**背景与愿景：**开发环境下，tomcat对热布署的支持还不够全面，致使开发人员浪费大量时间在重起服务上。为了提高开发效率，决定引入Jrebel，它对热布署的支持相对比较全面。虽然Jrebel官方号称使用它不存在内存泄漏问题，但是占用一定的资源是肯定的，因此不考虑在正式环境下使用热布署。Jrebel实际上支持非常多中间件，除了Tomcat还包括Jetty、Resin、Weblogic等等，从理论上来讲，他跟中间件也没什么关系，但实际配置的时候还是会根据中间件有所不同，具体可以上官网查看，本文要讲的是tomcat+ eclipse+ spring+ struts2+ maven的环境。在使用Jrebel后，我们期望看到开发人员早上开机启动一次tomcat后就够了。

**使用场景：**Tomcat对热布署的使用场景是Servlet+JSP+JaveBean。如果项目含有其他框架时，其热布署效果就会大大降低，在与同事一同测试观察后发现：tomcat6在spring+struts框架下的项目，对java文件修改后的成功热布署概率偏低。由于概率太低，而且有无热布署成功不能确定，大部分开发人员修改类后不管什么情况直接选择重起，长此以往，浪费的时间积累起来不在少数。下面把tomcat和jrebel对热布署测试结果对比一下：

|  |  |  |
| --- | --- | --- |
| 对比项 | Jrebel | Tomcat |
| Class文件 | 绝大部分能热布署 | 小部分能热布署 |
| Spring支持 | 改成用注释的方式后，可支持 | 不支持 |
| Struts配置文件 | 支持 | 支持 |
| 页面相关文件 | 支持 | 支持 |

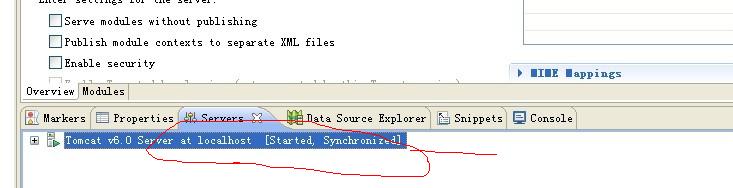
从对比可以看到，Jrebel最大的提升是对java类修改时，热布署大大提高；而对spring的支持实际上还是有限的，需要把IoC的实现改成使用注释的方式，而不能是配置的方式。如果你的工程的Spring已经是注释的方式，那就比较顺利，装好插件后，绝大部分情况下都能使用热布署了。如果你不是使用注释方式，那就麻烦了，要么全都改成注释方式，要么Jrebel对spring作用有限，看你自己的选择了。下面把已知Jrebel不能成功的热布署的情况作一列举：

1. 替换了父类。
2. 增加或删除了继承的接口。
3. Spring布署文件修改(如果改成注释方式，实际上spring只剩个别固定的第三方包的beans描述，比如数据库链接等)
4. web.xml，虽然jrebel和tomcat都支持web.xml修改的热布署，但是如果项目比较复杂，初始化工作较多的话，还是直接重起吧，直接热布署意义不大，而且重复初始化对于某些业务来说是会报错，所以建议有较复杂的初始化项目来说，还是直接重起算得了。

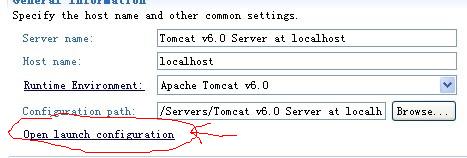
**Jrebel安装和使用**

1. jrebel是商用软件，而且价格不扉，最新的破解版是4.5。将jrebel文件夹放在任意一个盘下即可，（我是在D盘）。然后按照下面步骤配置。
2. 配置tomcat参数：

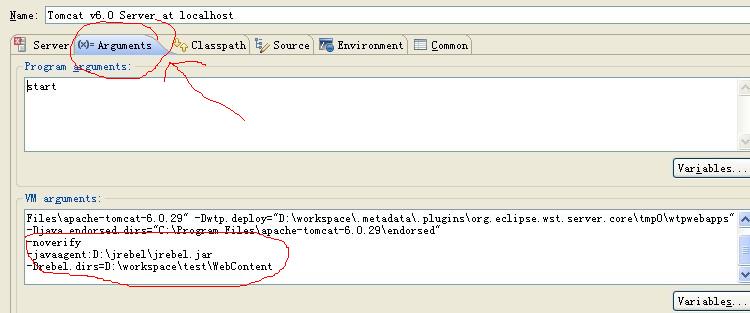
打开servers，双击tomcat，打开Overview界面，如下图红色标注



点击open launch configuration，如下图红色标注



在新窗口选择arguments选项，然后在VM arguments 最后加入参数，如下图：



需要添加的参数：

-noverify

-javaagent:D:\jrebel\jrebel.jar

-Drebel.dirs=D:\workspace\test\WebContent

-Drebel.spring\_plugin=true

-Drebel.hibernate\_plugin=true

-Drebel.struts2-plugin=true

参数说明：

-javaagent:D:\jrebel.jar 这里自行修改jrebel.jar正确的路径；

-Drebel.dirs 这里的路径是要监控的项目的路径，

-Drebel.spring\_plugin=true 支持spring框架

-Drebel.struts2-plugin=true 支持strut2（注意横线在中间，与其他不同）

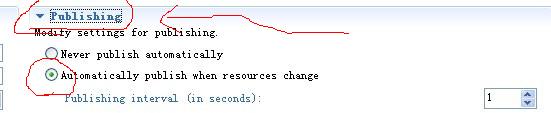
-Drebel.hibernate\_plugin=true 支持hibernate

如果你要支持更多的框架，可以参考官网http://www.zeroturnaround.com/jrebel/features/frameworks/

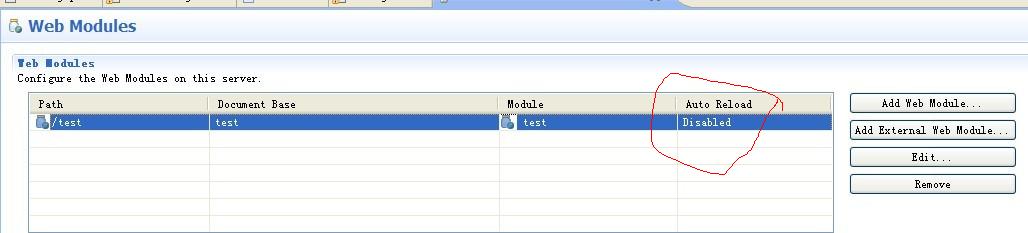
如果你要了解更多的参数配置，可以参考官网

<http://www.zeroturnaround.com/jrebel/configuration/>

1. 返回Overview界面，在右侧展开publishing选项，选择Automatically publish when resource changes ，如下图：



1. 切换到modules页，设置项目auto reload为disabled，如下图：



配置到这里，启动tomcat在控制台应该能看到如下内容：

#############################################################

JRebel 4.5 (201109201828)

(c) Copyright ZeroTurnaround OU, Estonia, Tartu.

Over the last 10 days JRebel prevented

at least 9 redeploys/restarts saving you about 0.4 hours.

This product is licensed to zhangthe9

The following plugins are disabled at the moment:

\* Click plugin (set -Drebel.click\_plugin=true to enable)

Reloads menu.xml menu definitions of Apache Click projects.

\* Jersey plugin (set -Drebel.jersey\_plugin=true to enable)

Reloads Jersey configuration from Java annotations.

\* Lift plugin (set -Drebel.lift\_plugin=true to enable)

Supports reloading singleton objects that extend RestHelper, MVCHelper,

LiftScreen or Wizard.

\* Oracle ADF Faces plugin (set -Drebel.adf\_faces\_plugin=true to enable)

JRebel ADF Faces Plugin

\* Seam-Wicket plugin (set -Drebel.seam\_wicket\_plugin=true to enable)

Integration with load time weaving seam annotations to wicket classes

(-javaagent:<path-to-jboss-seam-wicket-jar>)

\* WebObjects plugin (set -Drebel.webobjects\_plugin=true to enable)

WebObjects JRebel Plugin

#############################################################

JRebel: Directory 'D:\workspace\test\WebContent' will be monitored for class changes.

到这里就配置完了。后面的内容是对jrebel做更多了解。

jrebel支持监控多个目录下的classes、配置文件、jar包是否被修改,因此建议新建并配置rebel.xml文件，如果Eclipse安装了官网的jrebel plugin，那么可以从eclipse菜单里产生rebel.xml文件。以下是rebel.xml的简单手动配置：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<application

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://www.zeroturnaround.com"*

xsi:schemaLocation=*"http://www.zeroturnaround.com/alderaan/rebel-2\_0.xsd"*>

<classpath>

<dir name=*"E:\projects\cmac\target\classes"*/>

<dir name=*"E:\projects\cmac\target\test-classes"*/>

</classpath>

<web>

<link target=*"/"*>

<dir name=*"E:\projects\cmac\src\main\webapp"*/>

</link>

</web>

</application>

rebel.xml更详细配置说明参考官网(<http://www.zeroturnaround.com/jrebel/configuration/>)

3、此时启动tomcat，会发现如下错误信息

严重: Exception starting filter Struts2

java.lang.NoClassDefFoundError: Lorg/apache/velocity/app/VelocityEngine;

at java.lang.Class.getDeclaredFields0(Native Method)

at java.lang.Class.privateGetDeclaredFields(Class.java:2291)

at java.lang.Class.getDeclaredFields(Class.java:1743)

at com.opensymphony.xwork2.inject.ContainerImpl.addInjectors(ContainerImpl.java:102)

at com.opensymphony.xwork2.inject.ContainerImpl$1.create(ContainerImpl.java:84)

at com.opensymphony.xwork2.inject.ContainerImpl$1.create(ContainerImpl.java:82)

at com.opensymphony.xwork2.inject.util.ReferenceCache$CallableCreate.call(ReferenceCache.java:155)

at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:303)

at java.util.concurrent.FutureTask.run(FutureTask.java:138)

at com.opensymphony.xwork2.inject.util.ReferenceCache.internalCreate(ReferenceCache.java:81)

at com.opensymphony.xwork2.inject.util.ReferenceCache.get(ReferenceCache.java:121)

at com.opensymphony.xwork2.inject.ContainerImpl$ConstructorInjector.<init>(ContainerImpl.java:333)

at com.opensymphony.xwork2.inject.ContainerImpl$5.create(ContainerImpl.java:299)

at com.opensymphony.xwork2.inject.ContainerImpl$5.create(ContainerImpl.java:298)

at com.opensymphony.xwork2.inject.util.ReferenceCache$CallableCreate.call(ReferenceCache.java:155)

at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:303)

at java.util.concurrent.FutureTask.run(FutureTask.java:138)

at com.opensymphony.xwork2.inject.util.ReferenceCache.internalCreate(ReferenceCache.java:81)

at com.opensymphony.xwork2.inject.util.ReferenceCache.get(ReferenceCache.java:121)

at com.opensymphony.xwork2.inject.ContainerImpl.getConstructor(ContainerImpl.java:578)

at com.opensymphony.xwork2.inject.ContainerImpl.inject(ContainerImpl.java:476)

at com.opensymphony.xwork2.inject.ContainerImpl$7.call(ContainerImpl.java:517)

at com.opensymphony.xwork2.inject.ContainerImpl.callInContext(ContainerImpl.java:565)

at com.opensymphony.xwork2.inject.ContainerImpl.inject(ContainerImpl.java:515)

at com.opensymphony.xwork2.config.impl.LocatableFactory.create(LocatableFactory.java:32)

at com.opensymphony.xwork2.inject.ContainerBuilder$4.create(ContainerBuilder.java:135)

at com.opensymphony.xwork2.inject.Scope$2$1.create(Scope.java:49)

at com.opensymphony.xwork2.inject.ContainerImpl$ParameterInjector.inject(ContainerImpl.java:447)

at com.opensymphony.xwork2.inject.ContainerImpl.getParameters(ContainerImpl.java:462)

at com.opensymphony.xwork2.inject.ContainerImpl.access$000(ContainerImpl.java:48)

at com.opensymphony.xwork2.inject.ContainerImpl$MethodInjector.inject(ContainerImpl.java:288)

at com.opensymphony.xwork2.inject.ContainerImpl$2.call(ContainerImpl.java:117)

at com.opensymphony.xwork2.inject.ContainerImpl$2.call(ContainerImpl.java:115)

at com.opensymphony.xwork2.inject.ContainerImpl.callInContext(ContainerImpl.java:558)

at com.opensymphony.xwork2.inject.ContainerImpl.injectStatics(ContainerImpl.java:114)

at com.opensymphony.xwork2.inject.ContainerBuilder.create(ContainerBuilder.java:495)

at com.opensymphony.xwork2.config.impl.DefaultConfiguration.reloadContainer(DefaultConfiguration.java:170)

at com.opensymphony.xwork2.config.ConfigurationManager.getConfiguration(ConfigurationManager.java:55)

at org.apache.struts2.dispatcher.Dispatcher.init\_PreloadConfiguration(Dispatcher.java:371)

at org.apache.struts2.dispatcher.Dispatcher.init(Dispatcher.java:424)

at org.apache.struts2.dispatcher.FilterDispatcher.init(FilterDispatcher.java:213)

at org.apache.catalina.core.ApplicationFilterConfig.initFilter(ApplicationFilterConfig.java:273)

at org.apache.catalina.core.ApplicationFilterConfig.getFilter(ApplicationFilterConfig.java:254)

at org.apache.catalina.core.ApplicationFilterConfig.setFilterDef(ApplicationFilterConfig.java:372)

at org.apache.catalina.core.ApplicationFilterConfig.<init>(ApplicationFilterConfig.java:98)

at org.apache.catalina.core.StandardContext.filterStart(StandardContext.java:4562)

at org.apache.catalina.core.StandardContext$2.call(StandardContext.java:5240)

at org.apache.catalina.core.StandardContext$2.call(StandardContext.java:5235)

at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:303)

at java.util.concurrent.FutureTask.run(FutureTask.java:138)

at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908)

at java.lang.Thread.run(Thread.java:619)

Caused by: java.lang.ClassNotFoundException: org.apache.velocity.app.VelocityEngine

at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1676)

at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1521)

at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:316)

... 53 more

竟然报出需要velocity相关包，那好吧，我的项目是用maven来做管理的，在pom.xml里加上相关依赖如下：

<dependency>

<groupId>org.apache.velocity</groupId>

<artifactId>velocity</artifactId>

<version>1.7</version>

</dependency>

<dependency>

<groupId>org.apache.velocity</groupId>

<artifactId>velocity-tools</artifactId>

<version>2.0</version>

</dependency>

再次重起tomcat后，一切正常，可以看到jrebel关键信息在console里输出：

JRebel: Directory 'E:\projects\cmac\target\classes' will be monitored for changes.

JRebel: Directory 'E:\projects\cmac\target\test-classes' will be monitored for changes.

JRebel: Directory 'E:\projects\cmac\src\main\webapp' will be monitored for changes.

4、如果你使用maven发布并启动tomcat，那么需要安装jrebel-maven-plugin。本文前面提到只是满足开发阶段而且启动tomcat方式不是使用mvn命令方式，因此不需要安装jrebel-maven-plugin。

**Spring利用注释方式实现IoC**

现在大部分java项目有使用Spring框架，为了能使Jrebel更好的对Spring相关资源发生热布署作用，就得充分使用注释的方式实现依赖注入。这里对Spring实现注释方式作一下最简单的介绍，首先在applicationContext.xml里配置如下两行代码：

<context:annotation-config/>

<context:component-scan base-package=*"\*"*/>

简单的说，以上的配置让spring支持了我们将要实现的注释依赖注入。以下以登录为实例，按action层、业务层、数据库操作层、PO层分别新建四个类：

LoginAction.java//struts action

UserServiceImpl.java//business layer

UserDaoImpl.java//dao layer

User.java //pojo

那么怎样通过注释方式进行调用的呢，首先给要被调用的类加上@Component注释，Spring为了区分不同层次的类，分别定义了以下四种注释

@Reposity

@Service

@Controller

@Component

目前阶段这四个注释实际上效果是一样的，我们约定如下：PO类如有需要使用@Reposity注释；Dao和Service使用@Service注释；Action使用@Controller注释；剩余分不出层次的类使用@Component注释。

如本例，action、service、dao分别加上注释

@Scope("prototype")

@Controller("loginAction")

public class LoginAction extends BaseAction{

}

@Service("userService")

public class UserServiceImpl implements UserService{

}

@Service("userDao")

public class UserDaoImpl extends BaseDao implements UserDao{

}

Scope注释默认是singleton，可以缺省。使用这四个标签时，如果不使用参数值，那么spring会按自己规范取名，比如LoginAction，使用@Controller()注释，默认取名为loginAction。取好了名，相当于在配置文件里配置了一组bean，接下来看怎么注入依赖，比如LoginAction要调用UserService，代码片段如下：

@Scope("prototype")

@Controller("loginAction")

public class LoginAction extends BaseAction{

…

@Autowired

Private UserService userService

…

}

就这么简单,添加xwork.xml配置，新加跳转页面，这些操作统统不用重起服务。

**弹出Continue or Terminate疑问**

装上jrebel后，可以进入你的tomcat/conf/context.xml或server.xml，其中有一个参数reload=true，把它改成false。表示关闭tomcat自身的热布署，在eclipse里启动tomcat，修改了类，有时还是会弹出Continue or Terminate框，难道是个Bug？不得而知。不过有jrebel在不用担心，继续continue，会发现你的修改是有效的。只有碰到前面提到的不适合jrebel热布署的场景时，即使没弹出Continue or Terminate提示框，你也要自己重起服务。

**Jrebel官方对热布署支持的场景列表**（查看官网说明http://www.zeroturnaround.com/jrebel/features/）

| **Java EE Support** | **Jrebel** | **JVM Hot Swap** |
| --- | --- | --- |
| **Time to reload** | **< 1s** | **< 1s** |
| **No memory leak** | YES | YES |
|  | | |
| **Changes to method bodies** | YES | YES |
| **Adding/removing Methods** | YES | NO |
| **Adding/removing constructors** | YES | NO |
| **Adding/removing fields** | YES | NO |
| **Adding/removing classes** | YES | NO |
| **Adding/removing annotations** | YES | NO |
| **Changing static field value** | YES JRebel 3.0+ | NO |
| **Adding/removing enum values** | YES JRebel 3.0+ | NO |
| **Changing interfaces** | YES | NO |
| **Replacing superclass** | NO | NO |
| **Adding/removing implemented interfaces** | NO | NO |
|  | | |
| **Skip builds for WAR directories** | YES | YES |
| **Skip builds for .WAR/.EAR class updates** | YES | YES |
| **Skip builds for .WAR/.EAR resource updates** | YES | NO |
| **Map multiple source dirs to one .WAR/.EAR target dir** | YES | NO |
| **Map classes and resources with include/exclude patterns** | YES | NO |
| **Map multiple source dirs with Ant-style patterns** | YES | NO |
| **Use system properties to make mapping machine-independent** | YES | NO |
| **Maven plugin** | YES | NO |
| **JSP EL changes** | YES | NO |
| **JSP Scriptlet changes** | YES Enterprise Add-on | NO |
| **EJB 1.x session bean interface changes** | YES Enterprise Add-on | NO |
| **EJB 2.x session bean interface changes** | YES Enterprise Add-on | NO |
| **EJB 3.x session bean interface changes** | YES JRebel 3.0+ | NO |
| **JSF changes (Mojarra)** | YES JRebel 3.0+ | NO |
| **JPA changes (Hibernate, EclipseLink, TopLink, OpenJPA)** | YES JRebel 3.0+ | NO |
| **CDI changes (Weld)** | YES JRebel 3.0+ | NO |
| **ResourceBundle** | YES | NO |
| **Spring Framework 2.x or later** | YES | NO |
| **Hibernate** | YES JRebel 3.0+ | NO |
| **JBoss Seam 2.x or later** | YES JRebel 3.0+ | NO |
| **Google Guice** | YES | NO |
| **Stripes 1.x or later** | YES | NO |
| **Apache log4j 1.2.x or later** | YES | NO |
| **Apache Struts 1.x** | YES | NO |
| **Apache Struts 2.x or later** | YES | NO |
| **Apache Tapestry4** | YES | NO |
| **Apache Velocity** | YES | NO |
| **Apache Wicket** | YES | NO |
| **CgLib** | YES JRebel 3.0+ | NO |
| **Javassist** | YES JRebel 3.0+ | NO |
| **Atlassian Confluence plugins** | YES | NO |
| **ClassWorlds** | YES Beta | NO |
| **Apache Felix** | YES Beta | NO |
| **Eclipse Equinox** | YES Beta | NO |
| **IntelliJ IDEA 7.x, 8.x plugins** | YES Beta | NO |
| **NetBeans plugins** | YES Beta | NO |

**Jrebel对第三方框架支持地应表** (查看官网说明http://www.zeroturnaround.com/jrebel/features/frameworks/)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AspectJ | aspectj\_plugin | * Enables the load-time-weaving of AspectJ aspects. Can alter class loading/initialization order during startup. | 1.5.3 1.6.8 | 1.5+ |
| Click | click\_plugin | * Reloads the menu definition from menu.xml. | 2.2.0 | 2.2+ |
| EclipseLink | eclipselink\_plugin | * Reloads an EntityManagerFactory when any of the mapping files or annotated entity classes changes. When an entity class is reloaded the corresponding cache entries are flushed. | 1.0.1 2.0.2 | 1.x 2.x |
| Facelets | facelets\_plugin | * Turns on facelet development mode for automatic reloading of facelets. * Also provides reloading of facelet tag libraries. | 1.1.15.B1 | 1.x |
| FreeMarker | *(integrated)* | * Reloads BeanWrapper. Remedies the flaws of FreeMarker’s own JRebel integration. | 2.3.16 | 2.3.7+ |
| Groovy | groovy\_plugin | * Refreshes Groovy class metadata and call site cache on class change. After reload, call site cache may be misaligned for obsolete methods on stack. |  | 1.5+ |
| Guice | guice\_plugin | * Wires changed dependencies in Guice singletons. | 1.0 2.0 | 1.0+ |
| GWT | gwt\_plugin | * Enables to reload client-side classes in hosted (development) mode. Server-side classes should be reloaded without the plugin. * *Limitations:* Native Javascript methods may not be supported. | 2.0.3 | 2.0.x |
| Hibernate | hibernate\_plugin | * Enables configuration reloading (rebuilding of Hibernate’s SessionFactory) when updates are detected to underlying configuration resources (either XML mapping files or model classes with JPA annotations). * Enables automatic schema updating if hibernate.hbm2ddl.auto=update or schemaUpdate=true on org.springframework.orm.hibernate3.LocalSessionFactoryBean. * *Limitations:* The SessionFactory rebuilding works when it is being built through classes HibernatePersistence, HibernateService, org.jboss.hibernate.jmx.Hibernate or org.springframework.orm.hibernate3.LocalSessionFactoryBean. All classes that create SessionFactory instances have to be handled separately. If user creates SessionFactory in his own code, the integration won’t capture that. | 3.3.2.GA | 3.x. |
| Hibernate Validator | hibernate\_validator\_plugin | * Enables adding/editing/removing Bean Validation constraint annotations on bean classes and parent constraint annotation types (i.e., supports composite constraints) * Enables creating/editing custom Bean Validation constraint annotation types * *Limitations:* Changes to META-INF/validation.xml are not propagated. * *Limitations:* Changes to XML constraint mappings are not propagated (initial mappings are retained) | 4.0.2.GA 4.1.0.Final 4.2.0.Beta2 | 4.x.x |
| iBatis | ibatis\_plugin | * Refreshes changes in iBATIS SQL maps. * *Limitations:* No support for MyBatis. * *Limitations:* Only supports refreshing sql maps and not anything else. * *Limitations:* Does not suppor mappingLocations in spring SqlMapClientFactoryBean. | 2.1.6 2.2.0 2.3.4 | 2.1.x 2.2.x 2.3.x |
| Jackson | jackson\_plugin | * Makes Jackson pick up changes (including changed annotations) to classes that it serializes to JSON. Does this by clearing JSON serializer caches for changed classes. * *Limitations:* Only serializer caches are updated. Doesn’t update de-serializer caches. | 1.0.0 1.1.0 1.5.0 1.6.1 | 1.0.0+ |
| JBoss AOP | jbossaop\_plugin | * Allows for load-time weaving of JBoss AOP aspects. (JRebel agent must be placed before JBoss AOP agent in JVM arguments.) |  | 2.0 |
| Lift | lift\_plugin | * Makes Lift pick up more changes in LiftScreens, Wizards, RestHelpers and MVCHelpers. Does this by re-instantiating any singletons that extend the above-mentioned classes. * Keep in mind that the singletons you write that extend the mentioned Lift classes may be initialized multiple times when using this plug-in. * *Limitations:* Some changes in closures/anonymous functions will still cause problems that require restarts. * See [here](http://www.zeroturnaround.com/blog/lift-support-in-jrebel/) for additional information. | Lift 2.1, 2.2 / Scala 2.8.0 | Lift 2.1+ / Scala 2.8+ |
| Log4J | log4j-plugin | * Remembers the configuration file used for Log4j initialization, monitors it for updates, makes Log4j reinitialize itself when this file is changed. * Includes a 3rd party contribution from Julien Richard. | 1.2.12 1.2.15 | at least all 1.2.x versions |
| LOGBack | logback\_plugin | * Remembers configuration files that LOGBack loaded it’s configuration from (e.g. logback.xml). If updates to those files are detected, framework configuration is rebuilt. | 0.9.12 0.9.13 0.9.26 | 0.9.12 – .. |
| Mojarra | mojarra\_plugin | * Refreshes Mojarra annotation and XML-based configuration. Turns on mojarra development mode and facelet refreshing. * Turns on Mojarra development mode and Facelet refreshing. | 1.2\_13-b01-FCS 2.0.2-FCS | 1.2+ |
| OpenJPA | openjpa\_plugin | * Reloads an EntityManagerFactory when any of the mapping files or annotated entity classes changes. When an entity class is reloaded, the corresponding cache entries are flushed. | 1.1 | 1.x |
| RestEasy | resteasy\_plugin | * Configuring through context-param javax.ws.rs.Application. * Configuring through listener (org.jboss.resteasy.plugins.server.servlet.ResteasyBootstrap). * Configuring through filter (org.jboss.resteasy.plugins.server.servlet.FilterDispatcher). * Configuring through servlet (org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher). * Only configuring an Application class is supported (a class that extends javax.ws.rs.core.Application – it is a standard JAX-RS way). * Reloading of JAX-RS annotated methods (adding/removing/changing). * See [the screencast](http://www.zeroturnaround.com/screencast-resteasy-application-instant-updates-with-jrebel/). | 2.0.1.GA | 2.0.x |
| Seam | seam\_plugin | * Supports loading new Seam component definitions, either from annotated Java classes or from components.xml. Renaming or removing existing definitions is not supported. * Refreshes seam component metadata and interceptors on class change. Supports reloading Seam method metadata annotations like @Observer, @RaiseEvent, etc. * Supports reloading navigation files (pages.xml, \*.page.xml) outside Seam’s development mode (supported for Seam 2.1.x and 2.2.x). * *Notice:* It is recommended to turn off Seam’s development mode while using Seam together with JRebel. | 2.0.2.SP1 2.1.1.GA 2.2.0.GA 2.2.1.CR1 | 2.x |
| Seam-Wicket integration | seam\_wicket\_plugin | * Lets Seam Wicket integration load-time-weave Wicket components. |  |  |
| Spring (core) | spring\_plugin | * Monitors Spring bean definitions in xml-files, reloads them when changed. * Monitors packages that could contain Spring beans defined by annotations. If new beans (or changes to old ones) are seen, reloads them. * Makes Spring re-do the bean auto-wiring and singleton bean configuration. * Disables caches for properties that are replaced replaced for property placeholders in Spring’s XML configuration files. When underlying properties files have changed, new values will be inserted on next reloading of these Spring beans. * (The integration doesn’t of course support features that Spring 2.0 itself didn’t yet support). | 2.0 2.5.6 3.0.2 3.1.0.M1 | 2.0 2.5.x 3.0.x 3.1.0.M1 |
| Spring MVC | spring\_plugin | * Reload URL mappings (defined by @RequestPath annotations) when Spring beans have changed. (Before processing a request, Spring MVC plugin tells Spring core plugin to check/reload the beans. If any bean was reloaded, Spring core plugin sends notification to Spring MVC plugin, which then triggers URL mappings rebuilding. * Also supports reloading other annotations: @ExceptionHandler, @ModelAttribute, @RequestParam. * The plugin ID intentionally collides with the one of Spring core plugin. Spring MVC integration depends on Spring core integration and cannot be enabled independently. | 2.0 2.5.6 3.0.2 3.1.0.M1 | 2.0 2.5.x 3.0.x 3.1.x |
| Spring Web Flow | spring\_webflow\_plugin | * Just turns on the development mode by default. No other customizations at all. | 2.1.1 | 2.x |
| Stripes | stripes\_plugin | * Fixes Stripes’s annotations metadata caching to work with reloading. Classes that have fields with @Validate annotations will be removed from caches and re-processed by Stripes after being reloaded by JRebel. * Fixes UrlBinding caching for ActionBeans. UrlBindings configured with the @UrlBinding annotations are reloaded by overriding the behaviour of the NameBasedActionResolver. * Supports stripes-guice and stripes-guicer plug-ins for instantiating Guice ActionBeans. *(available since JRebel 4.0 M2)* * Includes 3rd party contributions from Andreas Knifh. | 1.5.3 1.5.5 | 1.5.x |
| Struts1 | struts1-plugin | * Reloads full Struts configuration (by forcing Struts to re-create it’s ActionServlet when updates to underlying configuration XML-files are detected. (XML configuration files are remembered when Struts is initializing itself.) * “Reloads” Struts’s message resources by emptying the message caches when there are any updates to message resources. (Message resource files are remembered when Struts is initializing itself.) | 1.0.2 1.1 1.2.7 1.3.10. | 1.x.x (1.0.x  1.1.x  1.2.x  1.3.x) |
| Struts2 | struts2-plugin | * Keeps track of XML configuration file updates and triggers the Struts2 configuration reloading when there are any (duplicates Struts2’s own buggy file monitoring mechanism — see other comments). * Keeps track of annotated action-classes. If the set of action-classes has changed or there are changes to any of the action-classes, trigger configuration reloading. | 2.0.14 2.1.8.1 | 2.0.x 2.1.x 2.2.x |
| Tapestry4 | tapestry4\_plugin | * Enables adding new listener methods to Tapestry’s Page classes. * 3rd party contribution. | 4.1.6 | 4.x |
| Tiles 1 (embedded in Struts1) | tiles1\_struts\_plugin | * Enabled reloading of Tiles definitions from XML configuration files. Implements it by recreating DefinitionsFactory instance when the underlying XML configuration resources have changed. * See [the JRebel+Tiles tutorial](http://www.zeroturnaround.com/jrebel/how-to-use-jrebel-with-tiles/). | Struts 1.1 1.2.4 1.2.9 1.3.10. | Struts 1.1 1.2.x 1.3.x |
| Tiles2 | tiles2\_plugin | * In Tiles 2.0.x and 2.1.x, monitors XML-files containing Tiles definitions and makes Tiles reload them when there are any changes to those files. * For Tiles 2.2.x, just turns on Tiles’s own definitions monitoring and lets it do the job itself. * Tested and working also when set up through Spring. * See [the JRebel+Tiles tutorial](http://www.zeroturnaround.com/jrebel/how-to-use-jrebel-with-tiles/). | 2.0.7 2.1.4 2.2.2. | 2.0.x 2.1.x 2.2.x. |
| TopLink | toplink\_plugin | * Reloads an EntityManagerFactory (SessionFactory) when any of the mapping files or annotated entity classes change. * Works with JPA configuration and non-JPA configuration if using org.springframework.orm.toplink.LocalSessionFactoryBean (requires TopLinkSpringPlugin). * *Limitations:* Custom non-JPA configuration cannot be supported as we do not know how the configuration was loaded. | 10.1.3 | 10.x |
| TopLink-Spring integration | toplink\_spring\_plugin | * Reloads a SessionFactory when any of the mapping files or annotated entity classes changes. (Works with non-JPA configuration if using org.springframework.orm.toplink.LocalSessionFactoryBean). * Requires TopLinkPlugin. * See [here](http://static.springsource.org/spring/docs/2.5.x/api/org/springframework/orm/toplink/LocalSessionFactoryBean.html) for additional details. | TopLink 10.1.3 / Spring 2.5.6 | TopLink 10.x / Spring 1.2.x, 2.x, 3.x |
| Velocity | velocity\_plugin | * Velocity has a method cache for its known beans that prevents changes to those beans (new getter methods) to become accessible from associated Velocity templates. The plugin remedies this by triggering a cache rebuild each time a bean accessed by velocity has been reloaded by JRebel. The implementation is fine-grained: cache will be emptied only for the bean that was reloaded. | 1.6.3 | 1.6.x probably also others |
| Weld | weld\_plugin | * Refreshes Weld class metadata and wires changed dependencies on class change. | 1.0.1 | 1.x+ |
| Wicket | wicket\_plugin | * Automatically turns on Wicket’s HTML template reloading. * Tells Spring to try refreshing its context when Spring bean lookup operations fail. * Listens for class reload events. When received, cleans Wicket’s class metadata caches for the reloaded classes that cache the @SpringBean annotations, forcing Wicket to reload the metadata and enabling annotations to take effect. * See the [JRebel+Wicket tutorial](http://www.zeroturnaround.com/jrebel/how-to-use-jrebel-with-wicket/). | 1.2.7 1.3.7 1.4.12 | 1.2.x 1.3.x 1.4.x |
| JAX-WS Metro | metro\_plugin | * Keeps track of sun-jaxws.xml and class changes, invoking deploymentDescriptorParser to rebuild the mappings * Updates Metro’s dynamic servlets on servlet 3.0 environments | 2.1.7 2.2.3 | 2.1.x 2.2.x |
| Spring-WS | springws\_plugin | * Reloads endpoint mappings when classes have changed (annotation-driven conf) or new endpoint is added in via xml (xml conf) * Republishes wsdl after xsd is changed (when using dynamicWsdlDefinition) | 2.0.2 | 2.x |
| JAXB | jaxb\_plugin | * Enables support for reloadable classes in JAXBContext * Marshaller & Unmarshaller also pick up class changes immediately | 2.0.5 2.1.9 2.2.4 | 2.x |