**idea新手使用教程**

# 前言

本教程建立在建设你对idea有一个初步的概念，方便你更快的掌握和使用Intellij Idea开发工具。

由于本人使用的是Windows系统，故下方的所有演示均在Windows系统环境下

# Windows下安装

## 系统环境要求

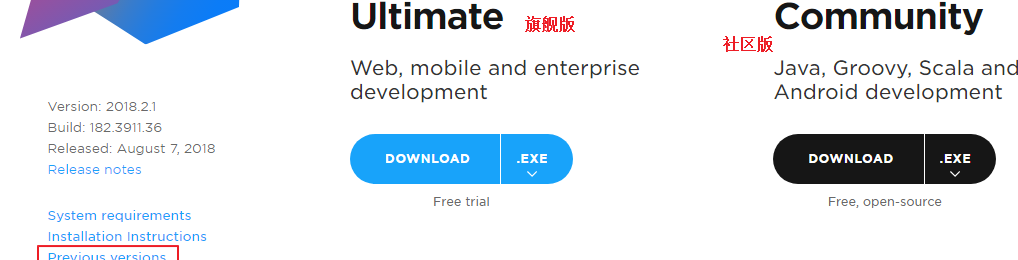
系统支持：Microsoft Windows 8 / 7 / Vista / 2003 / XP（每个系统版本的 32 位和 64 位都可以）

JDK 版本： JDK 1.8 或以上（本人使用的是ideaIU-2017.2.7）

## Intellij idea的下载

http://www.jetbrains.com/idea/download/#section=windows

当然你在进入到这个页面之后也可以选择以前的版本，如下图

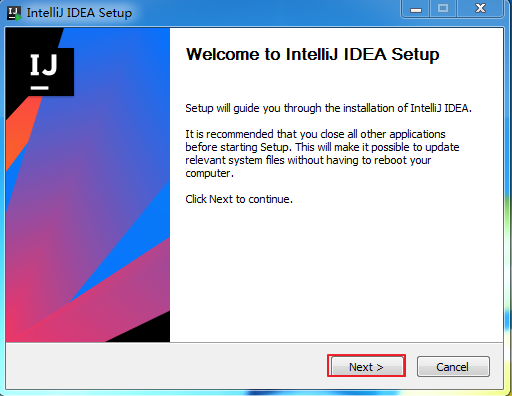


需要说明的是IDEA分旗舰版和社区版两种

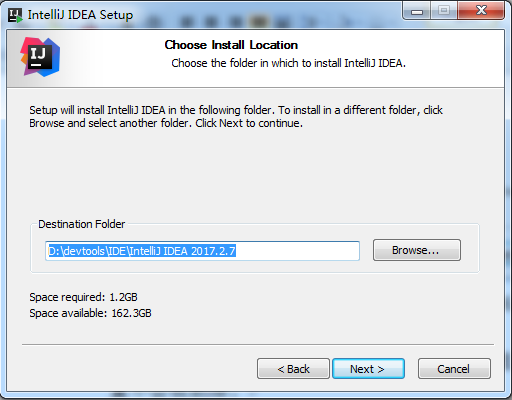
其中旗舰版（Ultimate Edition）为收费版本，有 30 天试用期。如果你是学生、老师、开源项目参与者都可以向官网免费试用旗舰版，具体你可以查看下面链接。社区版（Community Edition）为免费版本，功能较旗舰版少了很多。

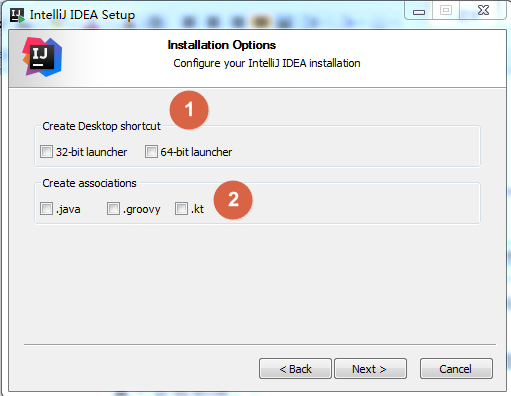
## 首次安装

Intellji Idea的安装是非常简单的，不需要做过多的选择，可以说是简单到都是next即可



指定安装目录

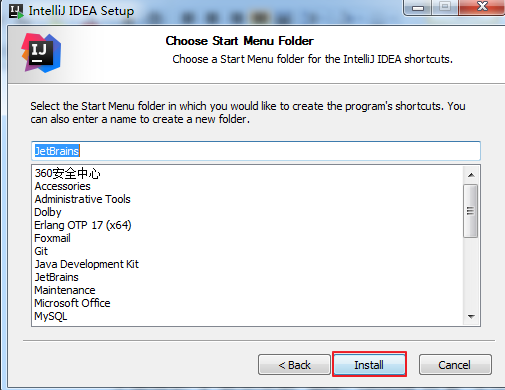


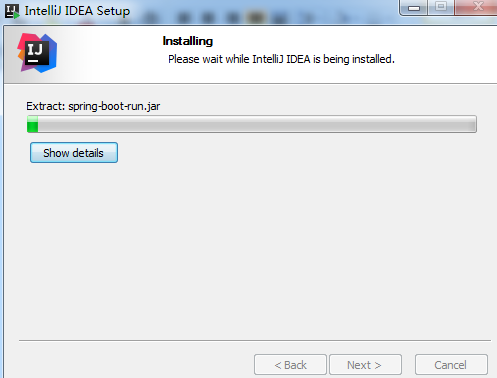


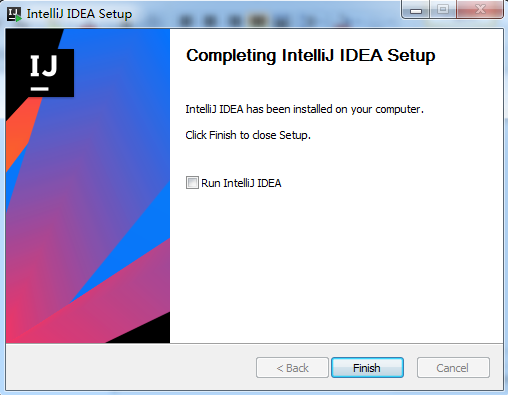
上图标记 1 表示在桌面上创建一个快捷图标（有32位和64位之分），建议勾选上，方便我们在安装后定位 IntelliJ IDEA 安装目录。

上图标记 2 表示关联 Java、Groovy和kt 文件，建议都不要勾选，正常我们会在 Windows 的文件系统上打开这类文件都是为了快速查阅文件里面的内容，如果用 IntelliJ IDEA 关联上之后，由于 IntelliJ IDEA 打开速度缓慢，这并不能方便我们查看。

建议在 Windows 系统上关联此类文件可以用 EmEditor、Notepad++ 这类轻便的编辑器。

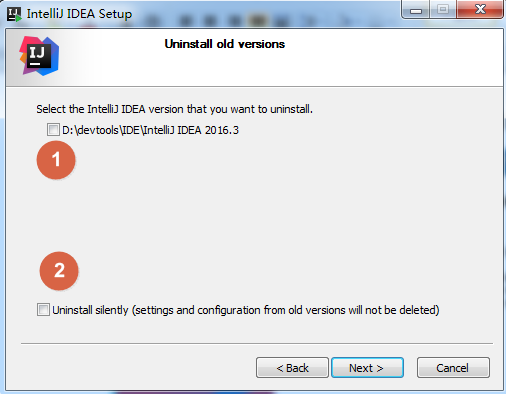






至此IDEA就安装完成了

## 已有旧版本的安装



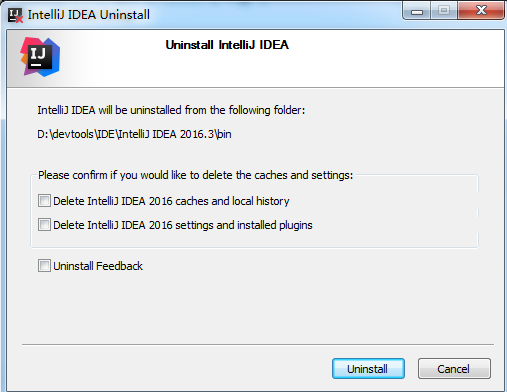
上图，显示我目前电脑中已经有一个 IntelliJ IDEA 版本，如果我勾选了标记 1，则表示安装之前会先卸载掉电脑上的旧版本。

上图标记 2，如果勾选了，则 IntelliJ IDEA 会直接安静地卸载旧版本，而旧版本的个性化设置不会被删除。

在小版本迭代中建议是卸载掉旧版本的，然后再进行新版本安装，因为小版本迭代一般都是 Bug 的修复，保留旧版本没有多大意义。

在大版本迭代中建议是保留旧版本，也就是不勾选上图标注 1，IntelliJ IDEA 是支持一台电脑装多个版本的。

接下来的步骤我们假设勾选标记1再进行安装。



上图，由于上一步勾选了卸载旧版本选项，所以出现了选择删除旧版本的配置选项。

第一个选项：删除旧版本的缓存和本地历史记录。

第二个选项：删除旧版本的个人个性化设置。

建议两个都不要勾选。

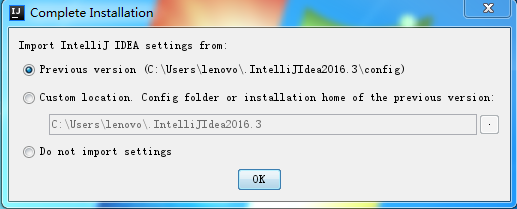
点击 uninstall，进入全自动的卸载过程，卸载完成接下来的步骤跟上文“首次安装”一致，这里不再进行说明。

# 首次运行

## 向导功能

假如你计算机上在过去已经有安装过 IntelliJ IDEA 17 的版本，且你在卸载 IntelliJ IDEA 的过程中，IntelliJ IDEA 的配置文件目录都没有删除，那安装新版本之后是不会有首次运行的功能向导的。

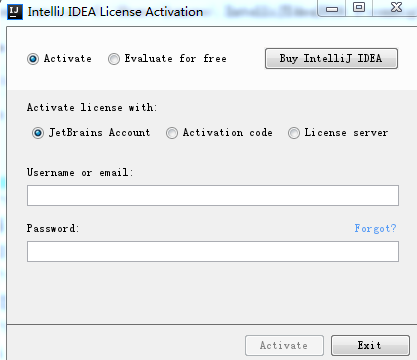
假如你计算机上没有安装过 IntelliJ IDEA，或是 卸载 IntelliJ IDEA 过程中你删除了 IntelliJ IDEA 的配置文件目录，则当你双击运行桌面上的 IntelliJ IDEA 快捷图标，将进入下面介绍的向导过程。



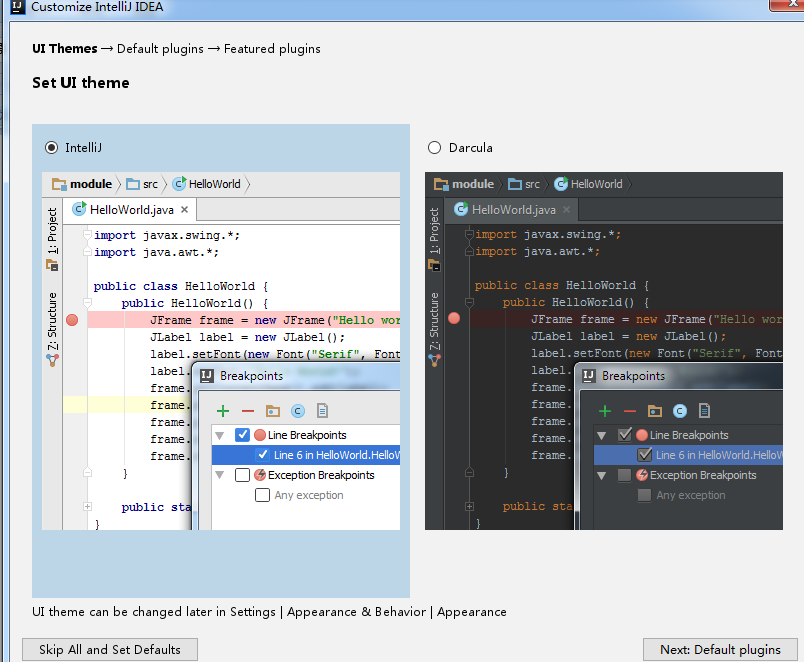
上图第一个单选按钮表示 IntelliJ IDEA 识别到我计算机上有 IntelliJ IDEA 16 版本的旧配置，如果我选择了该选项，则 IntelliJ IDEA 将自动把旧版本的配置文件转移到新版本的配置文件目录上。如果你计算机上首次安装一般是没有该选项的。

上图第二个单选按钮表示你可以指定 IntelliJ IDEA 导入你计算机上存在其他目录的 IntelliJ IDEA 配置文件目录，如果你有的话。

上图第三个单选按钮表示你没有任何早期版本的 IntelliJ IDEA 配置，你不导入任何配置，让 IntelliJ IDEA 生成一份新的配置。



上图默认选择中的是 Buy IntelliJ IDEA，验证 IntelliJ IDEA 的许可有如图三种方式，我们这里使用的是 30 天试用版本进行演示，顾单击 Evaluate for free for 30 days 进行下一步。



上图选择的时候 IntelliJ IDEA 主题 UI，在 Windows 系统版本中 IntelliJ IDEA 自带了 4 个主题，但是用的最多的就是上图这 2 种，其中大家基本偏爱黑色的 Darcula。这个没有好坏之分，根据你的喜好来进行选择，我们演示的版本就是用 Darcula。

下方的两个选项按钮

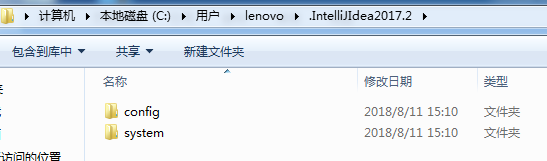
Skip All and Set Defaults：跳过该步骤并设置默认的插件

Next:Deault plugins 下一步跳转到设置默认插件页面，显示了 IntelliJ IDEA 支持的主要的一些扩展功能或者说是工具、插件也可以。你可以根据自己开发的需求进行禁用一些扩展，这样可以稍微减轻 IntelliJ IDEA 运行时所占内存，加快运行速度，但是效果并不会很明显就是。

点击Skip All and Set Defaults

恭喜你的IDEA安装完成

# 设置目录介绍



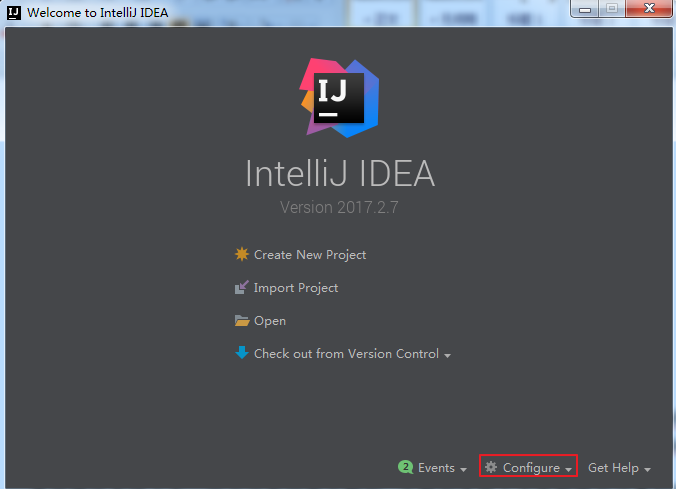
不管你使用的是哪个操作系统，IntelliJ IDEA 的设置目录命名是统一的、有规律：.IntelliJIdea17。2。其中17表示大版本号，如果你电脑上还同时装有 16 的版本，那则还应该会有一个：.IntelliJIdea16 的设置目录，其他版本道理一样。

在三大主流的操作系统上该文件夹都在当前用户的 Home 目录，Windows 上即 %userprofile%，Linux 和 Mac 上即 ~

对于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 会再自动帮你再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，我是很建议新人可以多自己摸索 IntelliJ IDEA 的配置，多几次还原，有助于加深对 IntelliJ IDEA 的了解。

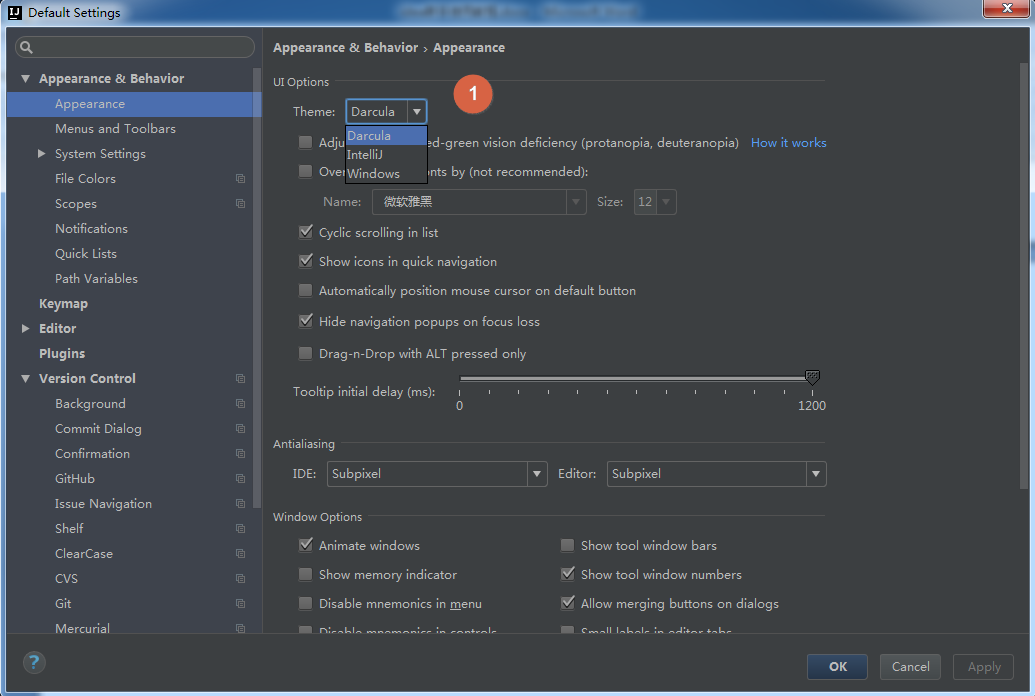
config 目录是 IntelliJ IDEA 个性化化配置目录，或者说是整个 IDE 设置目录。也是我个人认为最重要的目录，没有之一，如果你还记得安装篇的介绍的时候，安装新版本的 IntelliJ IDEA 会自动扫描硬盘上的旧配置目录，指的就是该目录。这个目录主要记录了：IDE 主要配置功能、自定义的代码模板、自定义的文件模板、自定义的快捷键、Project 的 tasks 记录等等个性化的设置。

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，但是也是最不可或缺目录之一。



# IDEA 主题、字体、编辑区主题、文件编码修改、乱码问题

## 主题修改

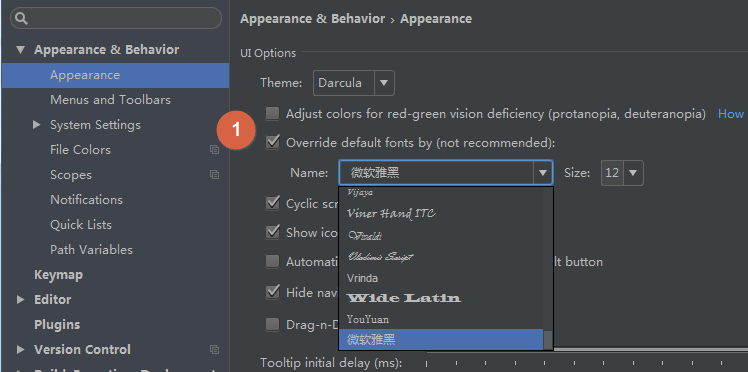


上图标注 1 所示为 IntelliJ IDEA 修改主题的地方。在 Windows 系统上 IntelliJ IDEA 默认提供的主题有三套：Darcula、IntelliJ、Windows。除了 Darcula 是黑色主题，其他三套都是以白色为背景的。

其他操作系统上不一定会也有三套主题的，主题的选择上大家根据自己喜好即可。改变主题需要重启 IntelliJ IDEA 方可看到效果。

## 字体修改

### 主题字体修改



如上图标注 1 所示，IntelliJ IDEA 主题字体的修改要先勾选 Override default fonts by 。默认 IntelliJ IDEA 是不推荐修改的，但是由于字体是有分包含中文和不包含中文之分的，一般使用英文的国家是不需要额外担心乱码问题的，而我们需要。

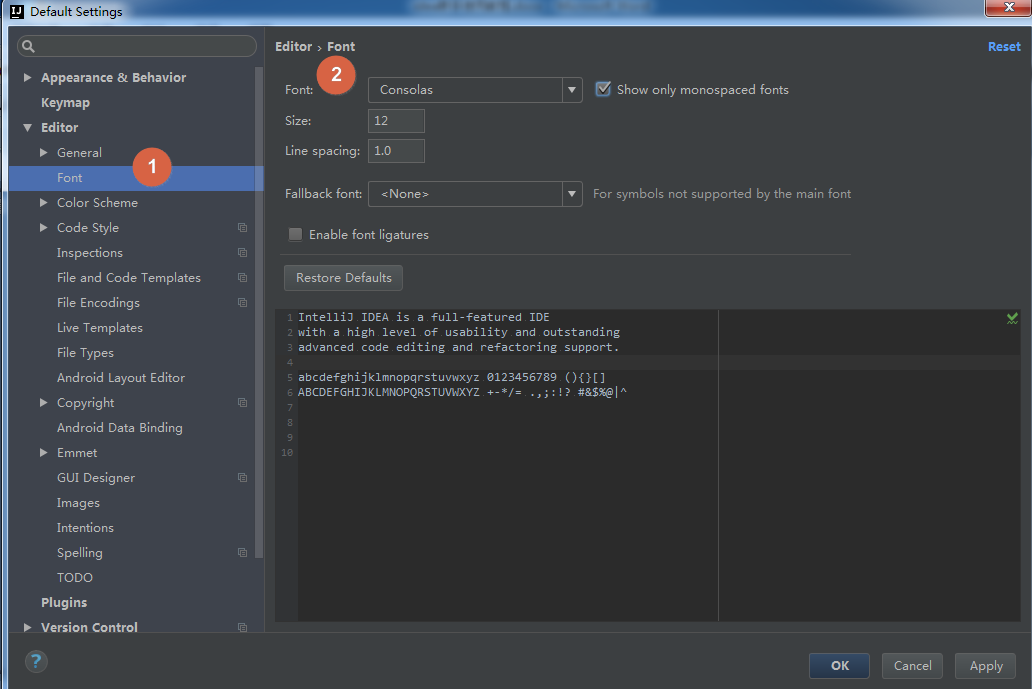
字体的审美上每个人不一样，但是如上一段说的，这里的字体修改是需要知道一个前提的，那就是你选择的那个字体必须含有中文，比如微软雅黑和宋体这类是包含中文的，而 Courier New 和 Monaco 这类只是单纯的英文字体。

如果你选择的字体不包含中文，那可能会在很多位置上出现类似 口口口口口 这样的乱码问题，比如文件名含有中文、字体是中文名字的都会变成 口口口口口。

在修改 IntelliJ IDEA 的主题字体的时候，不建议把字体调成很大，因为很多人遇到这样一种情况：显示器分辨率低，主题字体又大，在 IntelliJ IDEA 的某些操作的工具菜单、右键菜单选项中部分选项超出了分辨率显示范围，没办法被选中。当然了，如果你一定要把字体改大，又不用大分辨率显示器，那可以通过 IntelliJ IDEA 的 Menus and Toolbars 删除部分你认为用不到的菜单，但是一般不建议这样做。

还需要特别注意的时候，如果你是开着 IntelliJ IDEA 的时候，新装了一个字体的话，那必须重启IntelliJ IDEA 之后才能在下拉列表找到新装的字体。

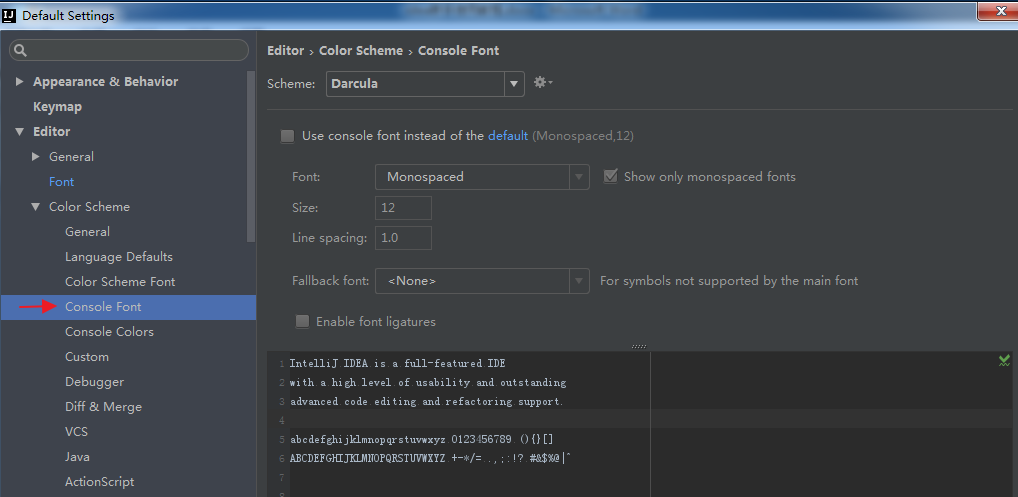
### 代码编辑字体修改



如上图标注 2 所示，勾选的 Show only monospaced fonts 表示筛选显示系统上的等宽字体。由于 Windows 系统上等宽字体并不多，勾选此选项出现的下拉字体可选择就很少。取消勾选之后，就可以显示系统上所有已安装的字体。

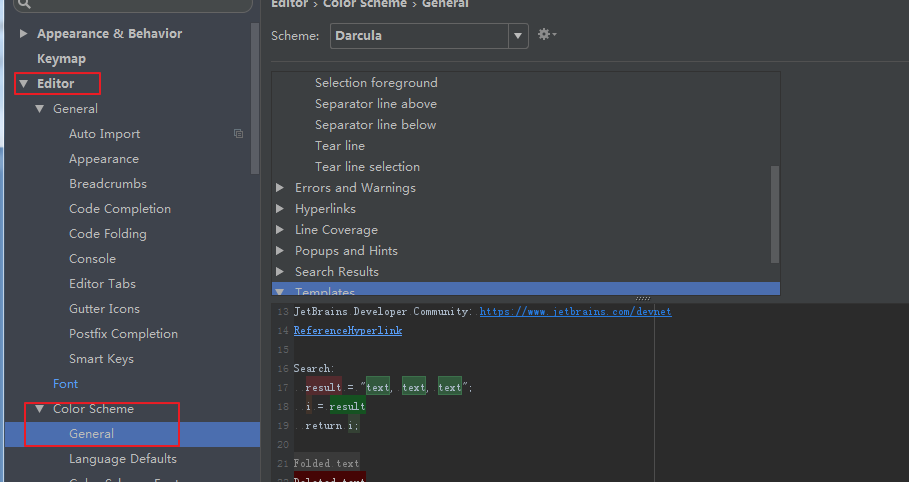
如果你的第一字体不包含中文的话，第二字体包含中文，那在有些地方也还是会出现 口口口口口 这类问题，比如 Ctrl + Shift + N 进行查找文件的时候，如果你输入中文也会变成 口口口口口 ，我个人文件名为中文的不多，所以就容忍了这种情况。如果你不愿意容忍这种情况，那还是回到最开始的要求：第一字体包含中文。

### 控制台输出字体修改



具体说明同上

## 编辑区主题修改

在 General 区，我常修改的有：

Default text，指的是默认代码文本，我一般会修改其 Background 属性。

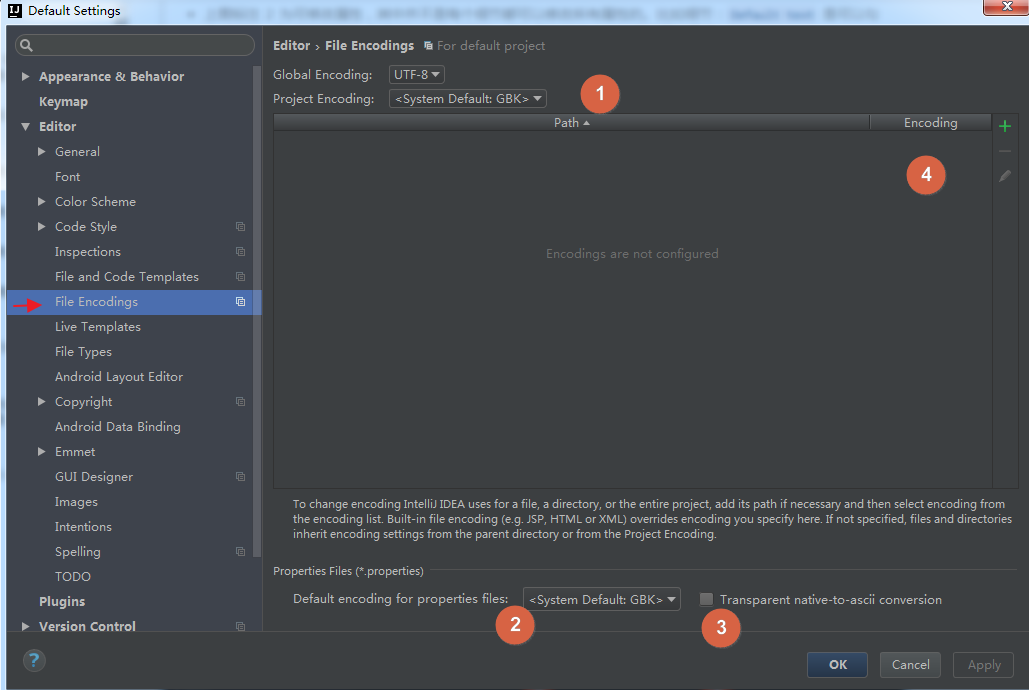
Caret row，指的是光标所在行，我一般会修改其 Background 属性。

Vertical indent guide，指的是垂直缩进线，我一般会修改其 Foreground 属性。

Identifier under caret，指的是光标所在位置的相同标识符呈现什么效果，我一般会修改其 Background 属性。

Text search result，指的是在查找模式下，匹配字符的样式，我一般会修改其 Background 属性。

## 文件编码修改



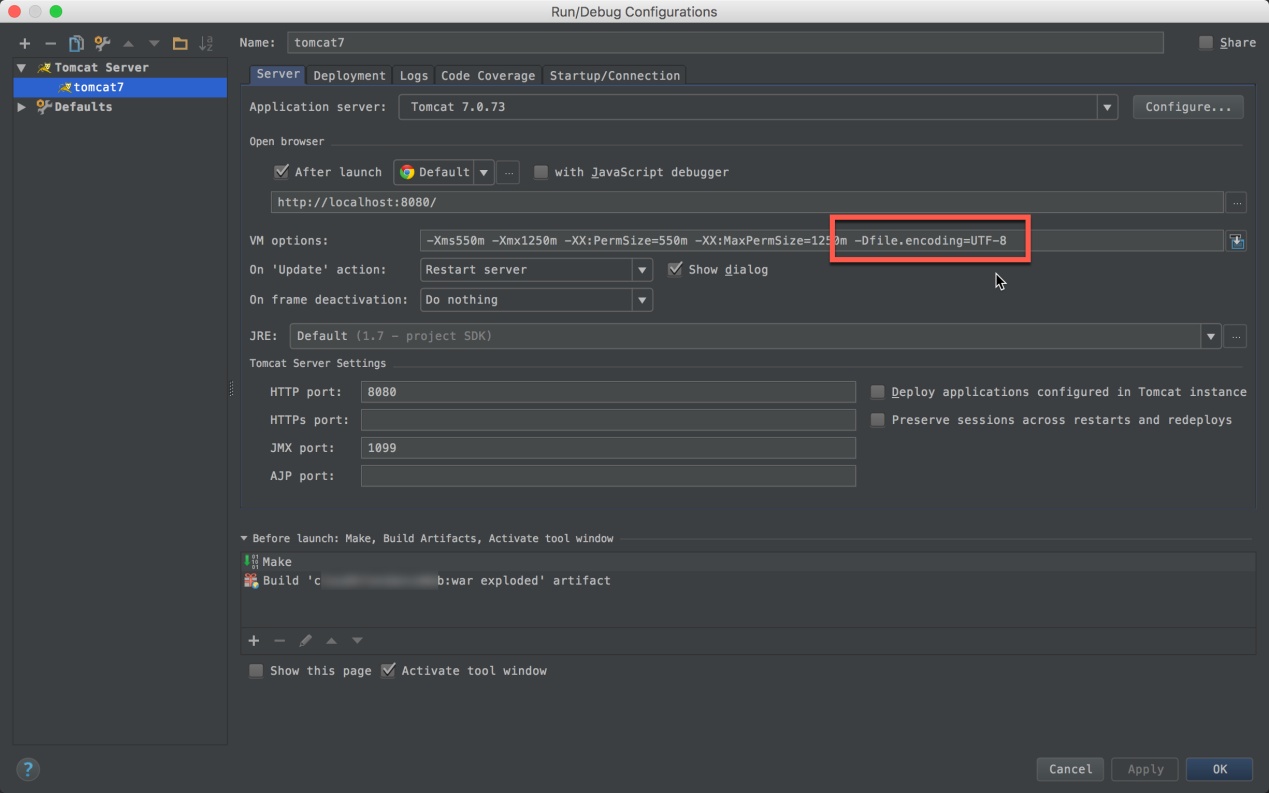
上图标注 1 所示，IDE 的编码默认是 UTF-8，Project Encoding 虽然默认是 GBK，但是一般我都建议修改为 UTF-8。

上图标注 2 所示，IntelliJ IDEA 可以对 Properties 文件进行专门的编码设置，一般也建议改为 UTF-8，其中有一个重点就是属性 Transparent native-to-ascii conversion，

上图标注 3 所示，对于 Properties 文件，重要属性 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。

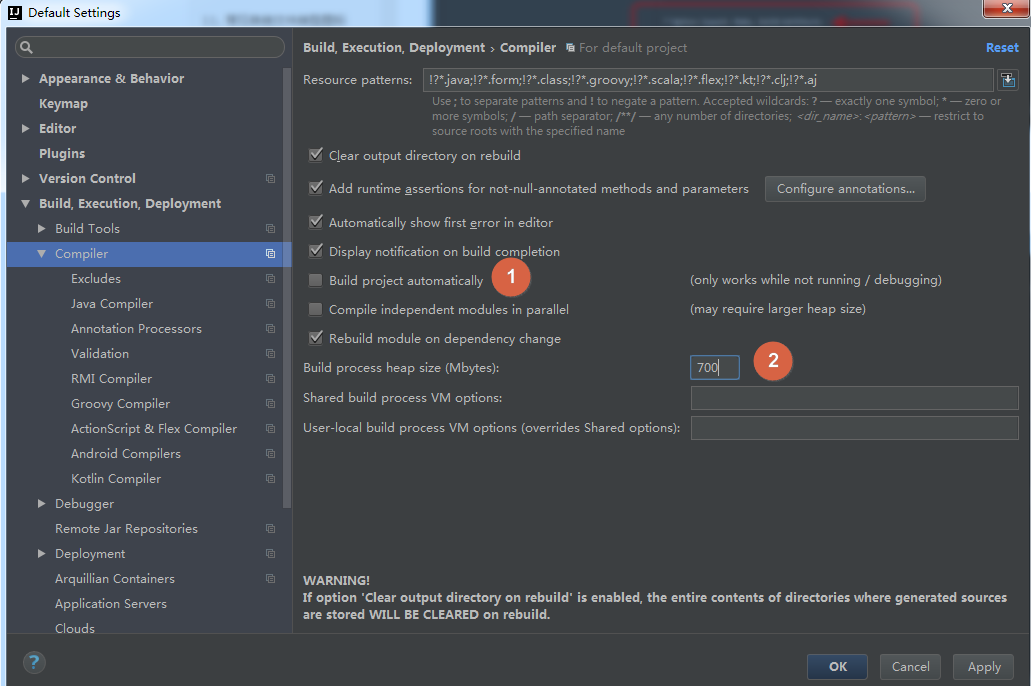
上图标注 4 所示，IntelliJ IDEA 除了支持对整个 Project 设置编码之外，还支持对目录、文件进行编码设置。如果你要对目录进行编码设置的话，可能会出现需要 Convert 编码的弹出操作选择，**强烈建议** 在转换之前做好文件备份，不然可能出现转换过程变成乱码，无法还原。

## Tomcat 控制台输出乱码

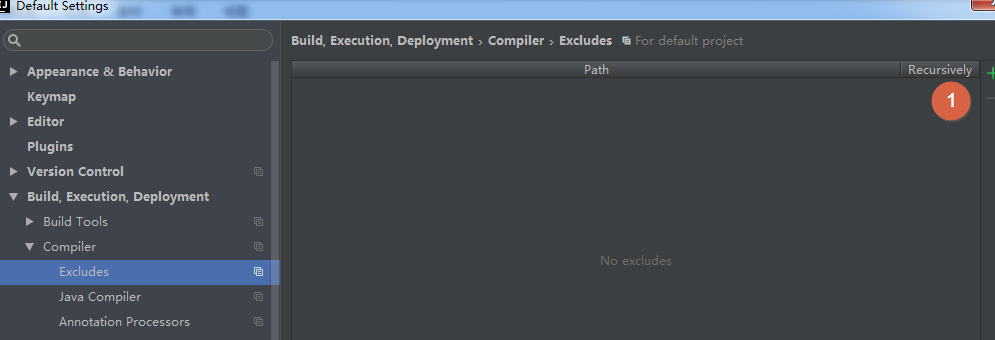
如果你的 Tomcat 控制台输出乱码，并且你已经保证了本文上面的控制台字体设置你设置的字体包含中文，那你还可以尝试下在 Tomcat 的 VM 参数上加上：-Dfile.encoding=UTF-8

如果你是 Mac 系统，很有可能是需要的。

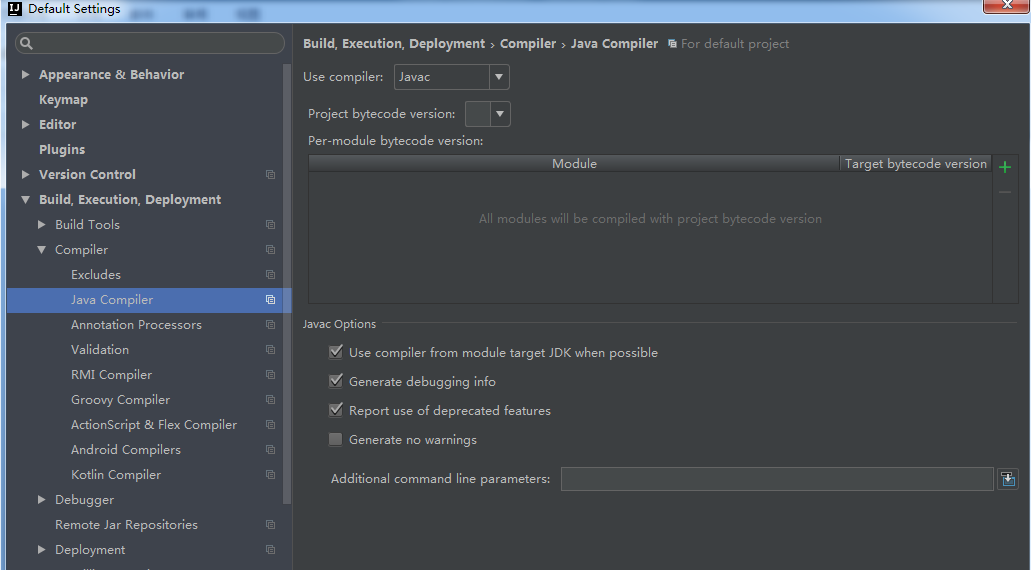
## 编译器的设置和选择

上图标注 1 所示，也是我们本文前面讲的，IntelliJ IDEA 是支持自动编译的，默认是不开启的，也建议不用开启，原因前面已经说了。

上图标注 2 所示，设置编译 heap 大小，默认是 700，建议使用 64 位的用户，在内存足够的情况下，建议改为 1500 或以上。如果你在编译的时候出错，报：OutOfMemoryError，一般也是要来改这个地方。

如上图标注 1 所示，可以添加目录 或 文件进行编译排除。

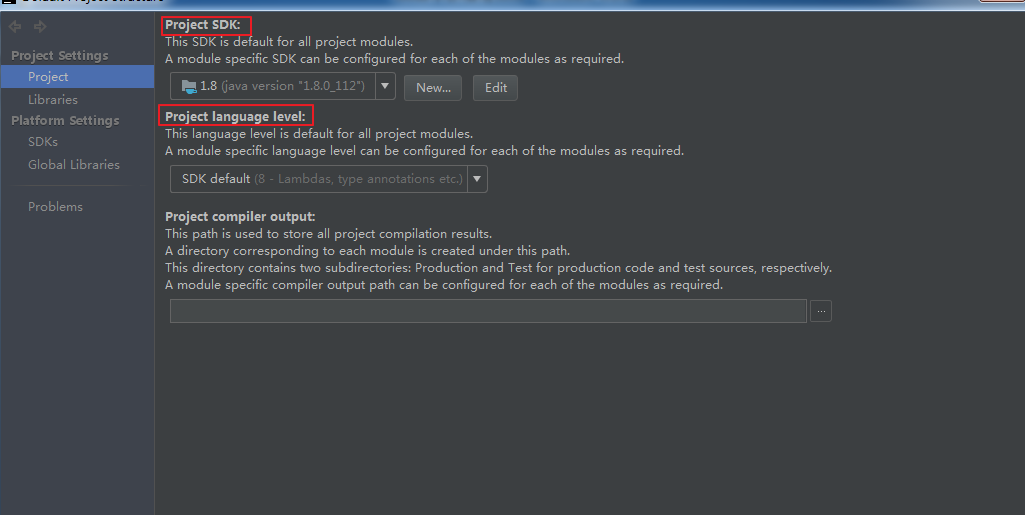
在项目中，如果有任何一个可编译的文件无法编译通过，则 IntelliJ IDEA 是无法运行起来的，必须等你全部问题解决，编译通过之后才可运行。但是可能开发过程中，某一个包目录的文件编译无法通过，但是我们又不急着改，那我们就可以考虑把该包加入到排除编译列表中，则项目就可以运行起来。

如上图动态 Gif 所示，IntelliJ IDEA 支持常见的几种编译器：Javac、Eclipse、Ajc 等。默认是 Javac，也推荐使用 Javac。

Project bytecode version 针对项目字节码编译版本，一般选择的是当前项目主 JDK 的版本。

Per-module bytecode version 可以针对 Project 下各个 Module 的特殊需求单独设置不同的 bytecode version，前提是电脑上必须有安装对应的 JDK 版本。

# SDK的设置

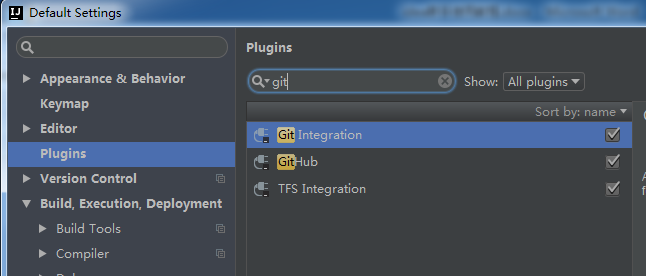


当我们使用 JDK 8 的时候，我们只能向下兼容 JDK 8 及其以下的特性，所以只能选择 8 及其以下的 language level。所以当我们项目使用的是 JDK 8，但是代码却没有使用 JDK 8 的新特性，最多使用了 JDK 7 的特性的时候我们可以选择 7 - Diamonds，ARM，multi-catch etc.。

对此我们总结 language level：限定项目编译检查时最低要求的 JDK 特性。

现在假设我们有一个项目代码使用的 JDK 8 新特性：lambda 语法，但是 JDK 选择的却是 JDK 7，即使 language level 选择了 8 - Lambdas，type annotation etc.，也是没有多大意义的，一样会编译报错。

# 版本控制的使用

很多人认为 IntelliJ IDEA 自带了 SVN 或是 Git 等版本控制工具，认为只要安装了 IntelliJ IDEA 就可以完全使用版本控制应有的功能。这完全是一种错误的解读，IntelliJ IDEA 是自带对这些版本控制工具的支持插件，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。旗舰版默认支持目前主流的版本控制软件：CVS、Subversion（SVN）、Git、ClearCase、Mercurial、Perforce、TFS。又因为目前太多人使用 Github 进行协同或是项目版本管理，所以 IntelliJ IDEA 同时自带了 Github 插件，方便 Checkout 和管理你的 Github 项目。

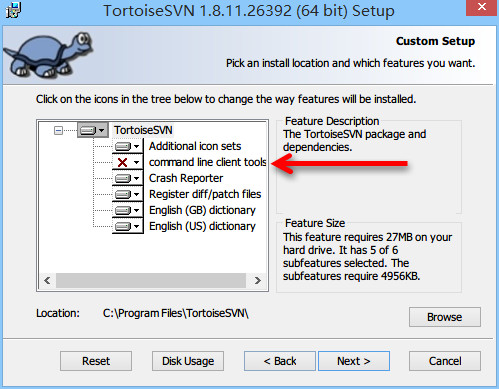
## SVN 的配置

要在 IntelliJ IDEA 中使用 SVN，需要先安装 SVN 客户端或是 TortoiseSVN 这类图形化工具，Windows 系统这里推荐安装 TortoiseSVN，即使在不使用 IntelliJ IDEA 也可以方便管理我们的项目。

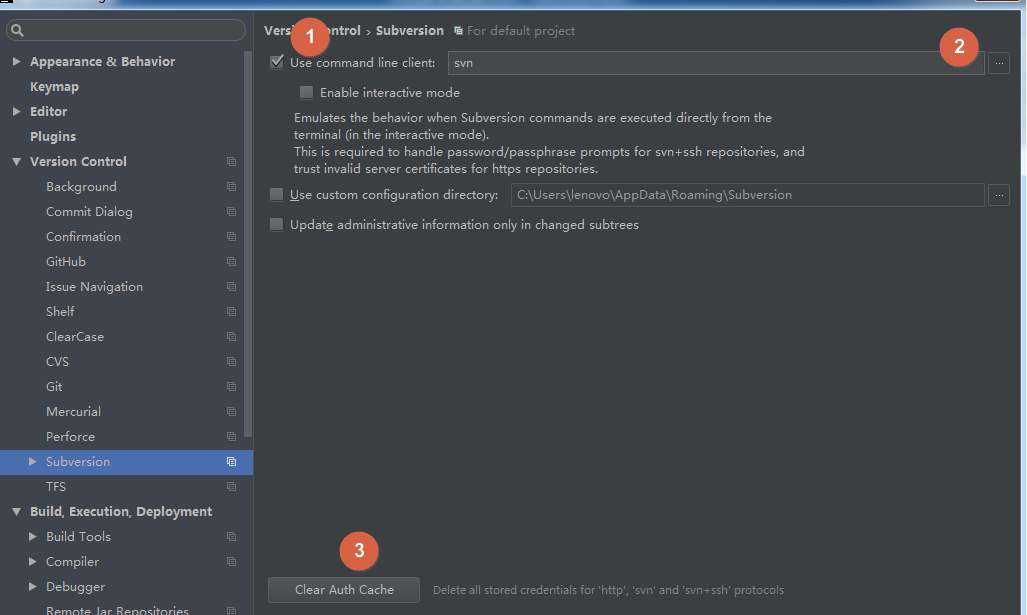
SVN 主要使用的版本有 1.6、1.7、1.8，最新的是 1.9。推荐大家使用 1.8 的。如果你的项目使用的是 1.6 的版本，在安装 1.8 之后是可以直接对项目文件进行升级的，所以无需担心，也因此更加推荐大家使用 1.8。

Subversion 官网下载：<https://subversion.apache.org/download/#recommended-release>

TortoiseSVN 官网下载：<http://tortoisesvn.net/downloads.zh.html>



如上图箭头所示，在安装 TortoiseSVN 的时候，默认 command line client tools，是不安装的，这里建议勾选上（否则bin目录下没有svn.exe文件）。



如上图标注 1 所示，勾选 Use command line client

如上图标注 2 所示，建议 svn 的路径自己根据安装后的路径进行选择（选到安装目录/bin/svn.exe），不然有时候 IntelliJ IDEA 无法识别到会报：Cannot run program "svn" 这类错误。

如上图标注 3 所示，当使用一段时间 SVN 以后，发现各种 SVN 相关问题无法解决，可以考虑点击此按钮进行清除一下缓存。（会同时清除掉账号等信息）

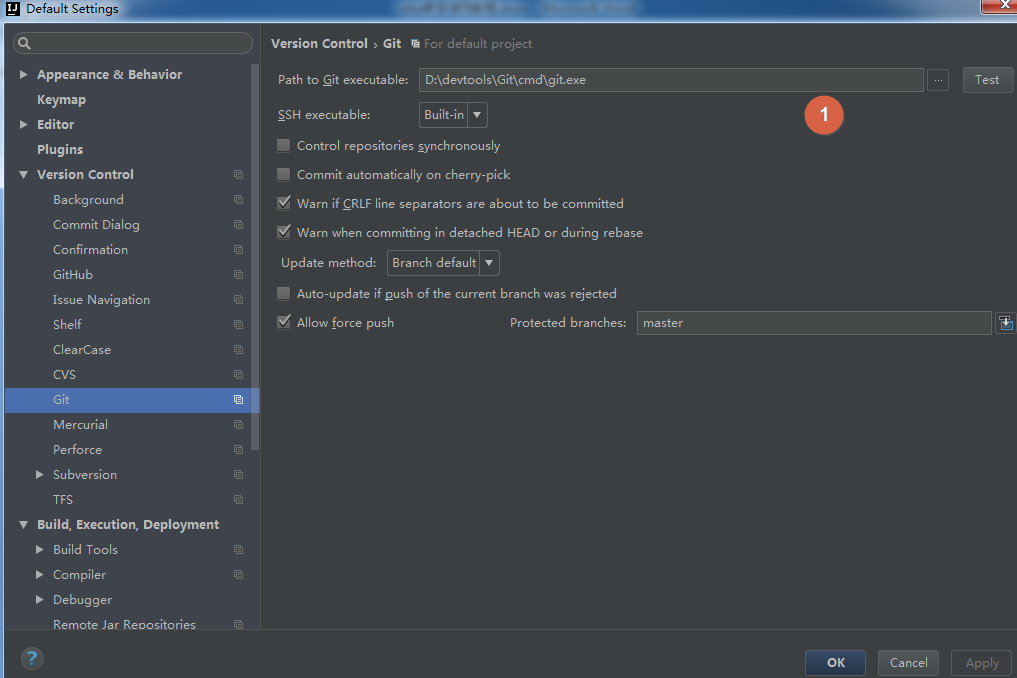
## Git 的配置

要在 IntelliJ IDEA 中使用 Git，需要先安装 Git 客户端，这里推荐安装官网版本。

Git 主要的版本有 1.X、2.X，最新的是 2.X，使用版本随意，但是不要太新了，不然可能 IntelliJ IDEA 小旧版本会无法支持可能。

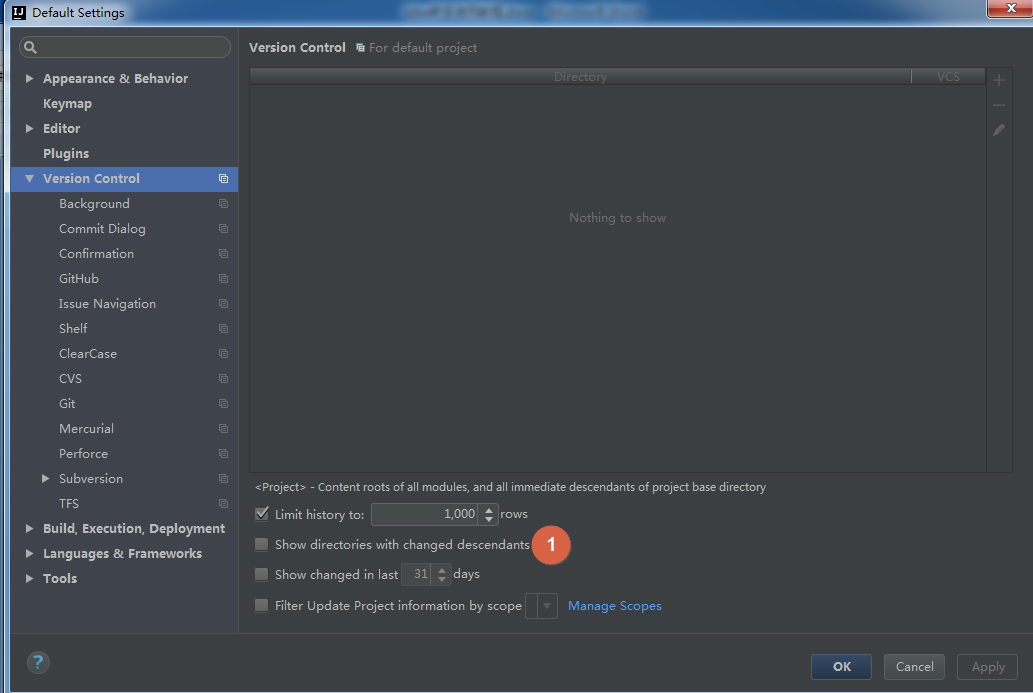
Git 官网下载：<http://git-scm.com/>

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

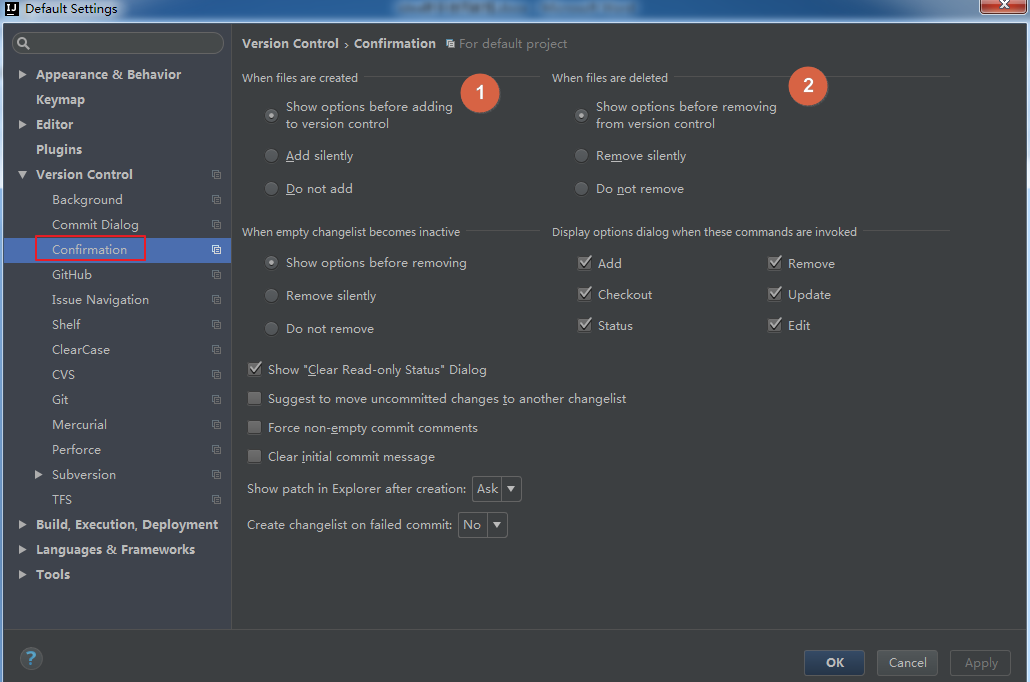


如上图标注 1 所示，确定好该路径下是否有对应的可执行文件

## 版本控制相关的常用设置说明



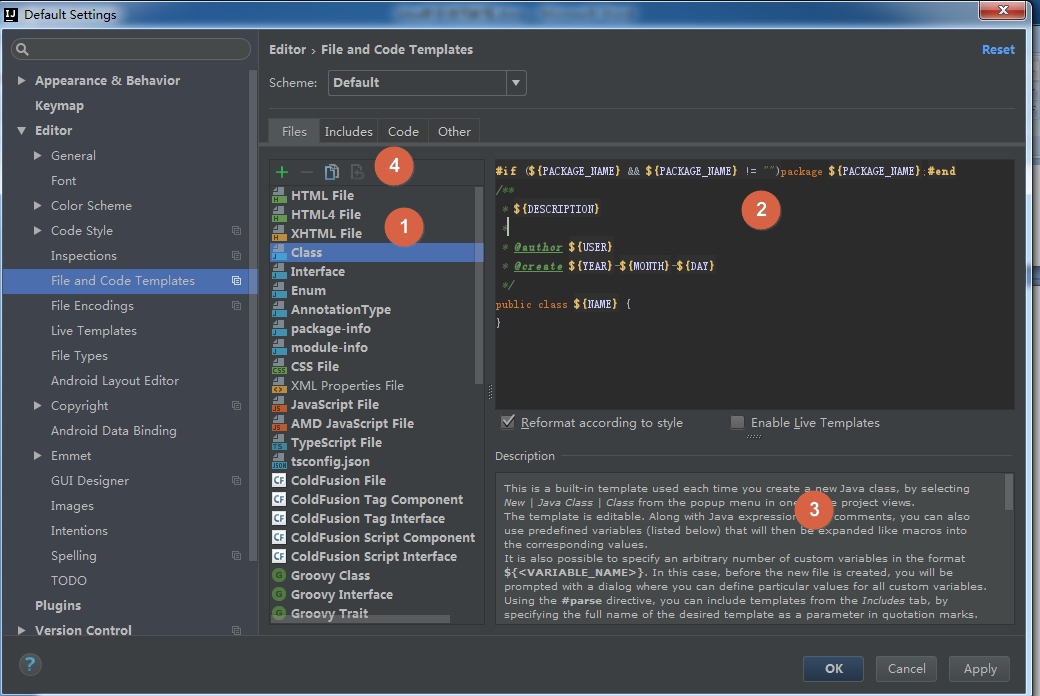
上图标注 1 所示，Show directories with changed descendants 表示子目录有文件被修改了，则该文件的所有上层目录都显示版本控制被修改的颜色。默认是不勾选的，我一般建议勾选此功能。

如上图标注 1 所示，When files are created 表示当有新文件放进项目中的时候 IntelliJ IDEA 做如何处理，默认是 Show options before adding to version control 表示弹出提示选项，让开发者决定这些新文件是加入到版本控制中还是不加入。如果不想弹出提示，则选择下面两个选项进行默认操作。

如上图标注 2 所示，When files are deleted 表示当有新文件在项目中被删除的时候 IntelliJ IDEA 做如何处理，默认是 Show options before removing from version control 表示弹出提示选项，让开发者决定这些被删除的是否从版本控制中删除。如果不想弹出提示，则选择下面两个选项进行默认操作。

# 实时代码模板

# 文件代码模板

如上图标注 1 所示，文件代码模板支持的类型基本常见的文件类型都涵盖了。

如上图标注2 所示，这是 Java 文件模板新建的代码模板，其中 ${PACKAGE\_NAME}、${NAME} 是 IntelliJ IDEA 预设的变量。

如上图标注 3 所示，介绍当前文件模板的一些信息，以及一些预设变量的作用介绍。

如上图标注 4 所示，这四个按钮是文件代码模板的主要操作按钮，四个操作分别是：

Create Template 创建一个文件代码模板。

Remove Template 删除一个文件代码模板，标注 1 所示的这些预设模板是不允许删除的，只能能删除预设之外的新增的。

Copy Template 复制一个文件代码模板。

Reset To Default 对被修改的预设文件代码模板，还原到默认状态。

## 文件代码模板预设的变量

因为变量的命名太明了不过了，所以这里不多讲解，直接贴出官网的英文解释：

${PACKAGE\_NAME} - the name of the target package where the new class or interface will be created.

${PROJECT\_NAME} - the name of the current project.

${FILE\_NAME} - the name of the PHP file that will be created.

${NAME} - the name of the new file which you specify in the New File dialog box during the file creation.

${USER} - the login name of the current user.

${DATE} - the current system date.

${TIME} - the current system time.

${YEAR} - the current year.

${MONTH} - the current month.

${DAY} - the current day of the month.

${HOUR} - the current hour.

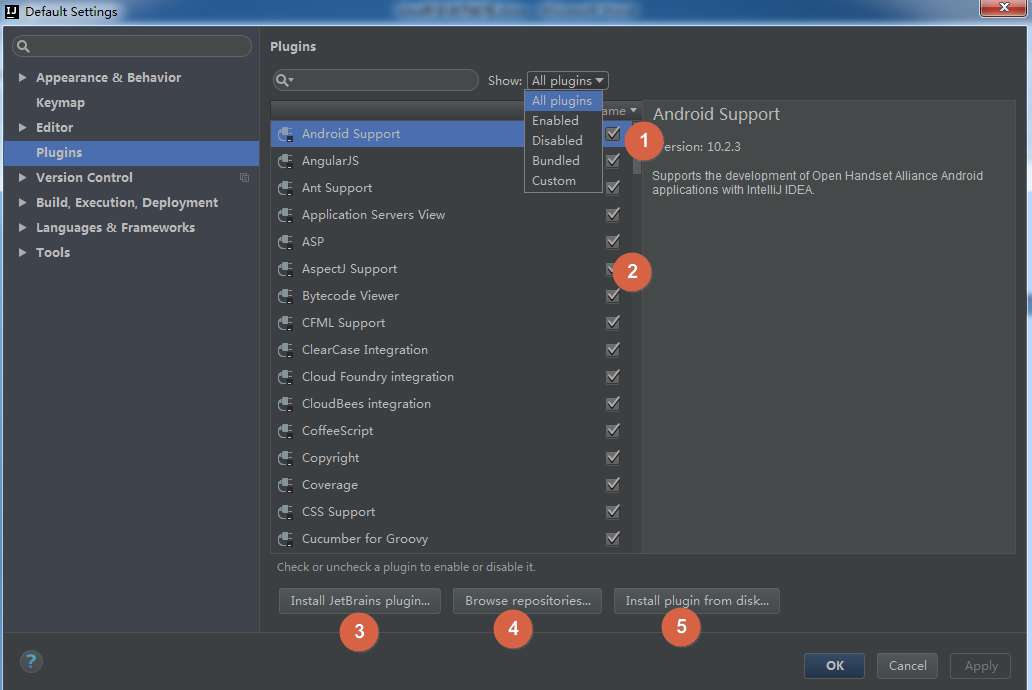
${MINUTE} - the current minute.

${PRODUCT\_NAME} - the name of the IDE in which the file will be created.

${MONTH\_NAME\_SHORT} - the first 3 letters of the month name. Example: Jan, Feb, etc.

${MONTH\_NAME\_FULL} - full name of a month. Example: January, February, etc.

# 插件相关功能

如上图标注 1 所示，IntelliJ IDEA 对插件进行了很好的分类：

All plugins 显示所有插件。

Enabled 显示当前所有已经启用的插件。

Disabled 显示当期那所有已经禁用的插件。

Bundled 显示所有 IntelliJ IDEA 自带的插件。

Custom 显示所有我们自行安装的插件，如果你自己装了很多次插件的话，这个选项会用得比较多。

如上图标注 2 所示，启用的插件是被勾选的，如果要禁用一个插件，去掉勾选即可。如上图标注 3 所示，弹出 IntelliJ IDEA 公司自行开发的插件仓库列表，供下载安装。如上图标注 4 所示，弹出插件仓库中所有插件列表供下载安装。

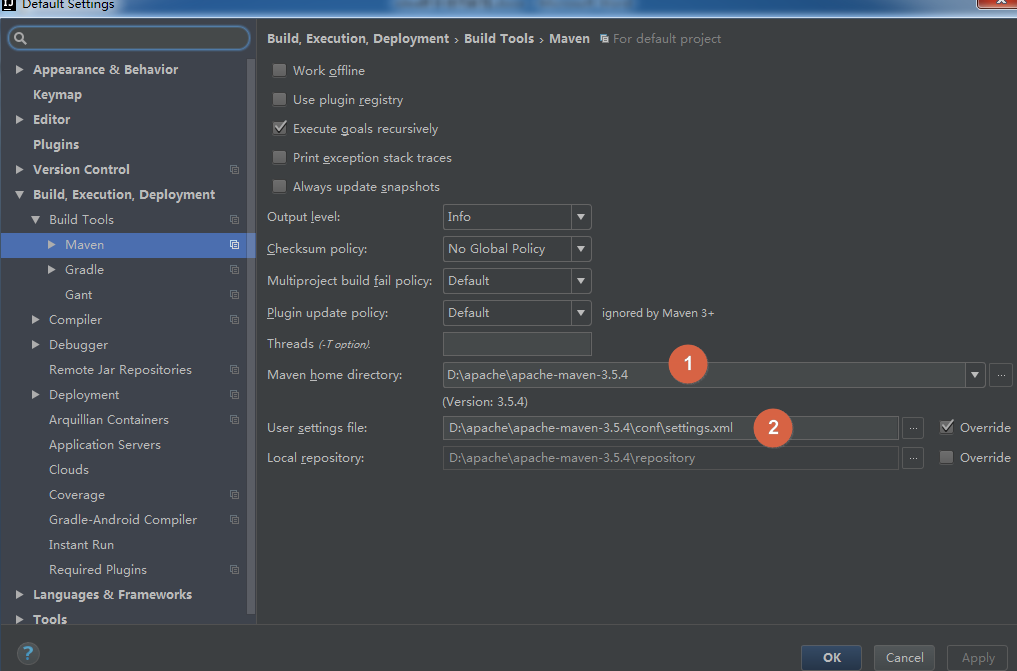
如上图标注 5 所示，浏览本地的插件文件进行安装，而不是从服务器上下载并安装。

需要严重注意的是：在国内的网络下，很经常出现显示不了插件列表，或是显示了插件列表，无法下载完成安装。这时候请自行开VPN，一般都可以得到解决。

## 插件推荐

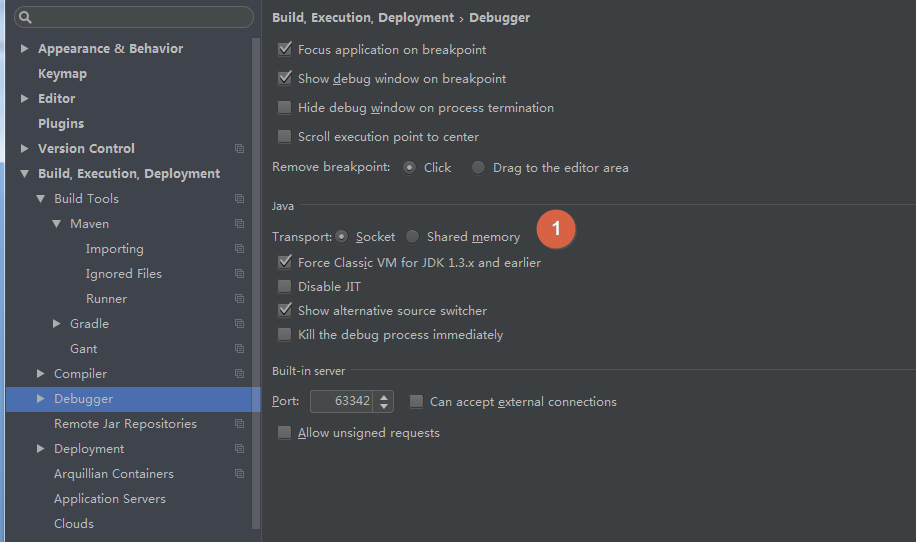
| 插件名称 | 插件介绍 | 官网地址 |
| --- | --- | --- |
| Gitee | 开源中国的码云插件 | <https://plugins.jetbrains.com/plugin/8383-gitee> |
| Alibaba Java Coding Guidelines | 阿里巴巴出的代码规范检查插件 | <https://plugins.jetbrains.com/plugin/10046-alibaba-java-coding-guidelines> |
| IDE Features Trainer | IntelliJ IDEA 官方出的学习辅助插件 | <https://plugins.jetbrains.com/plugin/8554?pr=idea> |
| Key promoter | 快捷键提示 | <https://plugins.jetbrains.com/plugin/4455?pr=idea> |
| Grep Console | 自定义设置控制台输出颜色 | <https://plugins.jetbrains.com/idea/plugin/7125-grep-console> |
| String Manipulation | 驼峰式命名和下划线命名交替变化 | <https://plugins.jetbrains.com/plugin/2162?pr=idea> |
| CheckStyle-IDEA | 代码规范检查 | <https://plugins.jetbrains.com/plugin/1065?pr=idea> |
| FindBugs-IDEA | 潜在 Bug 检查 | <https://plugins.jetbrains.com/plugin/3847?pr=idea> |
| MetricsReloaded | 代码复杂度检查 | <https://plugins.jetbrains.com/plugin/93?pr=idea> |
| Statistic | 代码统计 | <https://plugins.jetbrains.com/plugin/4509?pr=idea> |
| JRebel Plugin | 热部署 | <https://plugins.jetbrains.com/plugin/?id=4441> |
| CodeGlance | 在编辑代码最右侧，显示一块代码小地图 | <https://plugins.jetbrains.com/plugin/7275?pr=idea> |
| GsonFormat | 把 JSON 字符串直接实例化成类 | <https://plugins.jetbrains.com/plugin/7654?pr=idea> |
| Markdown Navigator | 书写 Markdown 文章 | <https://plugins.jetbrains.com/plugin/7896?pr=idea> |
| Eclipse Code Formatter | 使用 Eclipse 的代码格式化风格，在一个团队中如果公司有规定格式化风格，这个可以使用。 | <https://plugins.jetbrains.com/plugin/6546?pr=idea> |
| Jindent-Source Code Formatter | 自定义类、方法、doc、变量注释模板 | <http://plugins.jetbrains.com/plugin/2170?pr=idea> |
| Translation | 翻译插件 | <https://github.com/YiiGuxing/TranslationPlugin> |
| Maven Helper | Maven 辅助插件 | <https://plugins.jetbrains.com/plugin/7179-maven-helper> |
| Properties to YAML Converter | 把 Properties 的配置格式改为 YAML 格式 | <https://plugins.jetbrains.com/plugin/8000-properties-to-yaml-converter> |
| Git Flow Integration | Git Flow 的图形界面操作 | <https://plugins.jetbrains.com/plugin/7315-git-flow-integration> |
| Rainbow Brackets | 对各个对称括号进行着色，方便查看 | <https://github.com/izhangzhihao/intellij-rainbow-brackets> |
| MybatisX | mybatis 框架辅助（免费） | <https://plugins.jetbrains.com/plugin/10119-mybatisx> |
| Lombok Plugin | Lombok 功能辅助插件 | <https://plugins.jetbrains.com/plugin/6317-lombok-plugin> |
| .ignore | 各类版本控制忽略文件生成工具 | <https://plugins.jetbrains.com/plugin/7495--ignore> |
| mongo4idea | mongo客户端 | <https://github.com/dboissier/mongo4idea> |
| iedis | redis客户端 | <https://plugins.jetbrains.com/plugin/9228-iedis> |
| GenerateAllSetter | new POJO类的快速生成 set 方法 | <https://plugins.jetbrains.com/plugin/9360-generateallsetter> |

# Maven项目介绍



# Debug介绍

## Debug 设置

如上图标注 1 所示，表示设置 Debug 连接方式，默认是 Socket。Shared memory 是 Windows 特有的一个属性，一般在 Windows 系统下建议使用此设置，相对于 Socket 会快点。

## Debug 常用快捷键

| 快捷键 | 介绍 |
| --- | --- |
| F7 | 在 Debug 模式下，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，如果该方法体还有方法，则不会进入该内嵌的方法中 必备 |
| F8 | 在 Debug 模式下，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 必备 |
| F9 | 在 Debug 模式下，恢复程序运行，但是如果该断点下面代码还有断点则停在下一个断点上 必备 |
| Alt + F8 | 在 Debug 的状态下，选中对象，弹出可输入计算表达式调试框，查看该输入内容的调试结果 必备 |
| Ctrl + F8 | 在 Debug 模式下，设置光标当前行为断点，如果当前已经是断点则去掉断点 |
| Shift + F7 | 在 Debug 模式下，智能步入。断点所在行上有多个方法调用，会弹出进入哪个方法 |
| Shift + F8 | 在 Debug 模式下，跳出，表现出来的效果跟 F9 一样 |
| Ctrl + Shift + F8 | 在 Debug 模式下，指定断点进入条件 |
| Alt + Shift + F7 | 在 Debug 模式下，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，如果方法体还有方法，则会进入该内嵌的方法中，依此循环进入 |
| Drop Frame | 这个不是一个快捷键，而是一个 Debug 面板上的按钮。该按钮可以用来退回到当前停住的断点的上一层方法上，可以让过掉的断点重新来过 |

## Debug 特殊技能使用

### 查看所选对象的方法常用有三种方式：

选中对象后，使用快捷键 Alt + F8。

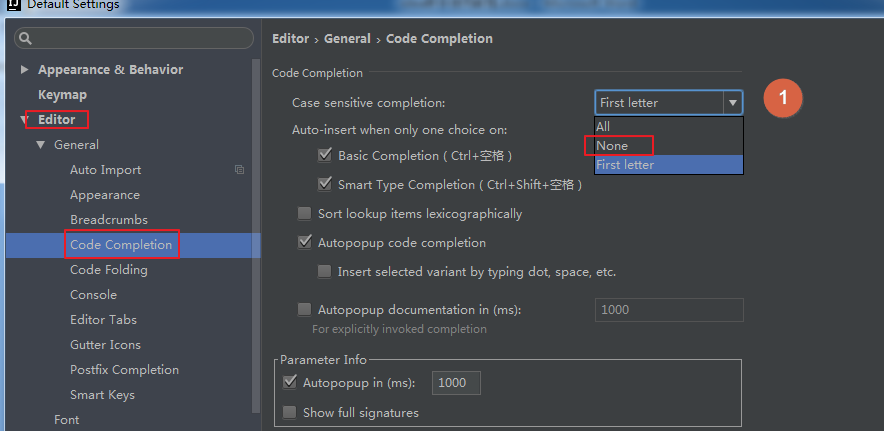
选中对象后，拖动对象到 Watches。

选中对象后，鼠标悬停在对象上 2 秒左右。

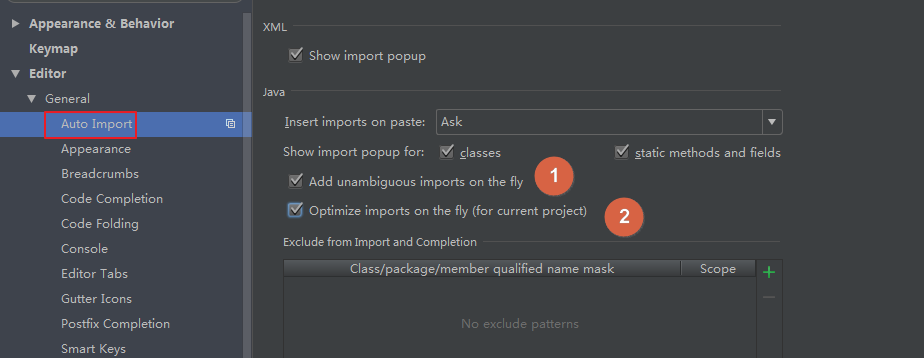
# IDIEA一些推荐设置和常用设置

## 提示区分大小写的设置

默认是区分大小的的（First letter），不想区分大小写选择None



## 自动导包

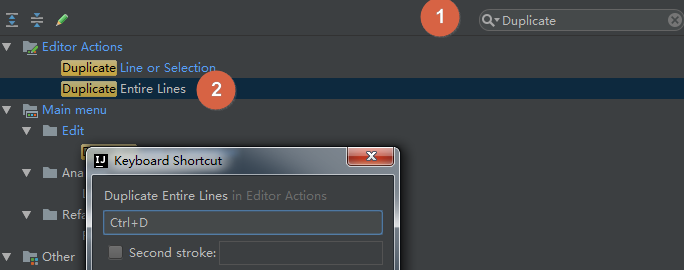


如上图标注 1 和 2 所示，默认 IntelliJ IDEA 是没有开启自动 import 包的功能。

勾选标注 1 选项，IntelliJ IDEA 将在我们书写代码的时候自动帮我们优化导入的包，比如自动去掉一些没有用到的包。

勾选标注 2 选项，IntelliJ IDEA 将在我们书写代码的时候自动帮我们导入需要用到的包。但是对于那些同名的包，还是需要手动 Alt + Enter 进行导入的，IntelliJ IDEA 目前还无法智能到替我们做判断。

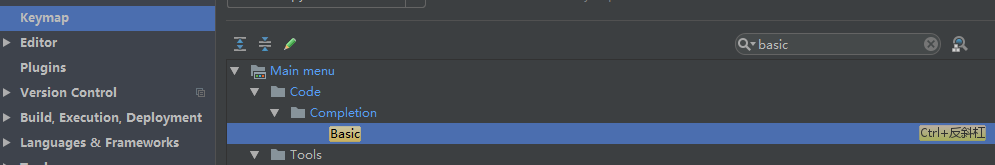
## ctrl+D快捷键优化



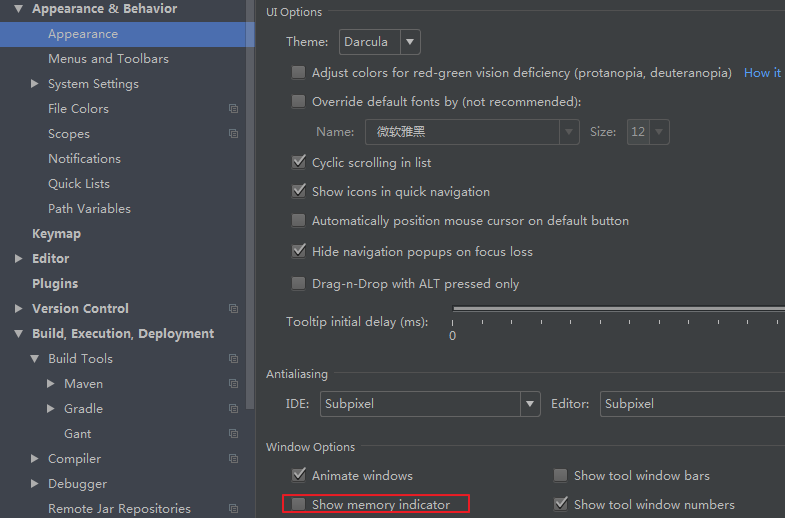
在图标1处输入Duplicate

在图标2处右键选择第一个，在出来的对话框中按 ctrl + D

## 提示快捷键的优化

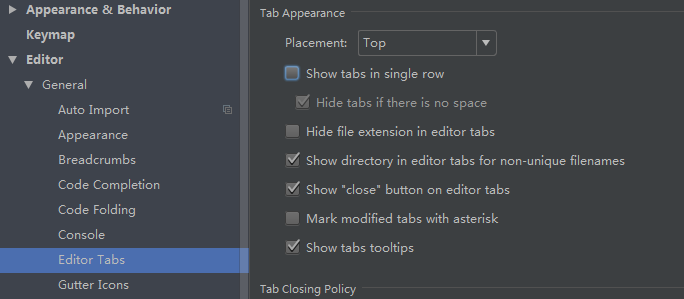
默认 Ctrl + 空格 快捷键是基础代码提示、补充快捷键，但是由于我们中文系统基本这个快捷键都被输入法占用了，所以我们发现不管怎么按都是没有提示代码效果的，原因就是在此。我个人建议修改此快捷键为 Ctrl + 逗号。

## idea 内存使用情况的显示



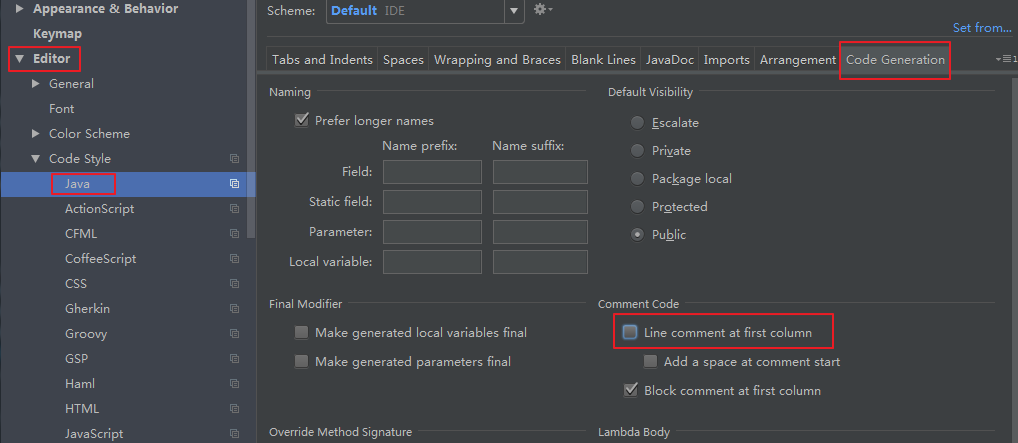
IntelliJ IDEA 14 以后版本默认是不显示内存使用情况的，对于大内存的机器来讲不显示也无所谓，但是如果是内存小的机器最好还是显示下。如上图演示，点击后可以进行部分内存的回收。

## Tab 单行显示优化

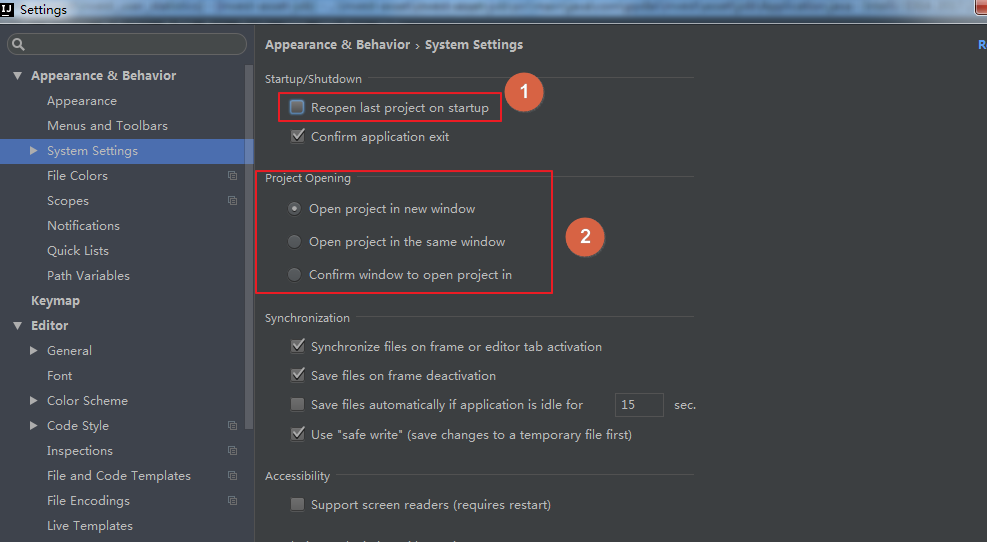


在打开很多文件的时候，IntelliJ IDEA 默认是把所有打开的文件名 Tab 单行显示的。但是我个人现在的习惯是使用多行，多行效率比单行高，因为单行会隐藏超过界面部分 Tab，这样找文件不方便

## 单行注释的优化



## 打开窗口的设置

如上图标注 1 所示，勾选此选项后，启动 IntelliJ IDEA 的时候，默认会打开上次使用的项目。如果你只有一个项目的话，该功能还是很好用的，但是如果你有多个项目的话，建议还是关闭，这样启动 IntelliJ IDEA 的时候可以选择最近打开的某个项目。

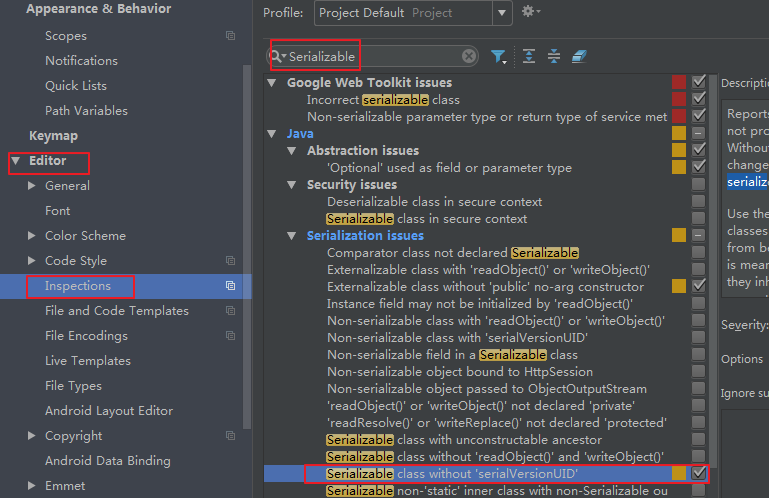
如上图红圈所示，该选项是设置当我们已经打开一个项目窗口的时候，再打开一个项目窗口的时候是选择怎样的打开方式。

Open project in new window 每次都使用新窗口打开。

Open project in the same window 每次都替换当前已打开的项目，这样桌面上就只有一个项目窗口。

Confirm window to open project in 每次都弹出提示窗口，让我们选择用新窗口打开或是替换当前项目窗口。

## 开启自动生成 serialVersionUID

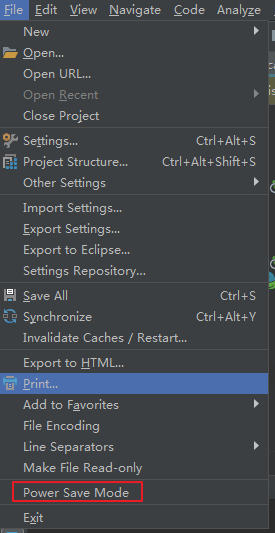
默认 IntelliJ IDEA 是没有开启自动帮你生成 serialVersionUID 的，需要我们自行设置。

如上图第一张，需要先勾选：Serializable class without serialVersionUID

如上图第二张，在已经继承了 Serializable 接口的类名上，把光标放在类名上（必须这样做），按 Alt + Enter，即可提示帮你生成 serialVersionUID 功能。

# IDEA常用细节了解

## 省电模式



如上图所示，IntelliJ IDEA 有一种叫做 省电模式 的状态，开启这种模式之后 IntelliJ IDEA 会关掉代码检查和代码提示等功能。所以一般我也会认为这是一种 阅读模式，如果你在开发过程中遇到突然代码文件不能进行检查和提示可以来看看这里是否有开启该功能。

## 包折叠和显示的问题



IntelliJ IDEA 默认是会折叠空包的，这样就会出现包名连在一起的情况。但是有些人不喜欢这种结构，喜欢整个结构都是完整树状的，所以我们可以去掉演示中的勾选框即可。