User / Developer Guide

2.1.0

Copyright © 2004-2008 Peter Thomas

Table of Contents

Preface .....
1. Introduction .....
1.1. About 1
1.2. Release Notes: Version 2.1.0 1
2. Features .....
2.1. Easy to Install 2
2.2. Custom Fields 2
2.3. Custom Workflow 3
2.4. Detailed History View 6
2.5. Attachments Support 6
2.6. Custom Roles 6
2.7. Search Custom Fields 7
2.8. Dashboard 7
2.9. Search Across All Projects 8
2.10. Full Text Search 8
2.11. Export to Excel 9
2.12. Simple Navigation 9
2.13. Cross Referencing of Items 10
2.14. Read-Only Access 10
2.15. E-mail Integration 11
2.16. Multi-Language Support 12
2.17. LDAP / CAS authentication support 12
3. Installation .....
3.1. Prerequisites 13
3.2. Quick Installation 13
3.2.1. Notes for Linux / Unix users 13
3.2.2. Security and Backup 14
3.3. Settings 14
3.4. Using only the WAR file 15
3.5. Custom Installation 15
3.5.1. Edit the "jtrac-init.properties" file 15
3.5.2. Set a Servlet context parameter 15
3.5.3. Set a System / JVM parameter 15
3.6. Using a different database such as MySQL 16
3.7. Using a JNDI Datasource 16
3.8. Configuring LDAP Authentication 17
3.9. Integrating with CAS for Single Sign On 17
3.9.1. Changes to web.xml 17
3.9.2. Changes to applicationContext-acegi-cas.xml 17
3.10. Installing as a Windows Service 18
3.11. What to Backup 19
3.12. Upgrading 19
4. FAQ .....
4.1. Do we really need another issue tracker? 20
4.2. What is the JTrac architecture like? 20
4.3. Why start with version 2.0? What about 1.0? 20
4.4. What is the username and password when you log in for the first time? 20
4.5. How can I help? 20

4.6. We are heavy users of JTrac but it is running on the embedded HSQLDB database. Can it cope? 21
4.7. Why does JTrac have a limit on custom fields? I really need more! 21
4.8. How do I report bugs or feature requests? 21
4.9. Why is JTrac not being used as the bug-tracker for this project? You should be eating your own dog food right? 21
4.10. Do you have a list of users or references? 22
5. Roadmap .....
5.1. Field-Level "Hide" Permissions 23
5.2. Nested Items 23
5.3. Custom Validation 23
5.4. Custom Scheduled Jobs 23
5.5. Submit By Email 23
5.6. Screenshot Capture 23
5.7. Saved Searches 24
5.8. Wiki Engine 24
5.9. Subversion Integration 24
5.10. Tags 24
5.11. RSS Feeds 24
5.12. Import from other tools 24
5.13. Single Sign On 25
5.14. XML API 25
5.15. Eclipse Mylyn integration 25
5.16. Time Tracking 25
5.17. Custom Reports 25
6. Interim Builds .....
6.1. Overview 26
6.2. Notes on using Interim Builds 26
7. Upgrading and Database Migration .....
7.1. Deleting webapp temporary files 27
7.2. Upgrading the Database 27
7.3. Upgrading Jetty 28
7.4. Database Migration 28
8. Developer Guide .....
8.1. Pre Requisites 29
8.2. Check Prerequisites 29
8.3. Download / Extract Source 29
8.4. Customize Ant Build Properties File 30
8.5. Generate Dependencies Properties File 30
8.6. Import project into your IDE 30
8.7. Building And Running JTrac 31
8.8. Adding a language translation for JTrac 31

Preface

This document is a reference guide for JTrac - the generic issue tracking web-application. This document is not only a user-guide but can also serve as a reference for developers interested in contributing to JTrac.

This documentation has been generated using the DocBook configuration used by the Spring development team. This particular simplified DocBook helper package was originally developed by Chris Bauer of the Hibernate project. Thanks go out to all those who perfected this very handy approach.

Chapter 1. Introduction

1.1. About

JTrac is a generic issue-tracking web-application that can be easily customized by adding custom fields and drop-downs. Features include customizable workflow, field level permissions, e-mail integration, file attachments and a detailed history view.

JTrac was created after the author felt that he could write a much better alternative to a commercial defect tracking tool that he was having to use. Development started in 2004. JTrac is ideal for issue tracking or bug-tracking, but it has been designed to be generic and you can define custom fields to track almost anything you need.

JTrac development used to be hosted at https://jtrac.dev.java.net but moved to SourceForge in early 2006. The older working version was based on Spring MVC, Spring JDBC and MS Access and is not being maintained any more. JTrac 2.0 uses Java 5.0 features and was completely re-written to use Hibernate for persistence and the Acegi Security framework for Spring. The presentation layer for version 2.0 was mainly using Spring WebFlow but as of early 2007 (version 2.1.0 onwards) JTrac switched to using the Apache Wicket framework.

JTrac is extremely easy to install and the only pre-requisite is a Java 5 (or higher) Runtime Environment. You can be up and running in seconds because JTrac comes bundled with a small-footprint web-application server called Jetty and an embedded database called HSQLDB. Please refer to the installation section of this documentation for details. You can also choose to drop the WAR file into an application server of your choice and start using JTrac right away.

1.2. Release Notes: Version 2.1.0

There are no database changes between version 2.0 and 2.1.0. Instructions on how to upgrade are available in the "upgrading" section of this document. The bundled Jetty web-app server has been upgraded from version 6.0.2 to 6.1.1 but it is not mandatory that you upgrade.

Chapter 2. Features

JTrac has all the features that you would expect from a standard issue-tracking application such as support for file-attachments and e-mail integration. JTrac offers powerful customization options, especially in the areas of workflow and field-level permissions and compares well to even commercial tools.

2.1. Easy to Install

One of the useful features of JTrac is that it is distributed with an embedded web-application server (Jetty) that has a very small footprint. If you have Java 5 installed you can be up and running after downloading and extracting JTrac - by simply using the provided start and stop scripts. JTrac scans its environment on startup and if a configured database is not detected, HSQLDB is used by default.

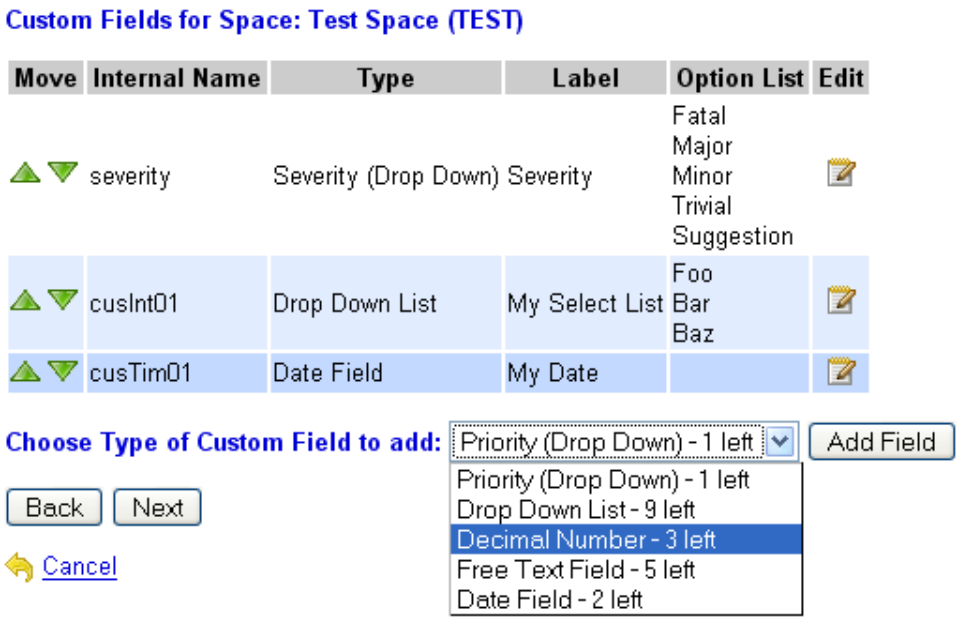
If you already have a web-application server that you wish to use (such as Tomcat), you can just drop in the WAR file provided and sign on into the application straight-away.

You can refer the installation section of this document for more details. JTrac uses Hibernate and so it can use any database supported by Hibernate.

2.2. Custom Fields

JTrac can be used to effectively track various kinds of things such as Bugs, Action Items and Tasks. You can easily add the following types of custom fields to a tracker Project (Space):

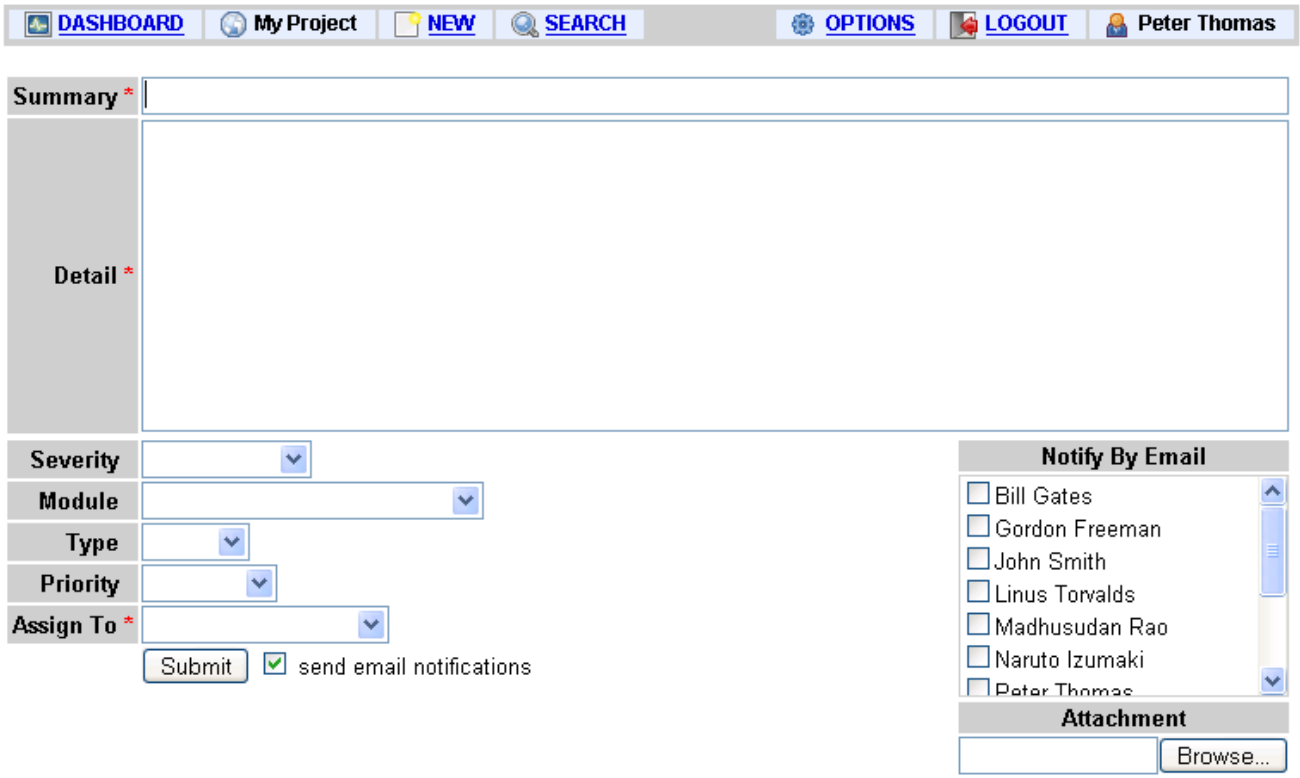
- Drop Down List
- Free Text Field
- Date Picker
- (Decimal) Number



Setting up custom fields for a project

There is a limit to how many custom fields of each type that you can configure for a Space. These limits have been set after careful consideration and would very rarely fail to suffice. They are as follows: Drop Down - 10, Free Text - 5, Numeric - 3, Date/Time - 3.

The screenshot below shows how creating a new item would look like for the typical project (space) where 'Severity', 'Module', 'Type' and 'Priority' have been defined as custom fields.

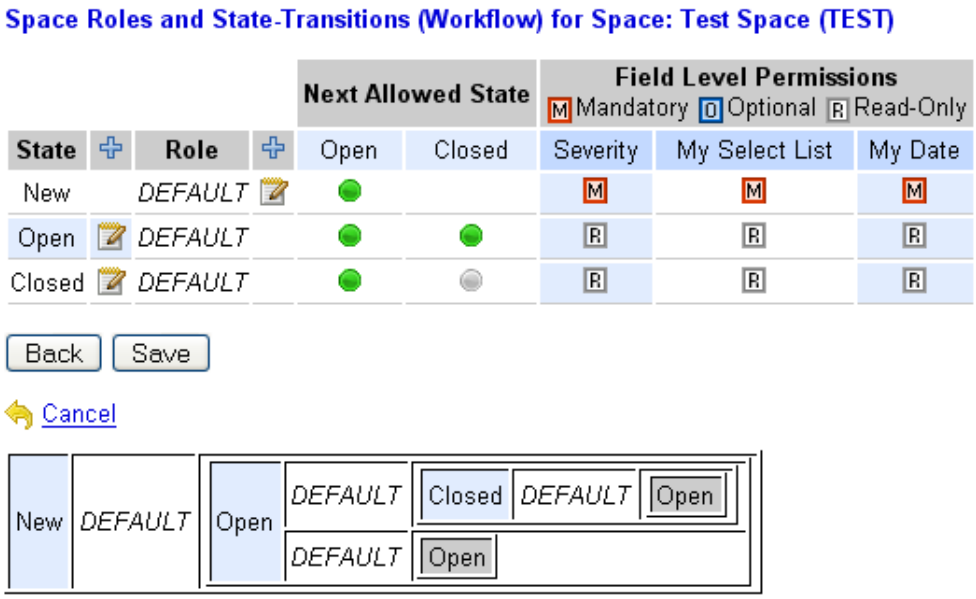


Creating a new item

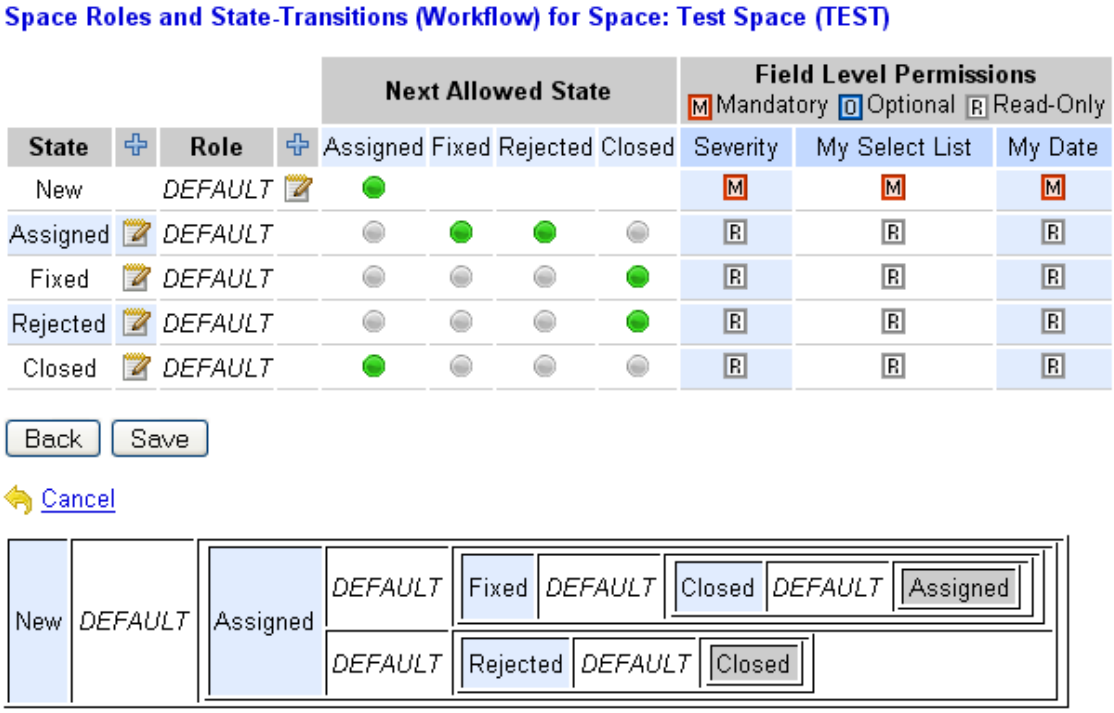
2.3. Custom Workflow

Each tracker project that you create can have a different workflow. JTrac allows for complete customization of the tracker-item lifecycle, right down to the names of each intermediate state. It is possible to create very sophisticated workflows and a visual "map" of the workflow being edited is displayed below the state-transition toggle-buttons to make it easier to manage.

The default workflow shown below is ideal if you simply want to track issues with just two states - Open and Closed. However, with just a few mouse clicks, you can turn this into a complex workflow involving multiple states.

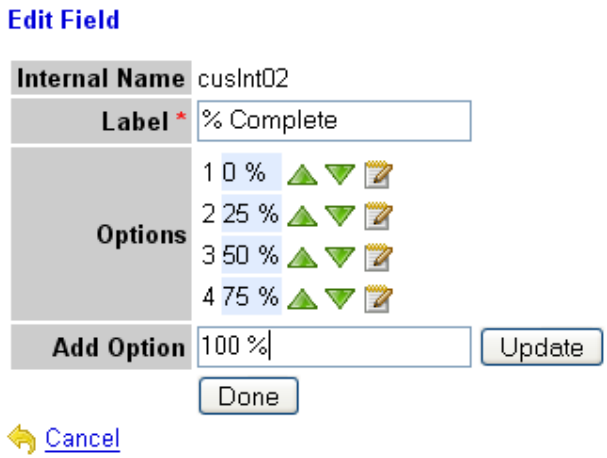


Default workflow - before customization



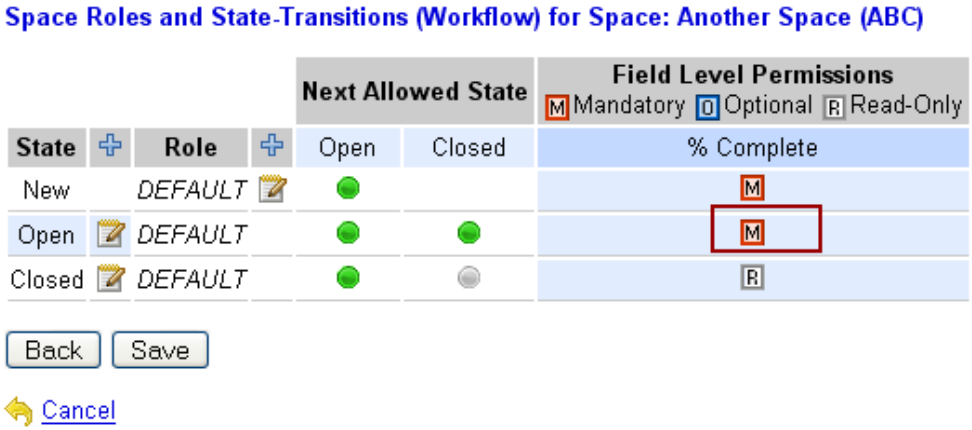
Example of a workflow - after customization

You can even configure whether or not changing of values is permitted given the current state within the workflow - and that too at a field-level. This flexibility allows for some interesting possibilities. For example, assume that you want to maintain a "percentage complete" field that can be updated at any time during the life-cycle of an item. The screenshot below demonstrates how you can set up a drop down field called "% Complete":



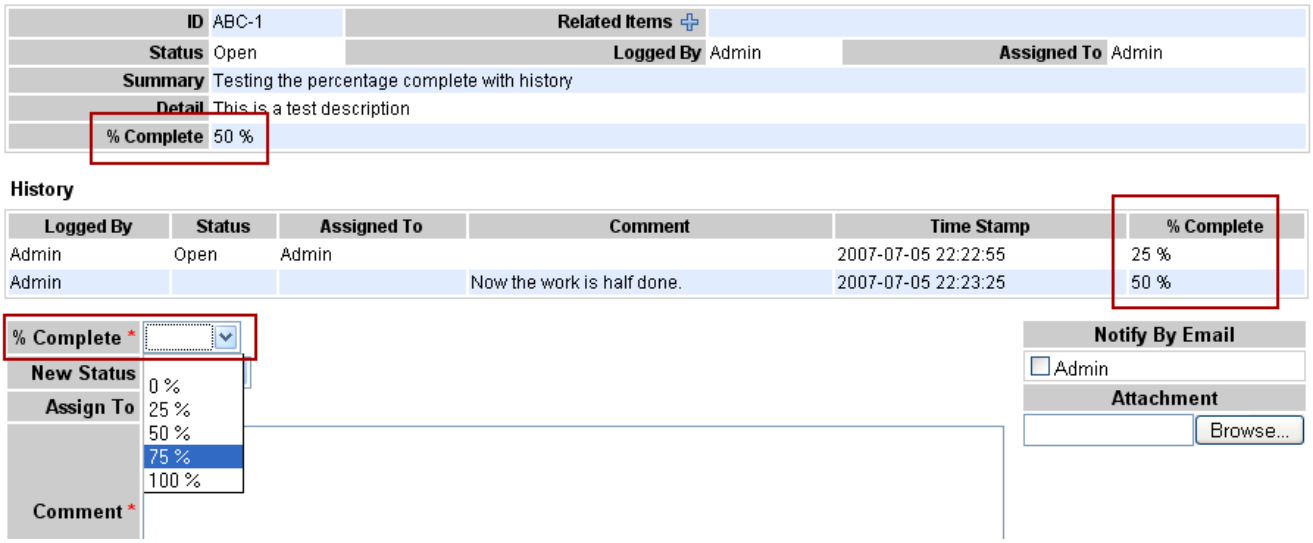
Setting up of a drop down custom field called "% Complete"

Then when customizing the roles and field-level permissions, all you need to do is specify that the "% Complete" field is editable even when the status is "Open".



Making a custom field editable when updating an item (after new item has been created)

This results in the "% Complete" field being available for updating whenever anyone views an item with a status of "Open". Notice also how the "% Complete" field is added as a column to the "History" section so that the complete history of changes to this custom field can be viewed.



Example of a custom field that can be updated even when status is "Open"

2.4. Detailed History View

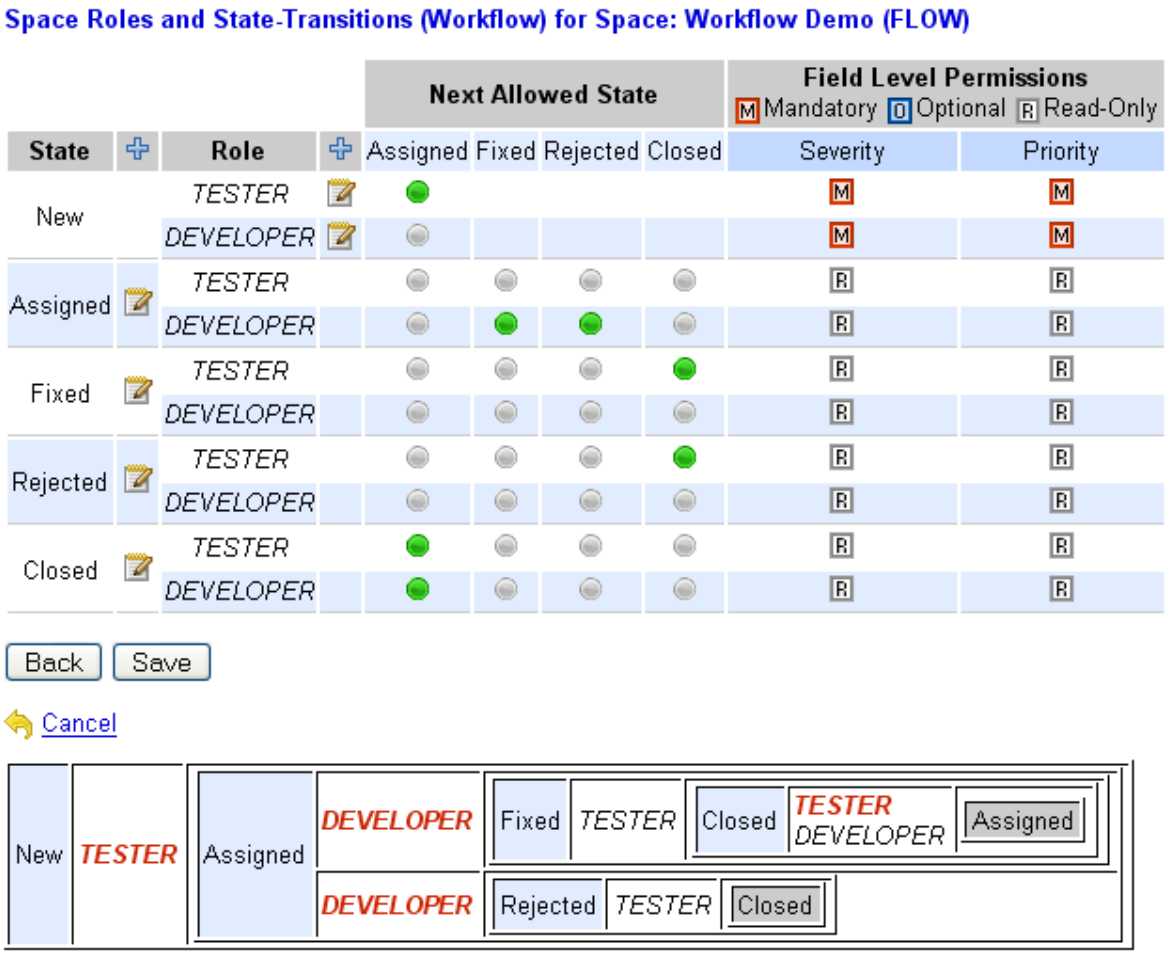
JTrac provides a very detailed history view that completely captures all comments or status changes for an item. The history can be seen when viewing an item as shown above.

2.5. Attachments Support

You can upload file attachments at the time of creating of an item or as many times as required after an item has been created.

2.6. Custom Roles

JTrac customization does not stop at workflow - you can also define different roles for each tracker project that you set up. This allows for power and flexibility. For example you could enforce that a "DEVELOPER" role can only mark "Assigned" items as "Fixed" and that only a "TESTER" role has the power to mark a "Fixed" item as "Closed". You can even disallow selected roles from being able to create new items. JTrac gives you complete control over the workflow and you can easily tweak it to fit your existing process rather than the other way around.



Example of workflow where each role has clearly defined responsibilities.

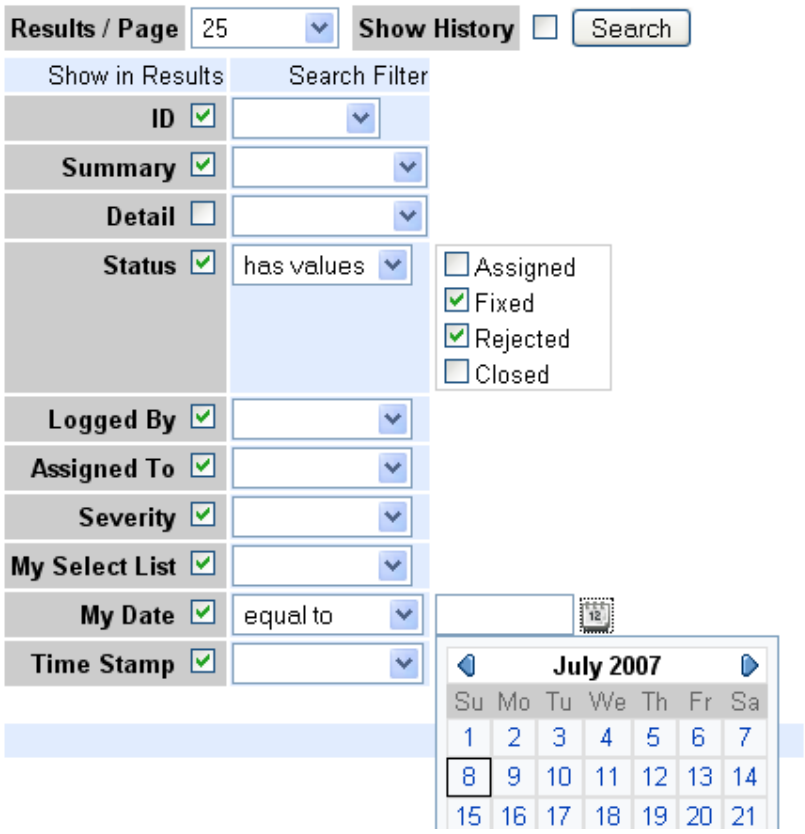
In the example above, the roles highlighted in red on the visual "map" of the workflow being edited mean that at these points in the workflow - a user with the given role can assign only to other roles. So in this case, the TESTER can assign a New item only to a DEVELOPER role and not another TESTER. And when an item is in the "Assigned" state, a DEVELOPER can only assign to a TESTER when changing the status to "Fixed" or "Rejected".

Field-level permissions can be mapped to roles which allows for even more flexibility. For example, you can set a project up so that only a "MANAGER" can change the "Severity" level after a bug is submitted. JTrac makes it possible for you to define very complex workflows, complete with both Status and Role dependent field-level permissions, and you can do all this within a single admin screen.

2.7. Search Custom Fields

The intuitive search screen (shown below) allows for easy execution of common queries such as find all items "logged by" or "assigned to" a user (or users). JTrac allows for filtering even on custom fields.

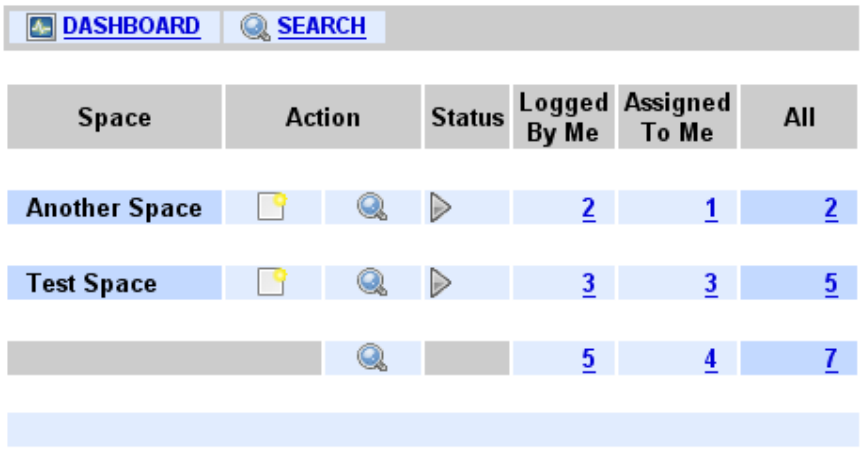
You can navigate back to the search setup screen from the search results screen and have the search parameters still active. This is useful when trying to tweak complex queries by trial and error.



The search screen - which allows filtering on custom fields

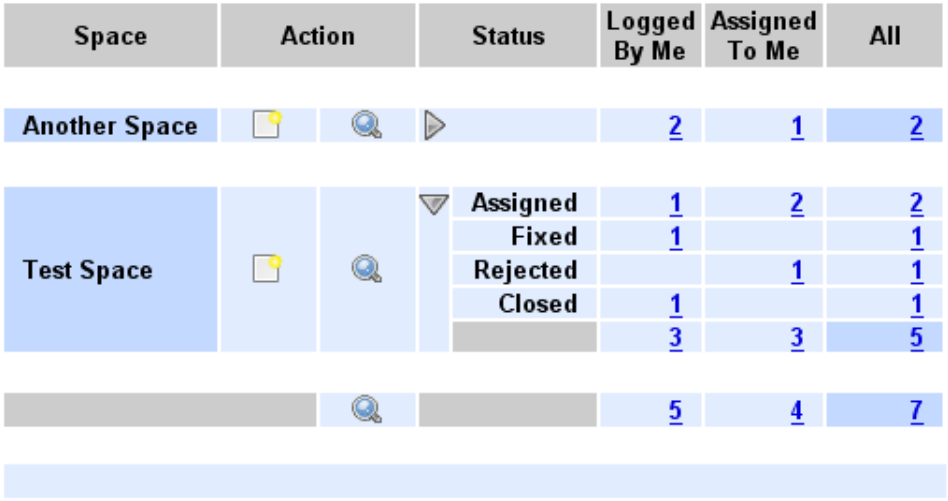
2.8. Dashboard

The default view when you log into JTrac is a very handy dashboard view that provides the most frequently used statistics at a glance. You can click on any of the numbers to immediately bring up the search results view listing the items matched. For example, clicking on the count of items "Logged By Me" would immediately display the list of items initiated by the currently logged in user.



Statistics at a glance and search results available with just one click

For each Space, you can also drill-down to a finer level of detail by clicking on the right-arrow 'expand' icon. This expands the dashboard rows so that you can see a break-up of item counts, not only by whether 'logged by me' / 'assigned to me' - but against the custom workflow states defined for the Space as well. Again, it is possible to directly navigate to the items by clicking on the dashboard numbers.



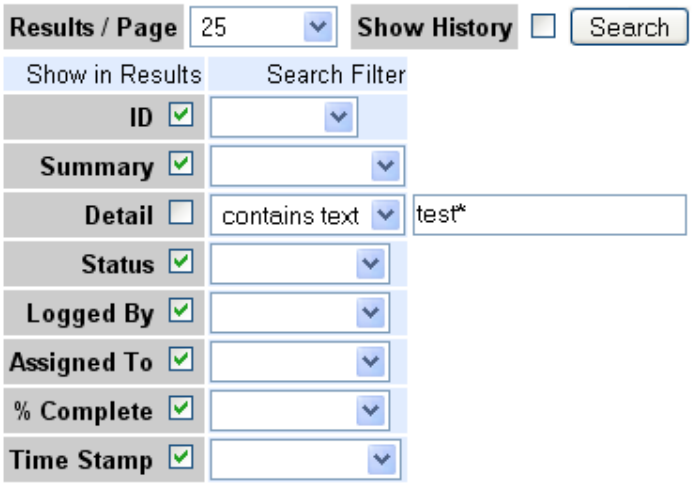
Expanded statistics for a Space showing counts by workflow status

2.9. Search Across All Projects

In addition to be able to search within a single space (project) JTrac allows for searching across all (or a sub set of) projects that a user is mapped to. For example, in the screenshot of the dashboard above, there are two projects. The "SEARCH" link at the top of the page (as well as the "search" link on the "totals" row) would bring up the search wizard that can search across all (or a subset of) the spaces assigned to the currently logged in user.

2.10. Full Text Search

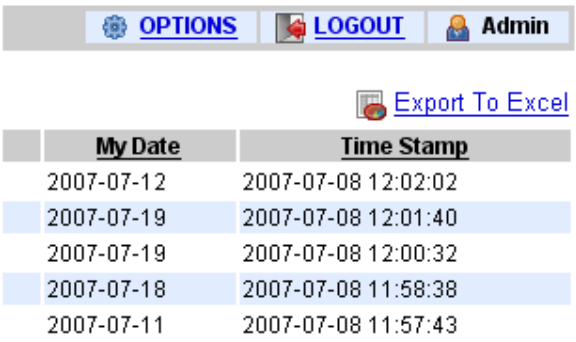
The summary and even the detailed descriptions of items are indexed and can be searched. Note that if you are upgrading or migrating databases, you may need to "re-index" which is an option available to admin users.



Lucene powered full text search with wildcard support

2.11. Export to Excel

JTrac is able to export the results of any search (or all items in a Space) as an Excel Sheet. The link for this appears at the top right of the search results screen.



Download search results as an excel sheet

One the data is in a spreadsheet, it is very easy to analyze data and do things like creating pivot-tables and charts. Many users use the 'export to Excel' feature to fulfil custom reporting requirements.

2.12. Simple Navigation

Most users will use only five screens in JTrac: Dashboard, Create, View, Search and Results. Navigation has been designed with usability and simplicity in mind. A screenshot of the search results screen is shown below.

DASHBOARD | My Project | NEW | SEARCH | OPTIONS | LOGOUT | Peter Thomas

137 Records Found << 1 2 3 4 5 6 >> Export To Excel

ID	Summary	Status	Logged By	Assigned To	Severity	Module	Type	Priority	Time Stamp
MYPROJ-137	User Admin - Percentage exceeds 1.0	Assigned	John Smith	Bill Gates	Major	TMS	Issue	High	2007-03-29 16:50:53
MYPROJ-136	Adding a project to a user	Closed	John Smith		Major	TMS	Issue	High	2007-03-29 16:44:09
MYPROJ-135	User Admin page - Enable and Disable button not working	Closed	John Smith		Major	TMS	Issue	Highest	2007-03-29 11:56:11
MYPROJ-134	user created twice when creating new user	Closed	John Smith	Linus Torvalds	Major	TMS	Issue	Highest	2007-03-28 17:13:01
MYPROJ-133	CodeReview	Assigned	Peter Thomas	Peter Thomas	Major	TMS	Task	High	2007-03-28 10:21:37
MYPROJ-132	Knowledge Sharing Session on WebServices in WBS.	Assigned	Peter Thomas	Naruto Izumaki	Major	WBS	Task	High	2007-03-28 10:18:22
MYPROJ-131	Delete button for the Project members	Closed	John Smith		Major	TMS	Issue	High	2007-03-27 16:33:40
MYPROJ-130	National Holiday - Same value twice	Closed	John Smith	John Smith	Major	TMS	Issue	High	2007-03-27 12:58:31
MYPROJ-129	Editing National Holidays	Assigned	John Smith	Jane Doe	Minor	TMS	Issue	Low	2007-03-27 12:45:39
MYPROJ-128	Addning National Holiday	Closed	John Smith	John Smith	Major	TMS	Issue	High	2007-03-27 12:39:17
MYPROJ-127	Project Admin - Disable project/ Enable project	Closed	John Smith	John Smith	Major	TMS	Issue	Highest	2007-03-27 12:03:56
MYPROJ-126	Date validation for user details page	Closed	John Smith		Minor	TMS	Issue	Low	2007-03-27 11:44:11
MYPROJ-125	Editing a project that has been disabled	Closed	John Smith	John Smith	Major	TMS	Issue	High	2007-03-27 11:12:01
MYPROJ-124	Search results on User Admin Page	Closed	John Smith	John Smith	Major	TMS	Issue	Medium	2007-03-23 17:43:59
MYPROJ-123	User Admin - Add Project , Project added twice	Closed	John Smith		Major	TMS	Issue	High	2007-03-23 14:25:42
MYPROJ-122	User Admin - validation	Closed	John Smith		Major	TMS	Issue	Medium	2007-03-23 14:20:36
MYPROJ-121	Hieght of the Holidays display box	Closed	John Smith	John Smith	Minor	TMS	Issue	Low	2007-03-22 14:55:57
MYPROJ-120	Project Hyperlink on user admin page	Closed	John Smith		Major	TMS	Issue	Medium	2007-03-22 12:46:25
MYPROJ-119	Name field - user admin page	Closed	John Smith		Major	TMS	Issue	High	2007-03-22 12:35:59
MYPROJ-118	password validation on user admin page	Closed	John Smith		Major	TMS	Issue	Medium	2007-03-22 12:33:09
MYPROJ-117	Vacation Request Mail - user page	Closed	John Smith		Major	TMS	Issue	High	2007-03-22 12:29:12
MYPROJ-116	Daily Working time - Decimals entered	Fixed	John Smith	Jane Doe	Major	TMS	Issue	High	2007-03-22 12:26:06
MYPROJ-115	user Admin page - Add project	Closed	John Smith		Major	TMS	Issue	High	2007-03-22 11:51:32
MYPROJ-114	Search page - user	Assigned	John Smith	Bill Gates	Major	TMS	Issue	High	2007-03-22 11:43:57
MYPROJ-113	Edit user - Save	Closed	John Smith		Major	TMS	Issue	High	2007-03-22 11:41:46

The search results screen

2.13. Cross Referencing of Items

You can establish bi-directional links between items. For example you can mark an item as being a "duplicate of" another item. You can easily navigate to related items (and back) when viewing any particular item.

The option to add related items is available when viewing an item after it is created. The navigation is designed so that you have the option to execute a search before selecting an item to cross-reference.

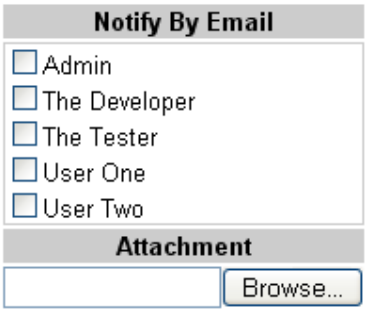
2.14. Read-Only Access

When setting up a Space you can define whether this space can be viewed by users without having to log-in.

Specify if read-only anonymous access is allowed per-space

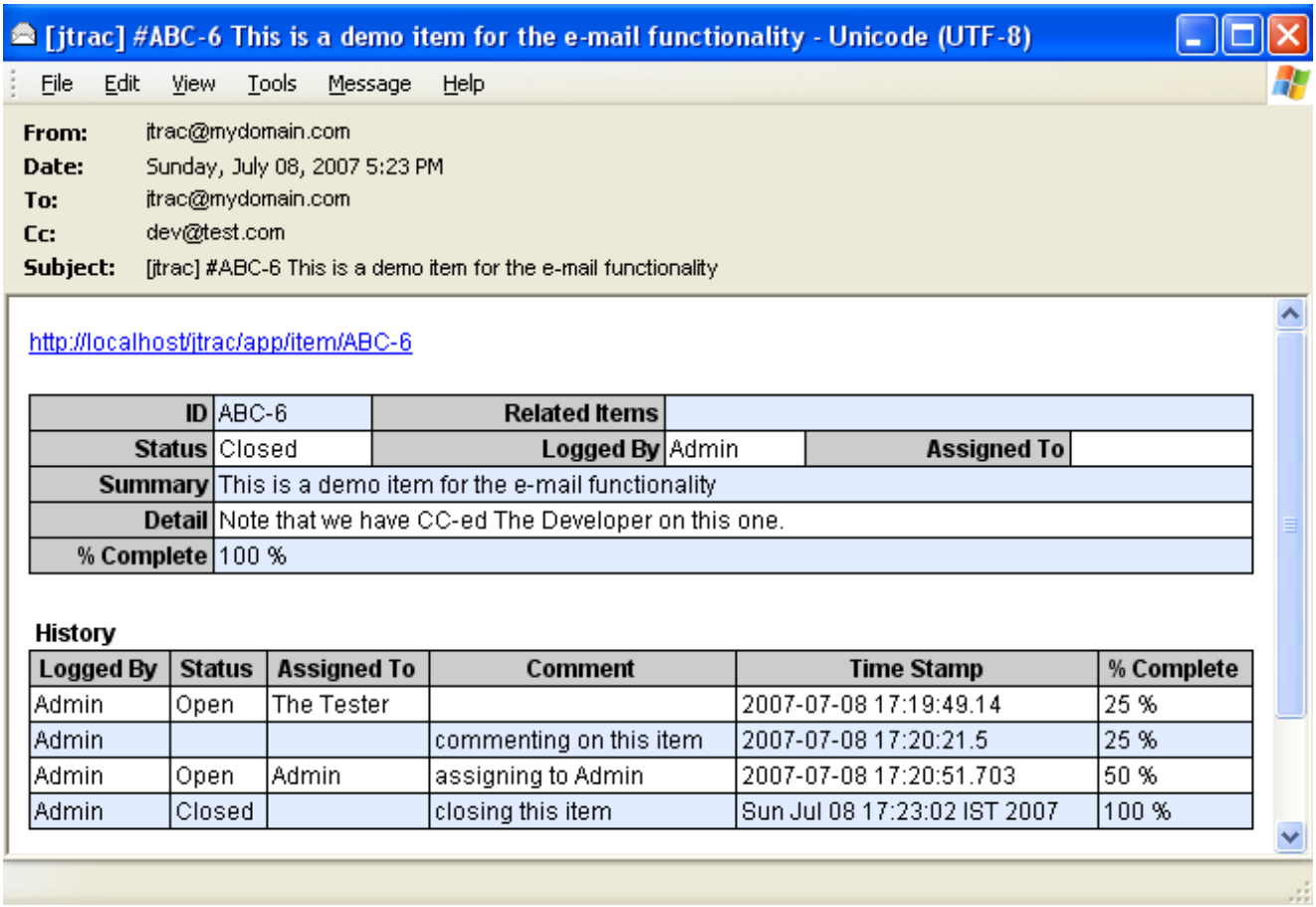
2.15. E-mail Integration

All status changes for an item will trigger e-mail notifications by default. You can choose to add multiple users into a "notify list" at the time of raising a new item or updating the history for an item.



Add multiple users to a "notify list" while creating or updating an item

The e-mail content is very nicely formatted HTML and is laid out in the same way as how the screen looks within JTrac when viewing an item. Users do not need to get used to a different or possibly less pleasing format such as plain-text.



Example of an e-mail sent by JTrac

Admin users can configure the SMTP / e-mail server that JTrac should use in the "Manage Settings" page. This page can be accessed from the "OPTIONS" menu. JTrac also can handle an e-mail server that requires authentication or even a secure connection over HTTPS.

2.16. Multi-Language Support

JTrac has complete internationalization support and you can change the user interface language instantly by editing your user-profile. Adding a translation is very easy and does not require any Java development expertise. See this section of the developer guide for more details and please consider contributing translations to the JTrac project.

JTrac has already been translated into the following languages:

- German - de
- Greek - el
- Spanish - es
- Spanish (Argentina) - es_AR
- Spanish (Mexico) - es_MX
- French - fr
- Italian - it
- Japanese - ja
- Dutch - nl
- Polish - pl
- Portugese (Brazil) - pt_BR
- Russian - ru
- Chinese (China) - zh_CN
- Chinese (Taiwan) - zh_TW

2.17. LDAP / CAS authentication support

JTrac has built-in support for authenticating user credentials against an LDAP or Active Directory server. Please refer the LDAP section of the installation guide for more details.

JTrac also has support for integrating with a Central Authentication Service (CAS) installation. The install guide has more details on how to configure single sign-on using CAS.

Chapter 3. Installation

3.1. Prerequisites

All you require in order to start using JTrac is Java 5 (or higher). Only a JRE (Java Runtime Environment) is required and not the full-blown JDK (Java Development Kit).

There is a good chance that Java is already installed on your system and you can verify whether it is ready to run JTrac by opening a command prompt and typing the following command:

```
java -version
```

If the system responds with a java version that is 1.5 or greater, you are ready to download and run JTrac.

You can download Java from here: http://www.java.com/download.

3.2. Quick Installation

Once you have Java you can start using JTrac right away because you do not need to configure a web-application server or a database. JTrac embeds both of these (Jetty and HSQLDB) to make it easy for you to evaluate JTrac.

- First, download the latest release from the JTrac downloads area
- Extract the zip file into a directory of your choice
- Double-click or run "start.bat" to start JTrac
- Point your web-browser to http://localhost/jtrac
- Sign on with the user name "admin" and password "admin"
- Start using JTrac

You can use the "stop.bat" script to stop the application. The "start.bat" file configures the Jetty server to use port 80 which you can easily change by editing "start.bat". You may need to do this if for example JTrac does not start because a web-server (like Apache or IIS) is already running on port 80.

Remote users should be able to access JTrac over the network by using the machine name instead of "localhost" in the URL shown above. If there is a problem, it could be due to a firewall running on the machine where JTrac is installed - and you may have to reconfigure things so that for example "java.exe" is not blocked.

3.2.1. Notes for Linux / Unix users

If you are not on Windows, you may want to rename the start and stop *.bat files to e.g. *.sh for Linux (*.bat is reported to work for Ubuntu). The commands within the batch files will work unchanged as long as Java has

been installed correctly i.e. "java" is in the PATH and it is the expected 1.5 or greater version. You may need to do things like apply executable permissions to the batch files on Linux, e.g. "chmod +x *.bat". And also - unless you have "root" permissions, you may face problems starting services on port 80 etc.

3.2.2. Security and Backup

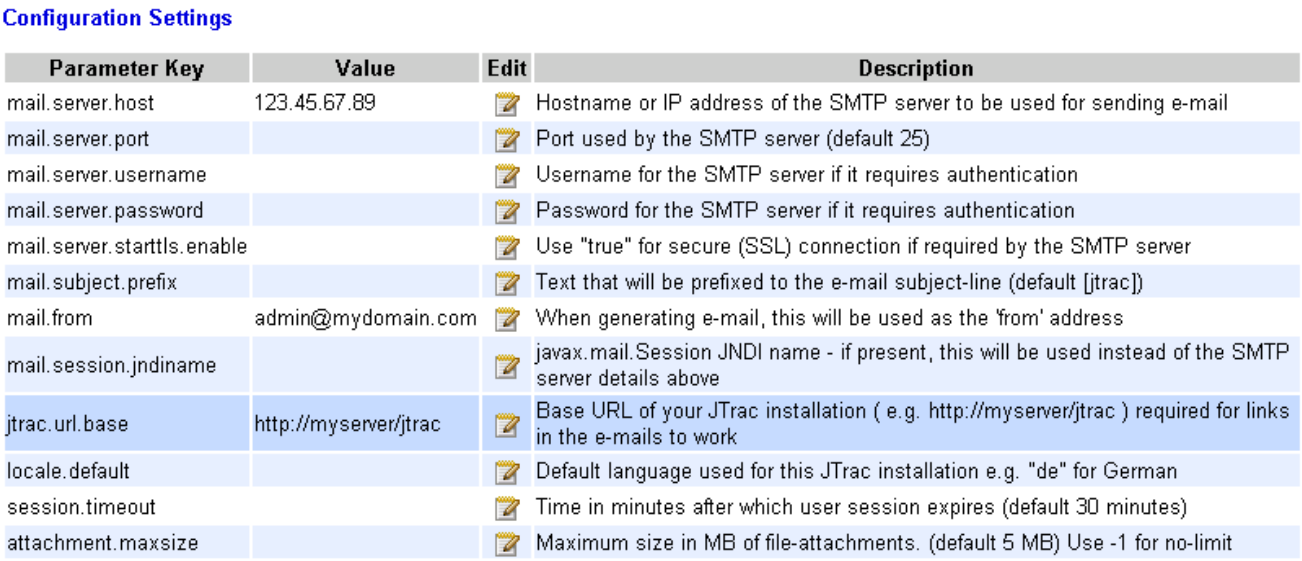
For security reasons, it is best that you change the default "admin" password after installing JTrac, especially if you are going to make the application accessible on a public-facing web-site over the internet.

JTrac by default will use a "data" directory that will be automatically created within the root folder "jtrac". The data directory will contain the files needed by HSQLDB (the embedded database) as well as hold uploaded attachments. When using JTrac in production, this is the location that you should consider backing-up on a regular basis.

3.3. Settings

For e-mail notifications to work, an SMTP server needs to be configured in JTrac. Users who have the ADMIN role will be able to access the JTrac configuration settings screen from a link on the "OPTIONS" screen. Note that you will need to provide values for the "mail.from" and the "jtrac.url.base" properties also, for things like the hyperlinks within e-mail to work properly. JTrac also supports mail servers that require authentication or secure connections using SSL. You can refer the descriptions of the available configuration property settings available from the settings screen as shown below.

If you have trouble getting e-mail to work, one of the things to watch out for is whether there is any firewall blocking the communication between JTrac and the mail server - for example you may have a firewall runnning on the machine where JTrac is installed. It is very common for mail servers to be configured to prevent mail-relay requests from unknown applications or IP addresses so you may need to check with your mail server administrator.



Configuration Settings

Parameter Key	Value	Edit	Description
mail.server.host	123.45.67.89		Hostname or IP address of the SMTP server to be used for sending e-mail
mail.server.port			Port used by the SMTP server (default 25)
mail.server.username			Username for the SMTP server if it requires authentication
mail.server.password			Password for the SMTP server if it requires authentication
mail.server.starttls.enable			Use "true" for secure (SSL) connection if required by the SMTP server
mail.subject.prefix			Text that will be prefixed to the e-mail subject-line (default [jtrac])
mail.from	admin@mydomain.com		When generating e-mail, this will be used as the 'from' address
mail.session.jndiname			javax.mail.Session JNDI name - if present, this will be used instead of the SMTP server details above
jtrac.url.base	http://myserver/jtrac		Base URL of your JTrac installation (e.g. http://myserver/jtrac) required for links in the e-mails to work
locale.default			Default language used for this JTrac installation e.g. "de" for German
session.timeout			Time in minutes after which user session expires (default 30 minutes)
attachment.maxsize			Maximum size in MB of file-attachments. (default 5 MB) Use -1 for no-limit

The JTrac settings screen where you can configure an e-mail / SMTP server

JTrac also can lookup and use a JNDI mail session (javax.mail.Session) if required. If you provide a JNDI name, JTrac will ignore the SMTP server details provided.

3.4. Using only the WAR file

JTrac is designed so that you can be up and running by just deploying the WAR file into your existing web-application server. Note that you need a servlet 2.4 compliant container. For example, if you have Apache Tomcat 5.5.X, you can either copy the WAR file into the [tomcat.home]/webapps folder or choose to upload the WAR file through the Tomcat console.

JTrac saves database information and uploaded attachments into a directory on the server. This directory is logically called the "jtrac.home". When you do a "quick-install" of JTrac by just deploying the WAR file, JTrac creates a ".jtrac" folder in the user home folder and uses that location as the home directory. This is great for quickly evaluating JTrac but when you actually use JTrac in production you should configure the JTrac home directory as a location that can easily be backed-up for example.

3.5. Custom Installation

To customize the location of "jtrac.home" you have the following three options. On startup, JTrac tries to detect the value of "jtrac.home", and checks the following configuration options in the order listed below.

3.5.1. Edit the "jtrac-init.properties" file

If you explode (unzip) the WAR file, you can edit the /WEB-INF/classes/jtrac-init.properties file and customize the location of "jtrac.home". Note that you can choose to deploy JTrac as an exploded-war and it is not mandatory that you re-package (zip) the WAR file after editing "jtrac-init.properties".

3.5.2. Set a Servlet context parameter

For example you can specify the jtrac.home using the Tomcat administration console or as follows using a <Context> parameter.

```
<Context docBase="${catalina.home}/jtrac/jtrac.war">
    <Parameter name="jtrac.home" value="C:/data/jtrac_home"/>
</Context>
```

3.5.3. Set a System / JVM parameter

If you can easily set a system property called "jtrac.home" within the environment of your application server then you don't need to edit the WAR file. Please look at the contents of "start.bat" to see an example of how the appropriate JVM (Java Virtual Machine) parameters can be set when deploying JTrac. For example, if you are using Apache Tomcat 5.5.X, you can set an environment variable called JAVA_OPTS before invoking the startup script (startup.bat / startup.sh). Here is an example for Tomcat 5.5.X on Windows:

```
set JAVA_OPTS=-Djtrac.home=C:/data/jtrac_home -Dfile.encoding=UTF-8
```

If "jtrac.home" has not been configured in any of the above ways, JTrac will create a folder called ".jtrac" in the user home directory and proceed to use that as "jtrac.home".

3.6. Using a different database such as MySQL

You can configure the database that JTrac uses by editing the "jtrac.properties" file that JTrac expects within the "jtrac.home" folder. Note that if JTrac does not find a "jtrac.properties" file in the expected location, JTrac creates a fresh one which is pre-configured for HSQLDB. The contents of this file are as follows:

```
database.driver=org.hsqldb.jdbcDriver
database.url=jdbc:hsqldb:file:${jtrac.home}/db/jtrac
database.username=sa
database.password=
hibernate.dialect=org.hibernate.dialect.HSQLDialect
hibernate.show_sql=false
```

You can change the database values and the hibernate dialect to match the settings of a database that you have already. When JTrac starts, it will connect to the database and create the schema if one does not already exist.

If you edit the "jtrac.properties" file, ensure that there are no trailing spaces in the entries shown above (especially when cutting and pasting from somewhere). This can save you a lot of frustration!

Don't forget to make the database driver available for the web-application. If you are using the bundled Jetty web-app server, this is as simple as copying the jar file into the "lib" folder along with the jetty jar files. If you are using Tomcat, you can either place the jar file under "shared/lib" or within the "/WEB-INF/lib" folder of the "jtrac" (exploded) WAR itself.

Given below is a sample jtrac.properties file for MySQL. Using the default "root" user is obviously not advisable from a security point of view and you should try and have a separate user for the database used by JTrac.

```
database.driver=com.mysql.jdbc.Driver
database.url=jdbc:mysql://localhost/jtrac
database.username=root
database.password=
hibernate.dialect=org.hibernate.dialect.MySQLDialect
hibernate.show_sql=false
```

For MySQL, do not use the "autoReconnect=true" parameter in the MySQL URL, this is not recommended by the MySQL team. The Apache DBCP connection pool used internally by JTrac is configured to refresh stale database connections.

The default query to test idle database connections is "SELECT 1". This has been reported to not work for databases such as Derby and Progress because they consider this invalid SQL. You can customize the validation query by adding a "database.validationQuery" property in "jtrac.properties. If you set the validation query to an empty string, this effectively disables the periodic testing of idle database connections that Apache DBCP performs.

3.7. Using a JNDI Datasource

You can provide a JNDI datasource name as an additional property called "database.datasource.jndiname" in the "jtrac.properties" file. JTrac will ignore the other database properties in this case.

3.8. Configuring LDAP Authentication

A couple of extra entries in "jtrac.properties" are required in order to configure LDAP authentication. JTrac will attempt LDAP authentication if an "ldap.url" config entry is present.

```
ldap.url=ldap://myldaphost
ldap.searchBase=DC=foo,DC=bar,DC=org
```

The following additional entry is recommended if you are using Active Directory. This switches on an LDAP optimization that is specific to Active Directory.

```
ldap.activeDirectoryDomain=MYDOMAIN
```

The benefit of LDAP authentication is that users can sign-on to JTrac by using their existing LDAP credentials (username and password). At the moment, LDAP support is limited to authentication and not authorization. Users still have to be created by an administrator and mapped to the relevant Spaces using the JTrac administration screens before they can start using JTrac.

3.9. Integrating with CAS for Single Sign On

In order to integrate JTrac with CAS you have to edit a couple of XML files within the "WEB-INF" folder ('web.xml' and 'applicationContext-acegi-cas.xml'). The changes that you need to make are described below.

3.9.1. Changes to web.xml

- Change "applicationContext-acegi.xml" to "applicationContext-acegi-cas.xml". This appears towards the start of the file in the 'contextConfigLocation' context-param.
- Uncomment the filter and filter-mapping XML for the 'acegi' FilterToBeanProxy
- Uncomment the servlet and servlet-mapping XML for the 'casProxy' ProxyTicketReceptor. You should also uncomment the associated context-param called 'edu.yale.its.tp.cas.proxyUrl' and customize the value for your environment.

3.9.2. Changes to applicationContext-acegi-cas.xml

In the XML corresponding to bean id "casProxyTicketValidator" - edit the following property values to suit your environment:

- casValidate
- proxyCallbackUrl
- trustStore
- loginUrl
- logoutUrl

For more details and troubleshooting, you can refer to the Acegi documentation on CAS integration.

3.10. Installing as a Windows Service

Note that this approach can reportedly apply to other operating systems. If you have been able to do this successfully on Linux or Mac environments, please consider contributing some documentation on this.

If you choose to use the bundled Jetty web-application server, you can easily install JTrac as a Windows service so that it can start automatically when the host machine is switched on. One of the big advantages of this is that the console window will be invisible and will run in the background. You can therefore avoid the possibility of the Jetty server getting terminated because somebody decided to close "that pesky DOS Window that gets in the way". Having JTrac start automatically without requiring someone to log-in is ideal when you have a server that may be restarted occasionally due to power outages or maintenance.

JTrac can be configured as a Windows service by using the Java Service Wrapper project. For convenience, the JTrac distribution contains a "wrapper.conf" file in the "etc" folder that is pre-configured for the Java Service Wrapper. But first you have to download the Java Service Wrapper distribution separately. Note that the Java Service Wrapper project supports other operating systems such as Linux or Mac also.

Place some of the files from the Java Service Wrapper distribution into the JTrac folder structure as follows:

- From the "bin" folder, copy "wrapper.exe" to the "jtrac" root folder
- From the "lib" folder, copy "wrapper.jar" and "wrapper.dll" into the "jtrac/etc" folder

To install JTrac as a windows service, open a command prompt, change to the "jtrac" directory (where you extracted the JTrac distribution) and run the following command:

```
wrapper.exe -i etc/wrapper.conf
```

This will install a windows service called "JTrac-Jetty". You can now go to the Windows - Control Panel :: Administrative Tools :: Services dialog to start the "JTrac-Jetty" service. Note that this service is configured to automatically start when Windows starts, by default. To remove the service, you can use the following command:

```
wrapper.exe -r etc/wrapper.conf
```

The "wrapper.conf" file in the "etc" folder can be modified in case you need JTrac to run on a different port from the default (80). Look for the "-Djetty.port=80" entry. When running as a service under the Java Service Wrapper, you can look at the "wrapper.log" file in the "jtrac/logs" folder for troubleshooting.

3.11. What to Backup

Although in practice JTrac has been found to be extremely stable in production - to safeguard against hardware failures, you should regularly backup the database and stored attachments. If you are using JTrac with the embedded HSQLDB database, backing up the contents of "jtrac.home" will be sufficient. The HSQLDB data will be stored as files in a directory called "db" under the JTrac home.

If you are using the JTrac distribution as-is, i.e. if you are using the bundled Jetty application server - the JTrac home directory will be called "data" and this is created automatically under the folder where you un-zipped JTrac.

Of course, if you are using a different database you will have to backup the database separately. The "jtrac.home" would still need to be backed up because this is where uploaded attachments are stored.

3.12. Upgrading

Please refer to the "upgrading" section of this document.

Chapter 4. FAQ

4.1. Do we really need another issue tracker?

JTrac can be installed in a few minutes without having to type any commands and you don't even need to set-up a database. Configuring an e-mail server for notifications is something easily done through the user interface. One of the reasons why many people choose JTrac over other open-source issue trackers or bug trackers is because of how easy it is to set up.

JTrac is also written from the ground up to make use of the very best and latest in Java. For example JTrac is developed using JDK 5.0 features, uses Apache Wicket, Spring and Hibernate and has full internationalization support.

JTrac is designed to be able to handle requirements management in the future and this an area where there is a clear lack of open source tools that support the needs of agile development teams. JTrac also has flexible workflow capabilities and offers a rare level of control over field-level permissions.

4.2. What is the JTrac architecture like?

The presentation layer is using the Apache Wicket framework.

The service layer uses the Spring Framework. Also the Acegi Security framework for Spring has been used. JTrac is effectively a "light-weight" Java EE application and any version 2.4 compliant servlet engine is sufficient for deployment.

Hibernate is used for data persistence. This allows for JTrac to be database neutral and any database supported by Hibernate can be used.

4.3. Why start with version 2.0? What about 1.0?

There was an earlier version of JTrac which used to be hosted at https://jtrac.dev.java.net That project was called "jTrac" (with a lowercase 'j') and it was based on Spring JDBC and MS Access. JTrac was completely re-written to use Spring and Hibernate - and the 2.0 version number reflects this.

4.4. What is the username and password when you log in for the first time?

username: admin

password: admin

Of course, it is recommended that you change the default password as soon as possible!

4.5. How can I help?

We welcome code contributions in the form of patches which you can submit here. JTrac has a detailed

developer guide which can get you up and running even if you are new to Java development.

Do consider contributing a translation which is very easy to do and does not require any knowledge of Java at all. You can refer to this section of the developer guide for details.

Read the section on interim builds on how you can help by testing the development builds.

4.6. We are heavy users of JTrac but it is running on the embedded HSQLDB database. Can it cope?

HSQLDB is surprisingly resilient and we have an instance of JTrac running in the place where I work that over a span of 2 years has accumulated more than 2000 items across multiple spaces with around 200 users. The database file has grown beyond 10MB but there are still no signs of problems and the application screens still load fast.

But depending on what you are comfortable with, you can consider moving to another database if your usage is really high. The upgrading section of this document has details on how you can migrate data from HSQLDB to other databases such as MySQL.

4.7. Why does JTrac have a limit on custom fields? I really need more!

JTrac provides an ample number of custom fields which can be used for all kinds of tracking needs. In very rare cases where people run out of custom fields, it is invariably for "free text" fields. The limit on custom fields is primarily for performance reasons especially where filtered searches are involved. So when JTrac introduces the feature of being able to "tag" items, these kinds of things are likely to be covered.

Anyway, if you really, really want more custom fields, the JTrac code is very clean and you can easily customize a version that meets your requirements. A few people have reported having done this and for example - this forum thread can help you get started.

4.8. How do I report bugs or feature requests?

Use the SourceForge trackers here:

- Bugs: http://sourceforge.net/tracker/?atid=825941&group_id=162983
- Feature Requests: http://sourceforge.net/tracker/?atid=825944&group_id=162983

4.9. Why is JTrac not being used as the bug-tracker for this project? You should be eating your own dog food right?

We are just trying to use the SourceForge infrastructure as far as possible and the activity on the SourceForge trackers translates into higher rankings which is important for project visibility. At this point, we would rather not get into the details of hosting an instance of JTrac on the internet.

4.10. Do you have a list of users or references?

Plenty. Have a look at this page: http://j-trac.wiki.sourceforge.net/references

Chapter 5. Roadmap

Have a look at our detailed developer guide and you will find that JTrac has one of the most developer-friendly set-ups among open-source projects. It will take you only a few minutes to be up and running no matter which Java IDE you use. We actively encourage contributions and with your support JTrac can become better than it is today.

Apart from using the forums you can also consider joining the JTrac mailing-list.

5.1. Field-Level "Hide" Permissions

In a future version of JTrac, the ability to even hide fields depending on the role and status will be implemented. The design for this is already in place.

5.2. Nested Items

JTrac will support "nested" items under a first-level "parent" item and the design and database schema for this is already in place. This will allow users to split tasks into sub-tasks and opens up other possibilities. One of the plans in the roadmap is to build JTrac into a full-fledged requirements capture tool complete with test-case management and traceability. JTrac can then be used to manage user-stories or use-cases for agile development teams.

5.3. Custom Validation

Using Beanshell, users should be able to introduce custom routines to validate item data entered by a user. Even conditional validation across fields should be possible.

5.4. Custom Scheduled Jobs

Again using Beanshell, users should be able to define routines to execute periodically, say at a pre-determined time every day. This would allow triggers for events such as slippage of a "due-date" field. This can be combined with e-mail notifications. Also a daily dashboard summary could be mailed out once the scheduler is in place. [As of version 2.1.0 - the scheduler is already in place for those who are working on extending JTrac.]

5.5. Submit By Email

JTrac should be able to monitor an e-mail account and incoming e-mail should trigger creation / updation of items. It would be handy to have attachments submitted through email also.

5.6. Screenshot Capture

Using a Java applet, it should be possible for the user to upload a screenshot directly from the system clipboard. The user should be able to annotate (e.g. highlight a particular area of) the image before uploading.

5.7. Saved Searches

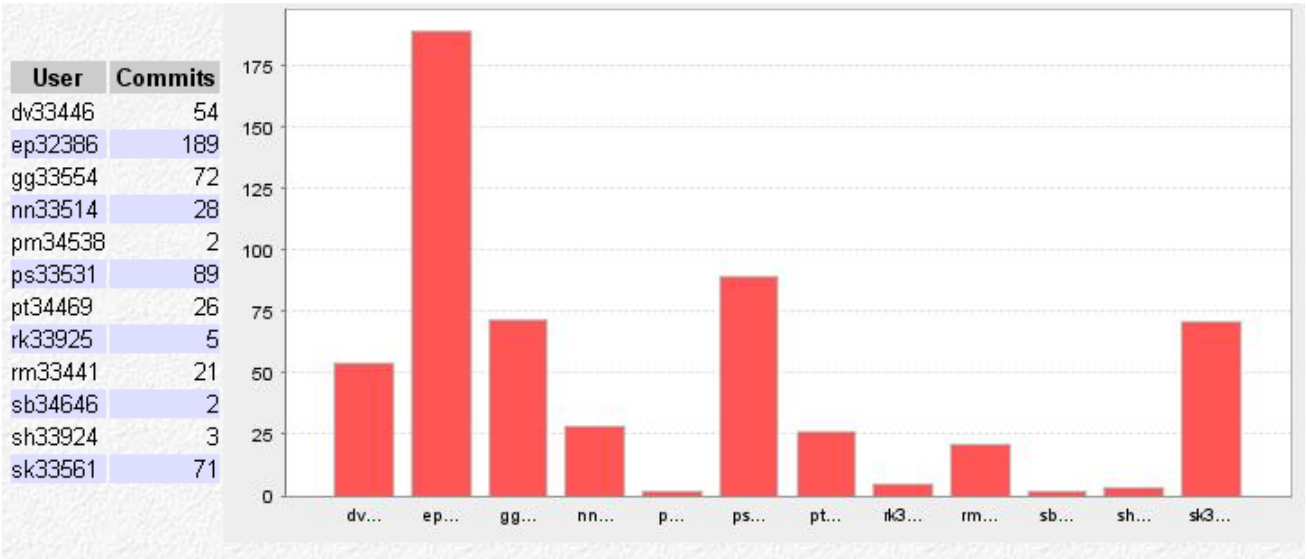
It should be possible for users to save search filters and re-use them.

5.8. Wiki Engine

There is a plan to embed a wiki-engine into JTrac. This will supplement the requirements management road map.

5.9. Subversion Integration

Integration with Subversion has already been implemented using the JavaSVN library and this will enable effective integration of bug reports with commit history in the future. You can try out the existing experimental support by going to the following url of your JTrac installation: http://[hostname]/jtrac/app/svn



User	Commits
dv33446	54
ep32386	189
gg33554	72
nn33514	28
pm34538	2
ps33531	89
pt34469	26
rk33925	5
rm33441	21
sb34646	2
sh33924	3
sk33561	71

Experimental Subversion support: commits per user report

5.10. Tags

A one-to-many data model for being able to tag items "Web 2.0" style is already in place. This would be ideal to track things such as mapping bugs to software release version numbers, associating keywords with items and to do all kinds of other cool stuff.

5.11. RSS Feeds

RSS feeds will be introduced so that you can monitor new items and updates to items.

5.12. Import from other tools

Import from other issue tracking systems (such as BugZilla) to be considered. Once complete, this feature will

allow you to import raw data from a spreadsheet, apply any data cleanup needed, identify and load custom fields / drop down values and finally import the items, creating a new Space and users if required.

5.13. Single Sign On

JTrac uses the Acegi Security framework which supports integration with a Central Authentication Service (CAS) installation. You can refer to the installation guide for how to configure JTrac to use CAS for single sign on. As of version 2.1.0 JTrac supports LDAP authentication out of the box. This needs to be improved by adding an admin screen that can browse and add users from an LDAP server. Authorization using LDAP and allowing users to plug in custom authentication and authorization routines would be good to have as well.

5.14. XML API

JTrac should be able to expose a remote XML API (SOAP or REST) so that core functionality can be invoked / integrated with other systems.

5.15. Eclipse Mylyn integration

Mylyn (previously known as Mylar) integration in the form of a connector (Eclipse plugin) is currently being implemented. The remote API mentioned above is also being developed (REST).

5.16. Time Tracking

The database schema already has support for capturing planned vs actual effort against items. This needs to be implemented and obviously will open up a lot of possibilities.

5.17. Custom Reports

Although JTrac has an 'Excel Export' feature that should suffice for all your reporting and charting needs, it would be good to have a reporting engine that allows the user to create custom reports and even schedule reports to be sent out over e-mail.

Chapter 6. Interim Builds

6.1. Overview

For the convenience of those who would like to test the latest possible builds of JTrac without having to check-out and compile from version-control, binaries in WAR form are periodically uploaded to the JTrac web-site at this URL: http://j-trac.sourceforge.net/files/

You would be able to see the time-stamp of the last uploaded binary when you access the above URL. Subscribers to The JTrac mailing-list would be notified whenever a new binary is available. This is not on a daily basis at the moment and happens only when significant changes or fixes are made in the development version.

If you want to stay updated about the status of JTrac development as well as contribute and brainstorm ideas for incorporation into future releases, please consider joining the JTrac mailing-list at: https://lists.sourceforge.net/lists/listinfo/j-trac-users

6.2. Notes on using Interim Builds

- Interim builds are provided only as WAR files. If you are using the embedded Jetty webapp-server, you have to replace the WAR that came along with the formal downloadable distribution.
- When you are replacing the WAR file in an existing installation it is best that you delete the contents of the webapp-server temporary / work folder before re-starting. Details on how to do this are provided in the "upgrading" section of this document.
- In rare cases, there may be database schema changes. This will be made clear in the upload notification that is broadcast on the mailing-list. If you wish to retain existing data, you will have to run a database upgrade script. Details on how to do this are also provided in the "upgrading" section of this document.
- Even if the database schema has changed, JTrac will re-create the database if one does not exist on startup. So when trying out the interim builds, this may be all that you really want to do. If you are using JTrac bundled with Jetty and / or with the embedded HSQLB database, removing the old database is as simple as deleting the "db" folder in which HSQLDB keeps the data files. Of course, you don't have to disturb your existing data at all since running a fresh instance of JTrac is as easy as extracting the distribution to a new directory.

Chapter 7. Upgrading and Database Migration

Upgrading to higher versions of JTrac is usually as simple as replacing the WAR file and deleting the web-app server temporary / work folder before re-starting. In some rare cases the database schema may have changed and a script for updating the database would then be provided. It may also happen that upgrading the Jetty version would be recommended in case you happen to be using the bundled Jetty webapp-server for deploying JTrac.

7.1. Deleting webapp temporary files

In case you are using the bundled Jetty server in the JTrac distribution, you should delete the contents of the folder called "work". Note that Jetty expects a folder called "work" to exist when it starts, so ensure that you leave the empty "work" folder in place.

If you are using another application server please follow the steps for removing temporary files corresponding to the server that you are using.

7.2. Upgrading the Database

If the JTrac release notes mention database schema changes, you need to additionally run an upgrade script before restarting the webapp-server with the new WAR in place. Upgrade scripts come in the form of Beanshell scripts (*.bsh) and would be always released within the "etc" folder of the JTrac downloadable distribution.

These files are tested on HSQLDB and may need minor changes to work with other databases. No re-compilation is needed, therefore making it extremely easy for you to tweak them if required. Of course it is strongly recommended that you backup your old data before an upgrade. Please use the JTrac forums for help and for helping other users with your experiences.

If you are shadowing the development builds you would be able to obtain the latest database upgrade scripts from here: http://j-trac.svn.sourceforge.net/viewvc/j-trac/trunk/jtrac/etc/ Look for files that end with ".bsh" - e.g: "jtrac-hsqldb-2_0rc3-to-2_0.bsh"

Steps to follow:

- First, stop the server, JTrac should not be connected to the database.
- The "java" command should be available on the command line.
- Get the latest beanshell jar file (~300 KB) from here: http://www.beanshell.org/bsh-2.0b4.jar
- Get the right JDBC driver for your database. If you have been using the embedded HSQLDB database, hsqldb-x-x-x.jar will be available in the jtrac war file within "lib".
- Edit the connection params (jdbc URL, username, password) to match your database. For HSQLDB, to make it easier - you can copy the beanshell jar and the hsqldb jar into the "db" folder of "jtrac.home". The db folder will contain a "jtrac.script" file. There may also be a "jtrac.log" file. So when you are in the same directory as the HSQLDB data files, the URL is "jdbc:hsqldb:file:jtrac". Otherwise, replace the "jtrac" part in the URL with the relative or absolute path of where the "jtrac.script" file exists on your file-system.
- Execute the beanshell script from the command line. The database driver should be added to the classpath

along with the beanshell jar. This is how the command would look like:

```
java -cp bsh-2.0b4.jar;hsqldb-1.8.0.1.jar bsh.Interpreter jtrac-hsqldb-2_0rc3-to-2_0.bsh
```

- If there were no errors, you can now upgrade the WAR (don't forget to delete the temporary files) and restart.

7.3. Upgrading Jetty

If you are using the embedded Jetty web-app server, you can also choose to upgrade the JAR files that make up Jetty. This is optional, but is as simple as overwriting the contents of the "lib" directory with the corresponding files within the new distribution. You can easily check if the Jetty related JAR file versions have changed because the file-names clearly include the version numbers.

7.4. Database Migration

Using the Beanshell based approach above, it is relatively straightforward to port data into another database. This is useful especially in those cases where you start off using JTrac with the embedded HSQLDB database and then decide to scale up and move to something like MySQL.

First, you should start the application re-configured to point to the new database and let JTrac do the hard work of creating all the database tables. It is not necessary for Jtrac to be shut-down before proceeding with the next step, but don't log-in yet!

A Beanshell script called "jtrac-hsqldb-to-mysql.bsh" is available in the "etc" folder of the JTrac distribution. You can execute this just like described in the previous section on upgrading. Note that you will need to have both the database drivers on the classpath in this case. You should of course edit the database connection parameters (for database 1 and database 2) to suit your environment.

This particular script is extremely generic and it should be possible to use this to migrate across combinations of databases other than HSQLDB and MySQL.

Chapter 8. Developer Guide

8.1. Pre Requisites

- JDK 5.0 (or higher)
- Maven 2.X
- Your Favourite IDE (e.g. NetBeans)
- Access to the internet (or a Maven 2 repository)

The JTrac development environment uses Maven 2 only for dependency management and generating and deploying the JTrac website hosted at SourceForge. Almost all other development actions (clean, compile etc) are fired through a detailed Ant script that is able to use the dependency information managed by Maven. Using Maven ensures that no binaries (JAR files) need to be checked into version control at all.

You can get some idea about the details and rationale of the custom Ant + Maven integration approach from this blog post.

But if you are comfortable using Maven, the pom.xml is Maven 2 compliant which means that you can directly build the war file without Ant. You can even choose to use the maven-jetty-plugin by uncommenting the section in pom.xml that declares this.

Although NetBeans is preferred by the JTrac developers, depending on an "IDE Neutral" Ant script ensures that you would easily be able to use Eclipse (or any other IDE of your choice) without any problems.

If you get stuck or have any questions about setting up and getting started, feel free to use the JTrac forums for help.

8.2. Check Prerequisites

Maven 2 should be in your "PATH". You can test this out by opening a command prompt and trying to run the command "mvn". If this is set, you are ready to move to the next step.

8.3. Download / Extract Source

Download the source code from SourceForge and extract it to a convenient location. Note that the source code is a separate download from the main (binary) and the file name will be of the form "jtrac-src-X.Y.Z.zip".

You could choose to check out the source code directly from the JTrac SourceForge Subversion repository. The Subversion URL for the JTrac source code is as follows:

```
https://j-trac.svn.sourceforge.net/svnroot/j-trac/trunk/jtrac
```

If you are behind a corporate firewall that requires NTLM authentication, you can try a tool called NTLMAPS

to check out from Subversion.

If you are using Windows, we recommend TortoiseSVN as a Subversion client.

8.4. Customize Ant Build Properties File

The downloaded source code should contain a sample "build.properties" file or you can look at the XML comment provided at the top of "build.xml" for the structure. Actually the couple or so entries are optional and you will need them only if you want to use Tomcat (instead of Jetty) or use the JMeter script available for load-testing.

8.5. Generate Dependencies Properties File

Open a command prompt and change to the "jtrac" folder. Run the following command:

```
mvn antprops:generate
```

This step will not only download all the required JAR files (which may take time only for the first time), but also generate a standard properties file that will contain all the information required for the Ant build script to operate. The file generated is called "build-deps.properties"

Note that you need to perform this step every time the "pom.xml" file changes. This happens rarely and typically when newer versions of JTrac dependencies (e.g. Spring and Hibernate) are available. Keep watching for changes in the versions of JTrac dependencies or specifically changes to "pom.xml" to avoid any problems.

If the command does not work, maybe you are not connected to the internet. Here's a tip that may help if you are behind an HTTP proxy. This has been reported to work even when the proxy requires NTLM authentication. You can append a parameter when running Maven commands as follows:

```
mvn antprops:generate -Dhttp.proxyHost=172.19.56.56
```

Note that if your proxy port is different from the default 80, you will have to add an "http.proxyPort" parameter as well.

8.6. Import project into your IDE

If you are a NetBeans user you are ready to build and run JTrac! Just use the "Open Project" option from the "File" menu, browse to and select the "jtrac" folder. You should be able to open it as a valid NetBeans project.

If you want to use Eclipse, that's easy as well. First you have to run the following command and Maven will then generate the Eclipse project descriptor files for you.

```
mvn eclipse:eclipse
```

```
```

That should create the ".project" and the ".classpath" files. Now its just a matter of importing "Existing Project into Workspace" within Eclipse. Note that the Eclipse project descriptor files also need to be re-generated when JTrac dependencies change, i.e. when "pom.xml" changes. Make sure you do this also if you ever run "mvn antprops:generate".

For Eclipse to work with Maven 2, you have to have a classpath variable called "M2_REPO" set correctly. Refer the Maven 2 documentation for details on how to do this.

While debugging, if you would like to step through the source of dependencies as well, you can tell Maven to download sources for dependent jar files. Note that source may not be available for some jars.

```
mvn eclipse:eclipse -DdownloadSources=true
```

8.7. Building And Running JTrac

You can explore all the Ant targets that are available. You can also try the "jetty-start" Ant target straight away that will compile, create and deploy the exploded war as well as boot a Jetty server. For NetBeans users, the "jetty-start" target is conveniently mapped to the "Run Main Project (F6)" shortcut, once you make JTrac the "main project". Note that this Ant target is smart enough to detect if Jetty is already running and will perform a shutdown if required before re-starting.

When starting with freshly checked-out source code, the Ant script may prompt you for a JTrac Version and then create a "version.properties" file. This process is just to make the JTrac release process easier so it does not really matter what you type. You would typically need to do this only once in development mode.

Maven has been configured to download the Jetty web server which drastically reduces the amount of setup that you need to do in order to set up your development environment. Since JTrac creates the HSQLDB database if required on startup, you do not need to worry about installing, configuring and creating any database either.

If you want to use Apache Tomcat instead of Jetty, just have "build.properties" point to a valid Tomcat instance. By now, you may have noticed that because of the Ant script, you don't even need a Tomcat or Jetty plugin in either NetBeans or Eclipse.

8.8. Adding a language translation for JTrac

JTrac has full internationalization (i18n) support and you can easily add a new language without re-compiling JTrac. All you need to do is introduce a translated version of the existing "messages.properties" file into the "/WEB-INF/classes" folder. Note that you will need to unzip the WAR file in order to add more files to it.

For example, if you need to add a translation into French, a file called "messages_fr.properties" should exist. You can refer the list of Java locale codes here. Note that country specific locales are also supported, for example "fr_CA" means "French, Canada".

You can look at the existing translations e.g. "messages_de.properties" as examples. Note that JTrac needs to be restarted for changes to the properties files to take effect. However, switching the language for any user can

easily be done anytime from the "Edit Your Profile" link on the options screen and this takes effect the instant the user saves his profile.

You can also get the latest versions of the "messages.properties" as well as the available translations directly from version control at this url: http://j-trac.svn.sourceforge.net/viewvc/j-trac/trunk/jtrac/src/main/resources/ Since these may change a lot, it is recommended that you check here as well before submitting a translation. We recommend that you submit translations as attachments on the JTrac patches tracker where we keep track of code contributions as well.

You can use this nice open-source tool called PRBEditor that really helps by showing the English and translated versions side by side on a single screen. It also takes care of handling special characters.