## 磨刀不误砍柴工—开发工具比较

搜索关键字

Intellij Idea；Eclipse

本章难点

**Eclipse**最初是由IBM捐献给开源社区的，目前已经发展成为人气最旺的Java IDE。Eclipse插件化的功能模块吸引了无数开发者开发基于Eclipse的功能插件，在这样一个健壮的社区环境下，Eclipse得到了无论是企业还是初学者的青睐。  
 **Intellij Idea**为JetBrains公司的产品，在业界被公认为最好的Java开发工具之一，尤其在智能代码助手、代码自动提示、重构代码审查、 创新的GUI设计等方面的功能可以说是超常的，也因此积累了一批特杆粉丝。  
本章将基于android开发环境，对他们进行比较并解剖分析二者的优点与缺点，让读者可以选择适合自己的IDE，提高开发效率。

### 1.项目简介

#### 1.1. UI界面

**Eclipse**的本身只是一个框架平台，但是众多插件的支持使得Eclipse拥有其他功能相对固定的IDE软件很难具有的灵活性。许多软件开发商以Eclipse为框架开发自己的IDE。

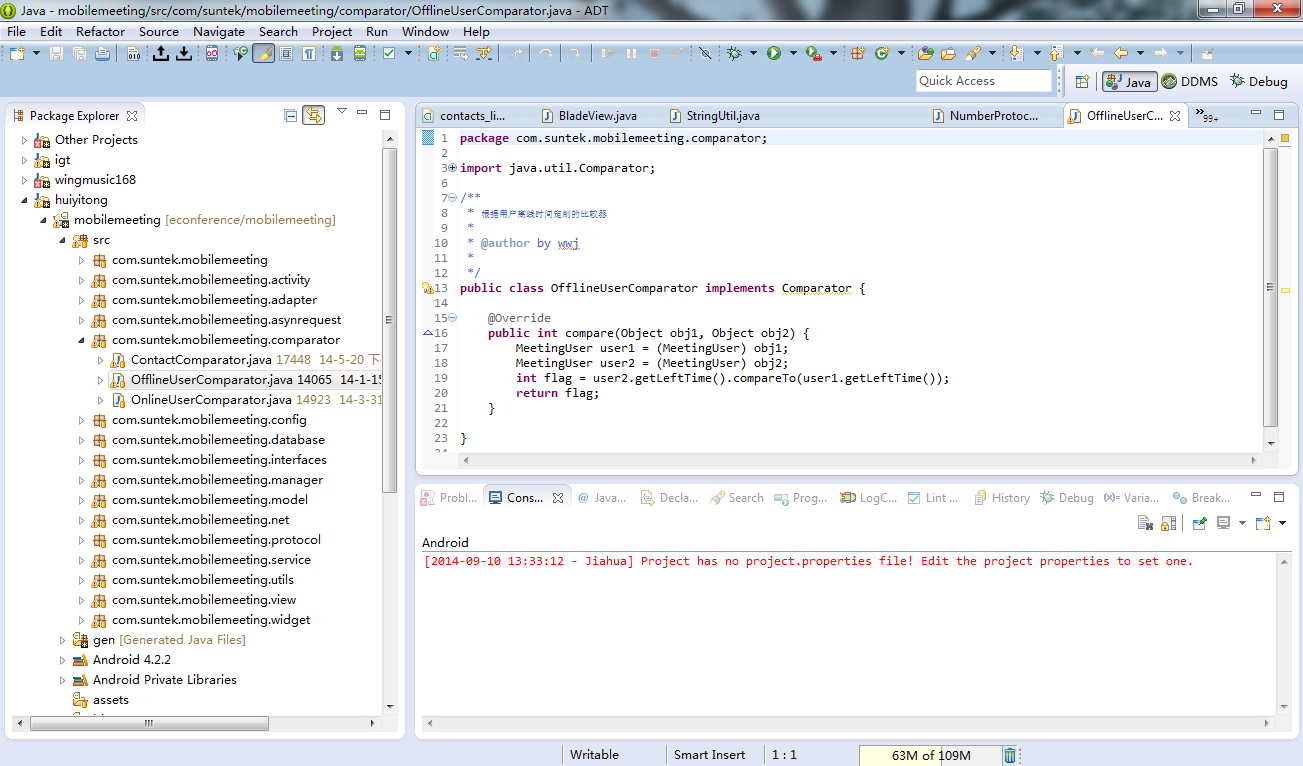
**Eclipse**最初是由IBM公司开发的替代商业软件Visual Age for Java的下一代IDE开发环境，2001年11月贡献给开源社区，现在它由非营利软件供应商联盟Eclipse基金会（Eclipse Foundation）管理。截止止今日，其仍然保持着比较频繁的更新频率，旨在为工程师打造更好用的IDE。

图1a-1 发行历史版本



界面方面虽然Eclipse没有Intellij Idea的华丽惊艳，但却给与一种务实稳重的感觉，相信下图的界面大家也比较熟悉了。

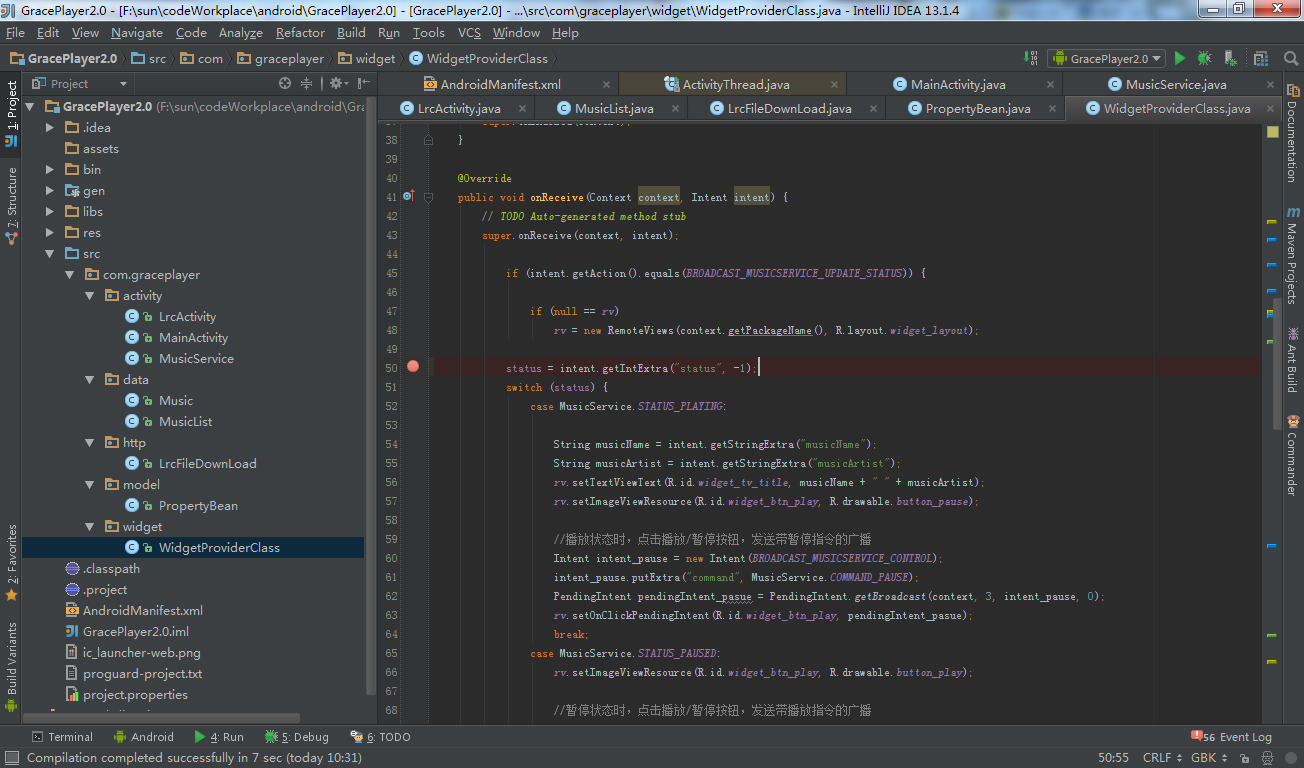
图1a-2 Eclipse默认主题的界面



**IntelliJ IDEA**(下面简称IDEA)被认为是当前Java开发效率最快的IDE工具。它整合了开发过程中实用的众多功能，几乎可以不用鼠标可以方便的完成你要做的任何事情，最大程度的加快开发的速度。简单而又功能强大。与其他的一些繁冗而复杂的IDE工具有鲜明的对比。

2001年1月发布IDEA 1.0版本，同年七月发布2.0，接下来基本每年发布一个版本(2003除外)，当然每年对各个版本都是一些升级。3.0版本之后，IDEA屡获大奖，其中又以2003年的赢得的“Jolt Productivity Award”，“ JavaWorld Editors’s Choice Award”为标志，从而奠定了IDEA在IDE中的地位。近日由谷歌推出的 Android Studio, 也建立在相同的基础之上  
目前版本为13.1，2014年7月22日发布，更新版本为13.1.4。IDEA的宗旨：“Develop with pleasure”。  
 初次使用IDEA，最具备吸引力的地方非它的UI界面莫属了，其中最著名的要数其自带的深蓝黑色为主调的Darcula主题，相信不少工程师对这种黑酷炫的颜色搭配有种独特的情怀。关于该主题的选择会在"常见问题"里进行详细说明。

图1a-3 IDEA的Darcula主题的界面

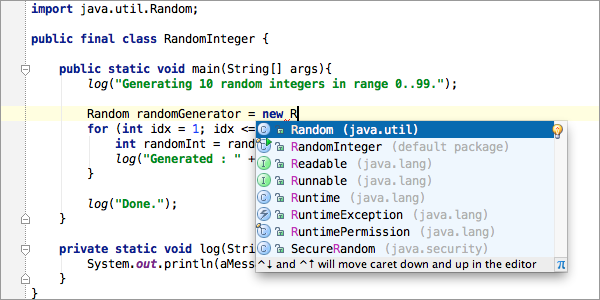


#### 1.2. 代码提醒

代码自动提示,智能补全是IDE与记事本的一个重要的区别之一，在这方面IDEA的表现就更强大，更智能了。它对代码的理解并不仅仅停留在【符合语法，编译无误】的层面，它对当前的上下文，相关的类、包、最近抛出过的异常、每种类型最近用过的变量、当前类有哪些成员哪些方法它们都是什么名字什么类型、上下文中所有驼峰变量名的首字母和大写字母……等等都有非常准确及时的了解和“猜测”，并且给予你最舒服最贴心的帮助。由于篇幅的问题下面就只简单介绍IDEA几个代码自动完成的特效。

##### 即刻完成 Instant completion

第一个也是最吸引我的就是“即刻完成”特性，不同于其他 IDE，IDEA 可在任意地方提供这个功能，而不只是当你要访问某个类的成员时。只需要输入单词的首字母，IntelliJ IDEA 就会立即给出最相关的、最适合此处代码编辑需要的选项共你选择。  
  
图1a-4 即刻完成



##### 链式自动完成 Chain completion

接下来的一个可让你更多的提升编码效率的自动完成特性，名为 链式自动完成(Chain completion)。区别于上文提到的**即时完成**，它是更智能地根据用户输入的信息进行索引，根据变量，属性和方法给出更深层次的代码提醒，具体到相关使用类里的具体的方法。

例如现在我们有个User类，该类为存储登陆用户的基本信息。当我需要调用该类的时候，需要用到该类的时候一般会先声明具体的对象，如下图。

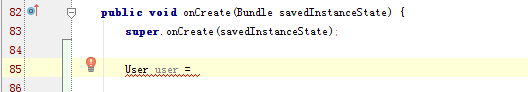
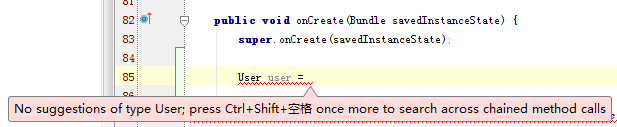


图1a-5声明对象

此时按下快捷键Ctrl + Shift + Space（也即智能的代码补全功能，在常见问题中会具体介绍该快捷键），会提示并没有找到相关的类，并且建议进行搜索更深一层的方法进行调用。

图1a-6 按下Ctrl + Shift + Space



接着再次敲下Ctrl + Shift + Space，IDEA立刻索引整个项目，并且快速给出返回User类的方法，也即下图MainApp.getUser()和MainApp.getUserAccount()。

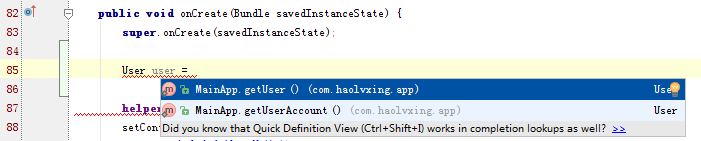
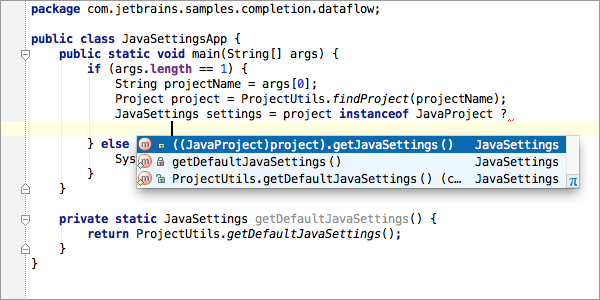


图1a-7 链式自动完成

##### 数据流分析 Data flow analysis

IntelliJ IDEA 被称为是最智能的 Java IDE 的原因是，IDEA 提供的各种功能都是基于对你代码分析的结果，数据流分析就是其中一项。IDEA 分析数据流以便猜测运行环境中的变量类型，并提供基于这个类型的自动完成以及自动增加类的强制类型转换。  
  
图1a-8 数据流分析



而IDEA之所以能够做到如此智能与人性化，很大程度上是因为它索引缓存了整个项目目录，这使得IDEA需要占用系统大量的内存资源，也就造成了当配置较低的电脑里使用IDEA会启动速度超慢，也非常卡，此时使用Eclipse就不会有这样的情况出现。

#### 1.3. 插件

虽然大多数用户很乐于将 Eclipse 当作 Java 集成开发环境（IDE）来使用，但 Eclipse 的目标却不仅限于此。Eclipse 还包括插件开发环境（Plug-in Development Environment，PDE），这个组件主要针对希望扩展 Eclipse 的软件开发人员，因为它允许他们构建与 Eclipse 环境无缝集成的工具。由于 Eclipse 中的每样东西都是插件，对于给 Eclipse 提供插件，以及给用户提供一致和统一的集成开发环境而言，所有工具开发人员都具有同等的发挥场所。  
因此Eclipse拥有非常丰富插件集群，如FindBugs,Checkstyle,PMD,SourceHelper等等，这是IDEA无法比拟的，目前IDEA官方公布的插件不足400个，并且许多插件实质性的东西并没有，在插件上的距离可见一斑。下面罗列了一些Eclipse常用的提高编写代码效率的插件:

1. **FindBugs**  
   FindBugs可以帮你找到Java代码中的bug，它使用Lesser GNU Public License的自由软件许可。
2. **Checkstyle**  
   Checkstyle插件可以集成到Eclipse IDE中去，能确保Java代码遵循标准代码样式。
3. **ECLemma**  
   ECLemma是一款拥有Eclipse Public License许可的免费工具，它提供了方便快捷的开发和测试环境。你可以使用代码覆盖模式下的“launch”功能，用起来就像是真正的运行/调试模式。你可以通过使用代码覆盖试图，高亮源文件，计数来分析代码。
4. **JDepend4Eclipse**  
   JDepend4Eclipse可以帮助你遍历文件夹，协助你量化设计的质量。它使用Eclipse Public License v1.0许可。
5. **PMD**  
   PMD是一款代码分析器，用来检测变量和写得不好的代码。（更新网址-[http://pmd.sourceforge.net/eclipse/）](http://pmd.sourceforge.net/eclipse/）" \t "_blank)
6. **SourceHelper**  
   SourceHelper可以协助编码和调试，对写好的代码提供说明。
7. **Structure101**  
   Structure101帮助修改代码的架构，就是说你可以改变架构，而不打乱代码。  
   文本编辑插件
8. **AnyEdit Tools**  
   AnyEdit为输出控制台和工具栏增加了新的编辑器，帮助导入和导出数据。它使用Eclipse Public License v1.0许可。
9. **Eclim**  
   Eclim给Eclipse引入了Vim的功能，Vim是最好的编辑器之一。开发者可以采用不同的语言来编写代码，它也提供bug纠错功能。Eclim使用GPLV3许可。
10. **Eclipse-rbe**  
    Eclipse-rbe用来编辑Java文件，它的功能有为缺失变量发出警告信息，排序键值，转变Unicode编码等。它使用GNU库或LGPL许可。  
    依赖管理
11. **Apache IvyDE**  
    Apache IvyDE集成了Apache Ivy的依赖管理功能。它能管理’ivy.xml’中的依赖，也能够配合WPT和Ant插件的使用。
12. **M2eclipse (Maven插件)**  
    M2eclipse能够管理简单的项目，也能管理多模块项目，它能在Eclipse中启动maven。它提供依赖管理，能提供自动下载。  
    版本控制插件
13. **Subclipse**  
    Subclipse是一个SVN插件，它为Eclipse IDE提供subversion支持，使用EPL 1.0许可。
14. **EGit**  
    EGit提供同步视图，可以读.git/ 下排除的文件，提供rebase功能，为pull和push提供精简的操作。
15. **MercurialEclipse**  
    MercurialEclipse是个流行的版本控制系统。它提供了clone repository功能，push-pull同步功能，以及简单的回滚功能。
16. **P4Eclipse**  
    P4Eclipse吸纳了Perforce管理系统的特长。它提供了以开发者为导向的一系列的功能，并且支持许多Agile开流程发以及传统开发流程。  
    框架开发插件
17. **Spring Tool Suite**  
    Spring Tool Suite提供了XML文件预览以及图形化的spring配置编辑器等开发环境和工具，它能让开发变得更简单。
18. **Spring IDE**  
    Spring IDE帮助你开发Spring应用，它提供了图形编辑界面。它还能用图形的方式显示bean之间的关系。（更新网址 – [http://springide.org/updatesite）](http://springide.org/updatesite）" \t "_blank)
19. **Hibernator**  
    Hibernator能够创建或更新数据库schema，运行hibernate查询语句，创建映射文档。用户可以将一个Java类和一个相关的映射文件同步起来。
20. **JbossTools**  
    JbossTools 3.2x支持JBoss，Hibernate，Drools，XHTML，Seam等。（更新网址-[http://download.jboss.org/jbosstools/updates/JBossTools-2.1.2.GA）](http://download.jboss.org/jbosstools/updates/JBossTools-2.1.2.GA）" \t "_blank)  
    其他功能插件
21. **ASM – 二进制预览插件**  
    ASM Bytecode Outline插件可以显示当前的Java文件或class文件的分解的二进制代码，便于Java/class文件的二进制比较，也可以显示当前二进制代码的ASMifier代码。你可以用Eclipse更新管理器来安装ASM。（更新网址-[http://download.forge.objectweb.org/eclipse-update/）](http://download.forge.objectweb.org/eclipse-update/）" \t "_blank)
22. **Mylyn – 任务管理**  
    Mylyn采用任务为中心的界面，它为开发者提供了任务管理工具。另外，也集成了富文本编辑工具，以及监测工具。
23. **Eclipse Launcher**  
    Eclipse launcher采用Delphi 6写成，使用它你可以自定义启动Eclipse IDE时的配置。
24. **AmaterasUML**  
    Eclipse的UML插件，支持UML活动图，class图，sequence图，usecase图等；支持与Java class/interface之间的相互导入导出。
25. **Log4E**  
    Log4E可以帮你更容易的写日志，而不需要受特定日志框架的约束。当然你也可以自定义你自己的日志模版。  
    总而言之，插件能让Eclipse用其他语言来编写。Eclipse插件是必不可少的，因为它们让Java应用无缝连接，降低了Java的复杂度。然而，装了过多的插件会让你的Eclipse变慢，所以你需要有选择性的安装插件。

我们简单得介绍其中的AmaterasUML插件.

在安装AmaterasUML之前，首先要按照GEF（Graphic Editing Framework）。其官方的文档地址是[http://wiki.eclipse.org/index.php/GEF\_Developer\_FAQ#Download\_and\_Install](http://wiki.eclipse.org/index.php/GEF_Developer_FAQ" \l "Download_and_Install" \t "_blank)

在线安装地址GEF - [https://hudson.eclipse.org/hudson/job/gef-maintenance/lastSuccessfulBuild/artifact/update-site/](https://hudson.eclipse.org/hudson/job/gef-maintenance/lastSuccessfulBuild/artifact/update-site/" \t "_blank)

在这里，可以通过连接，下载GEF的压缩包。下载好之后，将其拷贝的Eclipse的父目录下。然后，将其解压缩。其压缩包中，按照Eclipse的目录结构提供了plugins、features等目录。因此，如果在eclipse的父目录中进行解压缩，其中的内容会自动放入相应的文件中。

安装好GEF之后，在下载AmaterasUML的压缩包。其下载地址是：[http://amateras.sourceforge.jp/cgi-bin/fswiki\_en/wiki.cgi?page=AmaterasUML](http://amateras.sourceforge.jp/cgi-bin/fswiki_en/wiki.cgi?page=AmaterasUML" \t "_blank)  
 解压缩之后，将其jar包拷贝到Eclipse目录下的plugins文件夹中保存。

重启Eclipse，然后在File->New->Other下面就可以看见AmaterasUML的标志了。接着用户便可以再导出导入或者制作类图，时序图等。

图1a-9 数据流分析

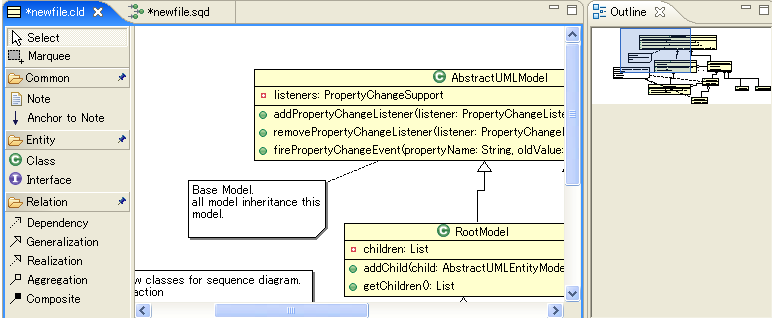
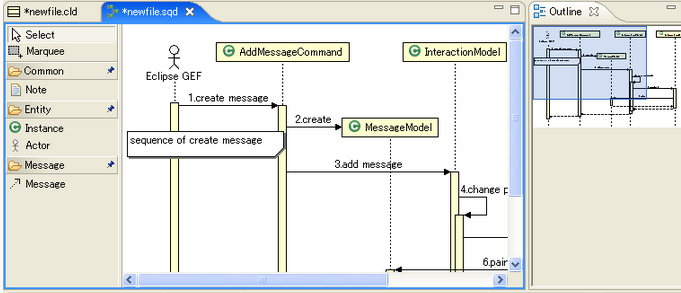
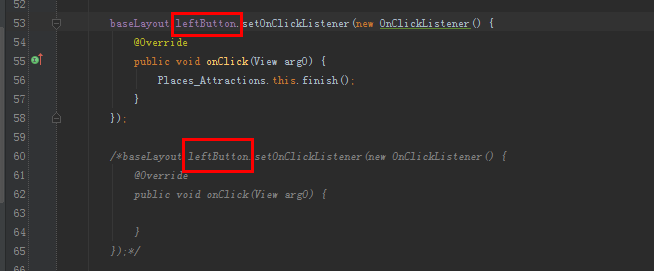


图1a-10 时序图



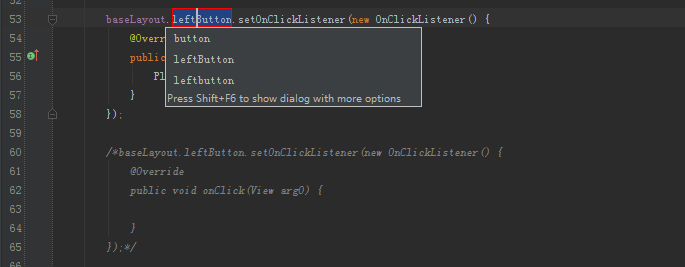
#### 1.4. 重构

高效的工程师能够熟练的使用IDE提供的重构功能。所有的现代IDE都提供许多印象深刻的重构功能。但是还是那句，IDEA的重构功能也很聪明智能。它们能读懂你需要什么，然后针对不同的情况提供给你最适合的解决方案。  
 例如下图中我们可以看到一段设置按钮监听器的代码，以及其被注释的另一份代码。此时我们想修改leftButton为mBackButton,此时就可以使用IDE的其中一个重构快捷, shift + F6。  
  
 图1a-11 重构代码举例

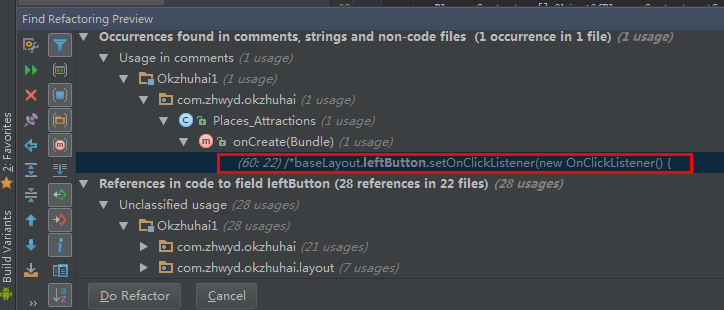


首先IDEA会给出修改的建议提醒。

图1a-12 重构代码举例

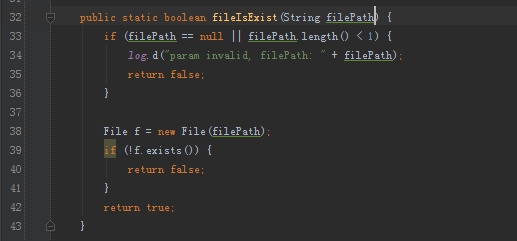


当输入完毕并且敲入回车后，IDEA会对整个项目进行扫描，并且提醒询问所有关联的地方是否进行修改，甚至是注释了的地方,这一来就大大得加快了编码的效率了。  
  
 图1a-13 重构代码提示

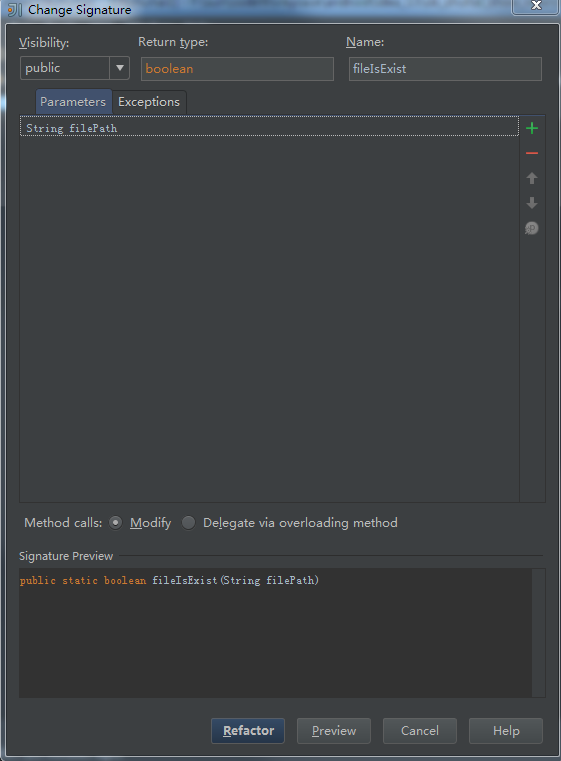


点击Do Refactor，既可以完成该次重构。

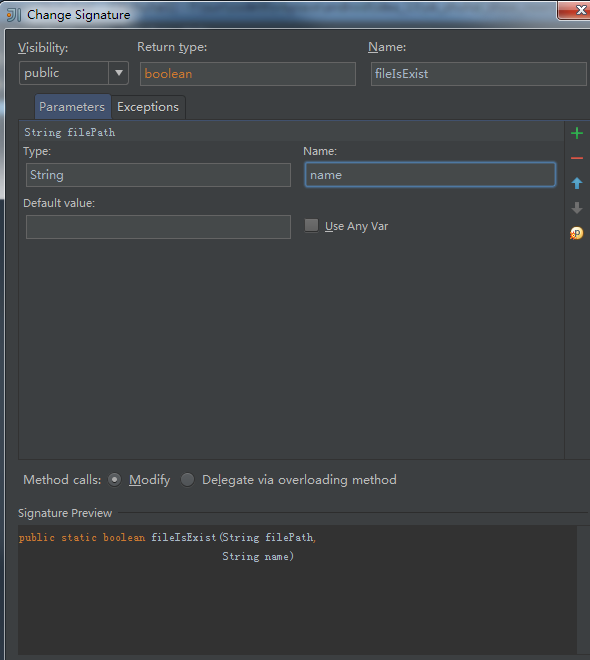
若是一些自定义的方法，需要添加或者删除某些参数时，可以使用ctrl + F6进行重构。如下图的fileIsExist(String filePath)方法，如果此时我需要往该方法添加String name的参数，并且不用每个调用它的地方都重复修改的话，使用该"重构"的方法是非常快的。  
  
 图1a-14 重构方法参数



鼠标移值该方法上，并敲下ctrl + F6，此时就会弹出下面这样一个窗口。  
  
图1a-15 重构方法参数

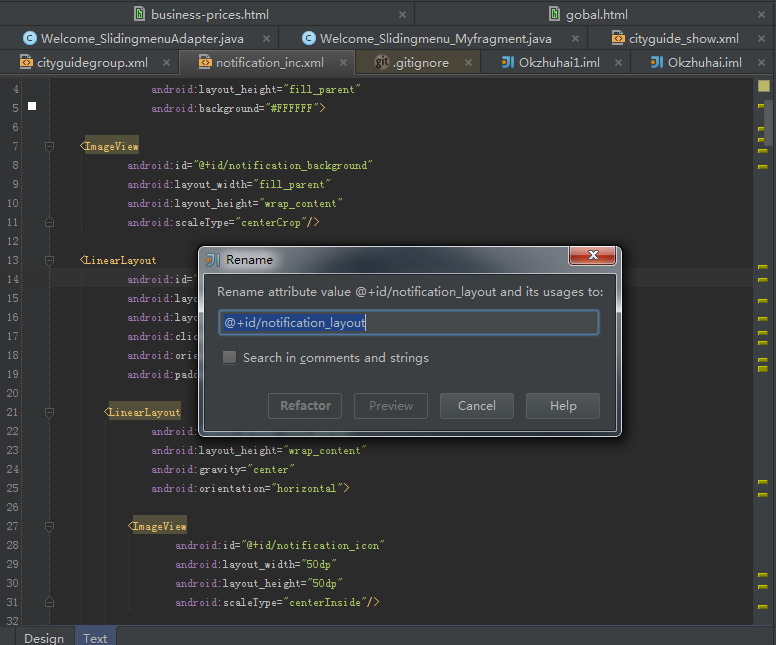


此时在窗口里添加相关的参数既可完成重构。  
  
 图1a-16 重构方法参数



就算是重构布局文件中id也是可以的，在这方面IDEA就显得比Eclipse智能得多。

图1a-17 重构方法参数



#### 1.5. 自定义模板

用过eclipse的人都会对其中的 System.out.println()的快捷输入sysout印象很深刻，在IDEA里其实也有类似的方式：输入sout，按**Tab**键即会完整输出System.out.println()。这些快捷方式使我们用最少的字符就能完成较长的代码语句，非常的实用，相信读者们用过一次就会对它爱不释手。

更深入一层，我们能否自定义这些快捷方式呢？答案是肯定的。在Eclipse和IDEA都同样具有类似的功能，称之为**自定义模板。**两个编辑器在这方面得功能都不相伯仲，这里就简单通过大家都比较陌生的IDEA来讲解这个功能。

首先可以通过Ctrl + J快捷键组合来获得模板列表。一般快捷方式均为模板关键字首字母的组合。

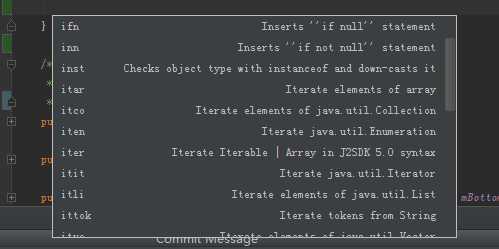


图1a-18模板列表

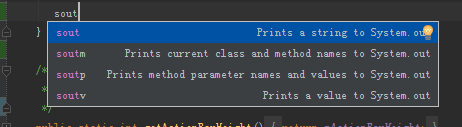


图1a-19 sout模板列表

接着我们来自定义我们常用的模板吧！在IDEA里是没有提供try-catch模板的，它只提供了选中一句话后围绕着该句代码生成try-catch的方法，也即当我们想要输入空的try-catch代码块时，是需要手动输入的。

那么我们下面需要的操作是

Settings（Ctrl+Alt+S） -> Live Templates -> Add -> Edit Live Template

下面会看到如下的界面：

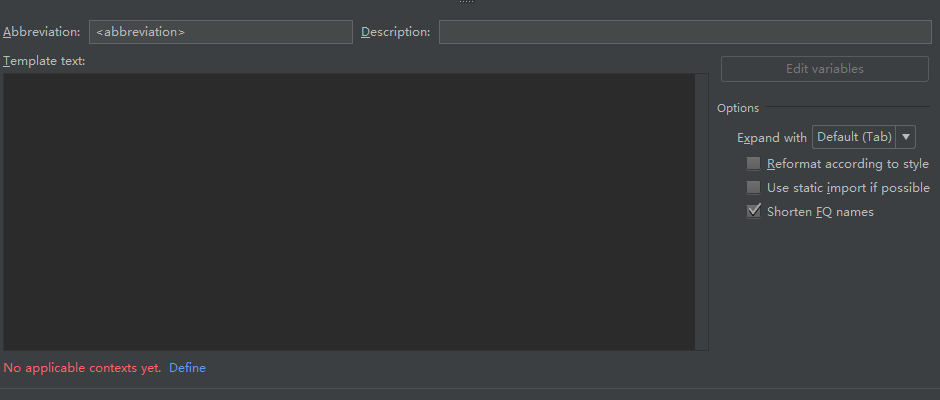
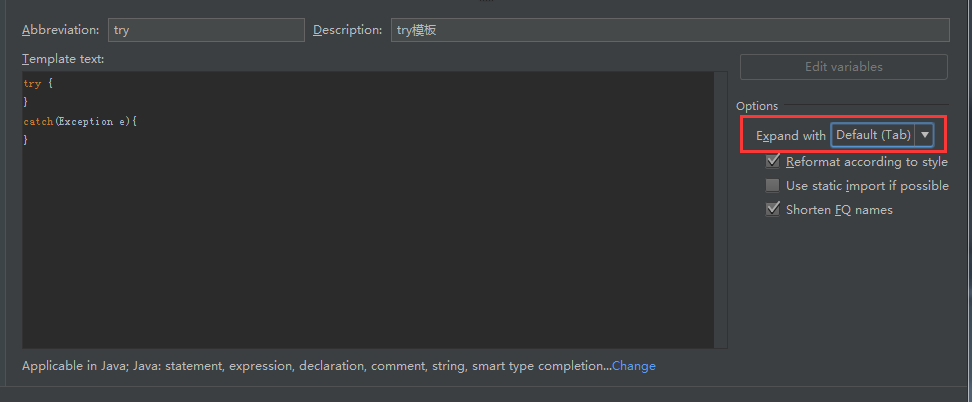


图1a-20 添加模板

我们只需要输入Abbreviation和try-catch代码，然后点击“Refine”进行定义该模板适用的场景，点击后勾选“java”即可。

图1a-21 输入相关信息



在上图中红框部分为选择触发该模板的快捷键，默认的是“Tab”，当然也可以根据自己的喜好选择空格或者回车。

点击保存后既可以在编辑代码时快速的使用该模板了。

图1a-22 输入try

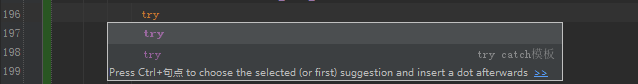


图1a-23 点击Tab展示出完整的try-catch

#### 1.6. 技术支持

由于Eclipse相比起IDEA已经相对较长的发展历史，并且用户相对更加多，所以网络中的相关技术文章会丰富得多，这让初学者可以很快捷得上手并且解决遇到的问题。

### 2.案例设计与实现

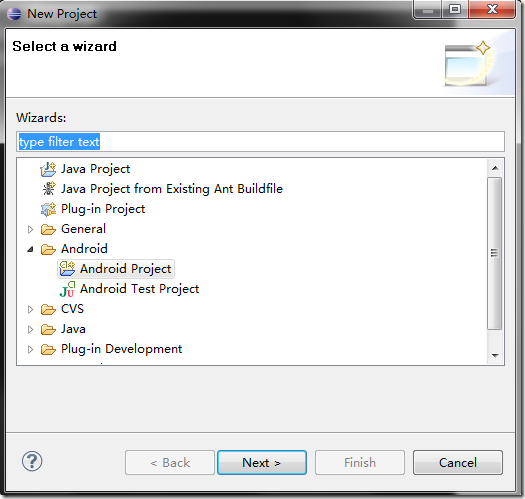
上面对两大IDE做出了粗略的对比，但是实际上更多的体验上的细节还是需要通过实践来把握的，所以接下来我们设计了3个案例让读者能够对2者之间有个更加清晰的对比。

#### 2.1. HelloWorld

##### 2.1.1. Eclipse

Eclipse里新建新的android项目相对会比较简单。

1. 【File】-> 【new】 –> 【Project】->【Android Project】  
    图1a-24 新建项目



1. 输入项目名称

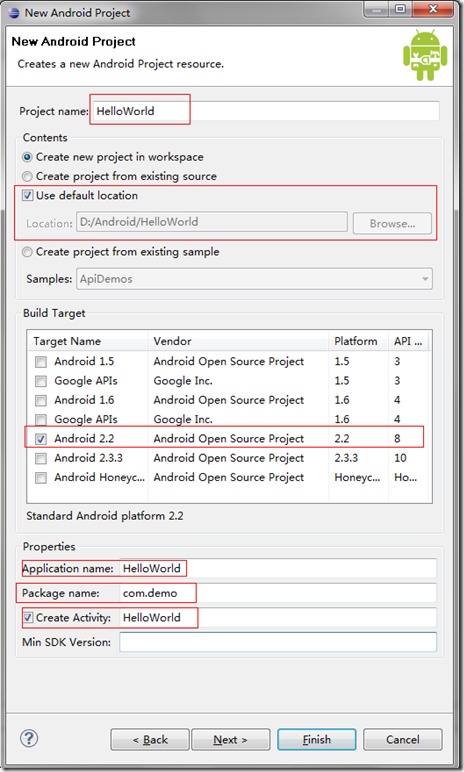


图1a-25 新建项目

按顺序介绍以上红色框内内容

a. Project name: 你建立的工程的名称, 类似于VS中的Project Name.

b. Contents: 主要用于制定你的工程代码的存放路径

c. Build Target: 说明你要开发基于Android那个版本的应用程序

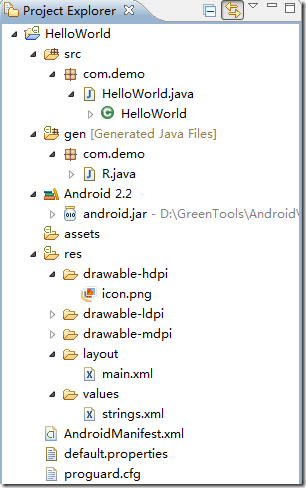
d. Application name: 这个是你开发出来的程序, 安装到设备中之后, 那个图标下面显示的名字.

e. Package name: 包名称了, 这个相等于.Net中的namespace

f. Create Activity: 选择了这个就会生成一个默认的类, 等同于VS中新建一个Console工程, 会自动创建一个Program.cs, 并且包含Main方法一样, 如果不选择, 则就是一个空的程序架子, 自己可以多建几次体会一下.

1. 创建完成之后, Eclipse中就会打开

图1a-26 项目目录  
至此, Android工程已经建立完毕。只需要点击Run运行即可让程序跑起来。



##### 2.1.2. IntelliJ IDEA

首先要说明的一点是，IDEA里面“new Project”就相当于我们eclipse的“workspace”，而“new Module”才是创建一个工程。

1. 【File】-> 【New Project】 -> 【Android】

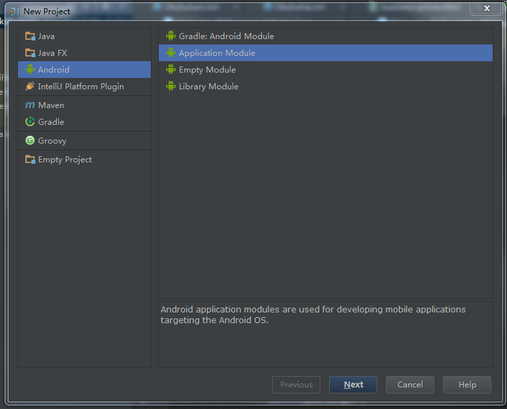


图1a-27 新建项目

1. 输入项目名称信息  
    图1a-28 新建项目

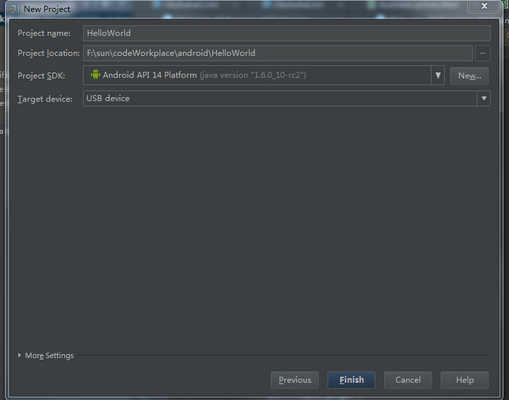
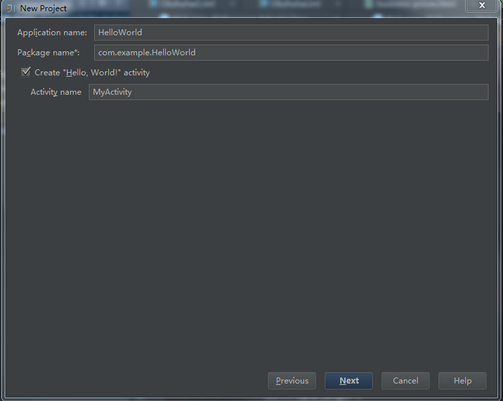


图1a-29 新建项目

1. 创建完成后， 对应可以看到如下目录。

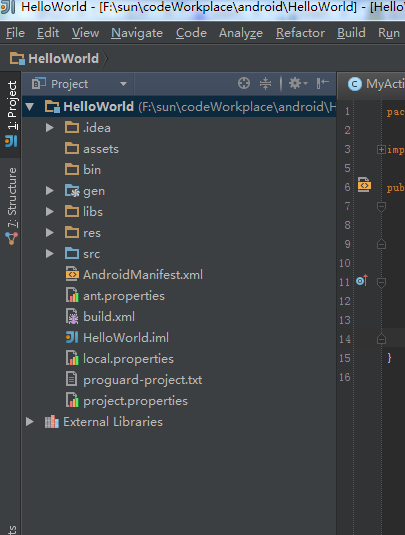


图1a-30 新建项目

#### 2.2. 依赖库导入

很多时候我们在进行android开发的时候，恰当的使用插件能让我们app应用如虎添翼，并且避免重复制造轮子。那么在Eclipse与IDEA里，2者引入依赖库的方式是大同小异的，不过IDEA提供了更为方便的一些特性。本案例中我们用android-support-v4.jar为例，分别介绍它们的导入方式。

##### 2.2.1. Eclipse

一般情况下，我们都是在项目目录下的libs文件夹里放置依赖库，如下图所示。

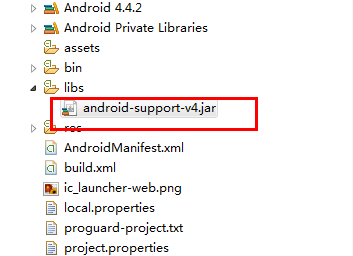
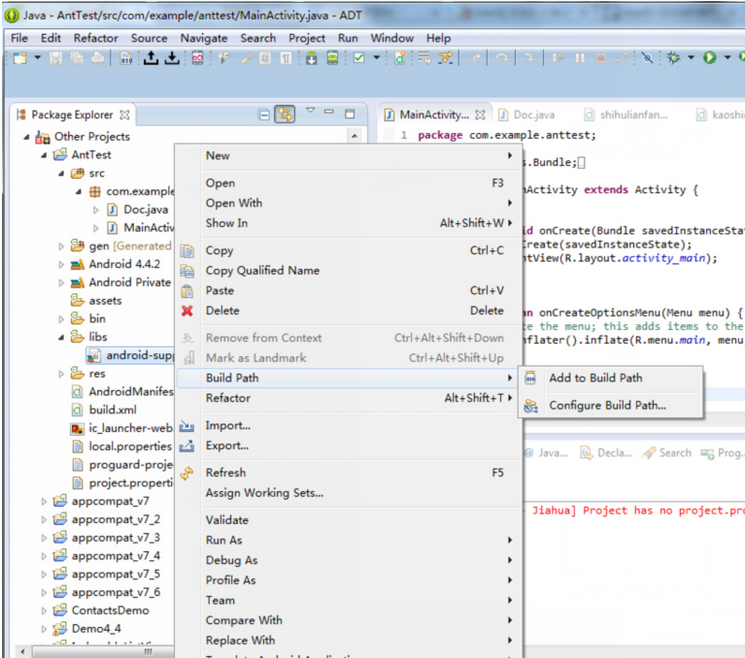


图1a-31 导入依赖库

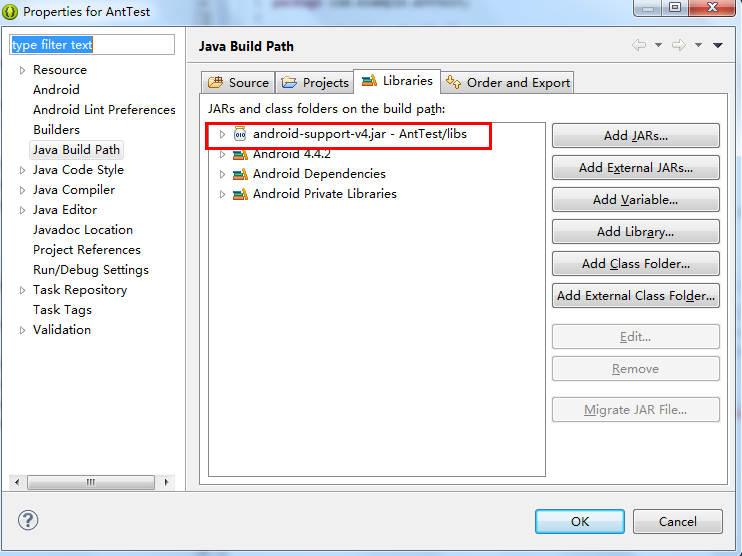
然后【右击】-> 【Build Path】-> 【Add to Build Path】就能够成功得把所选择的依赖库导入项目

图1a-32 操作演示



此时打开项目目录，可以看到对应得添加了Jar。

图1a-33 导入成功

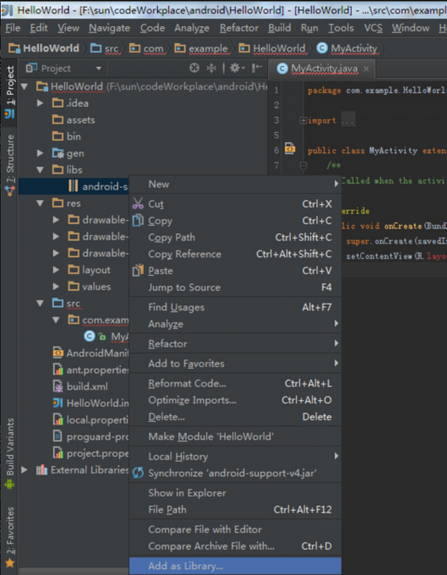


##### 2.2.2. IntelliJ IDEA

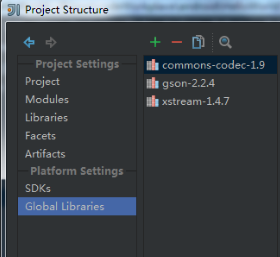
与Eclipse类似，IDEA也可以直接右击进行导入依赖库。具体操作如下：

【右击】->【Add as Library】

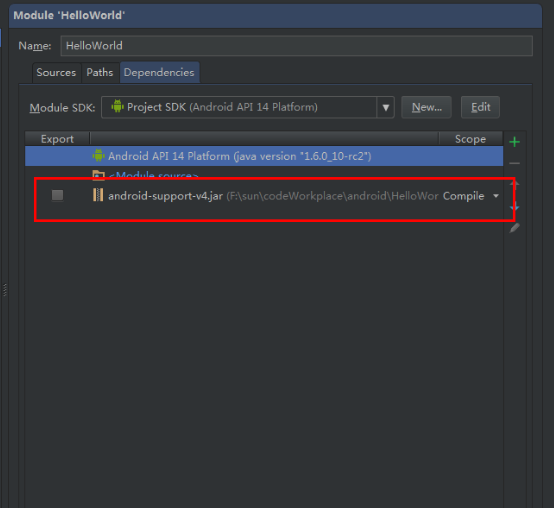
图1a-34 导入依赖库



在弹出的窗口里，可以该依赖库的使用范围，Global Library, Project Library, Module Library。从字面上就能够理解到使用范围分别为全局，工作目录，以及项目目录。这个功能是非常实用的，一些常用的依赖库可以更方便地管理起来，不用每次都去复制粘贴一遍。  
 在进入Project Structure目录后就能在对应的Global Libraries对已经被添加的依赖库进行导入，如下图。  
   
 图1a-35 Global Library



对应地，添加成功后，在Project Structure里会看到相关的依赖库。  
  
图1a-36 导入成功



#### 2.3. 版本控制的使用

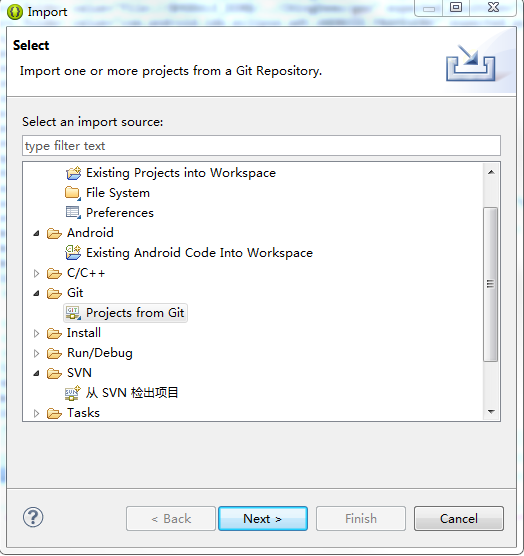
Git是一个免费的、分布式的版本控制工具，或是一个强调了速度快的源代码管理工具。每一个Git的工作目录都是一个完全独立的代码库，并拥有完整的历史记录和版本追踪能力，不依赖于网络和中心服务器。  
 Eclipse自身是不带版本控制功能的，所以它通过插件的形式来使用这方面的功能，其中egit为较为流行的Git插件。而IDEA则自带了版本控制的功能, 包括git, svn等。

##### 2.3.1. Eclipse

在本次案例中，我们通过演示使用EGit(Eclipse Git Plugin)获取Android源代码来让大家了解在Eclipse使用Git的情况。

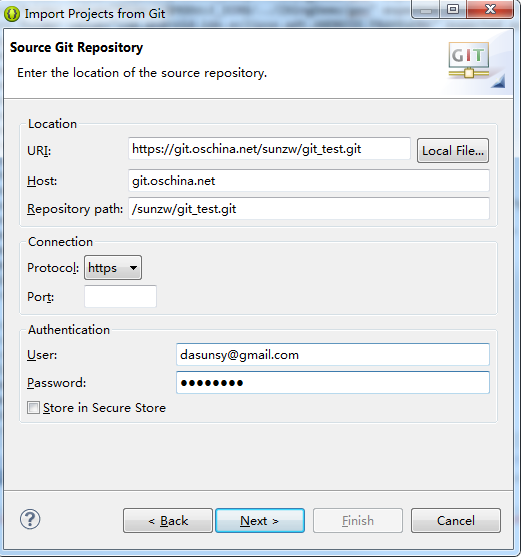
导入Git repository  
在Eclipse中选择 File -> Import... 打开Import对话框,在里面选择Projects from Git选项.

图1a-37选择Projects from Git



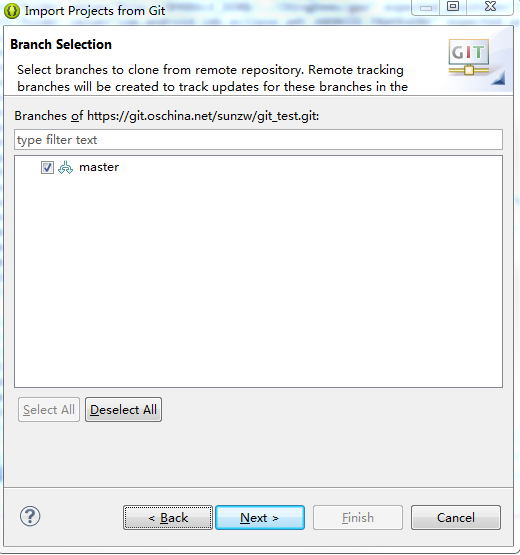
点击Next, 在Git URI中输入地址。

图1a-38 填入信息



点击Next,选择要clone的分支.

图1a-39 选择分支



点击Finish.导入完成后,项目既出现Eclipse中,你可以在你选择的目录中查看到clone的内容。（在旧的版本中，项目并不会出现在Eclipse中，需要手动导入）

##### 2.3.2. IntelliJ IDEA

由于IDEA自身就集成了Git的功能，所以其易用性，体验各方面都会比Eclipse好很多，下面简单介绍一下利用IntelliJ使用代码版本控制（Git）的操作过程（此部分作为初学者不做必须要求，但使用git是成熟程序员必备的技能。）

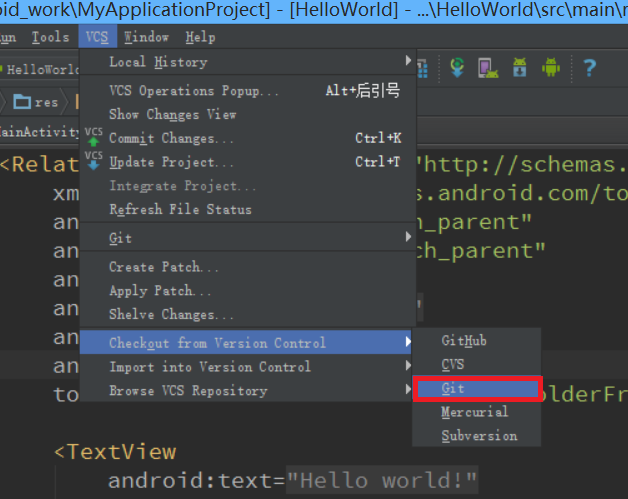
首先，判断是否成功连接上了git服务器（前提是你需要预先申请好帐号），点击【file】->【Setting】->【Version Control】->【Git】，选择右上角的Test，如图中高亮所示。如果成功连接上了Git，就会显示Successfully，如图1a-40所示。

【其他方法】新版本IntelliJ可以在【VCS】下的【Checkout from Version control】-> 【Git】找到Test选项。

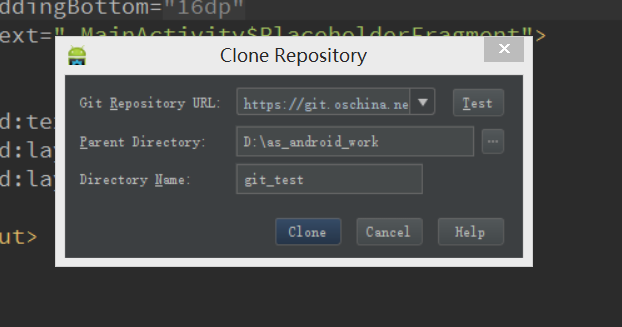


图1a-40 测试是否连上git

连接成功后，接下来需要实现Clone项目的操作，在IDEA只需选择：【VCS】-> 【Checkout from Version control】-> 【Git】  
  
图1a-41 准备Clone Git项目



然后在下图中的Vcs Repository URL填入地址即可，对应Dictory Name会自动生成。  
  
图1a-42 输入url



在修改项目之后需要上传新的项目文件至git，可以在【Setting】->【Version Control】中点击左上角的绿色加号，选择项目文件和上传的平台之后，就可以在IDEA中直接进行git操作了，如图1a-43所示：

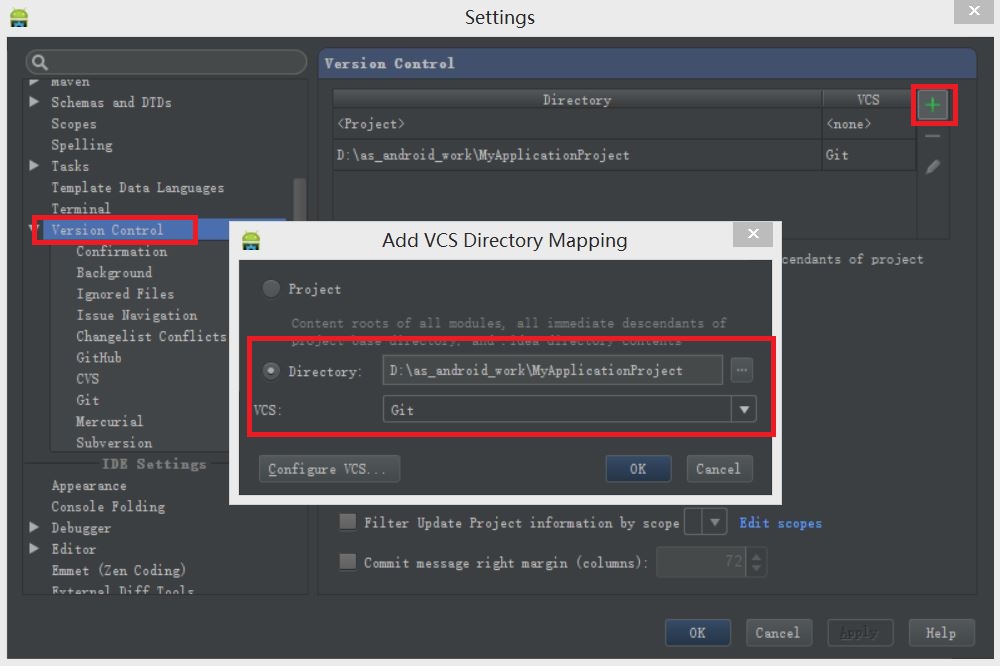


图1a-43 添加本地项目与git的关联

本地项目添加与远程git服务器的关联之后，就可以点击【vcs】->【git】中进行git操作了，如图1a-44所示的commit（提交代码），show history（显示历史版本，稍后此部分后面有细节介绍），push（推送）等操作：

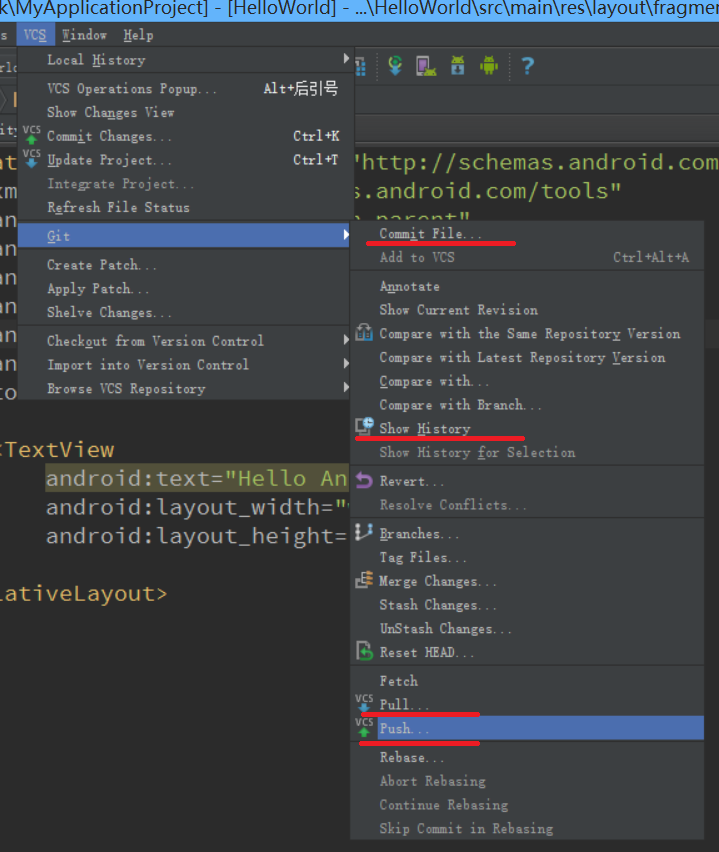


图1a-44 对项目文件进行git操作

下面演示一次push的操作，左边栏中选择项目文件，然后点击图1a-44中的Commit File，在comment中填写本次push的提示信息，确认后点击commit，如图1a-45所示：

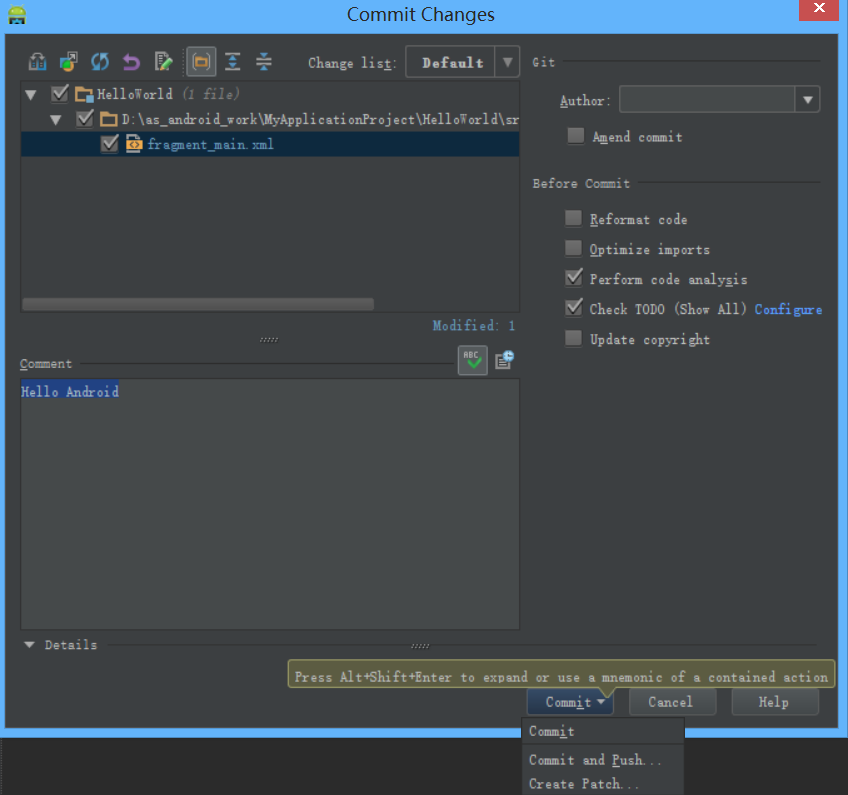


图1a-45 commit操作

【注意】点击commit会要求你填入你的帐号密码，Commit成功之后窗口下方会出现如图1a-46所示的提示信息：

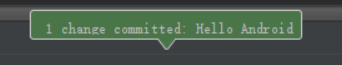


图1a-46 commit成功

然后点击push，选择Commit的版本，记得要勾上下图中的红框位置，确定后点击Push，如图1a-47所示：

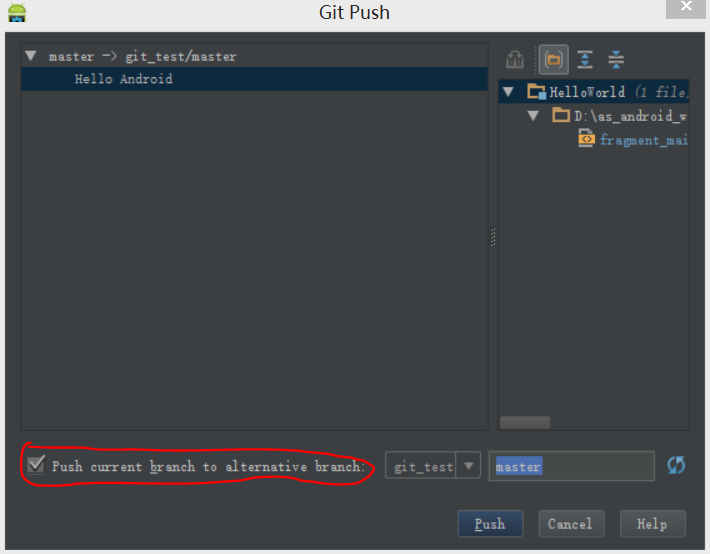


图1a-47 Push操作

Push成功之后，窗口下方出现提示信息如图1a-48所示就代表Push成功了。

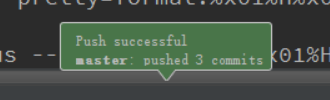


图1a-48 项目Push成功

如果需要查看push之前的几个版本代码可以点击图1a-44中的Show History，如图1a-49所示：

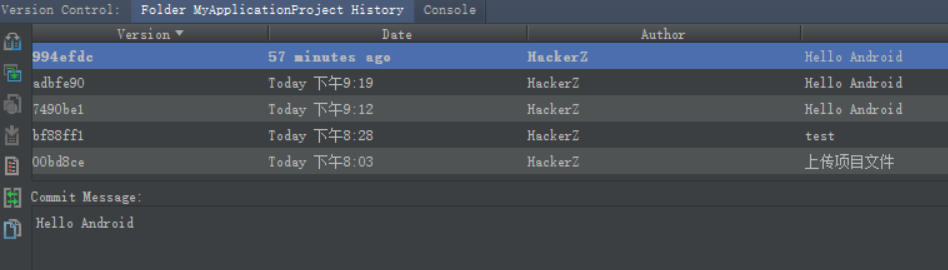


图1a-49 显示以前的版本

双击需要查看的某个版本之后，会出现修改过的文件列表，如图1a-50所示：

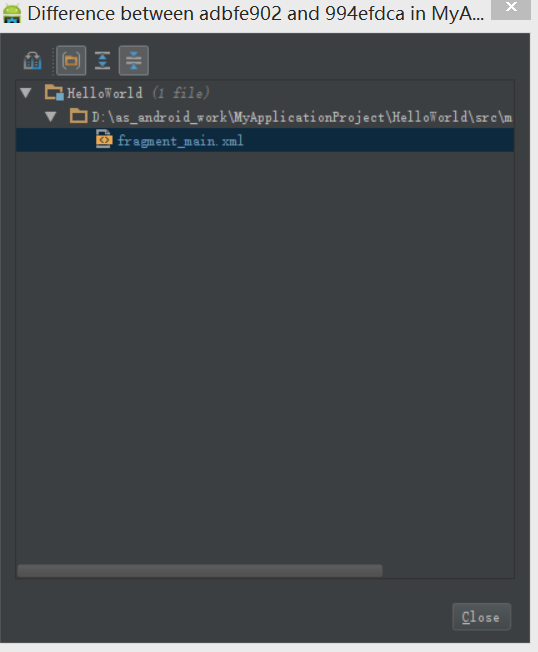


图1a-50 该版本修改过的文件列表

再次双击文件，会显示该文件的两个版本的对比，如图1a-51所示（右图比左图多加了感叹号，IntelliJ会高亮提示：

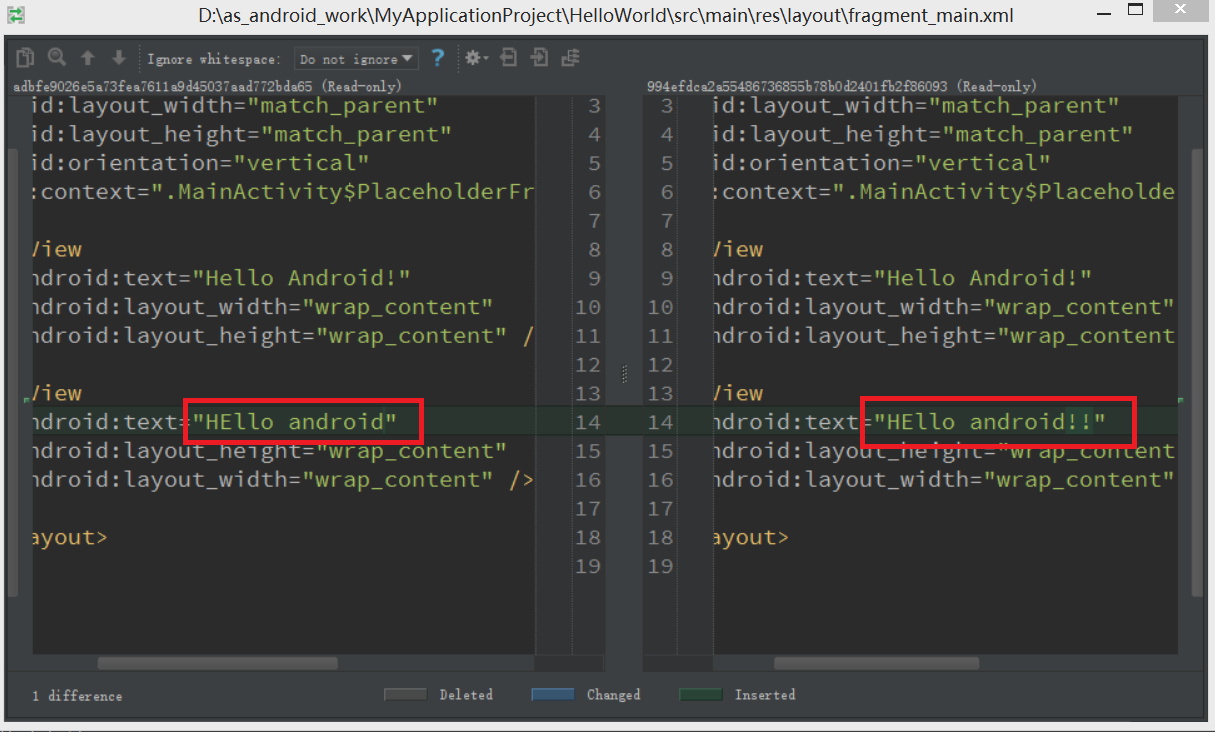
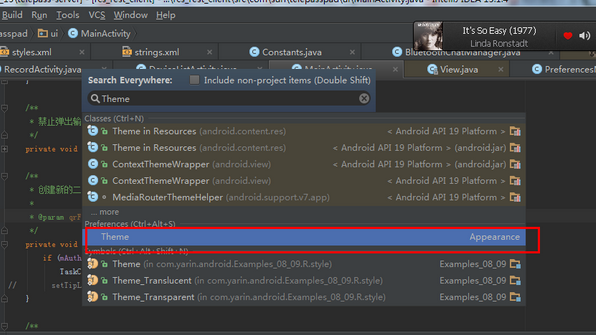


图1a-51 文件对比

### 3.常见问题

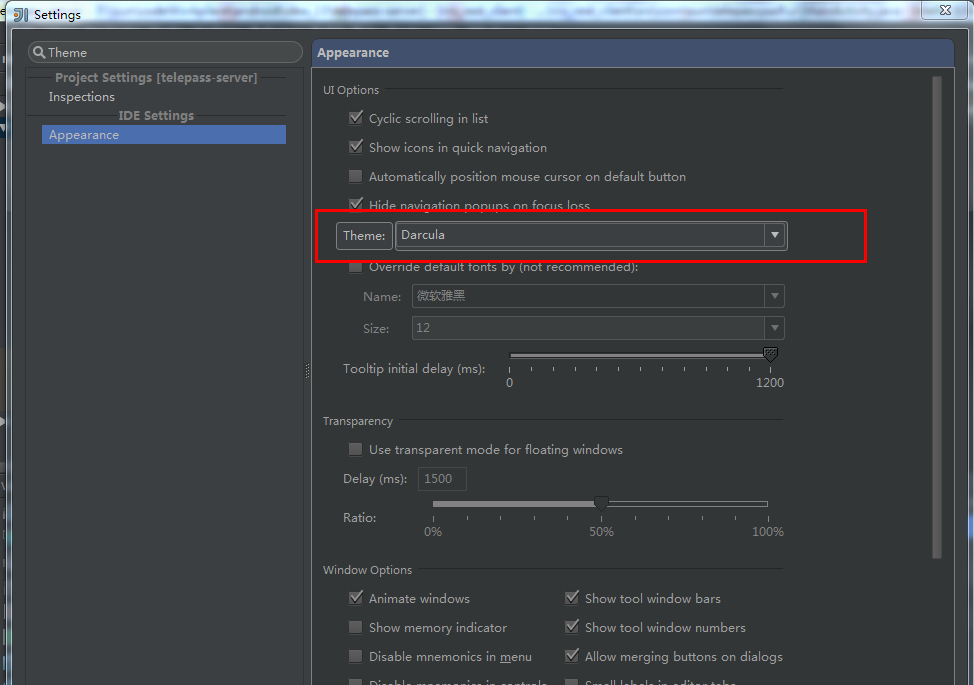
##### 3.1 切换主题

上文有提到过IDEA最受欢迎的一个主题：Darcula，那么该主题是怎么设置的呢 在这里介绍一个IDEA非常强大的快捷键：双击 shift, 它能够进行"全局"搜索, 这里的"全局"并不是指代码方面的搜索, 而是对整个IDEA IDE包括配置文件, 快捷键, 偏好设置, 代码, 资源文件的全方位搜索。 所以当我们需要切换主题的时候，只需要双击 shfit后输入"Theme"就会有相关的选择出现，如下图。  
  
 图1a-52 双击 shift



然后敲回车既可以进入切换主题的界面。

图1a-53 选择Darcula



##### 3.2 保存

一开始使用IDEA时你可能会找保存相关按钮或者快捷键，其实这是不需要的，因为IDEA是实时保存你的代码的，所以在这里大家并不用担心因为没保存而导致代码丢失的问题。

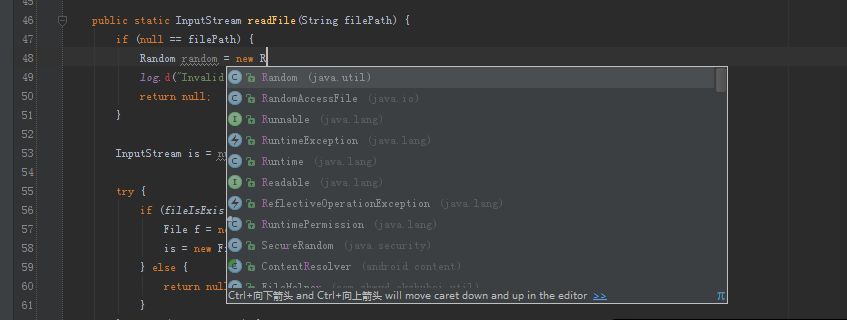
### 4. 快捷键

在IDEA中提供了让快捷键映射成其他IDE的快捷键的功能的，同上面类似，可以直接双击 shfit，然后输入keymap 来进行相关设定。下面罗列了IDEA默认常用的快捷键，只有熟悉使用这些快捷键才能达到“键盘流”的境界，把IDE发挥到极致，下面将对部分IDEA以及Eclipse常用的快捷键进行介绍对比。

#### 4.1. 代码提示

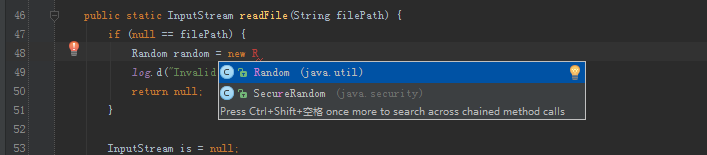
Eclipse：  
ALT + /， 提供内容辅助功能，当输入部分类、属性或方法的名字后，按此键会显示匹配的名称。  
Intellij Idea ：

1. Ctrl + Space，基本的代码补全功能，包括提示相关类名，方法名，变量名。  
    图1a-54 基本代码提示



1. Ctrl + Shift + Space, 智能的代码补全功能, 它会将建议列表中的不适用的条目过滤掉，只显示可用的类、变量、属性或者方法，这个提升了性能而且可以避免不必要的错误，如果你试用下这个功能，你肯定会时刻想到它。

图1a-55 智能代码提示



1. Ctrl + Alt+ Space, 类名补全，让你更加快速的补全类名，并且添加import语句。

#### 4.2. 注释

在注释上两者使用的快捷键并没有任何差异。

Eclipse，Intellij Idea：

1. Ctrl + / : 快速添加注释，能为当前行或选定行快速添加注释或取消注释。
2. Ctrl+Shift+/ : 添加注释块。

#### 4.3. 最近的文件

Eclipse，Intellij Idea：  
Ctrl + E：在窗口中列出最近打开的所有文件，切换文件更方便。

Intellij Idea：  
Ctrl + Shift + E： 在窗口中列出最近编辑过的文件。

#### 4.4. 编辑

Eclipse，Intellij Idea：

1. Ctrl + C，复制（IDEA可以复制当前整行）
2. Ctrl + X, 剪切 (IDEA可以剪切当前整)
3. Ctrl + V, 粘贴 (IDEA中可以使用 Ctrl + Shift + V选择选择最近复制的内容)
4. Tab : 选中部分 向右跳置tab 的距离
5. Shift + Tab : 选中部分 向左跳置tab 的距离

Eclipse：

1. Ctrl + Alt + ↑（↓）：复制正行或者整块，并且自动粘贴到新的一行。  
   例如我需要把下图红色框的语句复制并粘贴多一句，只需要把鼠标点击到对应位置，并敲下 Ctrl + Alt + ↓：

图1a-56 代码举例

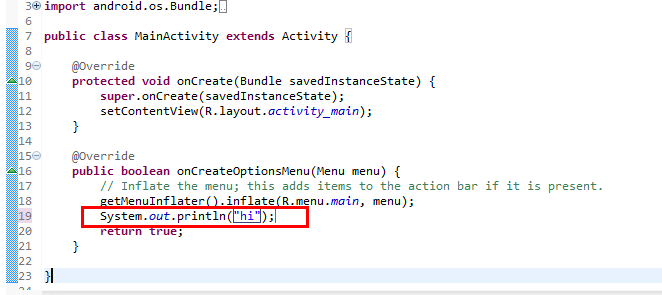
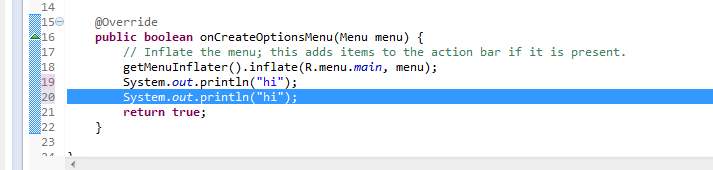


图1a-57 复制粘贴成功



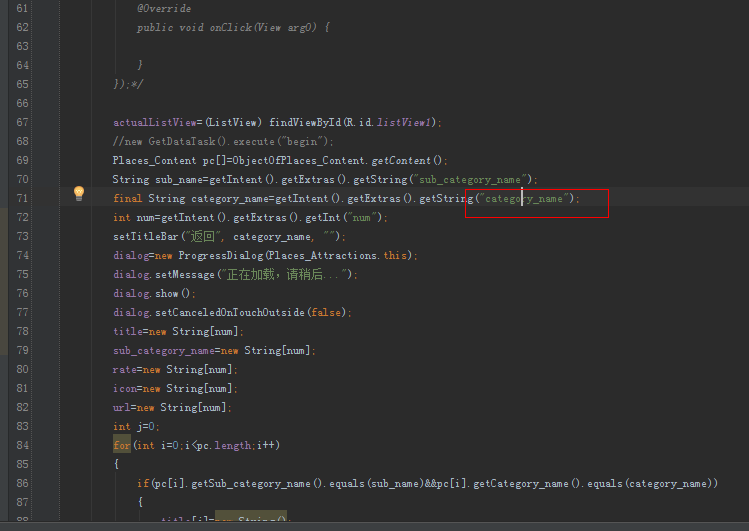
1. Ctrl + D ： 删除行

Intellij Idea:

1. Ctrl + D : 复制正行或者整块，并且自动粘贴到新的一行，效果类似上面提到的Eclipse。
2. Ctrl + Y ： 删除行
3. Ctrl + W : 可以选择单词继而语句继而行继而函数，此快捷键可以说是IDEA的杀手级快捷键，熟练使用它，效率肯定会上升一个级别。

如下图我鼠标定位在category\_name上。

图1a-58 Ctrl + W演示



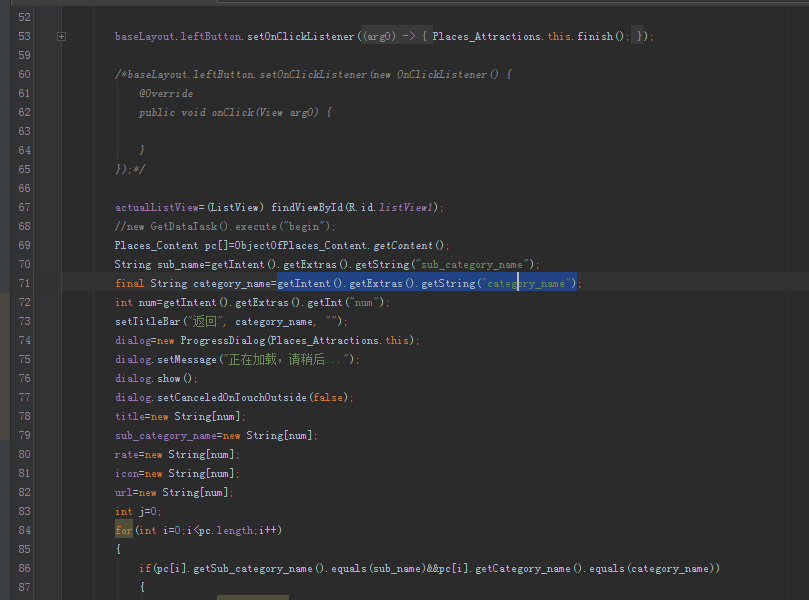
当我初次敲下 Ctrl + W时，将选中整个category\_name,如下图。

图1a-59 Ctrl + W演示



再次敲下，将选中"category\_name"", 再敲一次, 将选中getIntent .... "category\_name"的句子。

图1a-60 Ctrl + W演示



再敲入下就是选中整行，如此类推，跌进，直到整个方法，整个类都选中了。实践下是否爱上这个快捷键呢？

#### 4.5. 查找与定位

1. 全局搜索 （Eclipse：Ctrl + H， Intellij Idea ：Ctrl + Shift + F ）。
2. 查找类、方法和属性的引用（Eclipse：Ctrl + Shift + G , Intellij Idea : Ctrl + Alt + F7(Alt + F7) ）。
3. 跳转到当前类的指定行数(Eclipse：Ctrl + G, Intellij Idea : Ctrl + L)。
4. 快速定位光标位置的某个类、方法和属性(Eclipse：F3, Intellij Idea : Ctrl + B（Ctrl + 单击）)。
5. 返回到上次编辑的地方(Eclipse：Ctr + Q, Intellij Idea : Ctrl + Shift + Backspace)。

#### 4.6. 重构

1. 重命名(Eclipse： Alt + Shift + R ， Intellij Idea ：Shift + F6 )

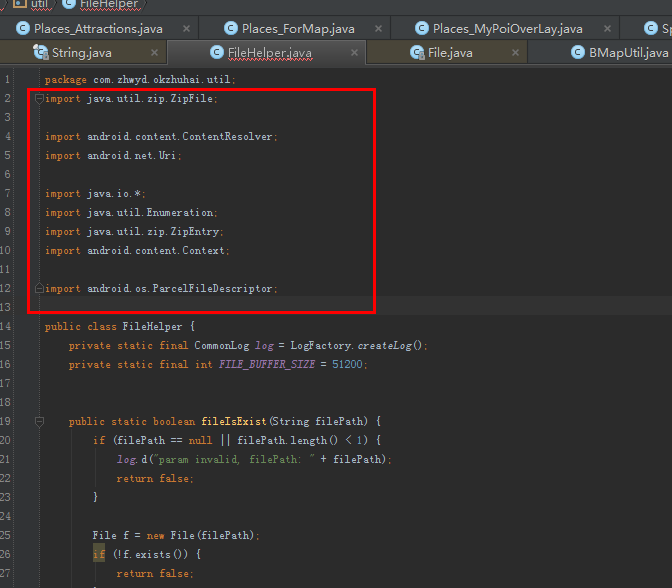
#### 4.7. 修正代码

1. 格式化代码（Eclipse：Ctrl + Shift + F， Intellij Idea ：Ctrl + Alt + L ）。

Eclipse：

1. Ctrl + Shift + O ：快速生成import，导入需要的类。

Intellij Idea ：  
Ctrl + Alt + O ： 优化import, 如下图中的红框部分，此时的“import”是杂乱无序的。  
  
图1a-61 乱序的import



这个时候，在IDEA里只需要输入上述的快捷键, 即可完美解决。

图1a-62 优化后的import

