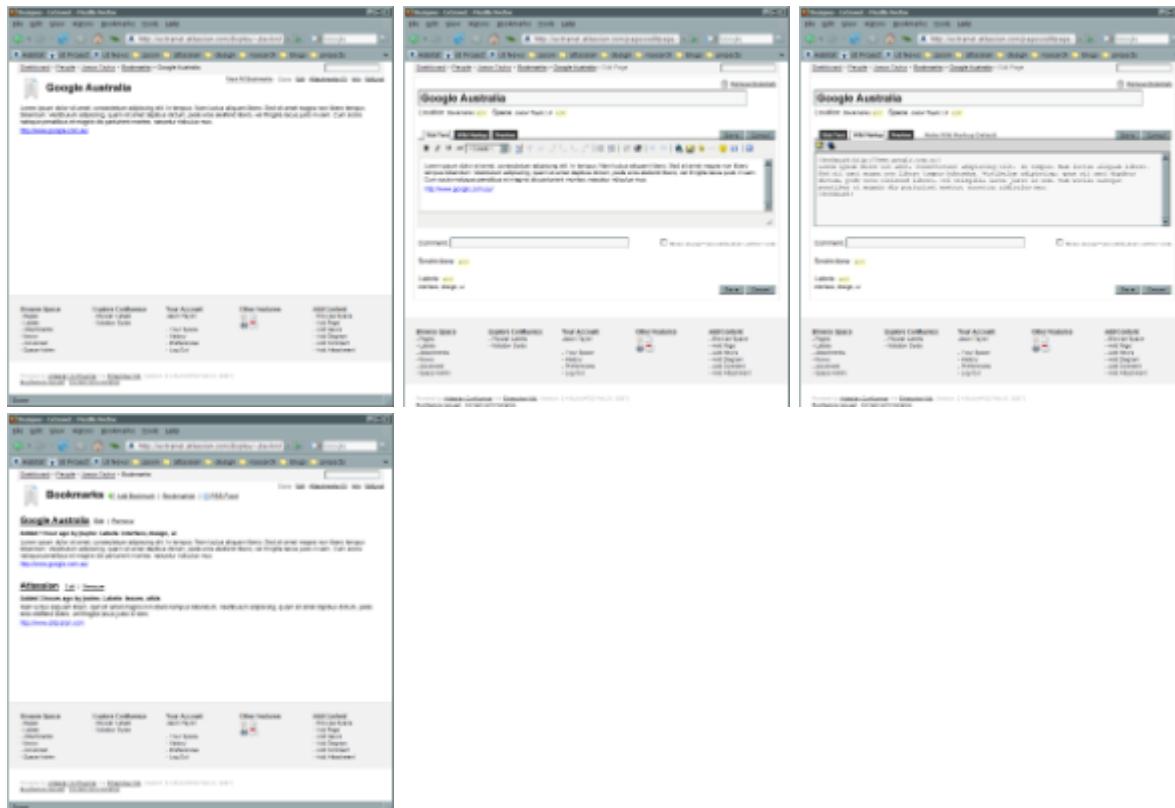
Title, tags, author and date will be stored using the page facilities that already exist. Some bookmark data needs to be stored in page metadata, for easy and fast aggregation. Three properties will be required: `bookmark:true`, `bookmark-url:http://google.com`, and `bookmark-desc:The google search engine`.

The bookmark pages should be open to comments per space permissions, so if someone wants to continue the discussion about a link, they can.

Bookmarklet

The bookmarklet needs to be a short javascript that can be dropped on the bookmark bar in all browsers that will get the title and URL of a document, post it to the "add bookmark" action, which will record the bookmark and then ensure that the user returns to the original page. Any selected text is automatically entered into the description field.

Mockups



UI Feedback

JT comments in bold

Bookmarks Space Tab

- ✓ there's a typo in the bookmarks RSS grey box
- ✓ The grey panels are very close to the right of the page - they should right align with the RSS icon at the top of the page. Mike liked the panels.
- ✓ remove comment icon (too heavy)
- ✓ Edit and remove functions are floating in the middle of nowhere. The placement is inconsistent with other yellow buttons links within Confluence.
- ✓ Mike would like to use normal links as opposed to the current style of yellow box.
- '0 Comments' label is redundant
 - Jonathan Nolen It's not redundant - bookmarks can have comments.
- The text blocks are also floating - can we style the headings with an underline or something akin to a H1.
 - Jonathan Nolen Waiting on mockup.
- It's unclear what the two 'view' options (on the Space Bookmarks page) are for * do we need to differentiate between 'in' and 'for'
 - Jonathan Nolen Agree - need a better solution.

- I'm still unclear regarding the function of these links - more info please
- Shannon Krebs I think it fits the personal space well where normally *bookmarks in* will be bookmarks created by the user and *bookmarks for* are bookmarks that other people have labeled for that user.
- the Space Bookmarks page needs a visual update - there's no delineation of content items, the layout isn't very tight and the type could be tweaked. This is one page that we will need to provide some form of mockup...
 - Jonathan Nolen Waiting on mockup.
 - Mockup should be ready by early next week...LMK if you need it sooner

Bookmark Entry

- ✓ The space drop down label should appear above the field - as per other fields.
- ✓ remove the dotted line it's not needed * and Mike doesn't like it...
- ✓ The 'Bookmark in Confluence' function should utilise the confluence favicon.
- After saving a Bookmark - using the 'Bookmark in Confluence' function - the page returned is the original site. I would expect the page displayed would be the newly saved bookmark. Throwing the user back to the original site doesn't provide appropriate feedback to the user.
 - Jonathan Nolen Totally disagree. This is exactly how a bookmarklet is supposed to work.
 - Shannon Krebs Making the bookmark page a popup might be a solution.
 - Hmmm. I'm not overly familiar with bookmarklets, however, the experience and flow is disjointed. It's also extremely unclear that my action has been successful. I don't think a popup is necessarily the answer, but we should be logical in our approach. Does delicious do it this way now?
- The 'Add Bookmark' page loses the page furniture...the presentation of this page should be no different to Add Page or Add News.
 - Jonathan Nolen Also disagree. The 'Add bookmark' is supposed to be an extremely light-weight, interstitial style page that takes your input and sends you right back where you started. I don't want to offer any other options. There's way too much crap on our input pages.
 - JT: We need to be consistent within Confluence, I'm not suggesting additional options, just that the page furniture is displayed - there's not even a Space logo on display!
 - Shannon Krebs If the user is coming from the bookmarklet it is hard to know which space theme should be used, especially if they don't have a personal space

.bookmarks Page

- ✓ there's a lot of space within the panels - is this consistent with other panels in Confluence
 - Jonathan Nolen Uses standard info macros
- ✓ what's the DOT bookmarks mean?
 - Jonathan Nolen This is the container page to hold all of the bookmarks - a compromise until we get content-types. It's like a hidden file. If you want to change, let me know what.
 - Shannon Krebs Was intended to be like admin files/directories in *nix filesystem to reflect that it is not a normal page that should be edited. Also need to pick a name that is unlikely to have already been created.

Single bookmark view

- ✓ lots of space on the bottom - is this consistent with other panels?
- ✓ remove the dotted line it's not needed - and Mike doesn't like it...

NAnt builder

| | |
|-----------------------|--|
| Name | NAnt Builder |
| Priority | Medium |
| Complexity | Medium |
| Author(s) | |
| Description | Create a Builder Plugin for NAnt |
| Plugin Implementation | BAMBOO:Builder Plugin Module |

Background

Create a Builder Plugin for [NAnt](#).

Zip Search Extractor

| | |
|----------|----------------------|
| Name | Zip Search Extractor |
| Priority | Medium |
| Status | Released |

| | |
|--------------------|--|
| Author(s) | |
| Description | a new Search Extractor for Confluence which is able to index the individual files within a Zip file. |

Your task is to create a new [Search Extractor](#) for Confluence which is able to index the individual files within a Zip file.

Requirements

The extractor must successfully search and index zip files containing all of our supported documents types:

- PDF
- Word
- Excel spreadsheets
- Powerpoint attachments
- Text docs

Content Searcher for OpenOffice docs

| | |
|--------------------|--|
| Name | Search Extractor for OOF and ODF |
| Author(s) | |
| Priority | Medium |
| Description | Search inside OpenOffice and OpenDocument docs |

Import Visio Diagrams

| | |
|--------------------|--|
| Name | Import Visio Diagrams |
| Author(s) | |
| Priority | Medium |
| Description | Import Visio files and display diagrams on the page. |

Allows users to import Visio files and then display the diagrams on the page. Currently in order to accomplish displaying visio diagrams within Confluence is to convert the Visio file to a jpeg and then attach to the appropriate wiki page to display as a picture.

Having this plug-in will be a great help to all folks, especially IT folks, who have tons of Visio diagrams that need to be communicated in a wiki-format!



WebDav Plugin

| | |
|--------------------|--------------------------------|
| Name | WebDav Plugin |
| Priority | Medium |
| Status | Released |
| Author(s) | |
| Description | Access Confluence over WebDAV. |

Finish the [WebDAV Plugin](#) so that it works reliably from both read an write. See also [Confluence as a WebDAV server](#).

Existing sourcecode: <http://svn.atlassian.com/svn/public/contrib/confluence/webdav-plugin/trunk/>

WebDAV Plugin JIRA Project: <http://developer.atlassian.com/jira/browse/WBDV>

[Developer Docs](#)

[Servlet Plugins](#) (WebDAV is implemented as a servlet plugin.)

Notes

- I'll need names and passwords for the developers working on this so that I can give them access to SVN.
- Please use the WebDAV JIRA Project for your development. File tasks for yourselves and use the issue keys in your commit messages.
- The feature list is priority order. Do your best to work down the list, and to fully complete and test each feature before moving on to the next.

Requirements

- WebDAV must work from Microsoft Office apps.
- WebDAV must work from the Mac OS.
- WebDAV should work from 'Web Folders' in Windows Explorer.
- There should be a thorough test suite, in as much as possible.
- Fix as many bugs as possible from the [WebDAV project](#).

Features

- You must be able to read and write the text-only (markup) files. (This is the big one. Reading mostly works right now, but writing is extremely dodgy. See [Chart](#). This will probably be the most work.)
- Access security (space and page level) must be enforced by WebDAV. (This may work already.)
- You must be able to view, add and remove attachments.
- Applications must be able to access attachments as if they are files: i.e.
 - Application opens file from within the Confluence WebDAV directory
 - Re-saving the file from the application will create a new version of that attachment
- Operations (copy, move, delete, add) must work with multiple files.
- The plugin should automatically create PDF, Word and HTML documents of each page, represented as files on the WebDAV server, with their contents generated when the files are accessed. (This may already work.)
- You must be able to drag-and-drop manipulate hierarchies of pages.
- Personal spaces and news must be accessible

Out of Scope

- WebDAV versioning - old versions of pages or attachments are unavailable via WebDAV

Crowd Ruby Integration Libraries

| | |
|--------------------|--|
| Name | Crowd Ruby Integration Libraries |
| Priority | High |
| Complexity | Medium |
| Author(s) | |
| Description | Allow Ruby and Rails apps to use Crowd for their user store and SSO. |

Description/features

Allow Ruby and Rails apps to use Crowd for their user store and SSO.

Usage

Technical Implementation

Other Resources

Crowd Extension page has this implemented : [Ruby client & Ruby on Rails plugin](#)

Crowd PHP Integration Libraries

| | |
|--------------------|---|
| Name | Crowd PHP Integration Libraries |
| Priority | High |
| Complexity | Medium |
| Author(s) | |
| Description | Allow PHP apps to use Crowd as their user store and SSO |

Description/features

Allow PHP apps to use Crowd as their user store and SSO

Usage

Technical Implementation

Other Resources

This is implemented; see the [Crowd extension page](#).

Usage Stats Plugin

| | |
|-------------|--|
| Name | Usage Stats Plugin |
| Author(s) | None - you can volunteer to contribute |
| Priority | High |
| Progress | No progress |
| Status | Awaiting developer resources |
| Description | Show graphs of site usage |



Complete! Atlassian now provides a supported [Usage Tracking plugin](#), which extends the stats provided in Confluence 2.3 and up. See also [How Do I Get More Statistics From Confluence?](#) and [DOC:Obtaining Confluence Instance Metrics](#).

Description

We could show much richer data about the usage, health and growth of a confluence wiki. We can start with what is provided in the [Confluence Activity Plugin](#).

- [List of most viewed pages in a space](#)
- [List of most edited pages in a space](#)
- [List of most frequent contributors to a space](#)
- [graph of space views](#)
- [graph of space edits](#)

Building from that, there are many other statistics that would be useful to have. We should record stats on creates, edits, views and size of change (same as diff+/-). All graphs should be filterable in several standard ways: time (hour, day, week, month, year, forever), specific time (e.g., March 3rd - March 7th), page(s), space(s), user, group, tag. Graphs should be drawn as line, area, bar or pie-chart. Here are several examples:

- Area graph of edits for pages in space in the last week.
- Area graph of edits by Michael in the last month.
- Area graph of edits on this page.
- Pie chart comparing edits vs. Views for this page or space.
- Line graph of views on all pages with tag:x.
- Line graph of views on this page in the last month.
- Line graph comparing views on space x vs. space y over the last month.
- Edits by 'sales' group in 2006.
- Bar chart comparing edits by Jim, Pam and Dwight.
- Area graph showing the growth of a page or space in total size (lines or pages).([similar to this, but less detailed.](#))
- Line graph comparing growth of space x vs. space y in total pages.

References

- http://www.research.ibm.com/visual/projects/history_flow/
- http://www.christine.net/2006/08/rd_technologies.html
- Site Statistics
- Activity Plugin Docs.
- Global Stats Plugin

If you are willing to contribute developer time or resources towards this project, please contact [Atlassian](#).

Trusted Application Authentication (Glossary Entry)

Trusted application authentication (or 'trusted apps') is an Atlassian-developed mechanism allowing two applications to exchange information

on behalf of a logged-in user. For example, an administrator can configure [JIRA](#) and [Confluence](#) to communicate in a trusted way, so that Confluence can request information from JIRA on behalf of the currently logged-in user. JIRA will not ask the user to log in again or to supply a password.

Approval Macro

| | |
|-------------|---|
| Name | Approval Macro |
| Author(s) | |
| Priority | High |
| Description | Allow users to read and approve content |



A page approval plugin implementation has been developed by [Saikore](#). More details can be seen at the home page.

Often in organisations, knowing who has read and who has approved a document are very important things. However modern document management systems make read and approval checking very tedious to implement and monitor. I'm wondering if we can't come up with a simple agile/Wiki like solution to this problem.

Simple Macro Approach

Perhaps it could be as simple as a macro like:

```
{approval:Bob, Jim, John}
```

which would print out a table something like this:

| Name | Read | Approved |
|------|------|----------|
| Bob | ✗ | ✗ |
| Jim | ✗ | ✗ |
| John | ✗ | ✗ |

As people start to read or approve the document, it might end up looking something more like this:

| Name | Read | Approved |
|------|------|----------|
| Bob | ✓ | ✓ |
| Jim | ✓ | ✗ |
| John | ✗ | ✗ |

How does it get read or approved? Simple links really. Imagine *you are John* who is logged in and looking at the page, you would see something like this:

| Name | Read | Approved |
|------|---------------------------|----------------------------------|
| Bob | ✓ | ✓ |
| Jim | ✓ | ✗ |
| John | Mark read | Approve document |

Simply by clicking on the links, you indicate that you have read the document, and give your approval for the document.

Why read and approved as separate steps? Well you might read the document, click the 'read' link and then add a comment requiring more information before you're willing to approve it.

It would be improved if the macro actually coloured the backgrounds of the cells I think, to make it look more like a [FatCow](#) test result. When

the table turns green, your document is approved!

Thoughts?

Advanced Macro Approach

The following points give a very rough specification of how a simple 'wiki style' approval system might work. I describe it as 'wiki style' because it does not require any administration and hence can be managed by the wiki community. If you edit these specifications, please keep in mind that the approach taken here is to make the process as simple as possible for users.

General Requirements

- Anybody can register as an 'approver' of any page. This would be done by clicking on an 'approval' icon which could be a tick icon (next to the star and envelop icons at the top right of a standard page e.g.)
- Users would be able to see a list of all the pages (and their approval status) that they have registered to approve. This could be a similar mechanism to the one that allows users to see favourites or watched pages.
- Pages with registered approvers would display the list of people who have registered as approvers. This could be in a table at the top (or bottom?) of the page, like this:

| Name | Read | Approved |
|------|---------------------------|----------------------------------|
| Jim | | |
| Bob | | |
| Fred | | |
| Jo | | |
| Jill | Mark read | Approve document |

- Note that the table above is what 'Jill' would see if she viewed the page (and has registered as an approver). Jim has read and approved the page, Bob has read and rejected the page, Fred has read the page, Jo has not done anything and Jill has not yet done anything either. Jill can mark the page as 'read' or 'approved' (or unread and unapproved as per the page state) by clicking on the links. It may be better to have the page automatically be marked as 'read' after a user first views it.
- A page would be marked as 'unapproved' until all the people in the approval list have approved it. An unapproved page would be marked as unapproved by some bold red text or background (or something....)
- Once the page is approved, it would have a special version number assigned (and perhaps a special label assigned such as 'APROVED_V1.1'). Some ideas from software version control might be applied here.
- If a page is changed after it was approved, the approval (and read) status for all users would be reset to blank. All users 'read' status would be reset on any page edit.
- Users who view a page that is not approved would be able to easily navigate to the last approved version. By default, the latest version would be viewed, but the user should have a preference somewhere so that their default is to see the last approved version. The users choice for each page could be persisted per page. There would need to be a very obvious visual indication of the approval state of the viewed page and easy navigation to the alternate version. In this way new versions of an approved page can be easily accessed for reapproval development.
- Approvers can unregister without affecting the approval status of documents they approved in the past. However, this may affect the approval state of pages that are currently unapproved (i.e. if the user was the only person left to approve a page and the user unregisters, then the page would be approved).
- There may need to be some way of automatically unregistering approvers if they are inactive (account closed etc.)
- The addition of macros on a page can invalidate the version number of a page by allowing changes without incrementing the version. One way to get around this problem would be to prevent any page authorisation actions on a page with a macro. Since this would disqualify many pages, some macros could be made acceptable if they identify themselves as being 'approvable'. This could be done by introducing some sort of macro property (the default would be unapprovable for legacy macros).

Potential problems

This approach was conceived for an intranet application. It would work because people on an intranet can generally be trusted to some extent and are at least accountable. There may be problems with this approach on the Internet because casual users would be able to register as 'approvers' and then disappear, leaving a page that would never be approved. This could be used as a form of vandalism.

Dynamic Data

Dynamic data can invalidate the approval of a page because the page can change without an edit. For instance, the {include...} macro could show different text based upon what is in the included page, independent on anything in the controlled Confluence repository. Hence, using some macros, an approved page could change without the authorisers knowing about it.

One way to get around this problem is to make macros and plugins 'unapprovable'. Any page with a macro could not be approved. However, some macros do have 'approvable' behaviour (such as the table of contents {toc} macro) and it would be very desirable to allow these macros to be used in approved pages. One way to facilitate this is by adding an 'approvable' property to macros. By default, all macros would not be 'approvable', but if the macro developer sets the 'approvable' property then the macro could be used.

Other Ideas

See http://meta.wikimedia.org/wiki/Article_validation for an extensive discussion of the issue in the Wikimedia community.

CVS Report Macro

| | |
|-------------|---|
| Name | CVS Report Macro |
| Author(s) | |
| Priority | Low |
| Description | Show pretty Reports of SCM usage in Confluence. |

Do something to integrate content from Fisheye/ViewCVS. For FishEye, one of the most popular queries is the "changelog query": checking what changed between versions X and Y. Now that the FishEye 1.1 beta has the query method added to its REST API, I would see two Confluence macros that would be useful: 1) one that allows any arbitrary query and 2) one that provides a shortcut for doing that changelog query.

I like the way [TikiWiki](#) has handled/implemented the [CVS monitoring](#) (CVS log actually) and the flexibility of different reports it's covering:

- Activity By a specific Day/Month
- Activity By Author
- A graphical chart to visualize the traffic
- Recent Changes

Gallery RSS Feed

| | |
|-------------|--|
| Name | Gallery Modifications / Media RSS Feed |
| Priority | |
| Complexity | |
| Author(s) | Joshua Johnston |
| Description | Add a Media RSS feed to the {gallery} so pictures are viewable with Media RSS enabled apps such as PicLens |

Description/features

This is a feature request for adding a Media RSS feed to the {gallery} macro so that pictures in a gallery will be viewable with Media RSS enabled apps such as the [PicLens Extension for Firefox](#)

Usage

PicLens enables full screen picture display and slideshow features... having it available in Confluence via the {gallery} macro would be great.

Technical Implementation

- See implementation details here: [Implementation Details](#)

Other Resources

- [PicLens Extension for Firefox](#)

HTML Diff

| | |
|-------------|--|
| Name | HTML Diff |
| Author(s) | |
| Priority | High |
| Description | A library that will allow us to show HTML diffs instead of markup diffs. |

New users get very scared and turned off when the edit something in a WYSIWYG editor, then compare two versions, and see all this strange psuedo programming code with `h2.` and `*+`. There is no reason a typical user who knows nothing about wiki markup has to see that. I know Twiki, Jot and other wikis show just the formatted text in diffs, not wiki markup. Perhaps this can be a seperate tab or checkbox when selecting two version to compare.

Moved to JIRA

- Confluence plugin for JIRA
- Link Visualization
- Sectioned Editing Macro
- Set field value using XPath post-function

Confluence plugin for JIRA



Renamed Confluence Crosslinks to JIRA Wiki tab

| | |
|-------------|--|
| Name | Confluence plugin for JIRA |
| Author(s) | |
| Priority | Medium |
| Description | Link Confluence page updates to JAC issues |

The Confluence plugin for JIRA should work similarly to the FishEye plugin for JIRA, but for Confluence updates.

Once you have configured a FishEye instance for your JIRA instance, the FishEye plugin for JIRA works something like this:

1. Add a JIRA issue number to your commit message when committing a file to your repository.
2. The source file is then available as a link from the 'Source' tab of the referenced issue.

My idea for a Confluence plugin for JIRA is pretty similar. You configure a Confluence instance for your JIRA instance, then do the following:

1. Add a JIRA issue number to the 'Comment' field when updating a wiki page.
2. The wiki page will be linked from a new 'Wiki' tab of the referenced issue in JIRA.

Link Visualization

| | |
|-------------|---------------------------------|
| Name | Link Visualization |
| Author(s) | |
| Priority | Low |
| Description | Build graphs of wiki structure. |

Dynamically visualise the links between different pages in a Java applet. See the JSPWiki implementation |

Sectioned Editing Macro

| | |
|-----------|-------------------------|
| Name | Sectioned Editing Macro |
| Author(s) | |

| | |
|--------------------|--|
| Priority | Low |
| Description | Edit sections of a page independently. |

<http://jira.atlassian.com/browse/CONF-2761>

Very hard to implement correctly.

Set field value using XPath post-function

| | |
|--------------------|--|
| Name | Set field value using XPath post-function |
| Priority | High |
| Complexity | Low |
| Author(s) | |
| Description | Provide a post-function allowing the value of a target field to be set by applying an XPath expression to the value of a source field. |

Description/features

Provide a post-function allowing the value of a target field to be set by applying an XPath expression to the value of a source field.

Usage

The workflow designer would add the post-function into a Transition, select the target field, select the source field and provide the XPath expression in a third field.

Technical Implementation

Other Resources

Attach a file custom field plugin

We should have a custom field that allows the user to attach the file at the issue creation time.

We can place this custom field on whatever screen we want and any number of times we want.

I guess it is a quite common requirement and someone would have already encountered it.

Any Suggestions?

To Sort

For Bamboo

- Bamboo - Mercurial SCM integration
- Builds triggered by commit

Bamboo - Mercurial SCM integration

| | |
|-------------------|-----------------------|
| Name | Mercurial integration |
| Priority | High |
| Complexity | Low |

| | |
|--------------------|---|
| Author(s) | |
| Description | Integrate the mercurial source code management system into Bamboo as a first class citizen. |

Description/features

Integrate the mercurial source code control system into Bamboo as a first class citizen.

Usage

Technical Implementation

plugin

Other Resources

Builds triggered by commit

| | |
|--------------------|--|
| Name | Builds triggered by commit |
| Priority | |
| Complexity | |
| Author(s) | |
| Description | provide a list of builds triggered by a particular code change |

Description/features

Provide a list of builds triggered by a particular code change vs. the list of code changes that triggered a particular build, which is already available.

Usage

Technical Implementation

Currently, if a commit message lists a jira issue, you can click on that issue and see builds triggered by code related to that issue. Maybe the implementation of this plugin can use a similar approach.

Other Resources

Add Excel cell range to Viewfile Macro

| | |
|--------------------|--|
| Name | Add Excel Cell Range to Viewfile Macro |
| Priority | High |
| Complexity | Low |
| Author(s) | |
| Description | Add Excel cell range to Viewfile Macro |

Description/features

Add Excel cell range to Viewfile Macro.

At the moment the viewfile macro renders the cells in an excel spreadsheet from row 0,column 0 to the lastrow,lastcolumn. It would be very useful to be able to pass firstrow,firstcolumn parameters to render from, rather than it defaulting to 0,0.

Usage

To enable data from specific cells to be embedded into confluence pages

Technical Implementation

Other Resources

Avail - Confluence

- Add contents to all pages in space
- admin log in as
- Better Flowchart Macro
- Collapsible Lists
- Personal Homepage
- Recently added pages
- Show confluence Permissions
- Space Provisioning Plugin
- Suspended Specs
- User Information
- UWC
- Voting and Vote tabulation
- Wiki Markup Editor AutoCompletion
- Wiki Markup Editor Toolbar
- Word, Open Office and Excel Importing

Add contents to all pages in space

| | |
|----------------|------------------------------------|
| Name | Add contents to all pages in space |
| Priority | medium |
| Complexity | low |
| Author(s) | Imran Aziz |
| Description | add contents to all pages in space |
| Done | not yet |
| New Plugin Ref | |

Description/features

add contents to all pages in space, for example if one wants to put a macro on every page of a space then one can add these contents to an admin section in space administration and the contents or macros show up or get executed on every page of a space. It would be wonderful to add some additional features like execute the contents or macros when only a specific label is added to the page, this will allow automation of tasks for all pages of a space using some criteria

Usage

insert contents to all pages

Technical Implementation

create a view page interceptor that adds contents to the end of the page contents

Other Resources

admin log in as

| | |
|--------------------|---|
| Name | admin log in as / Jira SU |
| Priority | normal |
| Complexity | trivial |
| Author(s) | Andy Brook |
| Description | Admin users can adopt any users profile to enable dashboard configuration, check security etc |

Usage

System admin goes to User Browsers, clicks SU against related user and 'becomes' that user.

Technical Implementation

Plugin, requires additional manual velocity template mod due to lack of plugin points.

See [Jira SU](#)

Better Flowchart Macro

| | |
|--------------------|---|
| Name | Better Flowchart Macro |
| Author(s) | |
| Priority | Medium |
| Description | A flowchart tool which can reference sub-pages. |

Create a flow chart tool which can reference pages in a flow chart. The excerpt of each page could be included in some kind of mouseover on each node in the flowchart. Could extend the existing [FlowChart Macro](#)

Collapsible Lists

| | |
|--------------------|--------------------------------------|
| Name | Collapsible Lists |
| Author(s) | |
| Priority | Low |
| Description | Ajax tree view of arbitrary content. |

Provide a way to use a new type of ordered/unordered lists to be collapsible. Something close to [this](#).

Similar to the page tree, but arbitrary content.

Personal Homepage

| | |
|--------------------|---|
| Name | Personal Homepage |
| Author(s) | Adaptavist.com Limited |
| Priority | High |
| Progress | Almost complete! |
| Status | Final stages of QA |
| Description | A default homepage for every personal space |

Description

Create a much more useful personal space homepage, that display relevant information to the space owner and slightly different information to other users. The page should be made up of a few dashboard portlet style boxes containing various bits of content:

Screenshots

Here's an example of the Facebook homepage, which captures some of what I'm imagining for Confluence. This shows you viewing your own homepage, and the feed of your friends' activities.

The screenshot shows a仿制的Facebook homepage for a user named Amy Weller. At the top, there's a navigation bar with links for 'home', 'search', 'browse', 'invite', 'help', and 'logout'. The main header displays 'Amy Weller's Profile (This is you)' and 'Manchester, NH'. Below the header is a large profile picture of Amy smiling. To the right of the picture, her basic information is listed: Manchester, NH, Harvard '06. A 'Share' button is also present. Below this, a sidebar titled 'Friends' lists 51 friends with small profile pictures and names: Marina Erhan, Shamus Minter, Scott Morrow, Pepe Lopez, Eddie Lin, and David Steinberg. The main content area is titled 'Mini-Feed' and shows a list of recent activity items:

- October 30: Amy posted a link, 10:51am. Description: Anthropologie Hodgepodge Scarf | Outblush. Image: A yellow and red striped scarf on a mannequin. Share button.
- October 27: Amy and Akhill are now friends. 2:59am.
- October 20: Amy wrote on the wall for the group Example Sponsored Group, 3:45pm.
- October 12: Amy and Philip are now friends. 8:21pm.
- October 12: Amy and Scott are now friends. 1:48pm.

Screen grab of a sidebar containing some example widgets from Bubbles plugin is shown to the right

-->

It should be noted that any number of widgets can be added (using simple wiki notation and existing and/or custom macros) and the portals can be shown anywhere, not just space home pages. Any number of portals can be defined and users with sufficient privileges (eg. space admins or site admins) can customise portal instances separately.

Portals can be multi-column and are customisable using CSS so the design shown to the left could easily be achieved. Using Scriptix you could easily create highly bespoke widgets with simple

JavaScript 😊

Viewing my own space

- User's information (link to edit)
- Some descriptive text
- Link to new blog post
- Feed of blogs from spaces/people I'm subscribed to.
- User's recent **bookmarks**
- User's favourite people (w/ stars for unfavouriting)
- User's favourite spaces (w/ stars for unfavouriting)
- Recent changes from favourite people and spaces (filterable)
- User's **calendar**
- User's tags
- Groups the user belongs to

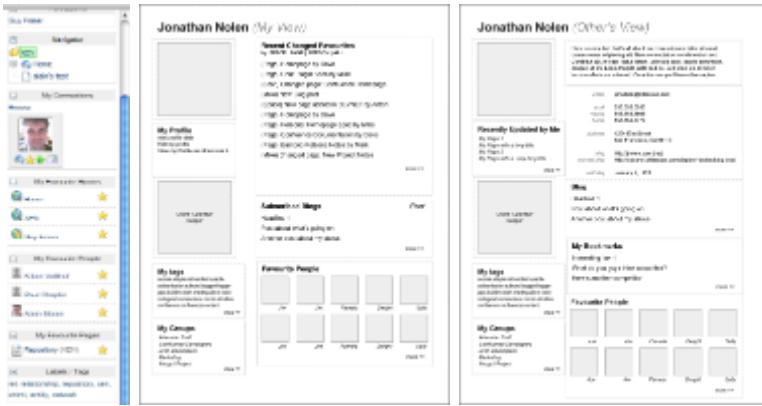
Viewing another's space

- User's information
- Some descriptive text
- User's recent changes
- User's recent blog posts
- User's recent **bookmarks**
- User's favourite people
- User's favourite spaces/pages
- User's **calendar**
- User's tags

- Groups the user belongs to

Screenshots

Here are some mockups of what I'm imagining.



Screen grab of some widgets in Bubbles plugin



Recently added pages

| | |
|-------------|--|
| Name | New pages |
| Priority | High |
| Complexity | low |
| Author(s) | |
| Description | Show recently added pages. No updated pages! |

Description/features

Usage

Technical Implementation

A technical implementation is currently in progress by [o-byte.com](#)!

Other Resources

Show confluence Permissions

| | |
|-------------|--|
| Name | Show permissions |
| Priority | medium |
| Complexity | low |
| Author(s) | |
| Description | Show permissions per user on space pages |

Description/features

Show permissions per user on space pages

Usage

By selecting space and username a table shows one line for each page with the user permission on that page.

eg:

User username:

| Page title | View | Edit | Comment... |
|------------|------|------|------------|
| Page1 | ✓ | ✓ | ✓ |
| Page2 | ✓ | ✗ | ✓ |

Technical Implementation

Other Resources

Space Provisioning Plugin

| | |
|-------------|--|
| Name | Space Provisioning Plugin |
| Priority | |
| Complexity | |
| Author(s) | Geof Corb |
| Description | Create new spaces on-demand with default configuration |

Description/features

This plugin would provide a user with the ability to automatically create a new space, with some default actions taken:

- User becomes a space administrator for the space
- Space name follows some naming convention (configurable system-wide)
- Two groups would be created: [spacename]-admin and [spacename]-public with default permissions applied to each (configurable system-wide)

Usage

As an academic institution, there is great interest among different user populations to have wiki spaces for a variety of purposes-- support classroom activities, student groups, intramural teams, research collaboration, etc. We would very much like to provide a general-purpose Confluence environment in which a user in our community could visit a public space in which this plugin would reside, complete a simple form, and have a space created for them automatically, following defaults dictated through system-wide configuration.

This proposed plugin, coupled with the existing [Invite plugin](#), would provide a powerful way to establish and grow wiki communities.

It would also be necessary to provide space-based user administration to make this effective, ala the [Custom Space User Management plugin](#).

Technical Implementation

Other Resources

Suspended Specs

Advanced Table Macro

| | |
|-----------|----------------------|
| Name | Advanced Table Macro |
| Priority | Medium |
| Status | Released |
| Author(s) | [|

| | |
|--------------------|--|
| Description | A macro that supports advanced table formatting. |
|--------------------|--|

A macro that support advanced table formatting, including text alignment within cells, colspan/rowspan, and sortable column headers.

Might not be necessary to use a macro - wiki notation could be extended to allow these things, see [CONF-3808](#) for ideas of wiki markup and also ideas for various table macros.

Tasklist Improvements

| | |
|--------------------|-------------------------------------|
| Name | TaskList Improvements |
| Author(s) | |
| Priority | Medium |
| Description | Improve the dynamicTaskList plugin. |

Description

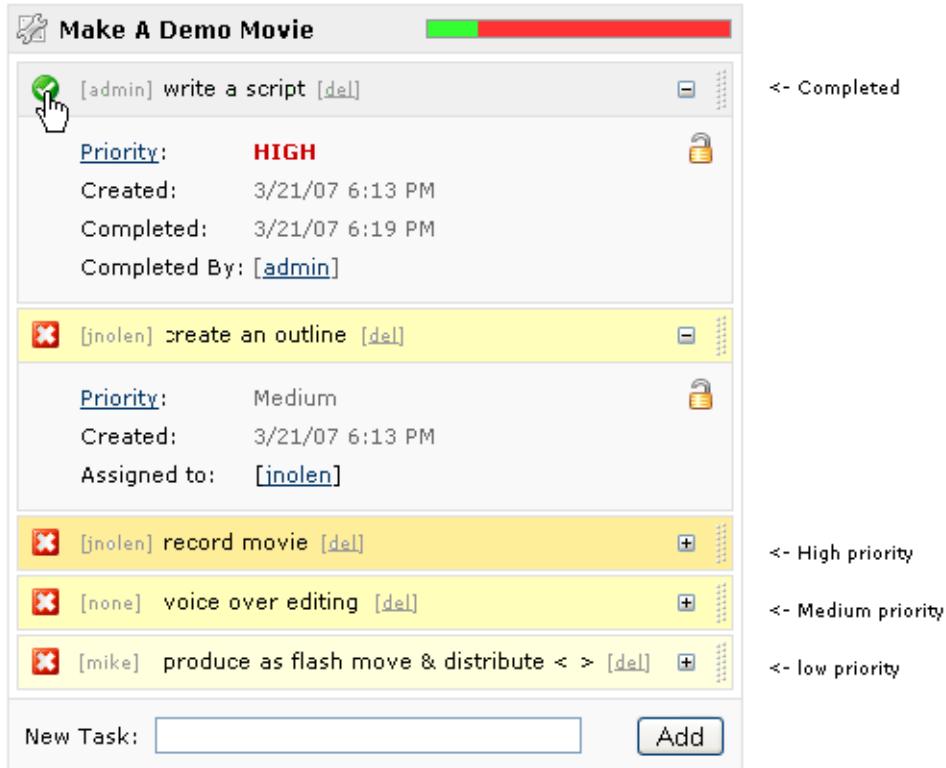
We have two (at least) task list macros, and neither of them are very good. We need to deprecate those two and replace them with a much better solution.

Features

Compatibility with old macro

We need to look for the presence of any old-style data and upgrade it to this new format. So any time the macro renders finds an old-style markup like `dynamictasklist`, it should print a link that says "Upgrade this tasklist". Clicking that link should invoke an action that does the upgrade by extracting the metadata, formatting it properly and inserting it into the page. Then it should redirect back to the original page where the data will appear correctly.

Change UI



Make changes versioned

We'd like to make sure that all changes to a task-list are versioned, and we think the best way to accomplish that is to make sure that the tasklist stores all content inside the page, rather than in page metadata. Whenever someone makes a change to the tasklist using then UI,

the macro makes an ajax request back to the server, and the server needs to write that data into the page in a tabular format in the macro body and then return success to the browser. If the user edits the page though the normal editor, then he is free to make changes to the text of the task list like normal, and the page save will write the change. An example:

```
{dynamictasklist}
|| Task      || Assignee || Reporter || Created   || Priority || Status  || Completed
| Buy Milk  | [~jnolen] | [~scott] | 01/01/07 | 1        | 0       |
| Get Laundry | [~mike] | [~david] | 04/05/07 | 3        | 1       | 04/07/07
{dynamictasklist}
```

Assignee should change to whomever closes the task, if different.

The plugin would have to be able to report errors in the event that the data get screwed up by a manual change. Authors would be able to step back in page history to recover from any data loss.

Allowing wiki markup

We should allow simple wiki-markup (bold, italic, linking, etc.) in the task descriptions. Probably a `SubRenderer` with `RenderMode.INLINE` should do the trick.

Cross-list drag & drop

Allow tasks to be dragged from one list on a page to another.

Fix task id generation

IDs are currently stronger than they were in the Dynamic Task List, but still need some improvement to support a multi-user environment. My current approach may not be compatible with the "write data to page approach," so I'll have to think on that a little yet. I switched the original approach used by the dynamic tasklist from using task descriptions as IDs to using some more or less numeric IDs to simplify task addressing and to avoid some special character issues. It's probably not too much work to find another solution that's friendlier to persisting the data on the page, but there's currently a logical mismatch there, I think.

New Mockup

| Shopping List | |
|--|---|
| Add task: | <input type="text"/> <input type="button" value="Add"/> Sort: |
| Buy Oranges | Jose |
| Priority: <input checked="" type="radio"/> High <input type="radio"/> Medium <input type="radio"/> Low | |
| Assignee: <input type="text" value="Jose"/> | |
| Created: 11 November 2007 | |
| Buy Apples | Janine |
| Priority: <input type="radio"/> High <input checked="" type="radio"/> Medium <input type="radio"/> Low | |
| Assignee: <input type="text" value="Janine"/> | |
| Created: 11 November 2007 | |
| Eat Fruit | Janine |
| Priority: <input type="radio"/> High <input type="radio"/> Medium <input checked="" type="radio"/> Low | |
| Assignee: <input type="text" value="Janine"/> | |
| Created: 11 November 2007 | |
| Buy Vegetables | Jose |
| Buy Meat | Janine |
| Cook Dinner | Janine |
| Plan Dinner | Janine |
| Buy Bananas | Janine |
| Priority: <input type="radio"/> High <input checked="" type="radio"/> Medium <input type="radio"/> Low | |
| Assignee: <input type="text" value="Janine"/> | |
| Created: 11 November 2007 | |
| Completed: 13 November 2007 | |

Later

- Configuration UI

References

- <http://rememberthemilk.com>
 - <http://backpackit.com>
 - <http://stikkit.com>
 - <http://www.tadalist.com>
 - <http://www.blablist.com/>
 - <http://www.zirr.us/>

Tasks

Required

| | | |
|--|--|---|
| Wiki markup in task names | Not yet researched, estimate still at 4 hours until more is known | |
| Versioning on every task list change | Looked at example; not clear yet how to get page/originalPage/pageManager objects | |
| Back port to JDK 1.4 | Remove generics, for loops, autoboxing, compilation testing. Swap XML libs. | |
| Safari DnD rubberbanding | Need to get my hands on a Mac to going to work on this; script.aculo.us may be a problem | 2 |
| Change over the namespace so we upgrade the old tasklist | | |

Other

| | | |
|---|--|---|
| DnD drop guide visualization | The switch to script.aculo.us for DnD is the cause of the loss of the guide; non-trivial | 4 |
| Line width/wrapping parameter | Not too tough, but testing multi-browser UI impact may take some work | 2 |
| Safari button normalization | May not require a mac | |
| Add user picker next to assignee edit box | No analysis done yet, assuming 4 hours | 4 |
| Extract English text to properties file | | 2 |

Task Scheduler

| | |
|-------------|--|
| Name | Task Scheduler |
| Priority | Medium |
| Complexity | High |
| Author(s) | |
| Description | Allow users to schedule task associated with a page. |

Description/features

Allow users to schedule task associated with a page. This may be an extension of the [Tasklist Improvements](#) wish items.

Some requirements are:

- A task shall be associated with a page.
- Each task shall have the following properties:
 - A date that the task is due (for repeating tasks, this is the first date)
 - Repeating parameters (eg, daily, monthly, every 20 days etc.)
 - Person, or people responsible
 - An option to send reminder emails to responsible people at some a defined warning time
 - An option for people to indicate when the task has was last completed
 - An option to send reminder emails when a task has not been completed
 - Metadata and label creation so that reports of task pages can be generated
 - A date (or dates) when the task was previously completed
- It would be nice if the task could synchronise with an iCal calendar

Usage

This macro is aimed at providing a maintenance management facility in Confluence. Some typical usage cases may be:

- Create a backup procedure page and use the {taskscheduler} macro to define when it should be run, who should do it and keep track of when it was done.
- Create a series of pages for machinery that needs to be maintained. Create a {taskscheduler} for each page for its particular maintenance details.

{taskscheduler}

Technical Implementation

The task schedule may have a form that would look something like this (where the second column contains notes about the input fields):

| Field | Notes |
|-------|-------|
| | |

| | |
|--------------------|--|
| Time | |
| Date | |
| Email alert | a number of minutes, hours or days prior. |
| Email recipient(s) | A group or list of individuals |
| Repeat | None, every week, every other week, every month, every year, other... Where 'other..' would be every x days/weeks/months/years until an end date (eg. every 2 days until 12th March 2009) |

Other Resources

Confluence plugins should be able to achieve all of the functionality using a [Trigger Plugin](#)

User Information

| | |
|-------------|---------------------------------------|
| Name | User Information |
| Priority | |
| Complexity | |
| Author(s) | |
| Description | Generalised for anyone not just users |

Description/features

Generalised for anyone not just users.

Usage

We use our wiki to place contact details for lots of people who aren't users of the wiki. It would be useful to generalise this plugin to simply insert contacts wherever.

Technical Implementation

Other Resources

UWC

First, familiarize yourself with the [Universal Wiki Converter](#). This is our new framework for managing wiki-to-wiki conversions. The framework is generic and handles all of the communication with Confluence. The specific conversions are done by sets of regular expressions called "converter packs." We'd like you to write two new converter packs:

Deliverables

MoinMoin UWC converter pack

Resources:

- [Ruby MoinMoin importer](#)
- [Brendan's MoinMoin notes](#)
- [Converter from Dan Woodhams](#)
- [Chariot Solutions converter](#)

MediaWiki UWC converter pack

Resources:

- [Mediawiki Translator](#)
- [Mediawiki Test Data 1](#)

The goals is to obsolete and remove these other translators after the "official" UWC convert pack is ready.

I'm trying to solicit data from our users that we can use as our test cases for this project. When I have them, I'll attach them here:

| Name | Size | Creator | Creation Date | Comment |
|---|---------|----------------|--------------------|---------|
| ZIP Archive WicketWiki_mysql41_javaguy_MW171.sql... | 2.75 MB | Jonathan Nolen | Sep 25, 2006 12:20 | |

Chariot Solutions converter

UWC MoinMoin to Confluence Status

| Notation | MoinMoin | Confluence | Ported to UWC | Notes |
|--------------------|---|---|---------------|---|
| Table of Contents | [TableOfContents] (macro) | <ul style="list-style-type: none">• UWC MoinMoin to Confluence Status<ul style="list-style-type: none">• Heading1• Heading2• Heading3• Heading4• Heading5 | Yes | toc macro needs to be installed on Confluence |
| Heading - Level 1 | =Heading1= | Heading1 | Yes | |
| Heading Level 2 | ==Heading2== | Heading2 | Yes | |
| Heading Level 3 | ====Heading3==== | Heading3 | Yes | |
| Heading Level 4 | =====Heading4===== | Heading4 | Yes | |
| Heading Level 5 | =====Heading5===== | Heading5 | Yes | |
| | | | | |
| | | | | |
| Text - Italic | "Italic" | <i>emphasis</i> | Yes | |
| Text - Bold | ""Bold"" | Bold | Yes | |
| Text - Monospace | `monospace` | monospace | Yes | |
| Text - Code |  |  | Yes | |
| Text - Underline | <u>Underline</u> | <u>Underline</u> | Yes | |
| Text - SuperScript | SuperScript | SuperScript | Yes | |
| Text - SubScript | ,,SubScript | SubScript | Yes | |
| Text - Smaller | <small>smaller</small> | | | No Confluence Notation |
| Text - Larger | <small>larger</small> | | | No Confluence Notation |
| Text - Stroke | (Stroke) | Stroke | Yes | |

| | | | | |
|-------------------------------|---|--|-----|--|
| Text - Mixing Italic and Bold | <p>""Mix" at the beginning"</p> <p>""Mix" at the beginning"</p> <p>"Mix at the "end""</p> <p>"Mix at the "end""</p> | | | |
| Syntax Highlighting | <pre>{{ Unknown macro: {#!Python} }} </pre> | | | |
| Smileys | | | | |
| Internal Links | | | | |
| | WikiName | [WikiName] | Yes | |
| | ["free link"] | [Free Link] | Yes | |
| | /SubPage or ["/Sub Page"] | | | |
| External Links | | | | |
| | http://example.net | http://example.net | Yes | |
| Blockquote | | | | |
| Bullet List | <ul style="list-style-type: none"> • item 1 • item 2 • item 2.1 | <ul style="list-style-type: none"> • item 1 • item 2 • item 2.1 | Yes | |
| Numbered List | 1. item1 | 1. item | Yes | |
| Descriptions and Definitions | Term::Description Label::Definition | | | |
| Tables | | | | |
| Tables - General Layout | | | | |
| Rules | <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> | | Yes | |
| Macros | | | | |
| Attachments | | | | |
| | | | | |

Converter from Dan Woodhams

We have written a MoinMoin wiki to confluence converter; we started out with the [ruby converter](#) and rewrote it.

It is now considerably enhanced and can deal with:

- Attachments
- Page hierarchy
- Excluded pages
- Links
- Logging (log file can be passed back to the application to rerun/fix pages that had errors)

Please feel free to use/distribute it. I have converted five 1000+ page wikis with a 95% success rate.

It's still not perfect and never can be because our source moinmoin pages have errors. However, it would be better if we used a MoinMoin parser rather than regexes.

Anyhow hope you find it useful.

Thanks,
Daniel Woodhams

P.S. The configurations are set in a convertWiki.rb here is an excerpt:

```
# Constructor
importer = MoinMoin::Importer.new(
  'confluence.host.com',           # Confluence host
  'moinmoin.host.com',            # MoinMoin host
  'ateam',                         # wiki's root path
  false,                            # use ssl?
  'MOIN_ID=1654515219.49.23543',  # cookie to access moinmoin
  'myuser',                         # Confluence user
  'myuser',                         # Confluence password
  'msn',                            # destination confluence space
  12706                            # id of home page to attach all content
)
```

| Name | Size | Creator | Creation Date | Comment |
|--------------------------------------|-------|----------------|--------------------|---------|
| ZIP Archive MoinMoinToConfluence.zip | 15 kB | Jonathan Nolen | Aug 21, 2006 12:43 | |

UWC Converter Packs

| | |
|-------------|--|
| Name | UWC Converter Packs |
| Author(s) | |
| Priority | High |
| Description | Write more converter packs for the UWC |

See [UWC Voting page](#)

Voting and Vote tabulation

| | |
|-------------|---------------------------------------|
| Name | Voting and Vote Tabulation |
| Author(s) | |
| Priority | Medium |
| Description | Allow users to vote for good content. |

Some work has already been done in the macro described at the [vote macro](#) page.

Wiki Markup Editor AutoCompletion

| | |
|-------------|--|
| Name | Wiki Markup Editor AutoCompletion |
| Author(s) | |
| Priority | Low |
| Description | auto complete macros in the Wiki Markup Editor |

It would sweet if the Wiki Markup Editor had a form of Autocompletion. Just like an IDE, that enables you complete macros.

If you type a {, a list would appear with all choices. Once you start typing, the list will narrow down and pressing Enter would fill in the rest of the current one selected.

For an example, check out the address autocompletion in Gmail.

Would help improve speed, accuracy and more importantly, stop people from always having to refer to the help docs to try and find a certain macro....

The more advanced version, would help you insert the macro parameters.

Wiki Markup Editor Toolbar

| | |
|-------------|--|
| Name | Wiki Markup Editor Toolbar |
| Author(s) | |
| Priority | Low |
| Description | Toolbar for inserting wikimarkup in RTE. |

The Rich Text editor will certainly be a valuable feature when all the bugs will have disappeared. Even so, many users prefer to use Wiki Markup. Most wiki software already have a javascript toolbar above the textbox that inserts wiki markup. It would be a very nice feature to have a (configurable) toolbar, one that either adds more buttons to the existing ones (so far: Insert Image, Insert Link), or a new toolbar altogether.

The javascript and html code is already out there and open source (see dokuwiki and others). It's just a matter of implementing it.

Example from [Dokuwiki](#):



Word, Open Office and Excel Importing

| | |
|-------------|---|
| Name | Word, Open Office and Excel Importing |
| Author(s) | |
| Priority | High |
| Description | Create or replace a page by uploading a Microsoft Word and OpenOffice document. |

A plugin that would allow a user to create or replace a page by uploading a Microsoft Word and OpenOffice document. It would function from the user's standpoint the same as uploading a document. [Jot](#) has implemented this in a sensible way. It would make reasonable approximations of the Word formatting. It might be useful to make the importer aware of links or link syntax so that it could auto-create links to other existing Confluence pages. See Jakarta POI: <http://jakarta.apache.org/poi/index.html>

Extra: I was thinking it would be cool if you could attach a Excel doc - available now - [Excel Plugin](#), then use a macro to render the sheet as a table, as well as provide really portable easy to use MS Word to Confluence importer - initial alpha available - [Word Plugin](#).

Confluence Gantt Charts

It could possibly work something like this.

```
{ganttchart:startdate=010105|enddate=030305}
{row1:type=milestone|label=specification ready|startdate=150105|enddate=150105}
{row2:type=duration|label=alpha functionality|startdate=150105|enddate=290105|pre=(row1,e-s)}
{row3:type=duration|label=gui design|startdate=150105|enddate=290105|pre=(row1,e-s)(row2,e-s)}
{ganttchart}
```

Gantt charts can get pretty complicated, so it may be beyond reason to expect to codify enough of it in a macro language. But maybe there is a small enough subset of functionality that will get the 80%.

Another option would be to build something like the excel plugin that just displays a nicely formatted chart built out of the data in a MS Project file. See this [java library](#) for MS Project exchange, similar to the one for MS Office.

We may be able to nick some code from here: [JIRA GANTT Chart Report Plugin](#) and [JFreeChart](#) can do GANTT charts as well.

Plugin to expose Confluence page ordering to the SOAP API

| | |
|--------------------|---|
| Name | Expose page ordering to the SOAP API |
| Priority | High |
| Complexity | Medium |
| Author(s) | |
| Description | Expose Confluence page ordering to the SOAP API |

Description/features

The plugin will expose Confluence page ordering to the SOAP API. Currently, not all Confluence functionality is exposed to the remote API. In particular, you cannot set the page order via the remote API. By "page order" I mean the sequential order of pages relative to their siblings. See [CONF-14328](#).

Usage

This API functionality would be useful specifically to the [DITA2wiki](#) open source project on SourceForge. The **DITA2Confluence** tool is part of DITA2wiki. It is a toolkit that you can use to publish DITA content (maps and topics) to Confluence. Typically, such a tool would be used regularly to publish documentation from the source XML to the wiki. The content is authored in DITA XML, as a single source tool. It is published on a regular basis to the wiki to make use of the wiki's collaboration and web publishing features.

Currently, DITA2Confluence cannot set the order of the pages that it creates. One way around this problem is to write a plugin that creates a new remote API call to give DITA2Confluence the capability it needs. You can mix and match the Confluence-provided remote APIs with those provided by plugins.

Other Resources

- [CONF-14328](#)
- [DITA2wiki](#) project on Sourceforge.
- [DITA Meets Wiki – Output DITA to Wikitext](#), a blog post by Anne Gentle
- [Playing with DITA2Confluence](#), a blog post by Sarah Maddox

Document Staging

| | |
|------------------|------------------------------|
| Name | Documentation Staging |
| Author(s) | Charles Miller |
| Priority | High |
| Progress | In development, help welcome |
| Status | pre-alpha |

| | |
|--------------------|---|
| Description | Support for staging documentation in one space, then publishing it into another |
|--------------------|---|

Support a process where documentation can be collaborated on in one space, and then published into another space when ready.

Status

A reasonably feature complete demo of this plugin has been written:

- svn repository
- TODO.txt

The plugin is [apache licensed](#), so feel free to send [me](#) patches if you want to see it finished sooner.

Use Cases

- A space administrator configures that one space (the Staging space) should publish changes to another space (the Public space). This person must also be an administrator of the Public space for the configuration to be successful.
- A space administrator configures which of the following have permission to publish changes:
 - All Administrators of the staging space
 - Everyone with "edit page" permission in the staging space
 - Members of one or more specific groups
- Viewers of a page in the staging space will see a visible indicator of whether it has never been published, whether the page has been modified since it was last published, or whether the page is 'up to date'.
- Anyone with permissions to publish changes can, while viewing a page, trigger that page to be published. This will copy over the current version of the page, its metadata (including labels), the current versions of the page's attachments, and other related data into the Public space, overwriting any page already there with the same name.
- Anyone who can label a page can add a "nopublish" label, which will prevent the page from being published, either individually or as part of a bulk publish
- Anyone with permissions to view the staging space can see a space-wide report shows pages that have been modified since they were last published
- Anyone with permissions to publish pages can trigger a "mass publish", which will cause all pages that have been modified since their last publish (or that have not been published at all) to be published.

Implementation details

- The publish status indicator / publish button can be the same thing - put it in the same set of web items where we currently have the favourite and watch icons.
- Store the version number of the last version of the page to be published in the page properties
- Use search extractors to add the publish status of a page to the Lucene index, this will make searching for unpublished/out of date pages a lot faster
- Bulk publish operation will need to be a long-running task of some kind, not sure how pluggable this is.
- Bulk publish should be done as a series of atomic single-page publishes, each in its own transaction.
- While not designed specifically for this purpose, you'll be able to copy the current contents of a space into another by configuring a new, empty Public space, then hitting "mass publish"

Who can see this page?

Add a tool menu item that will display all the users and groups that can see the page. With space permissions and hierarchical page permission, it can be very difficult to determine who is authorized to see (or edit) a page. Administrators often get complaints that someone cannot see a page even though they have been explicitly authorized to the page only to find out that somewhere higher up they are not permitted. Need a easy way for a normal user to see who can see a page. The [info](#) page just doesn't do that in a way that is understandable.

In Process

Set Field Value Using Regex Plugin

| | |
|----------|------------------------------------|
| Name | Set Field Value Using Regex Plugin |
| Priority | High |

| | |
|--------------------|---|
| Complexity | Low |
| Author(s) | |
| Description | Provide a post-function allowing the value of one field to be set by applying a regular expression to another field's value |

Description/features

Provide a post-function allowing the value of one field to be set by applying a regular expression to another field's value.

On transition, the post-function should run after all edited fields have been updated. Multiple instances of the post-function would run in order. If a regular expression returns multiple matches, maybe it would select only the first match? Or maybe provide the user with an option of first match or all matches (where the target field could accommodate a blob?).

Usage

Simply select the source field from one dropdown, select the target field from a second dropdown, and provider the regular expression in a text field.

Technical Implementation

Other Resources

Theory of Constraints

| | |
|-------------------|-----------------------|
| Name | Theory of Constraints |
| Priority | Minor |
| Complexity | unknown |
| Author(s) | |

Description/features

Theory of Constraints (TOC) is an overall management philosophy introduced by Dr. Eliyahu M. Goldratt that is geared to help organizations continually achieve their goals. Some of his ideas can be implemented in a JIRA plugin by setting limits on issues being in a certain status for a project, or across several projects.

Usage

Some kind of configuration that will enable to simplify limit specification for how many issues can be in a certain status, or in a group of statuses.

Technical Implementation

JIRA plugin that will prevent users from transitioning an issue from one status to another if the limits do not allow it.

Other Resources

- Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results
- Theory of Constraints
- Critical Chain Project Management

Better Recent Changes Macro

| | |
|-------------|-----------------------------|
| Name | Better Recent Changes Macro |
|-------------|-----------------------------|

| | |
|--------------------|---|
| Author(s) | |
| Priority | High |
| Description | Improve to show more detail and offer more options. |

The Recent Changes macro (which is used on the front page of Confluence) is a fairly simplistic view of what's been going on in Confluence. We would like to improve it to be more usable and convey more information.

Show more data:

We should show the following columns of data about each change:

- Title
- Change Comment
- % changed, or +/- lines
- User
- Date of change
- Version of page

New Features

- Allow the page author to filter by user
- Allow page author to specific which columns to show
- Allow page author to specify how many changes to retrieve
- Allow page author to show all changes in last X days
- Add a direct link to the equivilant RSS feed at the top of the list
- Add "Fetch more" link that retrieves another set of X changes forever.

Resources

- [Current Macro](#)
- [Version History Table](#)

JIRA Issues Maps

| | |
|--------------------|--|
| Name | JIRa Issues Map |
| Author(s) | |
| Priority | Medium |
| Description | Show an ajaxy map of all related issues. |

I think we could take this even a step further and implement a related issues map. Hence a quick glance on this map can give you an idea how many issues are related, incorporated, duplicated - kind of like a MindMap yet we could call it JIM (Jira Issues Map). (source)

I'd probably recommend creating an issue tab which displays the link-graph for all the issues connected to the one currently being viewed. You'll have to be smart about the UI, as that is potentially a lot of information to pack on the page. I'd recommend making it filterable by type of link so the graph doesn't get too huge.

Offline Confluence Editor

| | |
|--------------------|--|
| Name | Offline Confluence Editor |
| Author(s) | |
| Priority | High |
| Description | Provide an offline client that would allow a user to add/edit pages/news/comments when disconnected. |

The way I would see this working would be an off line application which would allow you to take a synchronized copy of the wiki spaces with a localized instance. Any work that was then done off line, would then be synchronized with the hosted, server instance. During the synchronization, any conflicts would be (where possible) merged or the user asked for an action (for example, keep/lose updates). In

addition, the wiki itself would require additional options to allow the admins to specify which wiki's could be taken off line to ensure that for more sensitive information, off line copies could not be taken.

See also [JIRA](#) entries for Off line client

See also [TimTam](#)

Calendar Improvements

| | |
|-------------|---|
| Name | Calendar Plugin Improvements |
| Author(s) | None - you can volunteer to contribute |
| Priority | High |
| Progress | No progress |
| Status | Awaiting developer resources |
| Description | Usability improvements to the Calendar Plugin |

Description

Calendaring is an important group activity that we need to support within the wiki. Basically, I want our calendar plugin to be as good and polished and joyful to use as the best of web calendars current out there: [Google Calendar](#), [30Boxes](#), [Backpack calendar](#). We already have a strong backend, so we need to focus on polish, aesthetics and usability (particular speed). We need to make increasing use of dynamic HTML and ajax to achieve a desktop-like user experience.

Things I'd like to tackle on Calendar, in priority order:

- BUG: When you click on day view, it goes to selected day -1, instead of the selected day.
- Try to keep the calendar width from jumping around when the ajax stuff happens.
- Change "add event" to happen all in one screen. The "choose a calendar" step should just be a dropdown stuck to the last selected calendar. (You'll need to use Ajax to reload the calendar fields if they change the calendar type.)
- Have an option for 24hr or 12hr time. Use the most concise format for time and date ('6pm') where possible to save space.
- Left align the text in the boxes.
- There will be some additional CSS/HTML improvements – designs to be uploaded.
- Checking/unchecking specific sub-calendars to show/hide their events
- More colour options
- More event details in the week/day views.
- Performance/caching
- FEATURE: Add an event in natural language: ('dinner next tuesday').
- Figure out how to do multi-day view that span the boxes. The rounded corners are nice too.
- FEATURE: Add drag-and-drop moving from day to day. (Scriptaculous or YUI should help with this).
- Add a 'Day View' in the top right corner listing the details of the selected days events, in order.
- FEATURE: Allow calendars to have a specific time-zone set for each one.
- FEATURE: A personal calendar for each user that has a personal space. Put it in a space tab and make it available via a macro.
- FEATURE: A macro that display the personal calendar in a compact format.
- As many [bugs](#) as possible.
- FEATURE: Allow iCal users to 'publish' their calendars into Confluence (with or without details) using [CalDAV?](#)
- FEATURE: Create a Space Tab that lists all the calendars in a space.
- FEATURE: hCal (microformats) format for calendar items.

References

- <http://calendar.google.com>
- <http://30boxes.com>
- <http://spongecell.com>
- <http://calendarhub.com>

Datatable

| | |
|-------------|--|
| Name | Datatable |
| Author(s) | Mike Cannon-Brookes, Jens Schumacher |
| Priority | High |
| Description | Allows for creating a dynamic table of data with forms for entry, predefined content types and configurable display. |

Example

```
{dataform:mode=form|inline}
Name
URL (hidden)
Company (link URL)
Organisation type (select Non-Profit, University, Corporation)
Yes / No (checkbox)
{dataform}
```

Mode determines whether the entry form is inline (ie the last row in the table) or a form (ie a shown/hidden form above the table). Default is form.

Options are

| Option | Description |
|----------------|--|
| hidden | field is not shown in the result table - must be first in the list (ie "(hidden textarea)" is valid, "(textarea hidden)" is not) |
| link A | field is an ahref to field A. if A is not entered, hyperlink the field itself |
| checkbox | field is entered as a checkbox |
| select A, B, C | select list with options A, B, C |
| textarea | field is entered in a text area |

Other requirements

- All entries in the dataform should be searchable.
- All data should be revision controlled.

HTML WysiWyg Editor like Google Docs

Google Docs (writely) WysiWyg Editor is HTML Editor .

There is no reason a typical user who knows nothing about wiki markup has to see that .

related Reference - [click\](#)

Live log viewer

| | |
|-------------|-------------------------|
| Name | Live Log Viewer |
| Author(s) | |
| Priority | Low |
| Description | Show app logs in Web UI |

A page in the admin section that will use ajax to display the contents of the app's log.

New Themes

We're interested in developing two new themes:

Simple Theme

This theme is targeted for the next version of Confluence. We're after Simple, content focussed theme with a clear content pane and a right hand 'utility' pane ([example page](#)). Take the simple, clean look of Stikipad and some of the Ajax utilities that SocialText has. Also see Facebook's new redesign

For more details, see the [Simple Theme](#) page.

Bamboo Theme

We're preparing to release the first version of our next project, Bamboo. We're planning to build the entire website in Confluence. Develop a theme that matches the design of the main [Atlassian website](#)

For more details, see the [Bamboo Theme](#) page.

Resources

- Clickr Theme shows good CSS examples.

Bamboo Theme

We're preparing to release the first version of our next project, Bamboo. We're planning to build the entire website in Confluence. Develop a theme that matches the design of the main [Atlassian website](#)

Pages should look like this: <http://www.atlassian.com/about/>. The navigation should be pulled from the "navigation" page, the same as in the left-nav theme. All the page operations should be at the bottom, and only visible if you are logged in and have edit permissions.

More Themes

| | |
|-------------|-------------|
| Name | More Themes |
| Author(s) | |
| Priority | High |
| Description | |

Simple Theme

This theme is targeted for the next version of Confluence. We're after Simple, content focussed theme with a clear content pane and a right hand 'utility' pane ([example page](#)). Take the simple, clean look of Stikipad and some of the Ajax utilities that SocialText has. Also see Facebook's new redesign.

The theme has been committed to the atlassian subversion repository as 'simpletheme-plugin'. See <https://svn.atlassian.com/svn/public/contrib/confluence/simpletheme-plugin>

Background

Mike and I saw a demo of a new web-app at a conference we were at last week. We were both impressed, and mike leaned over to me and said, "That's it! That's what I want the simple theme to look like!" So, check it out, and hopefully that will give you some more direction on the UI.

<http://wufoo.com> - Main page

<http://wufoo.com/tour/>

<http://wufoo.com/examples/> - See particularly the examples here. That's what we were looking at when Mike made his exclamation.

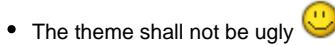
Other mockup versions of the page can be seen at

- Simple Theme Mockup 1
- Simple Theme Mockup 2

A prototype (under construction) can be seen at <http://wiki.saikore.com/display/theme/Home> . You will need to sign in to the site to see it.

For the right hand pane, here are some things that could / should be included, along with their headings. These are a series of sections. If implemented truly well, there would be a configuration screen for the theme *per space* and globally, which allow people to turn on / off which sections they wish to be displayed. Also ideally, the loading of the values in these sections should be Ajaxed, with a cookie to remember which sections are open / closed. We'll need to think about what the defaults should be.

General Requirements



- The theme shall not be ugly
- The background of the area outside the page text shall be selected from the colour schemes available with the theme.
- The top search, logout etc should move to the bottom of the right bar. All the icons too - move 'em into the right bar.
- The breadcrumbs should be the only thing above the title.
- Content shall not be 'fluid' but fixed. 750px of content, 150px of sidebar.

Link options

- **Contribute** (these items should be at the top, highlighted somehow more than the others)
 - Start New Page
 - Edit Page
 - Comment on Page
 - Attach File
- **Navigation**
 - An included page (optionally) which links to other pages. This should be configurable as well. Perhaps you can add multiple sections like this? Could they all just be like this?
- **Page Information**
 - Last edit by "Bob" on "date" [view change](#), [view all changes](#)
- **Page Contents**
 - Attachments (show this link if there are any attachments)
 - *Page Headings* - These headings are pulled from the page and hyperlinked. This is a neat trick I saw Accenture do. Uses the same idea as the {toc} macro I think.
 - Comments (show this link if there are any comments)
- **Page Options**
 - Copy Tiny Url
 - Printable View
 - Export to PDF, Word
 - Add as Favourite
 - Start Watching
- **Browse Space**
 - Standard browse space options here - pages, news etc
- **Recent Changes**
 - Show the recent changes in this space
- **History**
 - Show your personal page viewing history
- **My Favourites**
 - Show your favourite pages and spaces, possibly in two tabs.
- **Incoming Links**
 - Show pages which link to this page, possibly for a second tab for external referrers if there are any.

Table of possible links and their parameters

| Name | Group | Position | Default visibility | Notes |
|---------------------------|-------------------|----------|--------------------|--|
| edit page | Contribute | Top | Visible | |
| add page | Contribute | | | |
| add comment | Contribute | | | |
| attach file | Contribute | | | |
| Navigation | Navigation | | Visible | |
| show attachments | Page Contents | | Folded | This is the same as 'attach file' and hence redundant. |
| Table of contents | Table of Contents | | | Will not be implemented in the first version. |
| show comments | Page Contents | | | |
| copy tiny URL | Page Options | | | |
| print view | Page Options | | | |
| export to pdf | Page Options | | | |
| export to word | Page Options | | | |
| add favourite | Page Options | | | |
| watch page | Page Options | | | |
| pages (Space) | Browse Space | | Folded | |
| Space labels (Space) | Browse Space | | | |
| Space attachments (Space) | Browse Space | | | |

| | | | | |
|-------------------------|----------------|---------|--------------------|--|
| Mail (Space) | Browse Space | | | |
| News (Space) | Browse Space | | | |
| Advanced (Space) | Browse Space | | | |
| Space Admin (Space) | Browse Space | | | |
| History (user) | Account | Folded | Opens a pop up | |
| Preferences | Account | | | |
| Administration (user) | Account | | | |
| Logout/login/sign up | Account | | | |
| My Favourites | Account | | | |
| All changes (page) | Recent Changes | Folded | | |
| Recent changes (space) | Recent Changes | | | |
| Recent changes (global) | Recent Changes | | | |
| Last change (page) | Recent Changes | | | |
| | | | | |
| Breadcrumbs | Breadcrumbs | Visible | | |
| View Page | None? | | Refreshes the page | |
| Page Info | Page Contents | | | |
| Incoming links | Page Contents | Folded | | |
| Search | Search | Visible | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Simple Theme Mockup 1

Well, here's a version of that theme you used as an example thrown together in about 45 mins (had to go searching for some macro params) using Adaptavist Theme Builder 2.0:

The screenshot shows a web browser window with a single tab open to a page titled "Discovery Sessions". The page content is as follows:

Discovery Sessions

Hello, Discovery Session facilitators and conference participants.

All the results from Discovery Sessions will be captured here.

Another heading

(headings appear in sidebar)

Session facilitators, please create a page (or more) for your session and document what came out of your session so the rest of us may benefit from learning it.

You don't need to log in - just click 'Edit page' at the bottom of this page.

Session Number Session Title in CamelCase

DSn How To Create Wiki Page

.... etc...

You wanted a [Navigation](#) page 😊

tags: some, tags, and, stuff

Revised on 25 August 2006 by admin

[Add Comment](#)

Edit page Attachments Back in Time (5 revisions) See changes Theme

All rights reserved by respective content owners. Grab our RSS Feed or export to PDF or Word
Adaptavist Builder powered by Confluence

The sidebar on the right contains the following menu items:

- New Page
- Discuss
- Recent Activity
- Account
- Logout

Below the menu is a search bar with a "Search" button. To the right of the search bar is the text "navigation stuff goes here 😊". Further down the sidebar are links to "Home", "Discovery Sessions", "Another heading", and "Navigation".

I didn't get round to adding the links to printable/tiny url/etc (all easy to do without {menulink} macro 😊) because I'm eager to get on and do the Atlassian theme!

@ Jonathan: Log in to the Builder 2 test site (test.w-n.a.c, same place as repo v2) and you'll see the Simple space on the dashboard.

Simple Theme Mockup 2

Test Page JN

test test test
 test test test



Added by Jonathan Nolen, last edited by Jonathan Nolen on Oct 06, 2006

Edit

[Custom Navigation](#)
[My Page 1](#)
[Another Page](#)
[Custom Page 3](#)
[Somewhere else](#)
[Status](#)
[Contact](#)
[About](#)

Actions
[Edit Page](#)
[Add Page](#)
[Attachments](#)
[Page Info](#)
[Page History](#)
[Space Info](#)
[Watch](#)
[Email](#)
[Export](#)
[Print](#)

Attachments

- Picture2.png 71 kb Jonathan Nolen Oct 06, 2006
- Picture1.png 85 kb Jonathan Nolen Oct 06, 2006

[Edit](#) | [Remove](#)

[Edit](#) | [Remove](#)

Labels

[test](#), [2](#), [3](#), [4](#), [test3](#) [EDIT](#)

Children

[Hide Children](#) | [View in hierarchy](#) | [Add Child Page](#)

[Test Page JN Child \(theme\)](#)

Comments



This is a test comment

Posted by Jonathan Nolen at Oct 06, 2006 22:16 | [Permalink](#)



So is this

Posted by Jonathan Nolen at Oct 06, 2006 22:17 | [Permalink](#)

[Add Comment](#)

NOTES:

50px hard margin on both sides
 150px menu on right
 liquid content area that expands/contracts as necessary
 line up all text vertically
 Page content is contained in white box, extra content outside
 Navigation modules can be added by configuration
 Edit button should probably be a graphic
 Attachments/labels/children are in arbitrary order

EVALUATION LICENSE - Are you enjoying Confluence? Please consider [purchasing it](#) today.

Powered by Atlassian Confluence, the [Enterprise Wiki](#). (Version: 2.2.9 Build:#527 Sep 07, 2006) - [Bug/feature request](#) - [Contact Administrators](#)

Standards Compliant Theme

| | |
|-------------|---|
| Name | Standards Compliant Theme |
| Author(s) | |
| Priority | High |
| Description | A new theme that is web-standards-compliant |

Description

Create a new Confluence theme that looks exactly like Confluence does now, but enforces **strict** separation of content, presentation, and interaction design. (That is, HTML/CSS/Javascript.) Building a theme like this would give future theme designers a better foundation and greater flexibility when creating new themes for Confluence.

References

Things You Wish Were Pluggable

| Name | Description |
|-------------------------------------|-------------|
| Make Services in JIRA full plugins | |
| Make Listeners in JIRA full plugins | |

| | |
|--|--|
| Plugin-ize the Seraph authenticator | Both products |
| Plugin-ize the Confluence Startup Sequence | To allow for custom configuration for OEMs |
| AJAX Plugin Points | Atlassian - would it be possible to enhance the plugin/macro support that you could properly implement AJAX'd macros using the same ajax library Confluence uses internally? A "dwr" plugin type would be ideal. |
| Plugin-ize the xwork Interceptors | Allow custom Interceptors to be injected into the default stack: see http://jira.atlassian.com/browse/CONF-4570 |
| User object customization | Allow the Confluence administrator to add custom attributes to Confluence user objects. I see custom attributes as being much like custom fields in Jira, with a field type (select list, text field, multi-select, radio button), label, read/write permissions -- admin only, owner, public?.... Of course once the user objects potentially have custom fields, then we need to be able to base group membership on attributes and etc. My use case for this comes from using Confluence as a content management system for a collaboration community. I would like to categorize community members (teacher/parent/student/administrator) and have that categorization persist via their profile. Some categorizations would be self-selectable (include me in community email notifications) and some would be selectable only by the administrator (teacher/parent/student/administrator). |
| AttachmentManager | The subversion attachment manager has to do some nasty spring hacks in order to insert itself into the delegating attachment system. Would be much better if this was pluggable. Also the delegator should delegate based on space key. Even better would be allowing plugins to override confluence components without having to hack applicationContext.xml . I guess this is pretty tricky seeing as they are already wired together - every component would have to be proxied (yuck). |

Plugin System Improvements

Tasks to improve the plugin system

Confluence

Errors were reported by the JIRA trusted connection.

- APP_UNKNOWN; Unknown Application: {0}; ["confluence:4557196"]

| JIRA Issues (25 issues) | | |
|-------------------------|------------|---|
| Type | Key | Summary |
| | CONF-23139 | OpenSearch Plugin Modifications |
| | CONF-23011 | Field order on Sharepoint search configuration |
| | CONF-22570 | Number of daily users accessing Confluence |
| | CONF-22535 | Prevent custom images appearing blurry in PDF export |
| | CONF-22493 | Allow plugins to define their own caches' attributes |
| | CONF-22368 | Plugin upgrade tasks should also be executed after data is restored |
| | CONF-22336 | Gliffy diagram edits send out many notifications |
| | CONF-21047 | With the 'Plugin Repository' (Confluence <=3.3) it was possible to downgrade a plugin. Now (Confluence 3.4) with the 'Plugins' this feature seems to be lost? |
| | CONF-21003 | Recently updated list in the dashboard should not show raw wiki markup |
| | CONF-20116 | BandanaEditor / SAL API setValue() problems? |
| | CONF-19699 | Prevent bundled plugins from being upgraded |
| | CONF-19139 | There should be a way to list installed non-Atlassian plugins |
| | CONF-18989 | Retain plugin installations when backing up and restoring Confluence to a different server or standalone computer |
| | CONF-17921 | It should be a requirement that names listed in Plugin Repository for particular plugins be the same name as what shows up on the plugins screen |
| | CONF-17750 | Preserve previous plugin versions in the database with timestamp for easy troubleshooting and rollback |
| | | Plugin repository shows non-system core plugins as "Unknown" and "Unsupported" although they are supported by |

| | | |
|--|------------|---|
| | CONF-17323 | atlassian |
| | CONF-17109 | UserMacroLibrary is not available to Plugins v2 |
| | CONF-16839 | Resources should be exposed to plugins that don't have the WebAppClassLoader |
| | CONF-16696 | Application Chat Client |
| | CONF-16682 | Remove 'Plugins' Admin Link |
| | CONF-16668 | Skill Management Functionality |
| | CONF-16628 | Calendar: Ability to report on Events |
| | CONF-16567 | Provide user message for existing plugins when uploading |
| | CONF-16515 | Allow Confluence to embed Prezi presentation |
| | CONF-16046 | Add Community Bubbles to the list of plugins supported by the Enterprise Hosted version of confluence |

JIRA

Errors were reported by the JIRA trusted connection.

- APP_UNKNOWN; Unknown Application: {0}; ["confluence:4557196"]

| JIRA Issues (74 issues) | | |
|-------------------------|-----------|--|
| Type | Key | Summary |
| | JRA-25525 | Get plugin's module by Universal Plugin Manager error. |
| | JRA-25479 | PluginVersion tag missing from export xml |
| | JRA-25363 | Errors during ActiveObjects restore prevent JIRA restore |
| | JRA-25362 | ActiveObjects cannot encode value for unsupported column type "ntext" on MS SQL Server 2008 |
| | JRA-25348 | Upgrade task for Select Custom fields in upgrade to JIRA 4.4 wipes out customer data. |
| | JRA-25252 | Render velocity error in template from JIRA Calendar plugin |
| | JRA-25249 | Error when try to invoke Atlassian Support Tools |
| | JRA-25171 | JIRA logs a warning for every ActiveObjects-created table on startup |
| | JRA-25023 | Make Velocity files scale better for plugins |
| | JRA-24964 | Document the effects of jira.dev.mode |
| | JRA-24942 | Administration Search does not work for freshly installed plugin - JIRA restart required |
| | JRA-24818 | Initial startup of JIRA will log an "ArrayIndexOutOfBoundsException" exception. |
| | JRA-24215 | AJS.InlineDialog is closed when user clicks inside the dialog |
| | JRA-23720 | webwork1 "magic" for i18n is broken for v2 plugins |
| | JRA-22613 | Cascading select doesn't work in firefox and ie8 |
| | JRA-22151 | Web Item Link Elements: absolute attribute does not work |
| | JRA-22119 | JIRA Plugin Module(s) to enable more control over attachments on issues |
| | JRA-21834 | Misleading Error Message in case of wrongly configured wf function plugin: com.atlassian.util.concurrent.LazyReference\$InitializationException: java.lang.NullPointerException |
| | JRA-21565 | Field IDs incorrectly using colons |
| | JRA-21430 | v2 JIRA Service Plugins cannot reference custom <values class= classes (java.lang.ClassNotFoundException) |
| | JRA-21259 | Reporter not set from IssueInputParameters |
| | JRA-21034 | Allow for different CAPTCHA providers |
| | JRA-21010 | Keyboard Shortcuts plugin causing problems in Tempo plugin |
| | JRA-20646 | JiraWebAction uses inappropriate class loader to locate resource bundles |

| | | |
|--|-----------|--|
| | JRA-20574 | Document a way for plugin developers to add user/group pickers in their UI |
| | JRA-20537 | Plugins 1 - Shaded log4j - JIRA fails bootstrap with plugins that contain copies of log4j |
| | JRA-20403 | The MultiSelectSearcher cannot be used from Plugins 2. |
| | JRA-20113 | Renderer module descriptor plugin can be declared only in one system plugin |
| | JRA-20060 | Some i18n broken for JIRA's system plugin name and/or description. |
| | JRA-19716 | Change timeout of plugins on startup to be 60 seconds from the most recently started plugin |
| | JRA-19495 | Jira doen not accept exclamation mark (!) in the home directory |
| | JRA-19465 | JIRA should check on startup for incorrect core plugins deployment |
| | JRA-19446 | Investigate the publishing of jira-subproject-plugins in our maven repo |
| | JRA-19418 | Document how to see application exposed packages and components for use by plugins |
| | JRA-19108 | Provide a way to link a Confluence Changeset to a JIRA ticket |
| | JRA-18986 | v2 plugins cannot refer to built-in custom field searchers in their plugin descriptors |
| | JRA-18761 | It would be good to list all the plugin points and the plugins that implement them |
| | JRA-18514 | Disabled JIRA plugin components can cause JIRA to not start |
| | JRA-18303 | plugin system should handle non numeric plugin versions better |
| | JRA-18109 | API to enable or disable plugin modules in bulk. |
| | JRA-16976 | Do not allow a plugin that registers components with pico to be disabled or enabled - JIRA needs to be restarted |
| | JRA-16606 | system.browse.user.operations Web UI plugin extension point |
| | JRA-16596 | Unable to add tooltips to the Administration menu. |
| | JRA-16557 | Group the Assignee list by group, security level or project roles |
| | JRA-16432 | Custom Assignee Resolver Component Plugin is injected incorrectly |
| | JRA-15335 | Remote API Interface doesn't enforce Resolution Setting |
| | JRA-15045 | Make existing components such as services that are supplied by plugins visible in administration section even if the plugin is missing |
| | JRA-14914 | Stop administrators from disabling the Workflow plugin |
| | JRA-14587 | JIRA should verify, if a given plugin is compatible/suitable for its release |
| | JRA-14503 | Ability to add a new sub-task tab panel as a plugin module |
| | JRA-14170 | Extend issue view and search request views to be able to produce binary content |
| | JRA-13437 | Webwork actions in plugins should be able to use the 'roles-required' attribute |
| | JRA-13389 | Make JIRA respect the minVersion in atlassian-plugin.xml |
| | JRA-13277 | Plugin description and web item label does not fall back to inline text if i18n key does not exist |
| | JRA-13144 | Extend JIRA's system-webresources-plugin to offer all framework components |
| | JRA-12517 | Plugin Guide for Issue Views |
| | JRA-12432 | Web Fragment Conditions cannot depend on custom component plugins |
| | JRA-12183 | Provide atlassian-plugin.xml DTD or XSD |
| | JRA-12087 | [RPC JIRA Plugin] Documentation for which RPC modules are required and which are optional |
| | JRA-12032 | Fix up issue key matching in plugins |
| | JRA-10867 | Change plugins UI to handle new plugins framework |
| | JRA-10631 | Create source control plugin for PVCS |

| | | |
|--|-----------|--|
| | JRA-10608 | Rollback change in LazyAxisDecoratorServlet once we have a better scheme for GlobalRefresh |
| | JRA-10462 | Custom Field whose values get displayed in issue tab |
| | JRA-9951 | Custom Field searchers should be accessible when the Any issue type is selected in Create New Filter |
| | JRA-9695 | Bundle lastest charting plugin with Jira. |
| | JRA-8835 | Support Administrative Plugin |
| | JRA-8793 | If a plugin fails to initialise it should be marked as inactive |
| | JRA-8177 | Allow all Confluence macros/plugins in JIRA markup |
| | JRA-7974 | Allow dynamic (calculated) custom field default values |
| | JRA-7612 | Integrate the 'Available Workflow Actions' CF into Jira |
| | JRA-7571 | Introduce screen initialisation plugin point |
| | JRA-7558 | Document how to internationalise plugins. |
| | JRA-7520 | Turn services and listeners into plugins |

User object customization

Improve the MS Office Add-in

| | |
|-------------|--|
| Name | Microsoft Office Add-in |
| Priority | High |
| Complexity | |
| Author(s) | Michael McKeown |
| Description | Improve and stabilize the MS Office Add-in |

Background

Users contributed VB scripts that, installed within [MS Word](#) and [Excel](#), will convert documents into wiki markup for pasting into Confluence. Soon after, Michael McKeown combined them into the [MS Office Add-in](#) which combined these two scripts into an office plugin to allow simpler installation and use.

Existing Plugin Features & Assumptions

- .Net 2.0
- Office 2000 and Office 2003
- UTF8 character set
- Button launches conversion form

Word

- Font Conversion
 - Italic
 - Bold
 - Underline
 - Subscript
 - Superscript
- Simple Table
- Table of Contents
- Lists
- Images (no in-line shapes)
- Hyperlinks
- Escapes Wiki Characters

The *All Docs* options finds & loads all *.doc or *.xls files in the current directory and sub-directories.

Excel

For each sheet in a workbook, if there is more than 1 cell selected, then the selection is uploaded, otherwise the entire worksheet is loaded. Only simply font formatting where the entire contents of the cell are formatted the same is currently handled.

Tasks

UI Cleanup

- Use normal buttons, instead of icon buttons
- Hide the select 'Parent' tree view by default. Offer an option to let them type in the parent name like Confluence.
- Have a button that hides the 'Parent' textfield and exposes the tree view instead.
- Can we move the universal things into a separate preferences dialog? The 'Host', 'Username', 'Password', and 'Open' should really be elsewhere?
- The Conversion log should probably be hidden by default.
- the 'All Docs' checkbox is confusing. Perhaps we need a different dialog all together for "Convert Many Documents" or something.
- Option to simply copy wiki markup to the clipboard but not attempt to create a page (no need to log in or select a parent etc)

Improve conversion

For these, the preference is to handle the conversion gracefully, but in the event that it is impossible, the next best thing is to pass over them without failing, and then mark the things/places where conversion failed.

Word

- Inline images
- Nested Tables
- Merged Cells
- Performance
- Fonts
- Font Colours
- Multiple instances of the same image in a document
- Support non-english locales

Excel

- Partial cell formatting

Improve Infrastructure:

- Support SSL
- Remember login credentials and login session.
- Performance (is this really an issue?)

Outstanding Bugs

JIRA: <http://developer.atlassian.com/jira/browse/OFCE>

JIRA-Mylar plugin

| | |
|-------------|---|
| Name | JIRA-Mylar Plugin |
| Author(s) | |
| Priority | High |
| Description | A plugin to Mylar to interact with JIRA |

Description

The [Mylar project](#) is a set of Eclipse plugins to allow developers to limit their view into a large codebase to only those items relevant to the task at hand. Mylar needs to integrate with bug trackers in order to understand what those tasks are. The Mylar project has written a basic JIRA plugin, and we'd like to continue that work.

Tasks

This plan is for 2 week iterations, April 1st - June 22, 2007. The first two iterations are about getting the connector to the current quality of the Mylar Bugzilla Connector. The next two are about getting the feature completeness to the level of the 2.0 version of the Mylar Bugzilla connector. The last two are for ensuring the structure of JIRA tasks is represented and getting and for documentation. The goal is that for the June 30th release JIRA users have first rate integration with Eclipse that supports their workflow seamlessly (we still get reports of users complaining that they need to switch to Bugzilla because the JIRA connector is not up to par).

Iterations below the bar include things that we would like to get to if the core work is completed early, or that could be set aside for ongoing work. Each iteration will be 1/3 bug fixing, 1/3 tests, and 1/3 new features, so that the user community sees a steady rate of improvement leading up to the 2.0 release. Key deliverables for each iteration are listed. We have scheduled deliverables around the May 1st start of your developer.

- I1: key JIRA Connector bugs fixes (see <http://www.eclipse.org/mylar/bugs.php>)
 - I2: unit test suite to match that of Bugzilla & Trac
 - I3: attachment and context sharing support (may need WS enhancement)
 - I4: jira-specific comment management. i.e. delete, specify who can see it (may need WS..)
 - I5: show issue dependencies/linking/subtasks in task editor and Task List
 - I6: documentation and articles
-
- Ix: support for custom attributes (need WS enhancements)
 - Ix: workflow enhancements, time tracking integration
 - Ix: vote and watch support (may need WS enhancements)
 - Ix: support display of issue change history (may need WS enhancement)

Resources

- JIRA API Changes required

JIRA-VSS Integration

| | |
|--------------------|---|
| Name | JIRA-VSS Integration |
| Author(s) | |
| Priority | High |
| Description | Just like SVN, show commit messages from VSS. |

Outline Plugin

| | |
|--------------------|--|
| Name | Outline Plugin |
| Priority | High |
| Complexity | Unknown |
| Author(s) | Jim Dibble |
| Description | Automatically turns heading tags into #.#.# outline format |

Description/features

Automatically turns heading tags into 1.3.5 outline format

Usage

User would enclose the page markup in the {outline} macro. The macro would automatically generate #.#.# outline numbers at the beginning of each heading. h1 headings would have "#", h2 headings would have "#.#", h3 headings would have "#.#.#", etc. Each level would start numbering at 1, and incrementing for each heading of the same level within scope. Level number would reset whenever a heading of a higher level appeared in the document (for example, a sequence of h1, h2, h2, h1, h2 would be numbered "1", "1.1", "1.2", "2", "2.1").

In wiki markup, the macro would look like this:

```

{outline}
h1. Overview
This is an overview of the document.
h1. Assumptions
h2. Technical Assumptions
This should not be too hard to accomplish.
h2. Usage Assumptions
Everyone would be willing to use it.
{outline}

```

The output would look like this:

1. Overview

This is an overview of the document.

2. Assumptions

2.1 Technical Assumptions

This should not be too hard to accomplish.

2.2 Usage Assumptions

Everyone would be willing to use it.

Technical Implementation

Other Resources

Display Access Macro

| | |
|-------------|--|
| Name | display-access |
| Priority | high |
| Complexity | low |
| Author(s) | |
| Description | A simple way to display who has access to a given piece of content |

Description/features

- show inline in a wiki page (or used in a layout), groups or individuals with permission or restriction on a piece of content
- displays regardless of if you have access to the content. To be used to indicate, in some cases, who to talk to to GET access.
- perhaps a 'collapsible' display to conserve real-estate, that just shows "This page had restrictions, Click for more". Optional

Usage

```
\{access:[page=@SELF/spaceKey:PageName/^attachment][|restrictions=[TRUE/false]][|view=[TRUE/false]]
```

Examples

{display-access} - shows group and individual restrictions and permissions for current page.

{display-access:CONFEXT:Plugin Wishlist} - shows group and individual restrictions and permissions for specified page

{display-access:CONFEXT:access|view=false|permissions=false} - only show edit restrictions for specified page

Technical Implementation

- may work nicely if the bulk of the 'query' was done using the reporting plugin using a new 'permissions and restrictions' supplier. In that case, this plugin would simply make a nice user interface and GUI for using the reporting plugin backend.

Other Resources

- Reporting Plugin

Cobertura build complete action

| | |
|-------------|---|
| Name | Cobertura build complete action |
| Priority | Medium |
| Complexity | Medium |
| Author(s) | |
| Description | A post-build action for Cobertura, a code coverage tool |

Background

Implement code-coverage reporting based on Cobertura.

Look at the [Clover](#) coverage tool plugin as an example.

SCP Articfact Copier

| | |
|-------------|---|
| Name | SCP Articfact Copier |
| Priority | Medium |
| Complexity | |
| Author(s) | |
| Description | Copies off artifacts to a remote location |

Background

Emma post build action

| | |
|-------------|--|
| Name | Emma build complete action |
| Priority | Medium |
| Complexity | Medium |
| Author(s) | |
| Description | A post-build action for Emma, a code coverage tool |

Background

Look at the Clover coverage tool plugin as an example.

Checkout Macro

| | |
|-------------|--|
| Name | Checkout |
| Priority | medium |
| Complexity | low |
| Author(s) | James Mortimer |
| Description | A macro that can be added to a page, an include, or a page layout, which allows the page visitor to single-click 'restrict' or 'unrestrict' a page |

Background

Many users want to check out a document temporarily when they work on them, but looking for the 'restrict page' is non-intuitive. For areas where we have high-traffic of new users, a simpler checkin/checkout button would greatly facilitate collaboration

Suggested Usage Example

{checkout-button}

Suggested Parameters

- icon = true/false (default true)
- restricttext = string (default Check-out)
- unrestricttext = string (default Check-in)
- restricttext = string (default "Current checked out by \$username")

Other Requirements

- first show a simple button (and icon) at the location of the macro that indicates "check out" (green)
- clicking this will 'restrict' the current page to 'edit only by me'
- now, display name of person to whom the document is checked out, and a simple button and icon for "check in" (red)
- when the original author (or an admin if possible) clicked 'check in', the restriction is removed and we cycle back to the original button
- if the restriction is removed by an admin through the admin panel, the checkout cycles back to the fist state (i.e. it reads it's state from the current page restriction)
- if the restriction is added by someone manually in a page edit, the checkout cycles to the second state (i.e. it reads it's state fromt he current page restriction)

Crowd Shibboleth Authenticator

| | |
|-------------|--|
| Name | Crowd Shibboleth Authenticator |
| Priority | High |
| Complexity | Medium |
| Author(s) | |
| Description | Allow Crowd users to sign in to a Shibboleth application |

Description/features

Allow Crowd users to sign in to a Shibboleth application

Usage

Technical Implementation

Other Resources

- <http://shibboleth.internet2.edu/>

Crowd .NET Integration Libraries

| | |
|-------------|--|
| Name | Crowd .NET Integration Libraries |
| Priority | High |
| Complexity | Medium |
| Author(s) | |
| Description | Allow .NET apps to use Crowd as their user store and SSO |

Description/features

Allow .NET apps to use Crowd as their user store and SSO

Usage

Technical Implementation

Other Resources

Crowd Delegated-Crowd-Connector

| | |
|-------------|---|
| Name | Crowd Delegated-Crowd-Connector |
| Priority | High |
| Complexity | Hard |
| Author(s) | |
| Description | Allow one crowd to user another as it's backing user repository |

Description/features

Allow one crowd to user another as it's backing user repository

Usage

Technical Implementation

Other Resources

Google Chart Support

| | |
|-------------|----------------------|
| Name | |
| Priority | |
| Complexity | Low |
| Author(s) | |
| Description | Google Chart Support |

Description/features

Google Chart Support

Usage

Leverage googles free charting support with a very simple confluence plugin

Technical Implementation

{gchart:name=value|name=value|...}

The resulting image tag would be

Hello]>

Other Resources

<http://code.google.com/apis/chart/>

Who is online

| | |
|-------------|--|
| Name | Who is online |
| Priority | High |
| Complexity | |
| Author(s) | |
| Description | Create plugin which will show who is online - is realy helpful when you update jira and you have restart live system during when other people use jira. With this plugin you will be able to check who is online and if nobody then you could restart all and make your updates. |

Description/features

Create plugin which will show who is online - is realy helpful when you update jira and you have to restart live system during when other people use jira. With this plugin you will be able to check who is online and if nobody then you know that you could restart jira and make your updates, witout any lost of users data.

Usage

VERY HIGH

Technical Implementation

I created a rudimentary version, which should be good enough for non-distributed server, but I have not tested it properly. Here's the source code: <https://bitbucket.org/egaga/jira-extensions/>

If someone makes it a bit more prettier, or extends it otherwise, I can update the code.

Other Resources

Create Dependent Issue plugin

| | |
|-------------|---|
| Name | Create Dependent Issue plugin |
| Author(s) | |
| Priority | Medium |
| Description | An issue operation to create a dependent issue. |

Google Maps Custom Fields for JIRA

| | |
|-------------|--|
| Name | Google Maps Custom Field |
| Author(s) | |
| Priority | Medium |
| Description | Show location usimng Google Maps in a JIRA issue |

Mail this bug

| | |
|-------------|---|
| Name | Mail this bug |
| Author(s) | |
| Priority | Medium |
| Description | An issue operation that mails an HTML representation of a bug to a given address. |

Resources

- <http://forums.atlassian.com/thread.jspa?messageID=257218161>

Post-resolution voting

| | |
|-------------|--|
| Name | Post-resolution voting |
| Author(s) | |
| Priority | Medium |
| Description | Allow issues to be voted and unvoted even after they are closed. |

Investigate whether this can this really be done as a plugin?

Find Related Issues

| | |
|-----------|---------------------|
| Name | Find Related Issues |
| Author(s) | |
| Priority | Medium |

| | |
|-------------|---|
| Description | Use Lucene to find and suggest similar issues |
|-------------|---|

Description

Create a tab panel which scans the fields in this issue and other issues (presumably using some Lucene trickery) to somehow determine if there are any related issues.

Also, create a post workflow function that does a fuzzy Lucene search to suggest duplicates. Could also extend to search a specified confluence instance as well.

JIRA Macro Library

| | |
|-------------|--|
| Name | JIRA Macro Library |
| Author(s) | |
| Priority | High |
| Description | More detail and options in the JIRA macros |

Currently there is a single [JIRA macro](#) within Confluence, which allows you to display the issues matching any JIRA filter within Confluence.

With the upcoming release of JIRA 3, we (the Confluence team) have a lot more fun remote capabilities that we can play with to create cool macros.

As such, we're planning to release a JIRA 3 macro library soon after JIRA 3 is finally released and publically available.

Here is our thinking regarding the macros it will contain:

- **jiraissues** - Just like the existing macro displays a list of the issues matching a given filter. Additionally, we would like to add a flag to just display the total # of matching issues with a link to the remote JIRA instance (ie "There are 7 matching JIRA issues").
- **jiraissue** - Display details about a single JIRA issue with in Confluence, available via the new remote XML issue view feature.
- **jiraportlet** - Display a remote JIRA portlet within Confluence - this is *an awesome macro!* You can then display of the new JIRA portlets (for example a 2 dimensional statistics table of component vs assignee in a given project) within a Confluence page.
- **jiraversion** - Display details about a single version within JIRA, including a graph of the version's progress towards completion.
- **jiraprogress** - A macro to show project release progress, eg:

```
{jiraprogress:url=http://jirapathproject}
```

Progress:  1 of 5 issues have been resolved

If you have any ideas of what else you would like to see, please let us know by commenting on this page and we'll try our best to include your thoughts.

Space Search and Replace

| | |
|-------------|---|
| Name | Space Search and Replace |
| Author(s) | Jon Nermut |
| Priority | Medium |
| Description | Provide an interface to do a space level regex search and replace on the wiki text of all pages in the space. |

This is a really common use case - you've used a specific term all through a bunch of wiki pages and now you want to change it. At the moment that means a really tedious job.

Should live in the Space Admin page, as it is a potentially very destructive operation.

A flexible way to select pages would be good too - all pages, a certain pages and its children recursive, or a regex on page title.

AssignUP

| | |
|--------------------|------------------------------------|
| Name | AssignUP |
| Priority | BLOCK |
| Complexity | |
| Author(s) | |
| Description | assign more than one user to issue |

Description/features

assign more than one user to the issue

Usage

very usage

Technical Implementation

Other Resources

Chat Plugin

| | |
|--------------------|--|
| Name | Chat Plugin |
| Author(s) | |
| Priority | High |
| Description | A web-based chat plugin with transcripts |

Description

I'd like to offer a real-time, group-based, persistent chat inside a Confluence page. The main purpose of this plugin would be to allow groups to have an online discussion between multiple parties, collaborate over Confluence content with a real-time backchannel, and keep a permanent record of the conversation, which would then become searchable by Confluence search.

This Chat plugin is specifically **not** about person-to-person IM. There are plenty of good systems for that already, and we incorporate them to a small degree already with the [IM Presence Plugin](#).

Implementation

Any user can create a new chatroom by placing a chat macro on a page, like this: `chat :My Room Name`. That macro will render an HTML table that contains information: name, last activity, participants, files uploaded, transcripts by date, as well as a big link that says "Join Chat."

Clicking the "Join Chat" link will launch a new popup window with no browser chrome, but containing the chat-room UI. Across the top should be a title. Down the right side should be an info panel. At the bottom is the input area and submit button, and in the middle should be the content window. Just like regular IM, content should be appended to the bottom of the page and it should gradually scroll off the top. As each participant submits a message, it appears on the other screens with name and timestamp.

Whenever there is activity in the chat room, it should log the conversation. That conversation should be stored on a child page of the chat-room page, separated by date – probably one page for a full day's transcript. If there is no activity that day, don't create a transcript page. The transcripts should be formatted as close to the original as possible.

Permissions should be handled by using the permissions of the page on which the chat-room macro occurs. The Chat will also take the parent page's CEO id as its id, in order to ensure uniqueness.

Use Cases

Features

Beyond the basic chat feature, we could also add:

- Wiki-rendering of messages, or at least wiki-link rendering.
- Paste-support, that allows you to view pasted content at full size and formatted as <pre>.
- Sound. We could make a beep when a new message appears.

References

- <http://www.oregano-server.org/> - we could try to use this as a backend.

Screenshots

Here are a couple of very simple wireframes that describe what I'm imagining.



Gantt Chart Plugin

| | |
|-------------|--|
| Name | Gantt Chart Plugin |
| Author(s) | |
| Priority | Medium |
| Description | Allow creation of Gantt Chart from Wiki Markup |

<http://forums.atlassian.com/thread.jspa?messageID=257223025>



Chart version 1.8 released

1.8 contains the proposal below as a beta. I have moved some of the documentation from here to [Gantt Charts](#). Discussions and/or issues welcome.

Proposal

- Add a gantt chart to the Confluence Chart Plugin based on the information and comments below.
- One issue is how to specify the data for a chart - the following gives some detail on this

Example 1 - simple

| Plan | Start | End |
|------------------|-----------|----------|
| Create issue | 1/30/2007 | 2/1/2007 |
| Provide examples | 1/30/2007 | 2/7/2007 |
| Write code/tests | 2/3/2007 | 2/4/2007 |
| Write docs | 2/4/2007 | 2/5/2007 |
| Release update | 2/5/2007 | 2/5/2007 |

Example 2 - more features

| Category | Parent task | Task | Start | End | Percent complete |
|----------|-------------|------------------|-----------|----------|------------------|
| Plan | | Create issue | 1/30/2007 | 2/1/2007 | 30% |
| Actual | | Create issue | 1/30/2007 | +3 | 100% |
| Plan | | Provide examples | 1/30/2007 | 2/7/2007 | 40% |
| Plan | | Code | 2/3/2007 | 2/4/2007 | |
| Plan | Code | Initial code | 2/3/2007 | +1 | |
| Plan | Code | Tests | 2/3/2007 | +2 | |
| Plan | Code | Final | 2/4/2007 | +2 | |
| Plan | | Write docs | 2/4/2007 | 2/5/2007 | |
| Plan | | Release update | 2/5/2007 | 2/5/2007 | |

Notes

- Multiple tables can be used (just like other charts)
- Only **dataOrientation=vertical** is supported (there are no horizontal data layouts proposed)
 - Rows represent tasks (or subtasks)
- Dates are specified in format defined by the **dateFormat** parameter (like other time based charts)
- Percent complete column is optional
- Category column is optional, if not provided header column 1 will be the category name
- Parent column is optional, if not provided, you cannot specify subtasks
- End time can be specified as a full date or as a duration added to the start time
 - Duration is measured in number of units specified by the **timePeriod** parameter
 - Can be specified as a simple number like 1 or +1
 - full date and duration times can be intermixed
- To specify a subtask, the parent task must be provided
- Column selection would be done by a general **columns** parameter being added to the Confluence Chart Plugin via CHRT-28
- Other possibilities
 - Allow start time to be specified as a delta over some global start time
 - Allow start time to be determined as a delta from the end date of a "depends on" task - this would need another optional column
 - Allow subsequent category names to be blank and default to previous category name to make input tables easier to make and read

Numbered Sections Macro

| | |
|-------------|--------------------------------|
| Name | Numbered Sections Macro |
| Author(s) | |
| Priority | Medium |
| Description | Automatically number headings. |

I'd like to be able to surround a block of wiki-content with a macro that would automatically number and anchor all the h1..h5s. Kinda like Word would do it. It would be nice if this could be combined with the [Table of Contents](#) macro too.

Also, people want: Collapsible outline, multiple sections (eg, 3.2.2.4: technical spec, 3.2.2.4: testing spec match up), comment/approve at any level in the tree.

SERENA Dimension Integration

| | |
|-------------|--|
| Name | SERENA Dimension Integration |
| Priority | high |
| Complexity | Medium - High |
| Author(s) | |
| Description | Provides a SERENA Dimension Integration (Release 10.x) |

Description/features

Provides a SERENA Dimension Integration (Release 10.x)

Usage

- Bamboo builds
- JIRA SCM Panel

Technical Implementation

- Windows XP
- Linux
- AIX

Other Resources

Combined Filters JIRA Plugin

| | |
|-------------|--|
| Name | Combined Filters JIRA Plugin |
| Priority | High |
| Complexity | Medium |
| Author(s) | |
| Description | A plugin that would allow the "merging" of query output from multiple filters into a new filter. |

Description/features

A plugin that would allow the "merging" of query output from multiple filters into a new filter.

Usage

In all areas of filters, the ability to provide a union of filters for viewing, reporting, etc.

Technical Implementation

Other Resources

Communication Plugin

| | |
|-------------|---|
| Name | Communication Plugin |
| Priority | High |
| Complexity | High |
| Author(s) | |
| Description | Click a username to start VoIP, instant message, email or view profile. |

Description/features

Click a username to start VoIP, instant message, email or view profile.

Usage

Whenever a user account is listed, the visitor can click on the account to be presented with the option to start a VoIP call, instant message, email or view profile. The link can be added to their user profile.

Haven't had time to research the guts but am guessing:

VoIP can use `sip:myemail.provider.com`

Messaging might need a browser plugin to hook up a Jabber client?

Email is easy as it can just use `mailto: myemail.provider.com`

Technical Implementation

Other Resources

Confluence Bamboo Plugin

{metadata-list}

| | |
|-------------|--|
| Name | Confluence Bamboo Plugin |
| Priority | High |
| Complexity | Low |
| Author(s) | |
| Description | Show various data from Bamboo in a Confluence page |

{metadata-list}

Description/features

Show various data from Bamboo in a Confluence page.

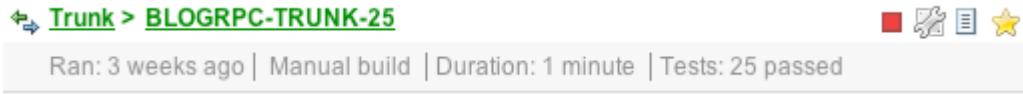
Usage

I'd like a macro that can do the following:

Current-Build Macro

`{current-build:PLAN_KEY}` should give me a representation of the most recent state of any given build plan. There should be three modes: inline, compact, and full.

- **Inline** should just print **PASSED** or **FAILED**.
- **Compact** should use the small icon to do the same thing.
- **Full** should bring the build summary from Bamboo, as illustrated in this screenshot.



Recent Builds Macro

`{recent-builds:PLAN_KEY}` Should show the recent build history of a plan.

`{recent-builds:user=USERNAME}` Should show the recent build history of a user across all projects.

My Latest Changes

| Build | Completed | Comments | [details] | Test Results |
|--|-------------|--|-----------|-----------------|
| RSVP-TRUNK-6 | 2 weeks ago | Restructuring along lines required by new acceptance test harness. | [details] | No tests found! |
| RSVP-TRUNK-2 | 2 weeks ago | Updating POM to newer setup to work on BDAC. | [details] | No tests found! |
| PLGNREPOCL-TRUNK-60 | 2 weeks ago | Fixed URL. | [details] | 25 passed |
| PLGNREPOCL-STABLE-58 | 2 weeks ago | Fixed URL. | [details] | 25 passed |
| PLGNREPOCL-PREVSTABLE-53 | 2 weeks ago | Fixed URL. | [details] | No tests found! |
| CONFPLUG-TINYMCE-7 | 3 weeks ago | Changing parent POM. | [details] | No tests found! |
| PLGNREPOCL-TRUNK-46 | 3 weeks ago | Formatting. | [details] | 1 passed |
| PLGNREPOCL-STABLE-42 | 3 weeks ago | Formatting. | [details] | 1 passed |
| PLGNREPOCL-PREVSTABLE-39 | 3 weeks ago | Formatting. | [details] | No tests found! |
| WEBDAV-STABLE-3 | 3 weeks ago | Changing parent pom. | [details] | No tests found! |

{recent-builds:project=PROJECT_KEY} Should show the recent build history of all plans in a project.

Recently Completed Builds

| Build Number | Reason | Completed | Duration | Test Results |
|---|--|-------------------------------------|--------------------|----------------------|
| ATLASSIANBANDANA-TRUNK-23 | Initial clean build | 15 hours ago | 10 seconds | 5 passed |
| ATLASSIANUSER-TRUNK-93 | Updated by akazatchkov | 16 hours ago | 47 seconds | 452 passed |
| ATLASSIANUSER-TRUNK-92 | Updated by akazatchkov | 16 hours ago | 1 minute | 452 passed |
| CONF-TRUNKtrigger-10 | Manual build | 17 hours ago | less than 1 second | No tests found! |
| CONF-TRUNK-57 | Updated by alynch | 17 hours ago | 22 minutes | 2115 passed |
| JIRAPLUGINS-FISHEYE-43 | Manual build | 17 hours ago | 2 minutes | 1 out of 33 failed! |
| JIRAPLUGINS-FISHEYE-42 | Manual build | 18 hours ago (11:52 PM, Tue, 9 Oct) | 1 minute | 1 out of 33 failed! |
| JIRAPLUGINS-FISHEYE-41 | Manual build | 18 hours ago (11:49 PM, Tue, 9 Oct) | 1 minute | 21 passed |
| JIRAPLUGINS-FISHEYE-40 | Manual build | 18 hours ago (11:44 PM, Tue, 9 Oct) | 1 minute | 10 out of 33 failed! |
| CONF-TRUNKtrigger-9 | Manual build | 18 hours ago (11:31 PM, Tue, 9 Oct) | less than 1 second | No tests found! |
| ATLASSIANUSER-TRUNK-91 | Updated by akazatchkov | 19 hours ago (10:49 PM, Tue, 9 Oct) | 55 seconds | 452 passed |
| ATLASSIANUSER-TRUNK-90 | Updated by akazatchkov | 19 hours ago (10:11 PM, Tue, 9 Oct) | 52 seconds | 452 passed |
| CONF-TRUNKtrigger-8 | Manual build | 20 hours ago (09:41 PM, Tue, 9 Oct) | less than 1 second | No tests found! |
| JIRAPLUGINS-FISHEYE-39 | Manual build | 20 hours ago (09:20 PM, Tue, 9 Oct) | 1 minute | 10 out of 33 failed! |
| JIRAPLUGINS-FISHEYE-38 | Manual build | 21 hours ago (08:42 PM, Tue, 9 Oct) | 1 minute | 10 out of 33 failed! |

Project Macro

{bamboo-project:PROJECT_KEY} should print the current status of all builds in a project, as shown in this screenshot.

Confluence Blogging RPC Collapse

↳ [PrevStable > BLOGRPC-PREVSTABLE-14](#)

Ran: 2 days ago | Manual build | Duration: 4 minutes | Tests: 25 passed

↳ [Stable > BLOGRPC-STABLE-11](#)

Ran: 2 days ago | Manual build | Duration: 18 minutes | Tests: 25 passed

↳ [Trunk > BLOGRPC-TRUNK-25](#)

Ran: 3 weeks ago | Manual build | Duration: 1 minute | Tests: 25 passed

Bamboo Charts Macro

{bamboo-chart:PLAN_KEY} should display the various charts that Bamboo can calculate, such as Duration and failures, percentage of test

failures, precentage of code coverage from Clover, lines of code, number of tests, and so forth.

- **builds** should allow you to change how many builds are charted. (default 25)
- **type** lets you select which charts to display.

Technical Implementation

- <https://bamboo.deveoper.atlassian.com/api/index.action>

Other Resources

- Bamboo Widgets

Email multiple users, groups

| | |
|-------------|--|
| Name | Admin: Email multiple users/groups |
| Priority | High |
| Complexity | |
| Author(s) | |
| Description | Admin should have the opportunity to mass-email all or some registered users and/or groups |

Description/features

Admin should have the opportunity to mass-email all or some registered users and/or groups

Usage

When important changes are implemented the Administrator often would like to send an email to all or some of the registered users and/or groups. Until now, this option is not available in Confluense.

Video Chat Plugin for Confluence

| | |
|-------------|---------------------------------------|
| Name | Video Chat Plugin for Confluence |
| Priority | High |
| Complexity | High |
| Author(s) | Matti Kiviharju (Teknologiaplaneetta) |
| Description | Video Chat with Red5 Flash Server |

Description/features

Video Chat with Red5 Flash Server

Usage

For Instant Messaging and Live Webcam Streaming (possible Desktop Live Sharing for computing and software usage help and support)

Technical Implementation

Confluence Macro/Plugin that makes Flex 2 Video Chat Application possible to use in Confluence Interface. First thing is Security with between Flash/Flex Apps, Confluence and J2EE.

Other Resources

Red5 Flash Server <http://osflash.org/red5>

Flex 2 <http://www.adobe.com/products/flex/>

Original Plugin: <http://confluence.atlassian.com/display/CONFEXT/Video+Conference+Plugin>

Embed Gantt Project

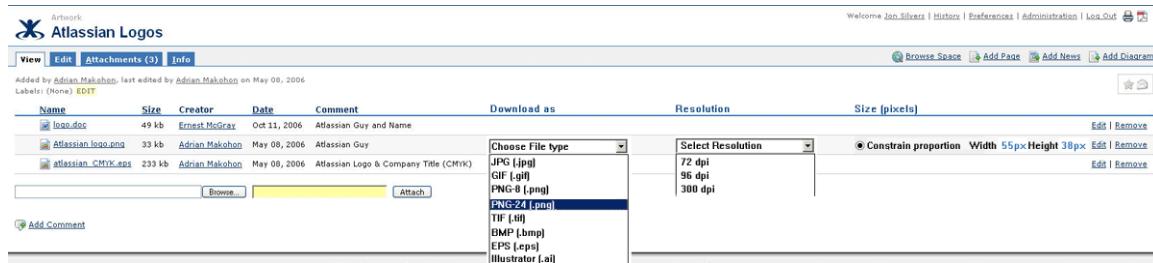
| | |
|-------------|--|
| Name | Embed Gantt Project |
| Priority | Medium |
| Complexity | High |
| Author(s) | |
| Description | Gantt Project provides a reasonable alternative to MS Project written in Java/Swing. What would be awesome would be to embed the app within a confluence plugin so you could launch it from a confluence page, and it saved the project xml file as an attachment to the page (maybe using its webdav support??), then rendered it on the page as an image in view mode. |

And it could integrate with Jira as well with Issue=Task. That would be off the show.

File download and transformation plugin

For anyone who has many images to format, a file transformation plugin would be great. The plugin would allow one to transform an image from any file format, to any file format (e.g., vector, web-ready, print ready, etc.).

Mockup



Usage

- Users could choose file format, resolution, and size
- Size would be constrained proportionally as default (user can enter width or height and size would adjust proportionally)
- After all choices are made, user clicks Download (not pictured in mockup)
- File transformation happens on server
- File transformations and downloads for large file sizes (>5MB) could take up significant bandwidth and/or processing, may be necessary to have transformation done on separate server

Background

- Digital asset management systems (usually retail for \$100k or more) have this feature built in. May be good to evaluate these tools to see how they perform this operation
 - Canto Cumulus
 - ClearStory

Technology

The Adobe Graphics Server will facilitate some of these transformations, though it's not practical or affordable for many organisations (the plugin could be free, but AGS won't be). Is there an open source alternative?

Repository Tagger build complete action

| | |
|--------------------|---|
| Name | Repository Tagger build complete action |
| Priority | High |
| Complexity | Medium |
| Author(s) | |
| Description | Tag a build in SVN or CVS when it succeeds. |

Background

Use a Build Complete Action to issue commands to a VCS repository.

Other Requirements

Friendly User Groups

| | |
|--------------------|---|
| Name | Friendly User Groups |
| Author(s) | Adaptavist are already planning to do some work on this |
| Priority | High |
| Progress | Initial concept formed |
| Status | Awaiting developer resources |
| Description | Allow user groups to have titles and descriptions |

Description

Currently, end-users are faced with user groups like confluence-users, etc., that mean little to them. As more advanced social networking/community plugins are developed, along with more plugins for managing user groups, there will be an ever increasing range of user groups stored within the average Confluence installation - some of these will be made by site admins, others by space admins (eg. delegating group management to space admins) and some will be auto-created to accomodate various plugins that store collections of users in groups.

I would very much like to see a way of making user groups far more friendly - for example:

| Group | Title | Description |
|---------------------------|----------------------|--|
| confluence-users | All Registered Users | This group contains all registered users on the ACME wiki. |
| confluence-administrators | ACME Administrators | This group contains all system administrators |
| sales-team | Sales Department | Everyone in our sales department is in this group |

This would allow far more friendly UI to be designed and de-techify the process of working with user groups for the average end-user.

A suitable API would be required to allow plugins that auto-create user groups to also assign titles and descriptions and an admin UI would be required to allow admins to update titles and descriptions for user groups as and when required.

Keyword Index

| | |
|--------------------|--|
| Name | Keyword Index |
| Priority | Medium |
| Complexity | medium to high |
| Author(s) | James Mortimer |
| Description | Create a keyword-based index for the content of pages. |

Background

like an index at the back of a book, the word 'computer' should link to all pages that have the word computer on it

Suggested Usage Example

```
{keyword-index:groupby=letter|root=CONF:page1|scope=@descendents|output=list|...}
```

would give you a table such as

A

- apple [Keyword Index](#) [SVN Commit Acceptance](#)

C

- computer [Keyword Index](#) [other page2] [other page3]

Suggested Parameters

| parameters | examples | meaning |
|-----------------|--|--|
| output | list / wiki / dynamic / table / other? | how to format the output |
| groupby | none / letter | specify grouping options |
| root | @self, @parent, @home, @none | specify the context of the index |
| scope | @self, @ancestors, @space, @all, @descendents, @children | which pages to include in the index, relative to the context |
| include | a page name or text attachment that contains a list of the keywords to include | |
| exclude | a page name or text attachment that contains a list of words to exclude | |
| maxpagesperword | <number>, default=n/s | any words with more than this number of pages are excluded (stop words) |
| minpagesperword | <number>, default=1 | any words with less than this number of pages are excluded (uninformative words) |

Other Requirements

- a method to specify which words to include or exclude. this might be by word-length, or cherry picking
For example, from a text file attachment
- the dynamic option would show just the list of keywords (grouped by letter). When you hover-over and/or click-on a keyword, a list of the relevant pages are shown in a popup.

Notes

An approximation of this, if you want to specify all keywords as you would in include, can be attained by using something like:

A

- apple

C

- computer

P

- plugin

the keyword-index would simplify having to create this complicated index, so you would only have to list the bare-words or just let confluence list all the barewords in its index.

RSS Mix macro

| | |
|-------------|--|
| Name | RSSMix |
| Author(s) | |
| Priority | Medium |
| Description | Aggregate multiple RSS sources for display |

A gorgeous mixrss macro:
like a minime version of javablogs

```
{rssmix: max =10| title ="Java News"
http://confluence/space/user1/blog
http://confluence/space/user2/blog
http://someexternalblog/rss
{rssmix}
```

would look something like this : (with perhaps some test from each post like the current rss macro)

The screenshot shows a Confluence page with a heading 'Java News'. Below it is a list of ten RSS items, each with a blue square icon, a title, a 'View Details' link, the author, the date, and the number of reads. The items are:

- DemoMobile Conference
- People behind this year's most talked-about technologies
- To a clueless analyst: boring is good
- java xml pull and push: a comparison
- update on rss autodiscovery
- From black boxes to enterprises
- Tired
- Java productivity
- PicoContainer

Each item includes a 'View Details' link, the author ('d2r'), the date ('19-Sep-03 05:36:40'), and the number of reads (e.g., 103, 15, 46, etc.).

So we can create mini topical rss 'newspapers' or have a space front end which is an aggregation of multiple user blogs.

Once we have user blogs or just topical blogs in confluence we can have :

```
{rss-space-users:max=10}
```

show a mix (sorted by post date) of all blogs on current space .

User History

| | |
|-------------|--|
| Name | User History |
| Author(s) | |
| Priority | Medium |
| Description | History of a user's view, edit and create actions. |

This plugin should consist of two macros:

user-recent-activity

This macro would show a list of all See also [Better Recent Changes Macro](#).

Params

- user - which user(s) to show.
- days - last X days.
- space - limit by space(s).
- label - limit by label(s).
- content - which types of content (page, comment, attachment, news, etc). Default to ALL.
- limit - last X items.

user-activity

A chart of a user's activity over last X days. It should show view, edit and create actions. See also [Usage Stats Plugin](#).

Params

- user - which user(s) to show.
- days - last X days.
- space - limit by space(s).
- label - limit by label(s).
- content - which types of content (page, comment, attachment, news, etc). Default to ALL.
- width - size of the graph.
- height - size of the graph.

Amazon API Macros

| | |
|-------------|--|
| Name | Amazon API Macros |
| Author(s) | |
| Priority | Low |
| Description | View item by ASIN, search by ISBN, maintain/view wishlists |

View item by ASIN, search by ISBN, maintain/view wishlists. See [Amazon API](#)

Now available as a Codegeist 2007 Contest Entry.

Project pages:

- [Amazon Web Services Plugin](#)
- [Amazon linking as a user macro](#)

Dictionary in Editor

| | |
|--------------------|-------------------------------|
| Name | Dictionary in Editor |
| Author(s) | |
| Priority | Low |
| Description | Add Dictionary plugin to RTE. |

Provide a dictionary integrated to the editor. The plugin coming with TinyMCE could certainly be used, but it would be better to have a server side dictionary.

Drag and Drop Attachments

| | |
|--------------------|--------------------------------------|
| Name | Drag and Drop Attachments |
| Author(s) | |
| Priority | Low |
| Description | Allow drag and drop of attachments . |

Allow drag and drop from the client to Confluence (add attachments), from Confluence to the client (download attachments, export page to pdf...), from Confluence to Confluence (change page location to another Space or another father page).

Recently Commented

| | |
|--------------------|---------------------------------|
| Name | Recently Commented |
| Author(s) | |
| Priority | Low |
| Description | List of recently comments only. |

Provide a simple way to list all the last comments in a Space or in the whole wiki; this should be done in a similar way as we can list the Recently Updated pages. The list should display the Space name, the Page name, the Author of the comment and the Date/Time when it was published; having the possibility to display a few words extracted from the comment would be a nice option; of course clicking on the list would open the relevant page and display the selected comment.

Search and Replace and Date and Time

| | |
|--------------------|--|
| Name | Search-and-Replace and Date-Time |
| Author(s) | |
| Priority | Low |
| Description | Allow date&time and serach&replace in RTE. |

Provide a search & replace function in the editor mode (Rich Text AND Wiki Markup). This is already available with the Rich Text editor used but not implemented in confluence ([TinyMCE](#)): there may be a reason? However, we also want it for the Wiki Markup editor.

Provide an easy way to insert today's date and time in the editor.

This can be achieved by replacing the following lines in ..\confluence\template\notable\wiki-textarea.vm:

| |
|---|
| before |
| theme_advanced_buttons2 : "", |
| after |
| theme_advanced_buttons2 : "insertdate,inserttime,separator,cut,copy,paste,separator,search,replace", |

and

| before |
|---|
| plugins : "table,confluence", |
| after |
| plugins : "table,insertdatetime,searchreplace,confluence",plugin_insertdateDateFormat : "%Y-%m-%d", plugin_insertdate_timeFormat : "%H:%M:%S", |

You finally have to unzip the attached files (1) (2) in your ..\confluence\includes\js\tiny_mce\plugins folder.

Default Charts Tab Panel

| | |
|--------------------|--|
| Name | Default Charts Tab Panel |
| Author(s) | |
| Priority | High |
| Description | Add a Project Tab Panel with a set of generally useful charts. |

Improved index macro

| | |
|--------------------|-------------------------------|
| Name | Improved Index Macro |
| Author(s) | |
| Priority | Low |
| Description | Include other display options |

Include other display options, such as page list only, alphabetical without content previews, and alphabetical with content preview (how it is now)

other thing ... let's display the alphabetical list of article titles without their content the same way as the {listlabels} does
(the best would be to get inspiration from MediaWiki)

Google Maps Custom Field for JIRA

There are a lot of interesting custom field (beyond the above) that could be done with Google Maps. For example, simple country, city or address custom fields could display Google maps of that particular area / region.

Checkbox Tree Custom Field

| | |
|--------------------|---|
| Name | Cascading Multi-checkbox CustomField |
| Author(s) | |
| Priority | Low |
| Description | A custom field for cascading multi-selects. |

CascadingSelect fields are nice, but allow only one selection. The ideal replacement imho would be something like
http://www.treemenu.net/treemenu/3fr_checkbox.html.

Reporter IP CustomField

| | |
|--------------------|--|
| Name | JIRA Reporter IP CustomField |
| Author(s) | |
| Priority | Low |
| Description | A custom field to record the IP of the reporter. |

A custom field to record the IP of the reporter. This is fairly simple to do, but what would be *neat* is if on the display page it then used Google Maps to *show you* there the issue was reported from. (see [this example](#) for the IP <-> Google Maps mashup)

cli

compress-resources

copy-bundled-dependencies

create

create-home-zip

debug

filter-plugin-descriptor

generate-manifest

generate-obr-artifact

idea

install

integration-test

jar

run

unit-test

validate-manifest