
目录

IntelliJ IDEA 教程.....	2
安装配置.....	2
初始配置.....	2
优化配置.....	10
项目管理.....	16
同时管理多项目.....	16
创建 Maven 项目.....	20
导入 Maven 项目.....	23
JRebel 热部署.....	29
常用技巧.....	36
Debug 跟踪条件变量.....	36
Live Template 使用.....	37
代码分析.....	39
代码重构.....	43
粘贴历史复制记录.....	55
查看本地历史记录.....	56
文本比较.....	57
SSH 远程管理.....	58
管理远程主机.....	60
快捷键大全.....	62
编辑.....	62
查找/替换.....	64
编译/运行.....	64
调试.....	64
导航.....	65
重构.....	66
版本控制/本地历史.....	66
Live Template.....	66
代码生成.....	67
如何查找.....	67
新特性.....	68
Terminal.....	68
Search Anywhere.....	69
LENS Mode.....	69
文件夹搜索.....	69
搜索注释内容.....	70
Spring Bean Explorer.....	70

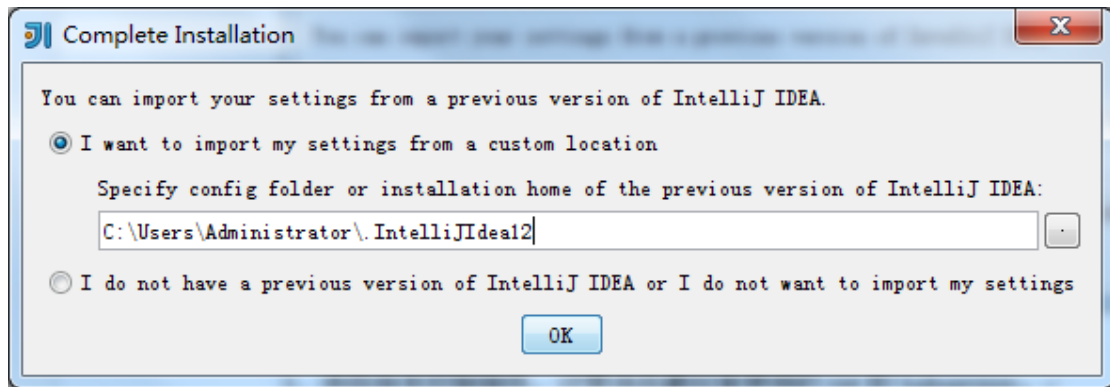
IntelliJ IDEA 教程

安装配置

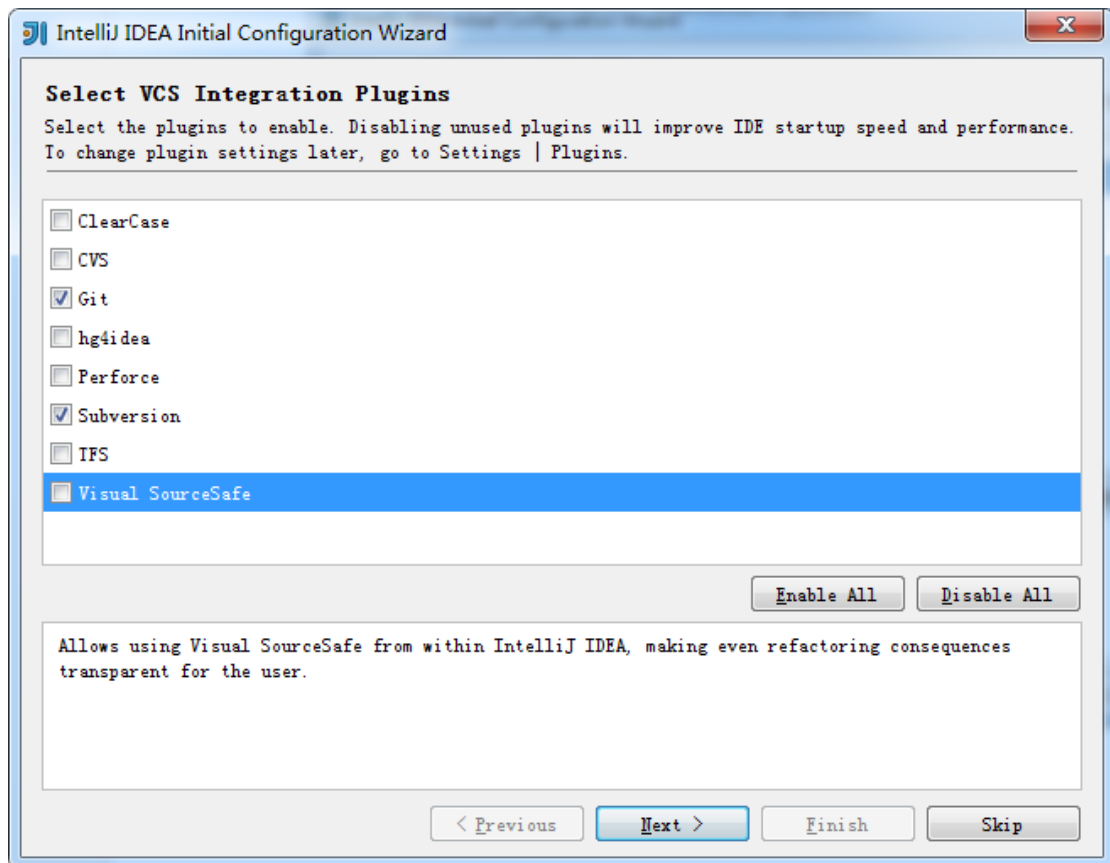
初始配置

安装完 IntelliJ IDEA 13 后会弹出初始配置过程。

1、提示是否导入旧版本的 settings 配置，settings 目录默认位于 C:\Users\{User}\.IntelliJ\Idea12 目录下，通过此步骤，可以将旧版本的配置及插件直接导入到新版本中。



2、选择版本控制插件。



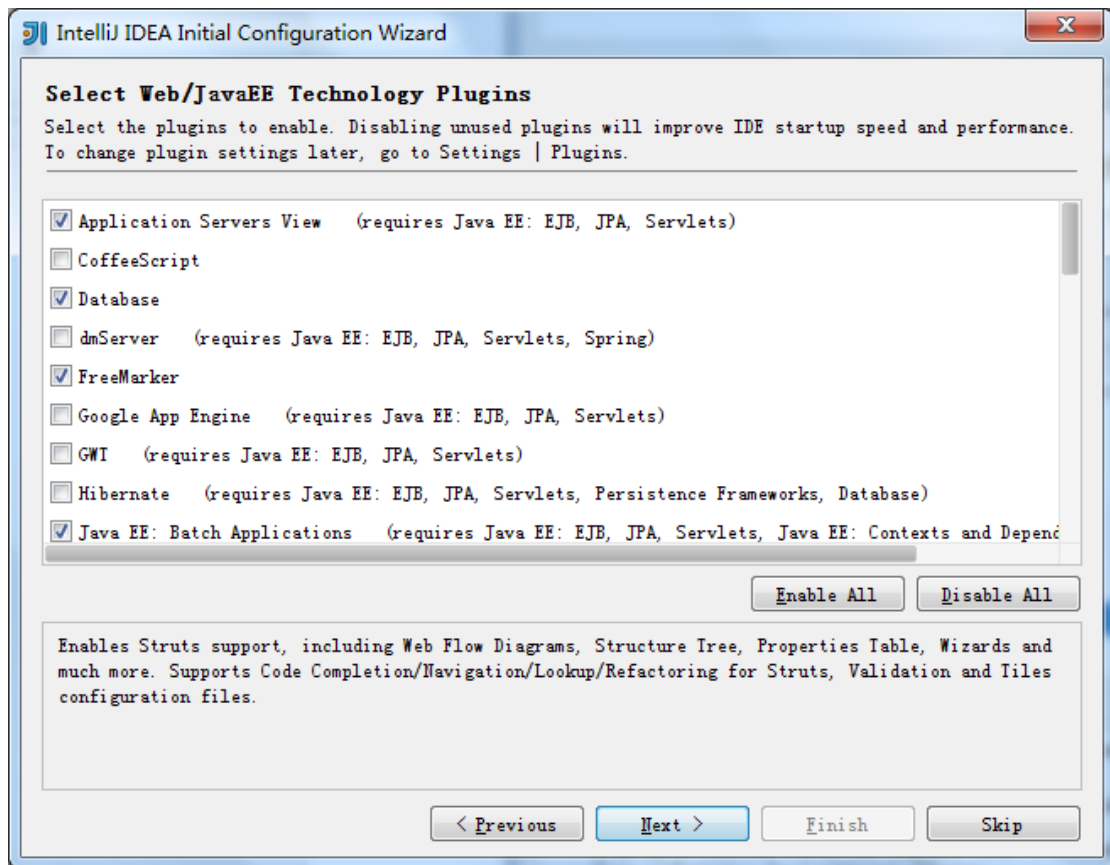
以下是我个人勾选的：

- Git: Git（分布式版本控制工具）插件，需本地安装 Git。
- Subversion: SVN 插件，新版本支持 Subversion1.8

其他插件介绍：

- ClearCase: IBM Rational 的 SCM 管理工具插件。
- CVS: CVS 插件。
- hg4idea: Mercurial 插件，与 Git 类似的分布式版本控制工具。
- Perforce: Perforce 插件，商业的版本控制工具。
- TFS: Team Foundation Server 插件，微软的客户端-服务器源代码管理系统。
- Visual SourceSafe: VSS 插件，微软的客户端的源代码管理系统。

3、选择 Web/Java 开发插件，勾选对自己有用的插件。



以下是我个人勾选的：

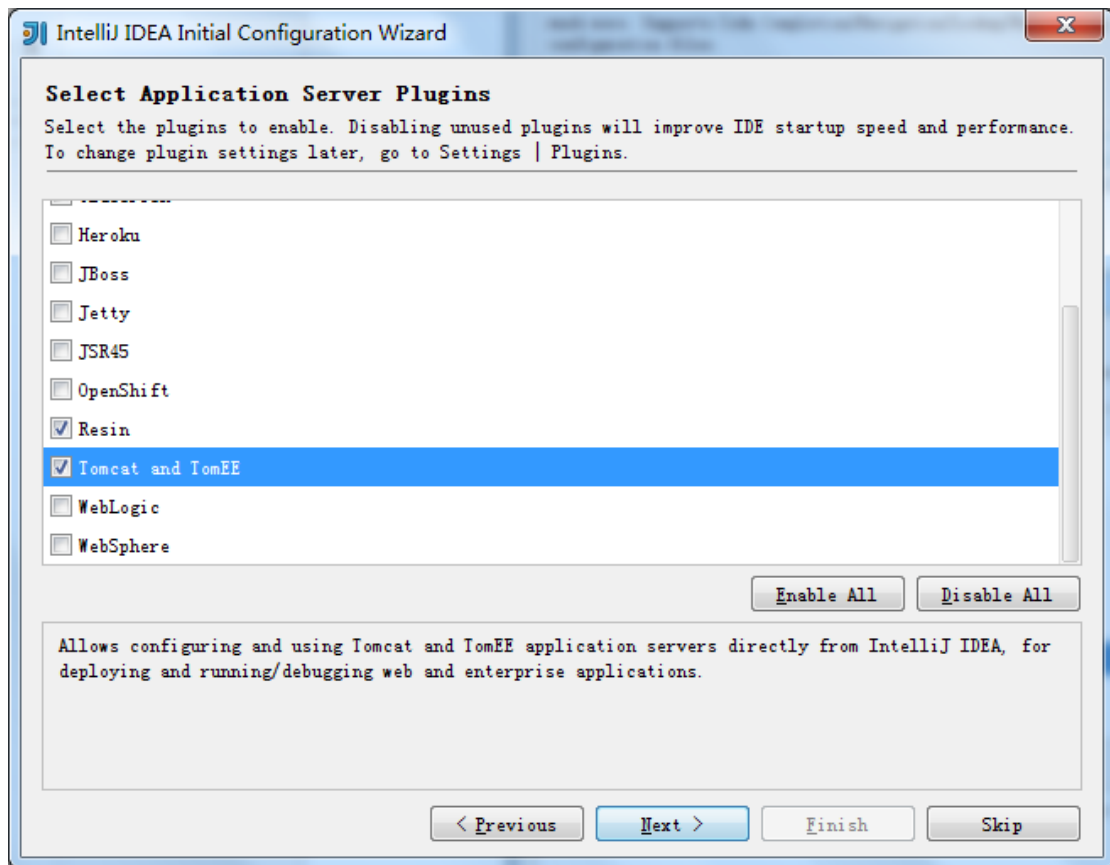
- Application Servers Views: 配置应用服务器插件。
- Database: 数据库插件，可用于管理 MySQL、Oracle、SQLite 等。
- Freemarker: 支持 freemarker 语法插件。
- Java EE: Batch Applications: 新版本增加的功能，支持 Java EE 7 批处理编程模型(JSR-352)。
- Java EE: Bean Validator: 支持 Java EE 6 的数据验证模型(JSR-303)。
- Java EE: Contexts and Dependency Injection: 支持 Java EE6 的依赖注入模型(JSR-299)。
- Java EE: EJB, JPA, Servlets: EJB、JPA、Servlet 的插件。
- Java EE: JMS, JSON Processing, Concurrency Transaction: JMS, JSON, Transaction 等的插件。
- Java EE: RESTful Web Services: JAX-RS 插件。
- Java EE: Web Services: JAX-WS 插件。
- Java Server Pages: JSP 插件。
- Persistence Frameworks: 持久化(JPA、Hibernate)插件。
- Spring Batch: Spring 批处理框架的插件。
- Spring Data: Spring 数据访问框架(Mongodb、Redis、Hadoop)插件。
- Spring Security: Spring 安全框架的插件。
- Spring: Spring 插件

-
- Spring Web Services: Spring Web Services 插件。
 - Spring-AOP and @AspectJ: Spring-AOP 和切面语言的插件。
 - SQL: SQL 插件

其他插件介绍:

- CoffeeScript: CoffeeScript 插件, 基于 Javascript 之上的一门编程语言。
- dmServer: dmServer 插件, 基于 OSGi 的模块化部署的 java 服务器。
- Google App Engine: GAE 插件, 用于创建 GAE 项目。
- GWT: GWT 插件, 支持 GWT 代码提示、编译、组件开发等。
- Hibernate: Hibernate 插件, 支持 Hibernate 代码提示、反向生成代码等。
- Java EE: Java Server Faces: JSF 插件, 支持 JSF 语法。
- Java EE: WebSockets: 13 版本新功能, 支持 Java EE WebSockets(JSR-356)。
- JBoss Seam Pageflow: Jboss Seam PageFlow 插件。
- Jboss Seam Pages: Jboss Seam Page 插件。
- Playframework: Playframework 插件, 一个 full-stack 的 Java web 框架。
- Spring Integration Patterns: Spring 企业应用集成框架插件。
- Spring OSGi: Spring OSGi 插件。
- Spring Roo Console: Spring Roo 控制台, 支持 Spring Roo 命令提示等。
- Spring Web Flow: Spring 工作流插件。
- Struts 1.x: Struts1 插件, 支持 Struts1 语法提示, 结构化显示 Action、Form 等。
- Struts 2: Struts2 插件, 支持 Struts2 语法(Xml、Tag)提示, 结构化显示 Action 等。
- Tapestry: Tapestry 插件, 一个 MVC 与模板技术结合的 Java 框架。
- Vaddin: Vaddin 插件, 一个基于 GWT 的 Web RIA 框架。
- Velocity: Velocity 插件, 支持 Velocity 语法提示。

4、选择应用服务器插件, 这里指选择了 Resin 和 Tomcat。



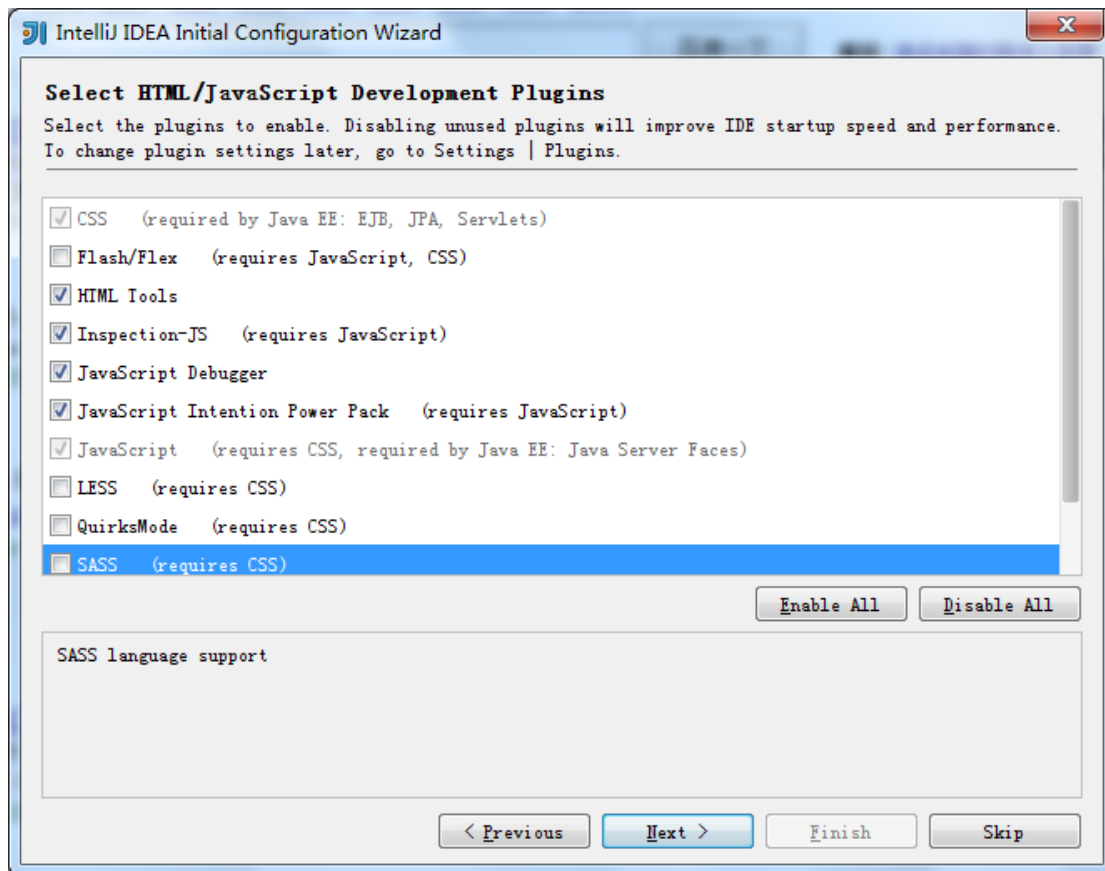
以下是我个人勾选的：

- Resin: Resin 插件。
- Tomcat and TomEE: Tomcat 或 TomEE 服务器插件，TomEE 是经过 J2EE 6 认证的 Tomcat 企业版本

其他插件介绍：

- Cloud Foundry: VMware 主导基于 Spring 的开源 PaaS 云计算平台。
- CloudBees: 基于 Tomcat 和 MySQL 的开源 PaaS 云计算平台。
- Geronimo: Apache 的 J2EE 服务器。
- GlassFish: Sun 的 J2EE 服务器。
- Heroku: Heroku 是一个商业的 Rails 的 PaaS 云计算平台。
- Jboss: Jboss 服务器插件。
- Jetty: 轻量级的 Servlet 服务器。
- JSR45: 兼容 JSR-45 的所有应用服务器, JSR-45(Debugging Support for Other Languages) 为那些非 JAVA 语言写成, 却需要编译成 JAVA 代码, 运行在 JVM 中的程序, 提供了一个进行调试的标准机制。
- OpenShift: 红帽的开源 PaaS 云计算平台。
- WebLogic: Oracle 的商业 J2EE 服务器。
- WebSphere: IBM 的商业 J2EE 服务器。

5、选择 HTML/Javascript 开发插件



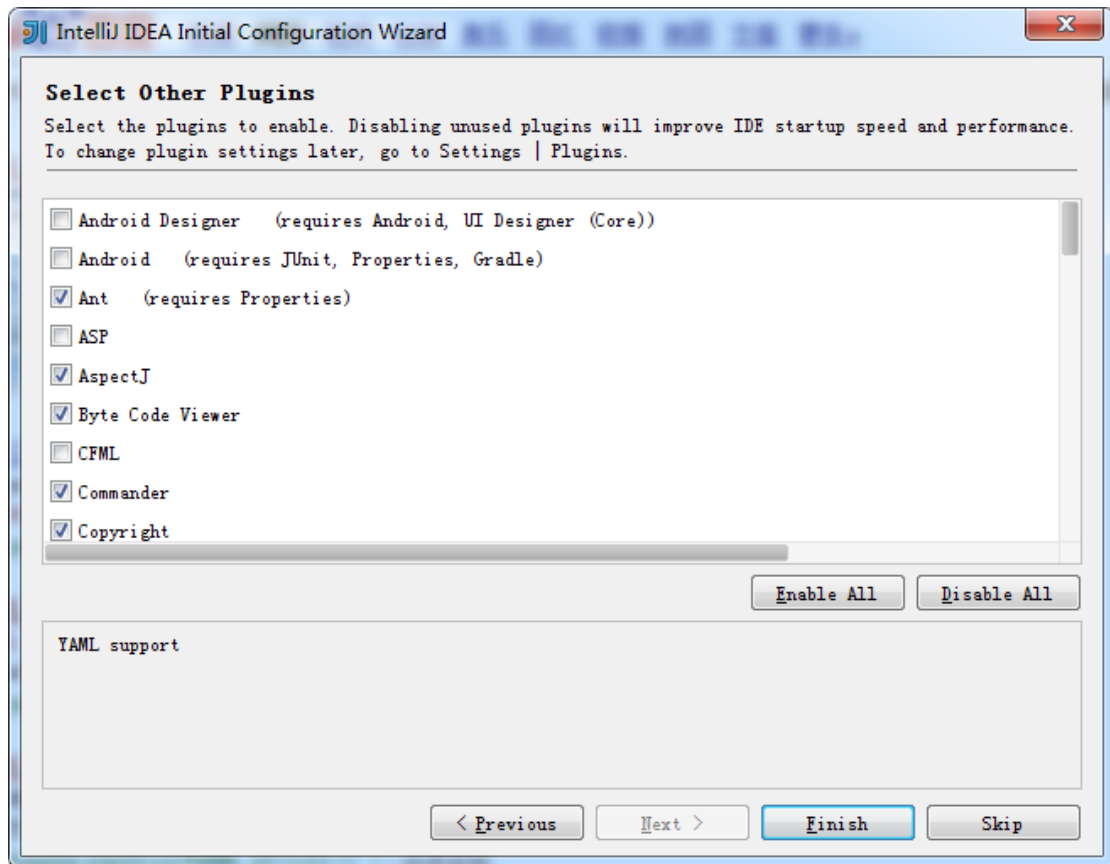
以下是我个人勾选的：

- CSS: CSS 插件，可以直接显示 css 配色的颜色。
- HTML Tools: Html 插件，支持 emmet 快速编写 html 代码。
- Inspection-JS: JS 代码检测，目前还没见过哪个 IDE 对 JS 的支持有这么智能。
- JavaScript Debugger: js 调试器，需 chrome 安装 Debugger 插件才可以支持。
- Javascript Intention Power Pack: 补充上面 JS 代码检测的不足。
- Javascript: Javascript 插件。
- QuirksMode: 用于检测 CSS 和 HTML 的主流浏览器兼容性问题。
- W3C Validators: W3C 标准检测插件。

其他插件介绍：

- Flash/Flex: Flash/Flex 开发插件。
- LESS: LESS 插件，LESS 是一个 CSS 预处理器，通过简单的语法和变量对 CSS 进行扩展。
- SASS: SASS 语法支持，SASS 扩展了 CSS，使用特定的语法来编写 CSS。
- Stylus: Stylus 插件，Stylus 是一个 CSS 预处理器。

6、选择其他插件



以下是我个人勾选的：

- Ant: Ant 插件。
- AspectJ: AspectJ 切面框架插件。
- Byte Code Viewer: java 字节码反编译查看插件。
- Commander: 提供了左右两个用于查看项目结构的插件，可用于项目结构对比或导航。
- Copyright: 版权声明插件，保证版权信息的一致。
- Coverage: 查看代码覆盖率插件。
- Cucumber for Java: Java 的 Cucumber 插件，Cucumber 是一个 BDD 驱动的自动化测试工具。
- DSM Analysis: 架构可视化插件，展示模块间的依赖信息。
- Eclipse: 支持导入 eclipse 结构的项目。
- Emma: 检测代码覆盖率插件
- Gherkin: Gherkin 语言插件，Cucumber 要用到。
- Github: Github 集成插件。
- IntelliLang: 主要用于注解语法的注入验证、正则表达式语法检查等
- Junit: Junit 单元测试插件。
- Maven: Maven 插件。

- Maven Integration Extension: Maven 依赖分析图插件。
- Properties: 属性文件(.properties)编辑插件。
- Refactor-X: Xml 代码格式化插件。
- Remote Hosts Access: 远程主机访问, 支持 ftp/ssh。
- REST Client: 用于访问 REST Web Service 的客户端插件。
- SSH Remote Run: 支持通过 Terminal 运行 SSH 脚本。
- Structural Search: 支持通过语法表达式进行搜索或替换。
- Task Management: 任务管理插件, 支持 YouTrack, JIRA, Lighthouse, Pivotal Tracker, GitHub, Redmine,Trac 等问题跟踪系统。
- Terminal: 终端命令插件。
- TestNG-J: TestNG 插件。
- Time Tracking: 任务管理插件中使用到的时间跟踪功能。
- Type Migration: 类型重构优化插件, 对不够完善的代码提示重构, 比如, 静态方法通过对象来调用而不是通过类调用等等。
- UML: UML 插件。
- XPathView+XSLT: XPath 和 XSLT, 支持高亮、分析, 自动补全等。
- XSLT-Debugger: XSLT 调试工具。
- ZKM-Unscramble: 分析 Java 堆栈跟踪插件。

其他插件介绍:

- Android Designer: 安卓 UI 设计器
- Android: 安卓插件
- ASP: ASP 编辑器
- CFML: ColdFusion 标记语言插件, ColdFusion 是一个动态 Web 服务器, 其 CFML 是一个类似 JSTL 的程序语言。
- Cucumber for Groovy: Groovy 的 Cucumber 插件, Cucumber 是一个 BDD 驱动的自动化测试工具。
- Gradle: Gradle 插件, Gradle 是一个类似 Maven 的 Java 构建工具。
- Grails: Grails 插件, Grails 是 Rails 的 Groovy 实现。
- Groovy: Groovy 插件, Groovy 是一种基于 JVM 的动态脚本语言。
- GuiceyIDEA: Guice 插件, Guice 是 Google 开发的 Java IOC 框架。
- HAML: HAML 插件, HAML 是一种 Rails 下的模板语言。
- IDEtalk: IDEA 的即时通讯工具, 用处不大。
- J2ME: J2ME 插件。
- JavaFX: JavaFX 插件, JavaFX 是 Sun 发布的 RIA 技术。
- Jboss Drools: Drools 插件, Drools 是一种 Java 业务规则引擎。
- Jboss jBPM: jBPM 插件, jBPM 是一种 Java 工作量引擎。

- Osmorc: OSGi 插件。
- Plugin DevKit: IDEA 插件开发工具。
- UI Designer: Swing UI 设计插件。
- UI Designer(Core): Swing UI 设计插件。
- YAML: YAML 插件, YAML 是一种数据序列化格式。

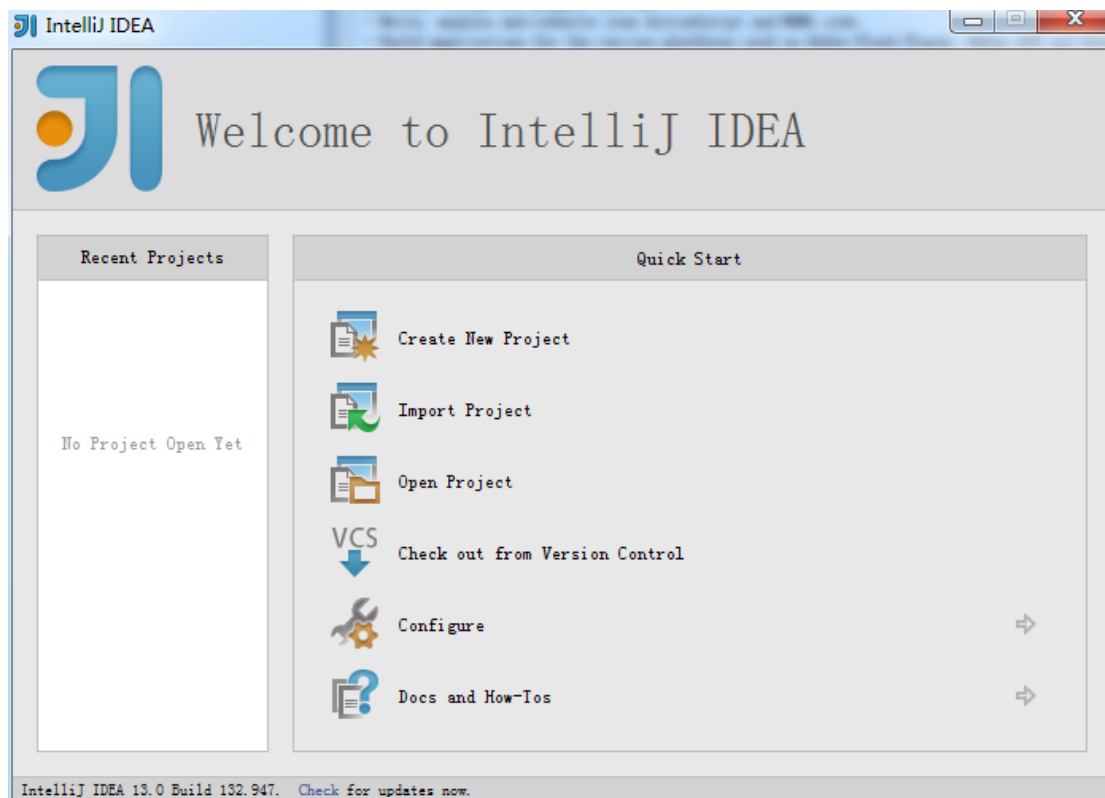
7、小结:

1、idea 是很吃内存的 IDE, 所以对于没用到或者不了解的插件尽可能不选, 有助于减少内存消耗, 以及提高启动速度。

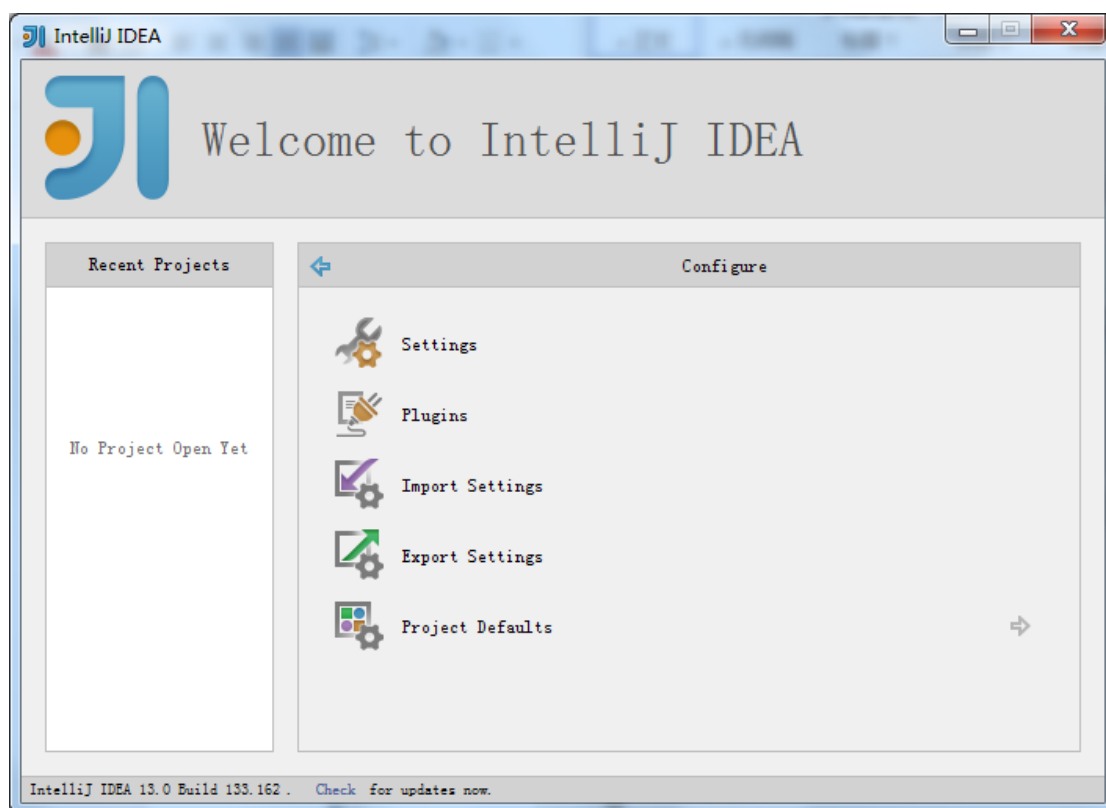
2、如果发现有些插件在初始配置中设置错误, 可通过 Settings-Plugins 来启用或禁用。

优化配置

1、打开 idea, 点击 Configure



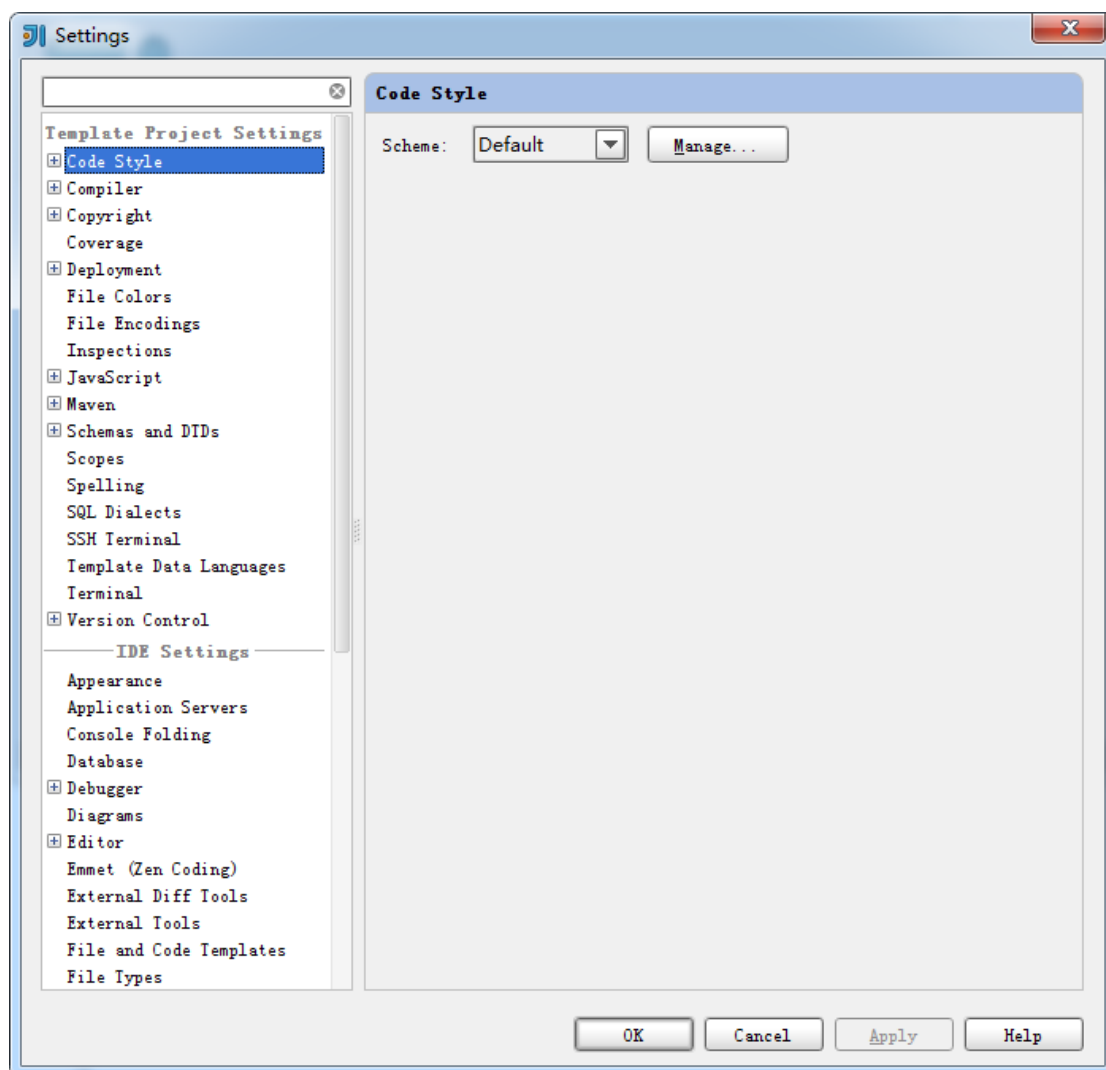
2、进入配置导航页



介绍以下功能：

- **Settings:** IDEA 配置，下面会重点讨论。
- **Plugins:** 插件管理，可以重新禁用或启用插件。
- **Import Settings:** 导入旧的配置文件，配置文件是 jar 格式。
- **Export Settings:** 导出配置文件，定期导出配置文件会减少很多不必要的麻烦。
- **Project Defaults:** 项目配置，包括 SDK、Server 等配置，可以在创建项目后再配置。

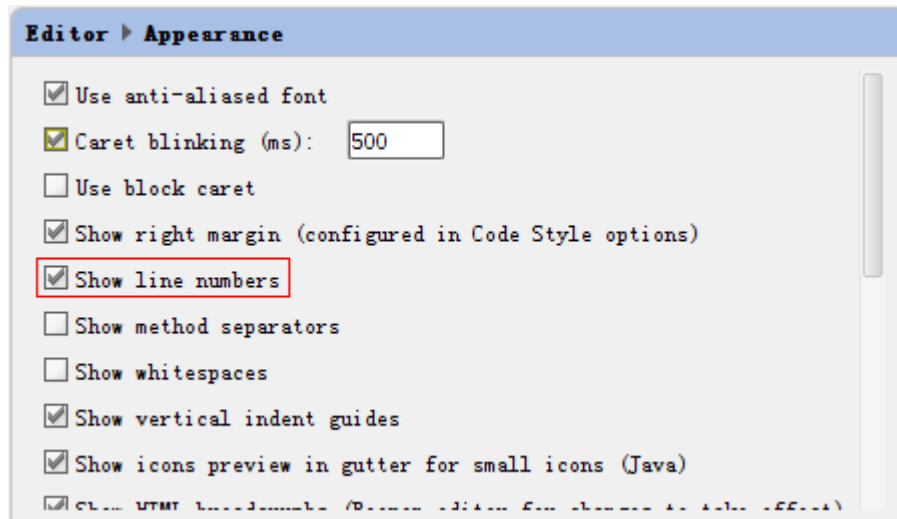
3、打开 Settings 窗口



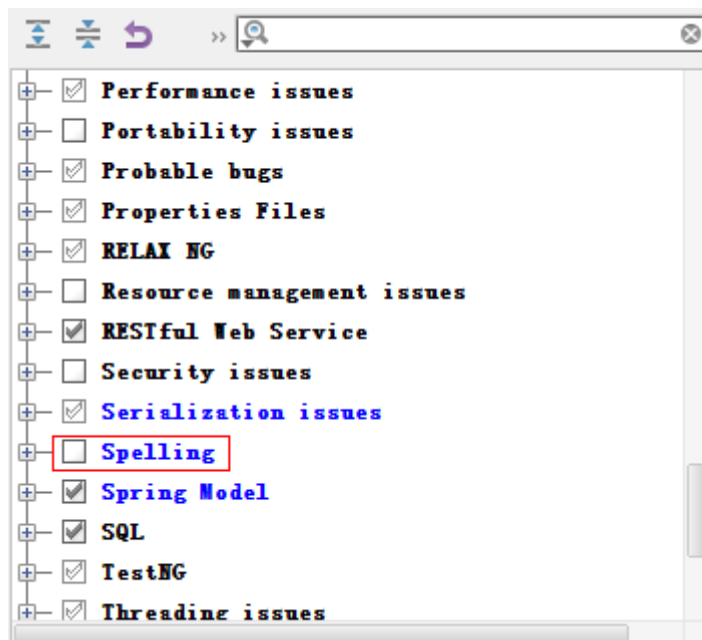
Settings 分为两部分，分别是 Template Project Settings 和 IDE Settings。

- Template Project Settings 是针对每个项目，不同项目的配置都不一样。
- IDE Settings 是 IDE 配置，所有项目的配置都一样。

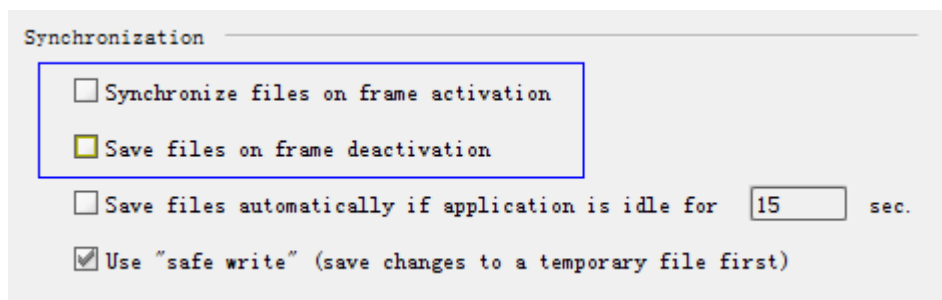
4、显示行号，打开 Settings->Editor->Appearance，勾选“Show line numbers”

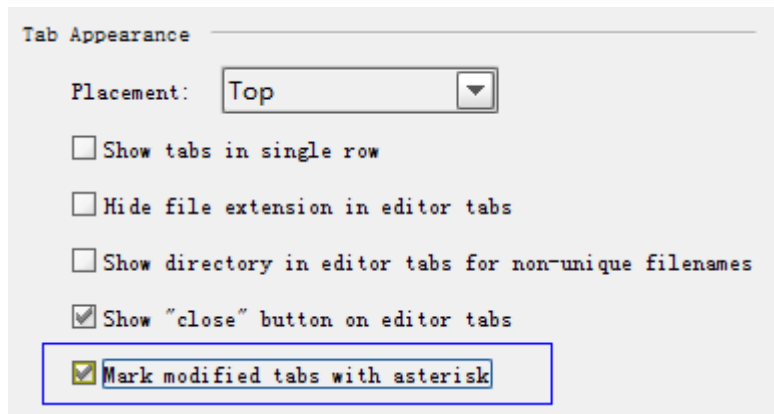


5、取消拼写检查，打开 Settings->Inspection，取消“Spelling”

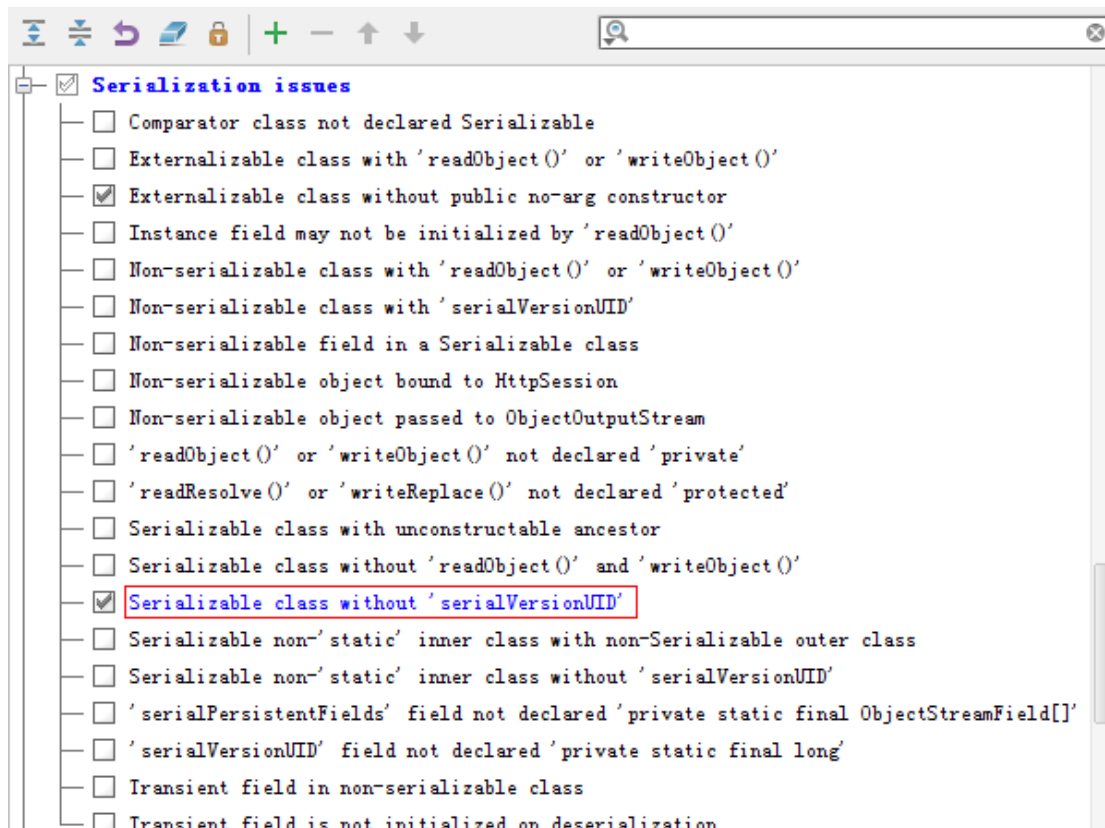


6、关闭自动保存，打开 Settings-General，反选“Synchronize file on frame activation”和“Save files on frame deactivation”。同时修改未保存的显示星号，打开 Settings-Editor->Editor Tabs，勾选“Mark modified tabs with asterisk”。

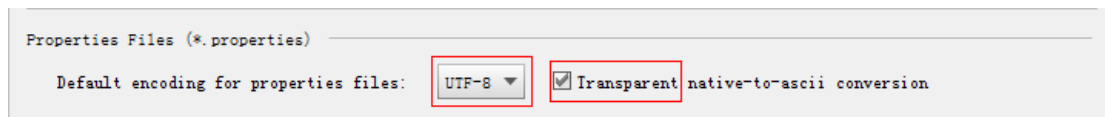




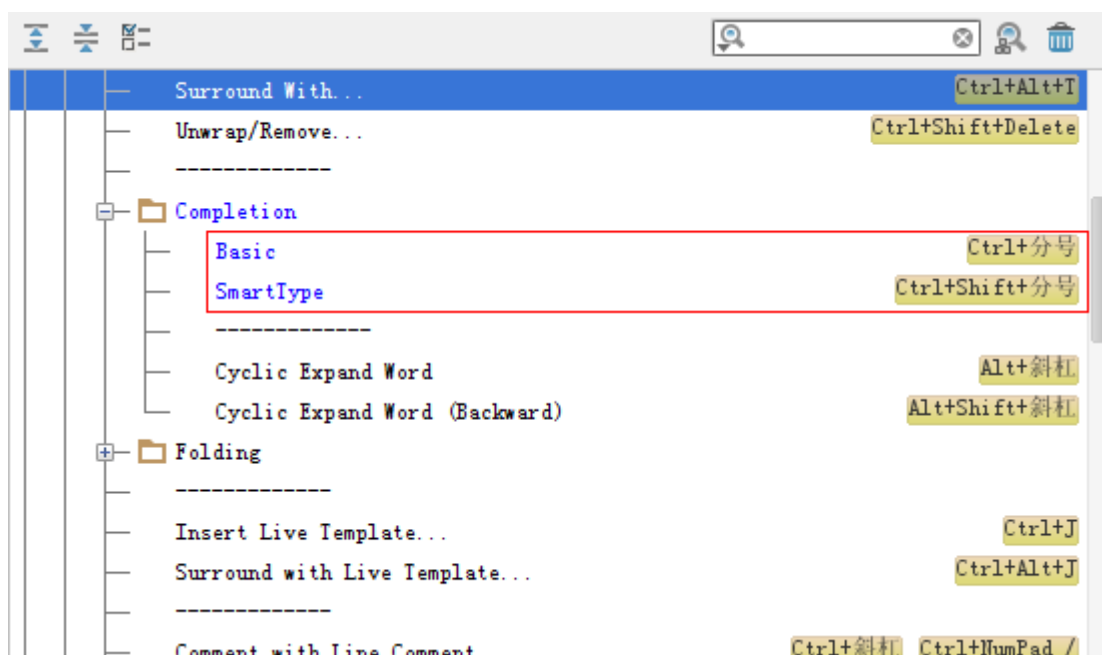
7、开启序列化 serialVersionUID 检测，打开 Settings->Inspections，



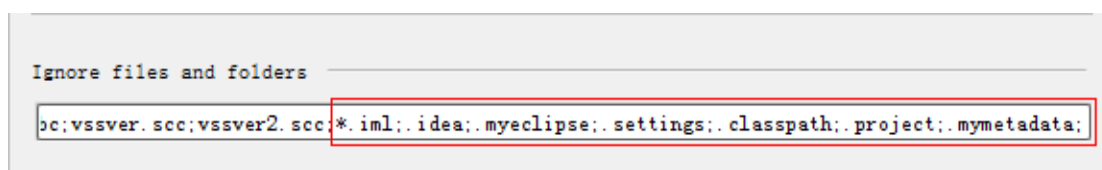
8、修改属性资源文件(.properties)的编码，打开 Settings->File Encoding，设置 Properties File 的编码为 UTF-8，并勾选上“Transparent native-to-ascii conversion”



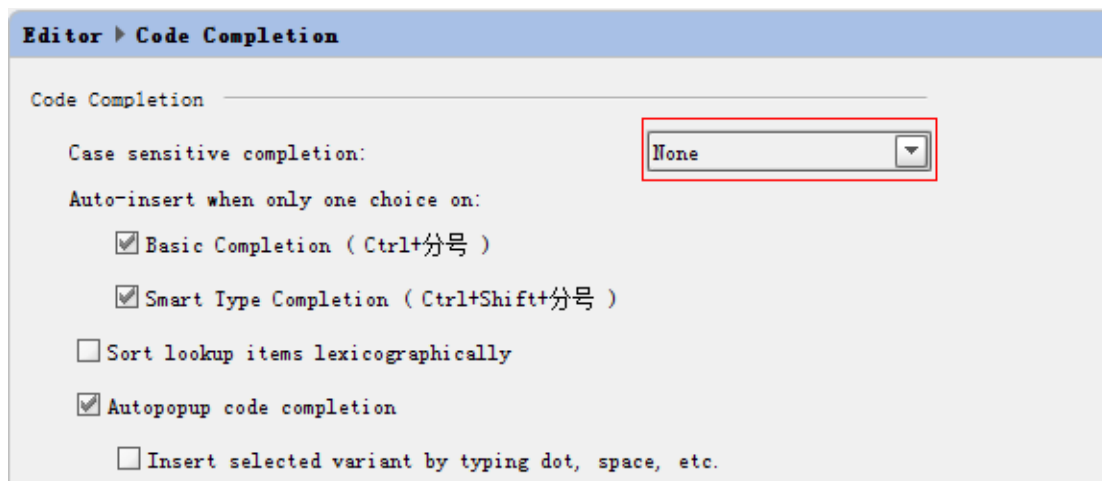
9、修改代码提示快捷键与输入法快捷键冲突的情况。打开 Settings-keymaps，展开下拉列表 Main menu->Code->Completion，修改 Basic 和 SmartType 快捷键为个人喜好。



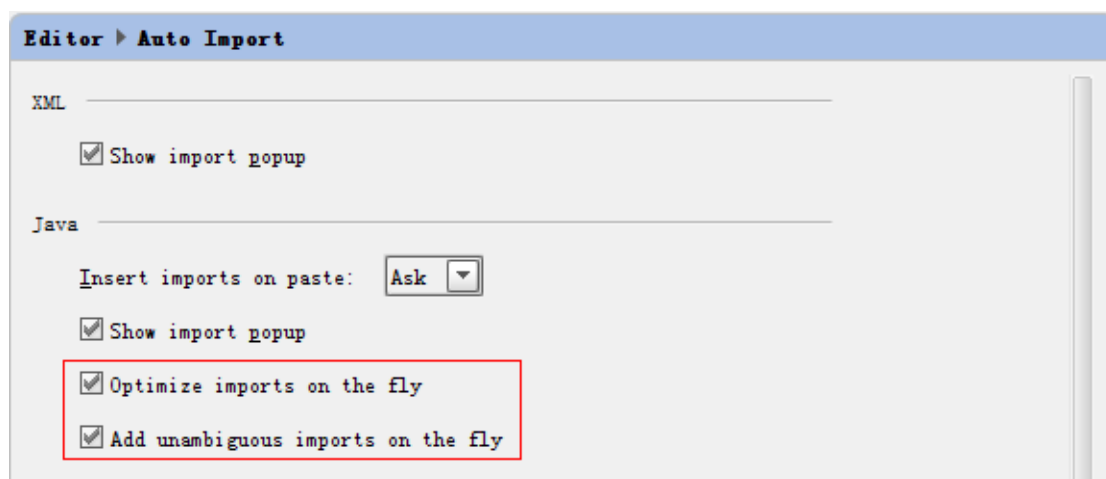
10、隐藏没用到的文件，比如 IDEA 的项目配置文件 (*.iml 和 *.idea)，打开 Settings-File Types，加入要隐藏的文件后缀。



11、代码提示不区分大小写，打开 Settings->Editor->Code Completion，将“Case sensitive completion”设置为 None。



12、自动 import 类型，打开 Settings->Editor->Auto Import，勾选“Optimize imports on the fly”和“Add unambiguous imports on the fly”。

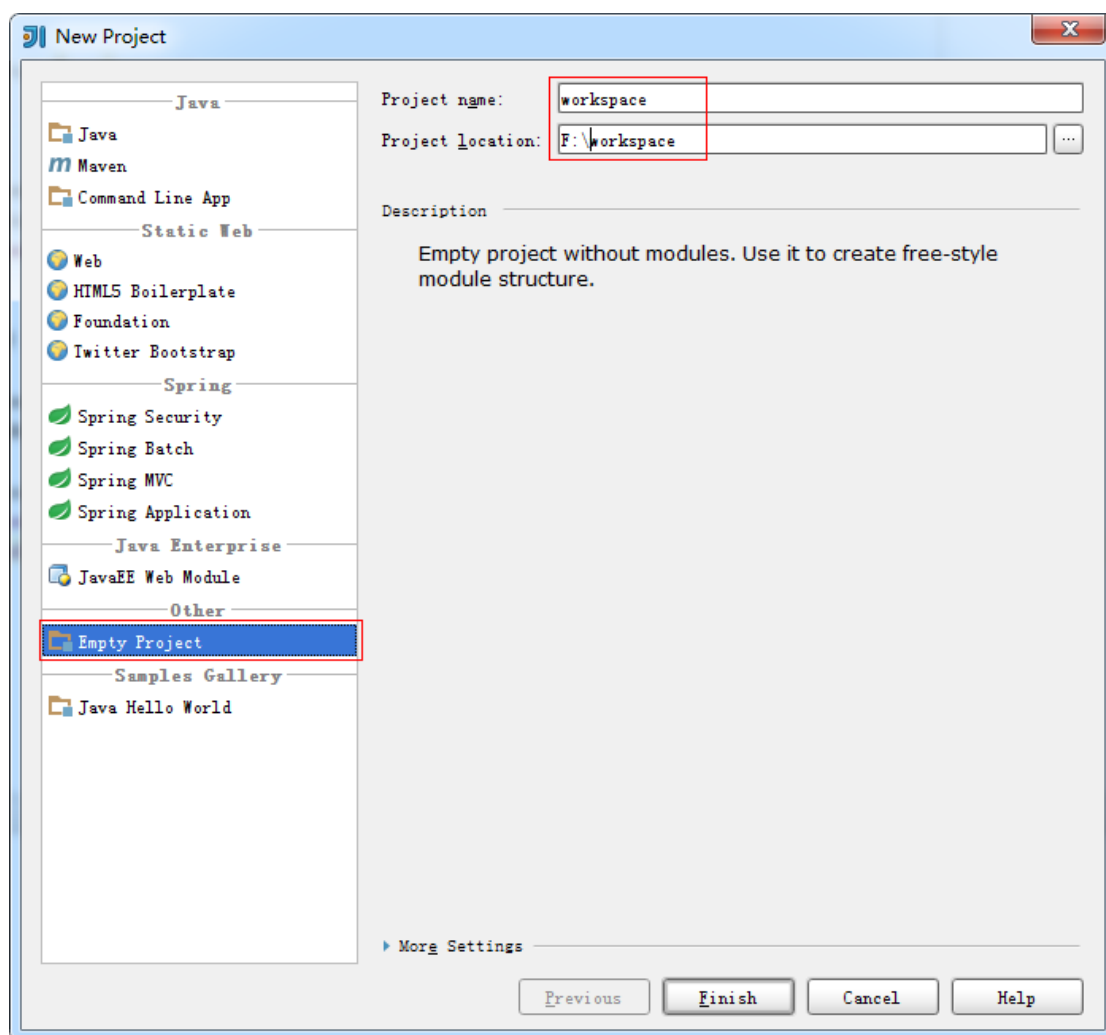


项目管理

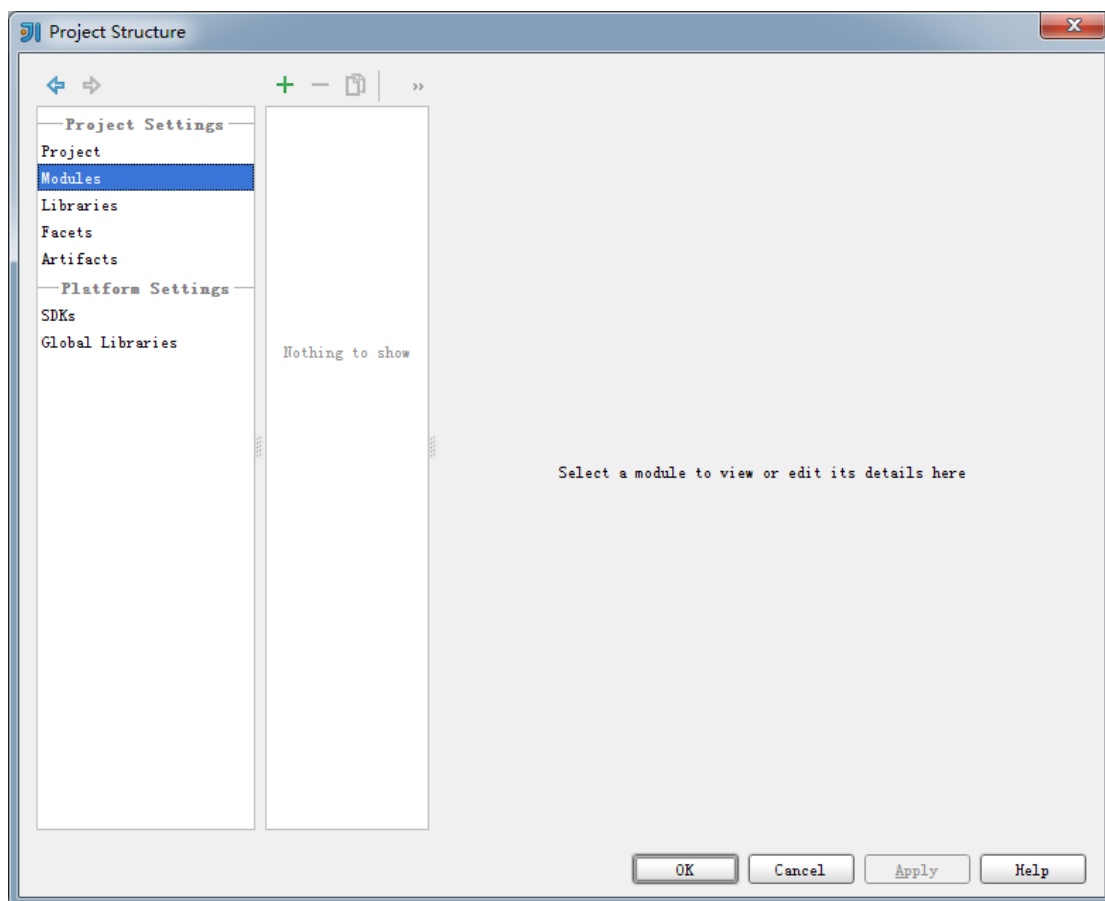
同时管理多项目

IDEA 一个窗口只能管理一个项目，对用惯了 Eclipse 的同学来说可能会不大方便。思维转换下，把 IDEA 项目看成工作空间，IDEA 模块看成项目，就可以实现一个窗口中管理多个项目。下面介绍下如何实现多项目管理。

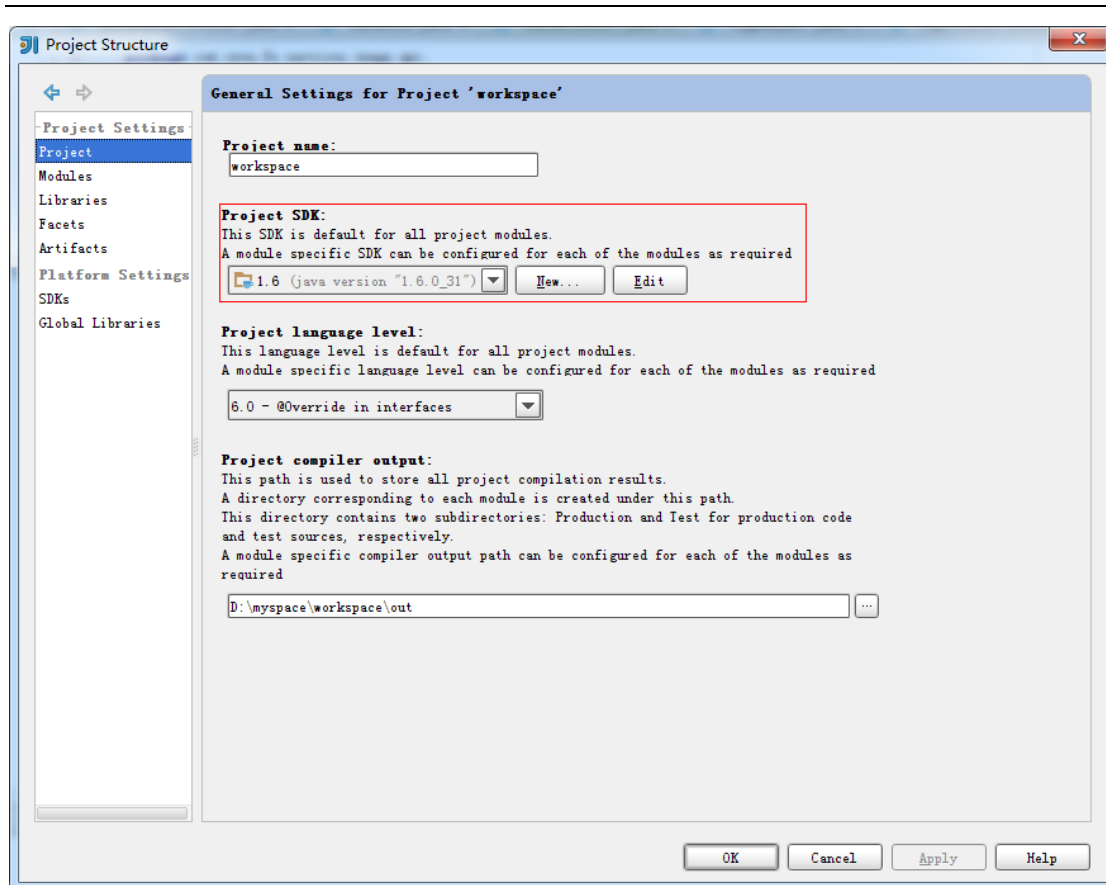
1、创建空项目（空项目当做工作空间）。选择 **File->New Project**，选择 **Empty Project**，修改 **Project Name** 为项目名称，**Project location** 为项目路径。



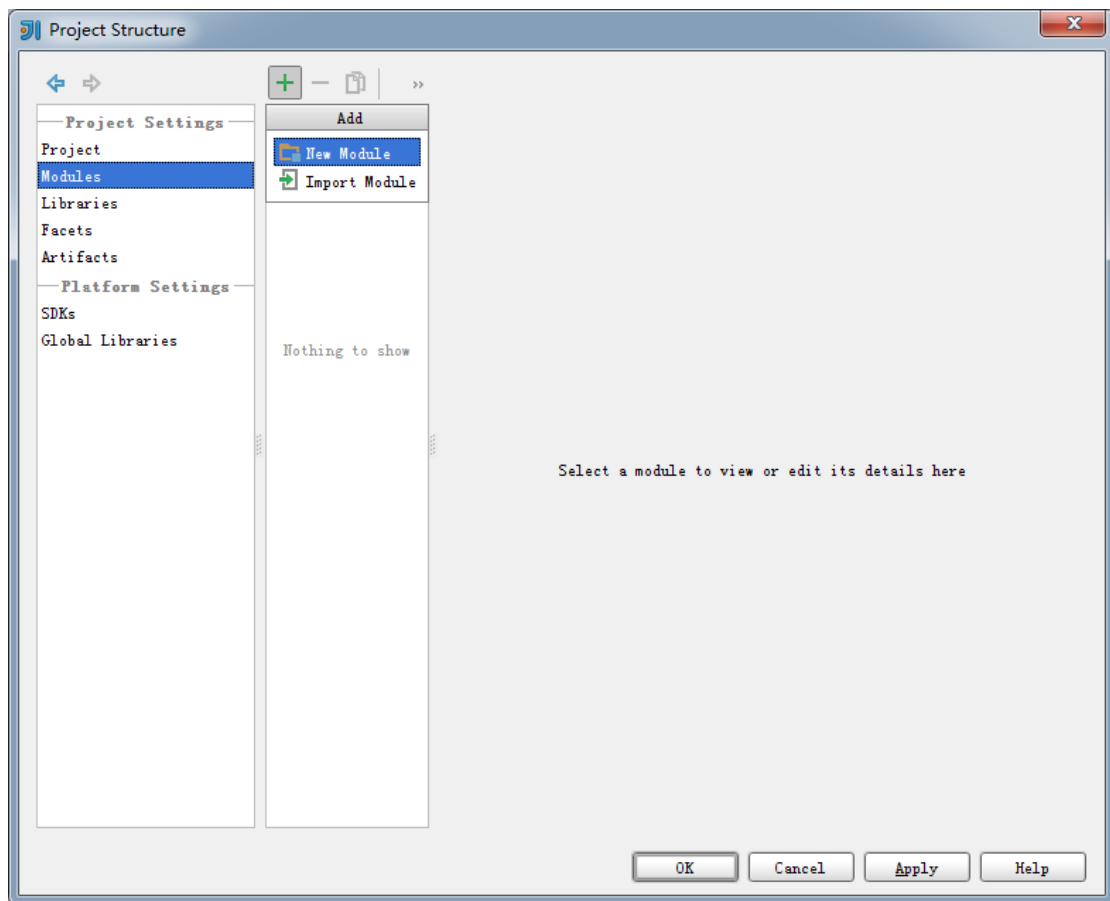
2、点击“Finish”完成空项目的创建，同时会打开空项目配置。



3、选择 Project，设置 Java SDK。

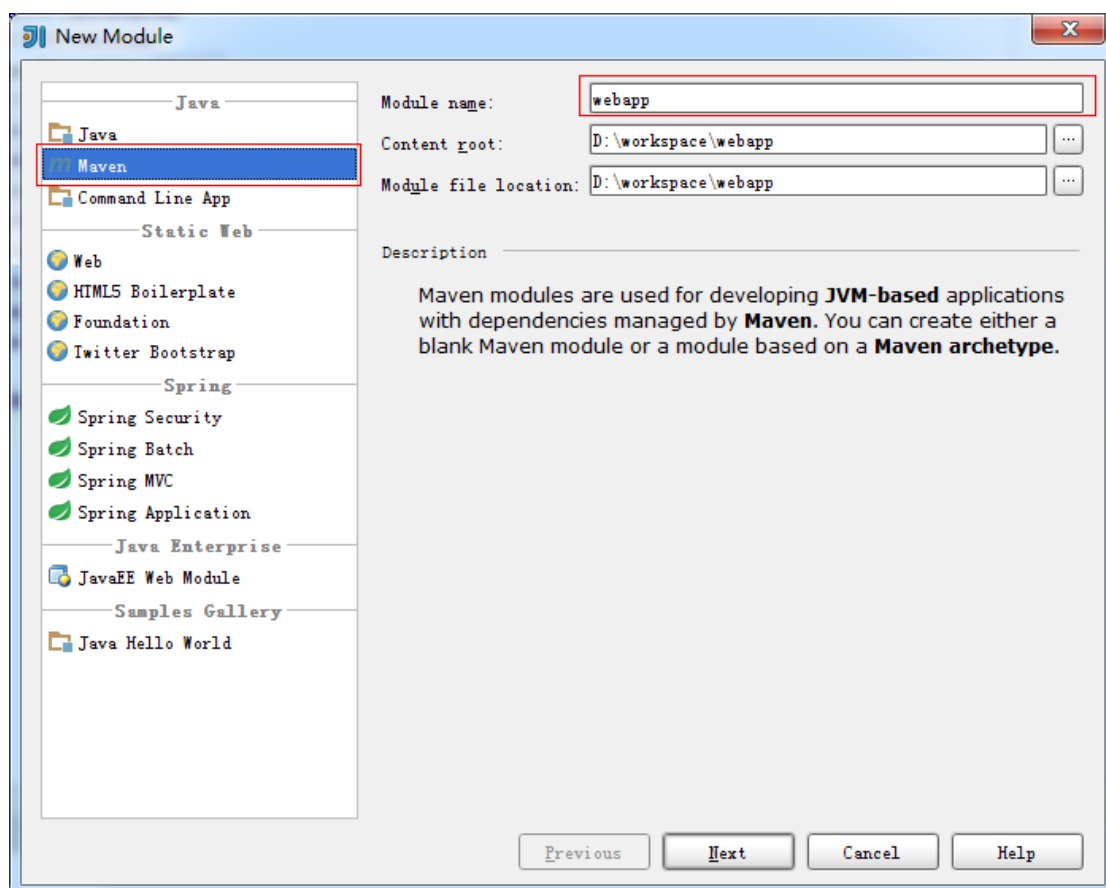


4、选择 Modules，可以通过“New Module”或者“Import Module”创建或导入项目。

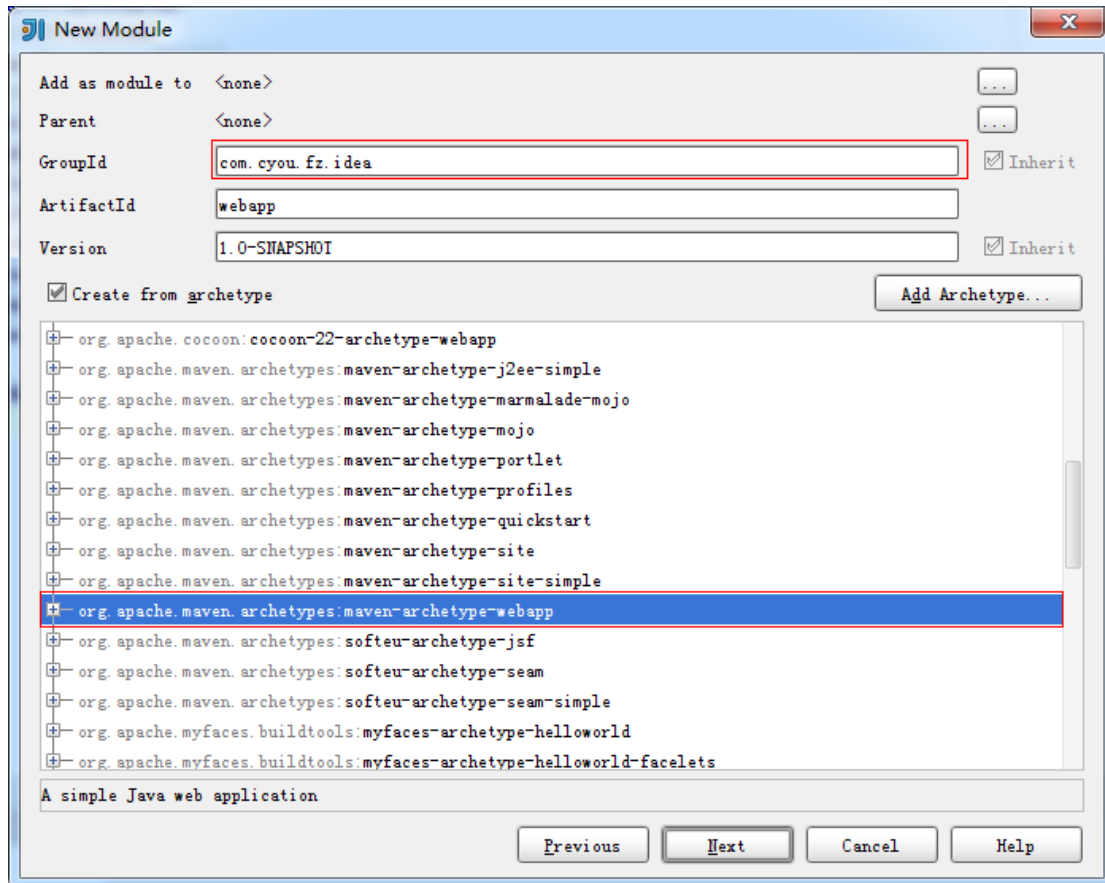


创建 Maven 项目

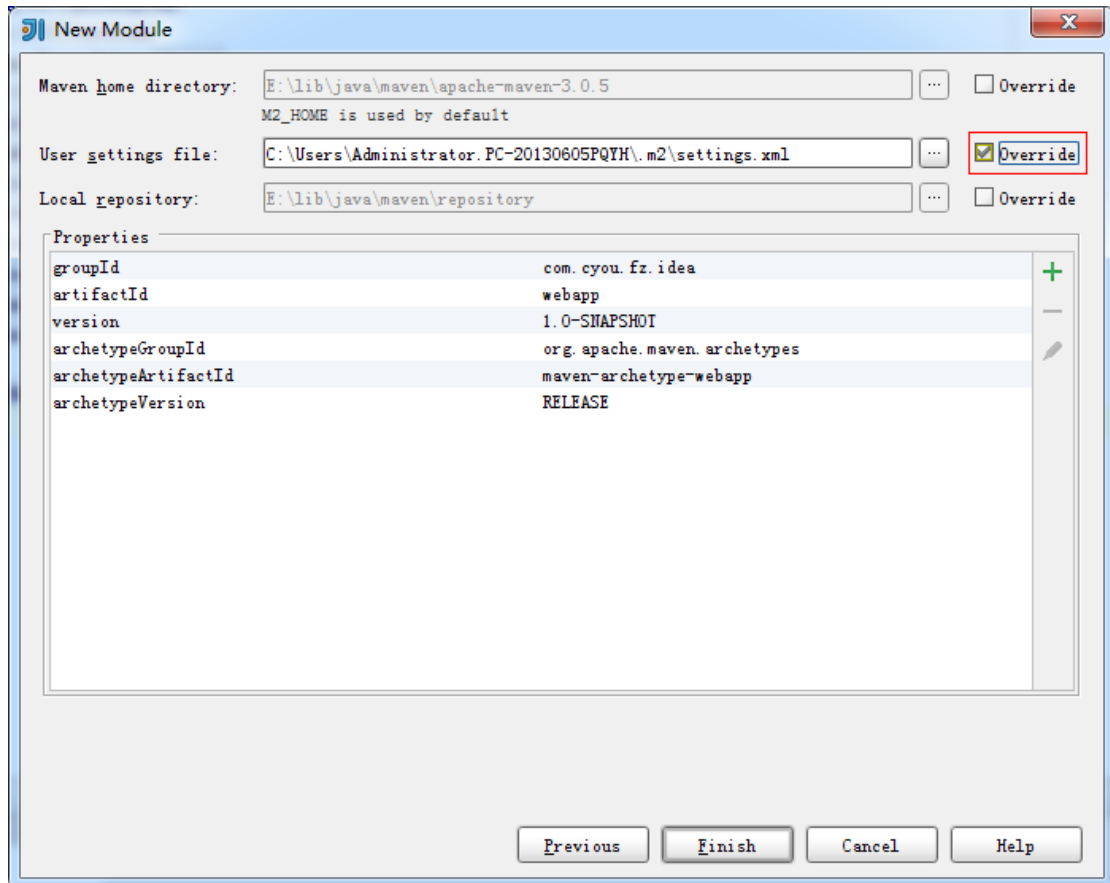
1、选择 File->New Module，选择 Maven，输入 Module Name。



2、点击“Next”，修改 groupId，如果是 web 项目，勾选“Create from archetype”，并选择“org.apache.maven.archetypes.maven-archetypes-webapp”。

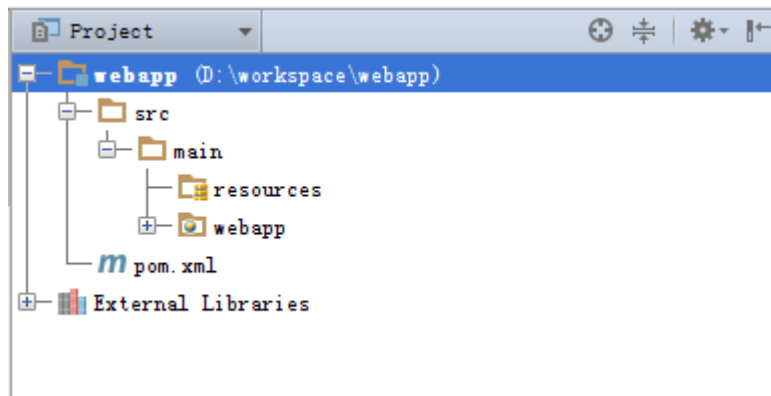


3、点击“Next”，检查项目配置信息和 Maven 配置文件是否正确。如果发现 Maven 的 settings.xml 路径错误，先勾选“Override”，选择正确的 settings.xml 路径。



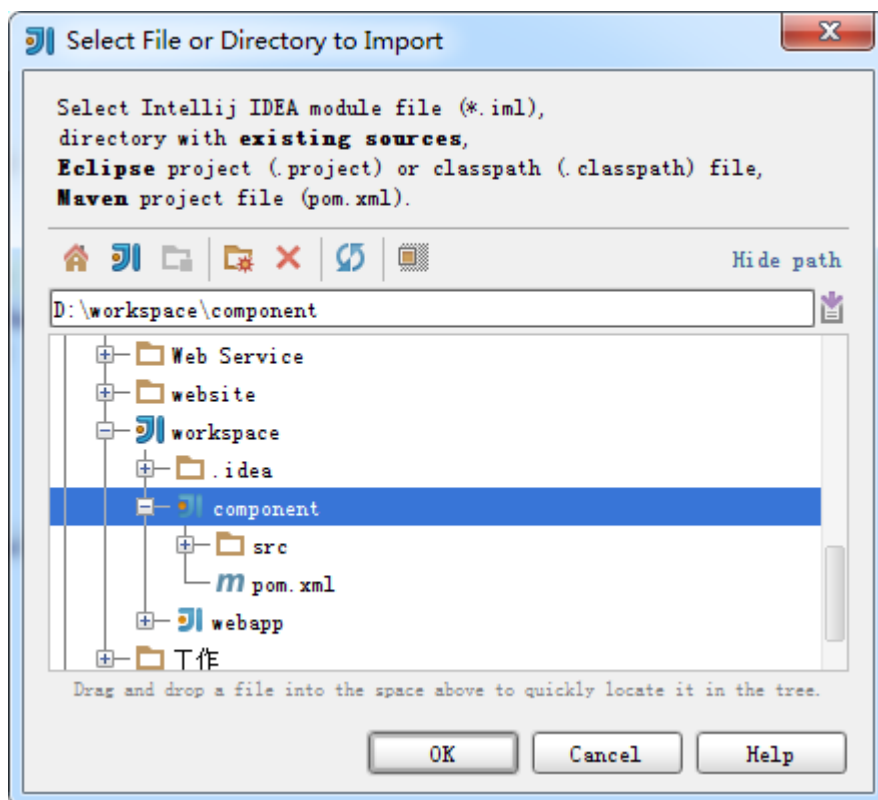
建议：把 Maven 的 settings.xml 复制一份到 C:\Users\{UserName}\.m2\目录下，这样就不用每次都修改 IDEA 的 Maven 配置了。

4、点击“Finish”完成空项目下 Maven 模块的创建。

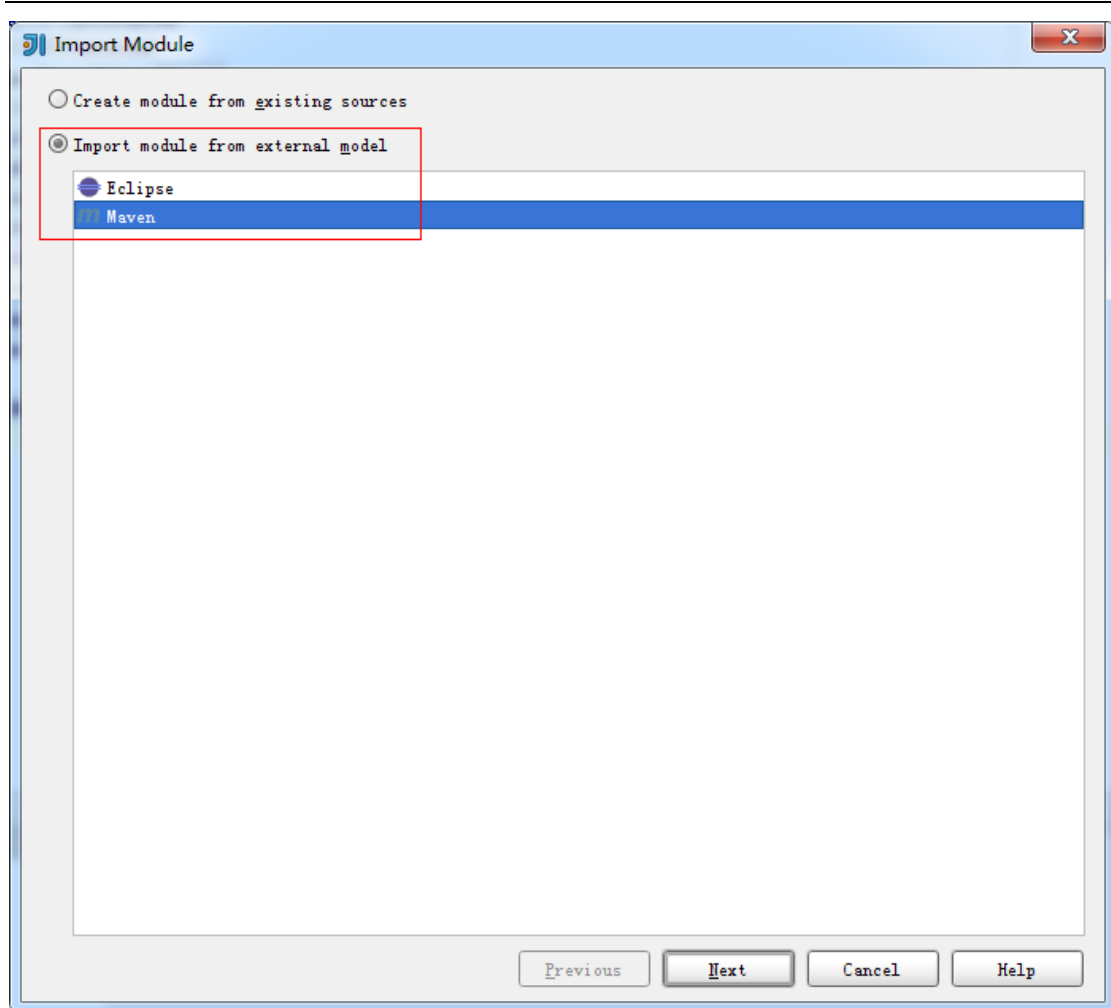


导入 Maven 项目

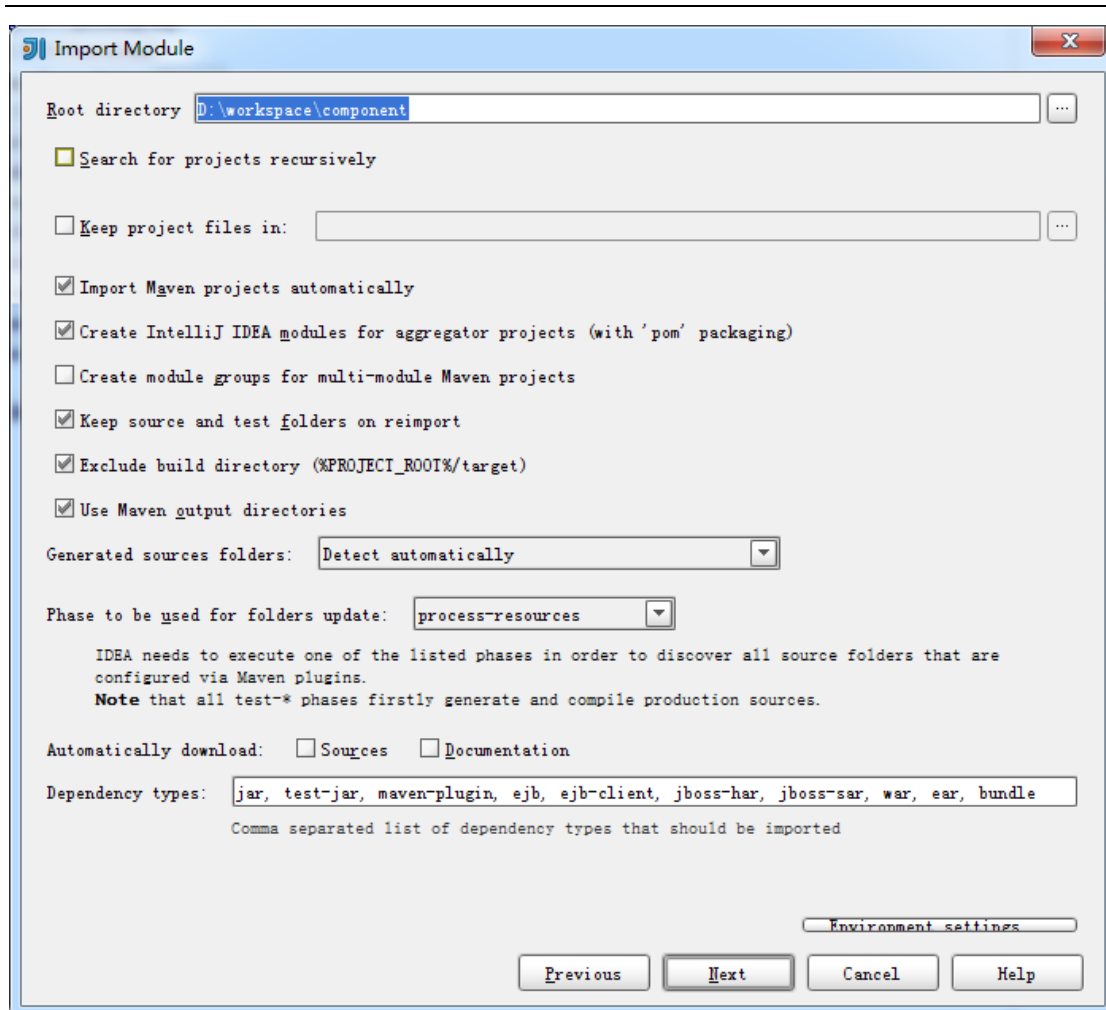
1、选择 File->Import Module，选择 Maven 模块路径。



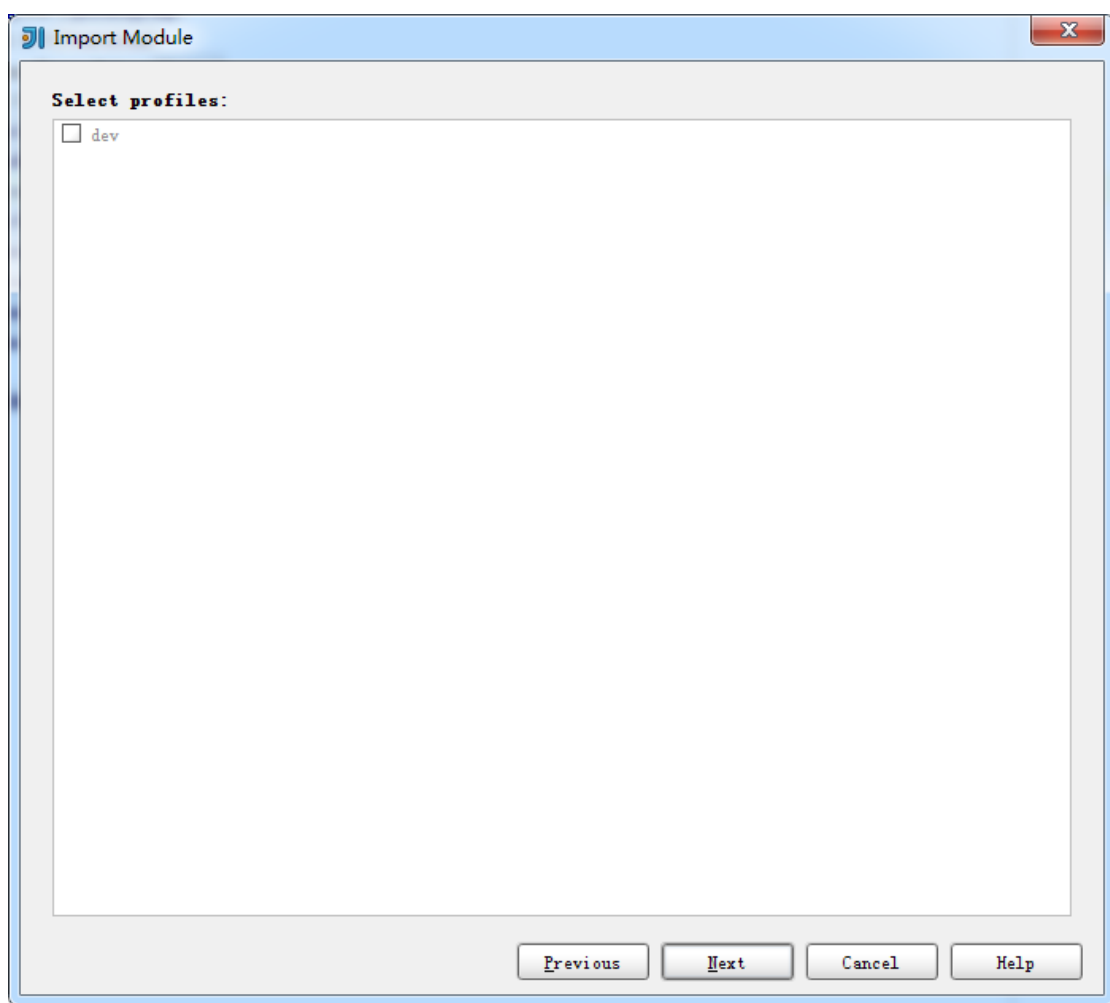
2、选择“Import module from external model”下的 Maven



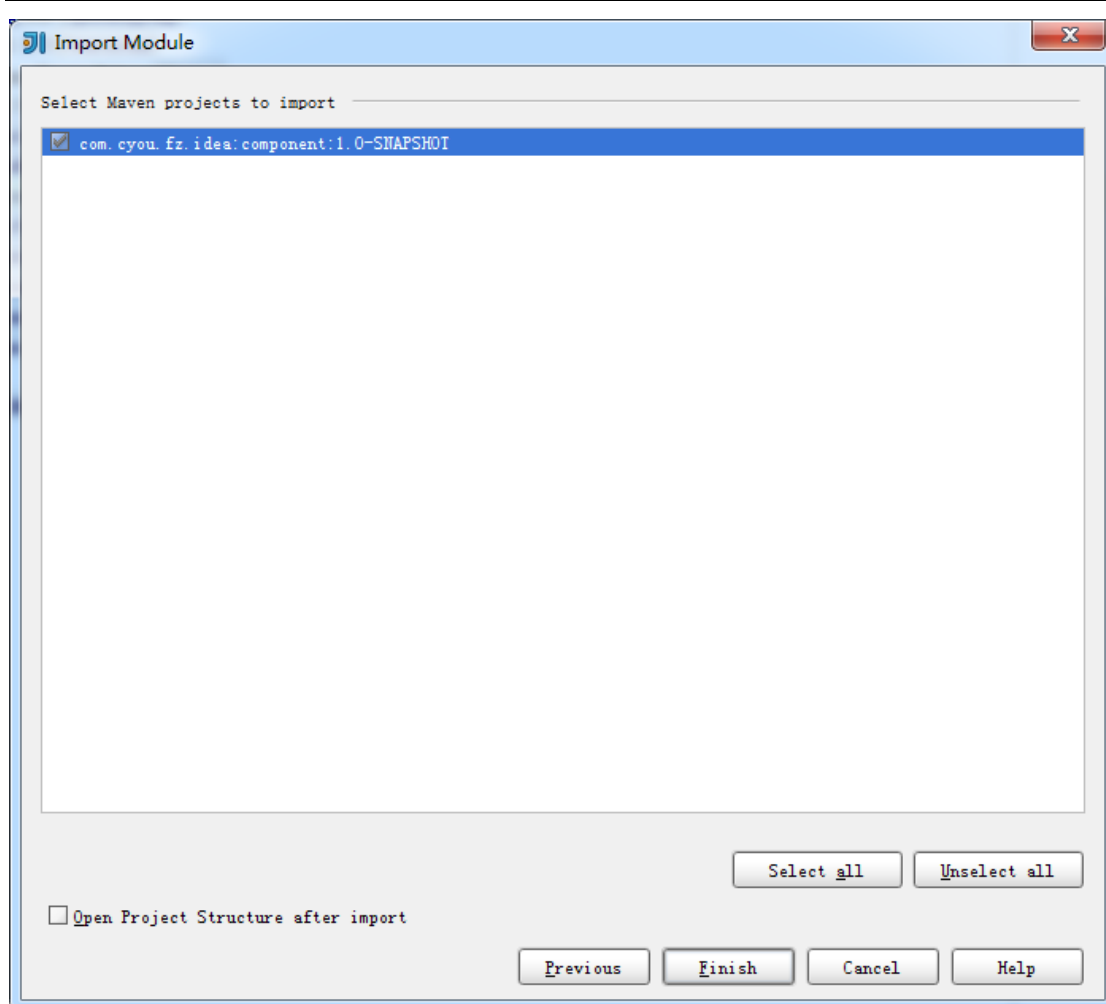
3、这一步保持默认即可



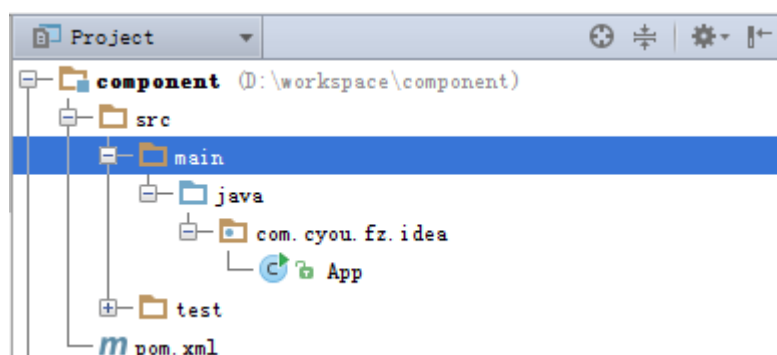
4、选择全局 profiles。如果使用 nexus 私服，配置好全局 profile，在这一步勾选 profiles。



5、确认下 groupId 和 artifactId。

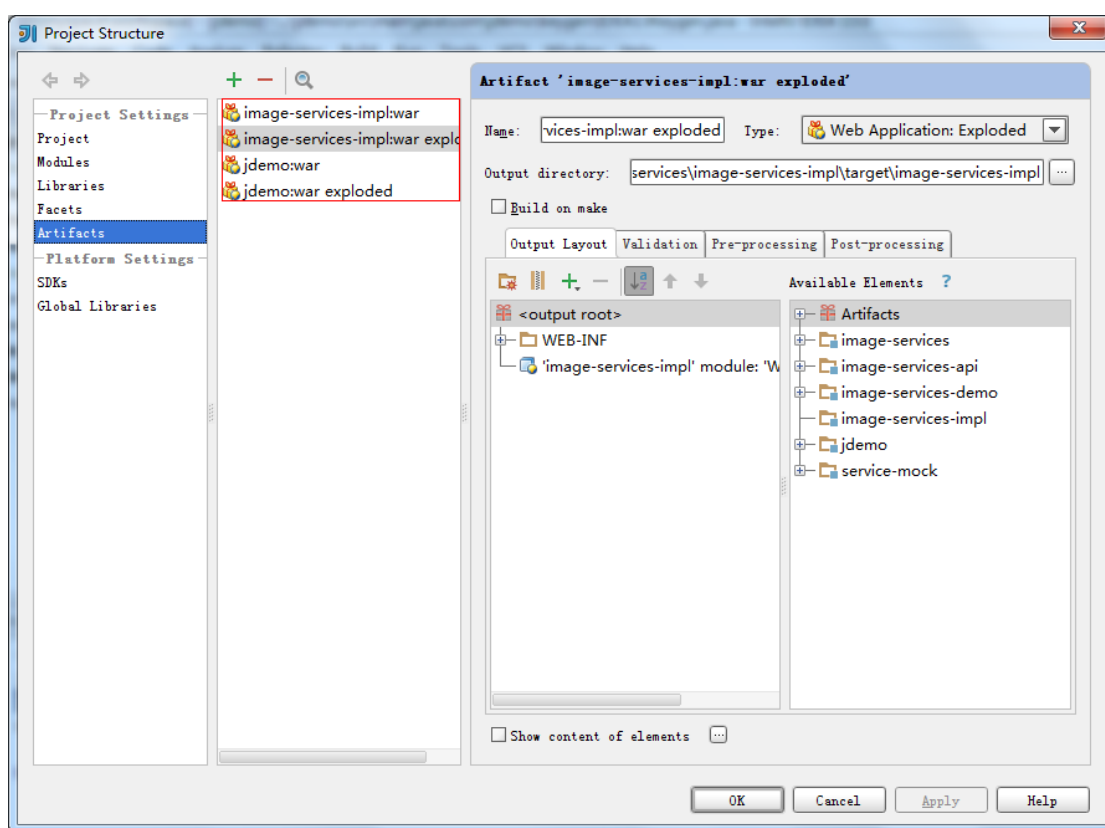


6、点击完成即可。

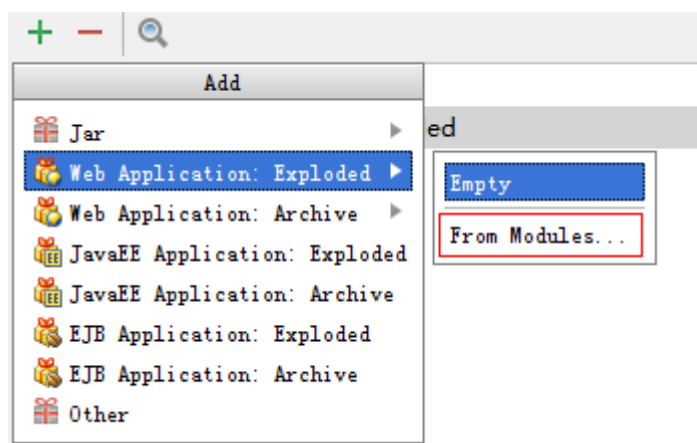


创建 Web 部署包

打开 File->Project Setting 或 Ctrl+Shift+Alt+S，打开 Artifacts 选项卡，会看到一些默认的部署包结构。



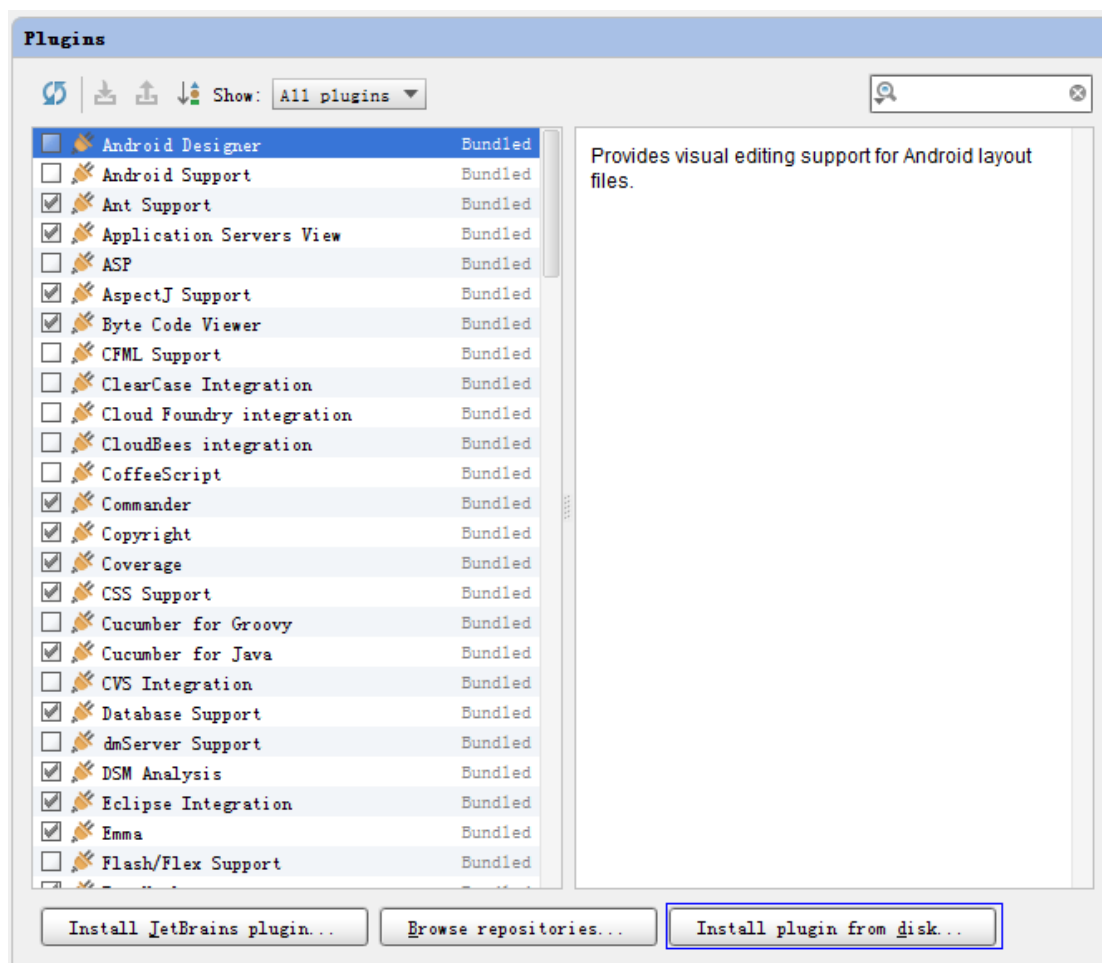
如果默认的部署包不符合要求，可以直接从 Maven 创建部署包。点击 $+$ ，选择“Web Application: Exploded” -> “From Maven”，便自动创建了 Web 部署包。



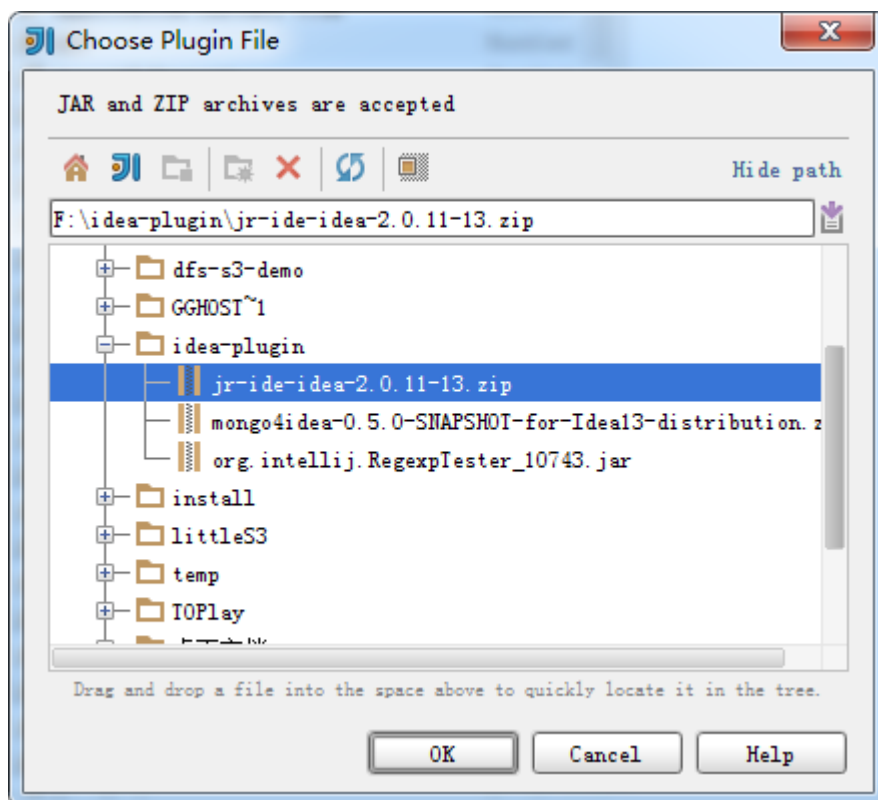
JRebel 热部署

使用 JRebel 可以解决 Java 热部署的问题，下面介绍 JRebel 插件的安装和使用。

- 1、安装 JRebel 插件，打开 Settings->Plugins，点击“Install plugin from disk”



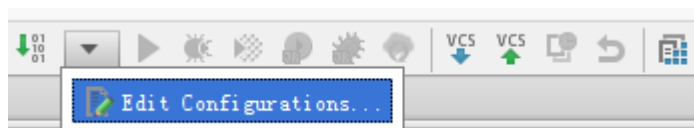
2、选择 jrebel 插件



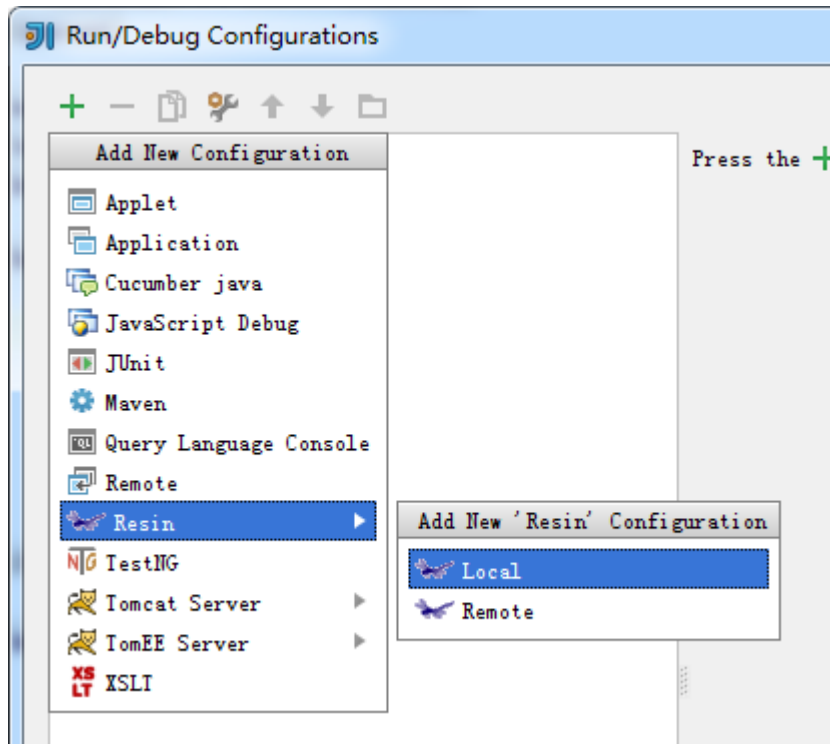
3、安装完，重启 IDEA。发现 Settings 下多了个 JRebel 选项，同时服务器管理面板多了“Run with JRebel ‘Resin’”以及“Debug with JRebel ‘Resin’”。



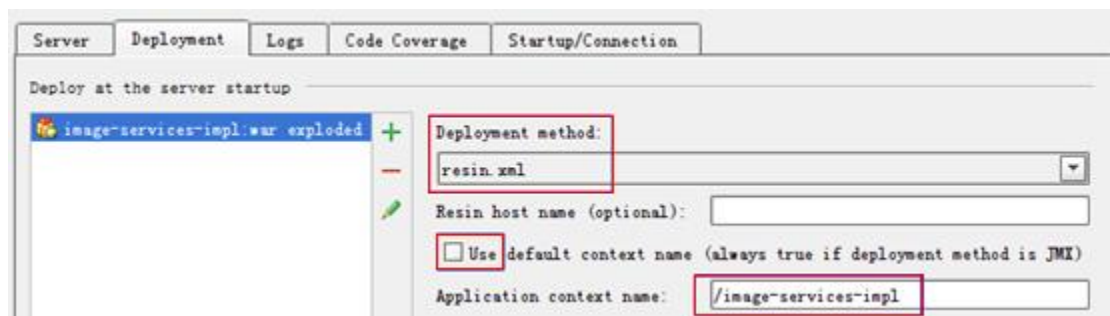
4、配置应用服务器，以 Resin 为例。点击“Edit Configurations”



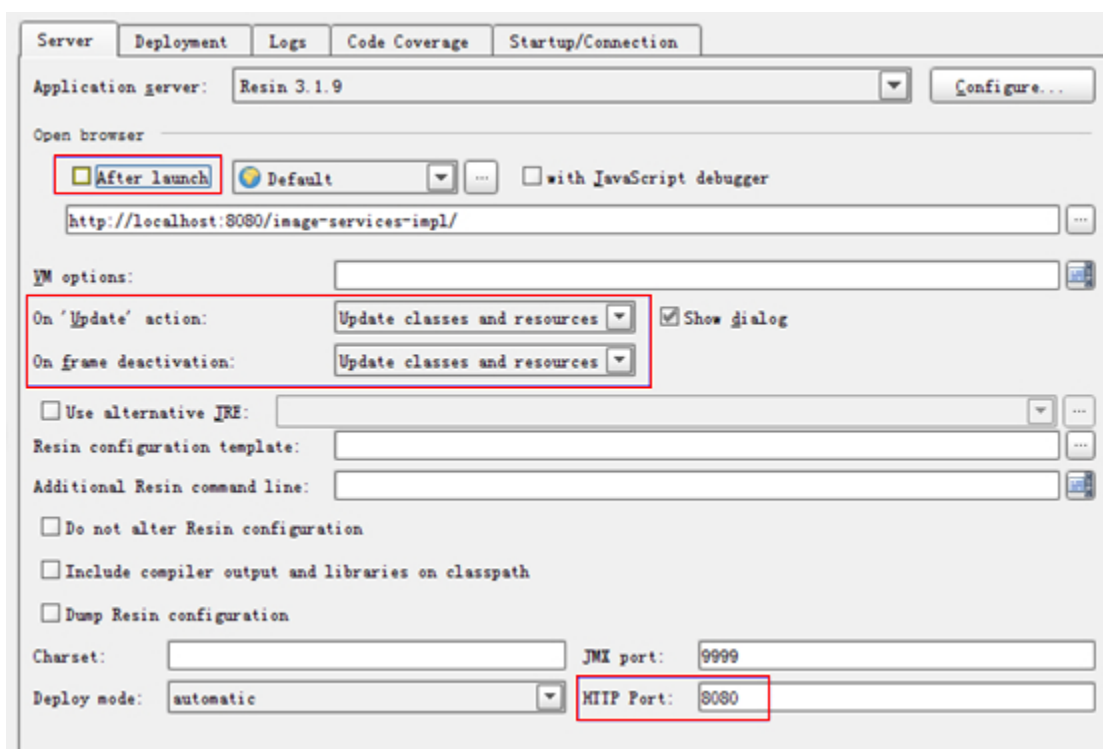
5、增加一个 Resin 服务器。



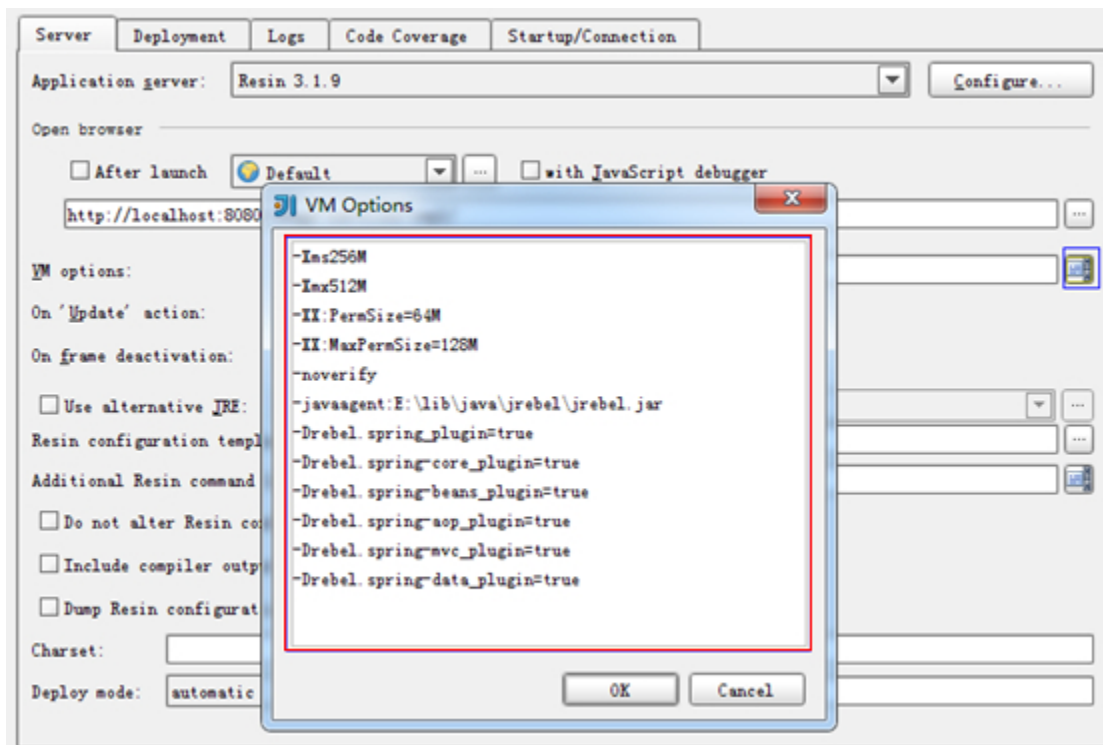
6、添加部署包，点击 $+$ ->Artifact，选择 exploded 包。设置 Deployment method 为 resin.xml。反选“Use default content name”，如果要设置 path，则填写 Application context name。



7、修改服务器配置。修改端口为 8080，关闭启动 Resin 后打开浏览器，修改“On ‘Update’ action”为“Update classes and resources”以及“On Frame deactivation”为“Update classes and resources”。



8、修改 VM options。



修改 VM Options 内容，以下是我个人配置，开启了 spring、spring-core、spring-beans、spring-mvc、spring-data 等插件。

```
-Xms256M
-Xmx512M
-XX:PermSize=64M
```

```

-XX:MaxPermSize=128M

-noverify

-javaagent:E:\lib\java\jrebel\jrebel.jar

-Drebel.spring_plugin=true

-Drebel.spring-core_plugin=true

-Drebel.spring-beans_plugin=true

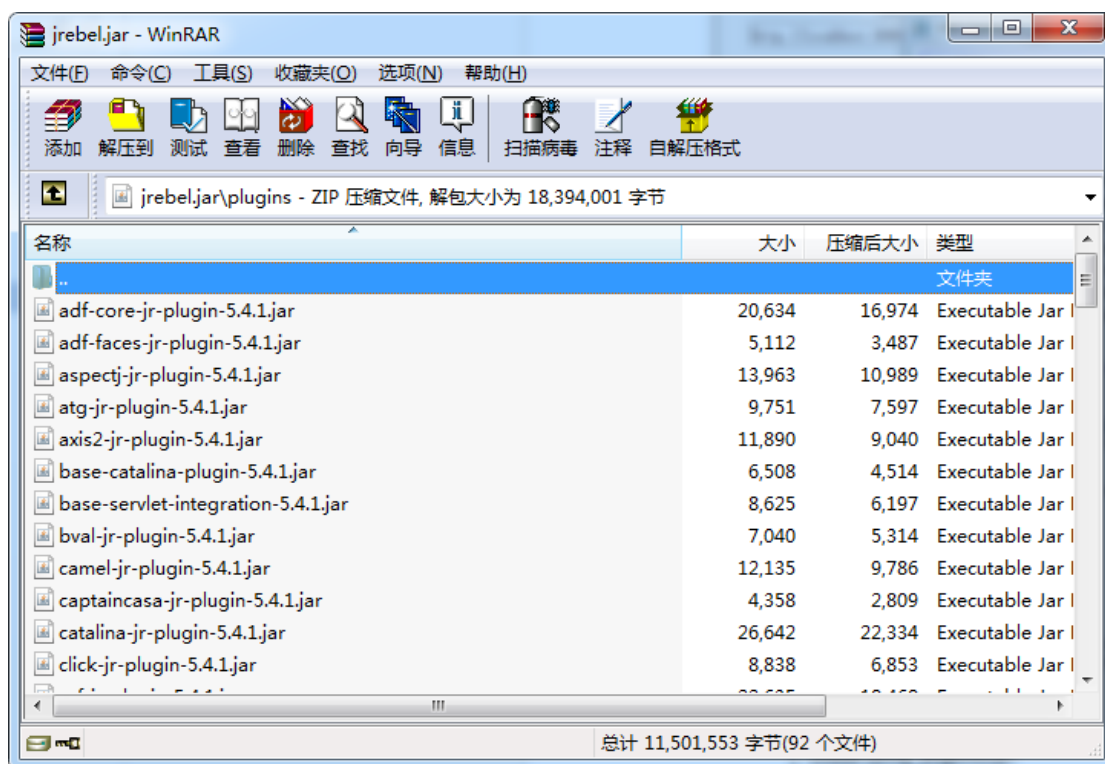
-Drebel.spring-aop_plugin=true

-Drebel.spring-mvc_plugin=true

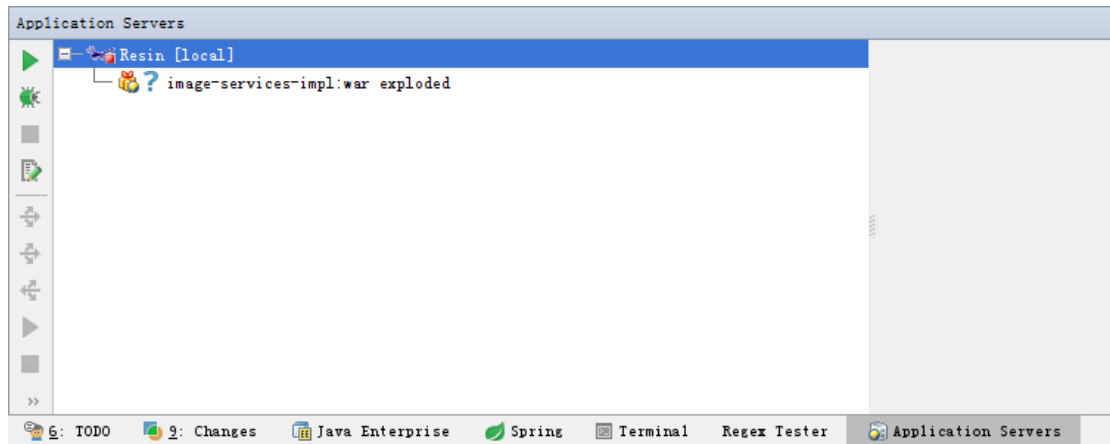
-Drebel.spring-data_plugin=true


```

如果想启用 JRebel 的其他插件，用 WinRAR 打开 jrebel.jar，进入 plugins 目录列出了所有。启用 xxx{-yyy}-jr-plugin-{version}.jar 插件，配置为-Drebel.xxx{-yyy}_plugin=true。



9、点击“OK”，Application Servers 窗口显示 Resin 选项。



10、点击, 启动 JRebel Debug 模式



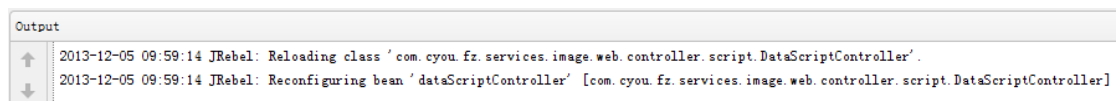
11、看到启动参数中包含如下内容，即表示热部署模式启动成功。

```

2013-12-05 09:53:12 JRebel:
2013-12-05 09:53:12 JRebel: #####
2013-12-05 09:53:12 JRebel:
2013-12-05 09:53:12 JRebel:  JRebel 5.4.1 (201310171404)
2013-12-05 09:53:12 JRebel:  (c) Copyright ZeroTurnaround OU, Estonia, Tartu.
2013-12-05 09:53:12 JRebel: |
2013-12-05 09:53:12 JRebel:  Over the last 1 days JRebel prevented
2013-12-05 09:53:12 JRebel:  at least 6 redeloys/restarts saving you about 0.2 hours.
2013-12-05 09:53:12 JRebel:
2013-12-05 09:53:12 JRebel:  This product is licensed to zhangthe9
2013-12-05 09:53:12 JRebel:  for unlimited number of developer seats on site.
2013-12-05 09:53:12 JRebel:
2013-12-05 09:53:12 JRebel:  The following plugins are disabled at the moment:
2013-12-05 09:53:12 JRebel:  * Axis2 plugin (set -Drebel.axis2_plugin=true to enable)
2013-12-05 09:53:12 JRebel:  * Camel plugin (set -Drebel.camel_plugin=true to enable)
2013-12-05 09:53:12 JRebel:  * Click plugin (set -Drebel.click_plugin=true to enable)
2013-12-05 09:53:12 JRebel:  * Deltaspike plugin (set -Drebel.deltaspike_plugin=true to enable)
2013-12-05 09:53:12 JRebel:  * Eclipse RCP Plugin (set -Drebel.eclipse_plugin=true to enable)

```

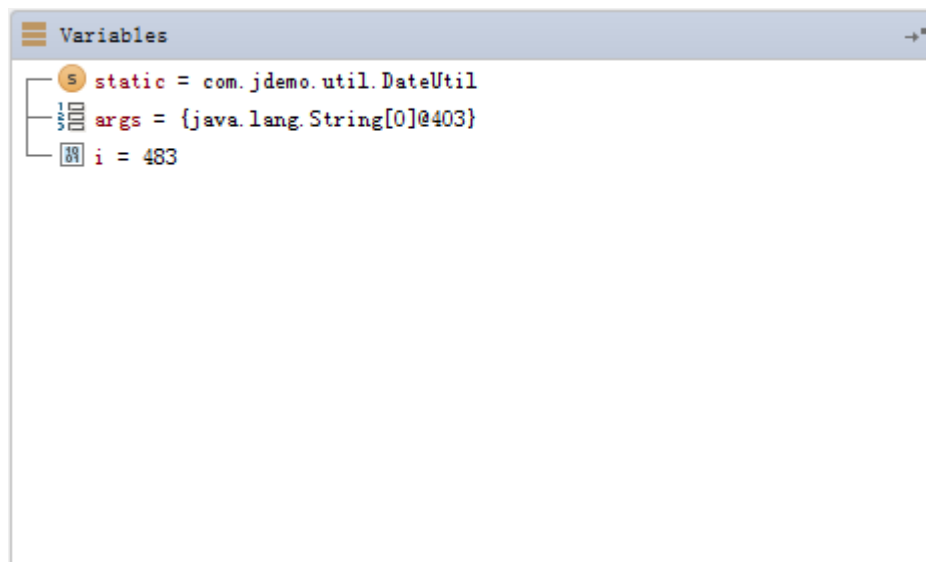
12、修改类，使 IDEA 失去焦点（可以切换到浏览器，或者点击下 Windows 任务栏等任何操作），IDEA 增量 Make 一次，JRebel 重新载入被改变的类，Console 显示如下信息表示 JRebel 加载成功过。



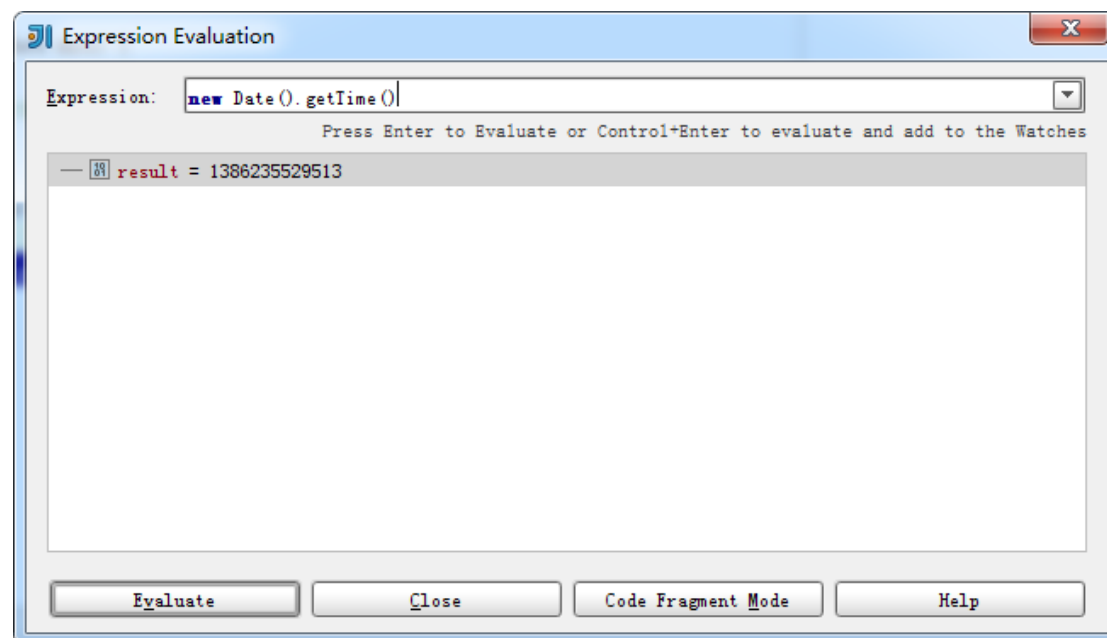
常用技巧

Debug 跟踪条件变量

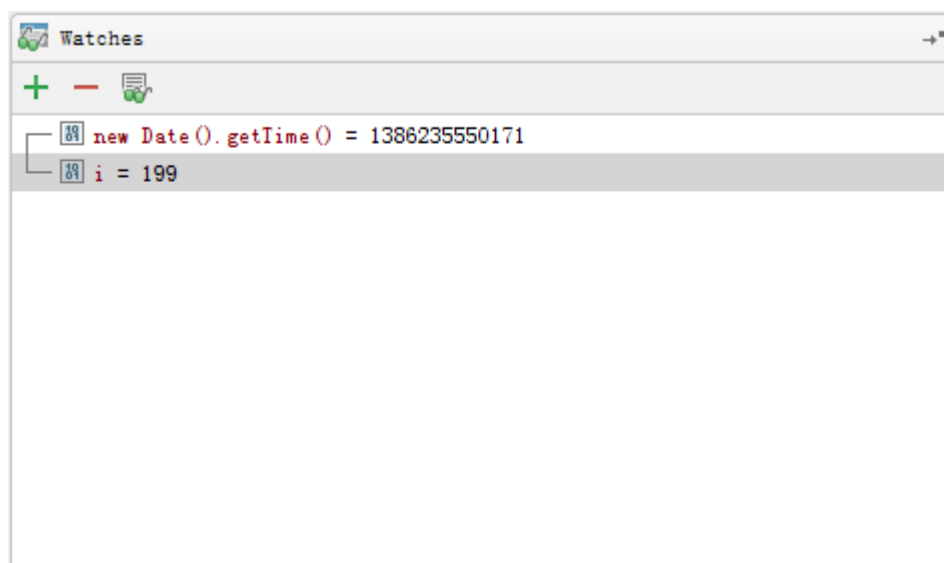
通过变量面板查看变量在断点的当前值。




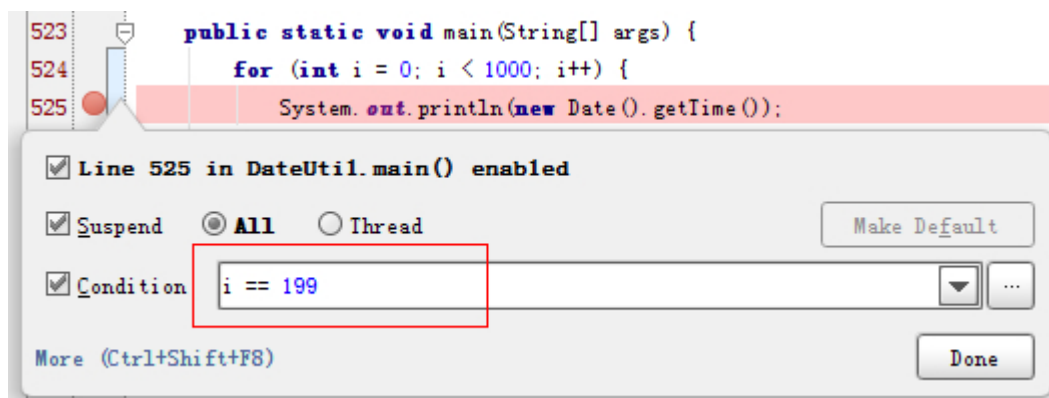
通过 Alt+F8 查看变量在断点的当前值。



通过 Watches 面板查看变量在断点的当前值



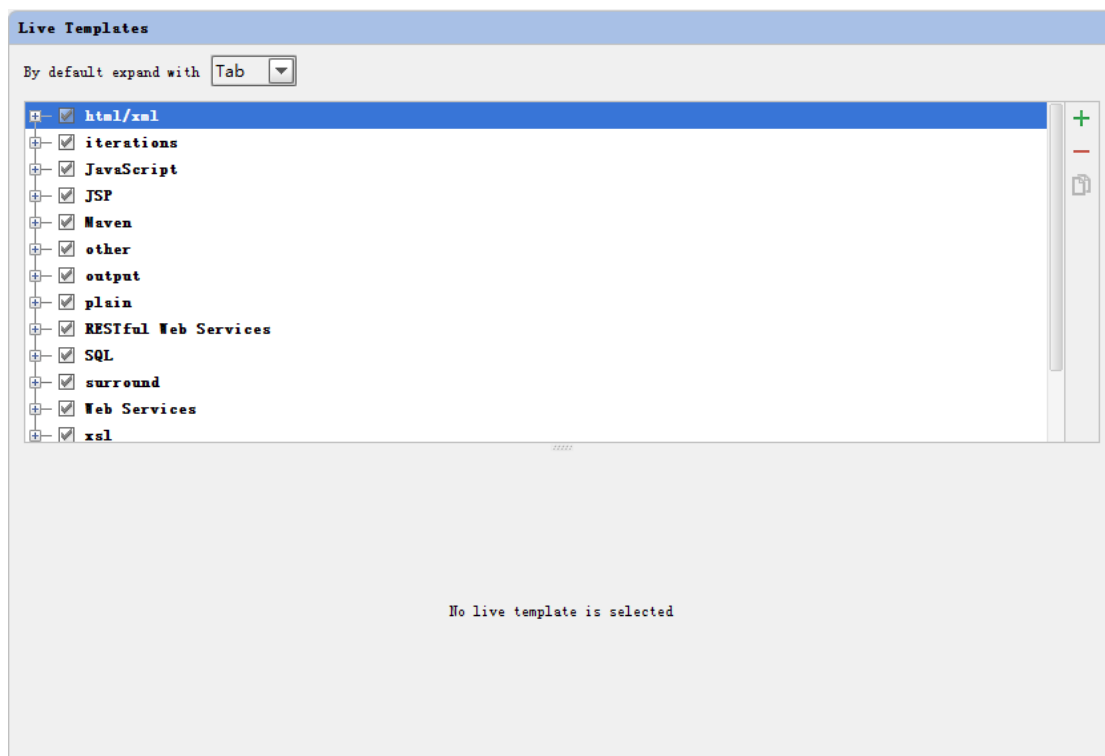
右键单击, 弹出断点设置条件, 重新 Debug, 进入该断点



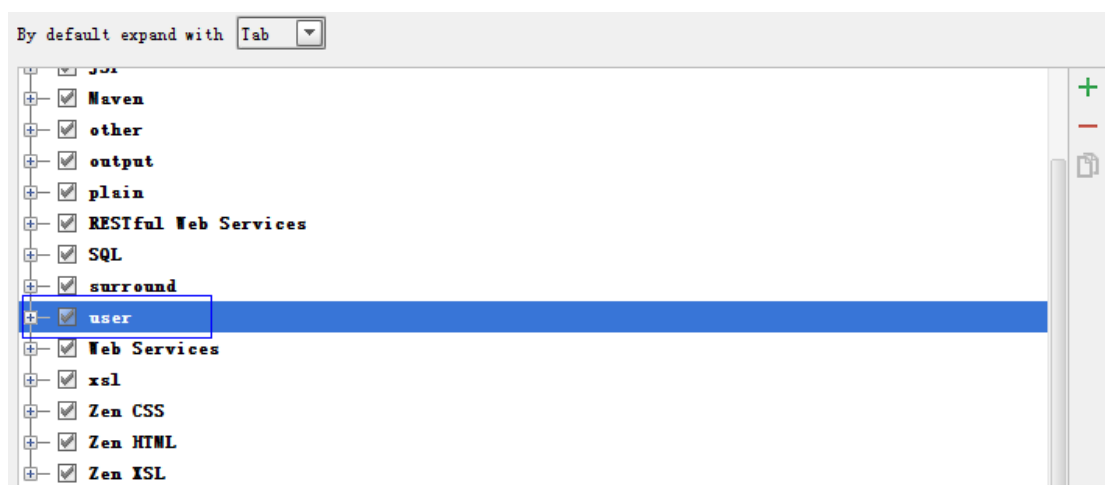
Live Template 使用

Live Template 用于创建代码模板, 使用 live template 可以快速生成代码, IDEA 自带了一些默认的模板, 比如 psvm、sout 等。下面介绍下如何创建适合自己的模板。

打开 Settings-Live Template, 显示默认的 Live Template 配置, 了解下这些默认的 template 对开发十分有帮助。



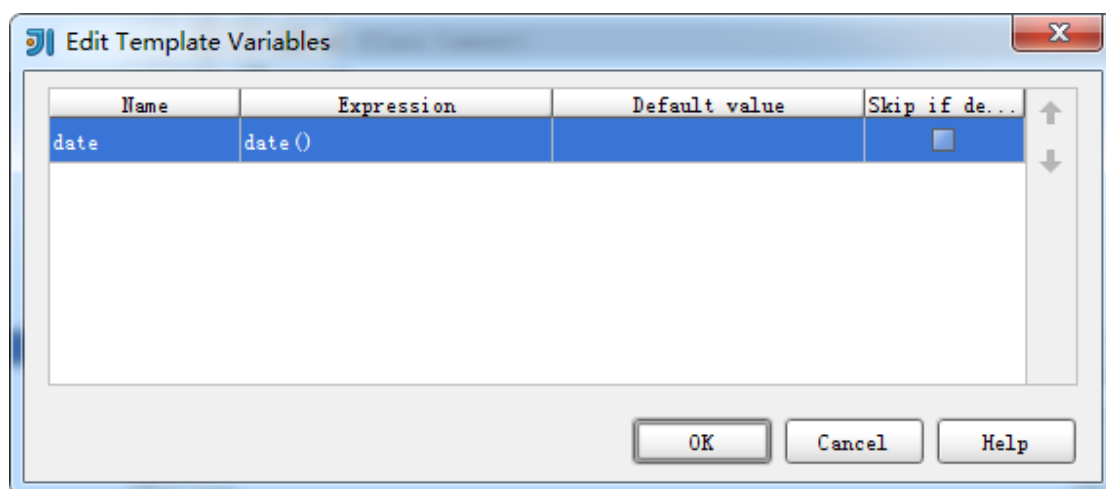
点击 $+$ ，选择 Template Group，创建名为 user 的组。



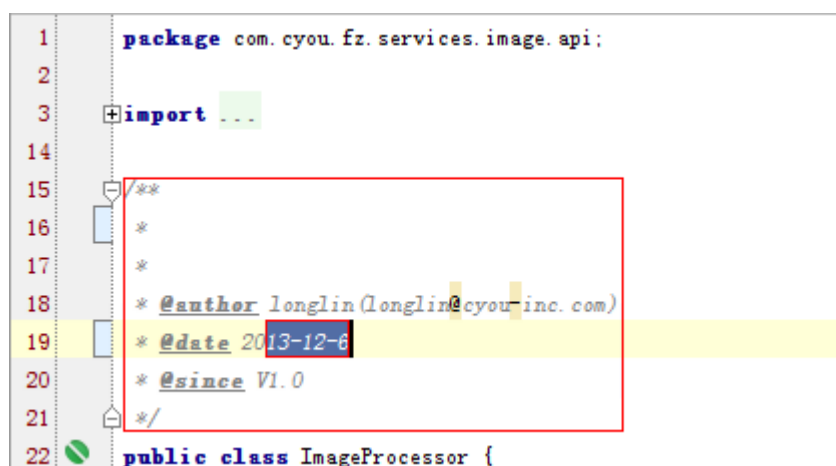
点击 $+$ ，选择 Live Template，创建类注释模板，快捷键为“cc”，描述为“Class Comment”，Applicable 设置为“Java: declaration”。



点击“Edit variables”，设置变量。

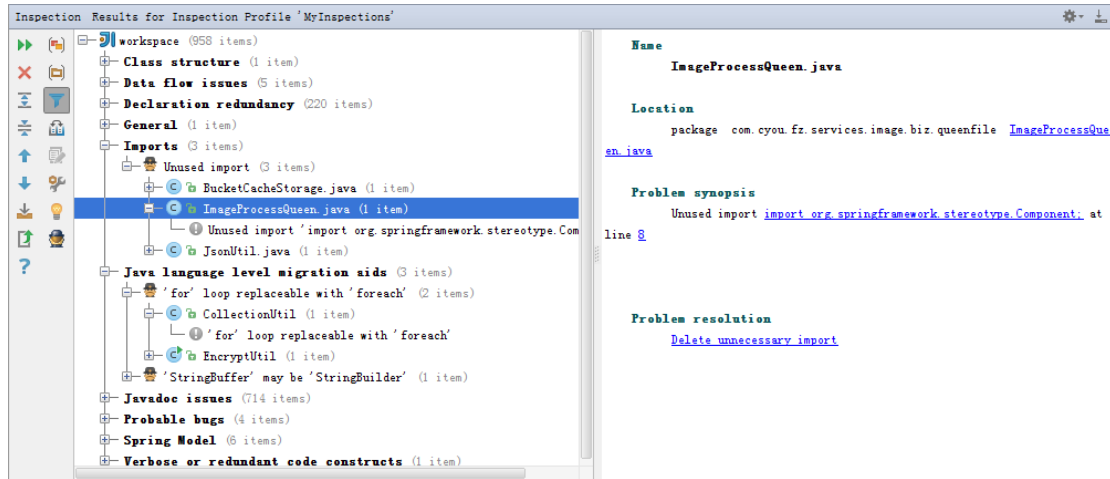


在类头部，按“cc”，再按 Tab 键，就生成了注释代码。

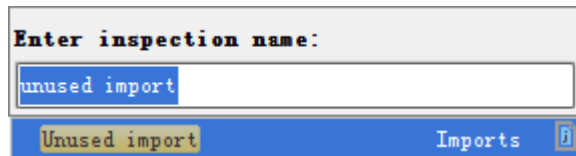


代码分析

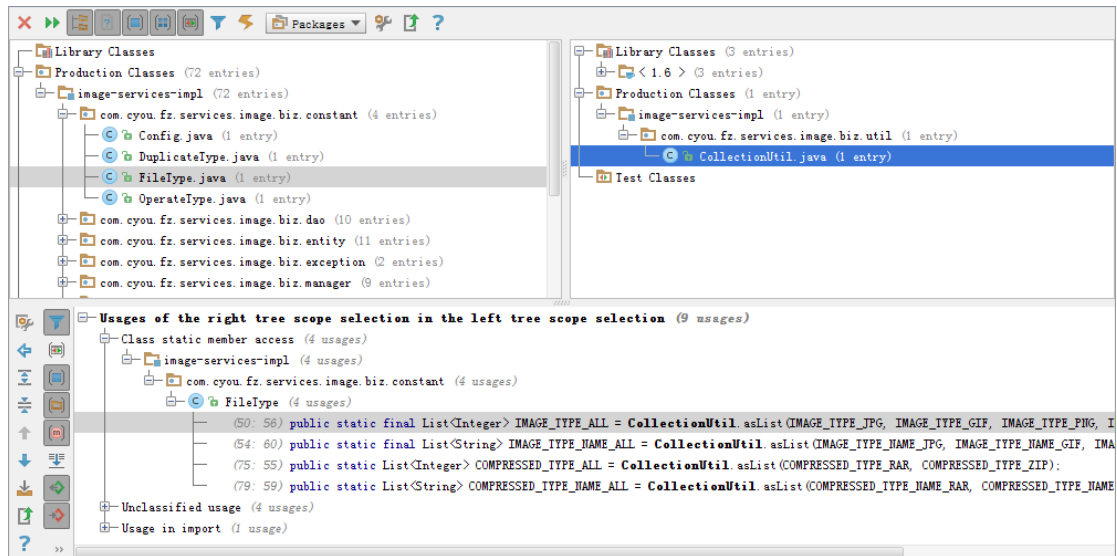
代码规范检查，打开 Analyze -> Inpect Code，检查代码是否符合 Settings -> Inspections 的设置。



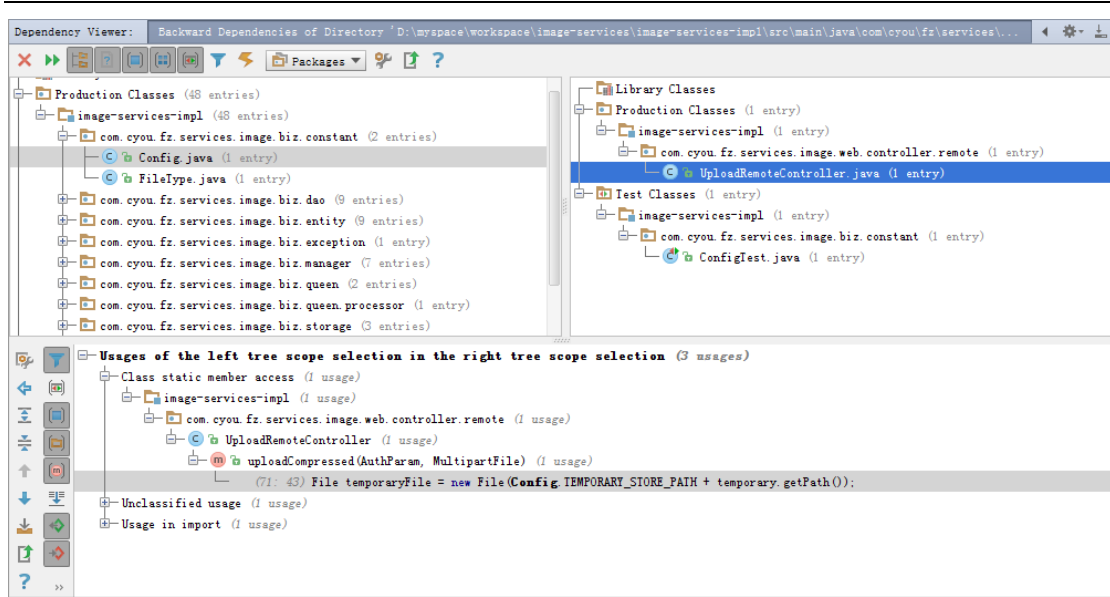
检查指定的 Inspections 项。打开 Analyze -> Run Inspection By Name, 输入要检查的项。



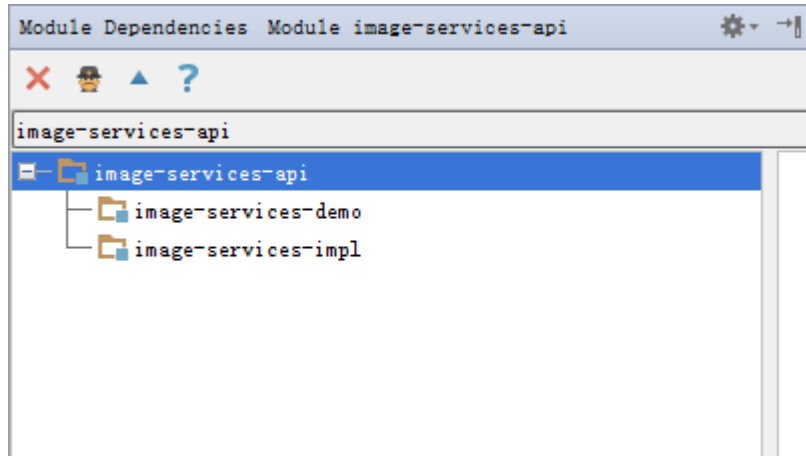
分析代码依赖。打开 Analyze -> Analyze Dependencies, 查看依赖结果。



分析代码反向依赖。打开 Analyze -> Analyze Backward Dependencies, 查看反向依赖结果。



分析模块依赖。打开 Analyze -> Analyze Module Dependencies, 查看模块依赖。



分析 DSM 分层依赖。打开 Analyze -> Analyze Dependency Matrix, 查看依赖结构矩阵。

DSM workspace

Package	manager	dao	queen	Mock53FileStorage	MockSearchStorage	*	BizCheckedException	GraphicsMagickException	task	Config	DuplicateType	FileType	OperateType	Bucket	Extension	Image	Duplicate	ExtensionValue	OperateLog	SecretKey	Temporary	Version	Watermark	BaseImage	support	util
manager	-	-	27	6	16																					
dao	32	37	-																							
queen																										
Mock53FileStorage																										
MockSearchStorage																										
*	63	41	2	2																						
BizCheckedException								4																		
GraphicsMagickException																										
task																										
Config	5	1	3							3																11
DuplicateType																										
FileType																										
OperateType																										
Bucket	32	38		10	36																					
Extension	36	6		4	11																					
Image	75	52	51	11	19																					19
Duplicate	52	48	65	4	14																					8
ExtensionValue		6		4																						
OperateLog		4																								
SecretKey	20	5																								
Temporary	4	2																								
Version	20	35	14																							30
Watermark	33	34	16																							8
BaseImage	9	41	36																							
support	52	44	6	2	1																					9
util	50	36	30	3	5	7																				-

分析循环依赖。打开 Analyze -> Analyze Cyclic Dependencies, 查看循环依赖关系。

Dependency Viewer: Cyclic Dependencies of Module 'image-services-impl'

Library Classes

- Production Classes
 - com.cyou.fz.services.image.biz.constant
 - com.cyou.fz.services.image.biz.dao
 - com.cyou.fz.services.image.biz.entity
 - com.cyou.fz.services.image.biz.queen
 - com.cyou.fz.services.image.biz.storage
 - com.cyou.fz.services.image.biz.support
 - com.cyou.fz.services.image.biz.util
- Test Classes

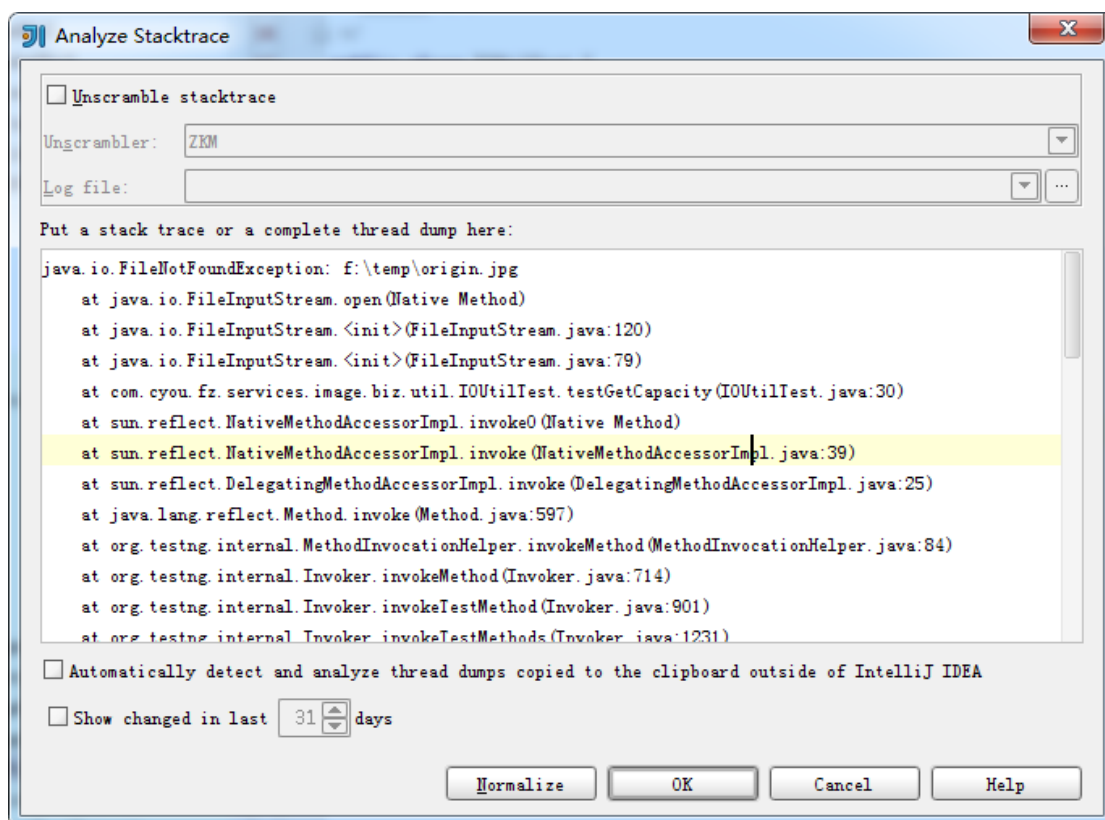
cycle

- com.cyou.fz.services.image.biz.dao
 - ImageProcessQueen.java
 - ImageProcessReceiver.java

Usage of package 'com.cyou.fz.services.image.biz.dao' in package 'com.cyou.fz.services.image.biz.queen' (12 usages)

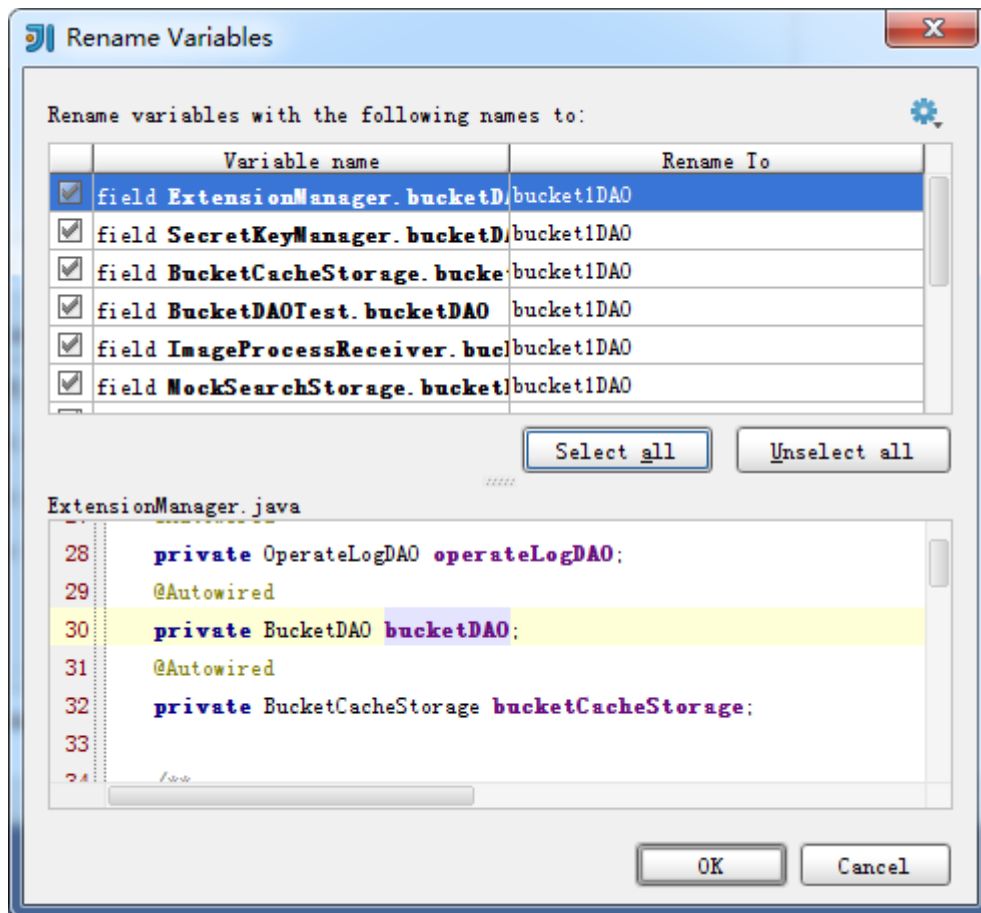
- Field declaration (3 usages)
 - image-services-impl (3 usages)
 - com.cyou.fz.services.image.biz.queen (3 usages)
 - ImageProcessReceiver (3 usages)
 - (43: 13) private BucketDAO bucketDAO;
 - (45: 13) private ImageDAO imageDAO;
 - (47: 13) private DuplicateDAO duplicateDAO;
- Unclassified usage (6 usages)
- Usage in import (3 usages)

分析堆栈跟踪信息。打开 Analyze -> Analyze Stacktrace, 输入堆栈信息。

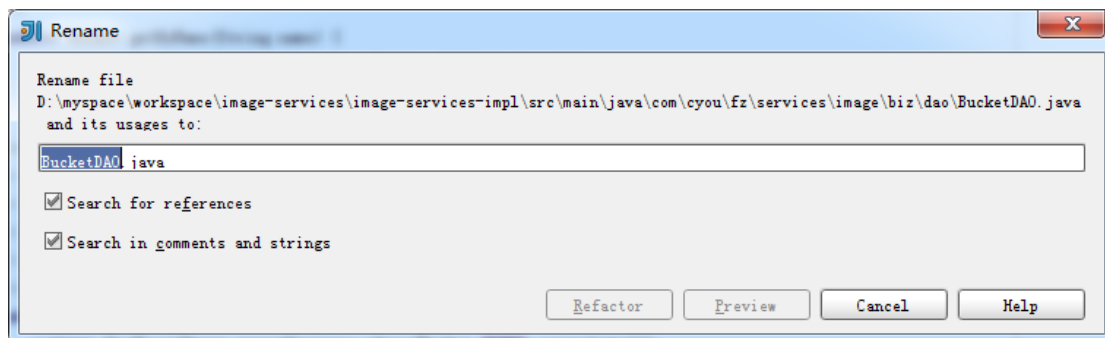


代码重构

重命名类，打开 Refactor -> Rename 或 Shift+F6，输入新类名，如果需要修改变量名，则勾选上要修改的代码。



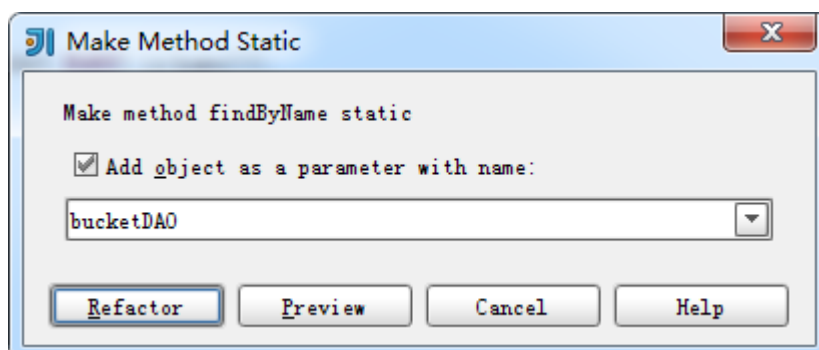
重命名文件，打开 Refactor -> Rename File，输入新文件名。



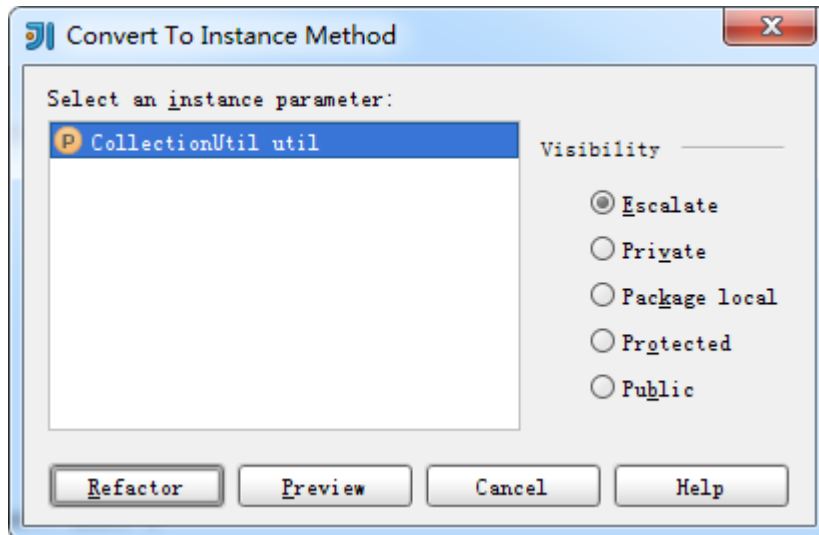
修改方法签名(参数、方法名、返回值等)，打开 Refactor -> Change Signature。



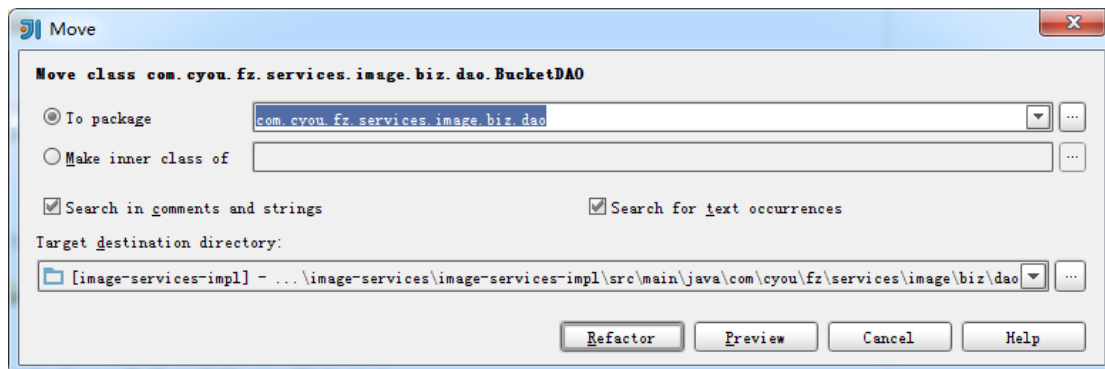
实例方法变成静态方法，打开 Refactor -> Make Static。



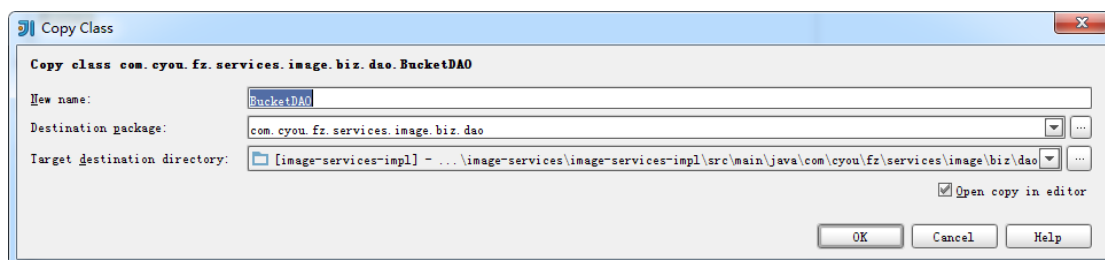
静态方法转为实例方法，打开 Refactor -> Convert To Instance Method，将参数中包含当前类对象的静态方法转为实例方法。



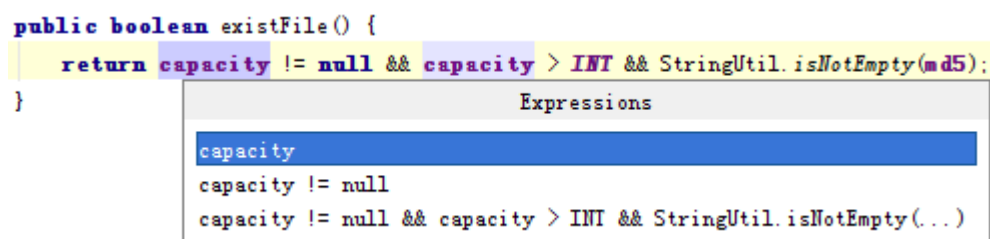
移动类，打开 Refactor -> Move 或 F6，指定目的包或作为内部类。



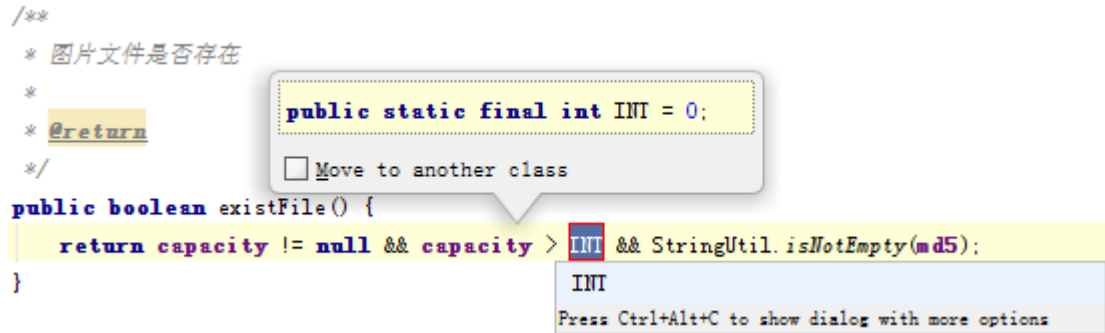
复制类，打开 Refactor -> Copy 或 F5，指定目的包。



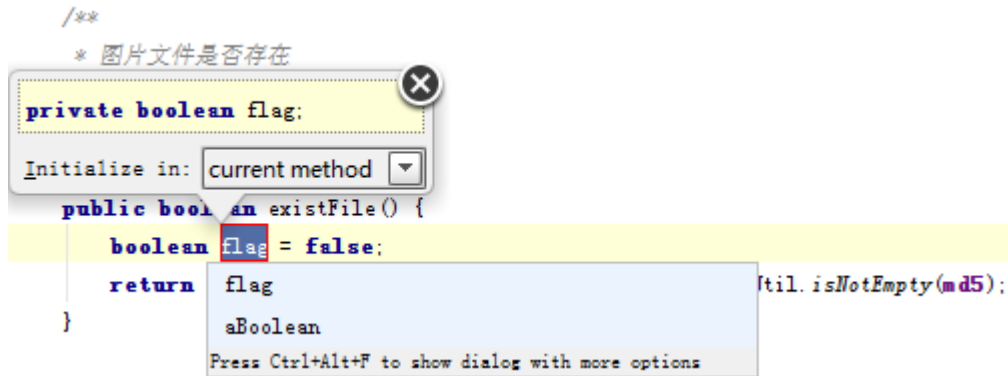
方法或属性转为变量。打开 Refactor -> Extract -> Variable 或 Ctrl+Alt+V，将对象属性或方法调用转本地变量。



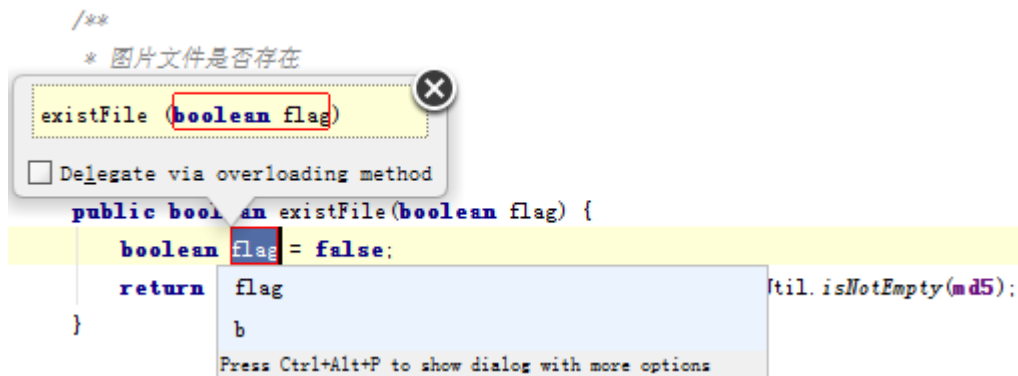
数值转为常量。打开 Refactor -> Extract -> Constants 或 Ctrl+Alt+C，输入常量值，回车确认，数值便转为常量。



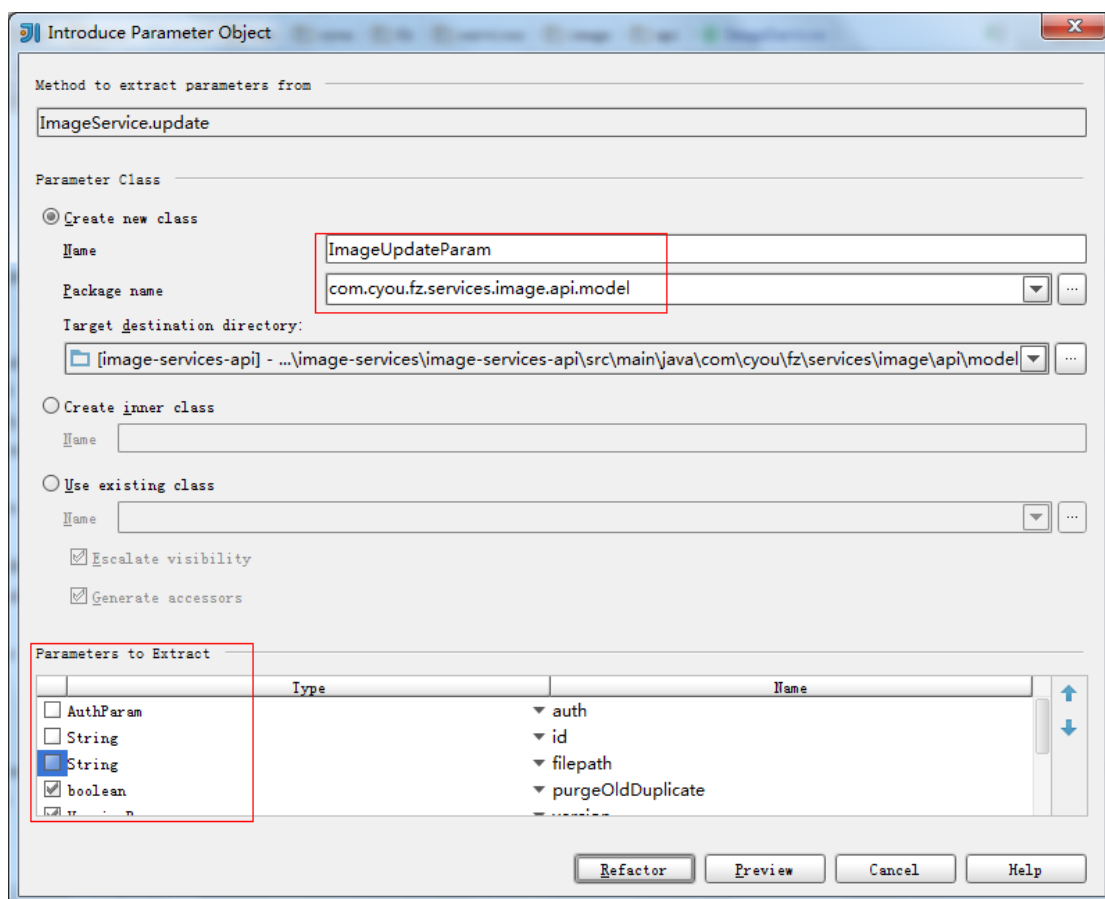
变量转为对象属性。执行 Refactor -> Extract -> Field 或 Ctrl+Alt+F，输入属性名，回车确认，变量便转为为对象属性。



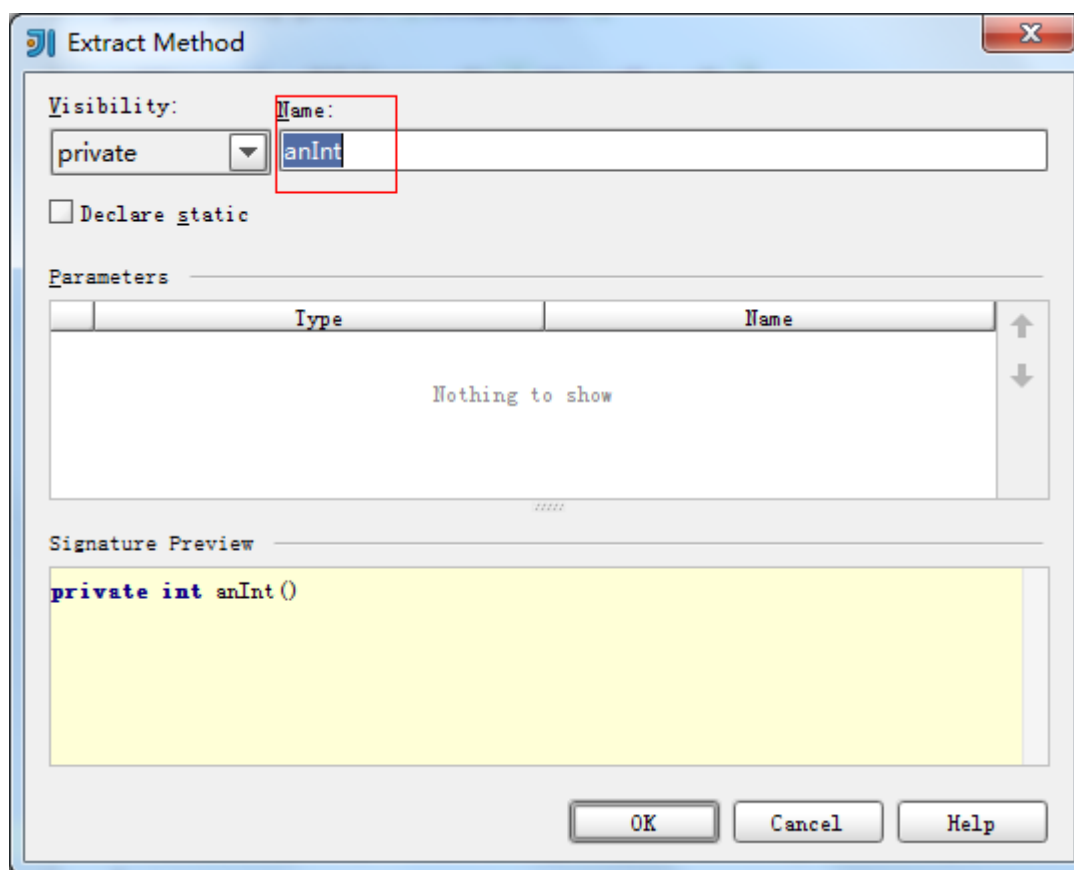
变量转为方法参数。执行 Refactor -> Extract -> Parameter 或 Ctrl+Alt+P，然后输入参数名，回车确认，变量便转为方法参数。



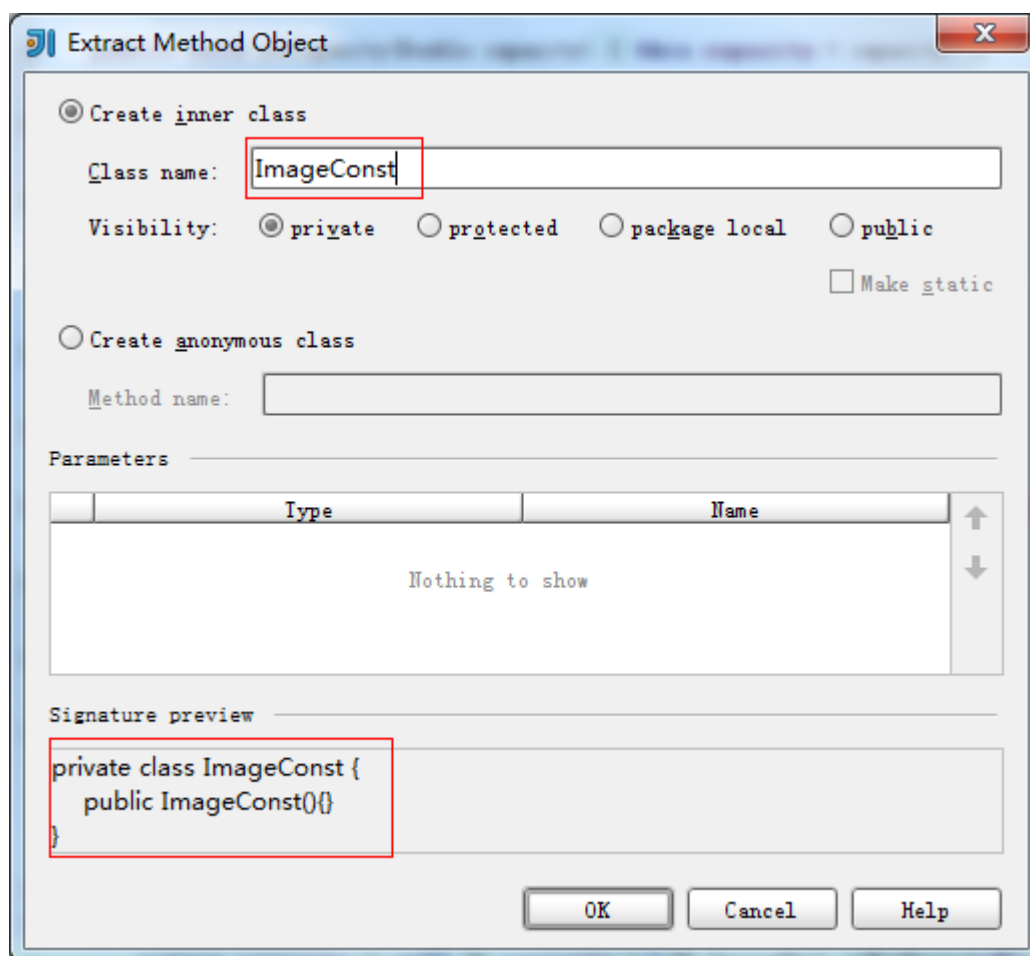
扩展多个参数为一个参数对象。光标移到方法名上，执行 Refactor -> Extract -> Parameter Object，选择要封装的参数，输入对象参数类型。



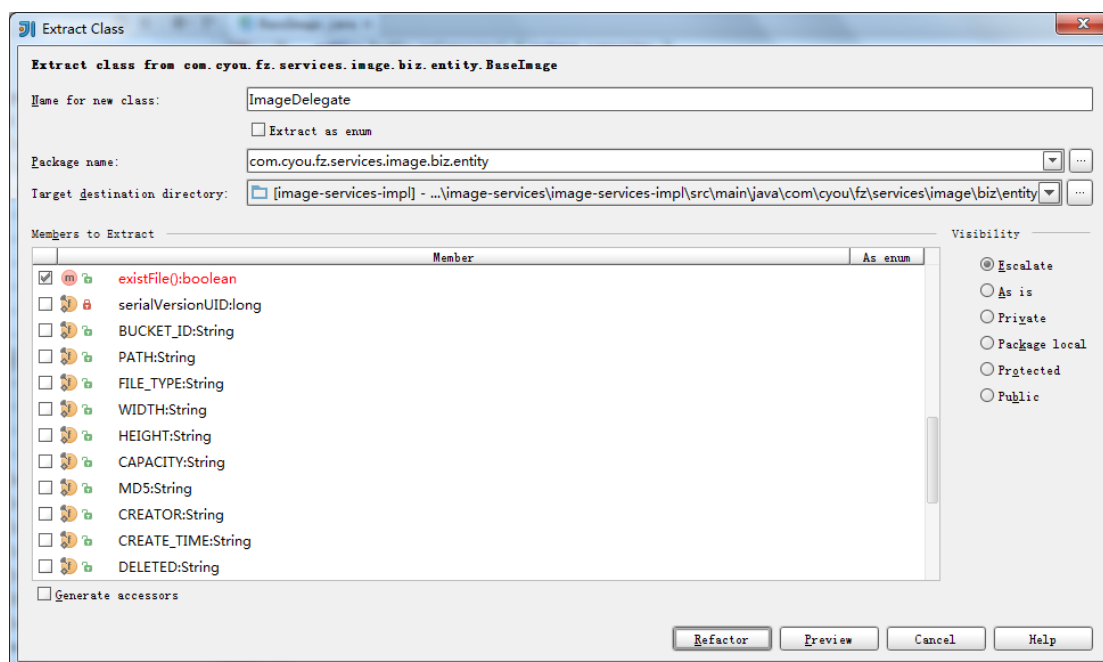
扩展变量或表达式为方法。光标移到要转换的变量上，执行 Refactor -> Extract -> Method 或 Ctrl+Alt+M，输入方法名。



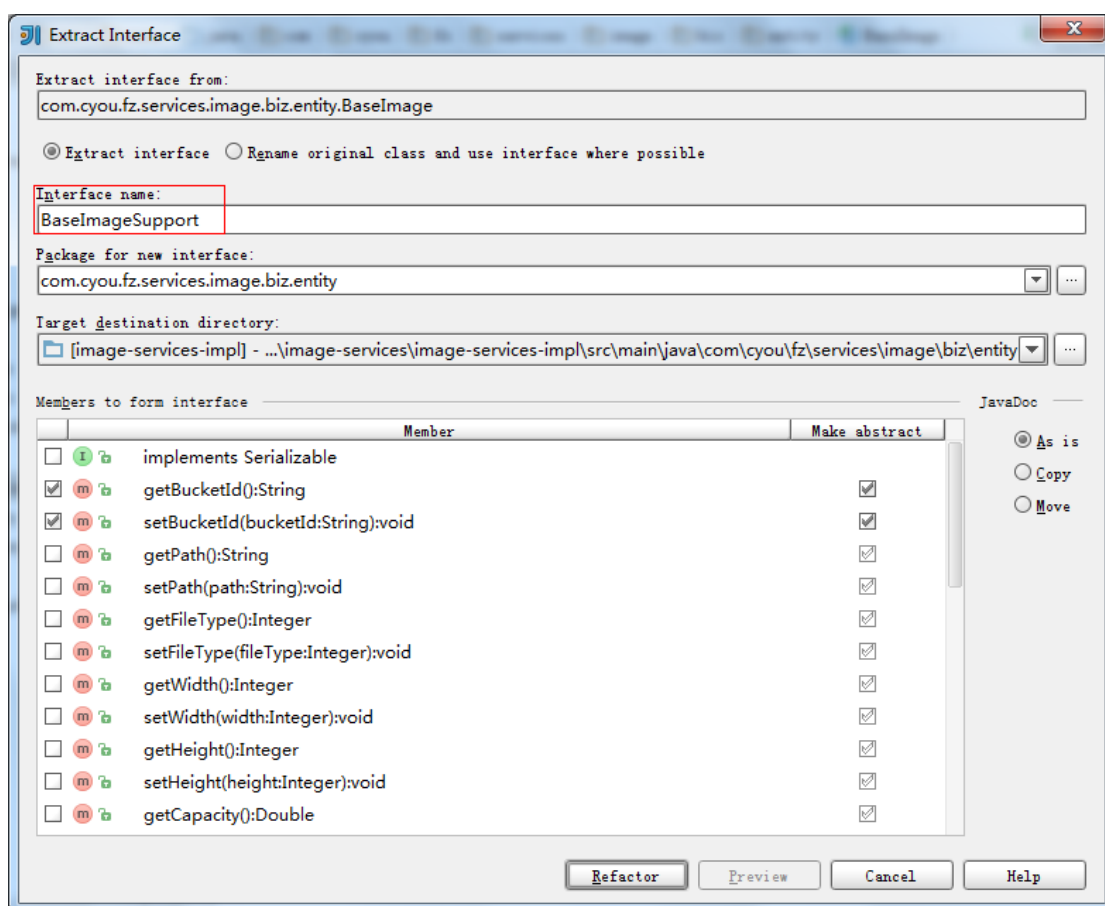
扩展变量或表达式为内部类或匿名类。执行 Refactor -> Extract -> Method Object，输入类名。



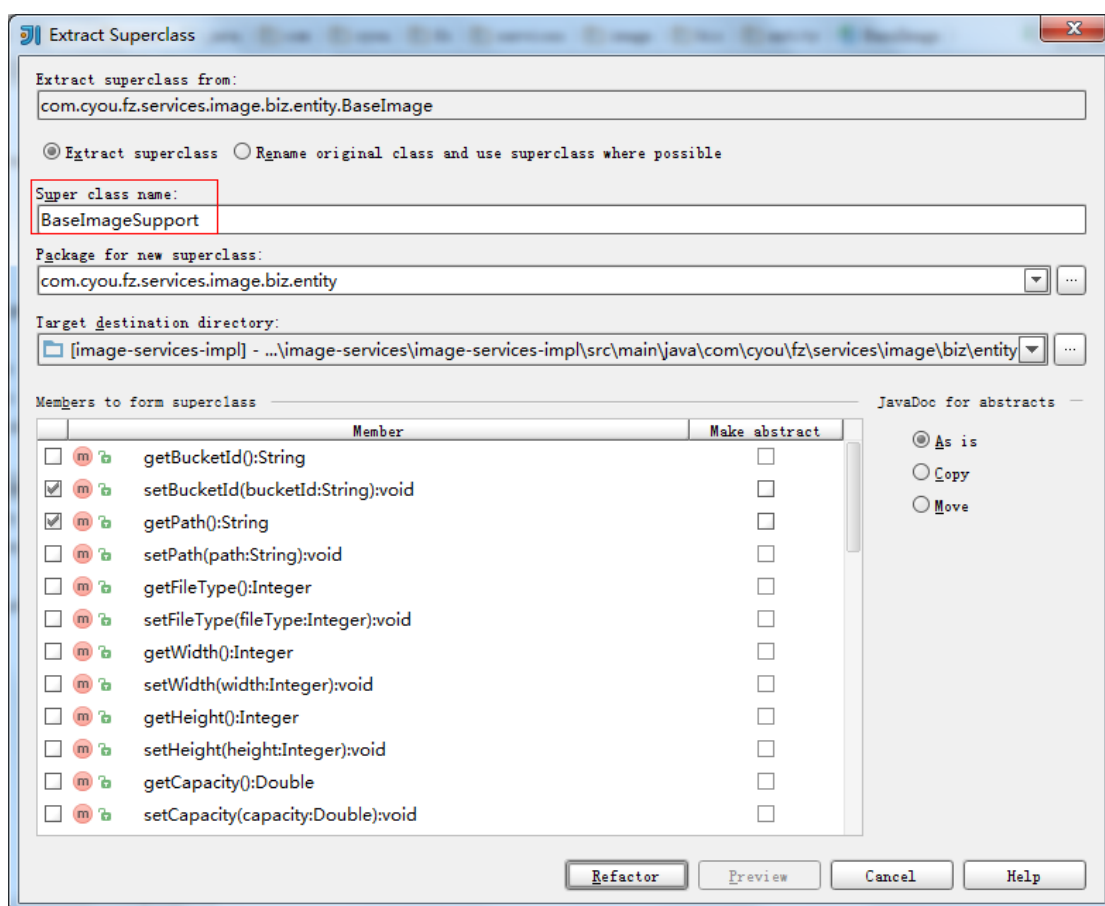
创建委托类。执行 Refactor -> Extract -> Delegate，选择要委托的方法，便生成了委托类。



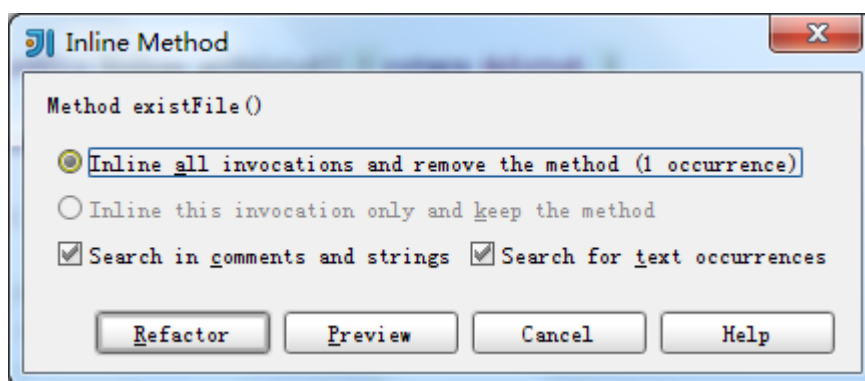
创建接口。执行 Refactor -> Extract -> Interface，选择接口方法，便生成了接口。



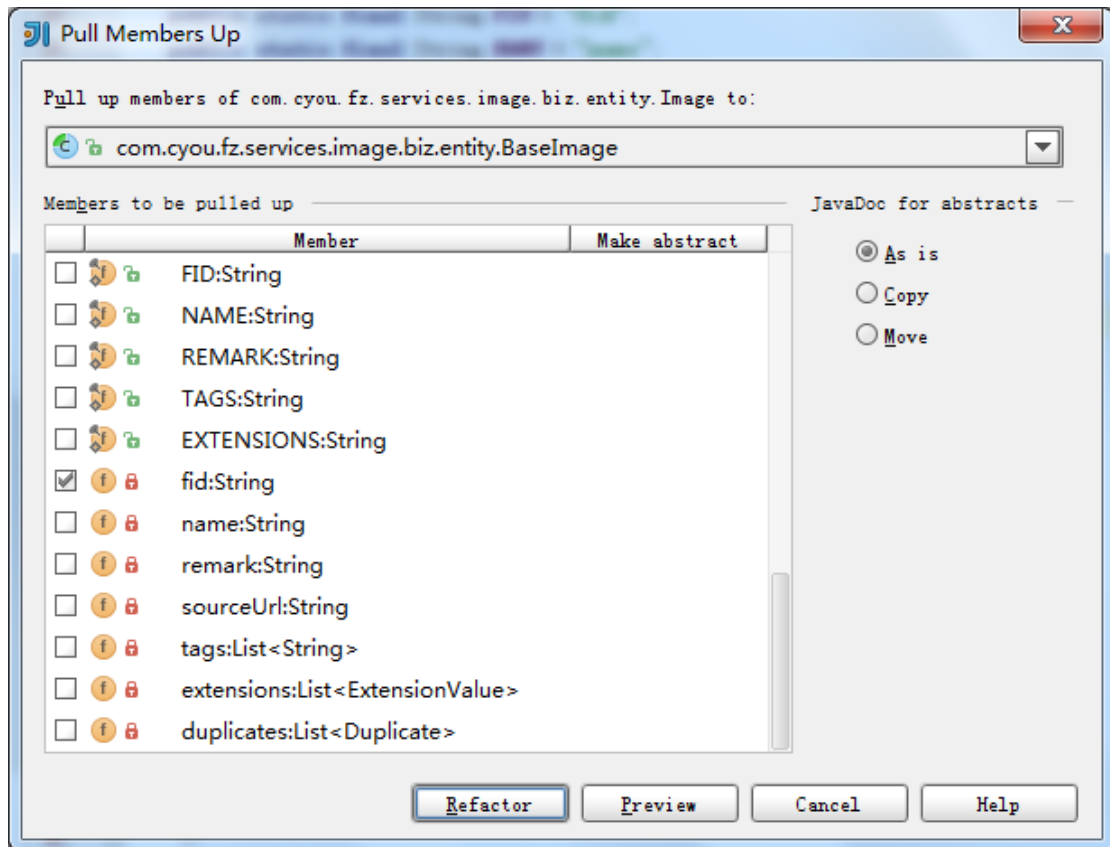
生成父类。执行 Refactor -> Extract -> Superclass，选择父类方法，便生成了父类。



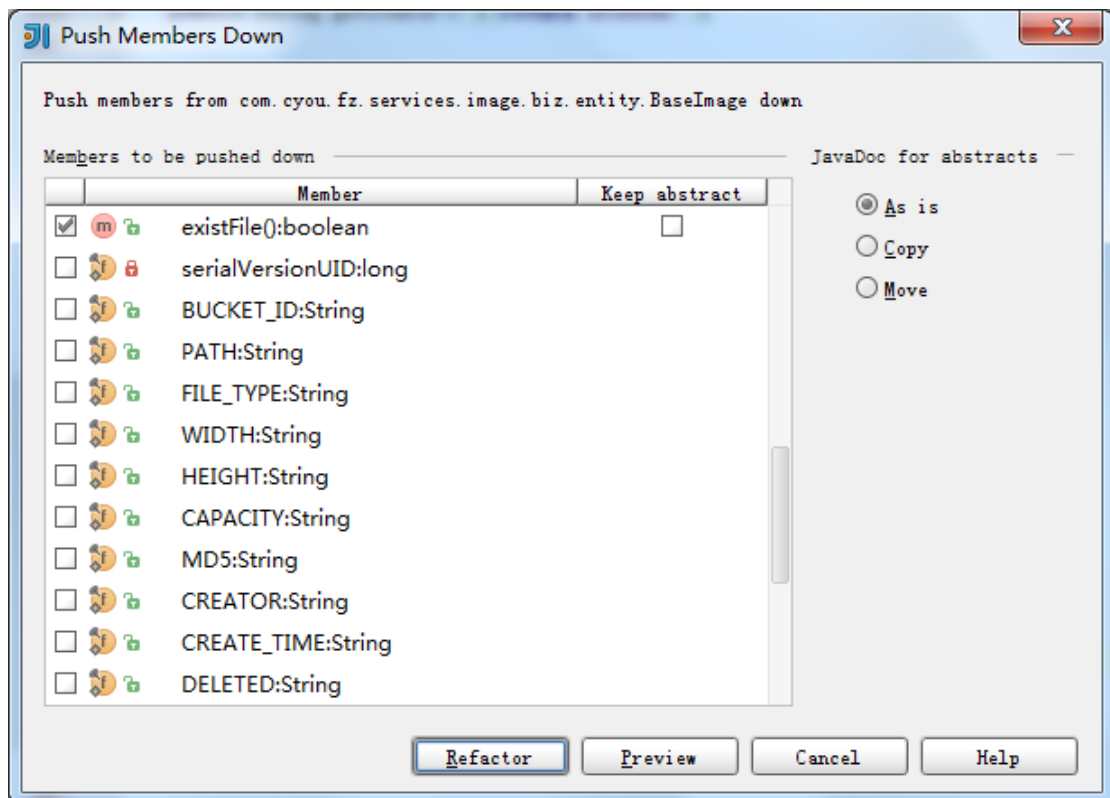
将方法的实现代码复制到调用处并删除方法。执行 Refactor -> Inline 或 Ctrl+Alt+N。



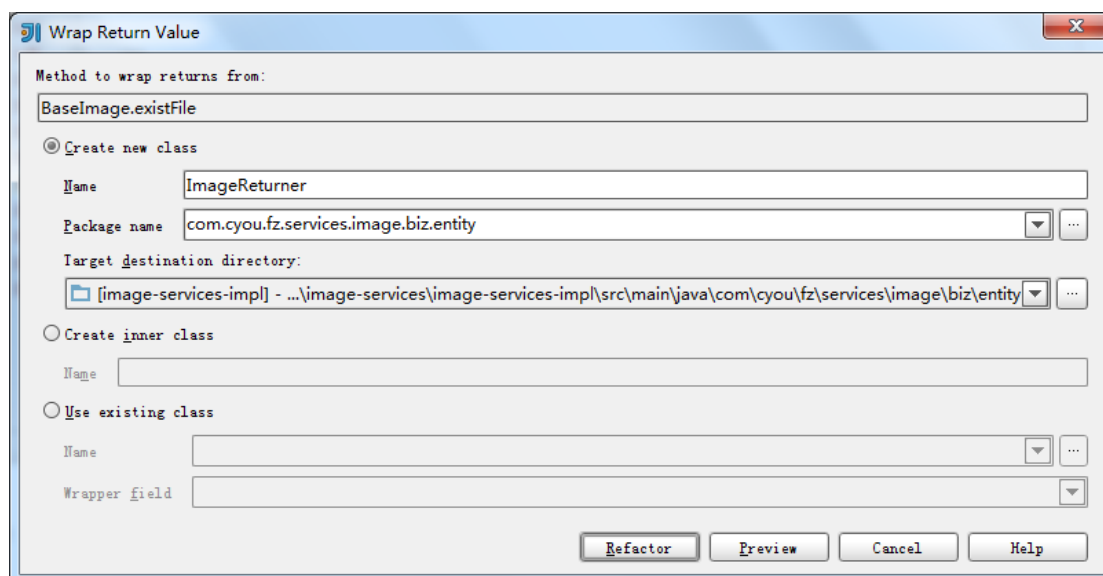
将方法或属性提升到父类中。执行 Refactor -> Pull Members Up。



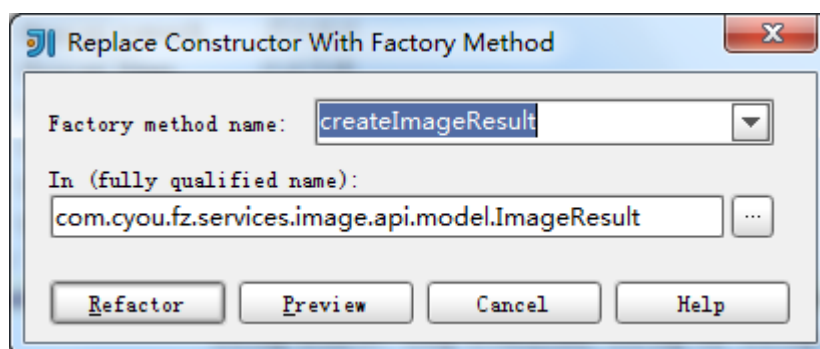
将方法或属性推到子类中。执行 Refactor -> Push Members Down。



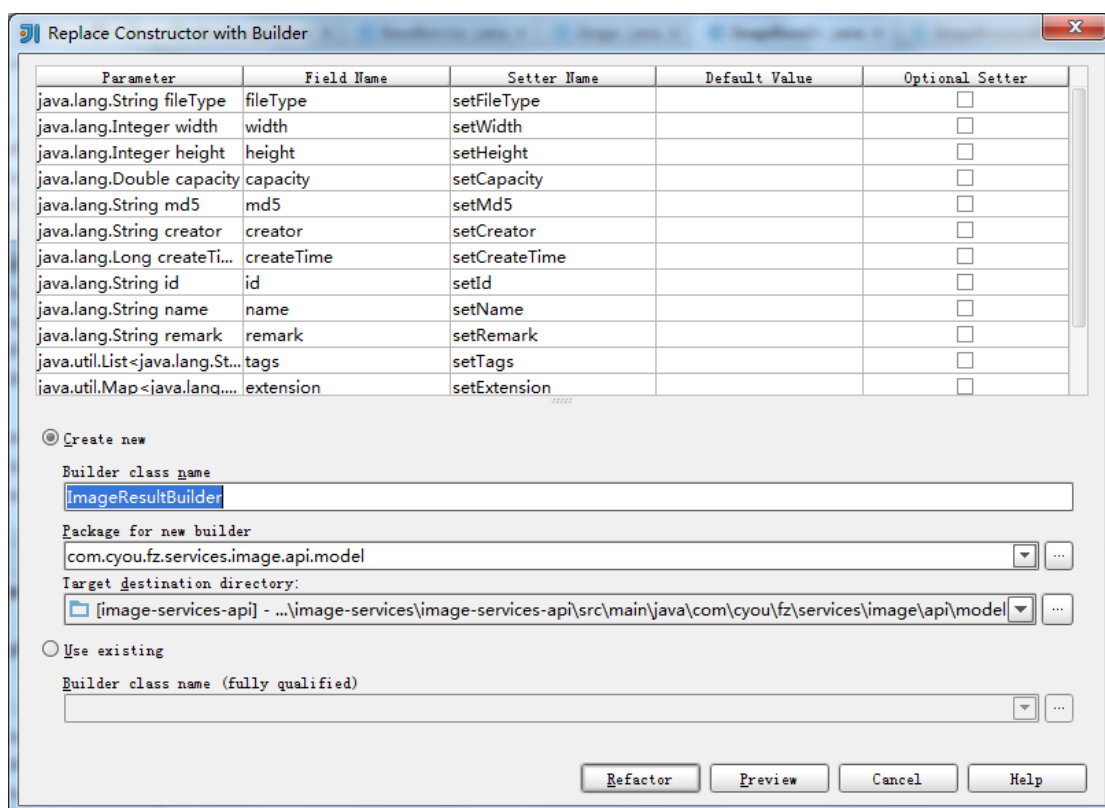
重新封装方法返回值对象。执行 Refactor -> Wrap Return Value，输入返回对象类名。



使用工厂方法模式替换构造器。执行 Refactor -> Replace Constructor With Factory Method，输入工厂方法名。

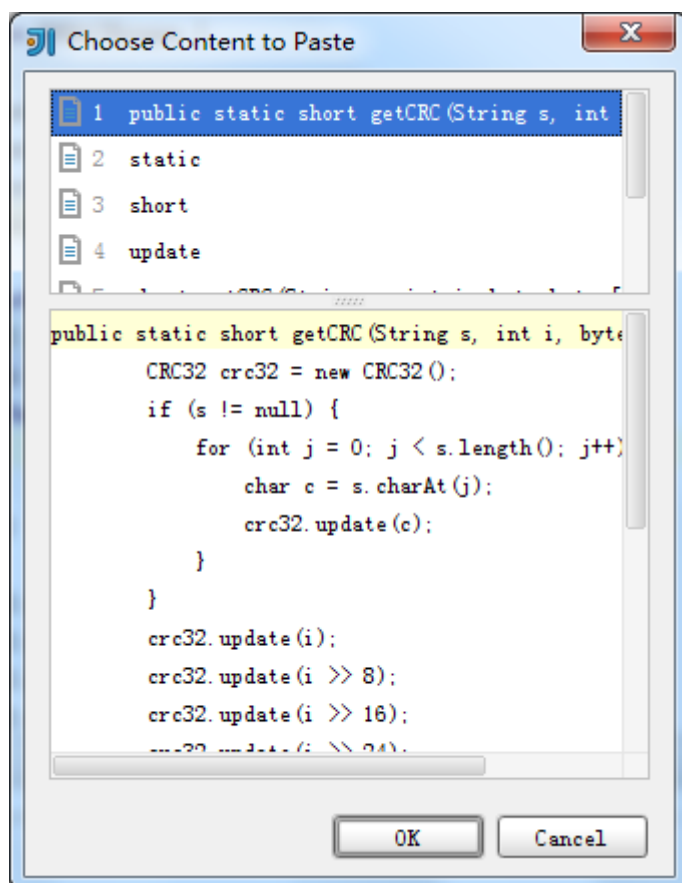


使用创建者模式替换构造器。执行 Refactor -> Replace Constructor With Builder，输入 Builder 的类名。

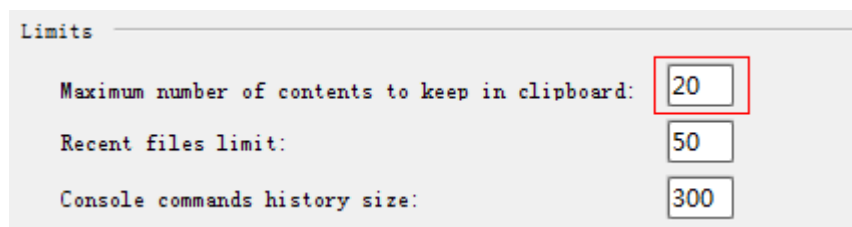


粘贴历史复制记录

快捷键 **Ctrl+Shift+V**，可以显示文本复制的历史记录。

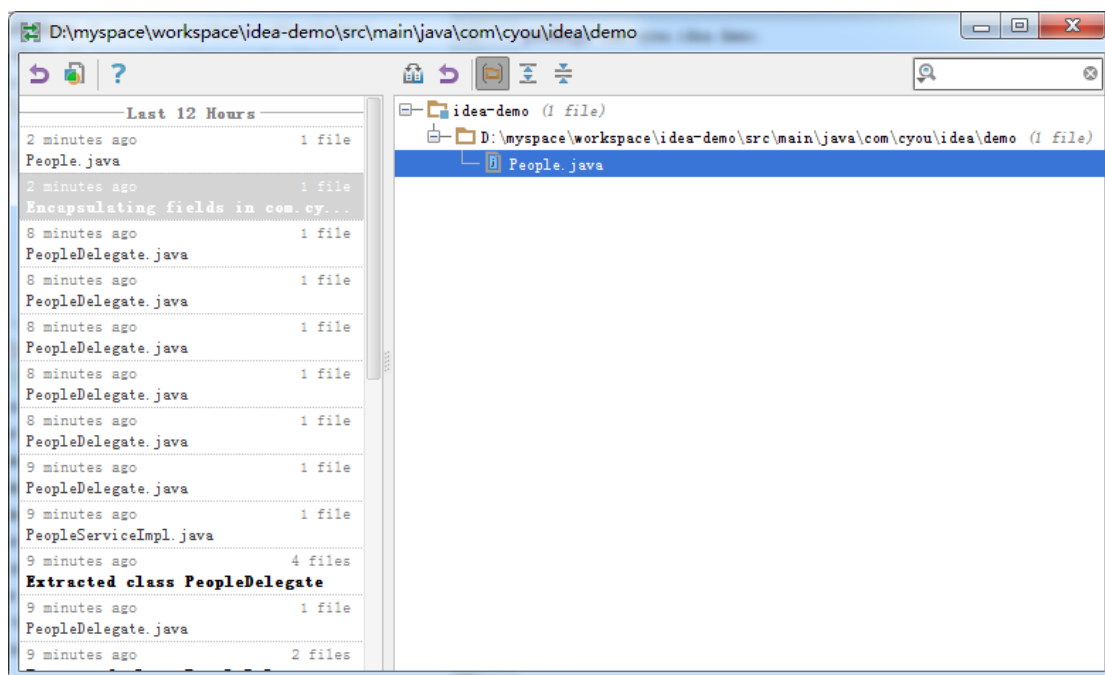


修改复制历史记录数量，执行 Setting-Editor，修改“Maximum number of contents to keep in clipboard”

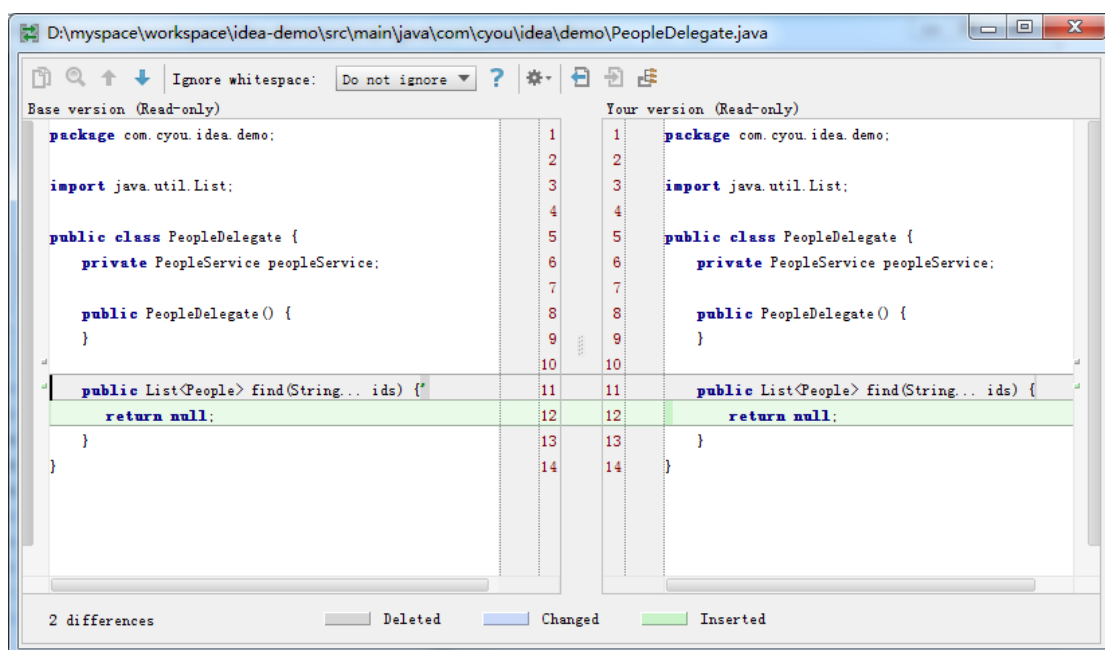


查看本地历史记录

选中文件或文件夹，右键 -> Local History -> Show History，显示本地历史记录。

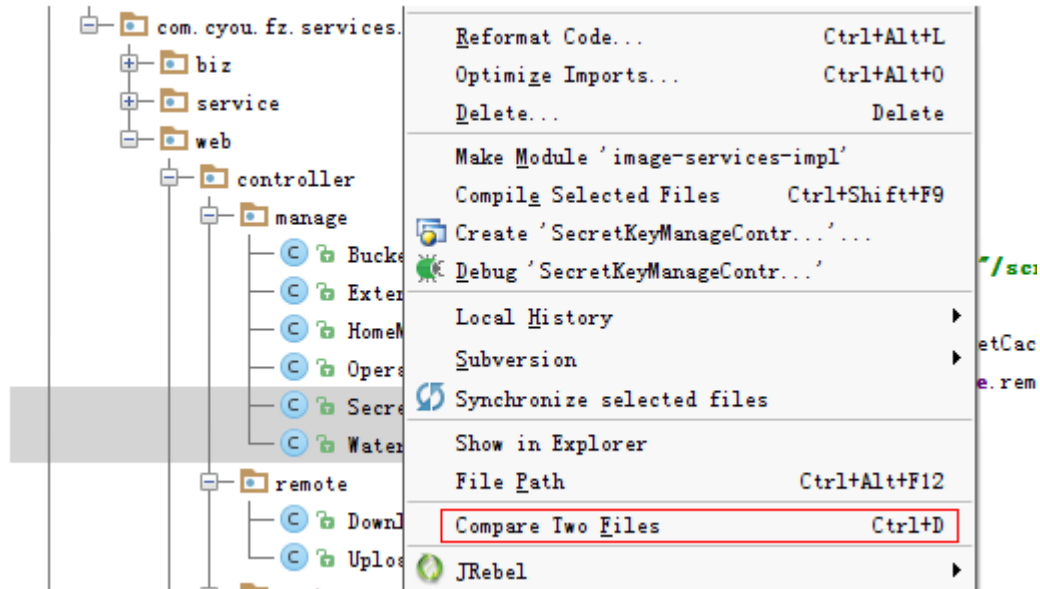


选中指定的版本，双击指定文件可查看与当前版本的差别。

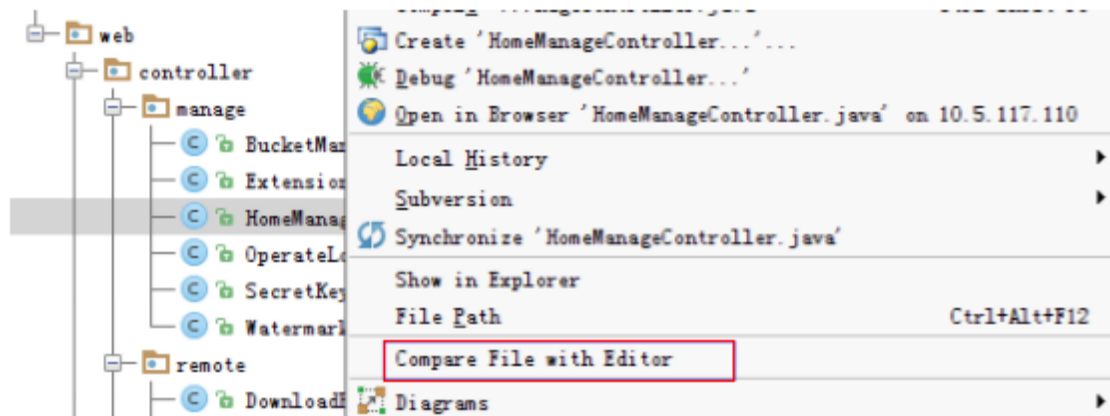


文本比较

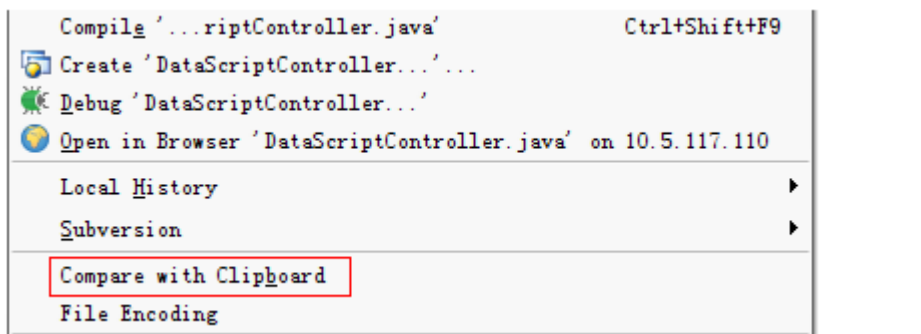
比较两个文件：选择两个文件，按下快捷键 **Ctrl+D** 或者右键选择“Compare Two Files”。



与编辑器比较：选择一个文件，右键选择“Compare File With Editor”。

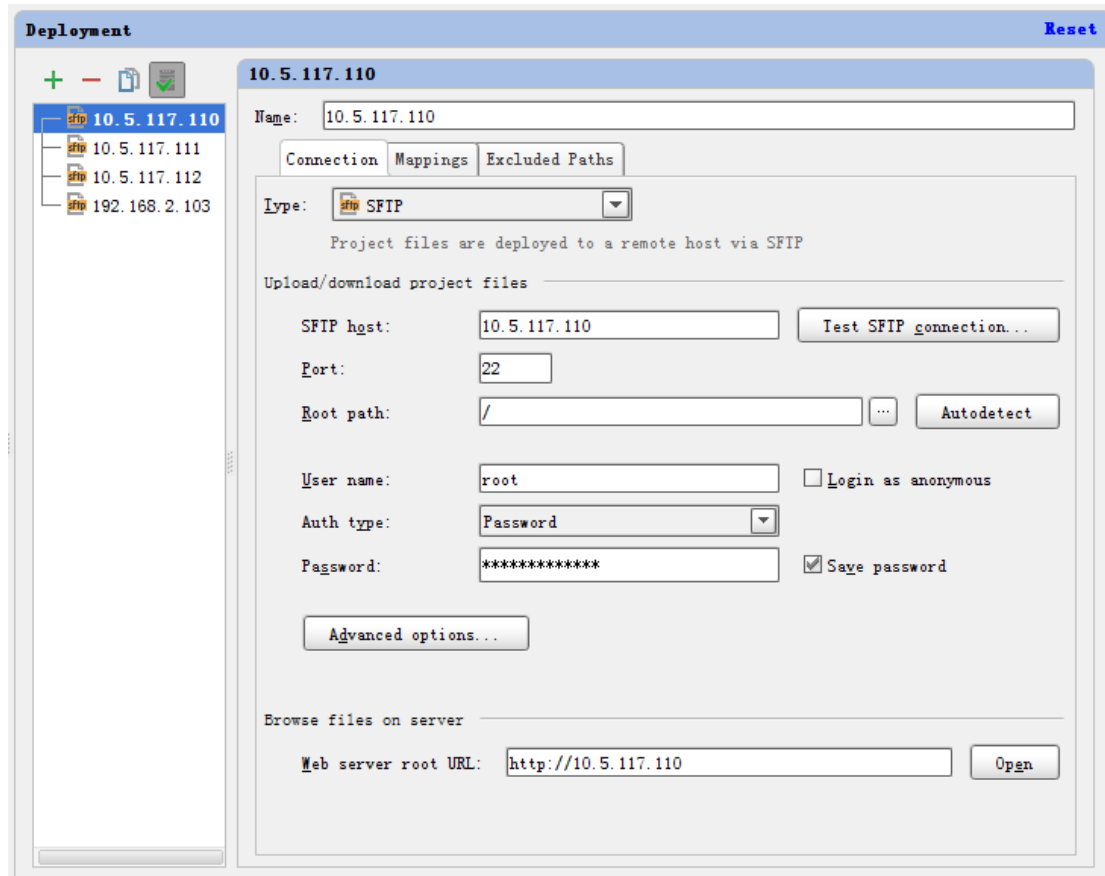


与粘贴板比较：打开一个文件，在编辑器中右键选择“Compare with Clipboard”

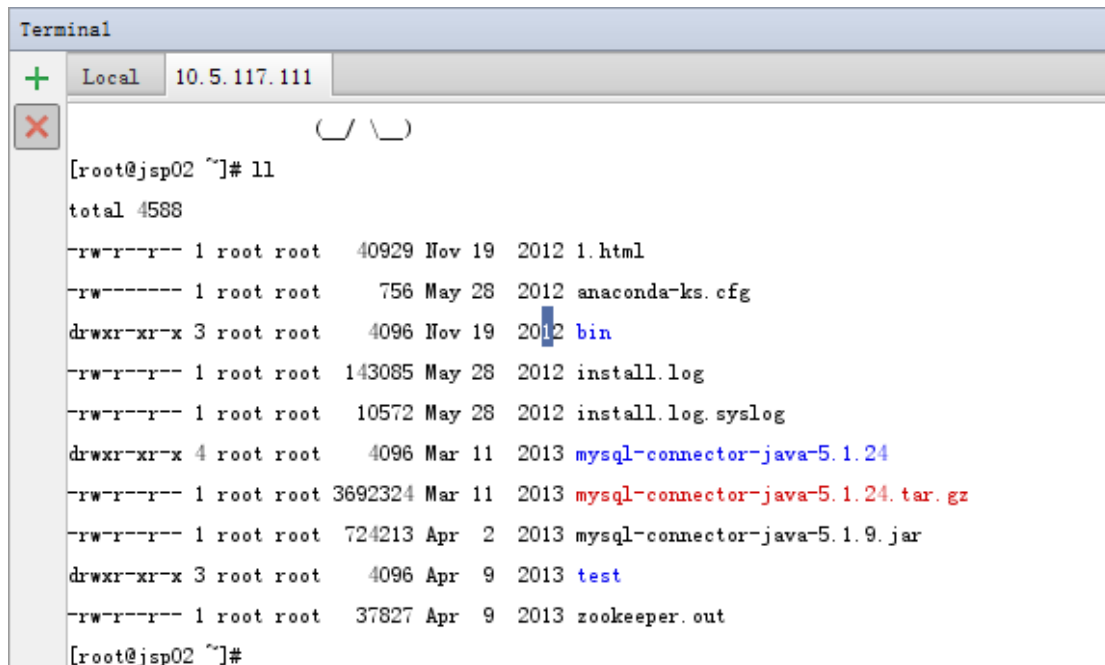


SSH 远程管理

打开 Settings-Deployment，添加 SFTP Server。‘

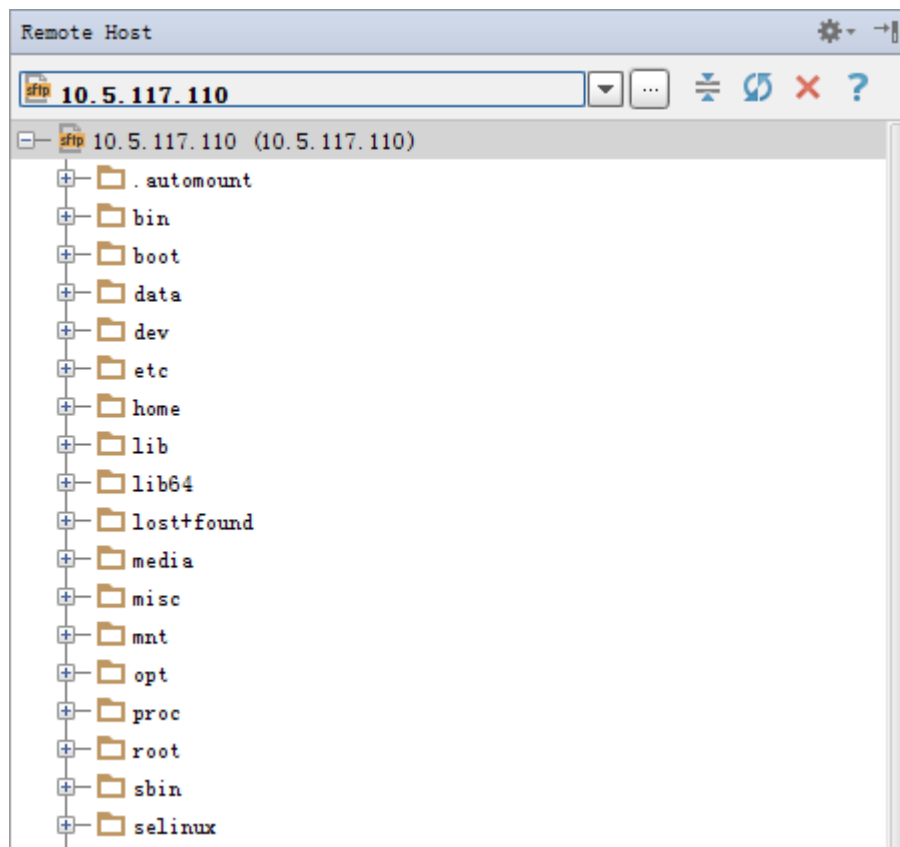


打开 Tool-Start SSH session，选择 ssh server。

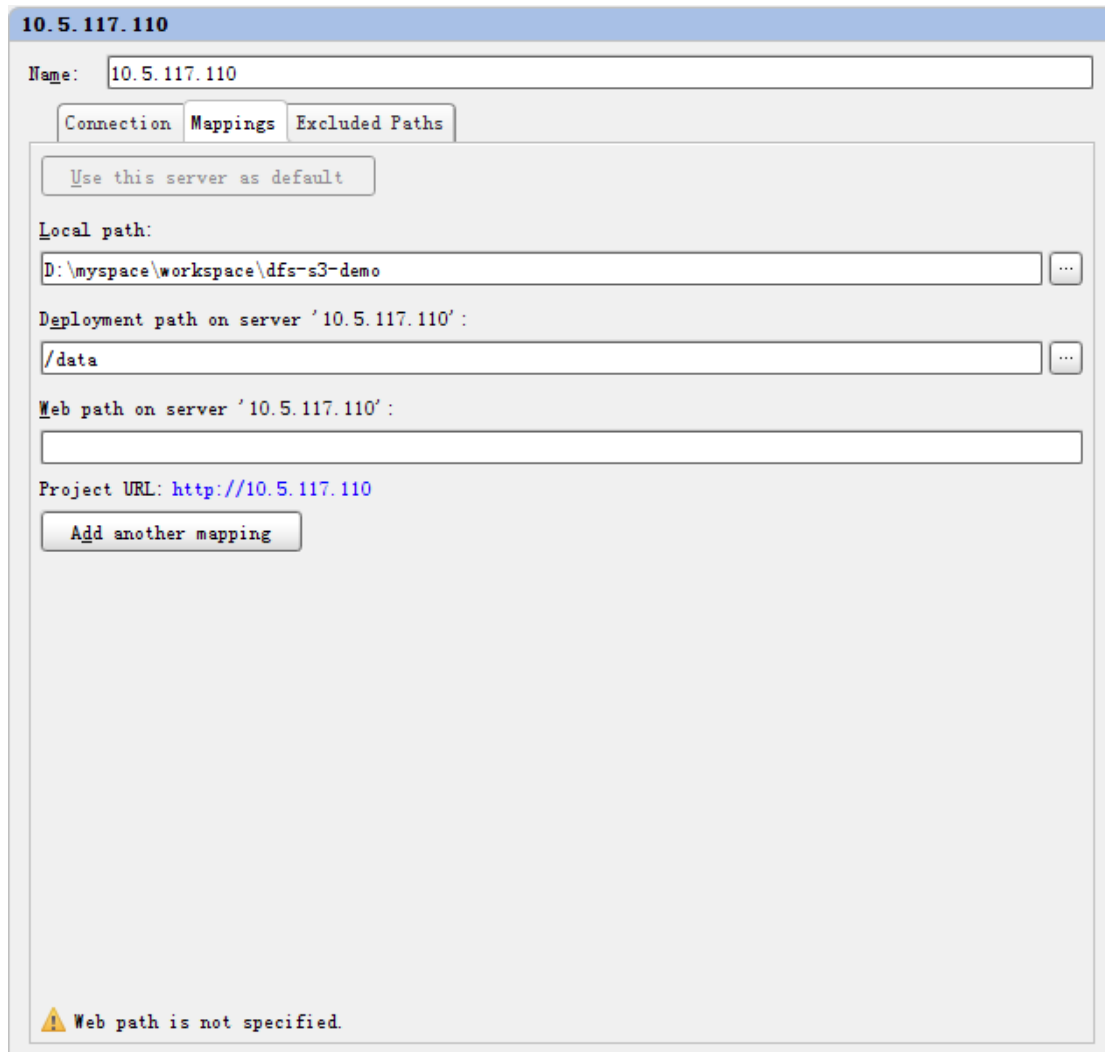


管理远程主机

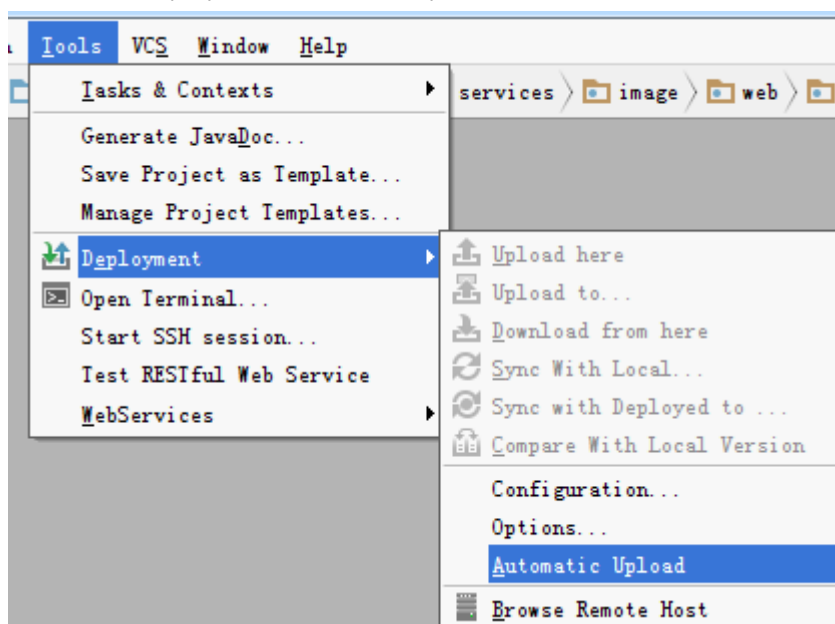
打开 Tool-Deployment-Browse Remote Host，弹出 Remote Host 窗口。可以通过拖动文件从远程服务器上传下载文件。



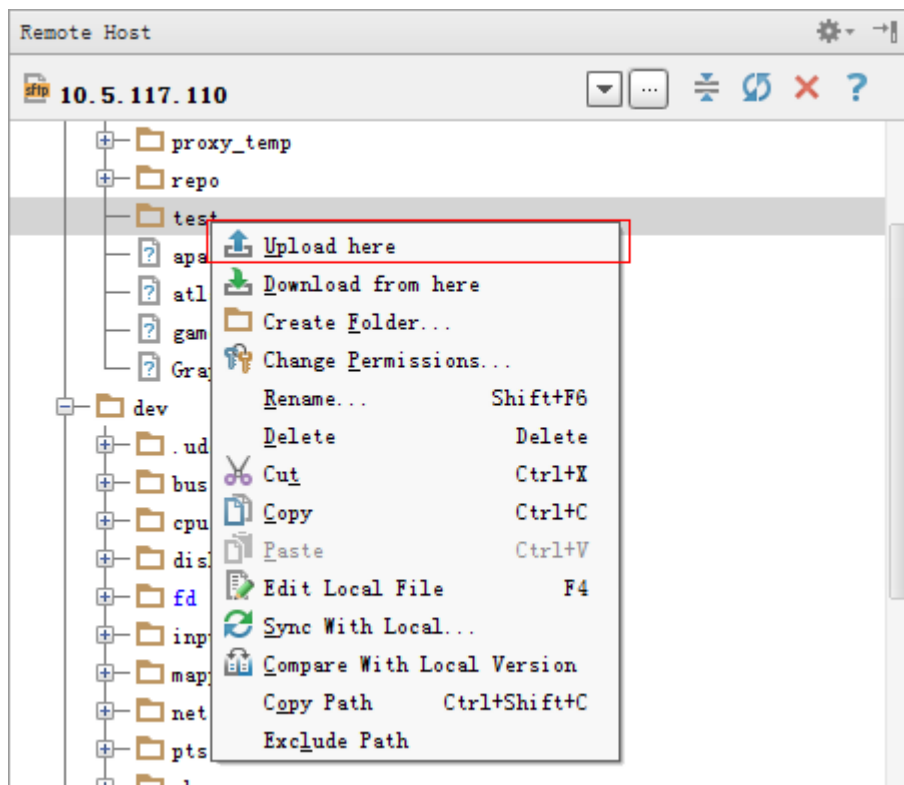
可以通过设置好 Mapping 规则，进行自动上传下载。打开 Settings-Deployment，设置好 Mapping 规则，支持多个。



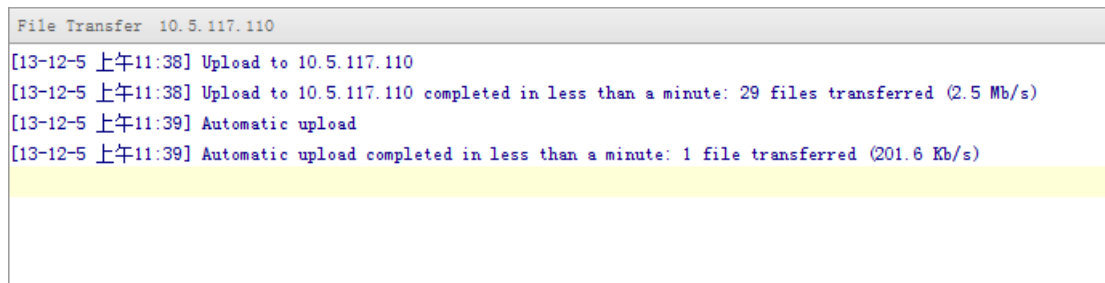
勾选 Tools-Deployment-Automatic Upload,



打开 Tools-Deployment-Browes Remote Host，进入要同步的文件夹，右键点击“Upload here”进行初始操作。



现在添加、修改、删除文件，都会自动上传到远程服务器。File Transfer 会显示上传消息。



快捷键大全

编辑

Ctrl+Space	基本代码补全，输入字母按后列出匹配的词组
Ctrl+Shift+Space	智能代码补全，列出与预期类型一致的方法或变量
Ctrl+Alt+Space	补全类名
Ctrl+Shift+Enter	补全语句
Ctrl+P	显示方法参数

Ctrl+Q	显示注释文档
Shift+F1	显示外部文档
Ctrl+mouse over code	显示描述信息
Ctrl+F1	显示提示、警告、错误等信息
Alt+Insert	生成代码，生成 Getter、Setter、构造器等
Ctrl+O	重写父类方法
Ctrl+I	实现接口方法
Ctrl+Alt+T	使用(if..else, try..catch, for, synchronized 等)包围选中语句
Ctrl+/ /	使用“//”注释或取消注释
Ctrl+Shift+/ /	使用“/** */”注释或取消注释
Ctrl+W	选择代码块，连续按会增加选择外层的代码块
Ctrl+Shift+W	与“Ctrl+W”相反，减少选择代码块
Alt+Q	显示类描述信息
Alt+Enter-fixes	显示快速修复列表
Ctrl+Alt+L	格式化代码
Ctrl+Alt+O	优化 Imports
Ctrl+Alt+I	自动优化代码缩进
Tab/Shift+Tab	缩进代码/取消缩进代码
Ctrl+X or Shift+Delete	剪切代码，未选择代码时剪切当前行
Ctrl+C or Ctrl+Insert	复制代码，未选择代码时复制当前行
Ctrl+V or Shift+Insert	粘贴代码
Ctrl+Shift+V	粘贴最近复制的内容
Ctrl+D	重复代码，未选择代码时重复当前行
Ctrl+Y	删除行，未选择时删除当前行
Ctrl+Shift+J	合并多行为一行
Ctrl+Enter	分割一行为多行
Shift+Enter	使光标所在位置的下一行为新行
Ctrl+Shift+U	对选中内容进行大小写切换
Ctrl+Shift+]/[选中到代码块的开始/结束
Ctrl+Delete	删除从光标所在位置到单词结束位置的字符
Ctrl+Backspace	删除从单词起始位置到光标所在位置的字符
Ctrl+NumPad+/-	展开或收起代码块
Ctrl+Shift+NumPad+	展开所有代码块
Ctrl+Shift+NumPad-	收起所有代码块
Ctrl+F4	关闭当前编辑页

查找/替换

Ctrl+F	查找
F3	查找下一个
Shift+F3	查找上一个
Ctrl+R	替换
Ctrl+Shift+F	目录内查找
Ctrl+Shift+R	目录内替换
Ctrl+Shift+S	语法模板搜索
Ctrl+Shift+M	语法模板替换
Alt+F7	查找被使用处
Ctrl+F7	查找当前文件中的使用处
Ctrl+Shift+F7	高亮当前文件中的使用处
Ctrl+Alt+F7	列出使用者

编译/运行

Ctrl+F9	Make 模块、项目
Ctrl+Shift+F9	编译选中的文件、模块、项目
Alt+Shift+F10	选择配置后运行代码
Alt+Shift+F9	选择配置后调试代码
Shift+F10	运行代码
Shift+F9	调试代码
Ctrl+F2	停止调试
Ctrl+Shift+F10	运行代码

调试

F8	单步调试，不进入函数内部
F7	单步调试，进入函数内部
Shift+F7:	选择要进入的函数
Shift+F8	跳出函数
Alt+F9	运行到断点
Alt+F8	执行表达式查看结果

F9	继续执行，进入下一个断点或执行完程序
Ctrl+F8	设置/取消当前行断点
Ctrl+Shift+F8	查看断点

导航

Double Shift	查找所有
Ctrl+N	查找类
Ctrl+Shift+N	查找文件
Ctrl+Alt+Shift+N	Go to symbol
Alt+Right/Left	左右切换 Tab
F12	回到上一个打开的窗口
Esc	焦点回到编辑器
Shift+Esc	隐藏打开的视图
Ctrl+Shift+F4	关闭当前 Tab
Ctrl+G	跳刀指定行
Ctrl+E	显示最近打开的文件
Ctrl+Alt+Left	跳到光标的上一个位置
Ctrl+Alt+Right	跳到光标的下一个位置
Ctrl+Shift+Backspace	跳到上一个编辑处
Alt+F1	选择当前文件显示在不同的视图中
Ctrl+B or Ctrl+Click	跳到类声明处
Ctrl+Alt+B	跳到实现类/方法
Ctrl+Shift+I	显示类/变量/方法定义
Ctrl+Shift+B	跳到类型定义处
Ctrl+U	跳到父类/方法
Alt+Up	光标移动到上一个方法
Alt+Down	光标移动到下一个方法
Ctrl+]]	光标移动到代码块的起始位置
Ctrl+]]	光标移动到代码块的结束位置
Ctrl+F12	显示文件结构
Ctrl+H	显示类层级
Ctrl+Shift+H	显示方法层级
Ctrl+Alt+H	显示类/方法调用层级
F2	光标移动到下一个错误

Shift+F2	光标移动到上一个错误
F4	编辑源码，光标移到编辑器内
Ctrl+Enter	查看源码，光标没移到编辑器内
Alt+Home	显示导航面包屑
F11	当前行设置书签
Shift+F11	显示所有书签
Ctrl+F11	设置书签号[0-9]
Ctrl+[0-9]	跳到书签号[0-9]所在位置

重构

F5	复制类
F6	移动类
Alt+Delete	安全删除，删除前会提示调用处
Shift+F6	重命名
Ctrl+F6	重构方法参数、Exception 等
Ctrl+Alt+N	合并多行为一行
Ctrl+Alt+M	提取为新方法
Ctrl+Alt+V	提取为新变量
Ctrl+Alt+F	提取为对象新属性
Ctrl+Alt+C	提取为新静态常量

版本控制/本地历史

Ctrl+K	提交改动到 VCS
Ctrl+T	从 VCS 上更新
Alt+Shift+C	查看最近的改动记录
Alt+BackQuote(`)	显示 VCS 操作列表

Live Template

Ctrl+Alt+J	使用 Live Template 包围选中代码
Ctrl+J	快速插入 Live Template
iter	快速生成 for...in 语句

inst	快速生成“if instanceof”语句
itco	快速生成 iterator 的 for 循环
itit	快速生成 iterator 的 while 循环
itli	快速生成 list 的 for(i)循环
psf	快速生成“public static final” 语句
thr	快速生成“throw new” 语句

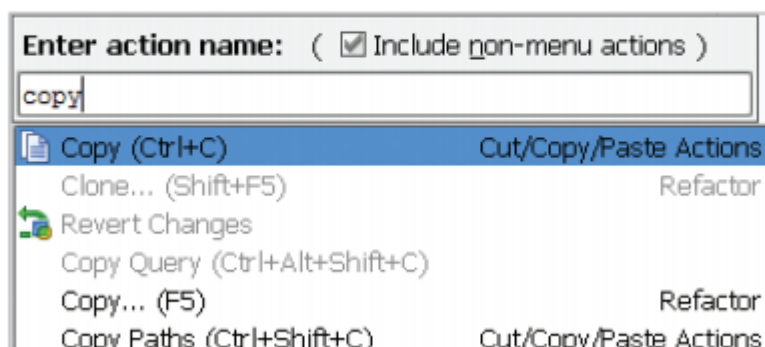
代码生成

Alt+0	聚焦到 Messages 窗口
Alt+1	聚焦到 Project 窗口
Alt+2	聚焦到 Favorite 窗口
Alt+3	聚焦到 Find 窗口
Alt+4	聚焦到 Run 窗口
Alt+5	聚焦到 Debug 窗口
Alt+6	聚焦到 TODO 窗口
Alt+7	聚焦到 Structure 窗口
Alt+8	聚焦到 Hierarchy 窗口
Alt+9	聚焦到 Change 窗口
Ctrl+S	保存文件
Ctrl+Alt+Y	与本地文件同步
Alt+Shift+F	添加到收藏夹
Alt+Shift+I	检查当前文件，包括 Javadoc 问题、可能存在的 bug 等
Ctrl+BackQuote (`)	模式切换，包括文本外观、快捷键、编辑器外观、代码样式。
Ctrl+Alt+S	打开 settings 窗口

如何查找

如果忘记了如何操作，可以使用 **Ctrl+Shift+A** 搜索操作方式：

To find any action inside the IDE use Find Action (Ctrl+Shift+A)



新特性

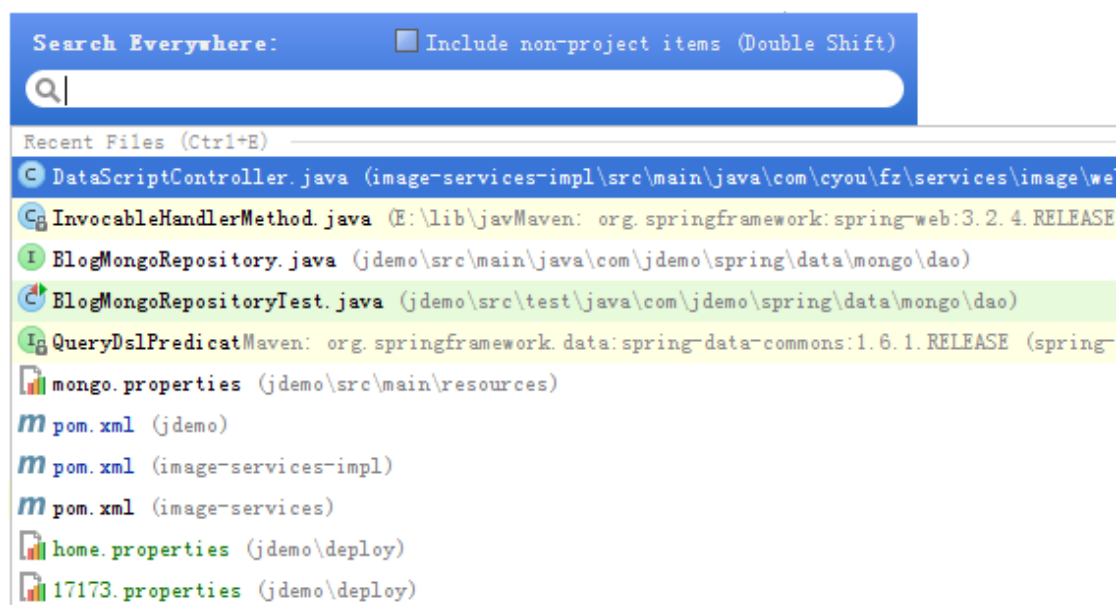
Terminal

命令终端。快捷键 Alt+F12。



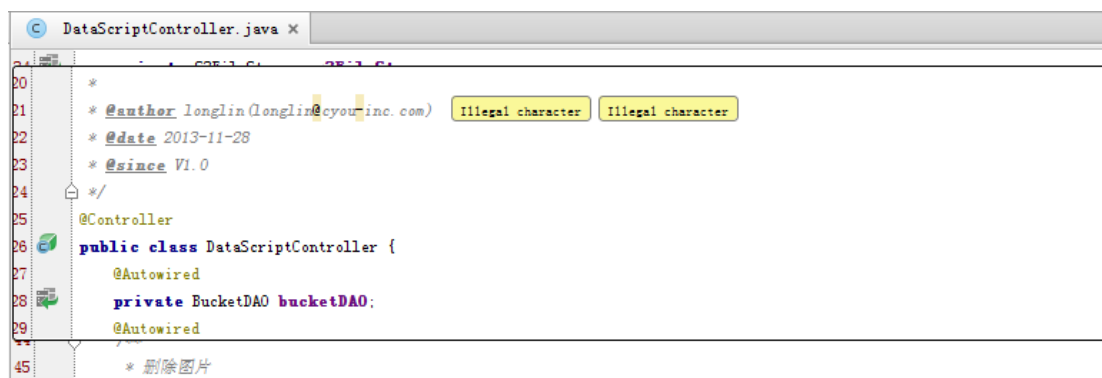
Search Anywhere

搜索所有文件，Shift 按两下。



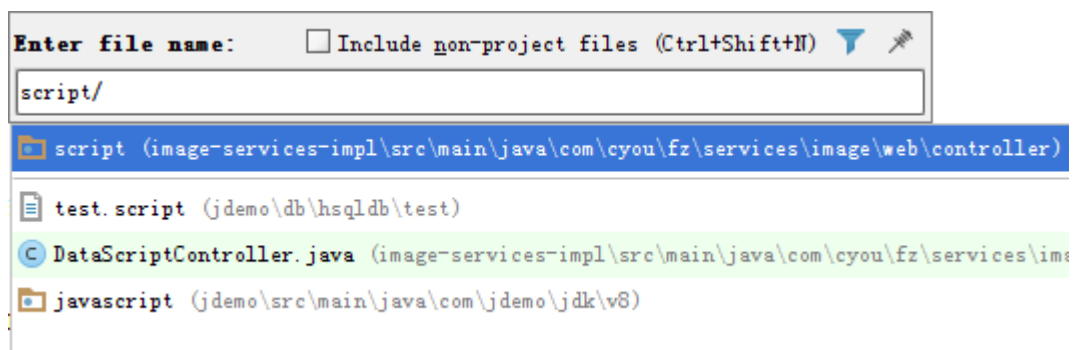
LENS Mode

透镜模式，鼠标移动到滚动条显示超出当前视图区的代码。



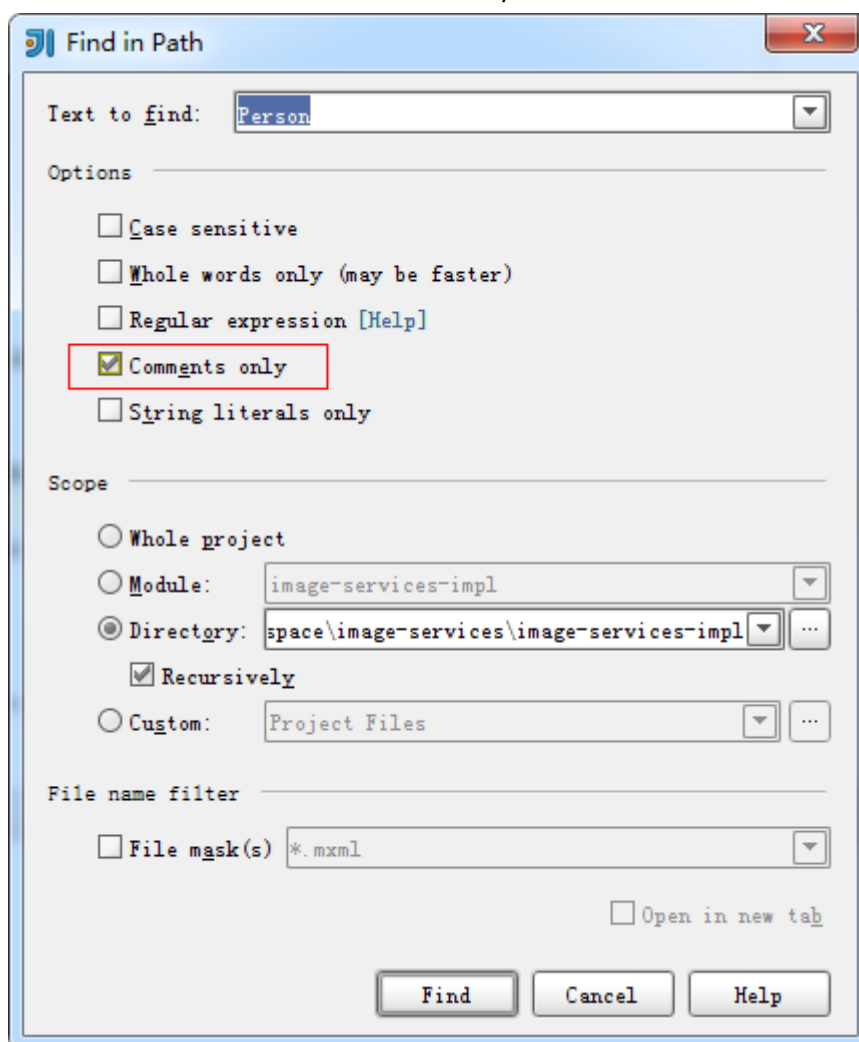
文件夹搜索

快捷键 Ctrl+Shift+N，文件夹以/结束



搜索注释内容

快捷键 Ctrl+Shift+N, 勾选“Comments only”



Spring Bean Explorer

SpringBean 查看器

