

# IDEA

---

@(IDEA)[2021年2月4日08:50:05]

## IDEA

IDEA目录结构

设置的目录结构

IDEA工程

设置视图

工程界面

模块

IDEA中创建Module

删除Module

查看项目配置

编辑器配置

Editor——General

设置自动导包功能

设置鼠标滚轮修改字体大小

显示行号和方法间的分割线

忽略大小写提示

设置单行显示tabs

Editor——Color Scheme

修改代码中注释的字体颜色

Editor——Code Style

Editor——File and Code Templates

新建文档的头部注释信息

Editor——File Encodings

设置项目文件编码

对单一文件的编码修改

编译配置——Build,Execution,Deployment

Compiler——设置自动编译

断点调试

窗口划分方式

Keymap

快捷键

模板

1.Live Templates(实时代码模板)功能介绍

举例

静态JavaWeb和Tomcat

关联数据库

启动Mysql服务

设置时区

设置自动同步

IDEA中使用git

配置

IDEA中的GIT基本操作

版本控制

查看历史版本——git log

版本切换

分支管理

新建分支

切换分支

合并分支

冲突处理

push到远程库

从远程库clone

关闭自动更新

清空所有缓存和索引

## # IDEA目录结构

---

#### 4.查看安装目录结构

developer (D:) > developer_tools > IntelliJ IDEA 2017.1.4 >			
共享 (S) 帮助 (H)			
共享 < 新建文件夹			
名称	修改日期	类型	大小
bin	2017/8/22 星期...	文件夹	
help	2017/8/22 星期...	文件夹	
jre64	2017/8/22 星期...	文件夹	
lib	2017/8/22 星期...	文件夹	
license	2017/8/22 星期...	文件夹	
plugins	2017/8/22 星期...	文件夹	
redist	2017/8/22 星期...	文件夹	
build.txt	2017/6/6 星期二 ...	TXT 文件	1 KB
Install-Windows.zip.txt	2017/6/6 星期二 ...	TXT 文件	3 KB
ipr.reg	2017/6/6 星期二 ...	注册表项	1 KB

bin: 容器，执行文件和启动参数等

help: 快捷键文档和其他帮助文档

jre64: 64 位java 运行环境

lib: idea 依赖的类库

license: 各个插件许可

plugin: 插件

其中：bin 目录下：



这里以我的电脑系统(64 位 windows7, 16G 内存)为例，说明一下如何调整 VM 配置文件：

```
1 -Xms128m
2 -Xmx750m
3 -XX:ReservedCodeCacheSize=240m
4 -XX:+UseConcMarkSweepGC
5 -XX:SoftRefLRUPolicyMSPerMB=50
6 -ea
7 -Dsun.io.useCanonCaches=false
8 -Djava.net.preferIPv4Stack=true
9 -XX:+HeapDumpOnOutOfMemoryError
10 -XX:-OmitStackTraceInFastThrow
11
```

1. 大家根据电脑系统的位数，选择 32 位的 VM 配置文件或者 64 位的 VM 配置文件
2. 32 位操作系统内存不会超过 4G，所以没有多大空间可以调整，建议不用调整了
3. 64 位操作系统中 8G 内存以下的机子或是静态页面开发者是无需修改的。
4. 64 位操作系统且内存大于 8G 的，如果你是开发大型项目、Java 项目或是 Android 项目，建议进行修改，常修改的就是下面 3 个参数：

`-Xms128m, 16 G 内存的机器可尝试设置为 -Xms512m`  
(设置初始的内存数，增加该值可以提高 Java 程序的启动速度。)

`-Xmx750m, 16 G 内存的机器可尝试设置为 -Xmx1500m`  
(设置最大内存数，提高该值，可以减少内存 Garage 收集的频率，提高程序性能)

`-XX:ReservedCodeCacheSize=240m, 16G 内存的机器可尝试设置为`  
`-XX:ReservedCodeCacheSize=500m`  
(保留代码占用的内存容量)

- 1 `-Xms128m, 16 G 内存的机器可尝试设置为 -Xms512m`
- 2 (设置初始的内存数，增加该值可以提高 Java 程序的启动速度。 )
- 3 `-Xmx750m, 16 G 内存的机器可尝试设置为 -Xmx1500m`
- 4 (设置最大内存数，提高该值，可以减少内存 Garage 收集的频率，提高程序性能)
- 5 `-XX:ReservedCodeCacheSize=240m, 16G 内存的机器可尝试设置为`
- 6 `-XX:ReservedCodeCacheSize=500m`
- 7 (保留代码占用的内存容量)

## 设置的目录结构

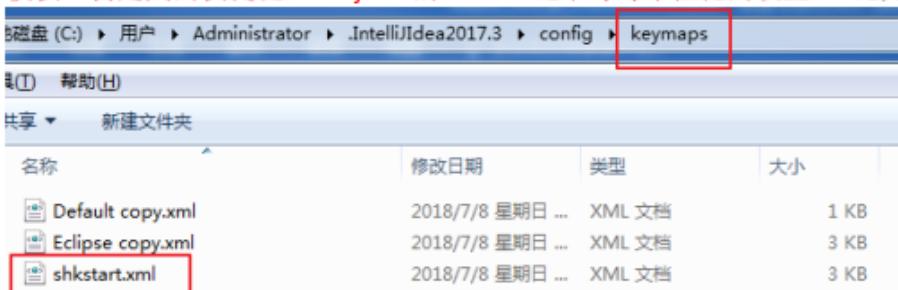
## 5.查看设置目录结构



这是 IDEA 的各种配置的保存目录。这个设置目录有一个特性，就是你删除掉整个目录之后，重新启动 IntelliJ IDEA 会再自动帮你生成一个全新的默认配置，所以很多时候如果你把 IntelliJ IDEA 配置改坏了，没关系，删掉该目录，一切都会还原到默认。

### 5.1 config 目录

config 目录是 IntelliJ IDEA 个性化化配置目录，或者说是整个 IDE 设置目录。此目录可看成是最重要的目录，没有之一，如果你还记得安装篇的介绍的时候，安装新版本的 IntelliJ IDEA 会自动扫描硬盘上的旧配置目录，指的就是该目录。这个目录主要记录了：IDE 主要配置功能、自定义的代码模板、自定义的文件模板、自定义的快捷键、Project 的 tasks 记录等等个性化的设置。比如：



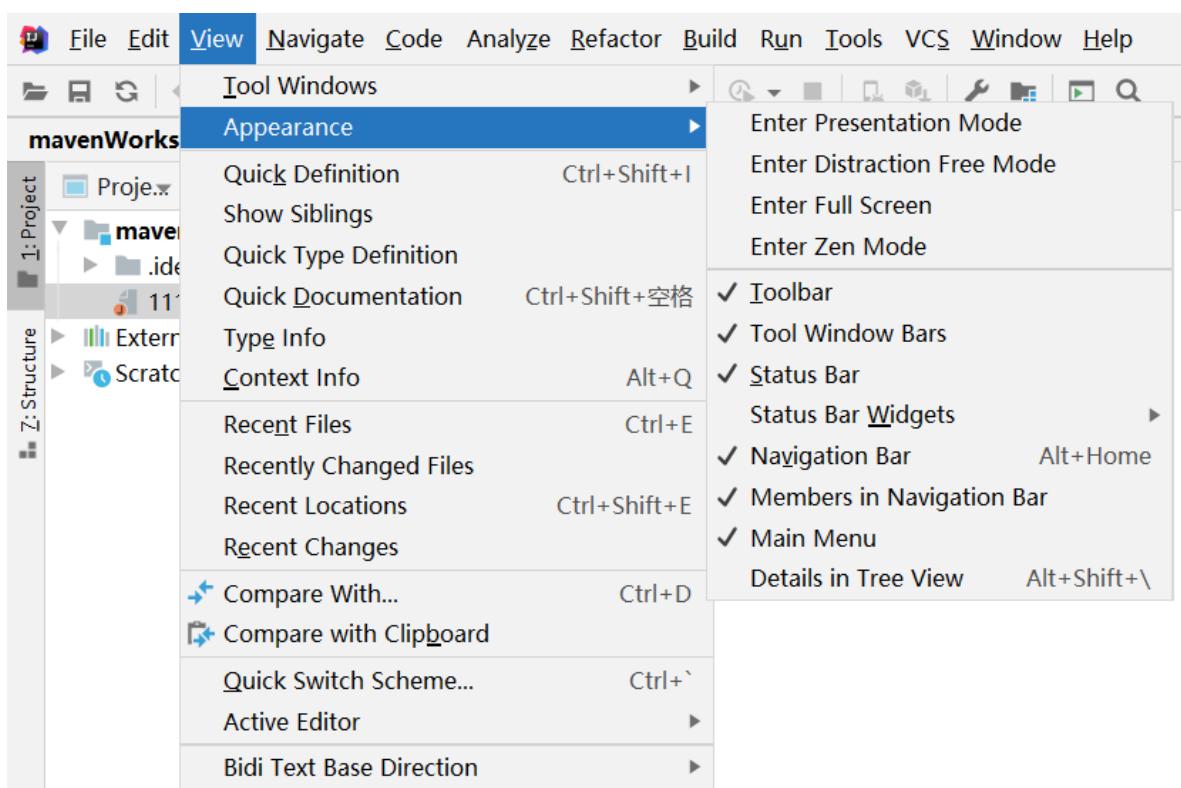
### 5.2 system 目录

system 目录是 IntelliJ IDEA 系统文件目录，是 IntelliJ IDEA 与开发项目一个桥梁目录，里面主要有：缓存、索引、容器文件输出等等，虽然不是最重要目录，但也是最不可或缺的目录之一。比如：

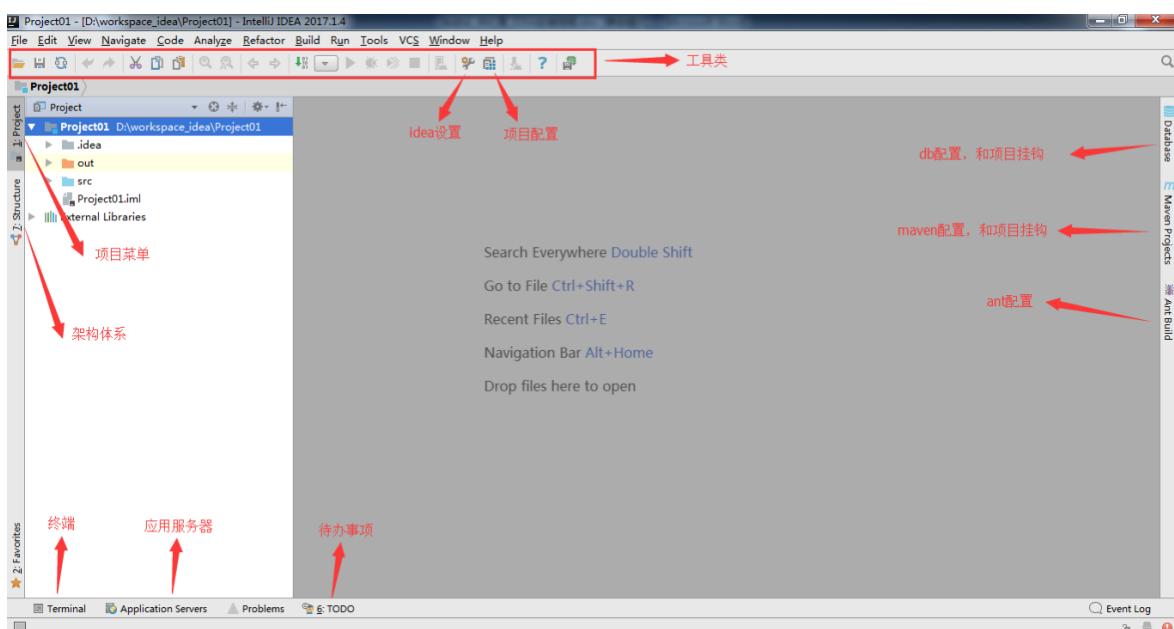


# # IDEA工程

## 设置视图



## 工程界面



- src: 存放源码

- .idea 和 [项目名].iml: IDEA工程特有

## 模块

1. 在 Eclipse 中我们有 Workspace (工作空间) 和 Project (工程) 的概念，在 IDEA 中只有 Project (工程) 和 Module (模块) 的概念。这里的对应关系为：

**IDEA 官网说明:**

An Eclipse workspace is similar to a project in IntelliJ IDEA

An Eclipse project maps to a module in IntelliJ IDEA

**翻译:**

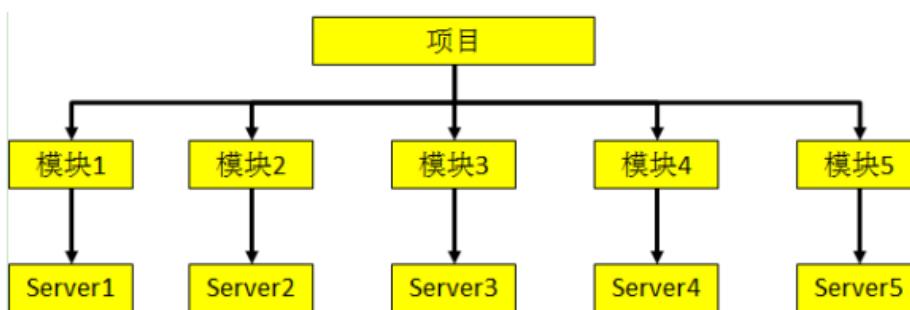
Eclipse 中 workspace 相当于 IDEA 中的 Project

Eclipse 中 Project 相当于 IDEA 中的 Module

这个地方刚开始用的时候会很容易理不清它们之间的关系。

2. 从 Eclipse 转过来的人总是下意识地要在同一个窗口管理 n 个项目，这在 IntelliJ IDEA 是无法做到的。IntelliJ IDEA 提供的解决方案是打开多个项目实例，即打开多个项目窗口。即：一个 Project 打开一个 Window 窗口。

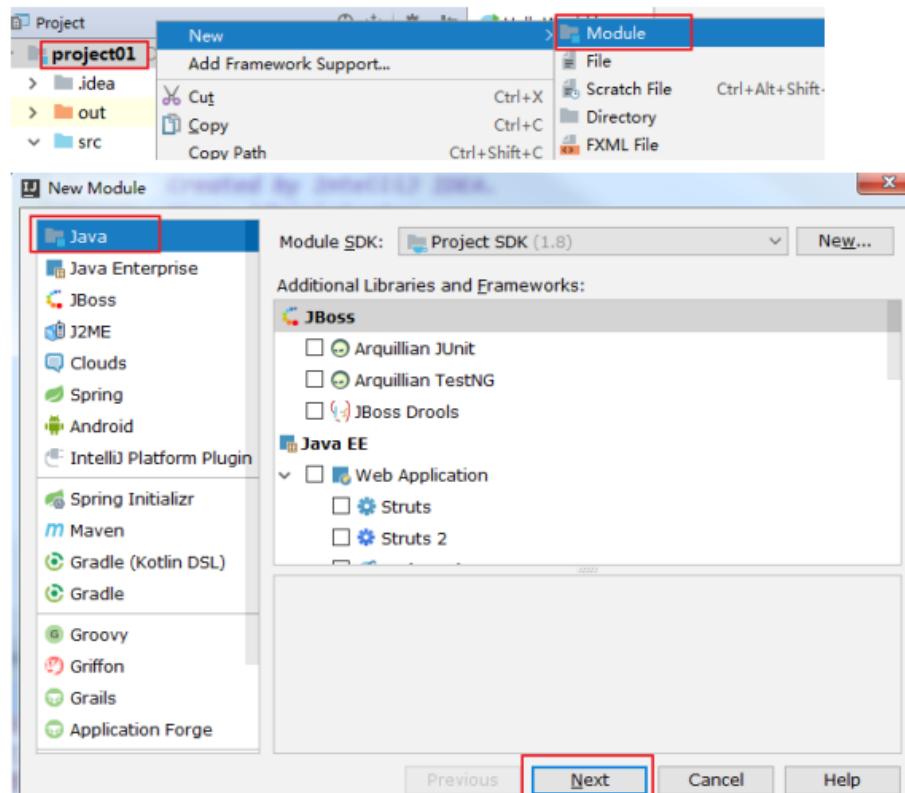
3. 在 IntelliJ IDEA 中 Project 是最顶级的级别，次级别是 Module。一个 Project 可以有多个 Module。目前主流的大型项目都是分布式部署的，结构都是类似这种多 Module 结构。



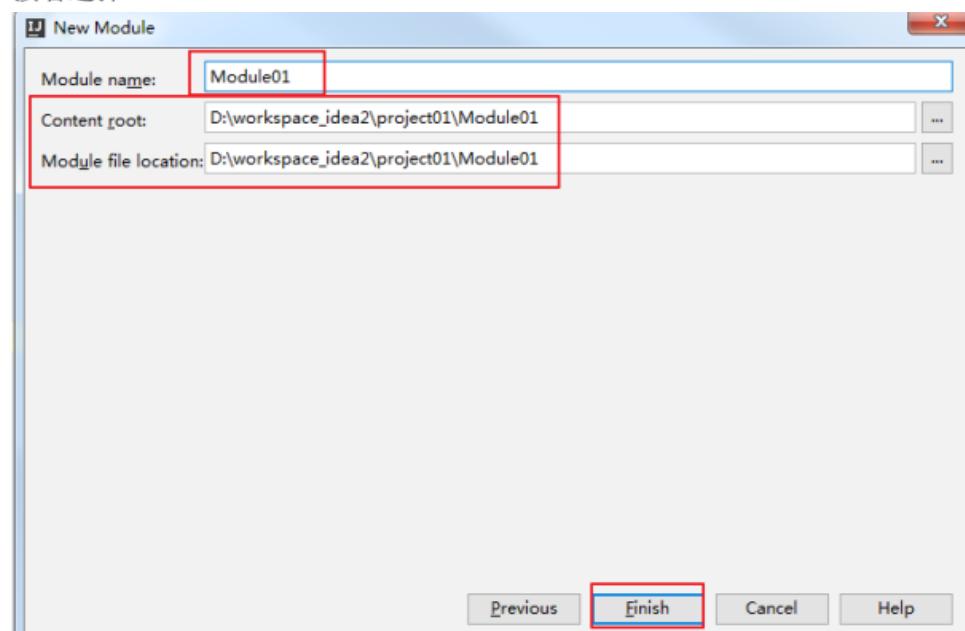
这类项目一般是这样划分的，比如：core Module、web Module、plugin Module、solr Module 等等，模块之间彼此可以相互依赖。通过这些 Module 的命名也可以看出，他们之间都是处于同一个项目业务下的模块，彼此之间是有不可分割的业务关系的。举例：

**IDEA中创建Module**

下面，我们演示如何创建 Module:



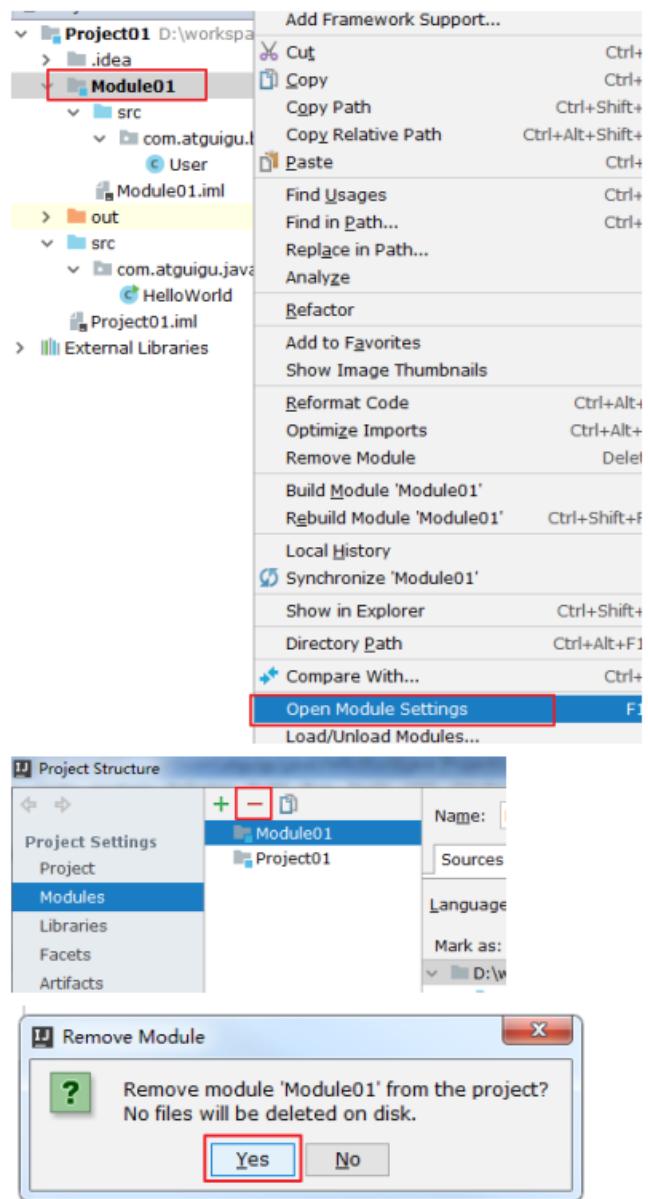
接着选择 Next:



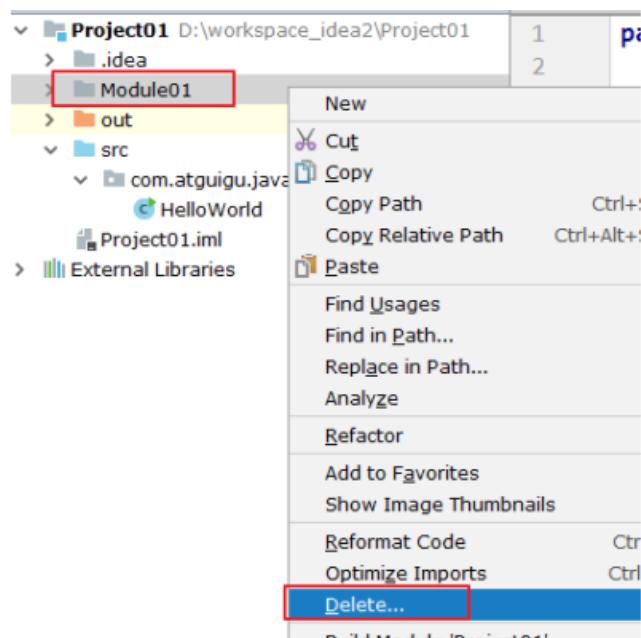
之后，我们可以在 Module 的 src 里写代码，此时 Project 工程下的 src 就没什么用了。可以删掉。

## 删除Module

1. 将Module与Project解除关联



## 2. 删除Module



此时的删除，会从硬盘上将此 module 删除掉。

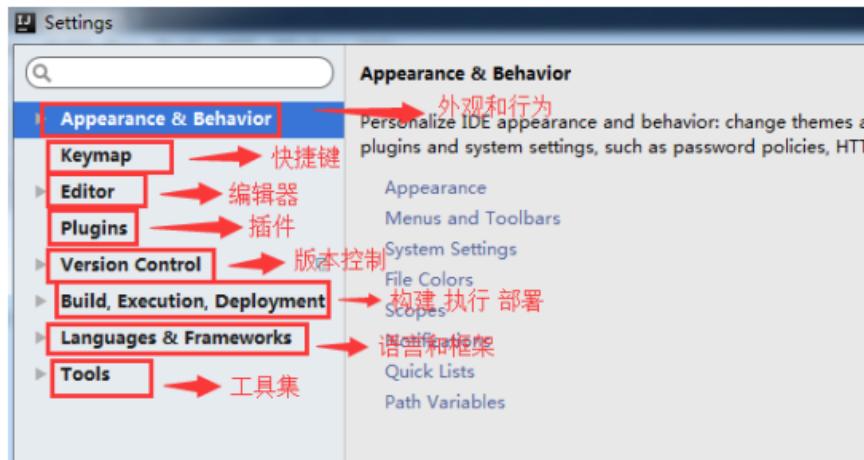
## 查看项目配置



# # 编辑器配置

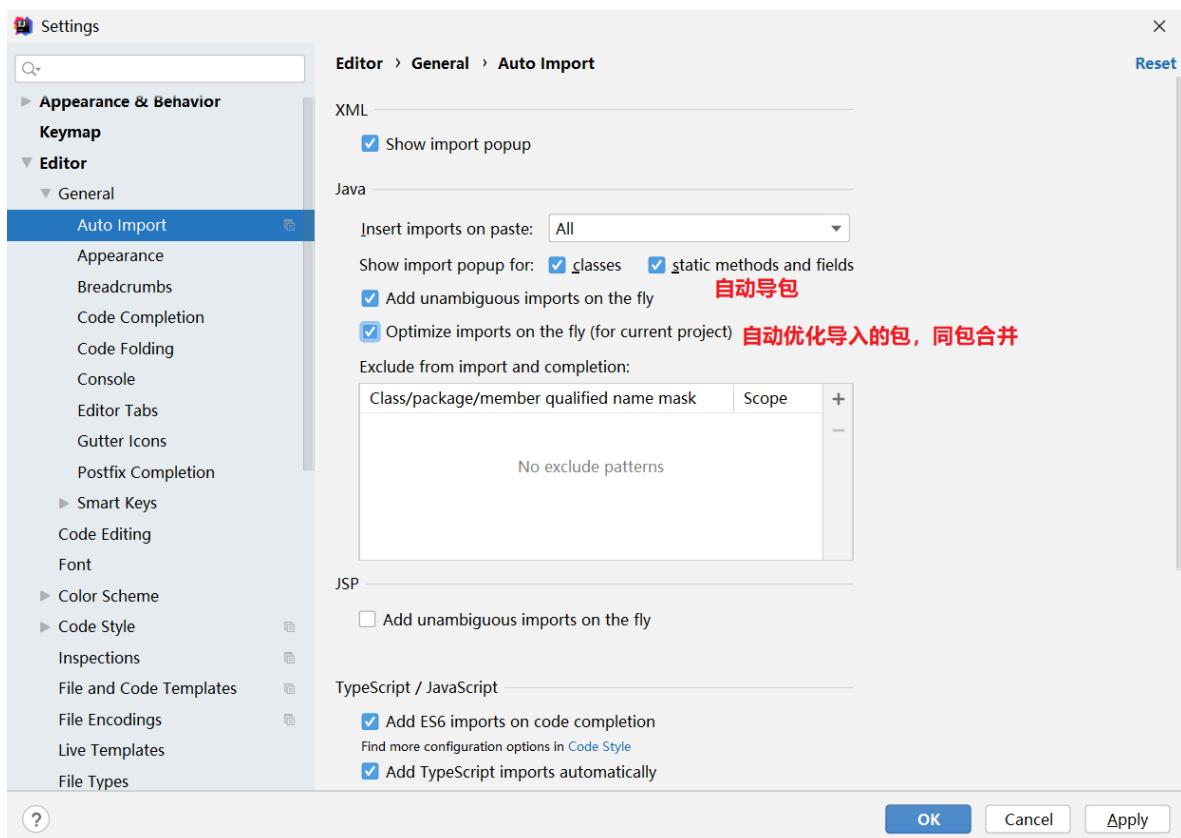
设置界面

目录结构如下：

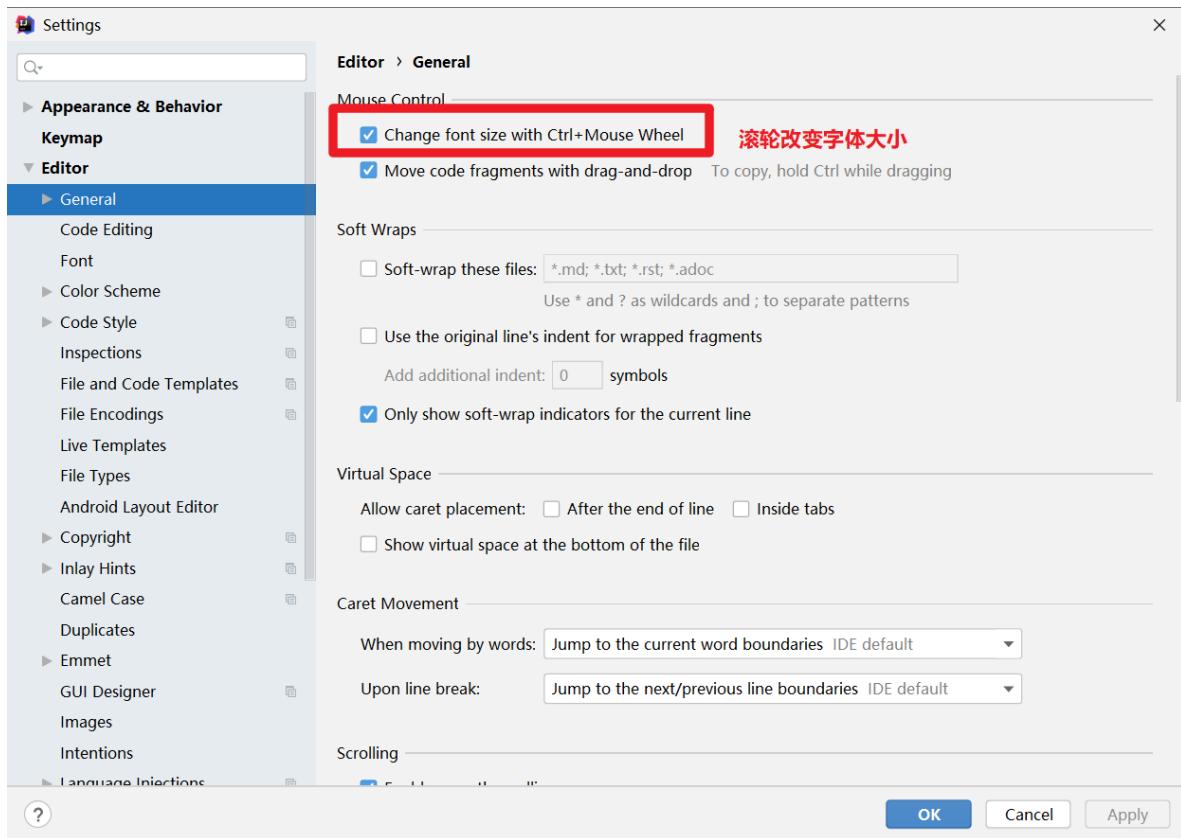


## Editor—General

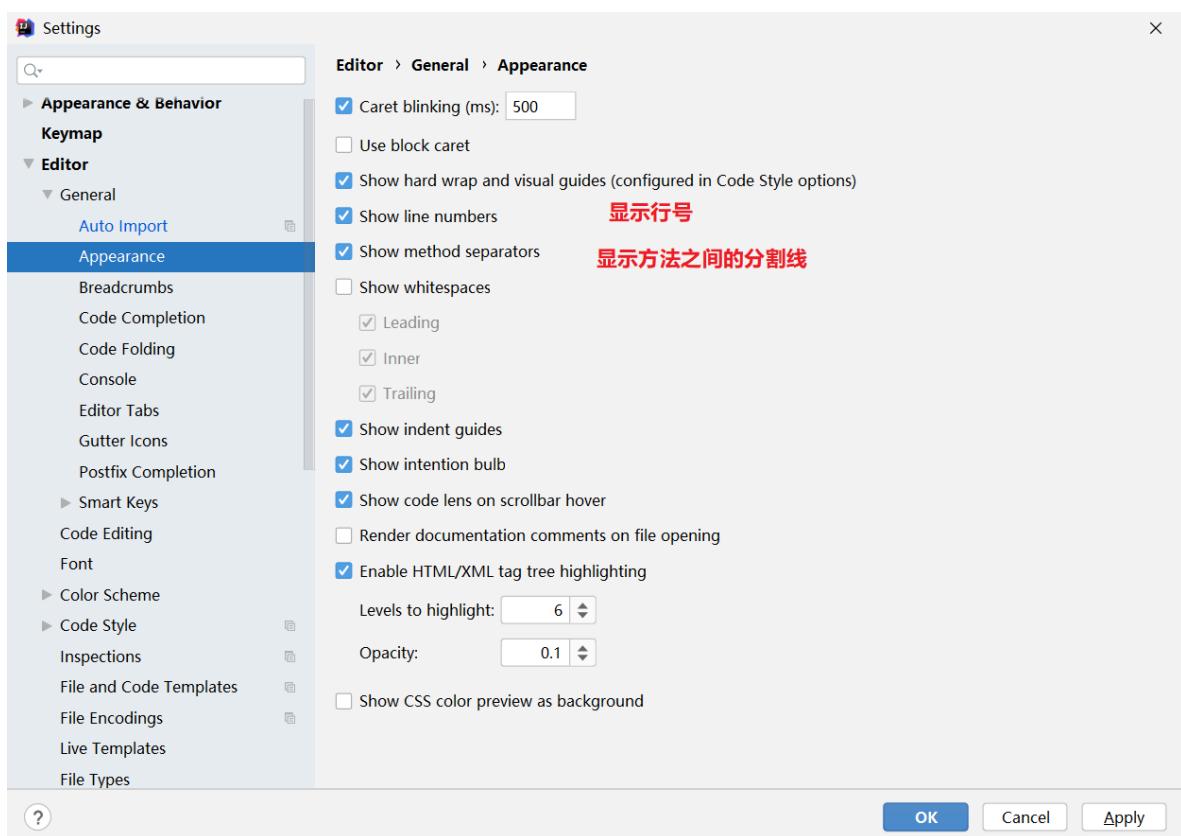
### 设置自动导包功能



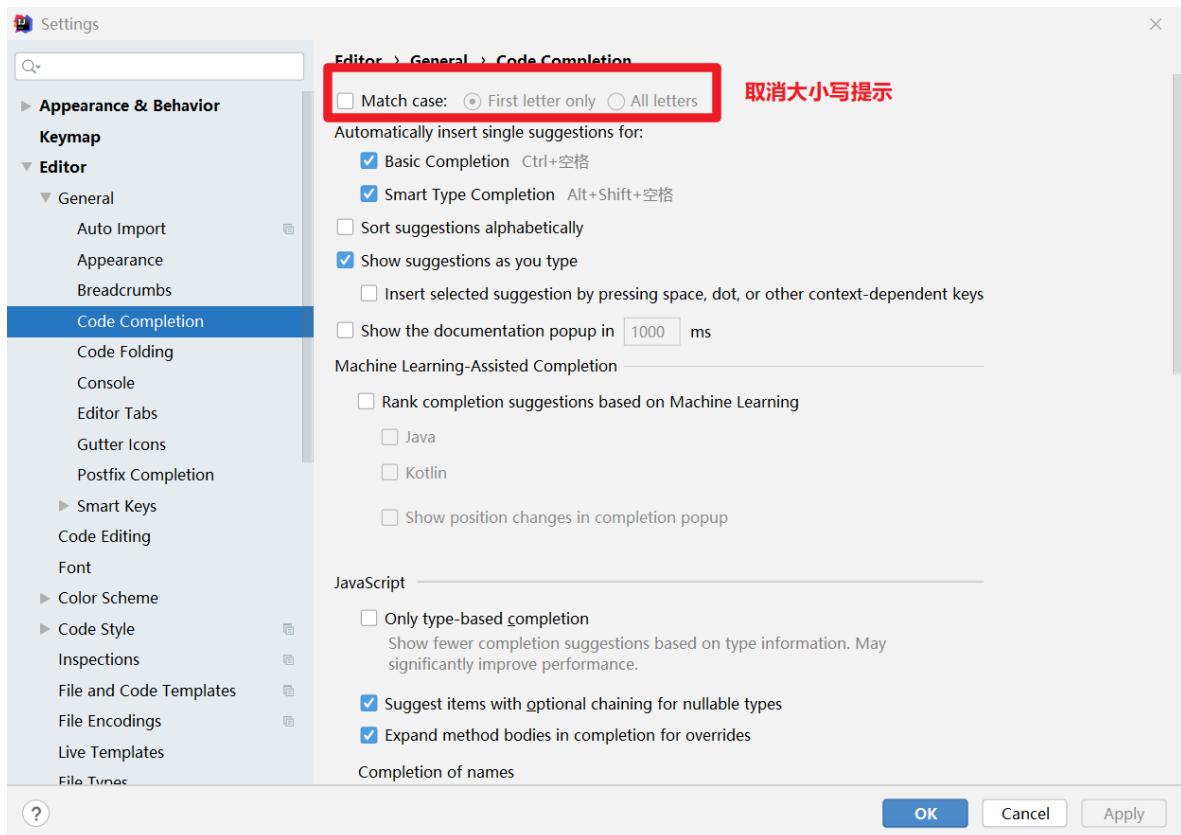
## 设置鼠标滚轮修改字体大小



## 显示行号和方法间的分割线

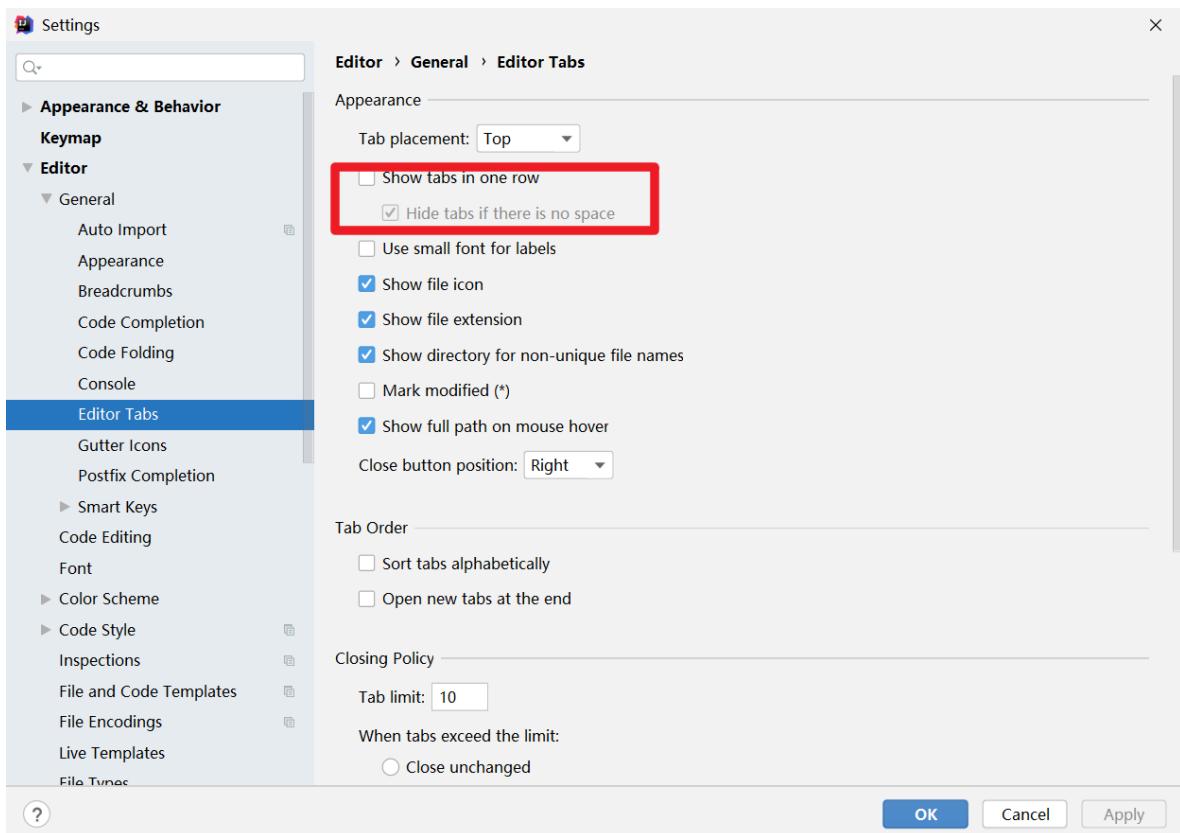


## 忽略大小写提示



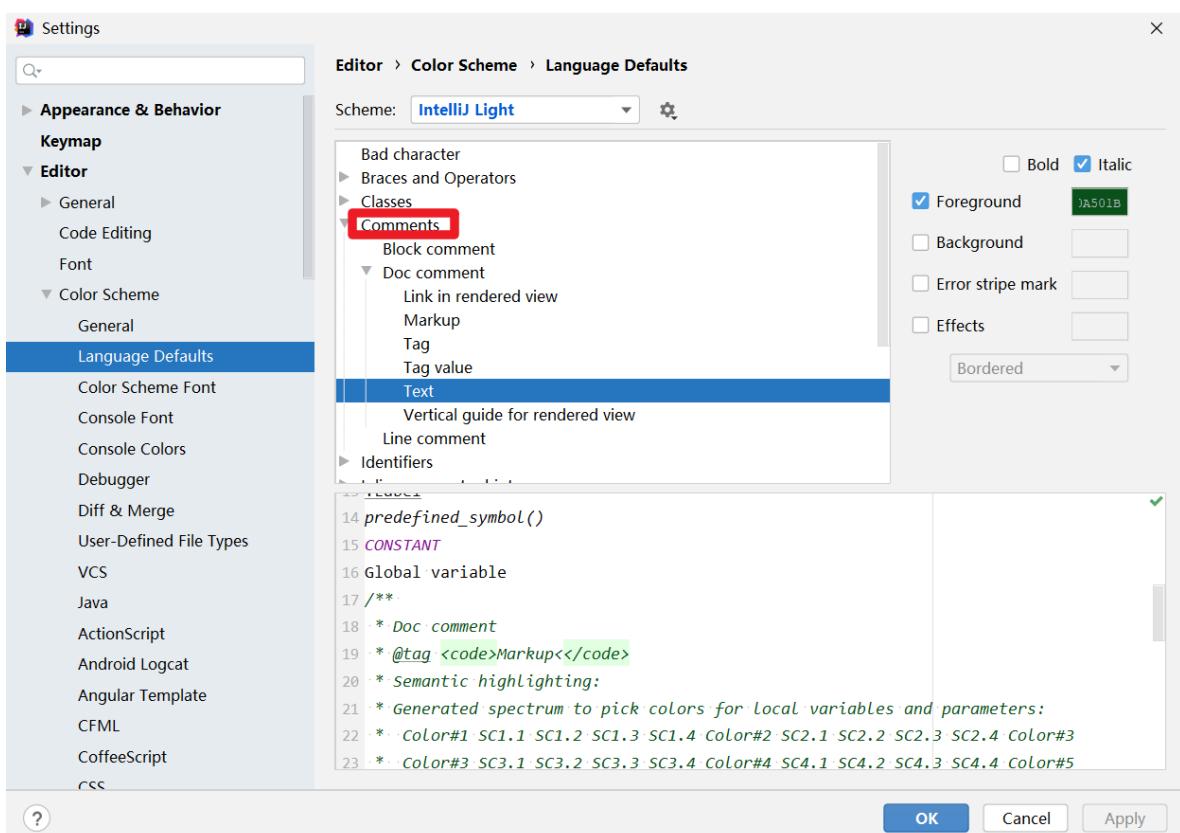
- `StringBuffer` & `StringBuffer`

## 设置单行显示tabs



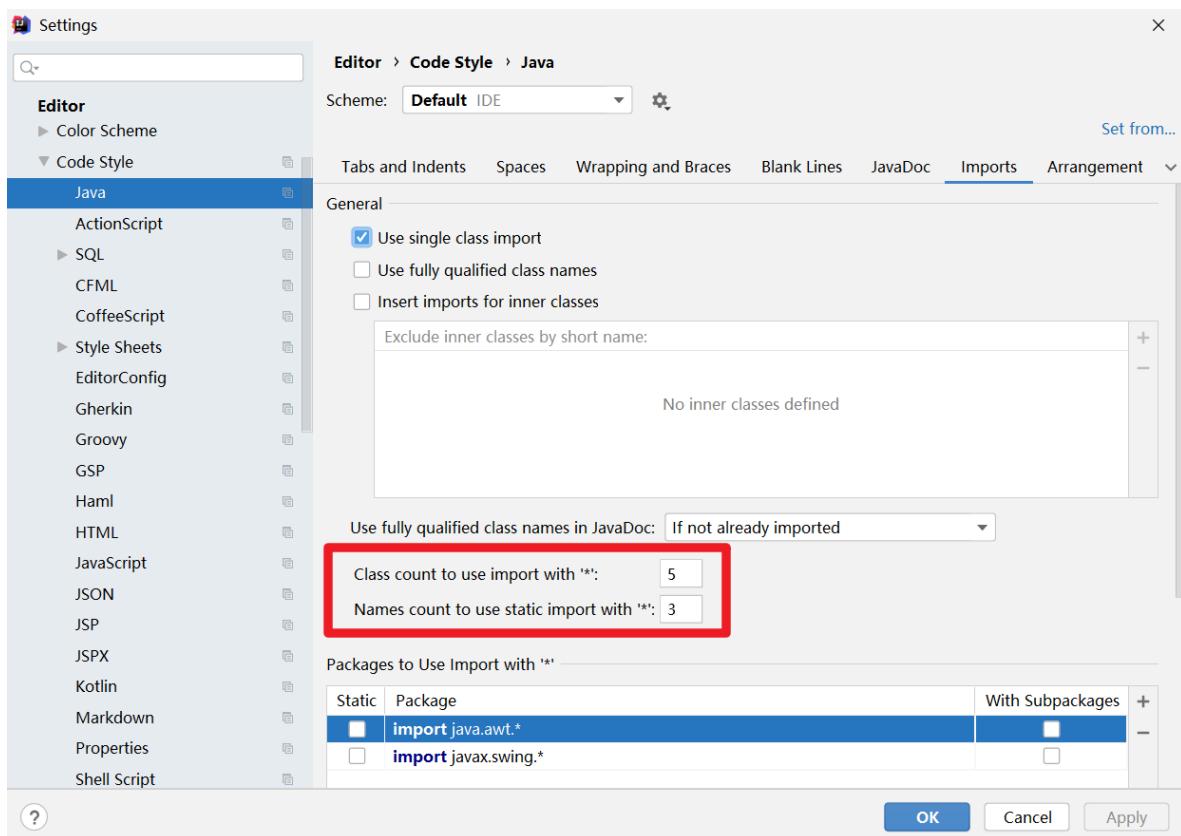
## Editor——Color Scheme

修改代码中注释的字体颜色



- Doc Comment-Text: 文档注释
- Block Comment: 多行注释
- Line Comment: 单行注释

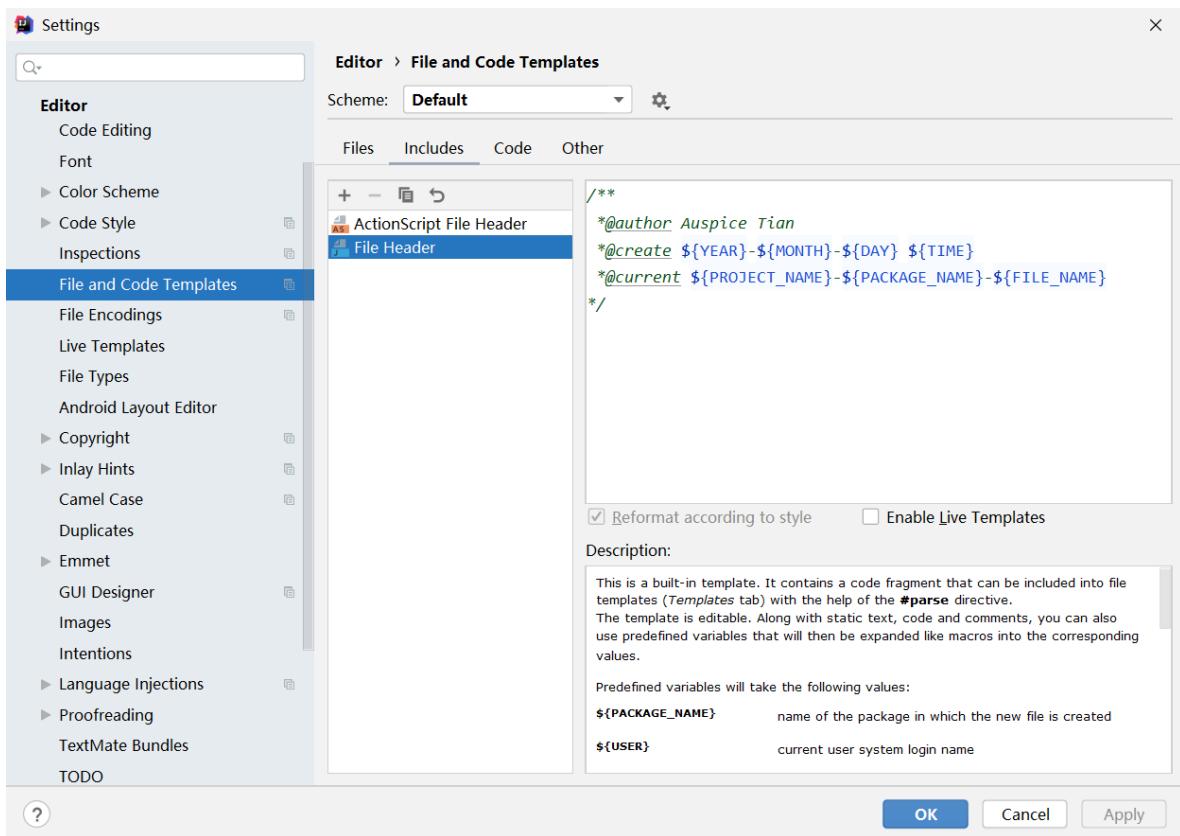
## Editor——Code Style



- 设置超过指定 import 个数，改为\*

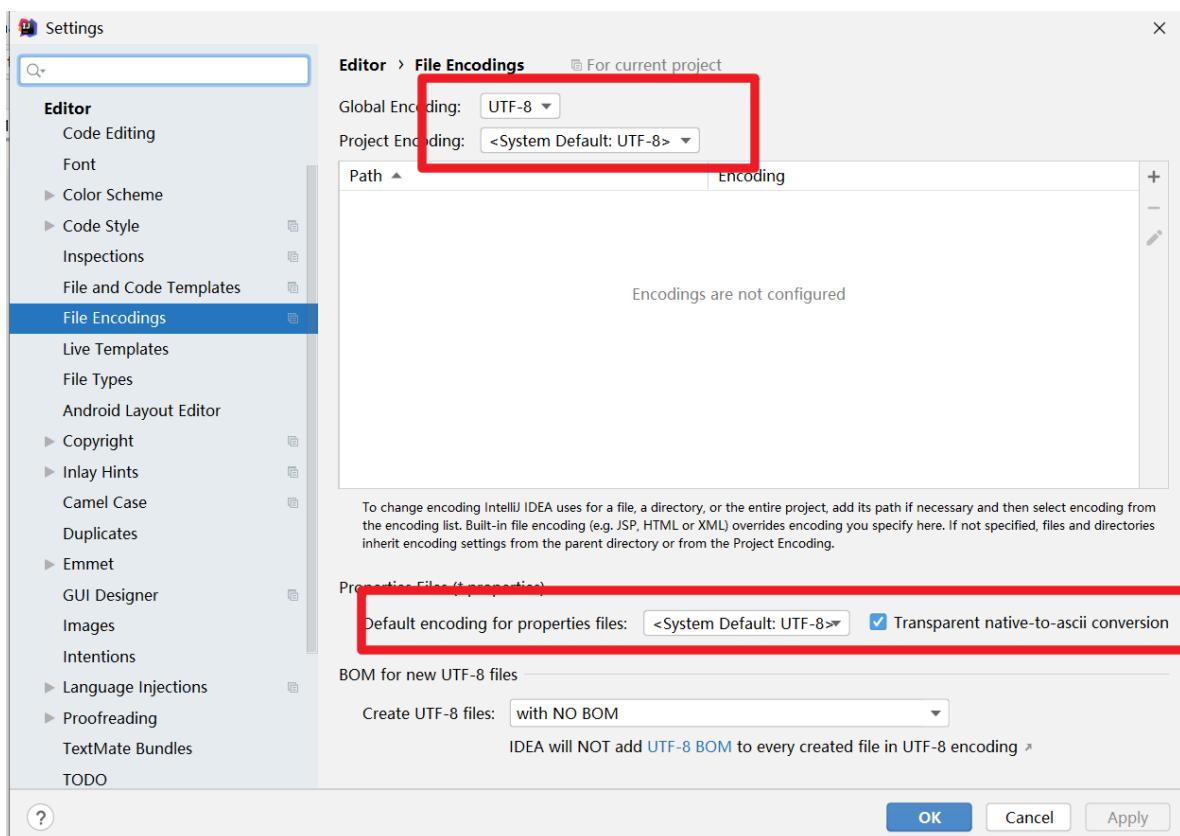
## Editor——File and Code Templates

新建文档的头部注释信息



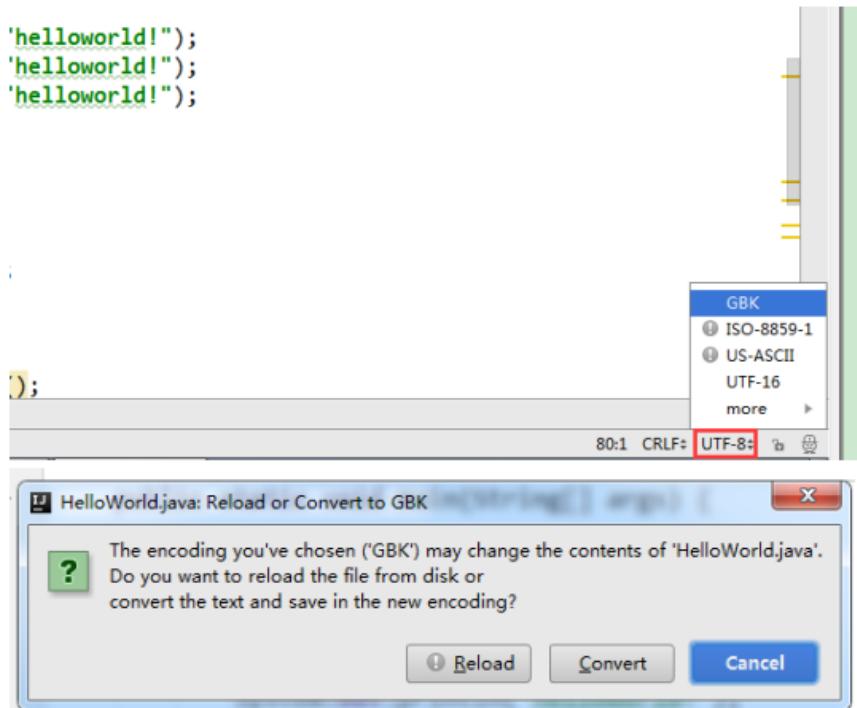
## Editor——File Encodings

### 设置项目文件编码



- Transparent native-to-ascii conversion 主要用于转换 ascii，一般都要勾选，不然 Properties 文件中的注释显示的都不会是中文

## H5 对单一文件的编码修改

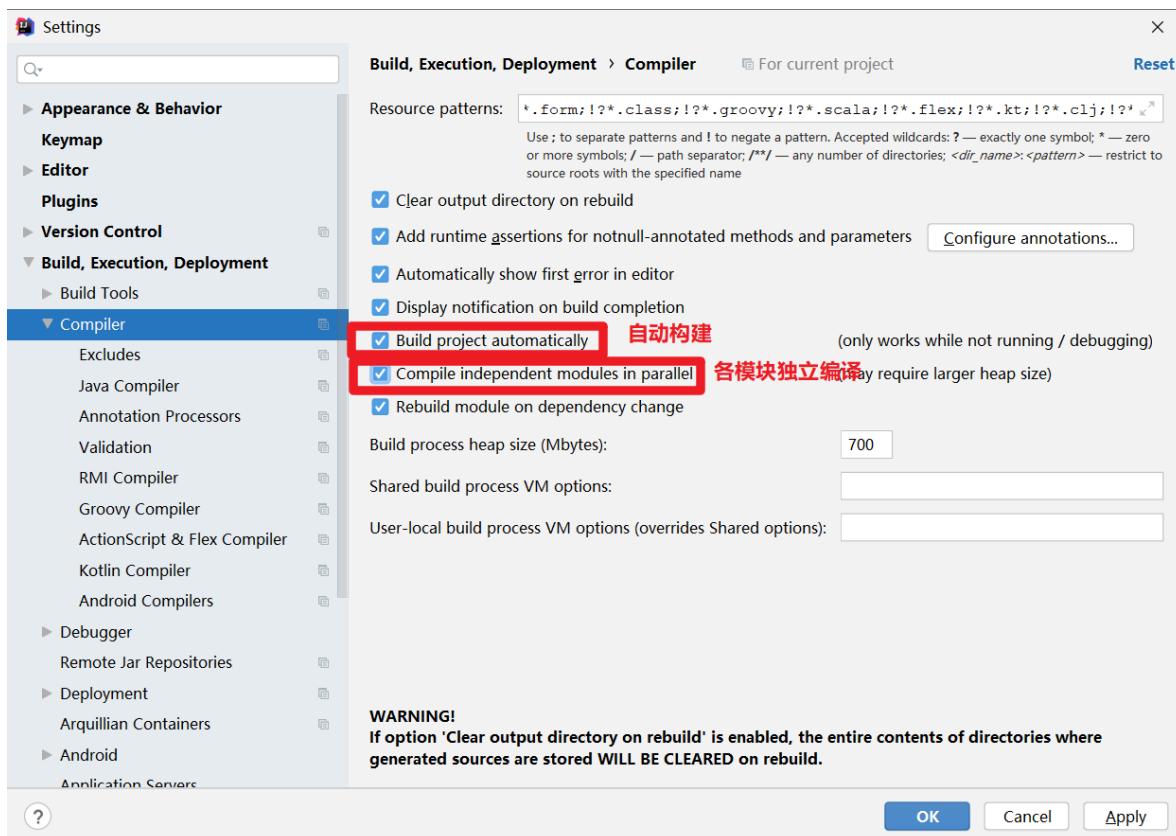


对单独文件的编码修改还可以点击右下角的编码设置区。如果代码内容中包含中文，则会弹出如上的操作选择。其中：

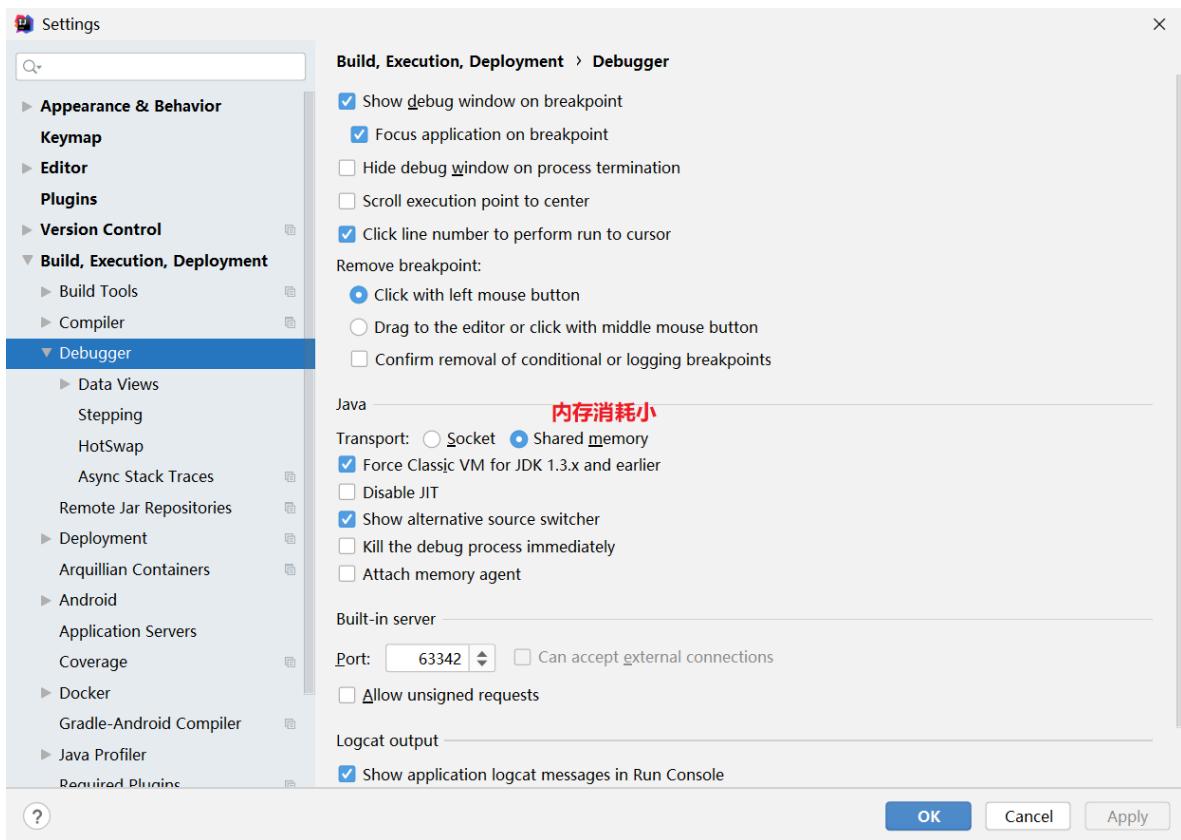
- ① Reload 表示使用新编码重新加载，新编码不会保存到文件中，重新打开此文件，旧编码是什么依旧还是什么。
- ② Convert 表示使用新编码进行转换，新编码会保存到文件中，重新打开此文件，新编码是什么则是什么。
- ③含有中文的代码文件，Convert 之后可能会使中文变成乱码，所以在转换成请做好备份，不然可能出现转换过程变成乱码，无法还原。

# # 编译配置——Build,Execution,Deployment

## Compiler——设置自动编译



## 断点调试



## 2. 常用断点调试快捷键

	step over 进入下一步，如果当前行断点是一个方法，则不进入当前方法体内
	step into 进入下一步，如果当前行断点是一个方法，则进入当前方法体内
	force step into 进入下一步，如果当前行断点是一个方法，则进入当前方法体内
	step out 跳出
	resume program 恢复程序运行，但如果该断点下面代码还有断点则停在下一个断点上
	stop 停止
	mute breakpoints 点中，使得所有的断点失效
	view breakpoints 查看所有断点

## 3. 条件断点

### 说明：

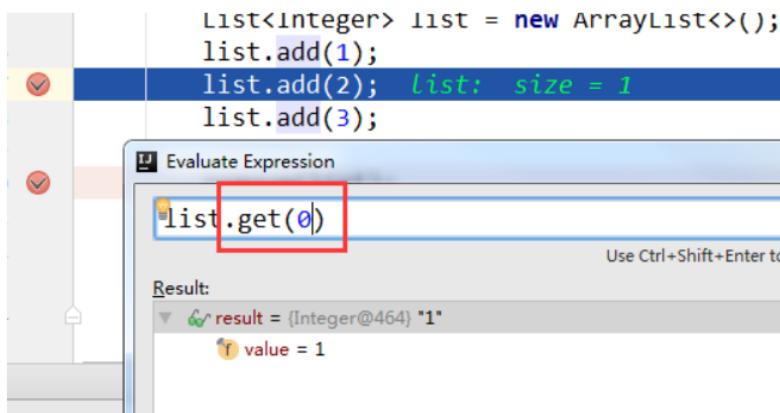
调试的时候，在循环里增加条件判断，可以极大的提高效率，心情也能愉悦。

### 具体操作：

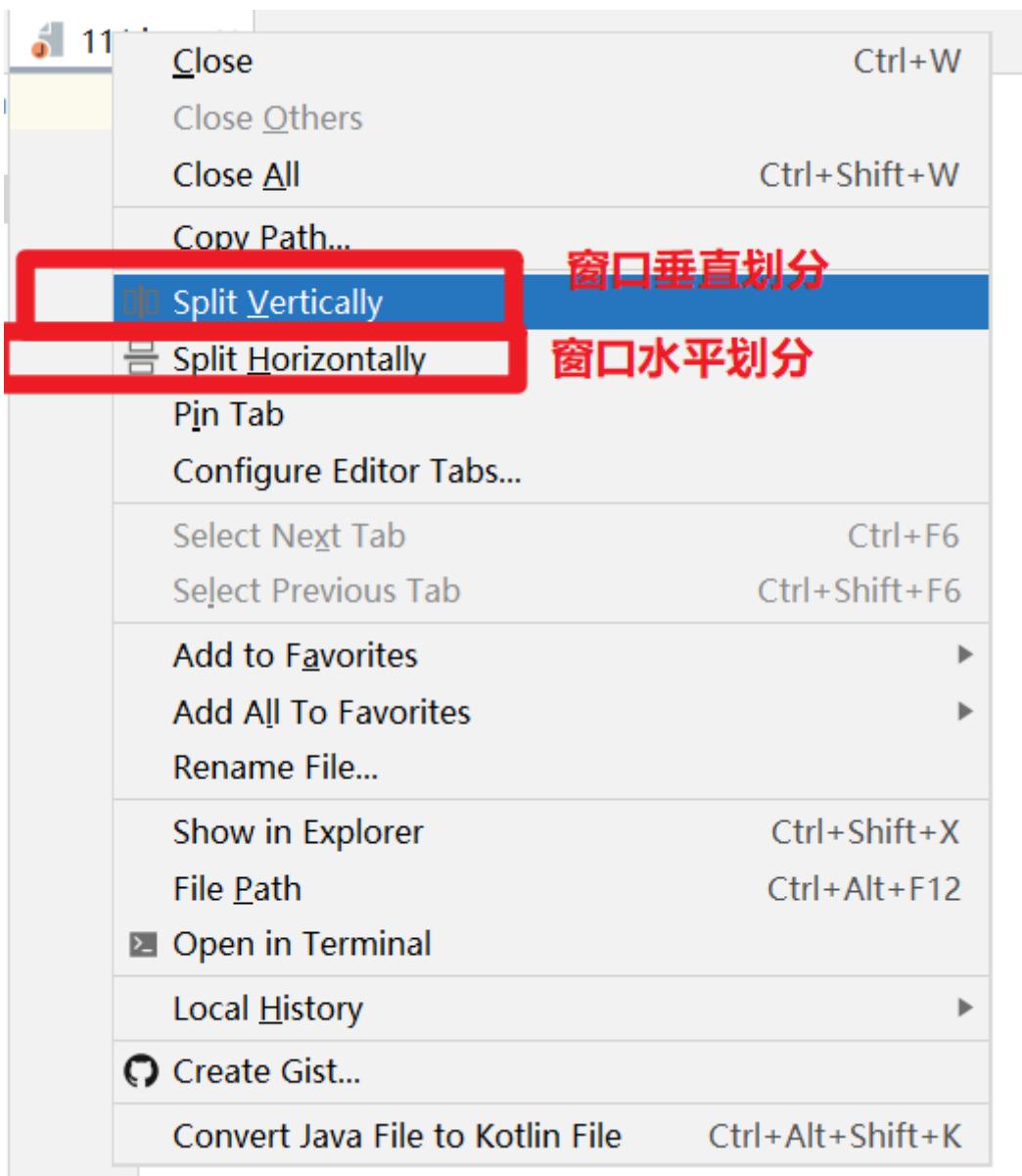
在断点处右击调出条件断点。可以在满足某个条件下，实施断点。

### 查看表达式的值(Ctrl + u)：

选择行，ctrl + u。还可以在查看框中输入编写代码时的其他方法：



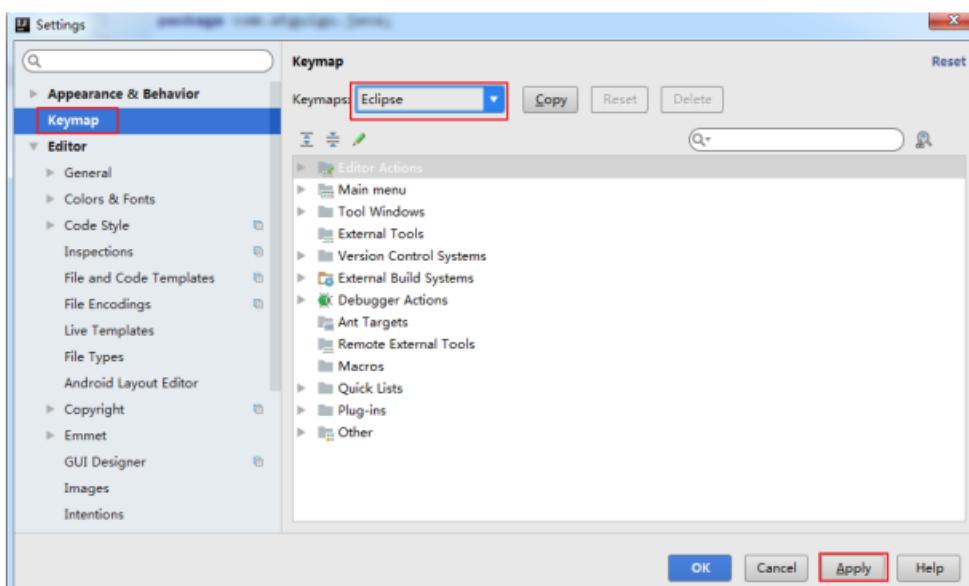
## 窗口划分方式



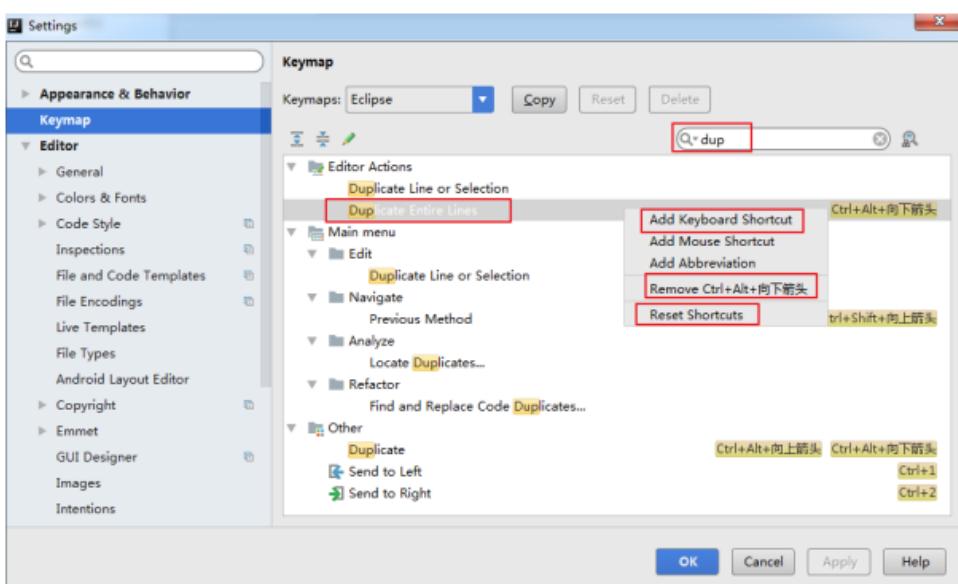
## # Keymap

---

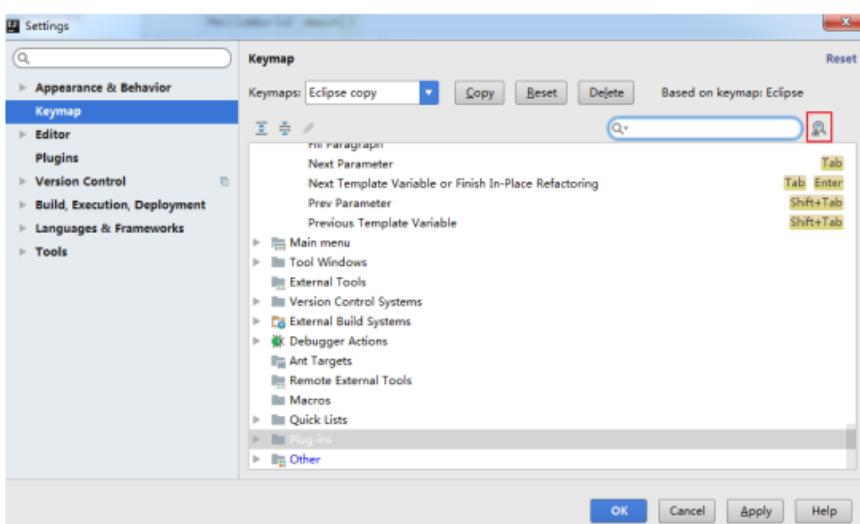
## 1. 设置快捷键为 Eclipse 的快捷键



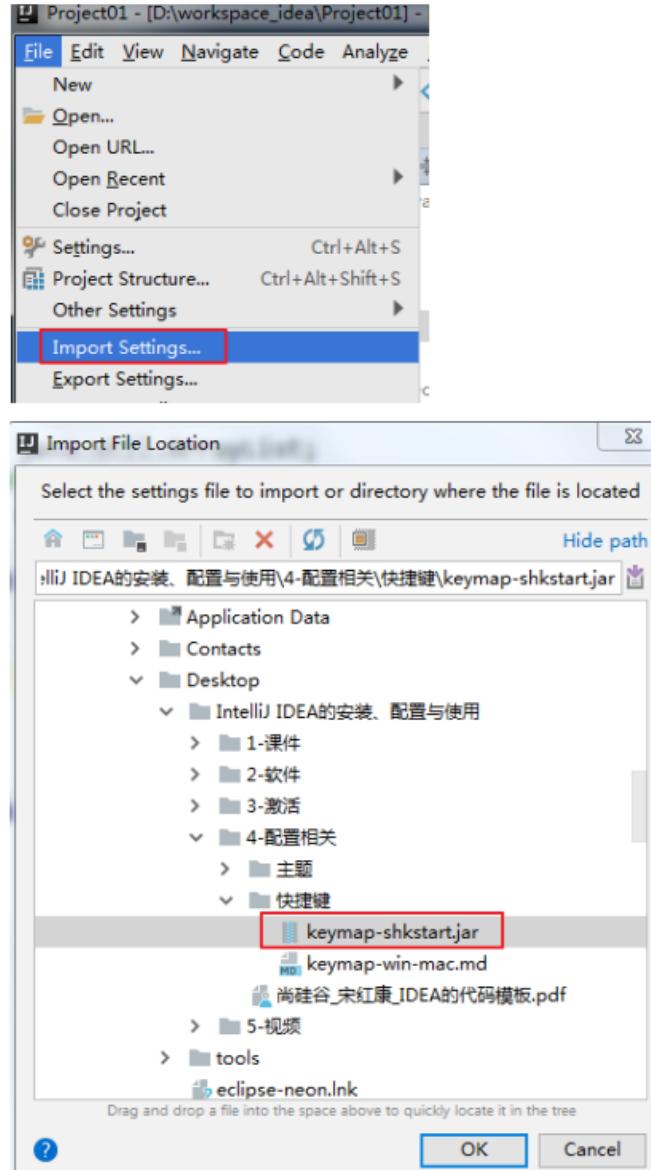
## 2. 通过快捷键功能修改快捷键设置



## 3. 通过指定快捷键，查看或修改其功能



#### 4.导入已有的设置



点击 OK 之后，重启 IDEA 即可。

## 快捷键

### 尚硅谷·宋红康 设置版

1	执行(run)	alt+r
2	提示补全 (Class Name Completion)	alt+/
3	单行注释	ctrl + /
4	多行注释	ctrl + shift + /
5	向下复制一行 (Duplicate Lines)	ctrl+alt+down
6	删除一行或选中行 (delete line)	ctrl+d
7	向下移动行(move statement down)	alt+down
8	向上移动行(move statement up)	alt+up
9	向下开始新的一行(start new line)	shift+enter
10	向上开始新的一行 (Start New Line before current)	ctrl+shift+enter
11	如何查看源码 (class)	ctrl + 选中指定的结构 或 ctrl + shift + t
12	万能解错/生成返回值变量	alt + enter
13	退回到前一个编辑的页面 (back)	alt + left
14	进入到下一个编辑的页面(针对于上条) (forward)	alt + right
15	查看继承关系(type hierarchy)	F4
16	格式化代码(reformat code)	ctrl+shift+F
17	提示方法参数类型(Parameter Info)	ctrl+alt+ /
18	复制代码	ctrl + c
19	撤销	ctrl + z
20	反撤销	ctrl + y
21	剪切	ctrl + x
22	粘贴	ctrl + v
23	保存	ctrl + s
24	全选	ctrl + a
25	选中数行, 整体往后移动	tab
26	选中数行, 整体往前移动	shift + tab
27	查看类的结构: 类似于 eclipse 的 outline	ctrl+o
28	重构: 修改变量名与方法名(rename)	alt+shift+r
29	大写转小写/小写转大写(toggle case)	ctrl+shift+y

30	生成构造器/get/set/toString	alt +shift + s
31	查看文档说明(quick documentation)	F2
32	收起所有的方法(collapse all)	alt + shift + c
33	打开所有方法(expand all)	alt+shift+x
34	打开代码所在硬盘文件夹(show in explorer)	ctrl+shift+x
35	生成 try-catch 等(surround with)	alt+shift+z
36	局部变量抽取为成员变量(introduce field)	alt+shift+f
37	查找/替换(当前)	ctrl+f
38	查找(全局)	ctrl+h
39	查找文件	double Shift
40	查看类的继承结构图(Show UML Diagram)	ctrl + shift + u
41	查看方法的多层次重写结构(method hierarchy)	ctrl+alt+h
42	添加到收藏(add to favorites)	ctrl+alt+f
43	抽取方法(Extract Method)	alt+shift+m
44	打开最近修改的文件(Recently Files)	ctrl+E

45	关闭当前打开的代码栏(close)	ctrl + w
46	关闭打开的所有代码栏(close all)	ctrl + shift + w
47	快速搜索类中的错误(next highlighted error)	ctrl + shift + q
48	选择要粘贴的内容>Show in Explorer)	ctrl+shift+v
49	查找方法在哪里被调用(Call Hierarchy)	ctrl+shift+h

## # 模板

---

### 1. Live Templates(实时代码模板)功能介绍

配置一些常用代码字母缩写，在输入简写时可以出现你预定义的固定模式的代码，使得开发效率大大提高，同时也可以增加个性化。最简单的例子就是在 Java 中输入 sout 会出现 System.out.println();

### 举例

## 2.1 psvm：可生成 main 方法

## 2.2 sout : System.out.println() 快捷输出

类似的:

```
soutp=System.out.println("方法形参名 = " + 形参名);  
soutv=System.out.println("变量名 = " + 变量);  
soutm=System.out.println("当前类名.当前方法");  
"abc".sout => System.out.println("abc");
```

## 2.3 fori：可生成 for 循环

类似的:

iter: 可生成增强 for 循环  
itar: 可生成普通 for 循环

## 2.4 list.for：可生成集合 list 的 for 循环

```
List<String> list = new ArrayList<String>();
```

输入: list.for 即可输出

```
for(String s:list){  
}
```

又如: list.fori 或 list.forr

## 2.5 ifn: 可生成 if(xxx = null)

类似的:

inn: 可生成 if(xxx != null) 或 xxx.nn 或 xxx.null

## 2.6 prsf: 可生成 private static final

类似的:

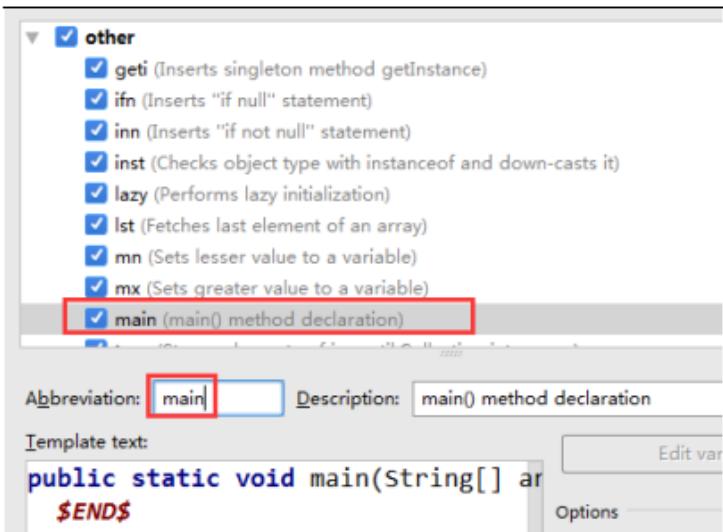
```
psf: 可生成 public static final  
psfi: 可生成 public static final int  
psfs: 可生成 public static final String
```

## 3.修改现有模板:Live Templates

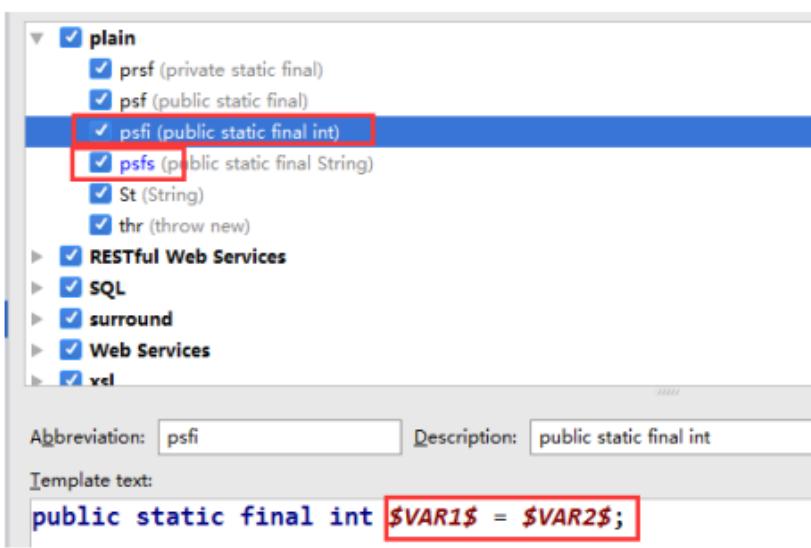
如果对于现有的模板，感觉不习惯、不适应的，可以修改：

### 修改 1:

通过调用 psvm 调用 main 方法不习惯，可以改为跟 Eclipse 一样，使用 main 调取。



## 修改 2:



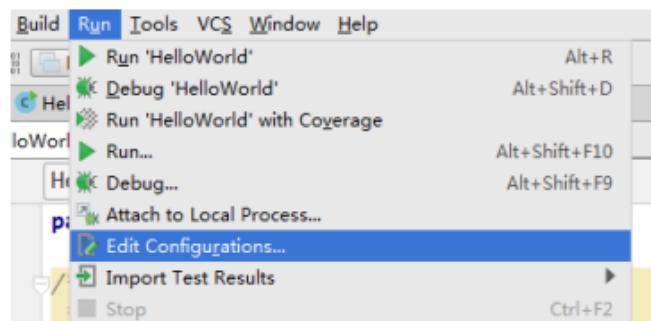
## # 静态JavaWeb和Tomcat

---

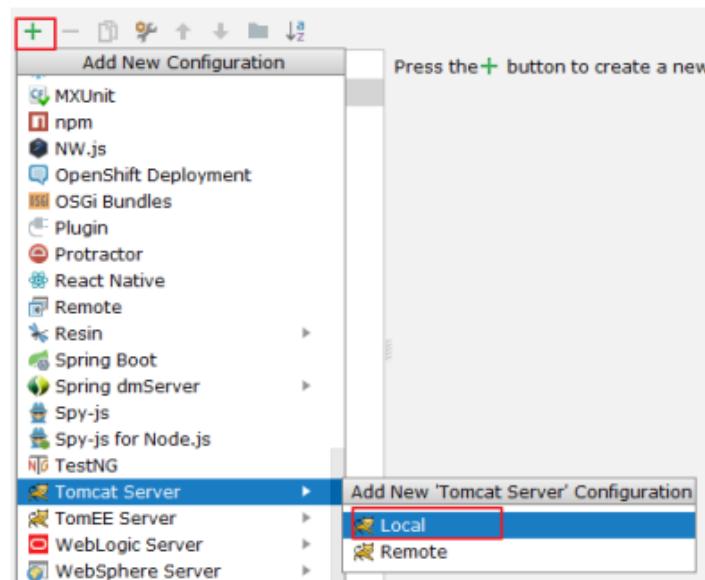
## 2.2 配置 Tomcat

在 IDEA 中配置 Tomcat 之前，需要保证已经安装并配置了 Tomcat 的环境变量。如果没有安装并配置，可以参考《尚硅谷\_宋红康\_Tomcat 快速部署.pdf》,配置完成以后，在命令行输入：**catalina run**。能够启动 tomcat，则证明安装配置成功。

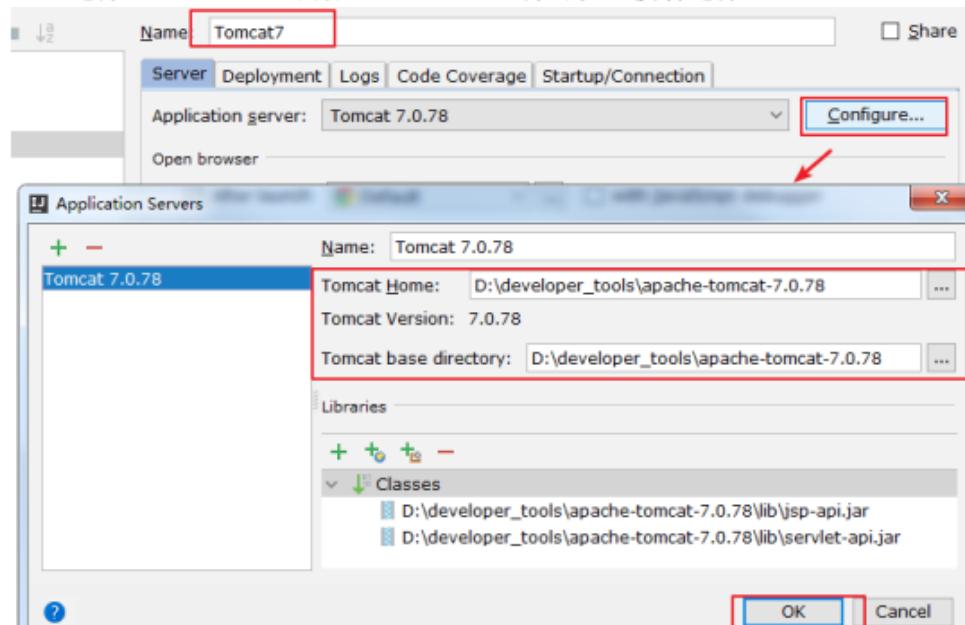
下面看如何在 IDEA 中配置：



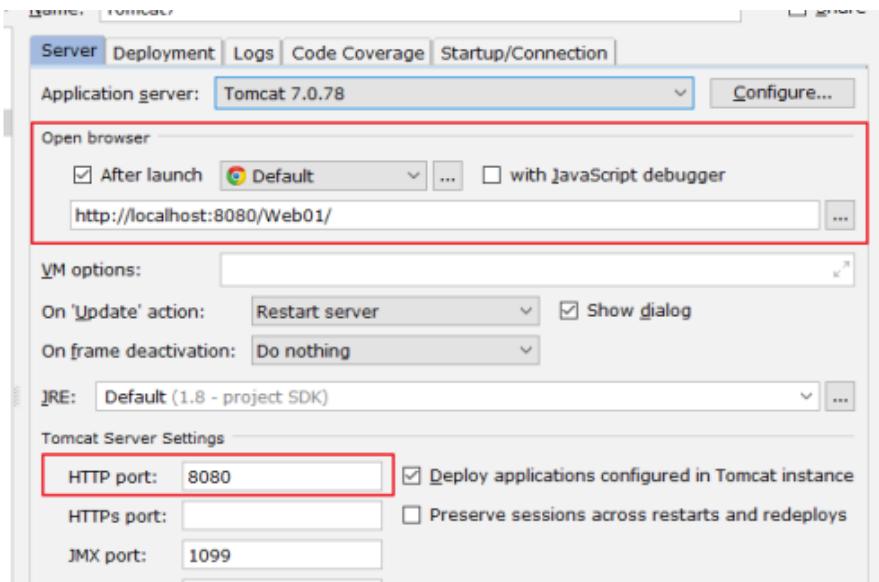
点击 Edit Configurations:



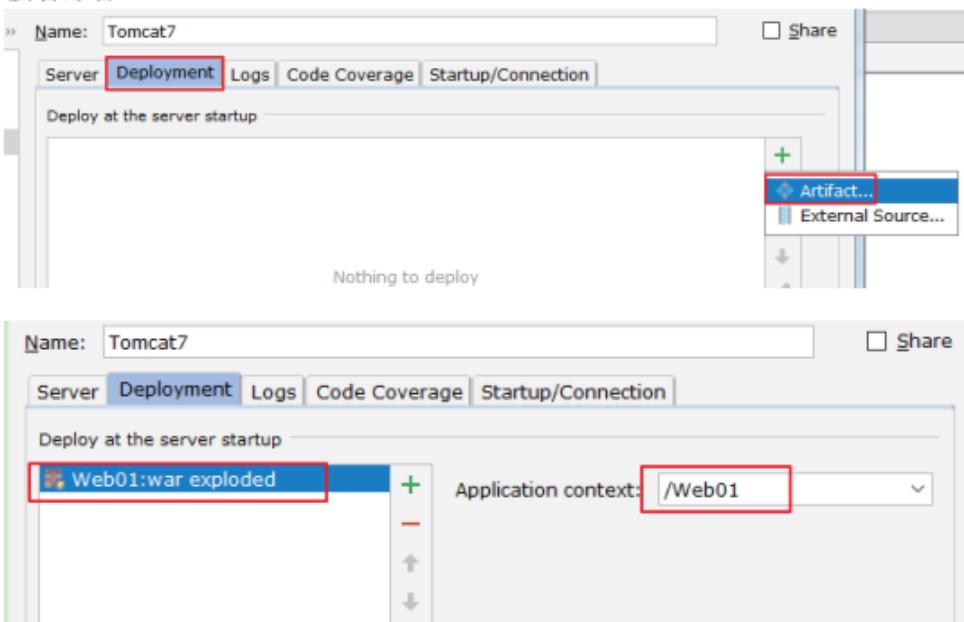
这里选择 TomEE Server 或者 Tomcat Server 都可以。接着选择 Local。



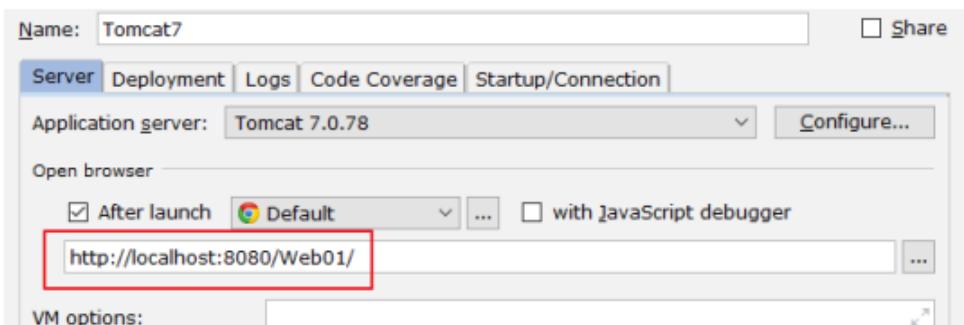
这里配置 Tomcat 的名称以及配置应用服务器的位置。根据自己 Tomcat 的安装位置决定。



其它位置使用默认值(设置要启动的浏览器以及端口号), 如上。  
接着部署:

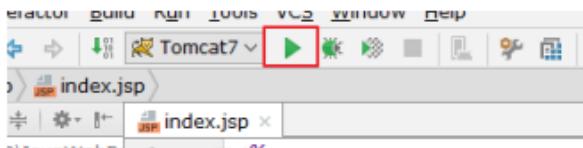


点击 OK 即可。此时:

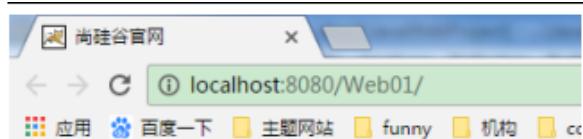


执行刚才创建的 index.jsp 即可:

执行刚才创建的 index.jsp 即可：

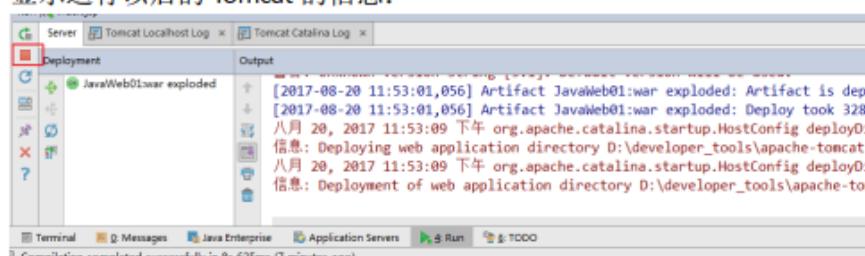


效果如下：



注意事项：

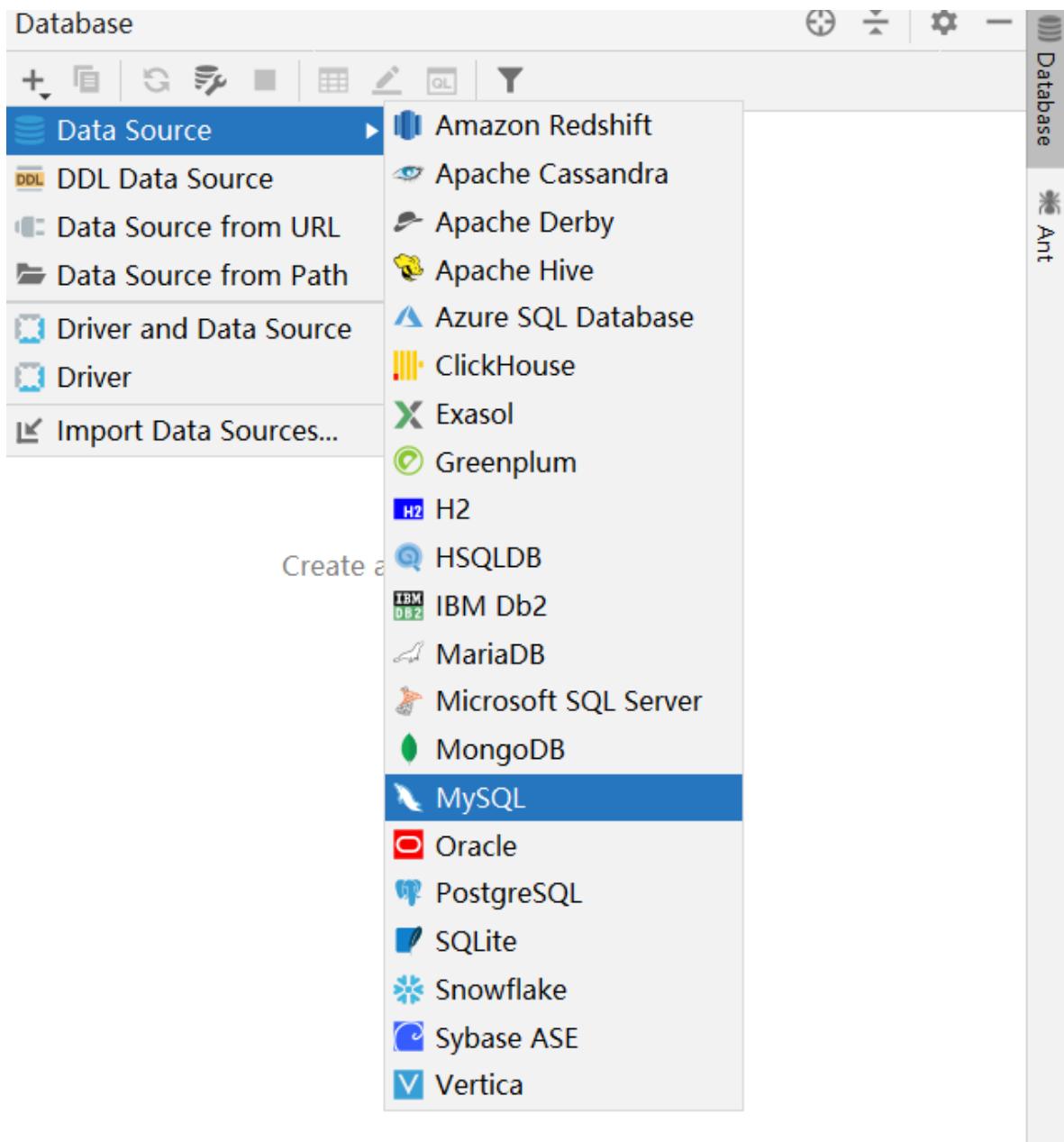
显示运行以后的 Tomcat 的信息：

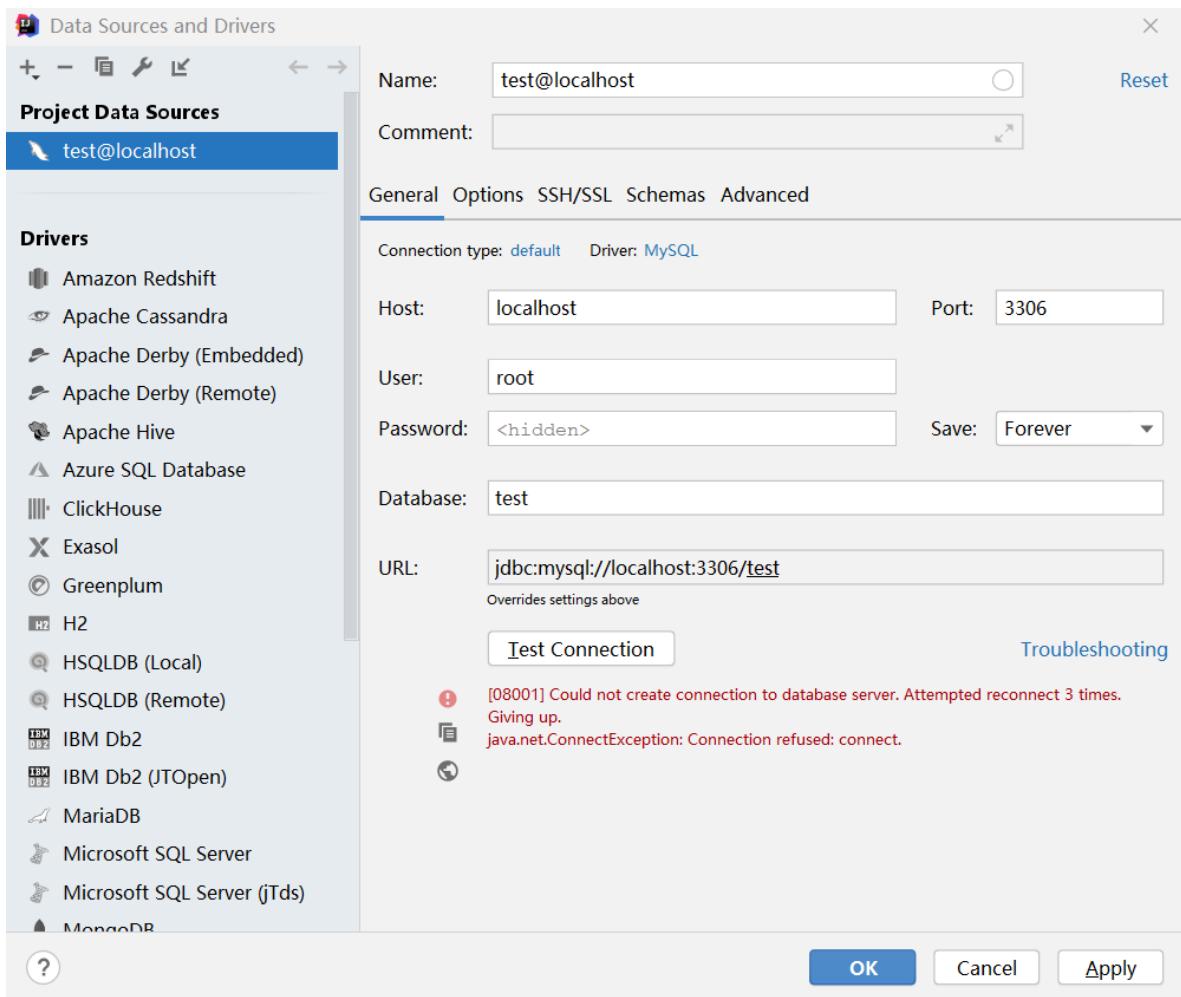


可以点击红框，刚点击完毕并不能马上关闭服务器，只是断开了与服务器的连接，稍后当停止按钮显示为灰色，才表示关闭。

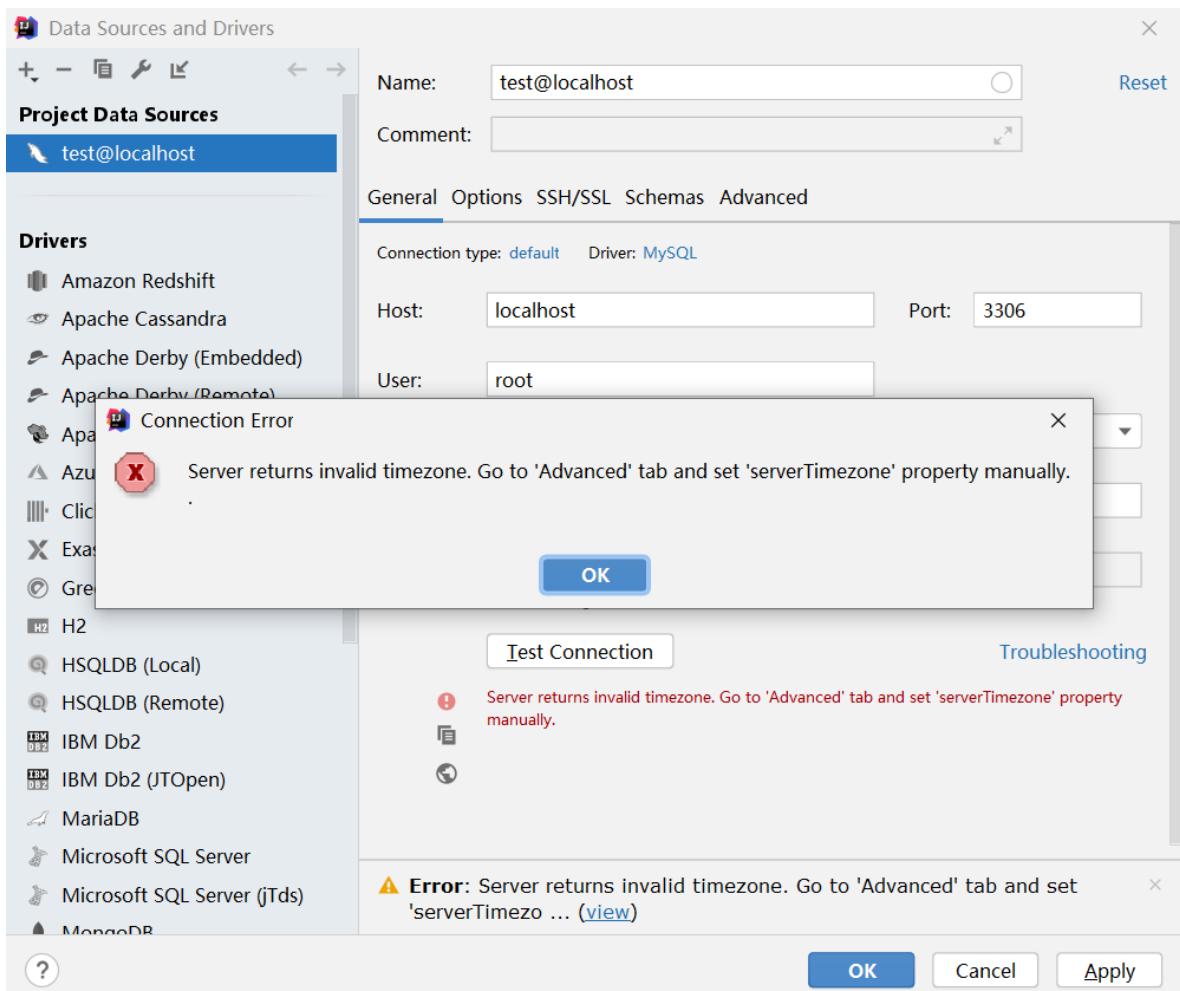
## # 关联数据库

### 启动Mysql服务





## 设置时区



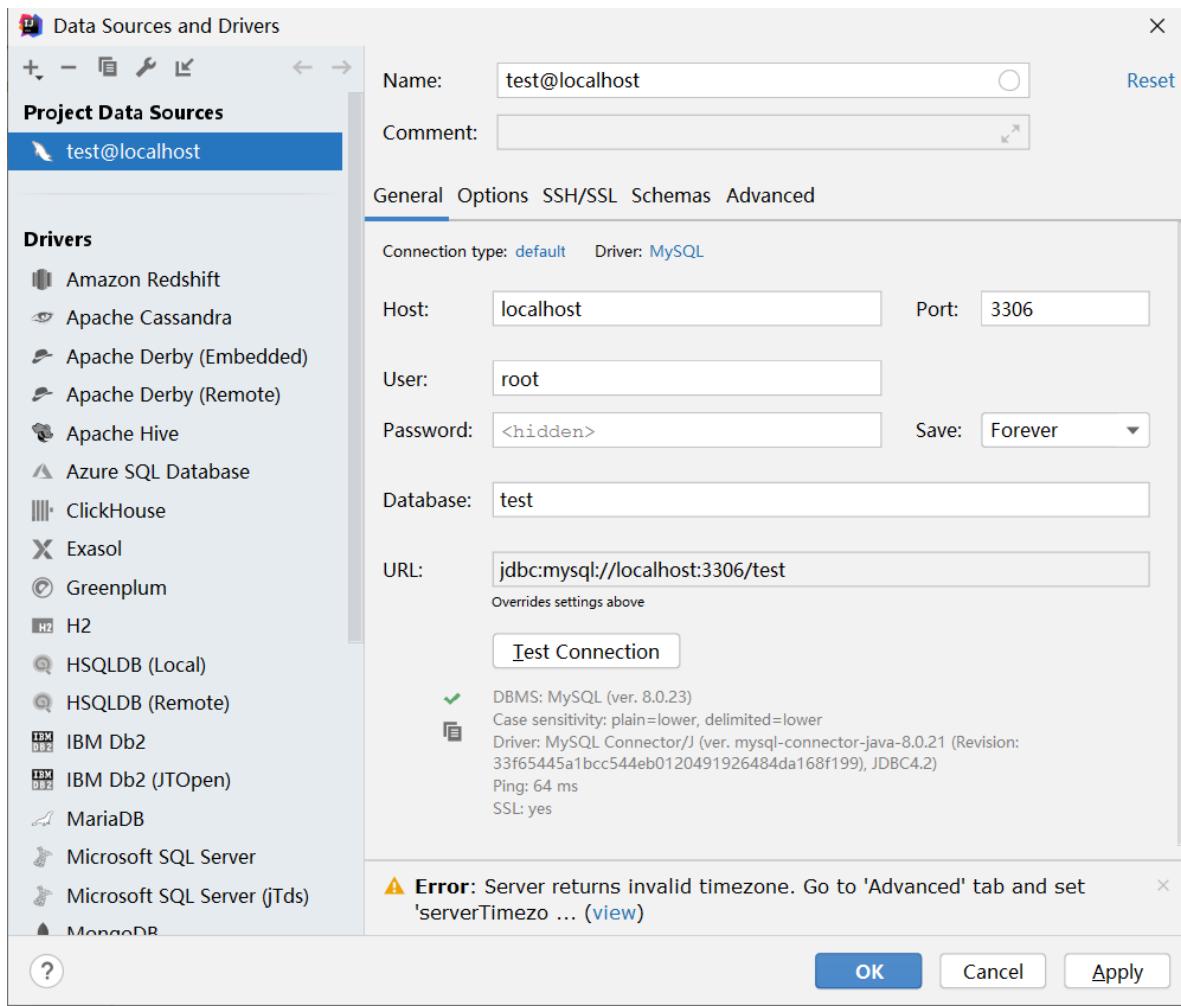
## 配置mysql时区

```
1 | show variables like '%time_zone';
```

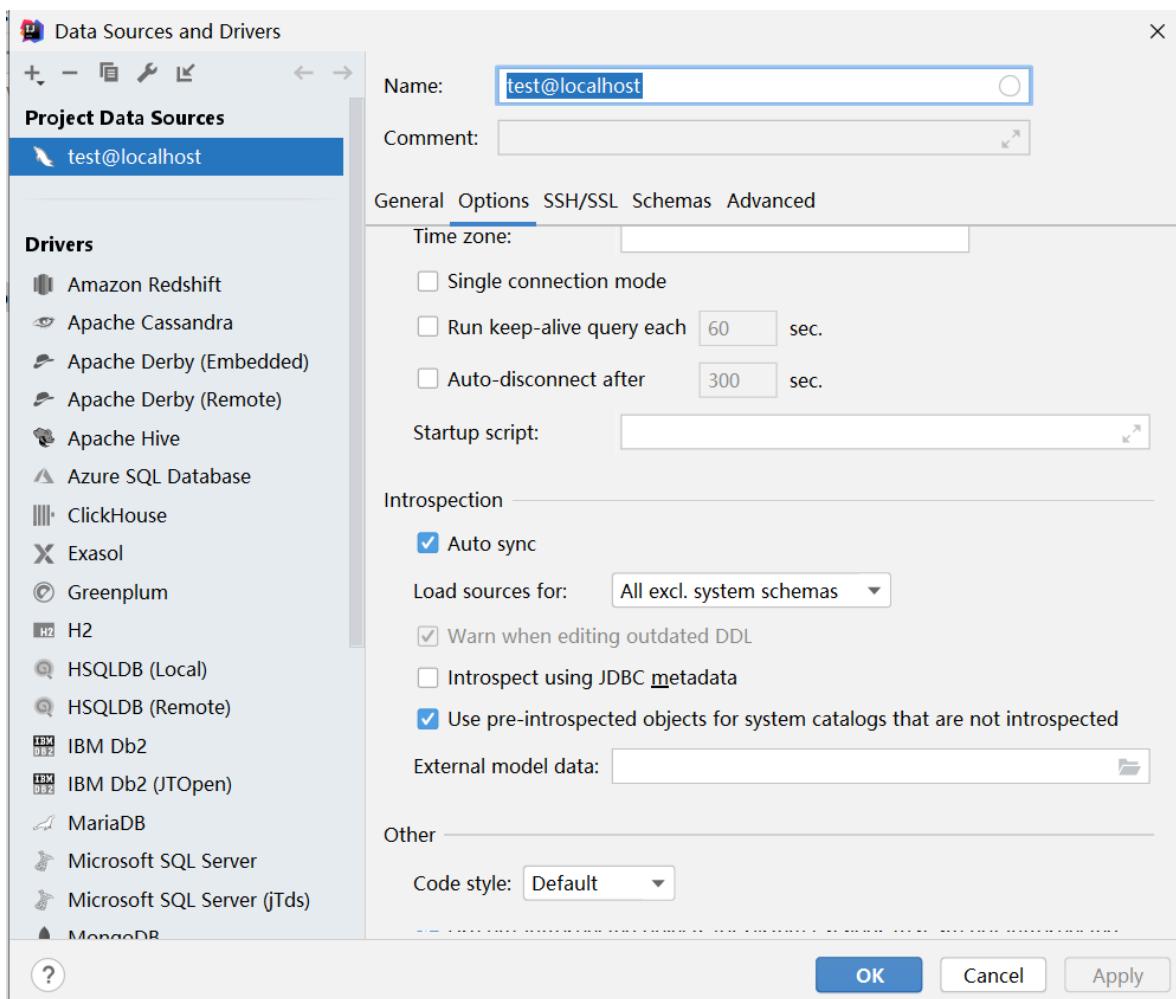
```
mysql> show variables like'%time_zone';
+-----+-----+
| Variable_name   | Value  |
+-----+-----+
| system_time_zone |        |
| time_zone       | SYSTEM |
+-----+-----+
2 rows in set, 1 warning (0.06 sec)
```

```
1 | set global time_zone = '+8:00';
```

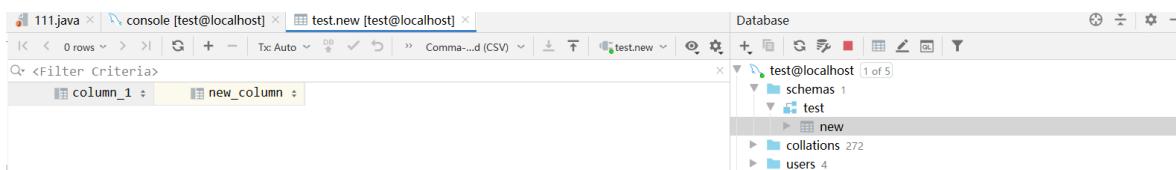
```
mysql> set global time_zone = '+8:00';
Query OK, 0 rows affected (0.00 sec)
```



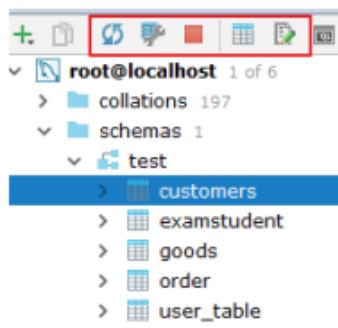
## 设置自动同步



配置好了 Database 之后，IntelliJ IDEA 会自动识别 domain 对象与数据表的关系，也可以通过 Database 的数据表直接生成 domain 对象



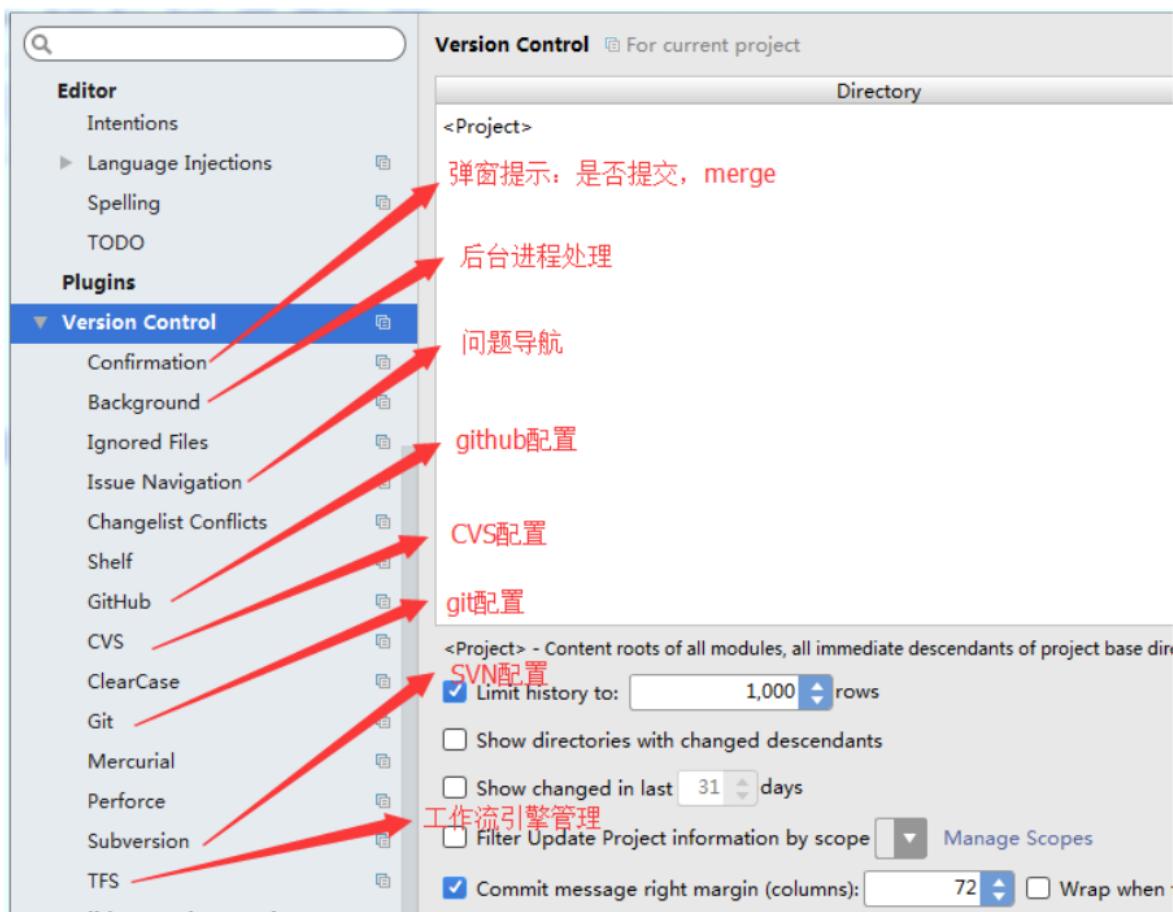
## 2.常用操作



- 图标 1：同步当前的数据库连接。这个是最重要的操作。配置好连接以后或通过其他工具操作数据库以后，需要及时同步。
- 图标 2：配置当前的连接。
- 图标 3：断开当前的连接。
- 图标 4：显示相应数据库对象的数据
- 图标 5：编辑修改当前数据库对象

# # IDEA中使用git

## 配置



IntelliJ IDEA 是自带对这些版本控制工具的插件支持，但是该装什么版本控制客户端还是要照样装的

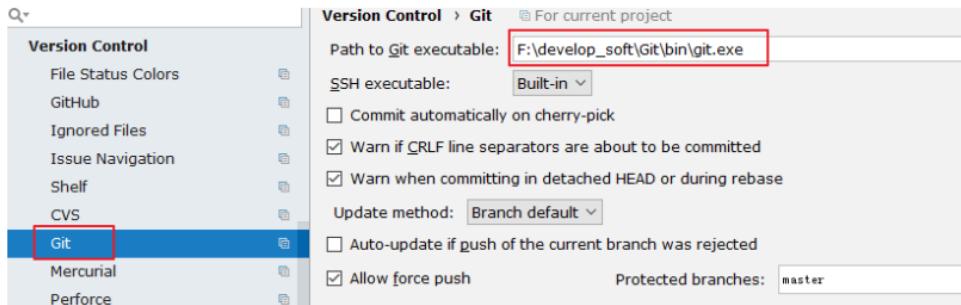
IntelliJ IDEA 对版本控制的支持是以插件化的方式来实现的

## 1. 提前安装好 Git 的客户端

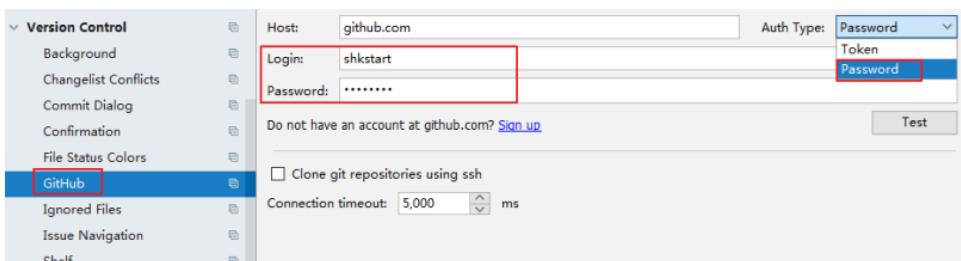
Git 的 msysGit 官网下载: <https://git-scm.com/>

Git 客户端 TortoiseGit 官网下载: <http://download.tortoisegit.org/tgit/>

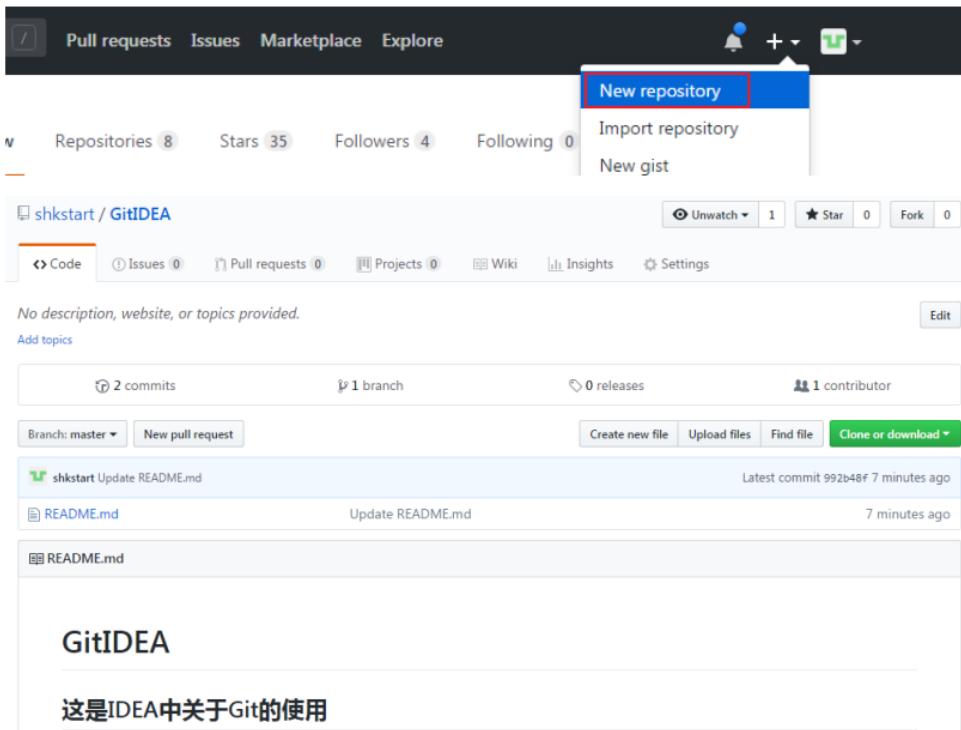
## 2. 关联 git.exe



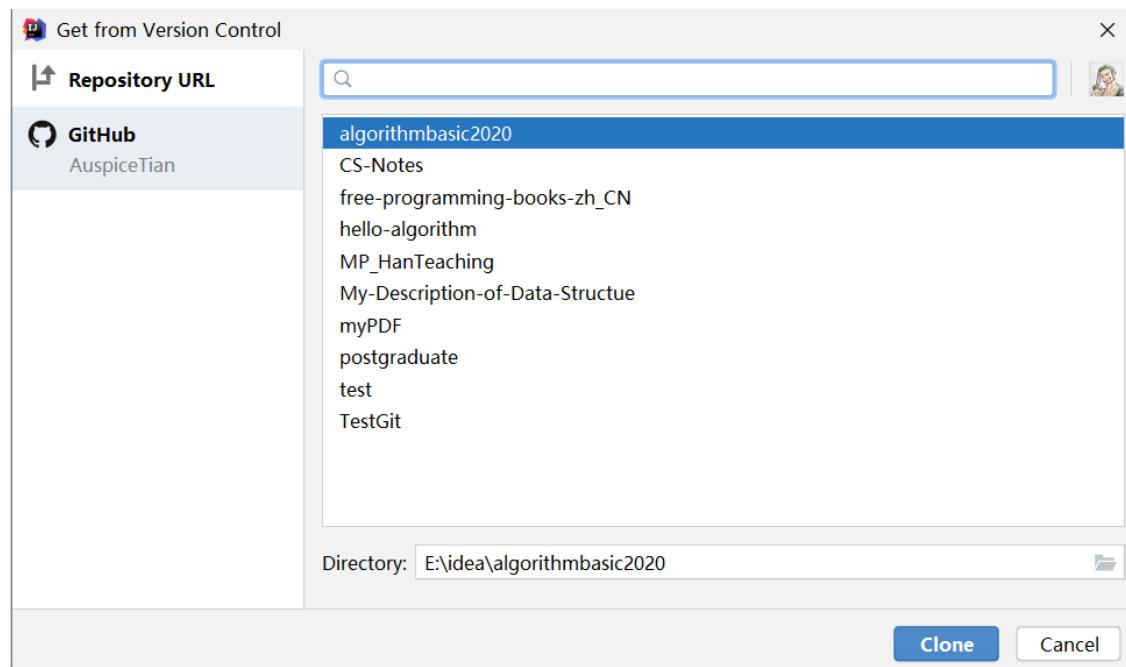
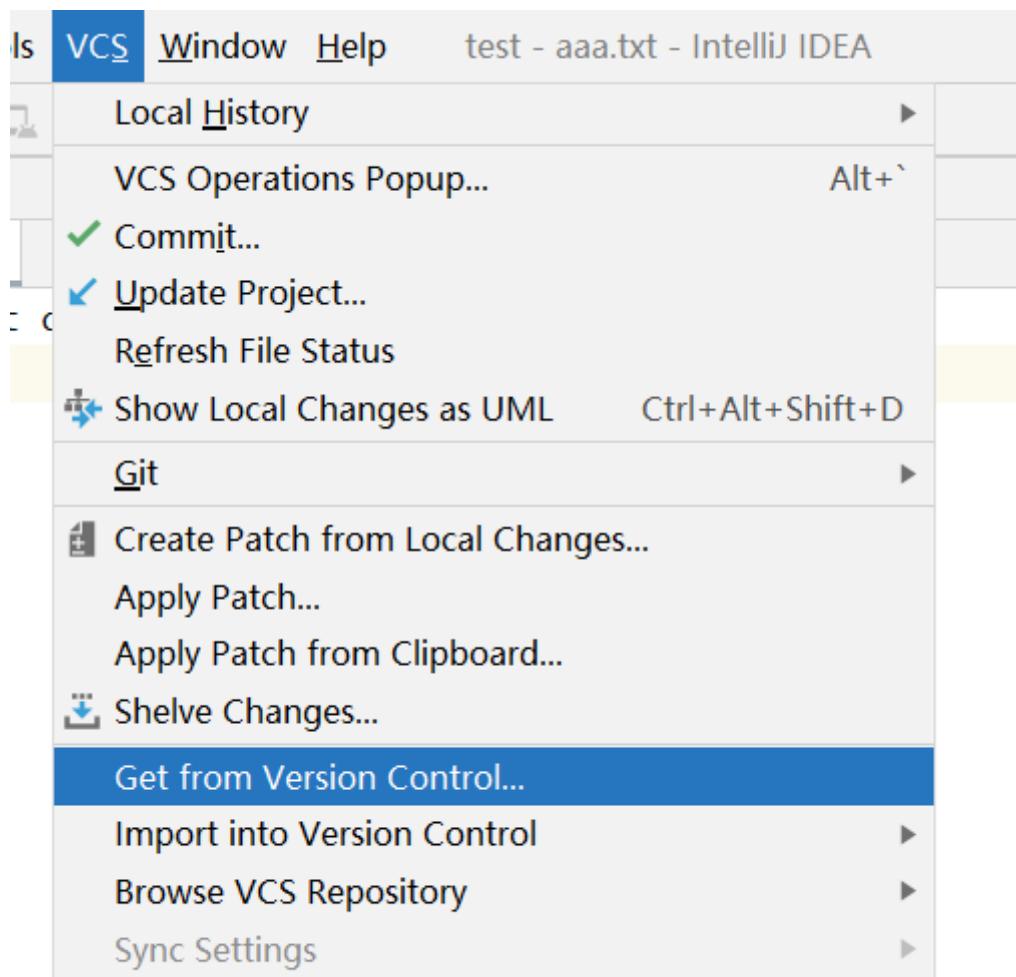
## 3. 关联 GitHub 上的账户，并测试连接



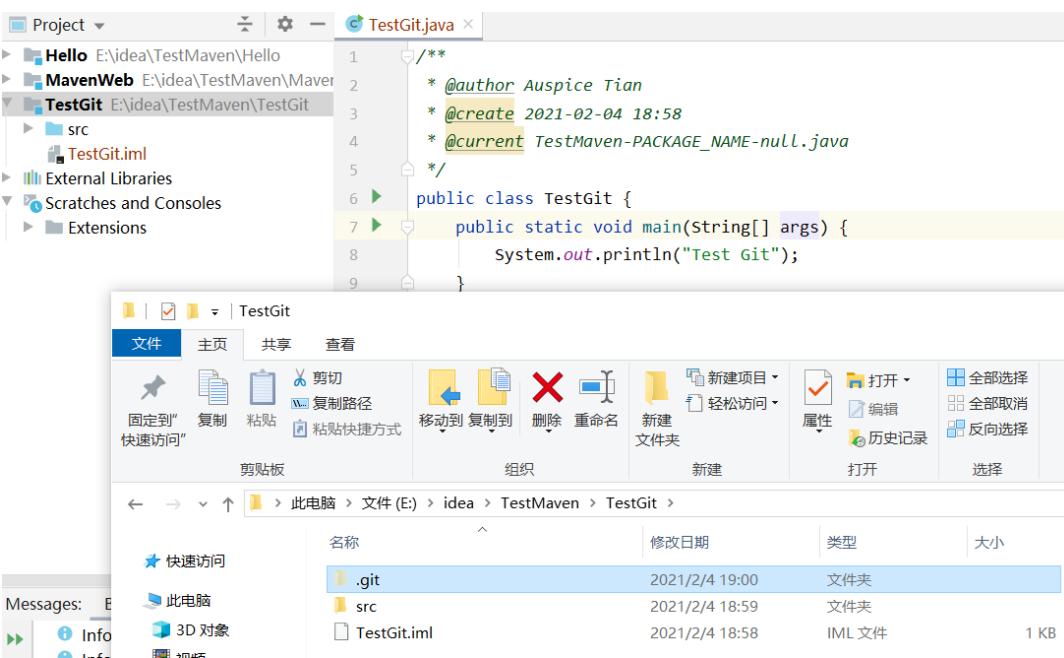
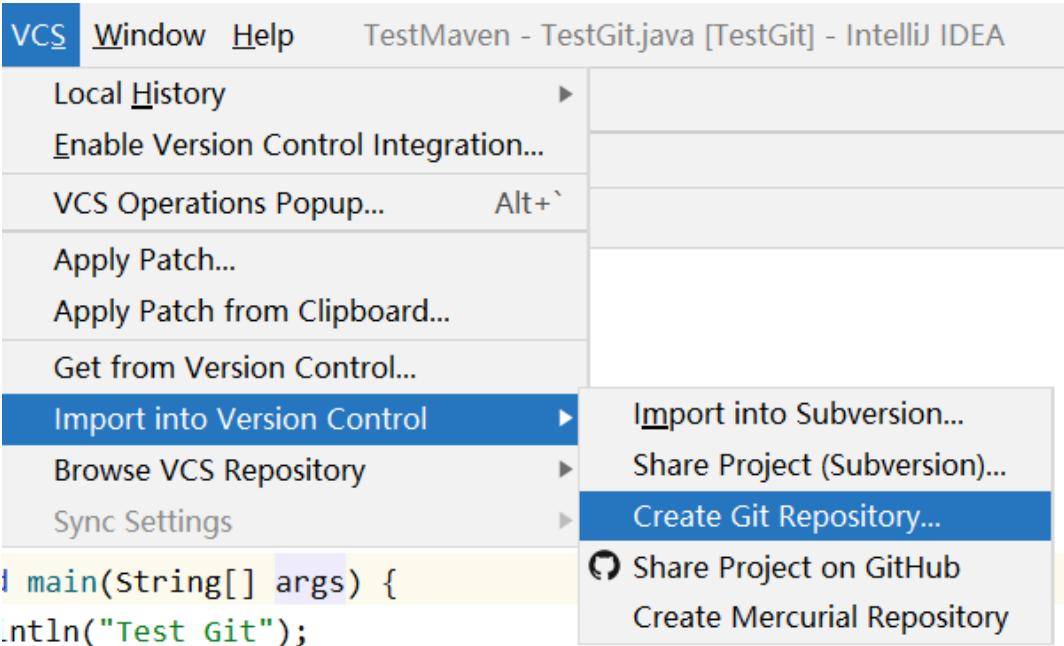
## 4. 在 GitHub 上创建账户下的一个新的仓库作为测试:



## 5. 从远程仓库获取项目



## 6. 新建git本地仓库



## 7. 添加忽略同步文件

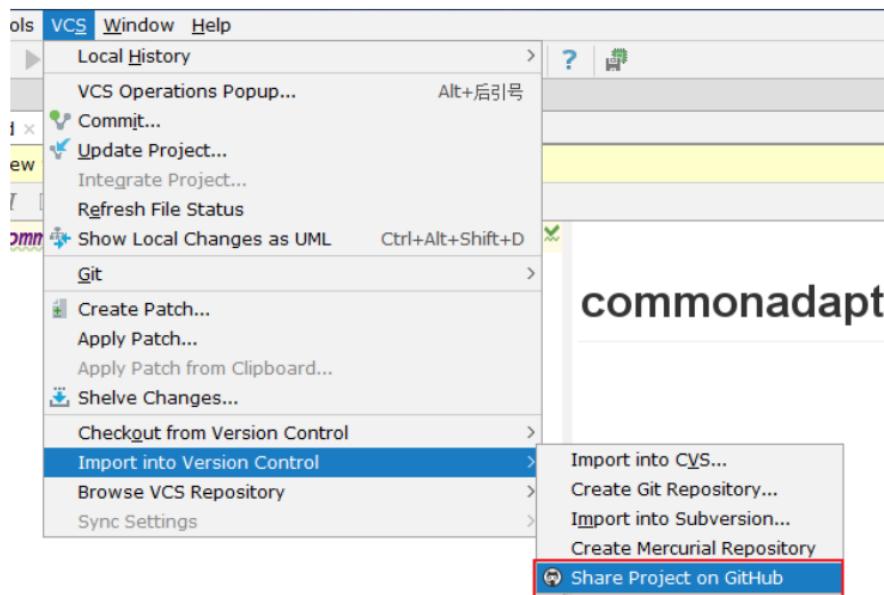
The screenshot shows the IntelliJ IDEA interface with a Java file named `TestGit.java` open. The code contains a single line of code: `System.out.println("Test Git");`. A context menu is open over this line, with the option `Add to .gitignore` highlighted.

```
* @create 2021-02-04 18:58
* @current TestMaven-PACKAGE_NAME-null.java
*/
public class TestGit {
    public static void main(String[] args) {
        System.out.println("Test Git");
    }
}
Commit File...
+ Add Ctrl+Alt+A
Add to .gitignore Add to .gitignore
Annotate
Show Current Revision
Compare with the Same Repository Version
Compare with...
Compare with Branch...
Show History
Rollback... Ctrl+Alt+Z
Repository backward reference inde
mpiling module 'Hello'
ompile java sources
eted with 2 errors and 0 warnings in 5 s 319 ms
\\atguigu\\HelloTest.java
```

Below the code editor, the `exclude` tab of the `TestGit.java` file is selected, showing the following content:

```
# git ls-files --others --exclude-from=.git/info/exclude
# Lines that start with '#' are comments.
# For a project mostly in C, the following would be a good set of
# exclude patterns (uncomment them if you want to use them):
# *.o
# *~
* .iml
.idea
```

## 9. 本地代码分享到 GitHub

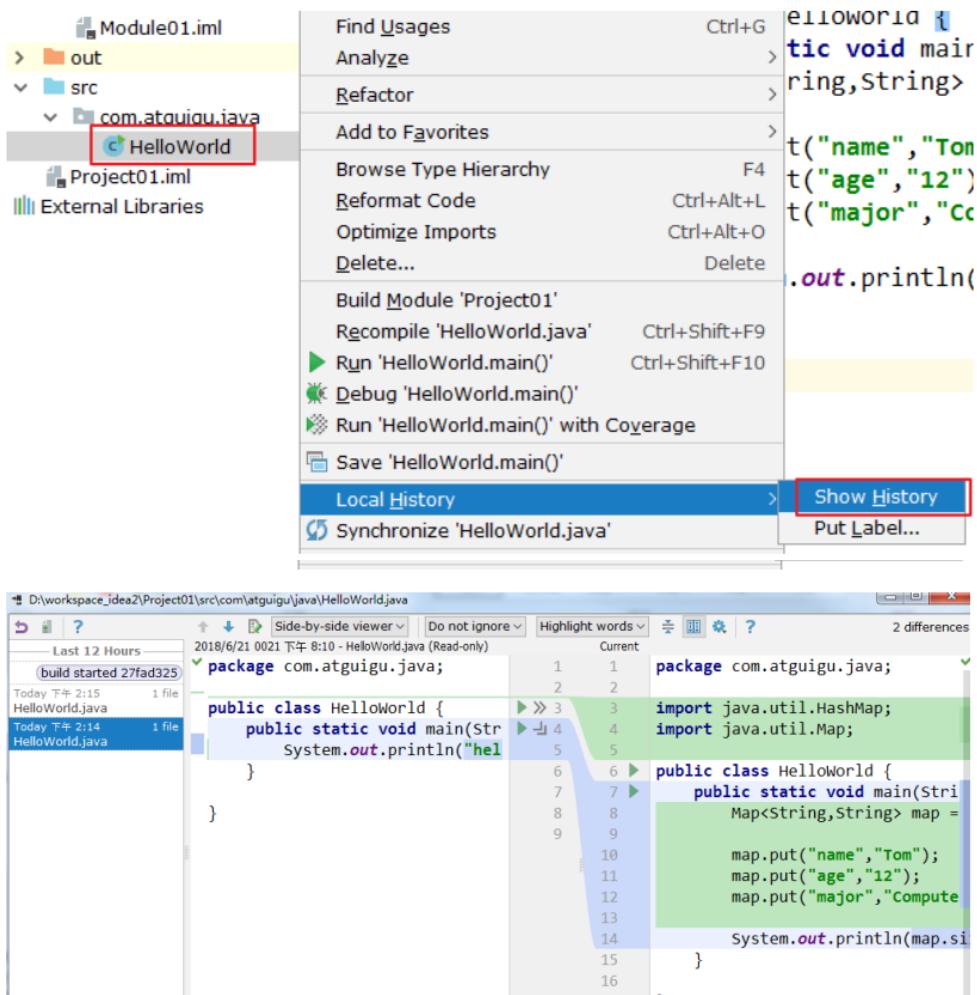


## 10. Git 的常用操作

The screenshot shows the IntelliJ IDEA context menu for a file. The 'Git' submenu is expanded, displaying common operations such as Commit File..., Add, Annotate, Compare with the Same Repository Version, Show History, Revert..., and Repository. The 'Repository' option is highlighted with a blue box. Below the menu, a box contains the following definitions:

- clone:** 拷贝远程仓库
- commit:** 本地提交
- push:** 远程提交
- pull:** 更新到本地

## 11. 没有使用 Git 时本地历史记录的查看



即使我们项目没有使用版本控制功能，IntelliJ IDEA 也给我们提供了本地文件历史记录。

## IDEA中的GIT基本操作

### 版本控制

#### H5 查看历史版本——git log

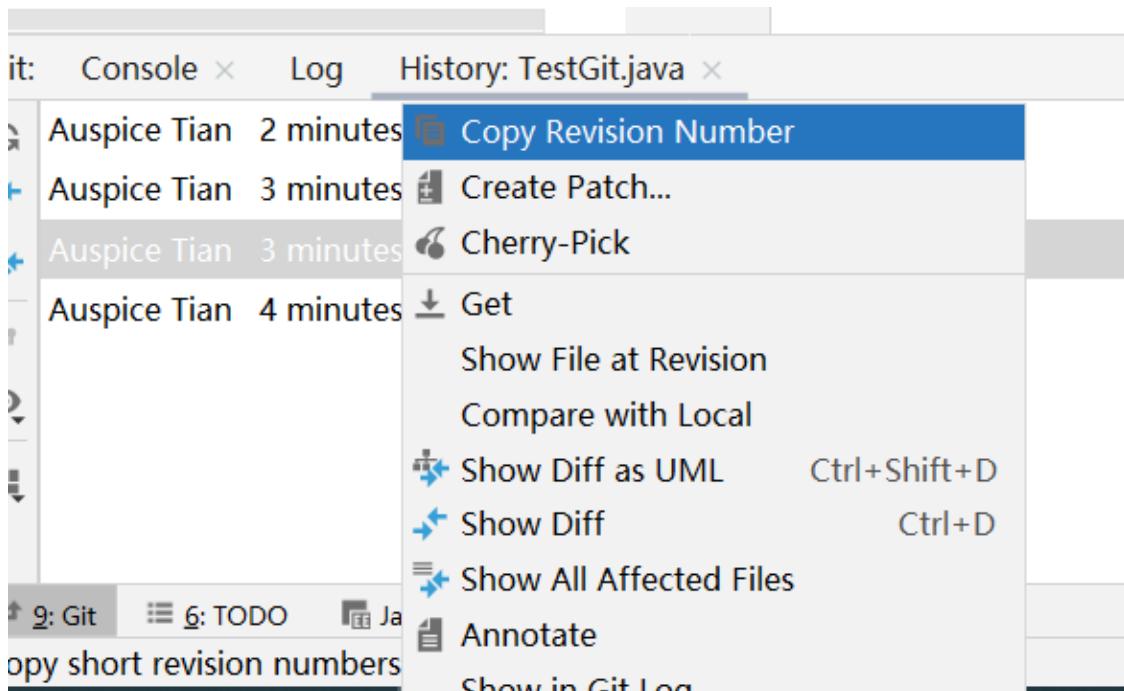
```
/*
 * @author Auspice Tian
 * @create 2021-02-04 18:58
 * @current TestMaven-PACKAGE_NAME-null.java
 */
public class TestGit {
    public static void main(String[] args) {
        System.out.println("Test Git");
        System.out.println("update 1");
        System.out.println("update 2");
        System.out.println("update 4");
    }
}
```

Git: Console Log History: TestGit.java

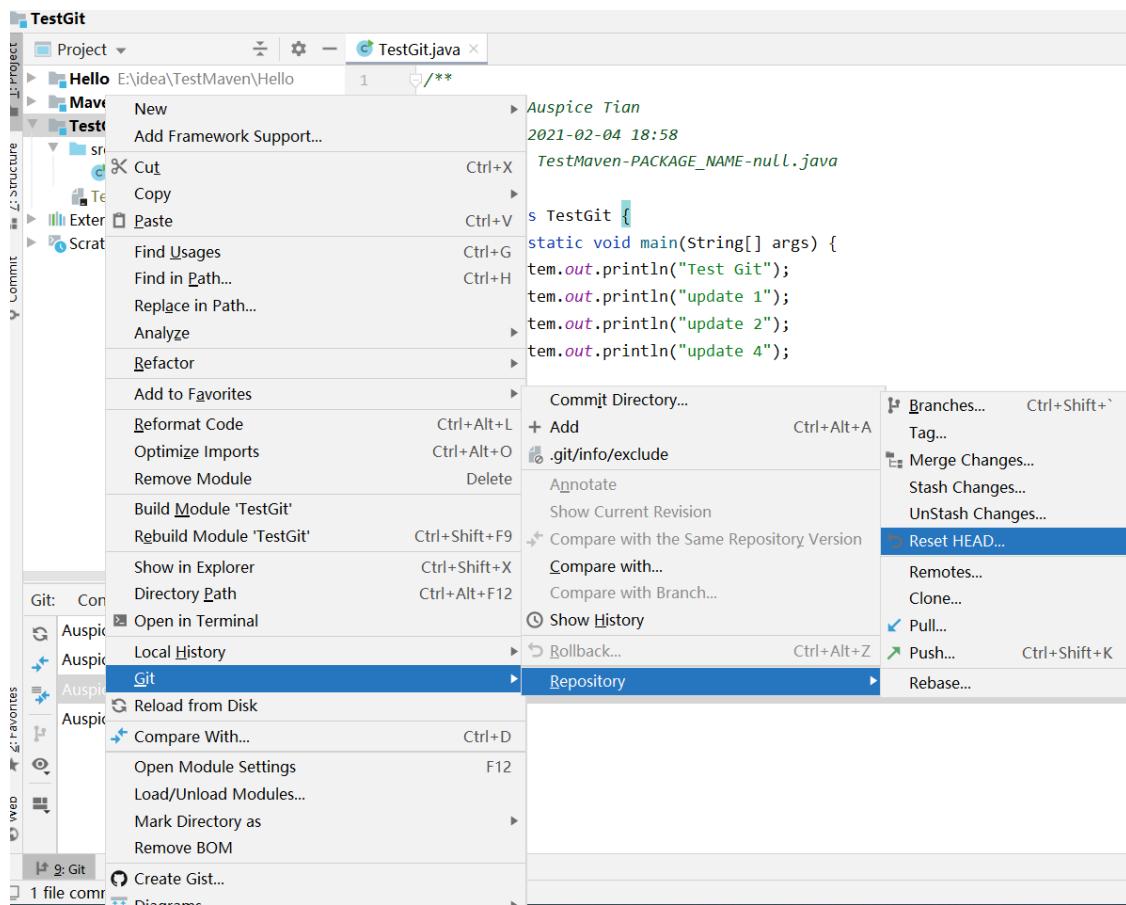
Auspice Tian Moments ago v4 更新4 2021年2月4日19:08:38  
Auspice Tian A minute ago v3 更新3 2021年2月4日19:08:38  
Auspice Tian A minute ago v2 更新1 2021年2月4日19:08:38

## H5 版本切换

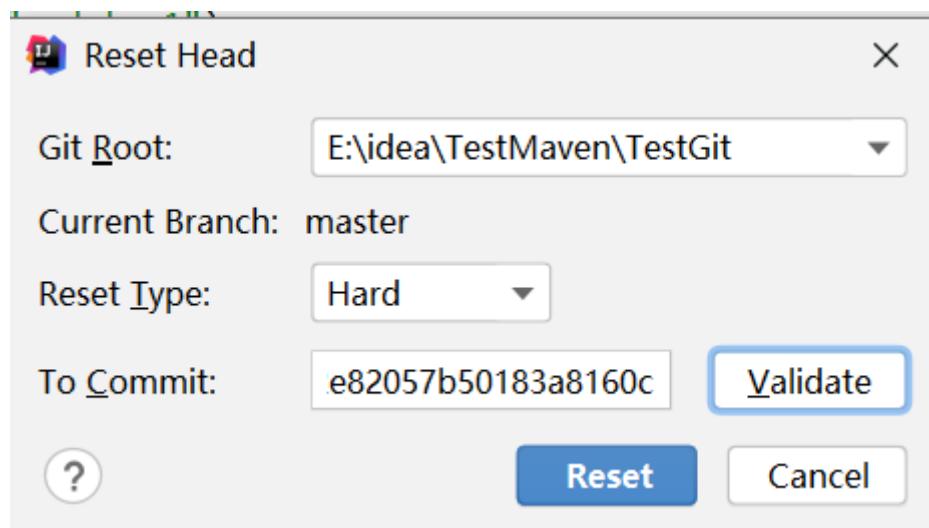
### 1. 获取目标hash



### 2. 切换HEAD指针



选择git reset方式



结果



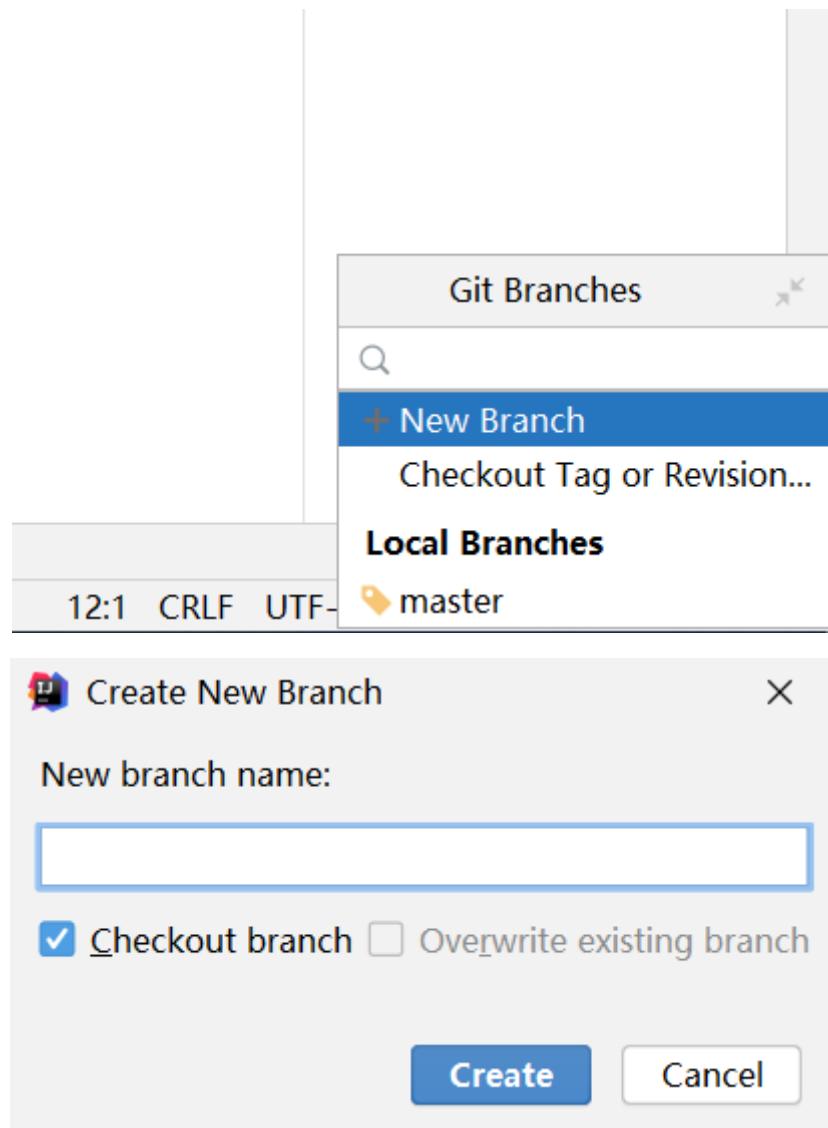
Terminal: Local × +

```
E:\idea\TestMaven\TestGit>git reflog
66b82cf (HEAD -> master) HEAD@{0}: reset: moving to 66b82cf007be8d560c6c02e82057b50183a8160c
b76fd5c HEAD@{1}: commit: v4 更新4 2021年2月4日19:08:38
dfe7bc1 HEAD@{2}: commit: v3 更新3 2021年2月4日19:08:38
66b82cf (HEAD -> master) HEAD@{3}: commit: v2 更新1 2021年2月4日19:08:38
7d4fdbd HEAD@{4}: commit (initial): v1 2021年2月4日19:08:38
```

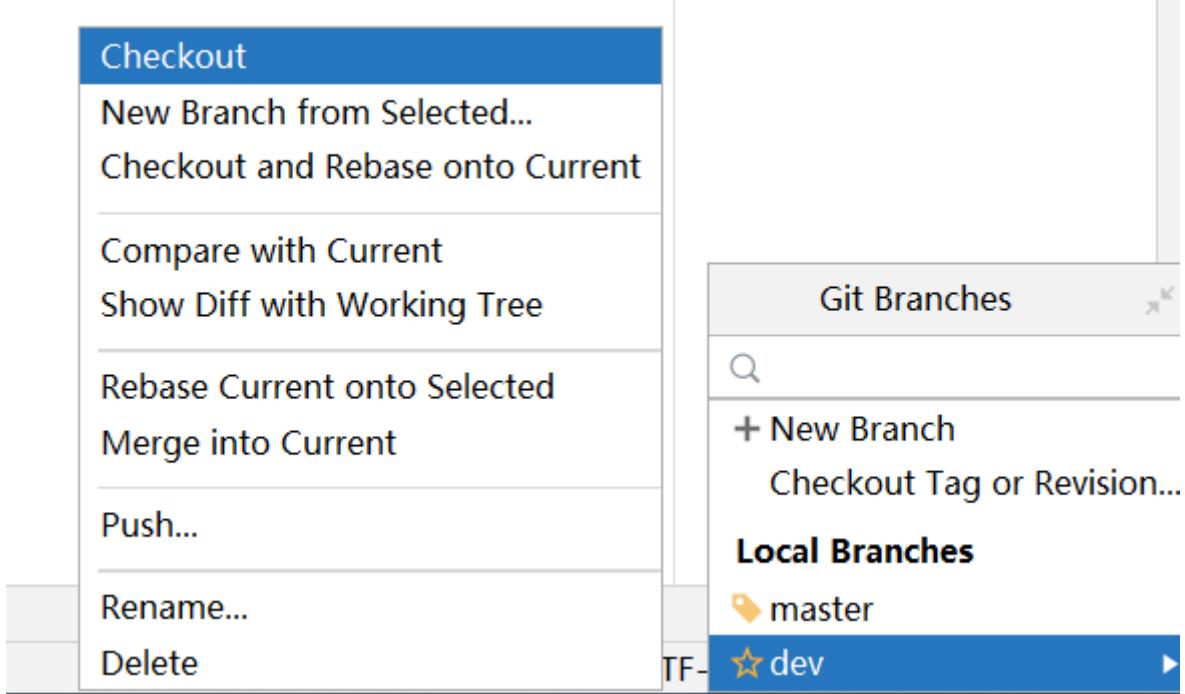
E:\idea\TestMaven\TestGit>

## 分支管理

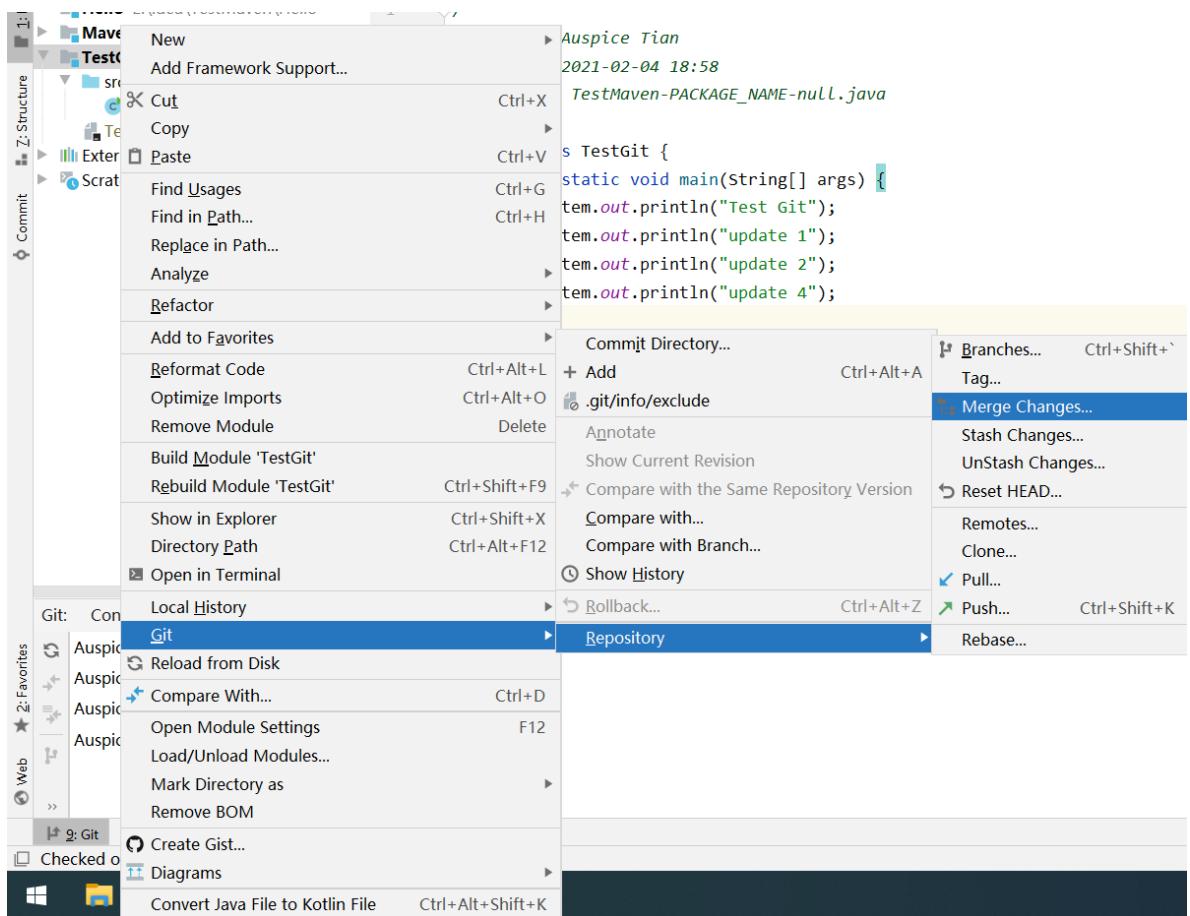
### H5 新建分支

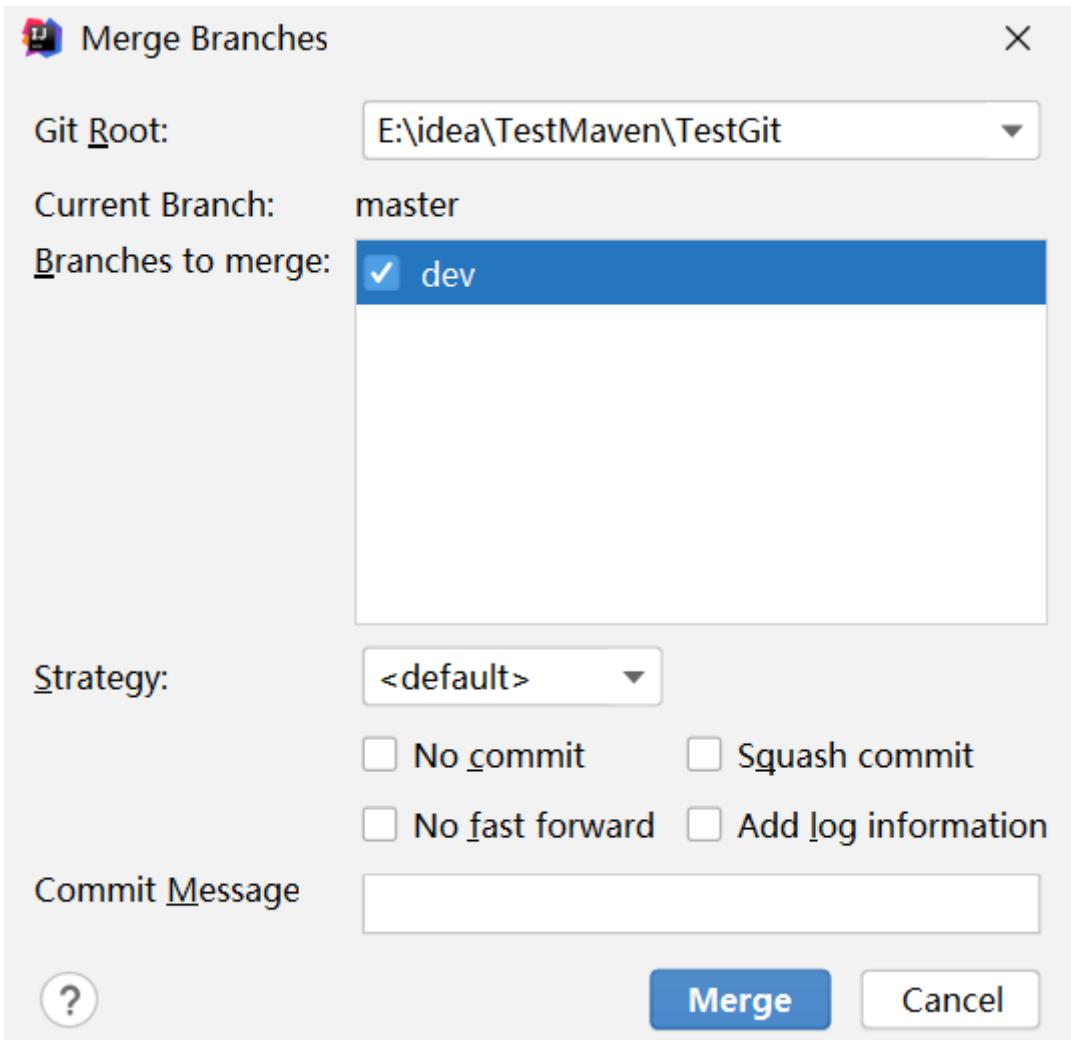


### H5 切换分支

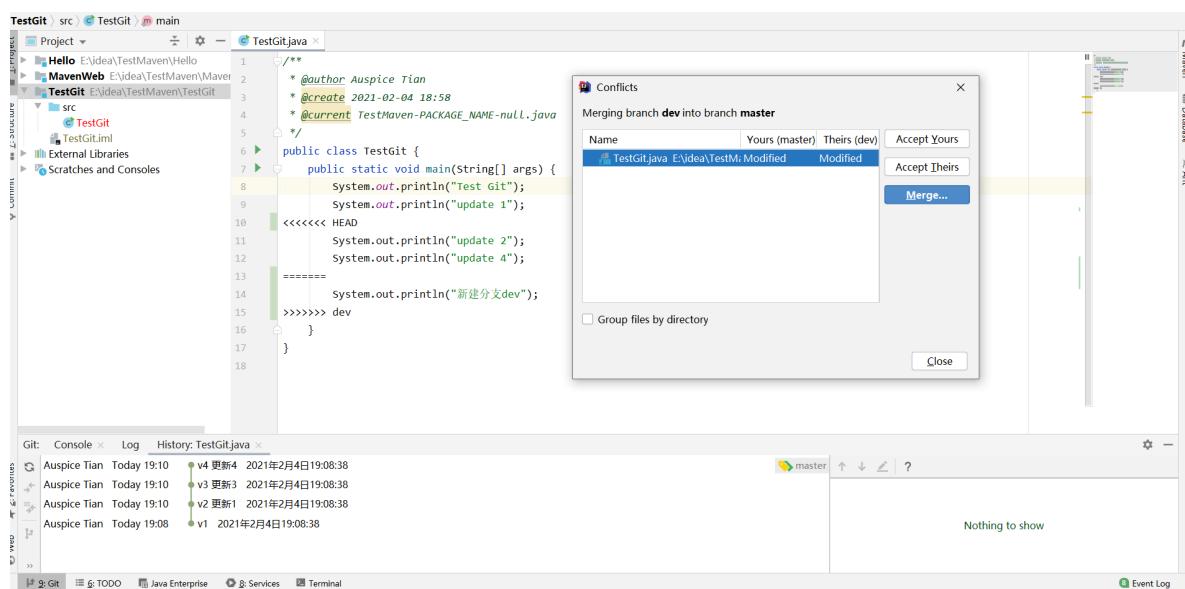


## H5 合并分支





## H5 冲突处理



- Accept Yours: master分支
- Accept Theirs: dev分支
- Merge: 手动合并

Merge Revisions for E:\idea\TestMaven\TestGit\src\TestGit.java

Apply non-conflicting changes: >> Left >> All << Right ↗: Do not ignore ▾ Highlight words ▾ ⚙ ? No changes. 1 conflict Show Details

**Changes from master**

```
1 1 /**
2 2 * @author Auspice Tian
3 3 * @create 2021-02-04 18:58
4 4 * @current TestMaven-PACKAGE_NAME-null.java
5 5 */
6 6 public class TestGit {
7 7     public static void main(String[] args) {
8 8         System.out.println("Test Git");
9 9         System.out.println("update 1");
10 10     System.out.println("update 2"); × >> 10
11 11     System.out.println("update 4");
12 12 }
13 13
14 14 }
```

**Changes from dev**

```
1 1 /**
2 2 * @author Auspice Tian
3 3 * @create 2021-02-04 18:58
4 4 * @current TestMaven-PACKAGE_NAME-null.java
5 5 */
6 6 public class TestGit {
7 7     public static void main(String[] args) {
8 8         System.out.println("Test Git");
9 9         System.out.println("update 1");
10 10     System.out.println("新建分支dev");
11 11 }
12 12
13 13 }
```

Accept Left Accept Right Apply Cancel

## master分支

Project: TestGit.java

TestGit.java

```
1 /**
2 * @author Auspice Tian
3 * @create 2021-02-04 18:58
4 * @current TestMaven-PACKAGE_NAME-null.java
5 */
6 public class TestGit {
7     public static void main(String[] args) {
8         System.out.println("Test Git");
9         System.out.println("update 1");
10        System.out.println("更新2 新建分支2");
11    }
12
13
14 }
```

Git: Console Log History: TestGit.java Update Info: 2021/2/4 21:05

Auspice Tian Moments ago Merge branch 'dev'

Auspice Tian 8 minutes ago v5 新建分支dev 2021年2月4日19:08:38

Auspice Tian Today 19:10 v4 更新4 2021年2月4日19:08:38

Auspice Tian Today 19:10 v3 更新3 2021年2月4日19:08:38

Auspice Tian Today 19:10 v2 更新1 2021年2月4日19:08:38

Auspice Tian Today 19:08 v1 2021年2月4日19:08:38

master dev

## dev分支：保持合并到master的状态

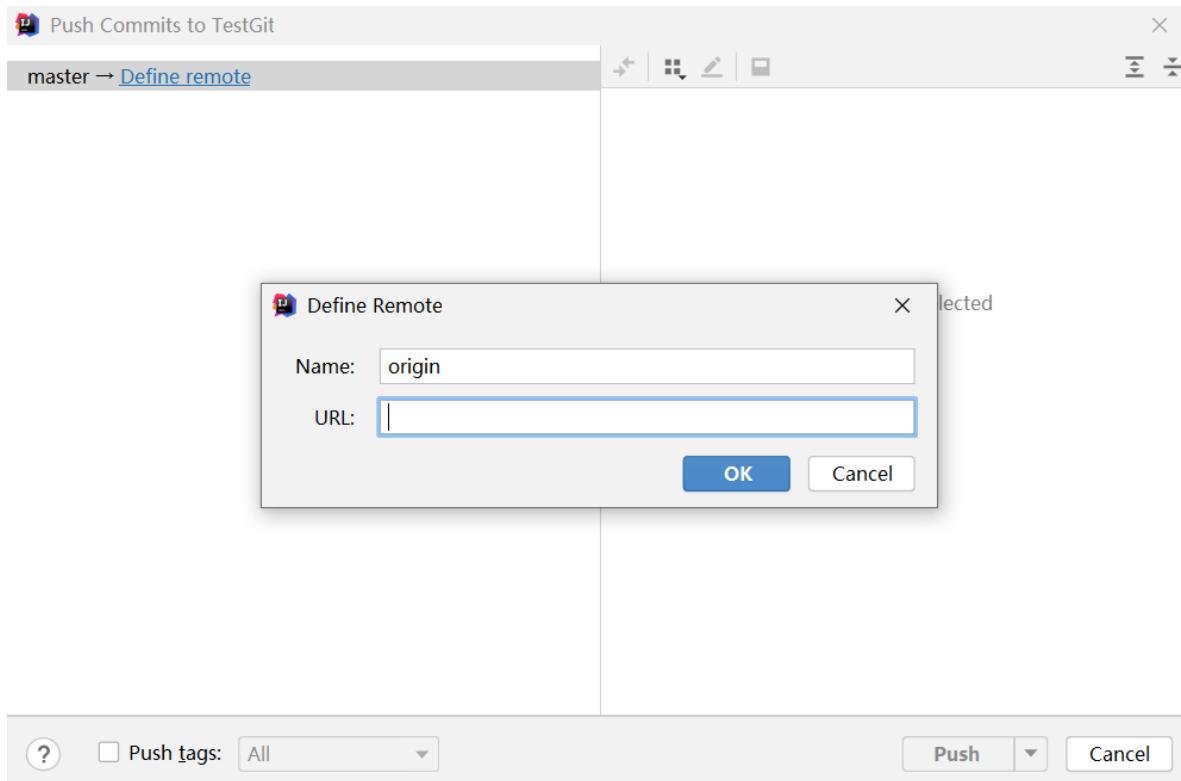
The screenshot shows the IntelliJ IDEA interface. On the left, the Project tool window displays three projects: Hello, MavenWeb, and TestGit. The TestGit project is selected, showing its structure with a SRC folder containing a TestGit.java file and an associated TestGit.iml file. The main editor window contains the code for TestGit.java:

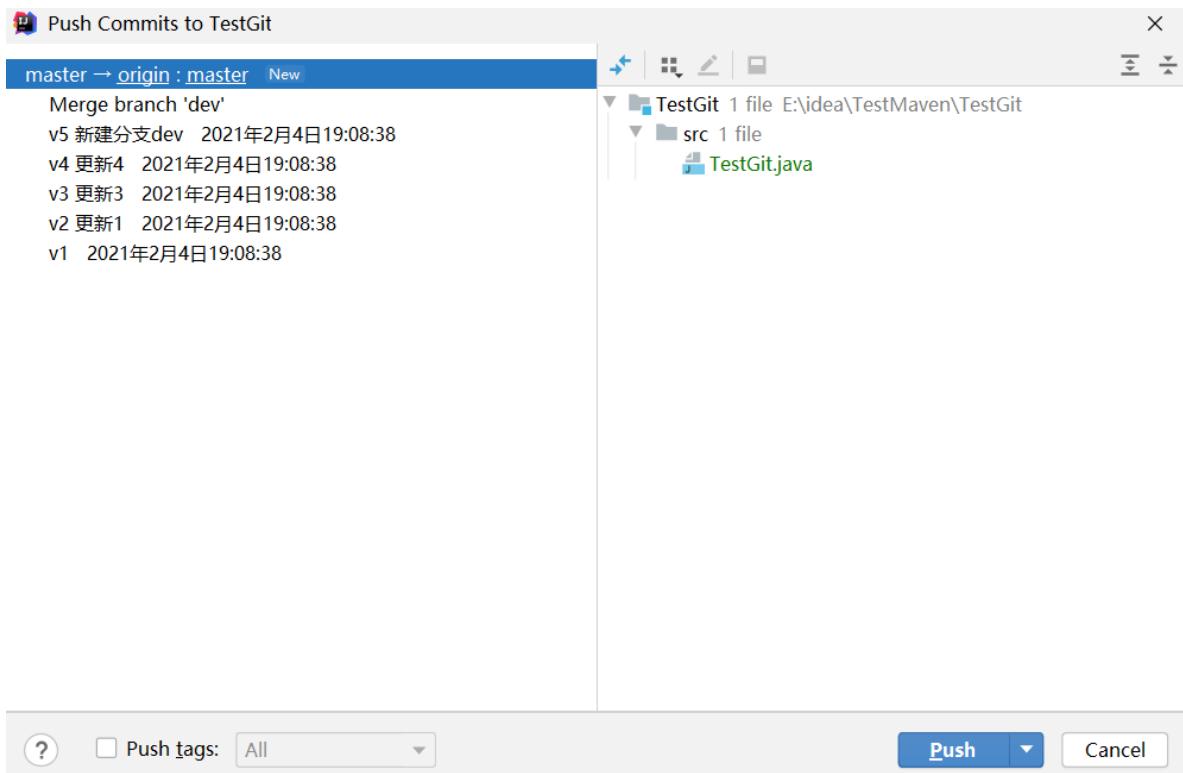
```
1  /**
2   * @author Auspice Tian
3   * @create 2021-02-04 18:58
4   * @current TestMaven-PACKAGE_NAME-null.java
5   */
6  public class TestGit {
7      public static void main(String[] args) {
8          System.out.println("Test Git");
9          System.out.println("update 1");
10         System.out.println("新建分支dev");
11     }
12 }
```

Below the editor, the Git tool window shows the commit history for TestGit.java:

Author	Date	Commit Message	Time
Auspice Tian	9 minutes ago	v5 新建分支dev	2021年2月4日19:08:38
Auspice Tian	Today 19:10	v2 更新1	2021年2月4日19:08:38
Auspice Tian	Today 19:08	v1	2021年2月4日19:08:38

## H5 push到远端库



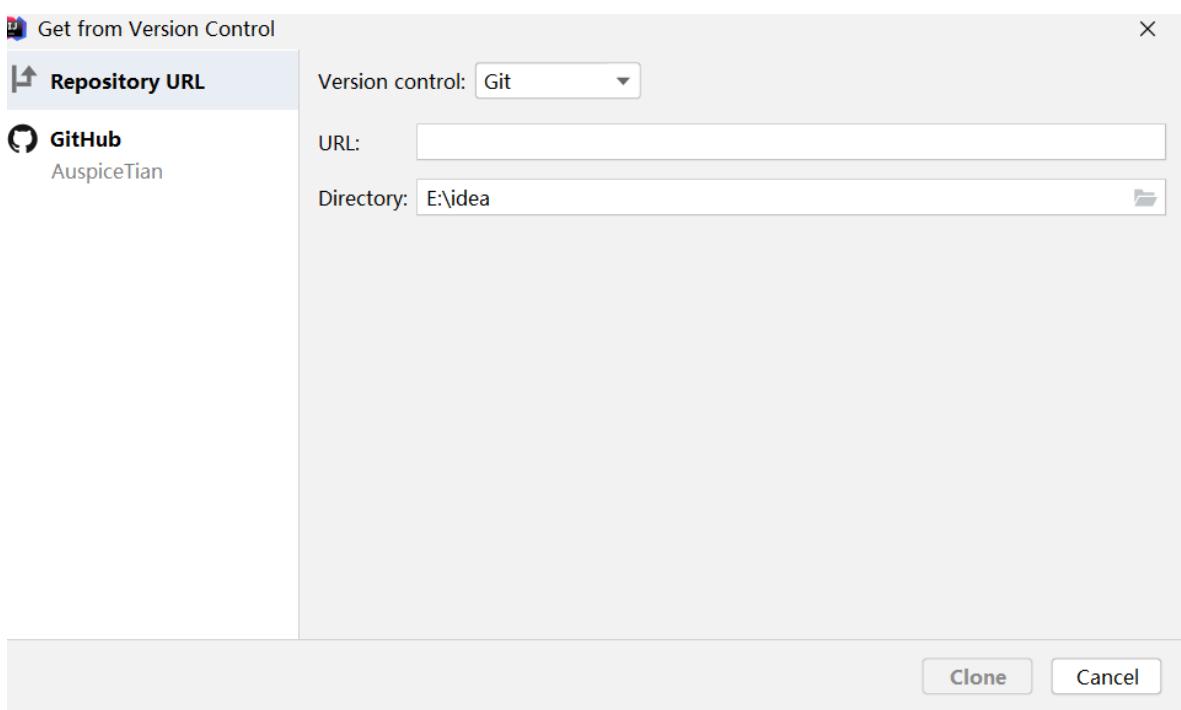
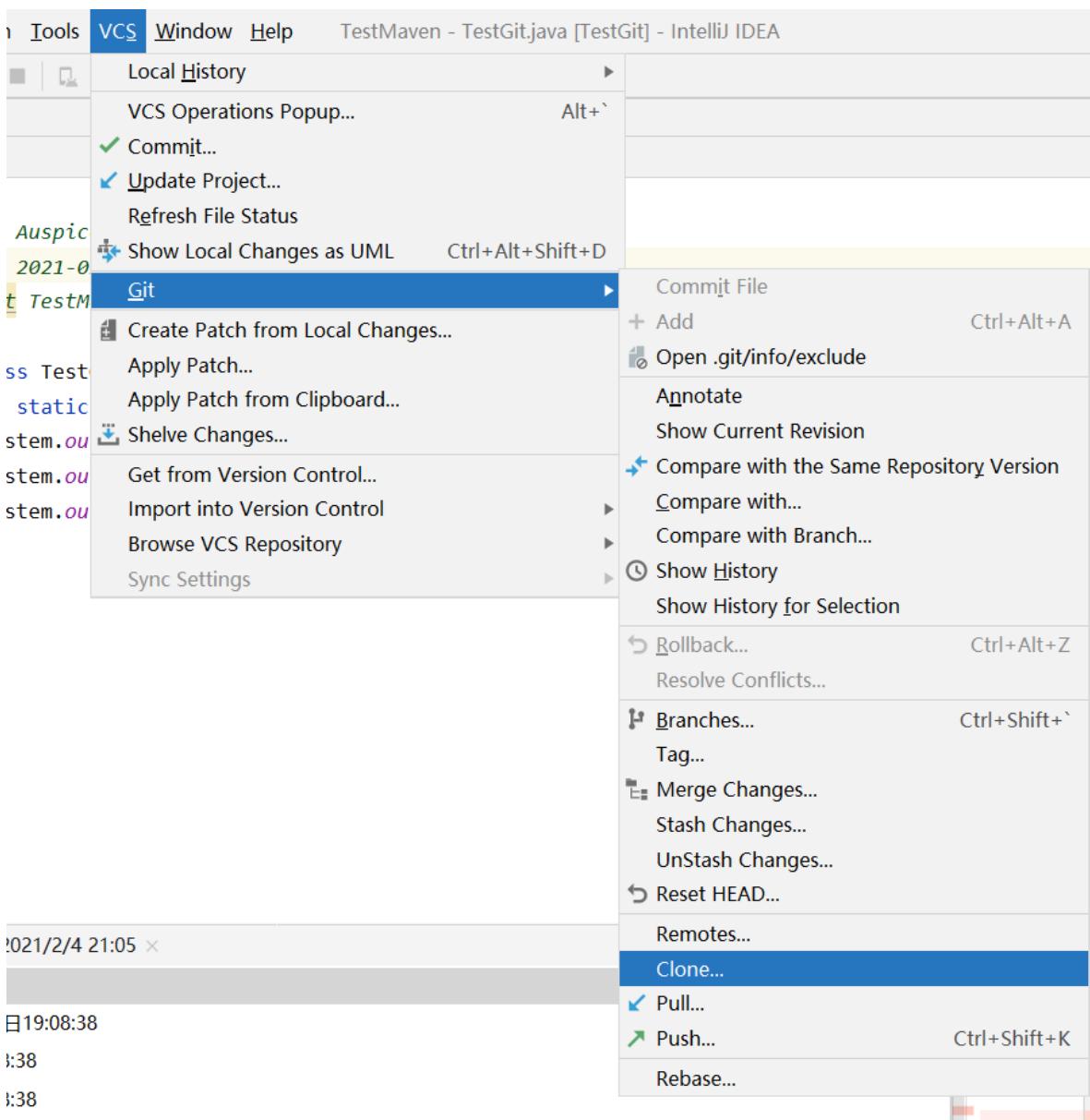


The screenshot shows a GitHub commit history for the 'master' branch. The top navigation bar includes 'master', 'Go to file', 'Add file', and a green 'Code' button.

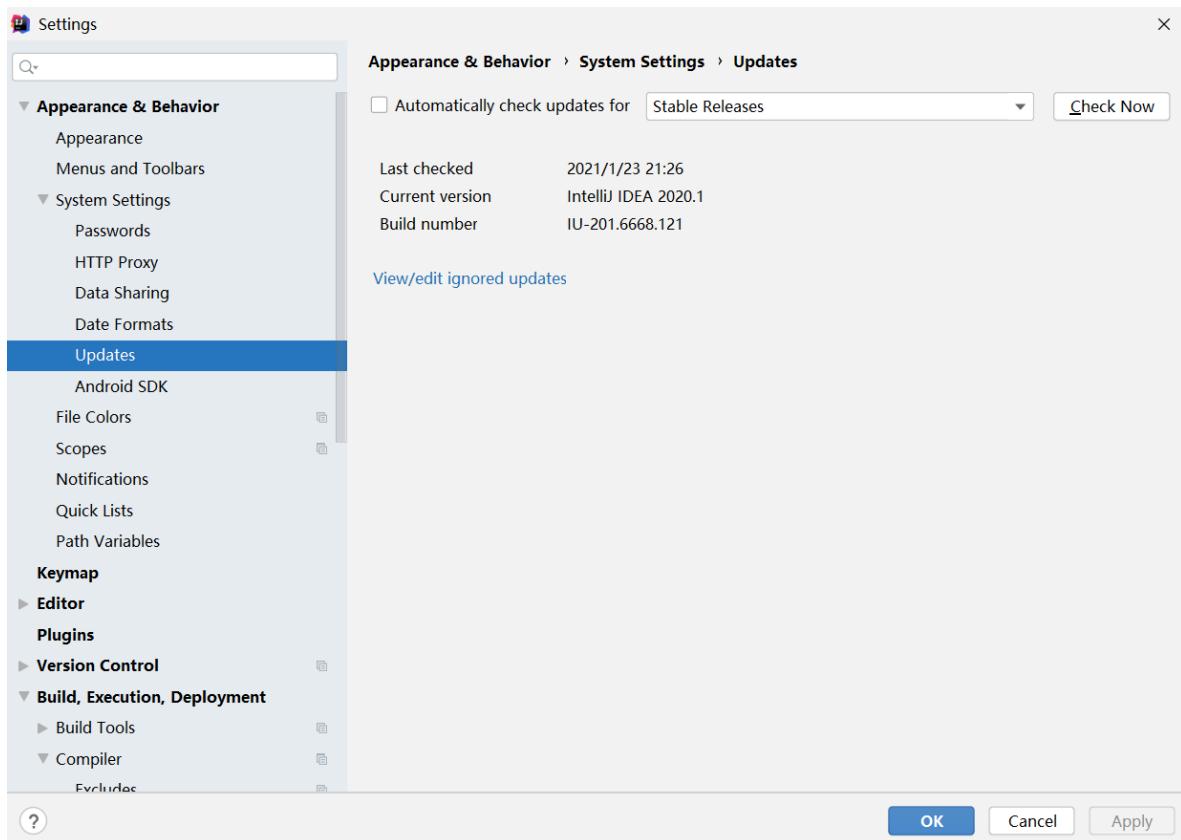
The commits listed are:

- AuspiceTian Merge branch 'dev' ... 28 minutes ago 6
- src Merge branch 'dev' 28 minutes ago

从远程库clone



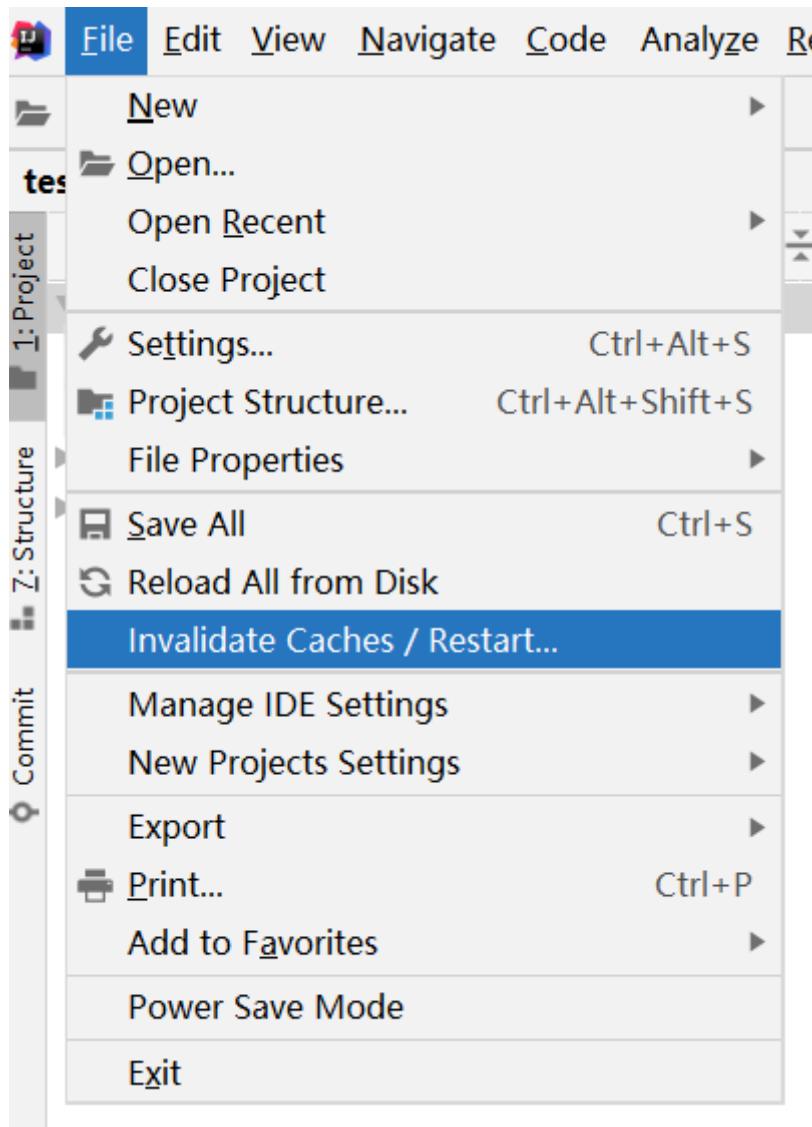
## # 关闭自动更新



## # 清空所有缓存和索引

IntelliJ IDEA 首次加载项目的时候，都会创建索引，创建索引的时间跟项目的文件多少成正比

IntelliJ IDEA 的缓存和索引主要是用来加快文件查询，从而加快各种查找、代码提示等操作的速度



- 清除索引和缓存会使得 IntelliJ IDEA 的 Local History 丢失。所以如果你项目没有加入到版本控制，而你又需要你项目文件的历史更改记录，那你最好备份下你的LocalHistory 目录。

目录地址: C:\Users\Auspice

Tian\AppData\Local\JetBrains\IntelliJIdea2020.1\LocalHistory

C:\Users\Auspice Tian\AppData\Local\JetBrains\IntelliJIdea2020.1

名称	修改日期	类型	大小
caches	2021/2/3 12:03	文件夹	
compiler	2021/2/4 12:12	文件夹	
compile-server	2021/2/4 12:12	文件夹	
conversion	2021/2/4 12:12	文件夹	
database-log	2021/1/23 21:59	文件夹	
external_build_system	2021/2/2 17:39	文件夹	
extResources	2021/1/22 20:21	文件夹	
frameworks	2021/1/22 20:07	文件夹	
grazie	2021/1/22 20:07	文件夹	
httpFileSystem	2021/2/4 10:36	文件夹	
icons	2021/2/4 12:12	文件夹	
index	2021/1/22 20:21	文件夹	
LocalHistory	2021/1/22 20:07	文件夹	
loq	2021/2/4 8:59	文件夹	