

JRebel - Say Goodbye to Java Redeploys



JRebel - Say Goodbye to Java Redeploys

1. JRebel介绍
2. HotSwap的极限性与JRebel的扩展
3. JRebel的实现原理
4. Eclipse JRebel 插件
 - 4.1. 安装
 - 4.2. 破解
 - 4.3. 使用
5. IntelliJ IDEA JRebel 插件
 - 5.1. 安装
 - 5.1.1. Tomcat 启动参数上面加上 JRebel 的参数
 - 5.1.2. 使用 IntelliJ IDEA 插件的方式启动 JRebel
 - 5.2. 破解
 - 5.3. 使用
 - 5.3.1. 本地Local热部署
 - 5.3.2. 远程Remote热部署

1. JRebel介绍

JRebel是一款JAVA虚拟机插件，它使得JAVA程序员能在不进行重部署的情况下，即时看到代码的改变对一个应用程序带来的影响。JRebel使你能即时分别看到代码、类和资源的变化，你可以一个个地上传而不是一次性全部部署。当程序员在开发环境中对任何一个类或者资源作出修改的时候，这个变化会直接反应在部署好的应用程序上，从而跳过了构建和部署的过程，节省时间。

官网地址:<http://zeroturnaround.com>

2. HotSwap的极限性与JRebel的扩展

JRebel Class Reload Comparison Matrix

	JRebel 6	JRebel 5	JVM Hot Swap
Changes to method bodies	✓	✓	✓
Adding/removing Methods	✓	✓	✗
Adding/removing constructors	✓	✓	✗
Adding/removing fields	yes	✓	✗
Adding/removing classes	✓	✓	✗
Adding/removing annotations	✓	✓	✗
Changing static field value	✓	✓	✗
Adding/removing enum values	✓	✓	✗
Modifying interfaces	✓	✓	✗
Replacing superclass	✓	✗	✗
Adding/removing implemented interfaces	✓	✗	✗
Initializes new instance fields	✓	✗	✗

JRebel是通过在class loading的时候，通过字节码处理，生成一套自有的类结构来，和HotSwap不同，JRebel工作在ClassLoader这一层，而HotSwap是工作在JVM这一层的。JRebel采用了类似动态语言(jruby)的机制来实现class reloading（新增类，而不是修改类）。

JRebel使用类重写(ASM和Javassist)和JVM集成版本单独的类。加上它与应用程序集成服务器类/资源重定向和web服务器查找到工作区。它还集成了大多数应用服务器和框架监控更改配置(元数据或文件)。

动态语言一般的机制是将“类”实现为一个holder，只是一个入口，真正的里面的方法都是在关联的一些匿名的类里，所谓的动态，其实就是在新增 / 修改方法的时候，产生一个新的类，并且关联到那个holder上去，从调用的层面上来看，就实现了“动态”改变一个类的行为。JRebel将尽量多的方法调用都是直接的方法执行，而不是绕一圈再去，这样可以变得轻量一点，不管是内存消耗还是性能损耗上尽量避免去搞JDK里的类，除了个别必须的以外，例如ClassLoader和reflection api相关的类（例如，Class,Field,Method,Constructor等），处理好了反射相关的类，就可以兼容原有的反射功能的表现了，达到兼容的目的。

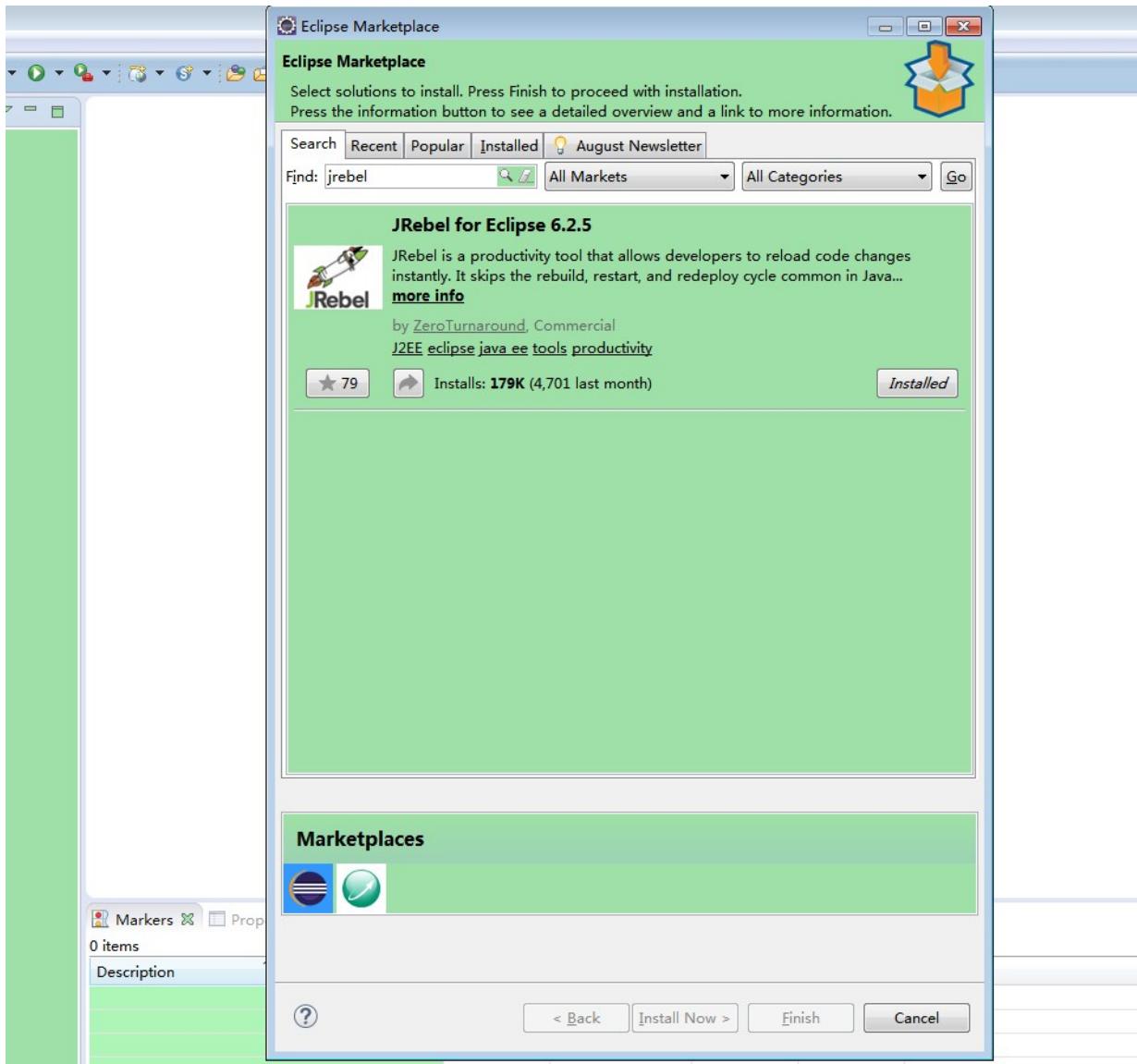
3. JRebel的实现原理

- 首先，需要通过instrument来修改ClassLoader和Reflection相关的类，加入自己的入口（比如监测类文件或者资源改变）
- 然后，在装载一个需要实现reload的类的时候，在修改过的ClassLoader里面，修改类的结构，加入特有的一些信息，比如一些call back类似的，因为无法修改已经装载的类，只能通过产生新的类来实现，因此在装载类A的时候，实际上是产生了一个A1给应用使用，通过一些字节码处理来达到和类A兼容，当类A的字节码修改的时候，再生成一个A2给应用，同时我们可以通过A的一个holder来保存一些运行状态，实现真正的“无缝”reload。
- 另外，修改的Reflection相关API，用来实现引用里使用reflection的时候的表现和原来一致，比如，应用得到和使用的是A1或者A2，如果在这上面做reflection，那岂不是得到一个A1/A2么，那肯定不行，因此要hack reflection API，告诉应用我们还是叫A，而不是A1/A2，对field和method的处理也一样。

4. Eclipse JRebel 插件

以Eclipse Mars Release (4.5.0)版本为例

4.1. 安装



安装后重启Eclipse，会在Eclipse安装目录的 `plugins` 下多类
似 `org.zeroturnaround.eclipse.embedder_6.2.1.SR1-201507221139` 目录

4.2. 破解

将破解的JRebel的lib文件夹下的jrebel.jar和jrebel.lic这2个文件覆盖拷贝
到 `\plugins\org.zeroturnaround.eclipse.embedder_6.2.1.SR1-
201507221139` 文件夹下的所有的有jrebel.jar的子文件夹里，然后重启eclipse即可破
解，破解效果图如下：



4.3. 使用

在Java的VM arguments 输入如下参数：

```
-noverify  
-agentpath:D:/Java/jrebel-6.1.1-agent-crack/libjrebel64.dll  
#Linux: -  
agentpath:/dev_env/jrebel/jrebel_running/lib/libjrebel64.so  
#Mac OS: -  
agentpath:/dev_env/jrebel/jrebel_running/lib/libjrebel64.dylib  
-Drebel.dirs=f:/myproject/test-jrebel/webapp/WEB-INF/classes  
-Drebel.disable_update=true  
-DJAVA_OPTS=-Xms256m -Xmx1024m -XX:MaxPermSize=512m
```

或者 Servers-tomcat --> Open launch configuration 配置如下

```
-Dfile.encoding=UTF-8 -Dspring.profiles.active=development -  
noverify -Drebel.disable_update=true -DJAVA_OPTS=-Xms512m -  
Xmx1256m -XX:MaxPermSize=512m
```

上述参数的相关说明：

- **-agentpath:** 这个是你使用的JRebel Agent版本的lib包的路径(路径后缀不要写成jrebel.jar)，注意其中的斜线方向。
- **-Drebel.dirs :**这个是你要监控的项目的 class 文件路径
- **-Drebel.disable_update:** 设为true,就不会联网检查更新
- **-DJAVA_OPTS:** 这个选项不是必须，当内存溢出的时候或其它特殊情况下才需要设置它的参数大小。

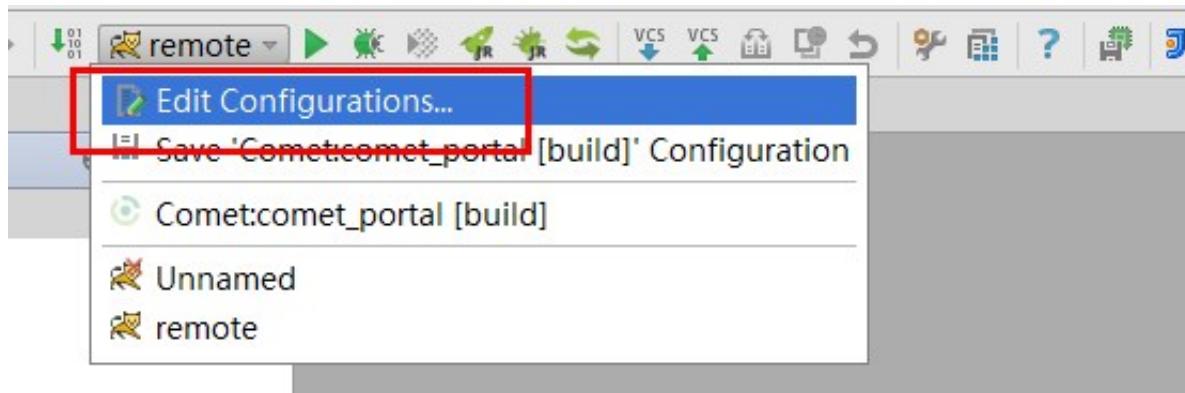
5. IntelliJ IDEA JRebel 插件

以IntelliJ IDEA 14.1.4版本为例，本地 Local 系统为 Windows 7 ,WEB服务器是 Tomcat 8.0.27 ；远程 Remote 服务器系统为 Centos7 Linux ，WEB服务器为 Tomcat 8.0.24 。

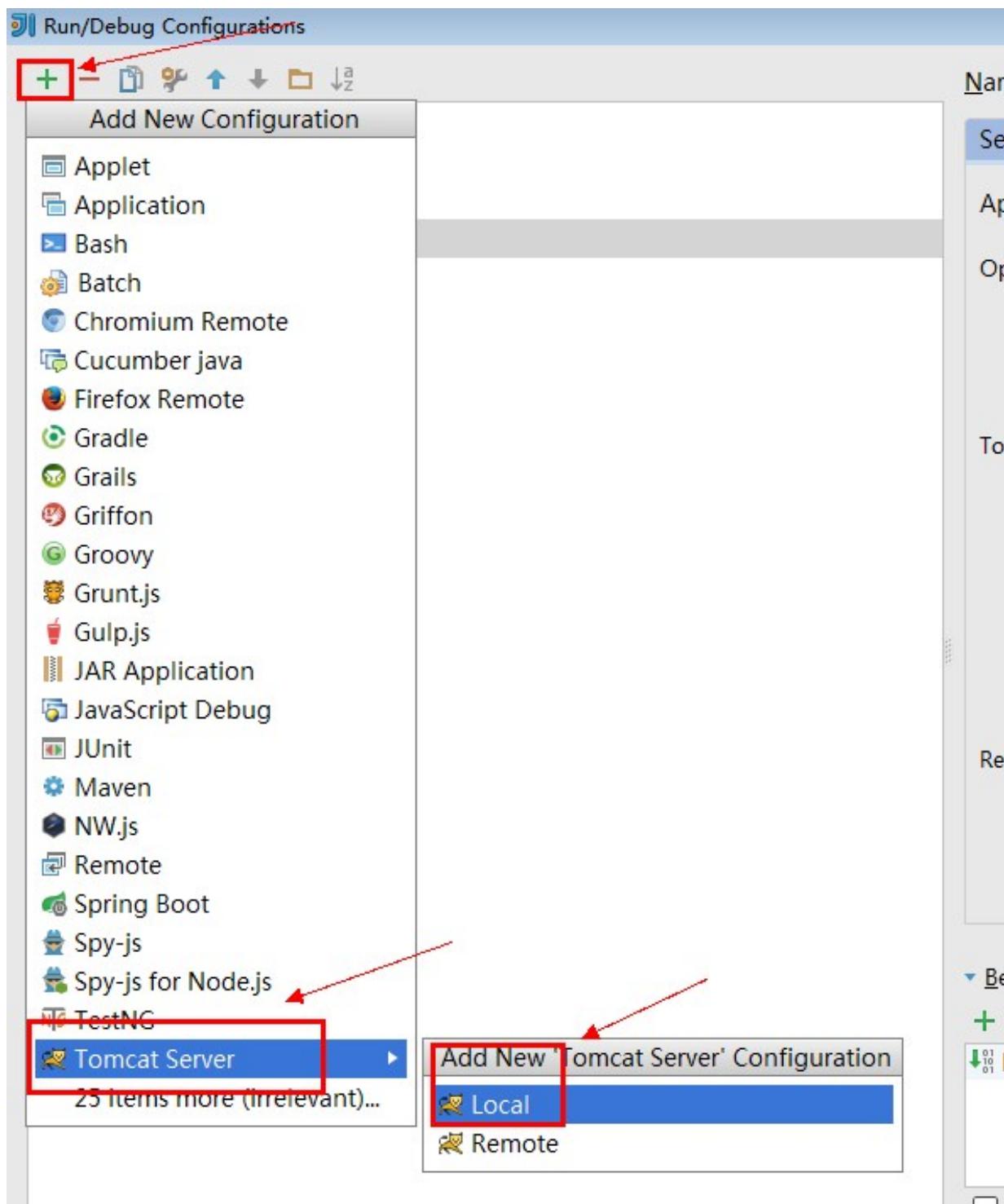
5.1. 安装

5.1.1. Tomcat 启动参数上面加上 JRebel 的参数

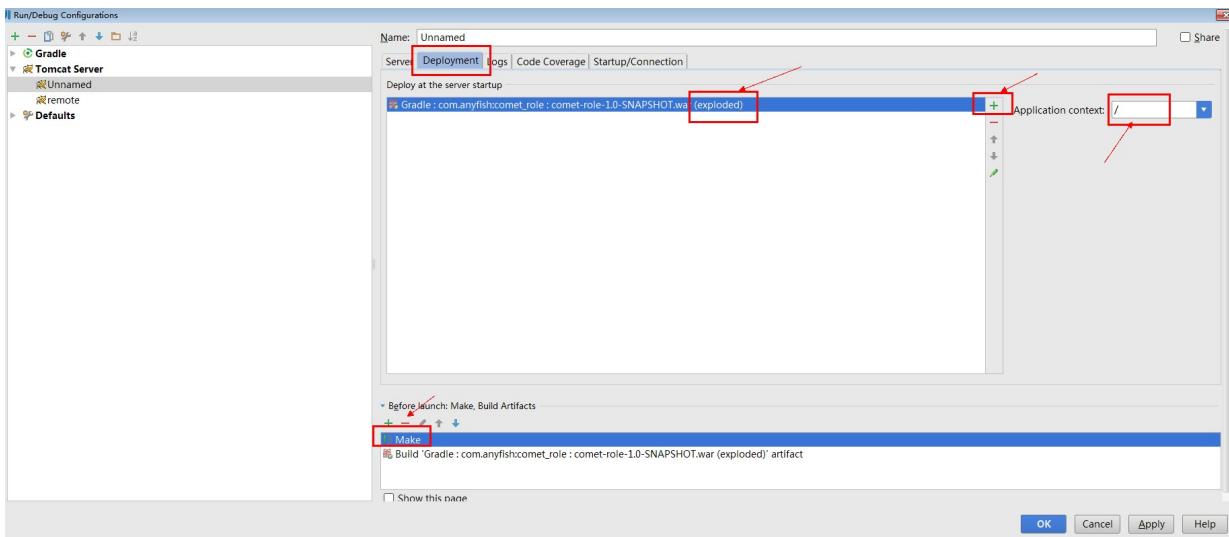
配置IntelliJ IDEA的 Tomcat



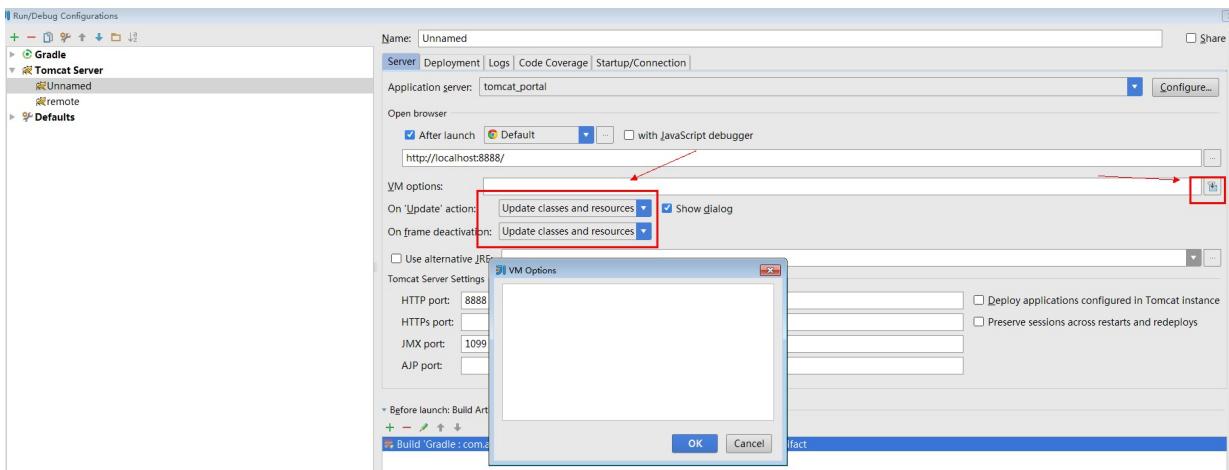
点 + 号选择 Tomcat Server -> Local



首先选择 Deployment 这个选项卡



根据实际项目，建议选择带 `exploded`，（这个相当于改 Tomcat 的 `CATALINA_HOME`，效率比较高），选择好后，删掉默认的 `Make`，提高效率，返回 `Server` 选项卡，会多了一项 `On frame deactivation`，若尚未配置 `Deployment` 选项卡则无此选项：



按如图所示的来配置，特别需要注意的是 `On 'Update' action` 和 `On frame deactivation` 这两项一定要选择 `Update classes and resources`，否则类修改热部署不生效，或者第三方模版框架例如 `Thymeleaf` 热部署不生效，接下来就是需要引入 `JRebel` 的地方了，在 `VM options` 的最右边有个箭头，弹出需要的 `VM options` 选项。

Windows

```
-noverify
-agentpath:D:/soft/jrebel/jrebel_running/lib/jrebel64.dll
```

Linux

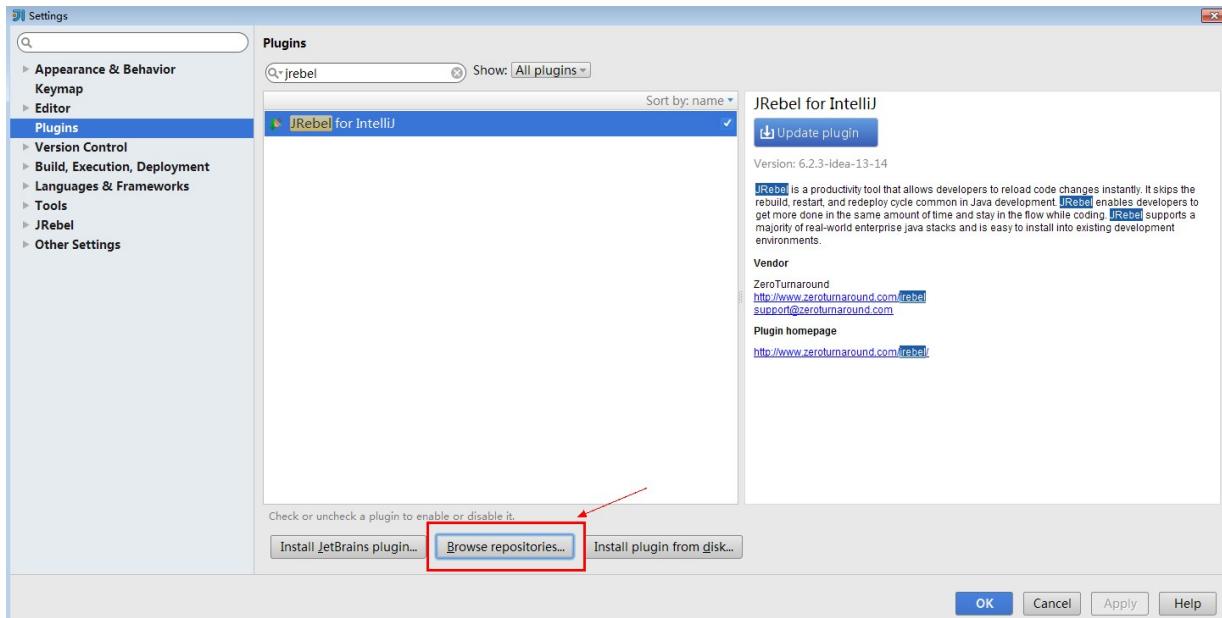
```
-agentpath:/soft/jrebel/lib/libjrebel64.so
```

Mac OS

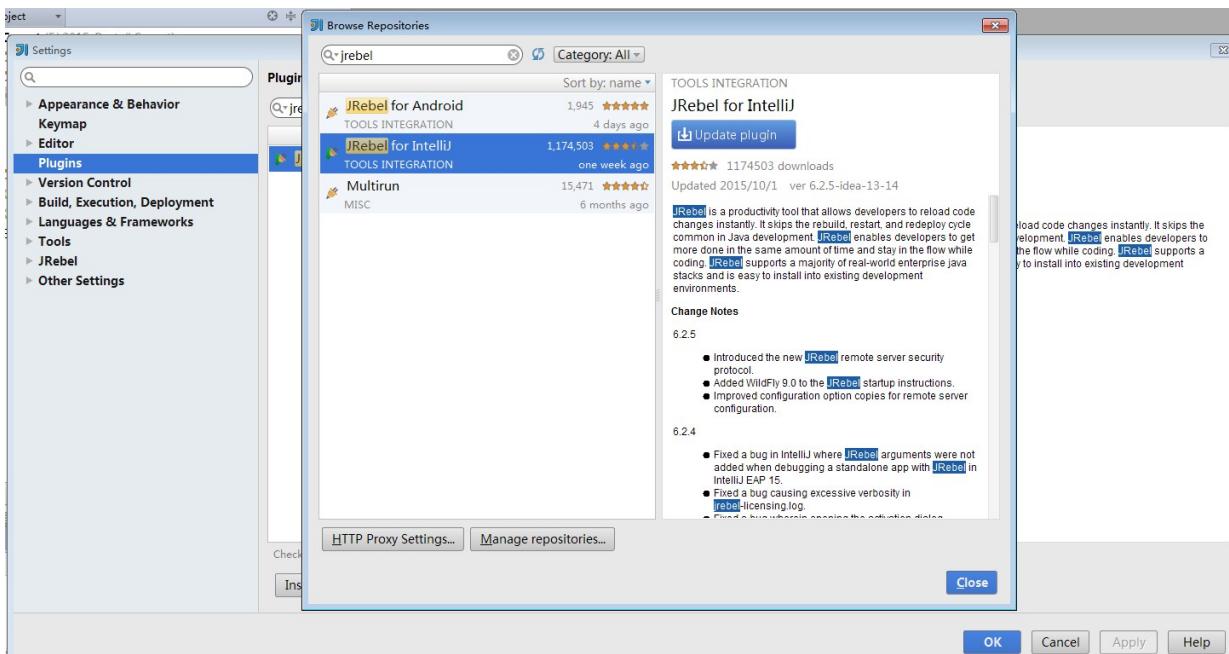
```
-agentpath:/soft/jrebel/lib/libjrebel64.dylib
```

5.1.2. 使用 IntelliJ IDEA 插件的方式启动 JRebel

采用在线安装插件方式



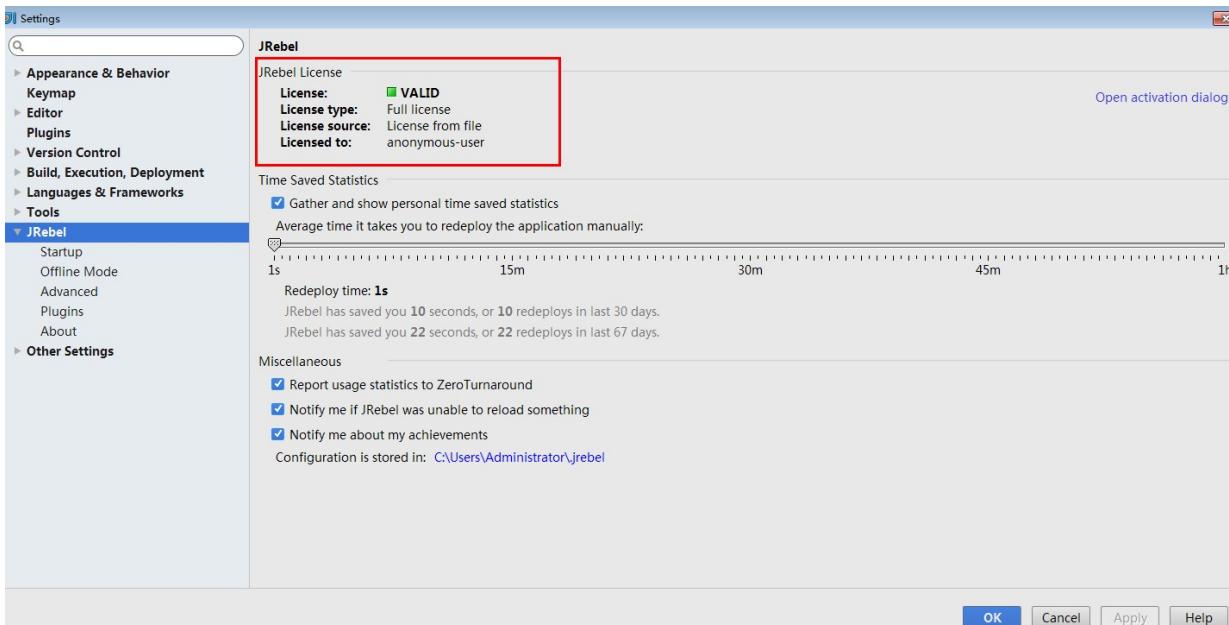
搜索 JRebel , 然后 Install Plugin



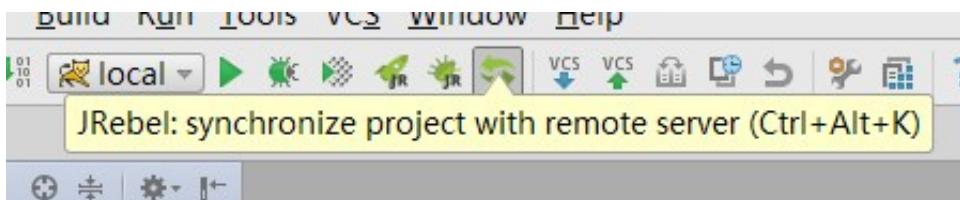
安装成功后，重启IntelliJ IDEA，在IntelliJ IDEA的配置目录（默认 \$USER_HOME/.IntelliJIdea14）中的 config/plugin 目录下多一个 jr-idea 插件目录，安装好后在IDEA的 settings 会多出一项JRebel的配置。

5.2. 破解

将破解的JRebel的lib文件夹下的jrebel.jar和jrebel.lic这两个文件覆盖拷贝到 \$USER_HOME/.IntelliJIdea14/config/plugin/jr-ide-idea 目录下所有的有 jrebel.jar 的子文件夹里，绿色的 VALID 表示破解成功

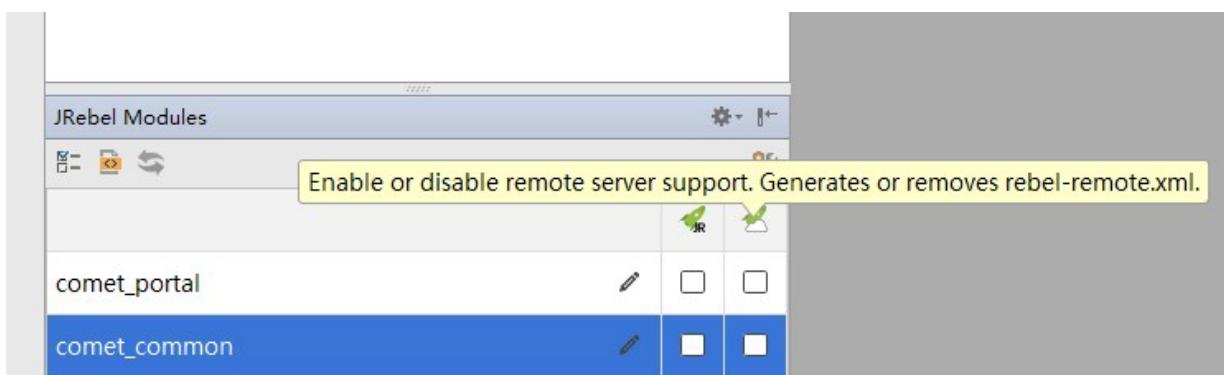
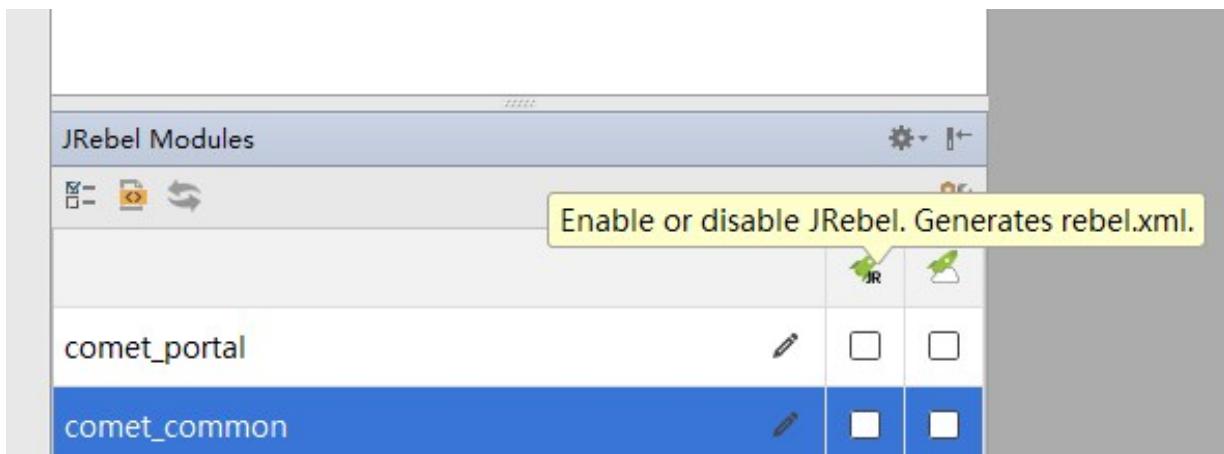


在原来运行项目的按钮边上会多出三个绿色的按钮，一个是 Run With JRebel ，另一个是 Debug With JRebel ，最后一个 Synchronize project with remote server



5.3. 使用

- JRebel在IntelliJ IDEA中的使用，分为本地 Local 热部署与远程 Remote 热部署，操作 Generates rebel.xml 或 Generates rebel-remote.xml，其中本地 Local 热部署需要 rebel.xml 文件，而远程 Remote 热部署需要 rebel.xml 和 rebel-remote.xml 两个文件，xml文件的配置规则。



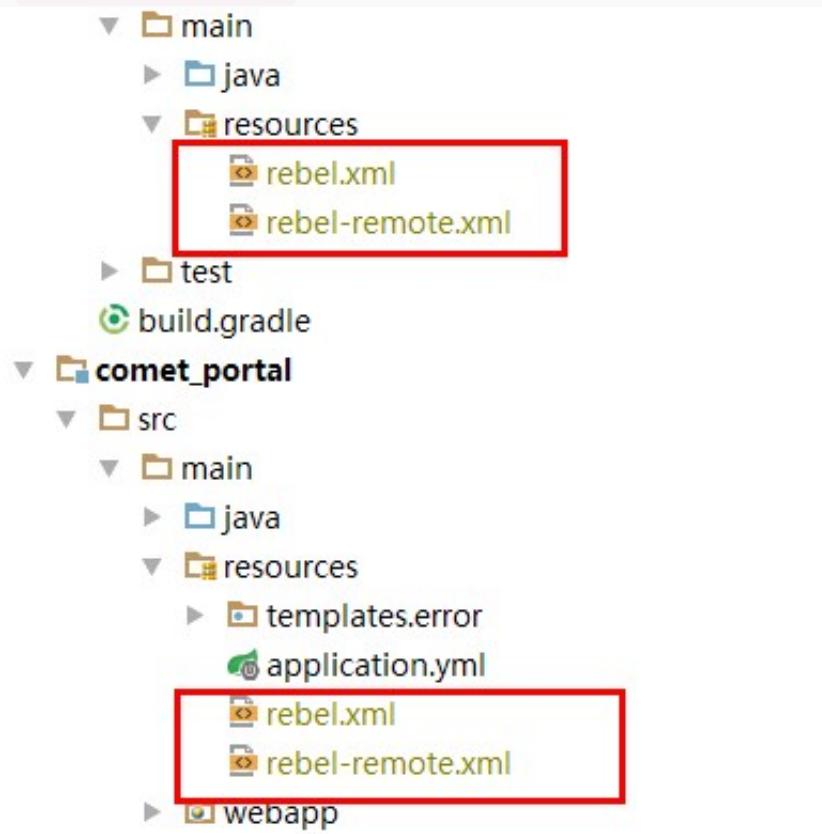
5.3.1. 本地 Local 热部署

- 若启用本地 Local 热部署，在项目的 resources 目录下会生成 rebel.xml
- IntelliJ IDEA 配置 Tomcat，参照上面配置，用 Debug With JRebel 启动项目，若控制台Console Log 有 JRebel 输出的版本信息，没有报错就是表示成功执行了

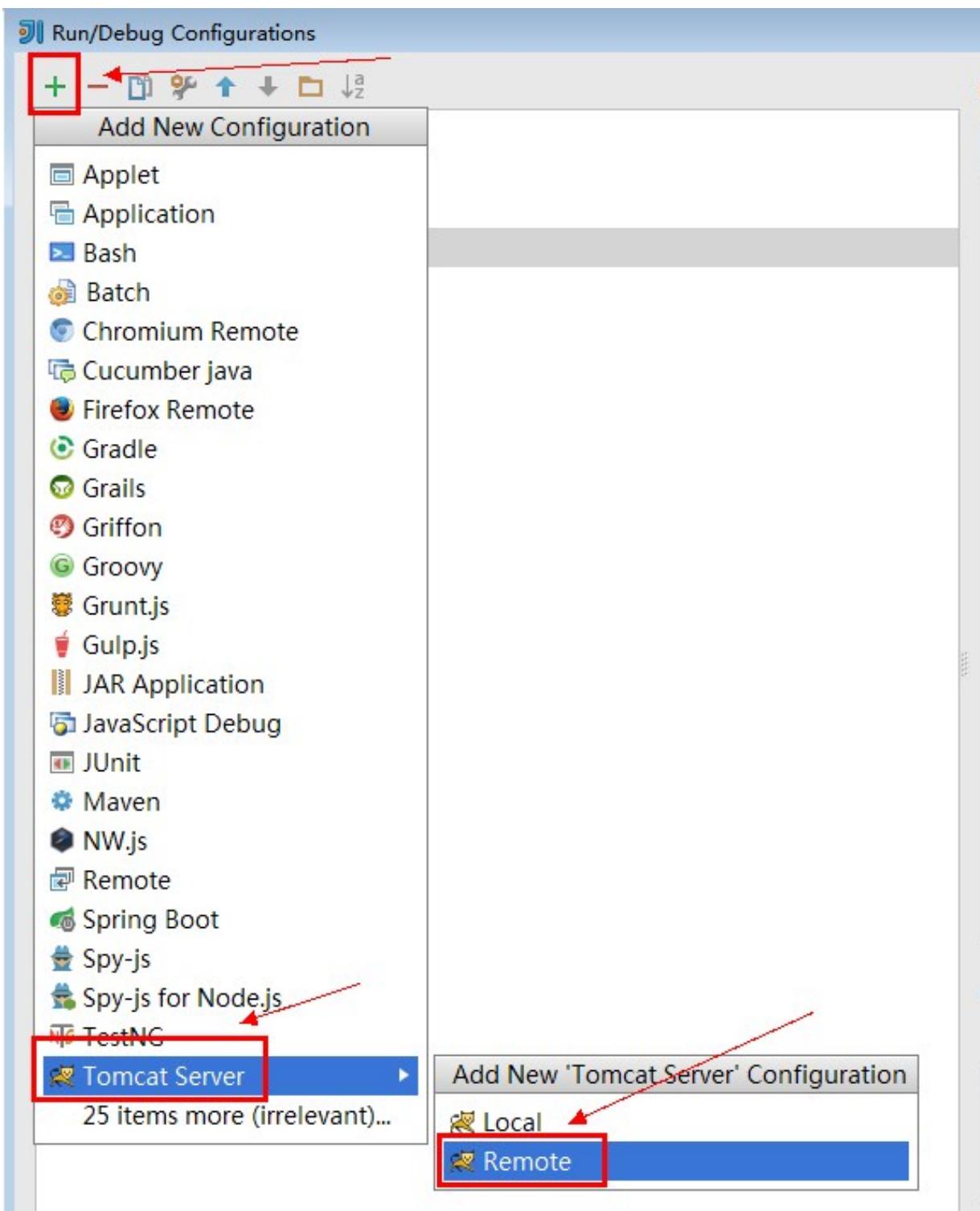
```
Using JRE_HOME:          D:\Java\jdk1.8.0_31
Using CLASSPATH:         "D:\Java\tomcat_portal\bin\bootstrap.jar;D:\Java\tomcat_portal\bin\tomcat-juli.jar"
2015-10-12 17:45:45 JRebel:
2015-10-12 17:45:45 JRebel: #####
2015-10-12 17:45:45 JRebel:
2015-10-12 17:45:45 JRebel: JRebel Agent 6.2.3 (201508181414)
2015-10-12 17:45:45 JRebel: (c) Copyright ZeroTurnaround AS, Estonia, Tartu.
2015-10-12 17:45:45 JRebel:
2015-10-12 17:45:45 JRebel: Over the last 30 days JRebel prevented
2015-10-12 17:45:45 JRebel: at least 10 redeloys/restarts saving you about 0 hours.
2015-10-12 17:45:45 JRebel:
2015-10-12 17:45:45 JRebel: Licensed to anonymous-user
2015-10-12 17:45:45 JRebel: with the following restrictions:
2015-10-12 17:45:45 JRebel: ##### Cracked by anonymous-user, For FUN! Unlimited! Enjoy! #####
2015-10-12 17:45:45 JRebel:
2015-10-12 17:45:45 JRebel: License type: perpetual
2015-10-12 17:45:45 JRebel:
2015-10-12 17:45:45 JRebel:
2015-10-12 17:45:45 JRebel: #####
2015-10-12 17:45:45 JRebel:
2015-10-12 17:45:45 JRebel:
```

5.3.2. 远程 Remote 热部署

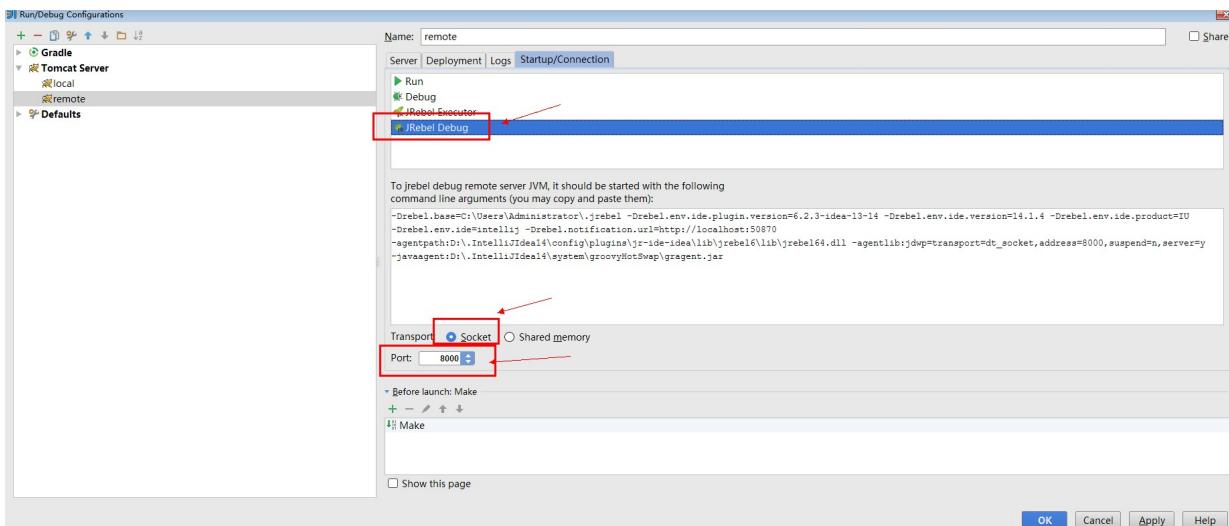
若启用远程 Remote 热部署，在项目的 resources 目录下会生成 rebel.xml 与 rebel-remote.xml 文件，如下图



IntelliJ IDEA配置 Tomcat，若需要远程热部署，则选择



点 + 号选择 Tomcat Server -> Remote



在 Startup/Connection 选择 JRebel Debug -> Transport 选择 Socket , 然后配置下 Port , 其中 Port 需参照远程服务器端Tomcat的启动脚本变量 (远端服务器 \$TOMCAT_HOME/bin 目录的 setenv.sh 或 setenv.bat) 中定义的 address , 以 setenv.sh 为例 :

```
# export CATALINA_OPTS="$CATALINA_OPTS -XX:MaxPermSize=1024m"

# Reset the default stack size for threads to a lower value (by
1/10th original)
# By default this can be anywhere between 512k -> 1024k depending
on x32 or x64
# bit Java version.
# http://www.springsource.com/files/uploads/tomcat/tomcatx-large-
scale-deployments.pdf
# http://www.oracle.com/technetwork/java/hotspotfaq-138619.html
export CATALINA_OPTS="$CATALINA_OPTS -Xss512k"

# Oracle Java as default, uses the serial garbage collector on the
# Full Tenured heap. The Young space is collected in parallel, but
the
# Tenured is not. This means that at a time of load if a full
collection
# event occurs, since the event is a 'stop-the-world' serial event
then
# all application threads other than the garbage collector thread
are
# taken off the CPU. This can have severe consequences if requests
continue
# to accrue during these 'outage' periods. (specifically
webservices, webapps)
# [Also enables adaptive sizing automatically]
export CATALINA_OPTS="$CATALINA_OPTS -XX:+UseParallelGC"

# This is interpreted as a hint to the garbage collector that
pause times
# of <nnn> milliseconds or less are desired. The garbage collector
will
# adjust the Java heap size and other garbage collection related
parameters
# in an attempt to keep garbage collection pauses shorter than
<nnn> milliseconds.
# http://java.sun.com/docs/hotspot/gc5.0/ergo5.html
export CATALINA_OPTS="$CATALINA_OPTS -XX:MaxGCPauseMillis=1500"

# A hint to the virtual machine that it's desirable that not more
than:
# 1 / (1 + GCTimeRatio) of the application execution time be
spent in
# the garbage collector.
# http://themindstorms.wordpress.com/2009/01/21/advanced-jvm-
tuning-for-low-pause/
export CATALINA_OPTS="$CATALINA_OPTS -XX:GCTimeRatio=9"

# The hotspot server JVM has specific code-path optimizations
```

```

# which yield an approximate 10% gain over the client version.
export CATALINA_OPTS="$CATALINA_OPTS -server"

# Disable remote (distributed) garbage collection by Java clients
# and remove ability for applications to call explicit GC
collection
export CATALINA_OPTS="$CATALINA_OPTS -XX:+DisableExplicitGC"
# how-to-remotely-debug-application-running-on-tomcat-from-within-
intellij-idea
# http://blog.trifork.com/2014/07/14/how-to-remotely-debug-
application-running-on-tomcat-from-within-intellij-idea/
export CATALINA_OPTS="$CATALINA_OPTS -
agentlib:jdwp=transport=dt_socket,address=8000,server=y,suspend=n"

export JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8"

# Remote reloading java classes jrebel plugin.
export CATALINA_OPTS="$CATALINA_OPTS -
javaagent:/home/vagrant/tools/jrebel/jrebel.jar -
agentpath:/home/vagrant/tools/jrebel/libjrebel64.so -
Drebel.disable_update=true -Drebel.remoting_plugin=true"
# Check for application specific parameters at startup
if [ -r "$CATALINA_BASE/bin/appenv.sh" ]; then
    . "$CATALINA_BASE/bin/appenv.sh"
fi

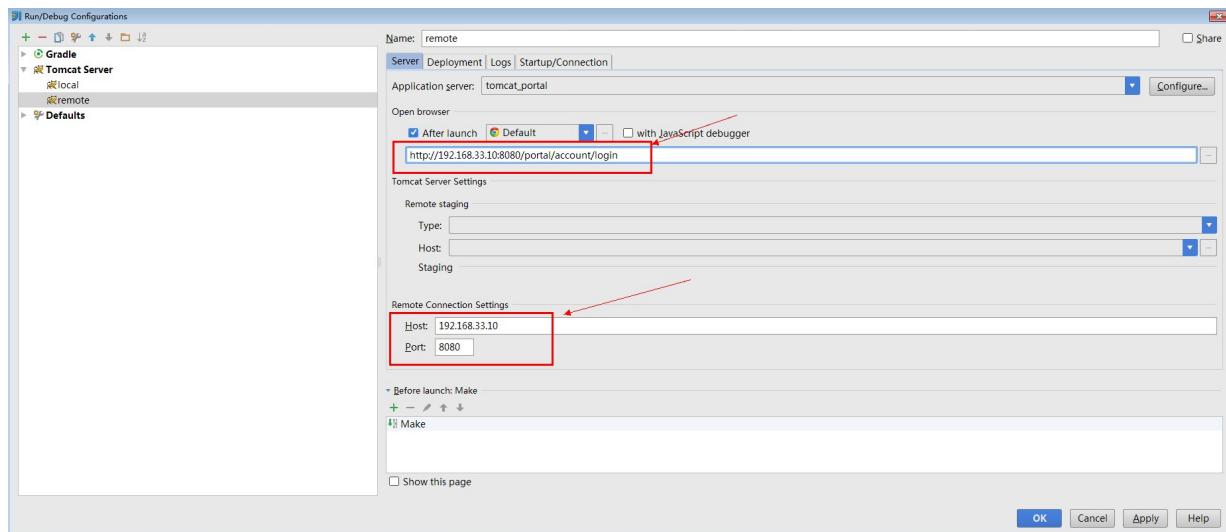
echo "-----"
echo ""
echo "Using CATALINA_OPTS:"
for arg in $CATALINA_OPTS
do
    echo ">> " $arg
done
echo ""

echo "Using JAVA_OPTS:"
for arg in $JAVA_OPTS
do
    echo ">> " $arg
done
echo "-----"
echo ""

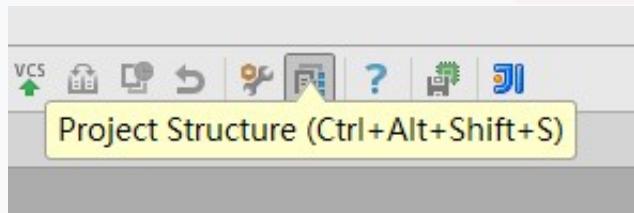
```

- 备注：`-Drebel.remoting_plugin=true` 表示启用JRebel远程热部署，而 `-agentpath:/home/vagrant/tools/jrebel/libjrebel64.so` 表示Linux环境，若为 Windows 环境则为 `-agentpath:D:/jrebel/lib/jrebel64.dll`。

返回 Server 配置下 Open browser 访问路径及 Remote Connection Settings 远程连接的主机名 Host 与端口号 Port

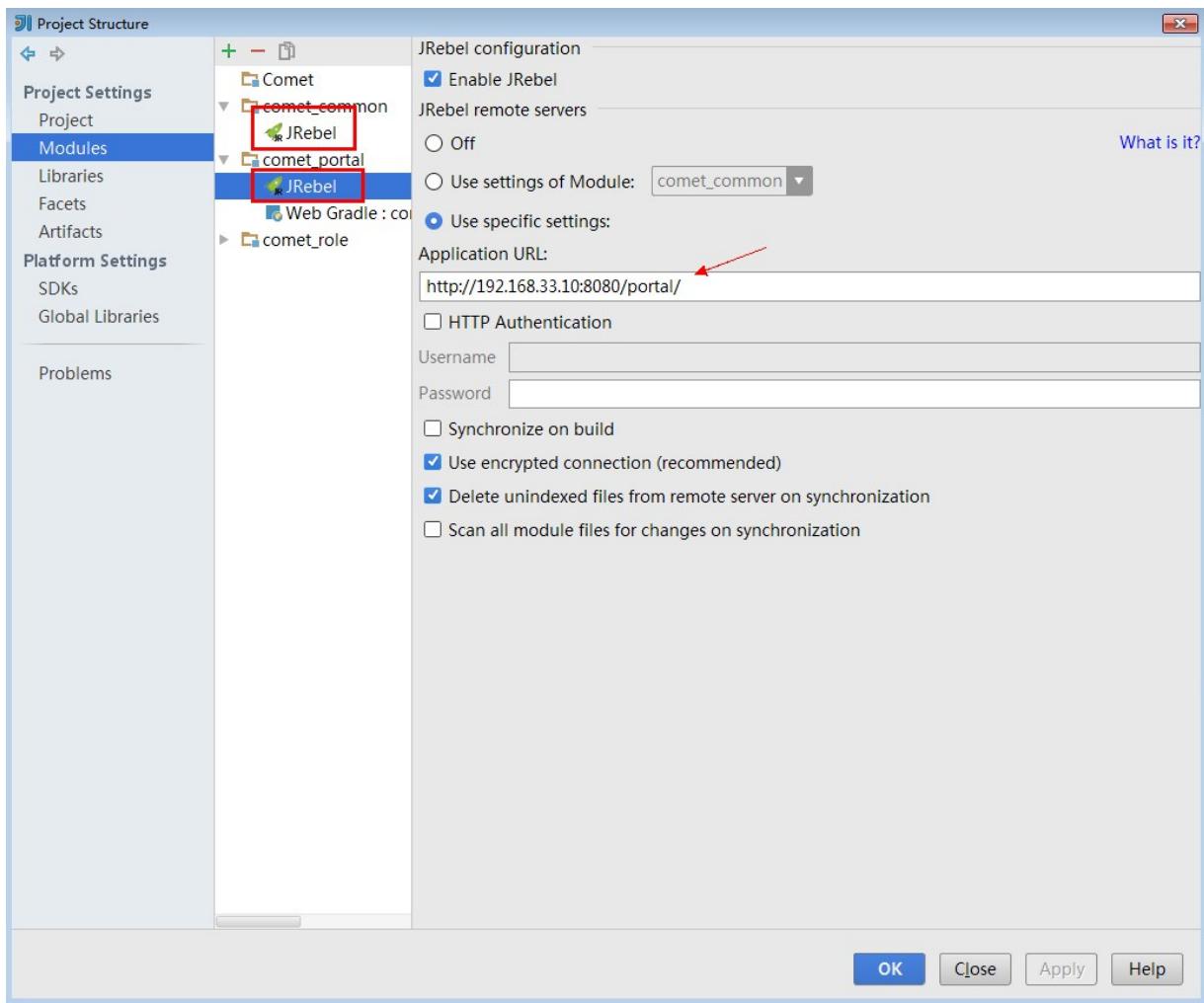


完成上述配置后，打开IntelliJ IDEA 的 Project Structure ，

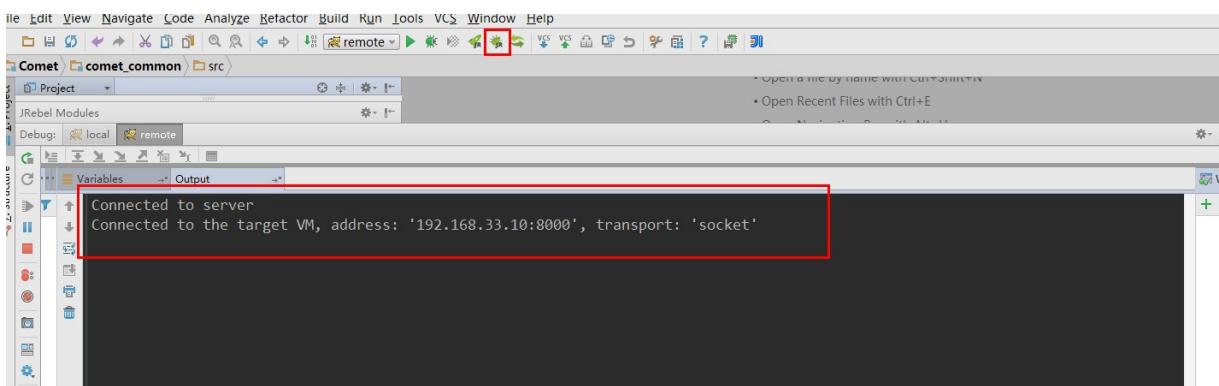


配置下对应 Modules 的 JRebel 的

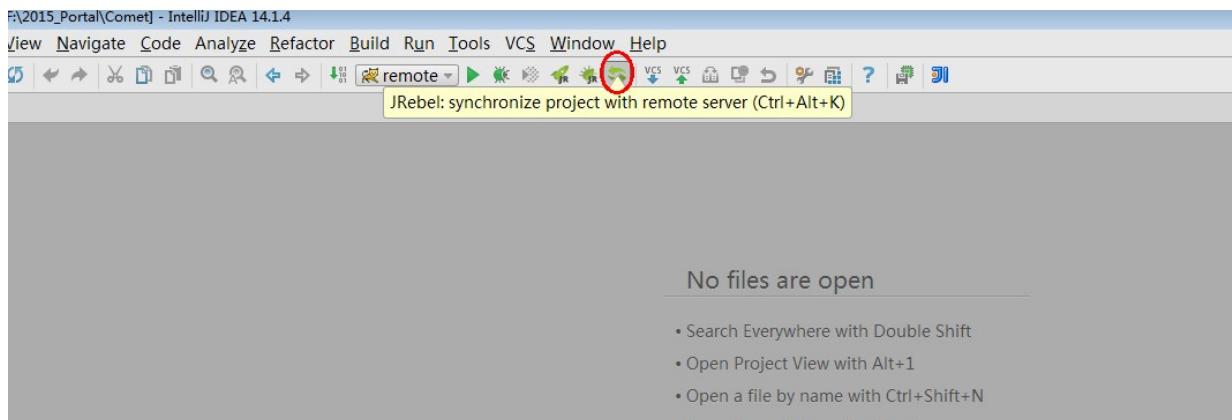
Application URL ，需配置此项否则远程热部署无效，如下图：



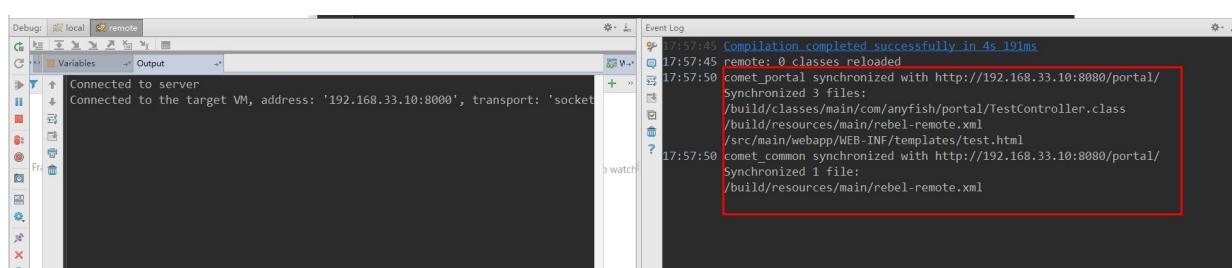
然后用 Debug With JRebel 'remote' 启动



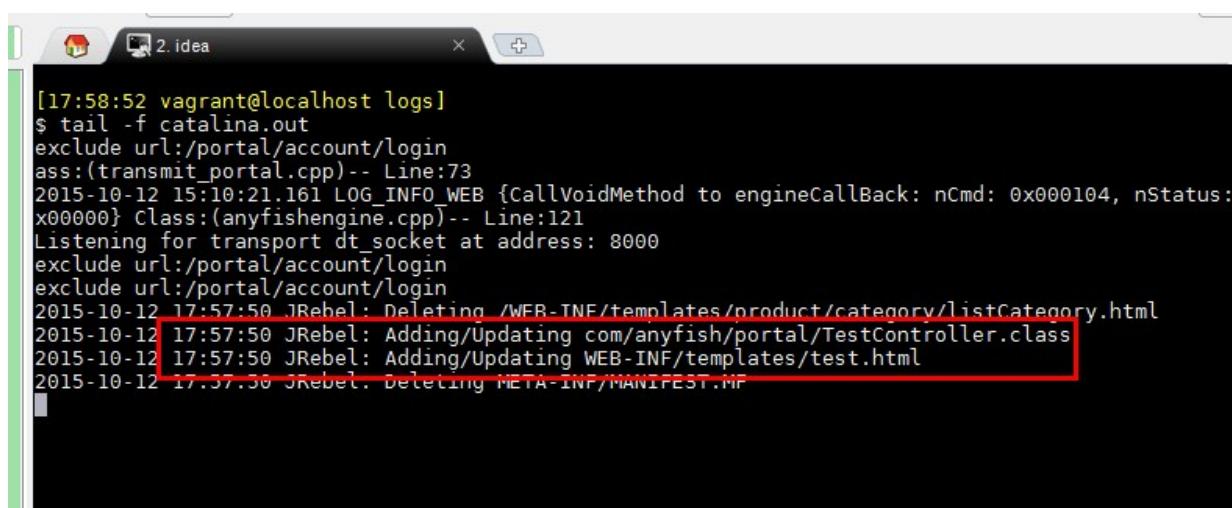
控制台打印出 Connected to server Connected to the target VM, address: '192.168.33.10:8000', transport: 'socket' 表示连接远程Tomcat服务器成功，可以进行模拟热部署操作。
若要远程Tomcat服务器认到新增的类，模板页面等信息，需先 Make 编译下项目并
JRebel: Synchrnoize project with remote server



当控制台输出如下信息：



并观察 remote server 的控制有如下的信息提示，则表示JRebel远程热部署成功。



zeroturnaround 公司的另一个产品xrebel的扩展， -

javaagent:D:/Java/xrebel-2.0.1-crack/xrebel/xrebel.jar

破解的 JRebel_crack 下载，密码： 3yw3

最后致谢 zeroturnaround 公司，<http://zeroturnaround.com/blog/>