

IntelliJ 4.5 中文安装手册



Copyright

© 2000-2004 JetBrains, Inc. All Rights Reserved.

JetBrains, Inc., JetBrains, IntelliJ, IDEA, and IntelliJ Labs are either registered trademarks or trademarks of JetBrains s.r.o. in the Czech Republic and in other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners. Information in this document is subject to change without notice. JetBrains, Inc. makes no warranties, expressed nor implied, in this document. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), or for any purpose, without the express written permission of JetBrains, Inc.

关于本手册

手册的目的是在不花费你较多时间去寻找大量IntelliJ IDEA的技术文档，帮助你快速得到和使用IDEA。IDEA整合许多功能，本手册旨在指导你如何快速使用这些功能，更多的功能在你熟悉IDEA后，发掘它更多的特性。

北京捷特慈朋软件技术有限公司（jetmaven）授权制作。

TOC

1. 安装	3
1.1. Java 2 SDK (1.4.1 Windows i586)	3
1.2. IDEA	5
1.2.1. 下载 / 安装	5
1.2.2. 获取试用license	9
1.2.3. 启动IDEA (输入 license)	10
1.2.4. IDEA中文设定.....	11
2. 基本操作	13
2.1. 创建项目	13
2.1.1. 指定项目名称 / 路径	13
2.1.2. 设置项目JDK	14
2.1.3. 确定项目模块	16
2.2. 创建包	18
2.3. 创建类	19
2.4. 使用编辑器	20
2.4.1. 打开 / 关闭文件	21
2.4.2. 输入文本	22
2.4.3. 查找 / 导航	23
2.4.4. 颜色和字体	24
2.4.5. 代码风格	24
2.4.6. 错误提示	25
2.4.7. Todo	26
2.5. 代码提示	27
2.6. 重构	28
2.7. 使用版本控制	29
2.8. 查看 JavaDoc	30
2.9. 编译	31
2.10. 调试	32
2.10.1. 创建程序运行配置	32
2.10.2. 运行程序	33
2.10.3. 设置断点	32
2.10.4. 调试步进	35

1. 安装

本章主要描述以下两种软件在 Windows 平台下的安装方法：

- 1.1. Java 2 SDK (1.4.2 Windows i586)
- 1.2. IDEA 4.5

1.1. Java 2 SDK (1.4.2 Windows i586)

本节主要描述如何下载和安装Java 2 Software Development Kit (J2SDK)，这也是用IntelliJ IDEA开发时必需的软件。遵照以下步骤，你将轻而易举完成软件的安装。

1. 从 <http://java.sun.com/j2se/downloads.html> 下载J2SE 1.4.2，下载后的文件名是j2sdk-1_4_2_05-windows-i586-p.exe。注意：本手册其它章节均假设你已经安装好了此版本的jdk。

2. 双击j2sdk-1_4_2_05-windows-i586-p.exe，文件将执行安装，将出现“Welcome...”对话框。



图1.1 J2SDK安装的欢迎界面

3. 单击“next”，将出现“License agreement”对话框。

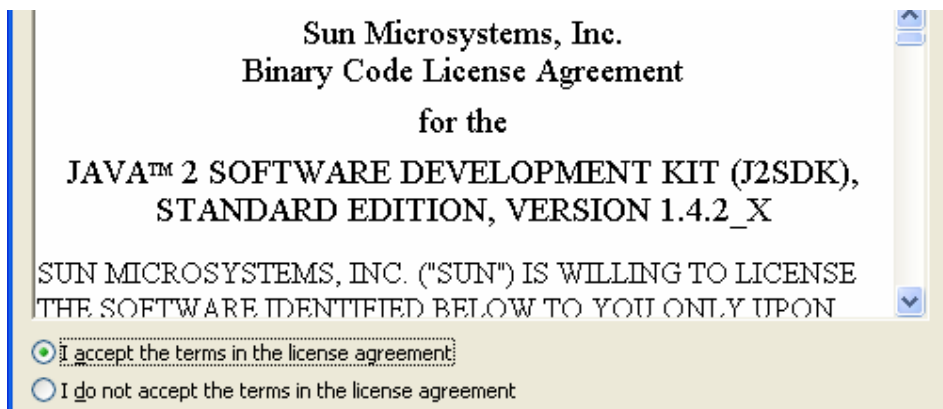


图 1.2 J2SDK 安装时的许可协议

4. 单击“**Yes**”，将出现“**Choose destination location**”对话框。请选择jdk的安装路径，默认为 C:\j2sdk1.4.2_05。如果更改为其它路径，在路径中最好不要包含空格或其它特殊字符。在组件选择框中选择你需要安装的组件，默认全部安装。

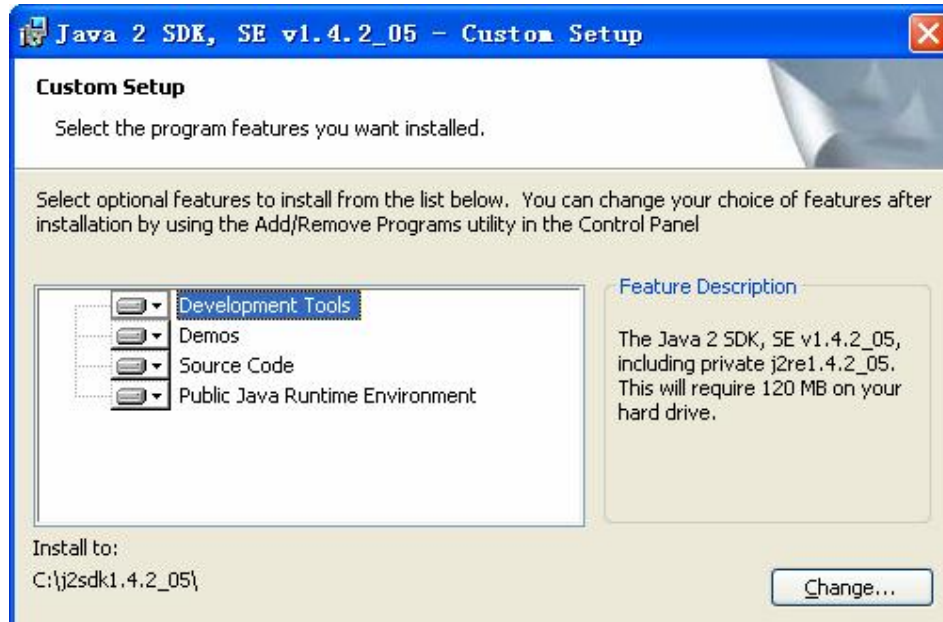


图1.3 J2SDK的安装路径

5. 单击“**Next**”，在“**Select Browsers**”对话框中选择需要应用的浏览器。在这里Java Plug-in机制将和选定浏览器进行绑定。



图1.4 J2SDK安装中选择浏览器

6. 单击“**next**”，安装开始。安装完成后，将出现“**InstallShield Wizard Complete**”对话框。



图1.5 J2SDK安装完成界面

7. 单击“Finish”，完成J2SDK1.4.2的安装。

1.2. IDEA 4.5

现在已经成功安装了J2SDK，下面开始安装IDEA。本章将包含以下内容：

- 1.2.1. 下载和安装
- 1.2.2. 获取试用序列号
- 1.2.3. 启动IDEA (输入序列号)

1.2.1. 下载和安装

1. 从<http://www.jetbrains.com/idea/download.jsp> 下载安装程序，文件名为idea2187.exe。
2. 双击“idea2187.exe”文件，将出现“Introduction”对话框。本手册中假定的IDEA的版本号为2187，4.5.x版的IDEA安装是相同的。

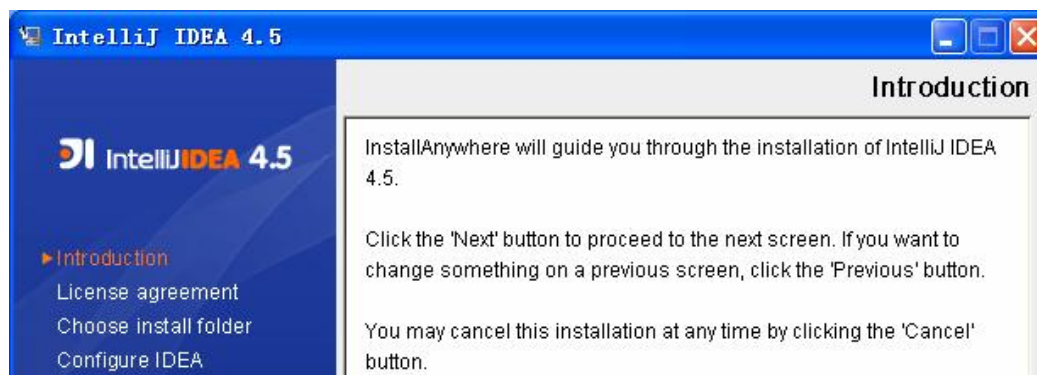


图1.6 IDEA的安装介绍

3. 单击“Next”，将出现“License agreement”对话框，选择“I accept the terms”进行下一步操作。

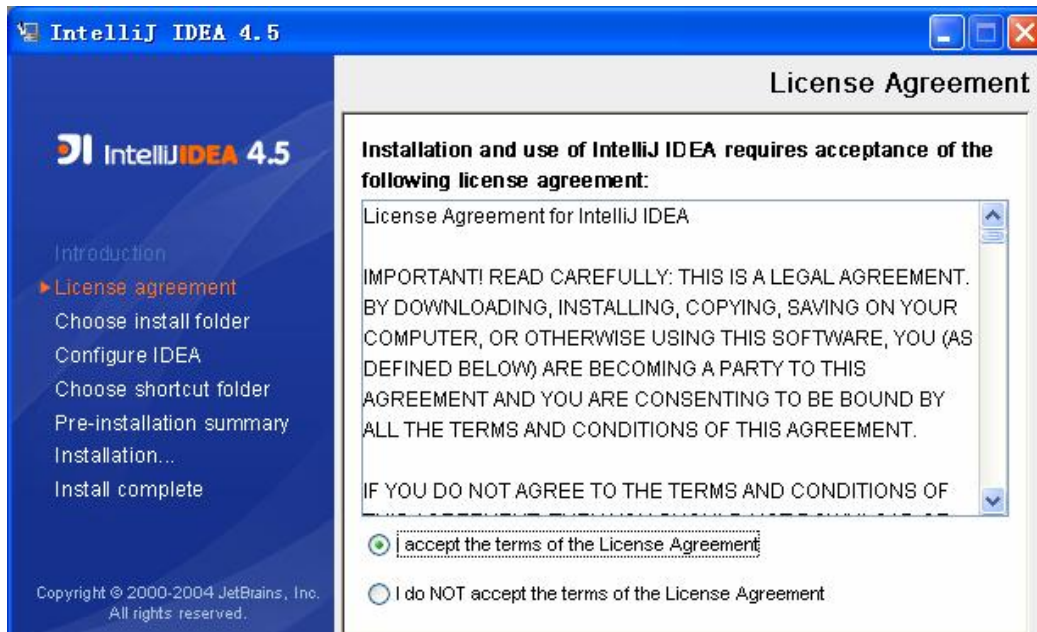


图1.7 IDEA安装时的许可协议

4. 单击“Next”，将出现“Choose install folder”对话框，选择你的安装路径。
注意：我们推荐你IDEA的安装目录名采用intelliJ+版本号方式，如目录名为intelliJ-2187，这样可方便你以后的版本升级，安装的路径中最好不要有空格和特殊字符。



图1.8 IDEA的安装路径

5. 单击“Next”，将出现保存IDEA配置信息对话框，选择“In the IDEA installation directory”。



图1.9 IDEA的配置信息保存路径

6. 单击“Next”，将出现数据缓存的对话框。选择“In the IDEA installation directory”。



图1.10 IDEA数据缓存的保存路径

7. 单击“Next”，将出现“Import Settings”对话框，此时为新安装程序，选择“I have no previous”。



图1.11 导入以前设置对话框

8. 单击“Next”，将出现“Choose shortcut folder”，主要是建立程序的快捷方式的文件夹。选择创建程序组或添加到某一程序组中。



图1.12 IDEA程序快捷方式的文件夹

8. 单击“Next”，将出现“File associations”对话框。选择是否将ipr文件与IDEA关联。如果关联，则ipr文件将默认被IDEA打开。

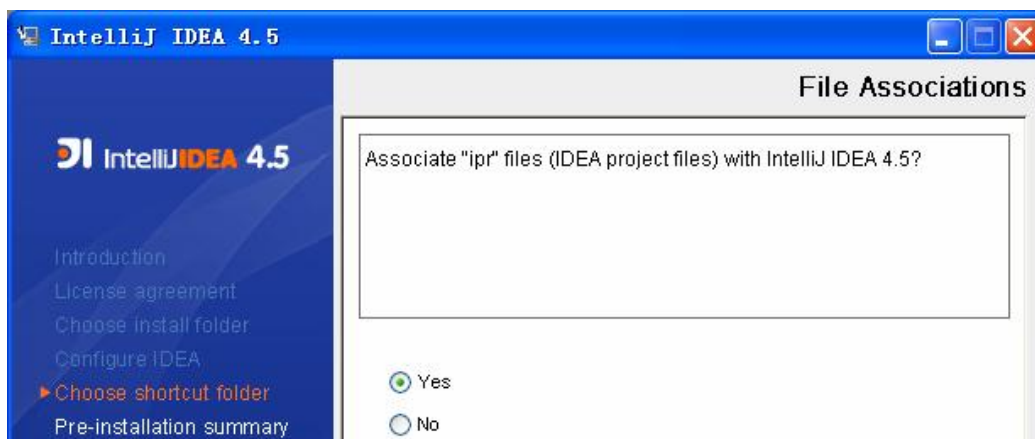


图1.13 IDEA文件关联选择

9. 单击“Next”，将出现“Pre-installation summary”对话框。请检查你选择的配置是否正确。如果不正确，请选择“Back”返回到相关的配置选择窗口进行选择，确保所有选择的配置正确。

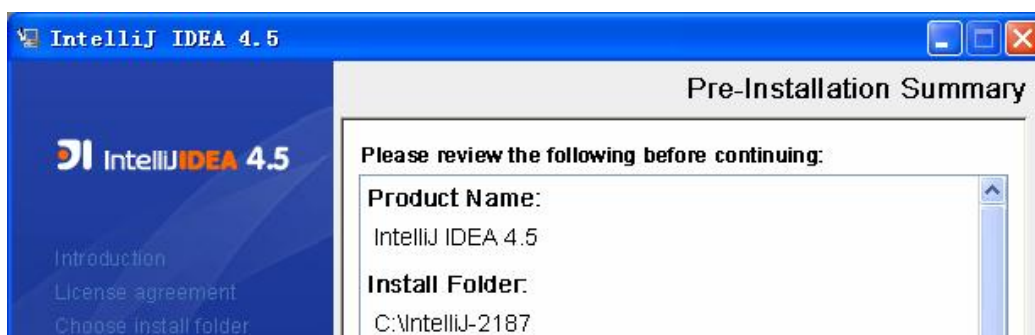


图1.14 IDEA安装前的配置概述

10. 单击“Install”，安装开始。安装完毕后将出现“readme.txt”对话框，选择是否查看readme.txt文件。

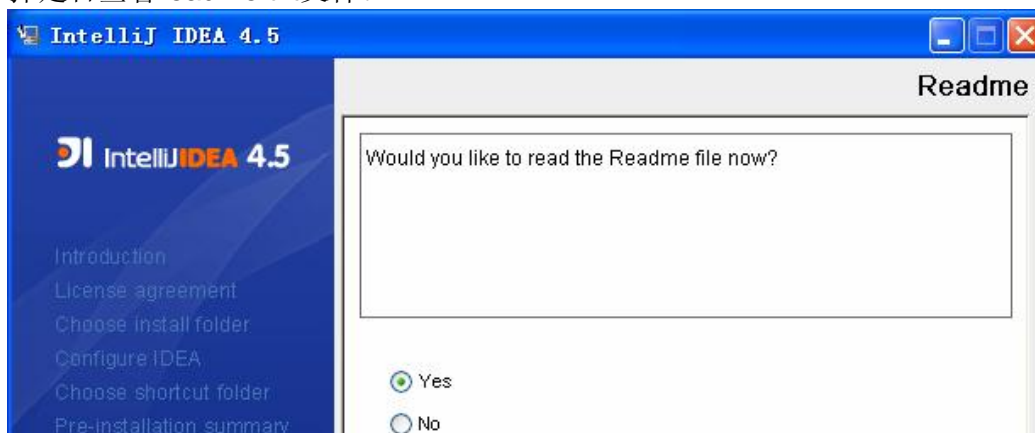


图1.15 IDEA安装后的README

11. 单击“Next”，将出现readme.txt文件的内容，主要是关于程序的一些概述。



图1.16 IDEA的readme文件内容

12. 单击“Next”，将出现“Install complete”，表示程序安装结束。

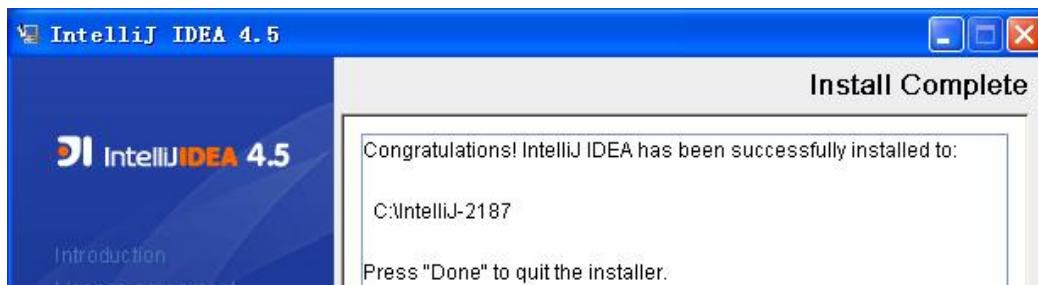
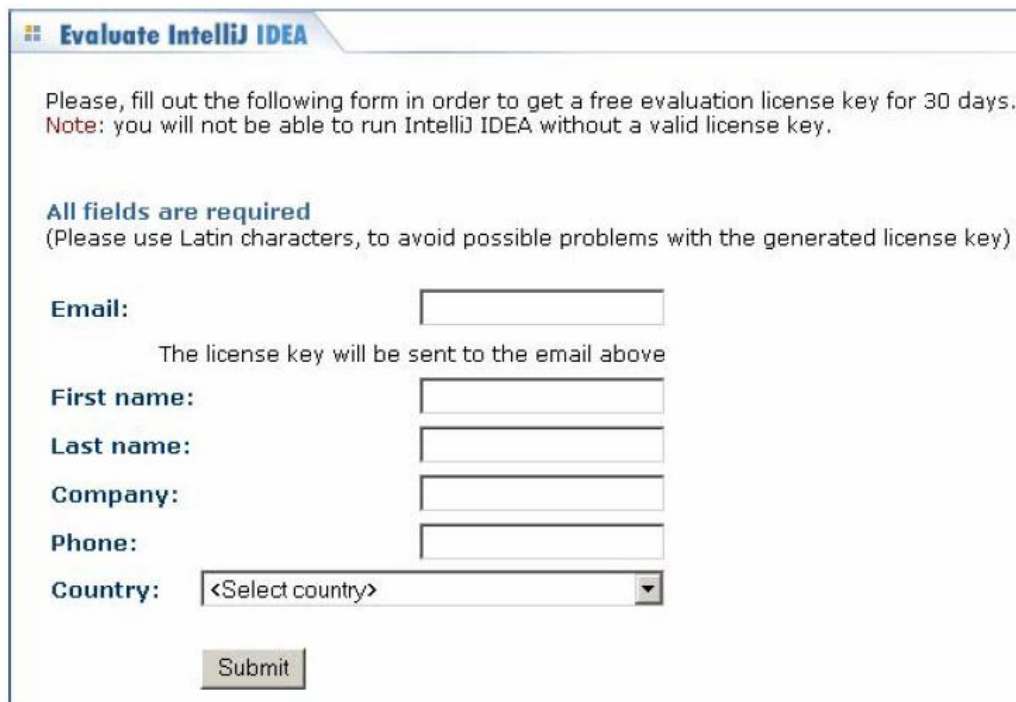


图1.17 IDEA安装完成界面

13. 单击“Done”，关闭安装的向导窗口。恭喜你，IDEA已经安装成功了！

1.2.2. 获取试用license

如果你下载的是IDEA试用版程序，则你必须取得有效的试用版序列号，这样程序才能运行起来。为了获取序列号，请在<http://www.intellij.com/idea/evaluate.jsp> 上填写正确的信息然后提交。



Evaluate IntelliJ IDEA

Please, fill out the following form in order to get a free evaluation license key for 30 days.
Note: you will not be able to run IntelliJ IDEA without a valid license key.

All fields are required
(Please use Latin characters, to avoid possible problems with the generated license key)

Email:

The license key will be sent to the email above

First name:

Last name:

Company:

Phone:

Country:

图1.18 协议表单

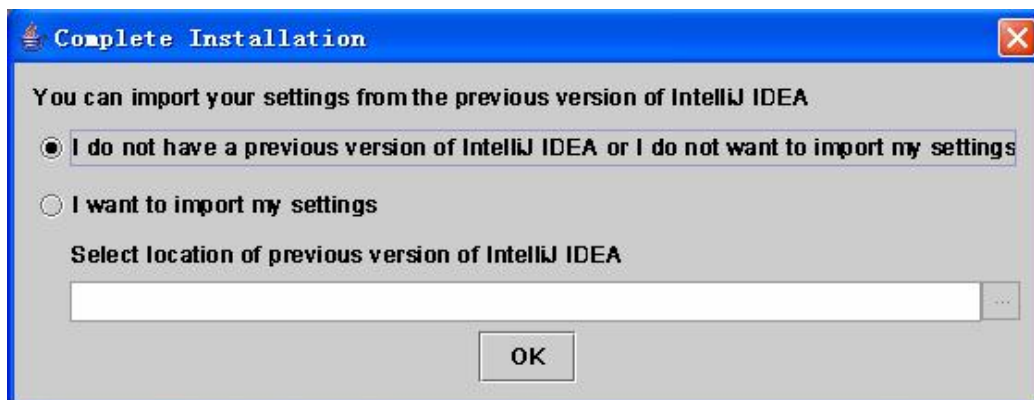
一封回复邮件将发送到表单中填写的email信箱中。回复的邮件包含以下内容：

- 用户名.
- 序列号 (区分大小写)

1.2.3. 启动IDEA (输入序列号)

如果你成功地到达这里，是时候启动IDEA啦。

1. 从Windows “开始” 选择 “所有程序|IntelliJ IDEA 4.5 | IntelliJ IDEA”,将出现询问你是否导入以前版本的IntelliJ的设置，如果你有以前版本的IntelliJ，我们建议你导入以前设置，这将节约好多时间进行重新配置。



Complete Installation

You can import your settings from the previous version of IntelliJ IDEA

☒ I do not have a previous version of IntelliJ IDEA or I do not want to import my settings

☐ I want to import my settings

Select location of previous version of IntelliJ IDEA

图1.19 导入设置窗口

2. 点击“OK”，将出现“Enter license Data”对话框，输入用户名和序列号。如果你输入的信息不正确，程序会要求你重新输入。



图1.20 IDEA的输入授权信息窗口

2. 单击“OK”，如果授权信息正确的话，则出现创建项目向导，IDEA启动成功啦。



图1.21 创建项目向导

1.2.4. 中文设定

IDEA的默认设定对中文支持比较弱，你可能会发现中文全是乱码，所以在使用IDEA之前，我们有必要进行相关的设定，让中文Show出来。J

1. 取消上一项目创建，点击工具栏的“Setting”按钮或快捷键(Ctrl+Alt+S)。

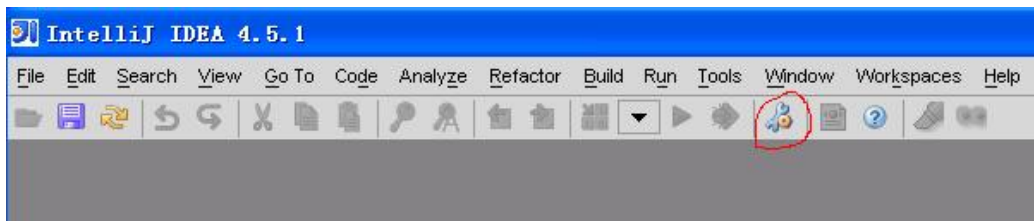


图1.22 打开设置面板

2. 在设置面板中，点击“Appearance”，进行IDEA界面相关的设定，在“Appearance”对话框中，将“Font”设置为“dialog.plain”，然后保存。



图1.23 设置IDEA界面的显示字体

3. 点击“Colors&Fonts”，我们将对编辑器（Editor）的字体进行设定，以保证编辑器中的中文能正确显示。在“Colors&Fonts”对话框中，我们点击“Save As...”按钮，我们将新建一个字体和颜色的模式，IDEA默认的配置是不允许修改的。在弹出的输入框中，输入新模式的名称，如“ChineseFont”，然后确认。

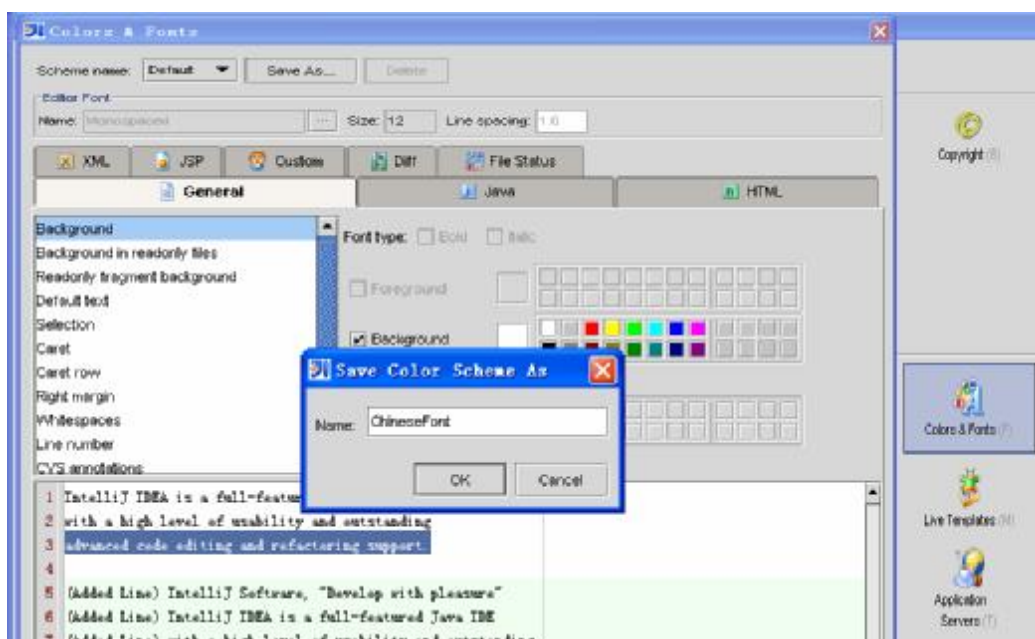
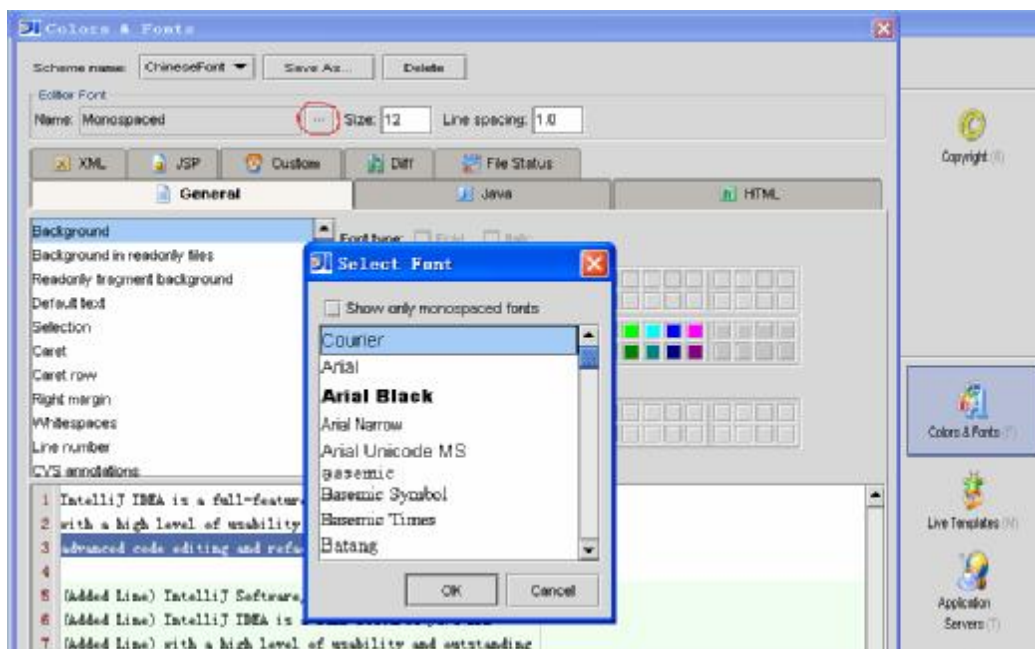


图1.24 新建字体模式

4. 在新字体和颜色模式下，修改编辑器的默认字体。点击“Editor Font”选项中的“...”按钮，在弹出的字体选择对话框中，选择“Courier”，然后确认。



1.25 设置新字体模式的字体名称

通过以上的设定，你就可以放心使用IDEA，而不用担心中文问题啦。同时你可以设定字体的大小，以方便你的阅读和编辑。在编辑模式下，你可以按住Ctrl键，同时滚动鼠标的翻页按钮，可以实现编辑器字体大小的动态调整。

2. IDEA使用基础

本章节将包含以下内容：

- 2.1. 创建一个项目
- 2.2. 创建包
- 2.3. 创建java类
- 2.4. 使用编辑器
- 2.5. 自动代码
- 2.6. 代码重构
- 2.7. 使用版本控制
- 2.8. 查看 JavaDoc
- 2.9. 编译
- 2.10. 调试

2.1. 创建一个项目

在上一章中，我们以“creating project dialog”对话框作为结束。在本节中，我

们将指导你一步一步创建你的第一个项目。为了创建一个项目，你必须：

- 2.1.1. 确定项目名称 / 位置
- 2.1.2. 设置项目 JDK
- 2.1.3. 确定模块的名称 / 路径

2.1.1. 确定项目名称 / 位置

1. 在项目名称栏输入项目名称：“MyProject”。

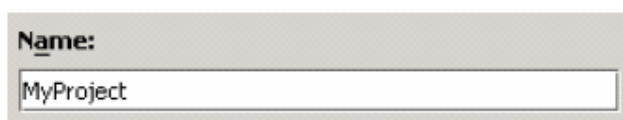


图2.1 项目名称

2. 在项目位置栏，单击省略号按钮，在“Select Path”对话框中，右击鼠标，在弹出的菜单可选择新建文件夹。

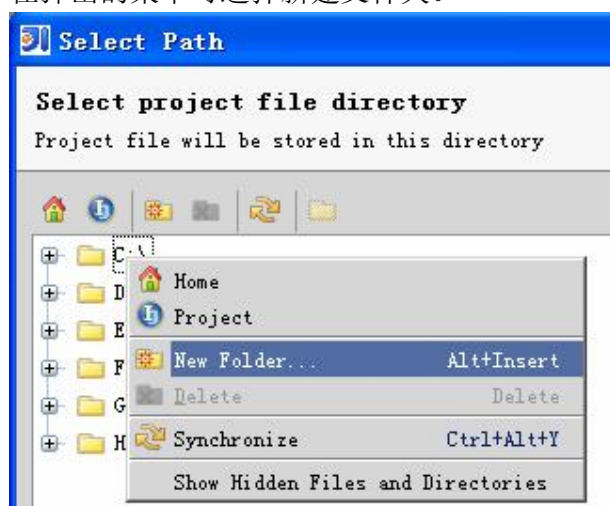


图2.2 设置项目存放的路径

3. 单击“New Folder...”，将出现“New Folder”对话框，在此对话框中输入存放的文件夹名称，如：MyprojectFolder。注意：存放的路径或文件夹名称不要包含空格或特殊字符。



图2.3 创建存放项目文件的文件夹

4. 单击“OK”，文件夹将被创建。双击“MyprojectFolder”，文件夹被选定，

将做为项目的存放路径，点击“Next”将进入JDK的设置

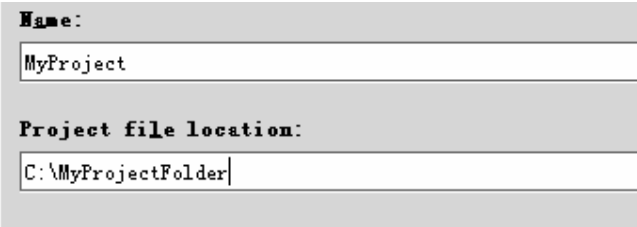


图2.4 项目的名称/位置

3.1.2. 设置项目 JDK

1. 此时将进入项目JDK设置窗口，在此窗口你可以选定该项目的JDK版本

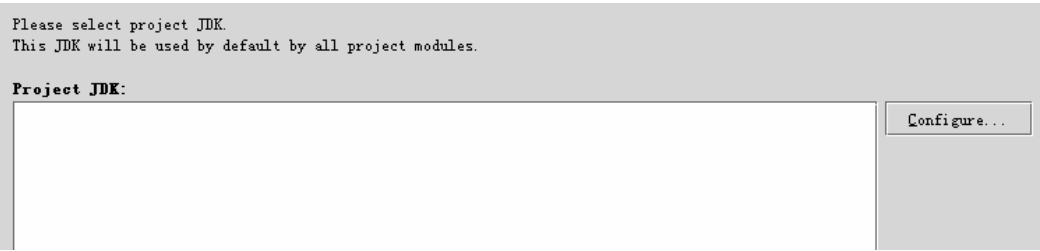


图2.5 选择JDK

2. 单击“Configure”。

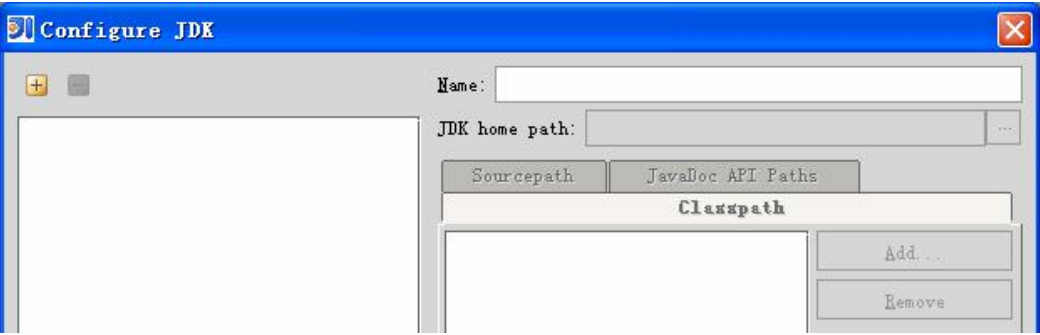



图2.6 配置jdk对话框

单击“Create JDK insert ”按钮，将出现“Select JDK Home Directory”对话框，选择一JDK的安装路径，如c:\jdk1.4.2_05。

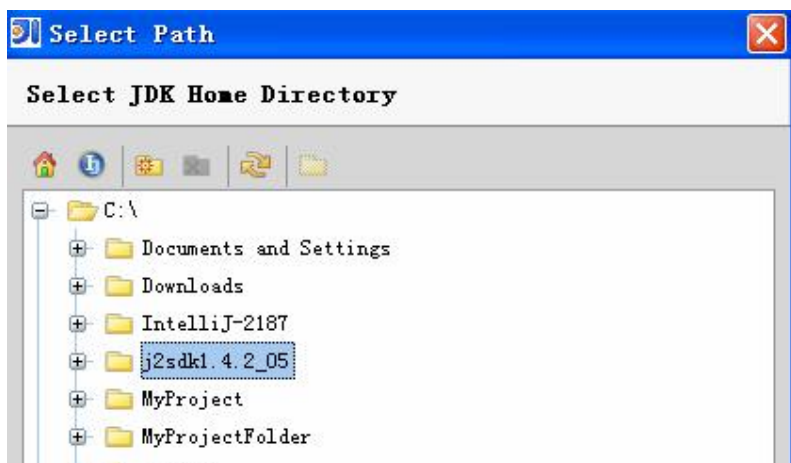


图2.7 选择jdk的home路径

3. 单击“OK”，jdk的配置就完成啦，且进入被选择使用的状态。

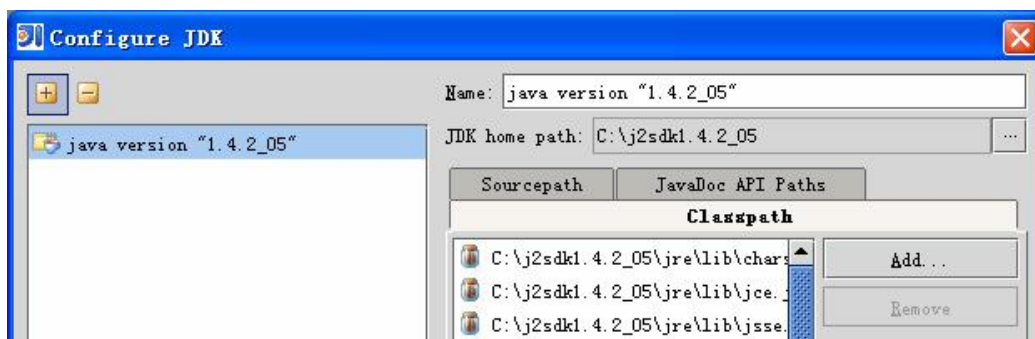


图2.8 配置jdk的窗口

单击“OK”，在“Project JDK”的设置栏将出现选中目标jdk。

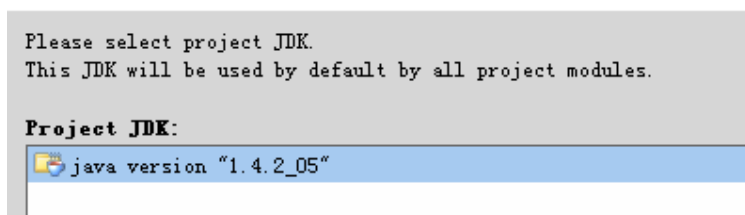


图2.9 选中目标jdk

2.1.3. 确定项目模块

现在需要确定项目的相关路径：

- 2.1.3.1. 模块属性选择
- 2.1.3.2. 模块名称/路径 (目录)
- 2.1.3.3. 源码路径
- 2.1.3.4. Classpath (25)

2.1.3.1. 模块属性选择

1. 此时你需选择你项目的模块属性，是否为多模块项目你的决定，也是根据实际项目而定的。

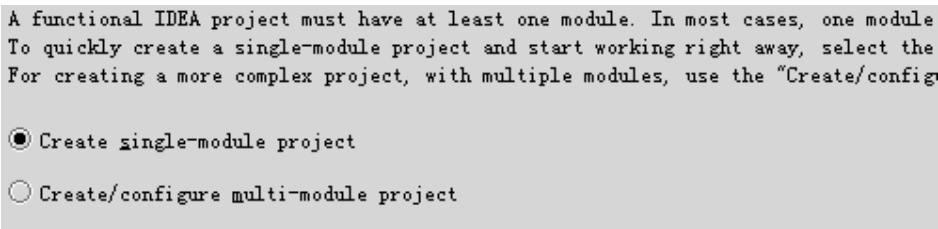


图2.10 项目模块属性选择

2. 选择单模块项目，点击“Next”，将出现欲创建模块的属性。

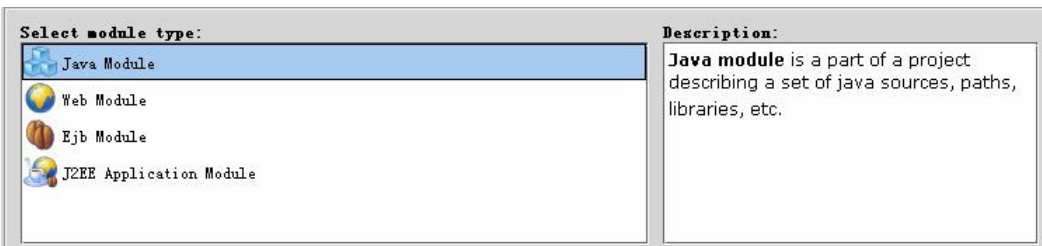


图2.11 所有的项目模块属性

3. 选择“Java Module”，我们将创建一个普通java模块。单击“Next”，将出现设置模块名称和存储目录的对话框。

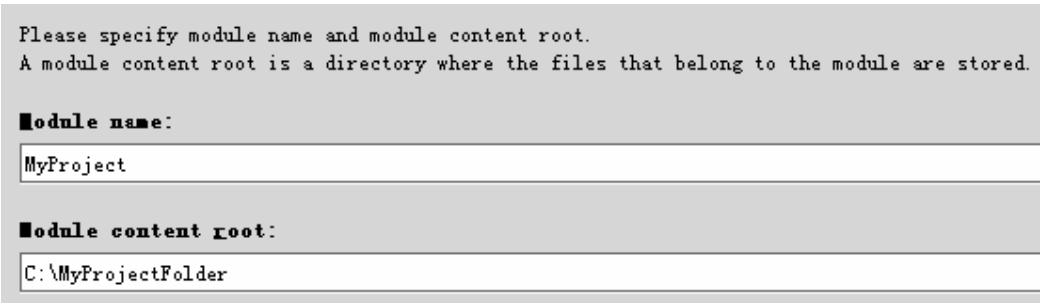


图2.12 设置模块名称和存储目录

4. 单击“Next”，将出现设置模块源文件存放的目录的对话框。

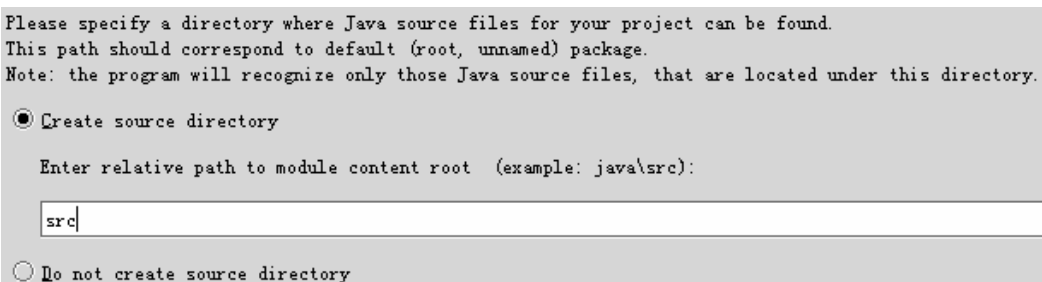


图2.13 设定模块的源文件路径

5. 单击“Next”，将出现设置模块编译输出路径。

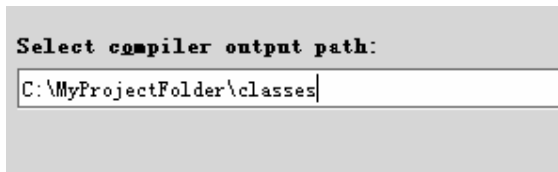


图2.14 模块编译的输出路径

6. 单击“Finish”，项目文件将被创建。出现主窗口和每日提示窗口。每日提示为显示IDEA平时操作的技巧，单击“Close”，关闭提示窗口。

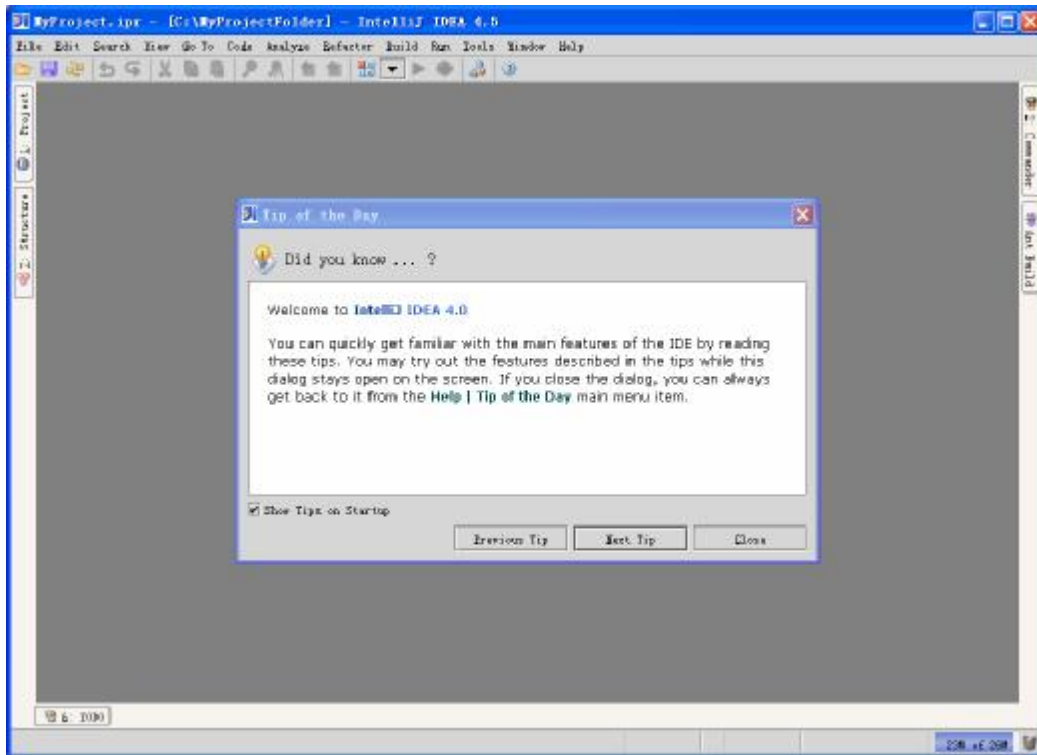


图2.15 主窗口和每日提示窗口

7. 单击左侧工具栏的“Project”按钮，将呈现项目的结构（源码和目录等）。

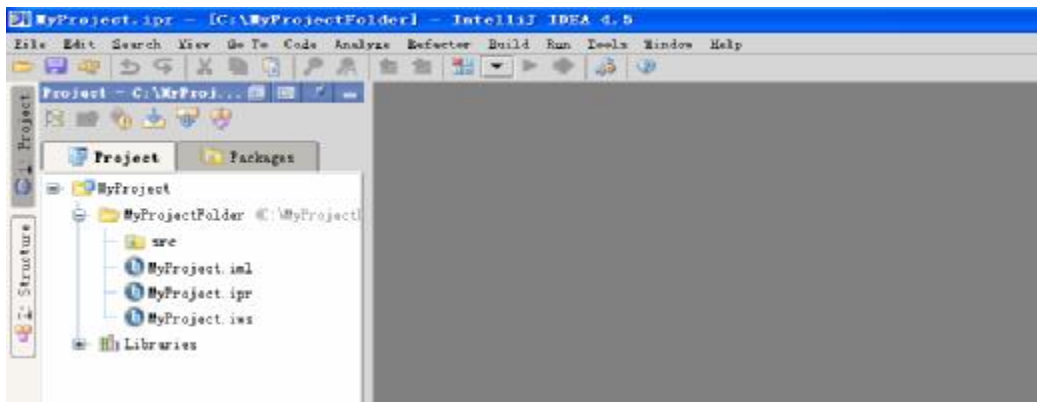


图2.16 项目的主窗口和项目的工具窗口

2.2. 创建包

1. 在项目的工具窗口中，右击src文件夹（项目的源码目录），在弹出的菜单中选择“New|package”。

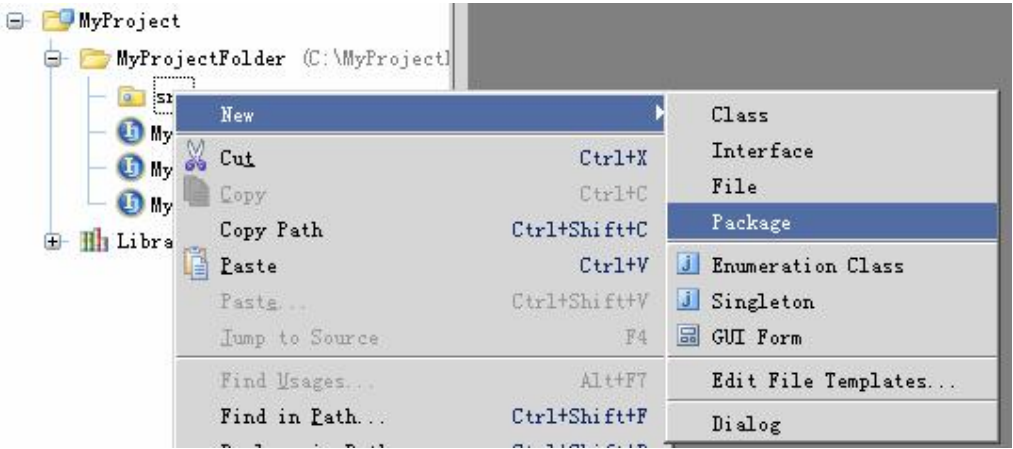


图2.17 创建包

2. 在出现的“New Package”对话框中输入需要建立的包名。**注意：**包可以级联创建，如你输入net.jetmaven，将建立两级目录；或基于父包创建子包等。



图2.18 输入包的名称

单击“OK”，包就被创建成功啦。

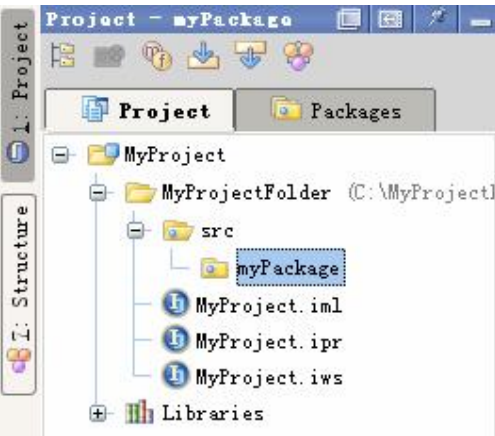


图2.19 被创建的包

2.3. 创建类

1. 右击包的名称，在弹出的菜单中选择“New|Class”，将出现“New Class”的对话框。

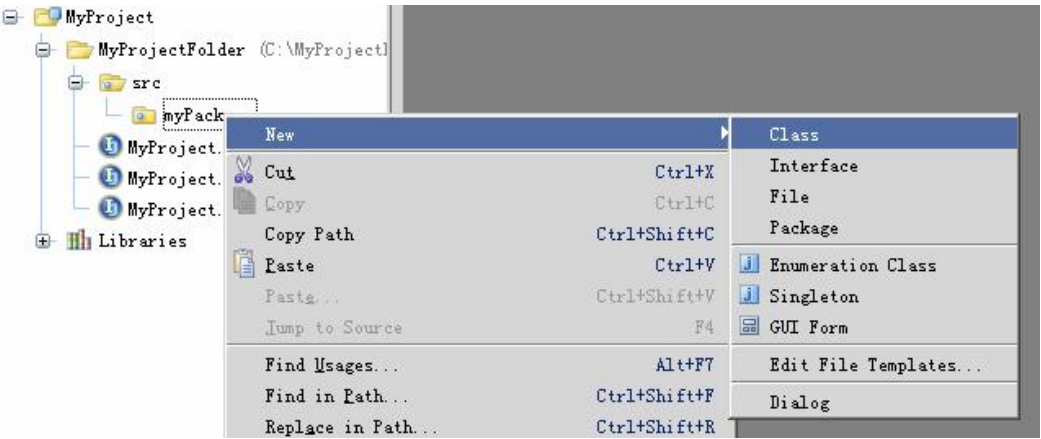


图2.20 创建class

2. 在“New Class”对话框中，输入要创建的类名，如MyClass。



图2.21 输入类名

单击“OK”，一个新类就被成功创建。

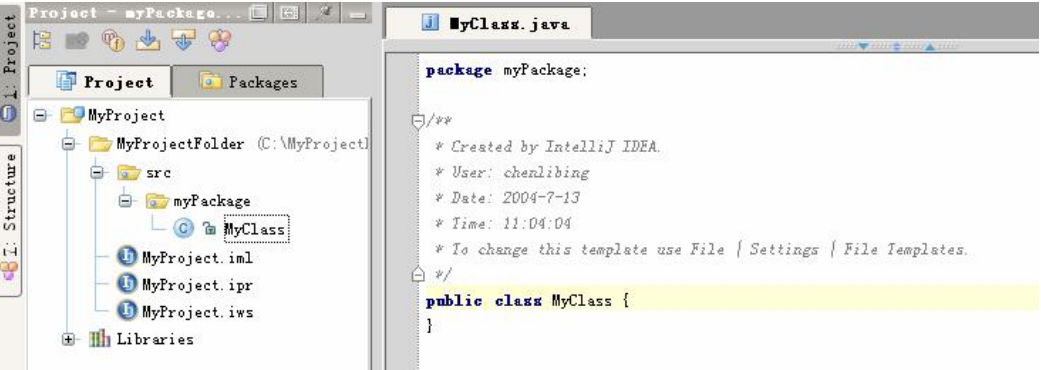


图2.22 创建的类MyClass

2.4. 使用编辑器

如果你毫无问题地抵达到这里，那么即将进入用idea编辑器编辑java源文件的时候。在这一节里，我们将讲述以下内容：

- 2.4.1. 打开 / 关闭文件
- 2.4.2. 输入文本
- 2.4.3. 查找 / 导航
- 2.4.4. 颜色和字体
- 2.4.5. 代码风格
- 2.4.6. 错误提示
- 2.4.7. Todo

2.4.1. 打开 / 关闭文件

在上一节中，我们创建了一个java类，并由编辑器自动打开。本节中我们将介绍如何打开和关闭文件。

1. 在文件标题栏右击文件名称，将弹出一个菜单。

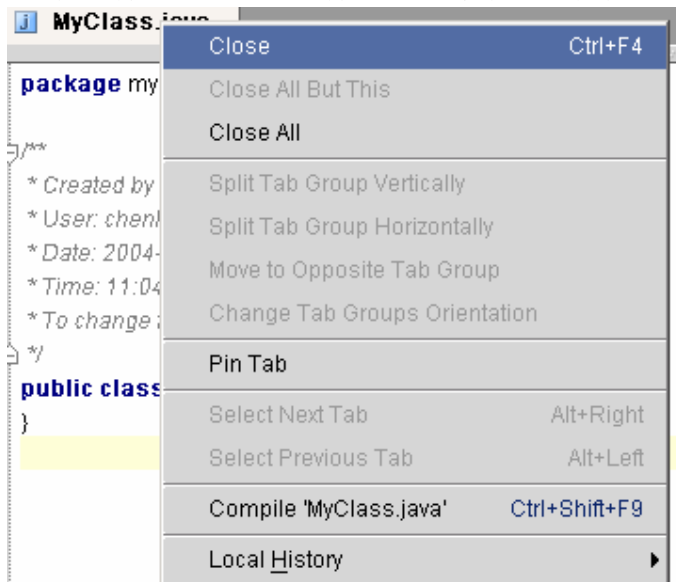


图2.23 右击标签栏的文件

选择“Close”关闭文件。

2. 从菜单栏选择“View|Recent Files”，将弹出一个菜单，显示以往打开的文件列表。

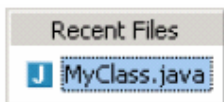


图2.24 最近打开文件列表

3. 关闭文件，在项目工具窗口双击要打开的文件，文件将被编辑器打开。在编辑窗口进行轻微改动，一个星号将出现在文件的标题栏上。

4. 按住“CTRL+S”，文件将被保存，标题栏的星号也将消失。



图2.25 文件未保存提示

2.4.2. Entering text

本节将示范用idea编辑文件的基本功能和操作。

1. 将光标置于第二行，要复制该行，请按“CTRL+D”。

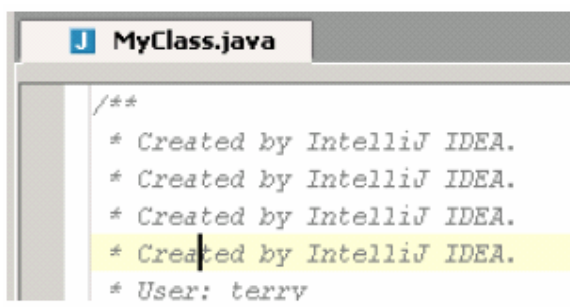


图2.26 复制一行

按住“CTRL+Y”，删除刚增加的行。

2. 进行标准的CTRL-C, CTRL-X, CTRL-V操作，执行拷贝，剪切，粘贴。

3. 按住CTRL+Z多次，撤销你刚才的所有动作。持续按多次，直至出现以下对话框。单击“OK”，可以撤销创建java类。按住CTRL+SHIFT+Z，将会出现“Redo create class”对话框，单击“OK”可以重新创建改类。

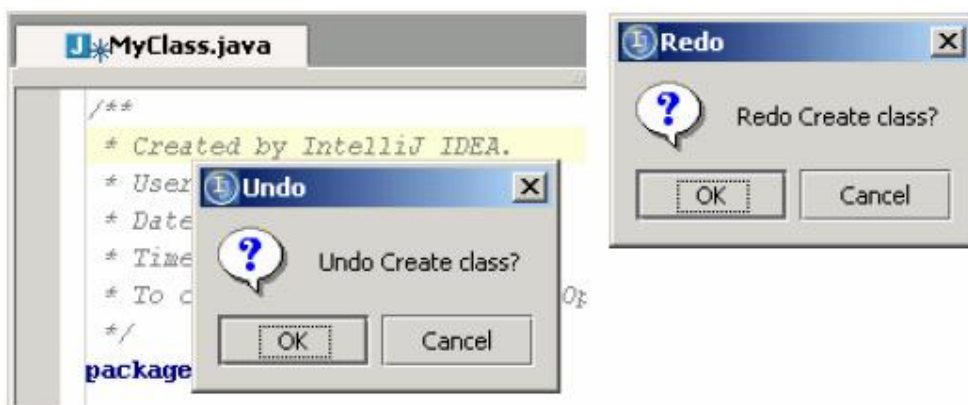


图2.27 撤销/重复类的创建

2.4.3. 查找 / 导航

1. 从菜单栏选择“Search|Find”，将出现“Find Text”对话框，键入你要查找的文本。

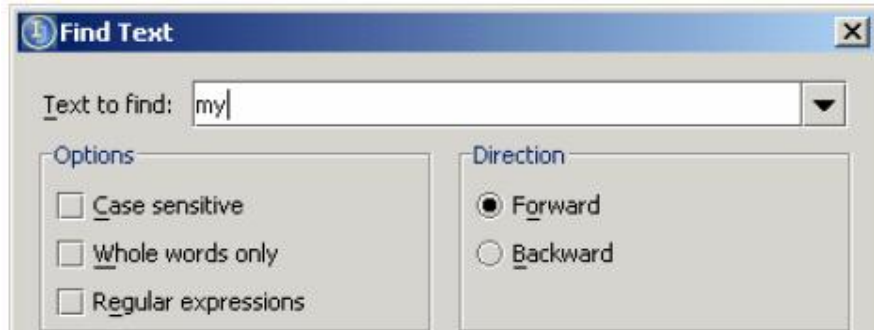


图2.28 查找窗口

2. 单击“Find”，当找到匹配的文本时，按下F3，将查找文件中下一个匹配的地方。在查找的对话框中，你可以测试一下“Case sensitive”，“Whole words”，“Regular expressions”等选项功能。

3. 关闭MyClass，选择“Goto |Class”，在弹出的窗口中键入你要跳转到的类名称。

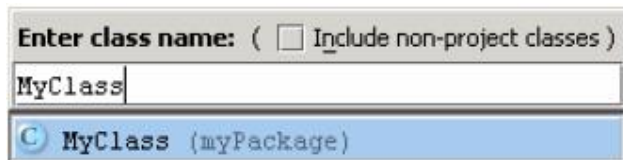


图2.29 跳转到指定的类

4. 按下回车，编辑器将打开指定的java类，将光标定位到某一行。

5. 从菜单选择“Edit|Toggle bookmark”，将当前的编辑信息（含行号）加入到收藏夹中。



图2.30 添加至收藏夹

关闭MyClass。从菜单中选择“Edit|Show bookmarks”，将会出现“Editor bookmarks”对话框。

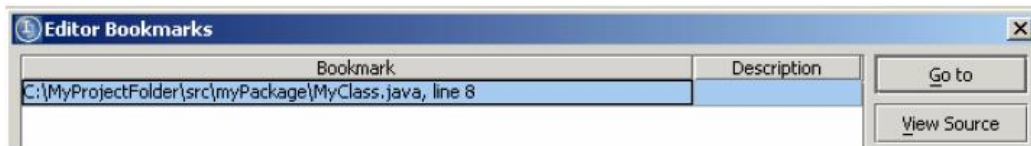


图2.31 收藏夹的列表

双击收藏夹列表中的一项，编辑器将打开文件并定位到指定位置。

2.4.4. 颜色和字体

将以下代码键入到MyClass类中：

```
package myPackage;
public class MyClass {
    public static void main(String[] args) {
        System.out.println("hello");
    }
}
```

注意颜色和字体。如果你项自定义字体和颜色，请打开“Options|IDE settings|colors & fonts”对话框进行设置。

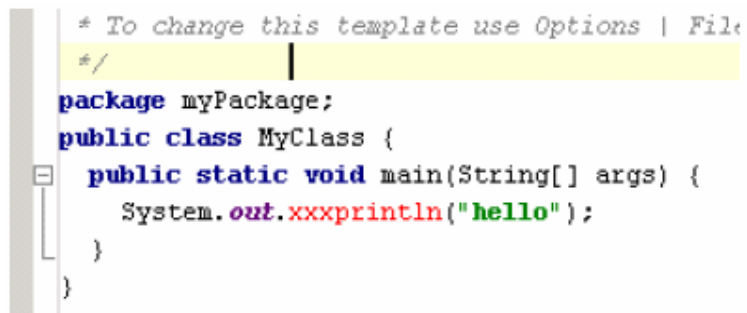


图2.32 颜色和字体

2.4.5. 代码风格

IDEA的编辑器使你的代码风格很容易维护。而且，在整理代码格式和风格的功能也很强大的。将MyClass的代码编辑如下：

```
package myPackage;public class MyClass {
public static void main(String[] args) {
System.out.println("hello");}}
```

尽管没有代码，以上的代码也是很难读的。解决以上问题，按照以下步骤：

1. 选中刚添加的代码，然后选择“Code|Auto-indent lines”优化代码的可读性。

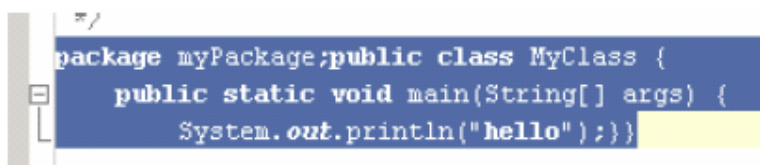


图2.33 行的自动缩进

2. 选择“Tools|reformat code”，将出现“Reformat Code”对话框。

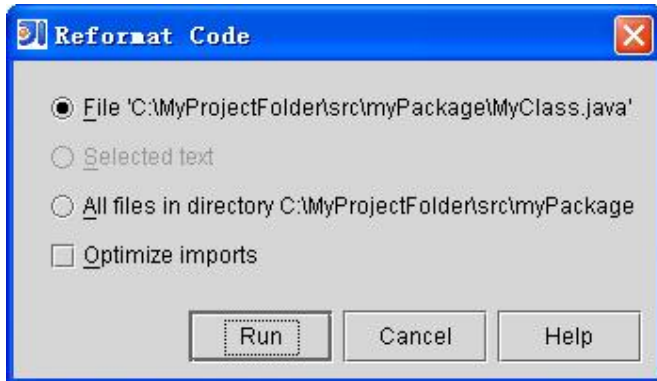


图2.34 格式化代码

单击“Run”，编辑器将根据默认代码风格设置对代码进行格式化。你可以打开“Options|IDE settings|code style”对话框对代码风格进行设置。

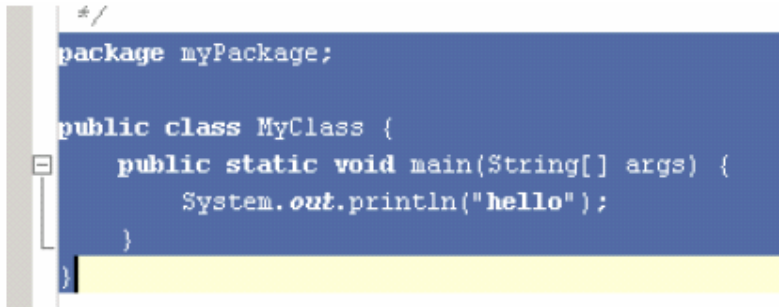


图2.35 格式化后的代码

2.4.6. 错误提示

将MyClass代码编辑如下：

```
package myPackage;
public xxclass MyClass {}
```

这时编辑器将提示出现错误。

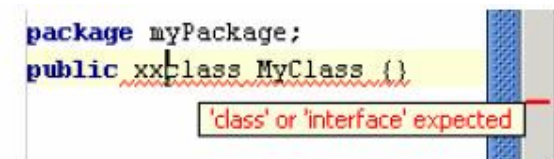


图2.36 错误提示

将MyClass代码继续编辑如下：

```
package myPackage;
import java.util.*;
public class MyClass {}
```

尽管没有代码错误，但是import是没有必要的。

如图2.39所示，IDEA将已灰色高亮显示不必要的import语句，并且在编辑器的右侧竖条中以黄色显示该警告。

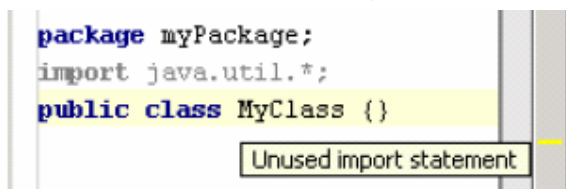


图2.37 无用的import警告

将光标置于import语句末尾，这是将出现一个球型灯的图片，单击建议图标将展现对此问题的解决方案。

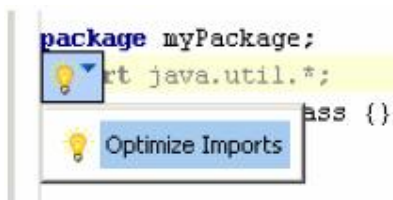


图2.38 优化import建议

双击“Optimize imports”删除没有必要的import语句。注意：如果你想自定义如何高亮显示错误，请打开“Options|IDE Settings|Errors”对话框进行相关设置。

2.4.7. Todo

将MyClass代码编辑如下：

```
package myPackage;  
//@todo add code  
public class MyClass {}
```

关闭MyClass类。

选择“Window|ToDo”打开TODO工具窗口。

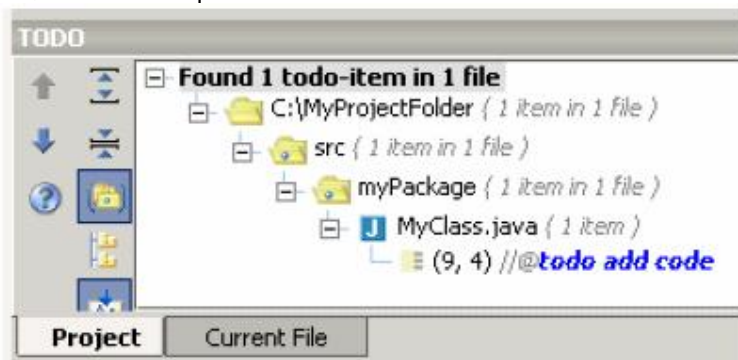


图2.39 TODO工具窗口

双击一todo条目，MyClass.java将被编辑器打开，将定位到指定的todo所在位置。

2.5. 自动化代码

按下图键入“psvm”文本。

```
public class MyClass{  
    psvm
```

图2.40 键入问题psvm

按下“Tab”键，此时新的代码将被添加到编辑器中。

```
public class MyClass{  
    public static void main(String[] args) {  
          
    }  
}
```

图2.41 “psvm”被动态模板文本替换

键入“psvm”然后按下“Tab”键将会使IDEA调用“public static void main(String[] args){}”替换“psvm”文本，这就是动态模板的例子。IDEA允许你编辑或创建动态模板。

我们再来尝试更多的例子。按下图键入文本“itar”：

```
public class MyClass{  
    public static void main(String[] args) {  
        itar  
    }  
}
```

图2.42 键入文本“itar”

按下“Tab”键。

```
public class MyClass{  
    public static void main(String[] args) {  
        for (int i = 0; i < args.length; i++) {  
            String arg = args[i];  
        }  
    }  
}
```

图2.43 itar动态模板文本

键入“myI”然后按回车，请注意变化。

```
public static void main(String[] args) {  
    for (int myI = 0; myI < args.length; myI++) {  
        String arg = args[myI];  
    }  
}
```

图2.44 itar动态模板文本

再按回车，一个推荐的字符串名称将呈现出来。

```

for (int i = 0; i < args.length; i++) {
    String arg = args[i];
}

```

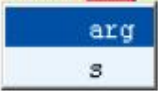


图2.45 推荐的字符串名称

双击或选择回车选择名称s。在下一行输入动态文本“sout”：

```

String s = args[myI];
sout

```

图2.46 sout

按下“Tab”键，将产生的文本修改为如下文本：

```

System.out.println("arg " + myI + " = " + s);

```

2.6. 代码重构

选中变量s，然后选择“Refactor|rename”，将出现“Rename”对话框，键入sReNamed。

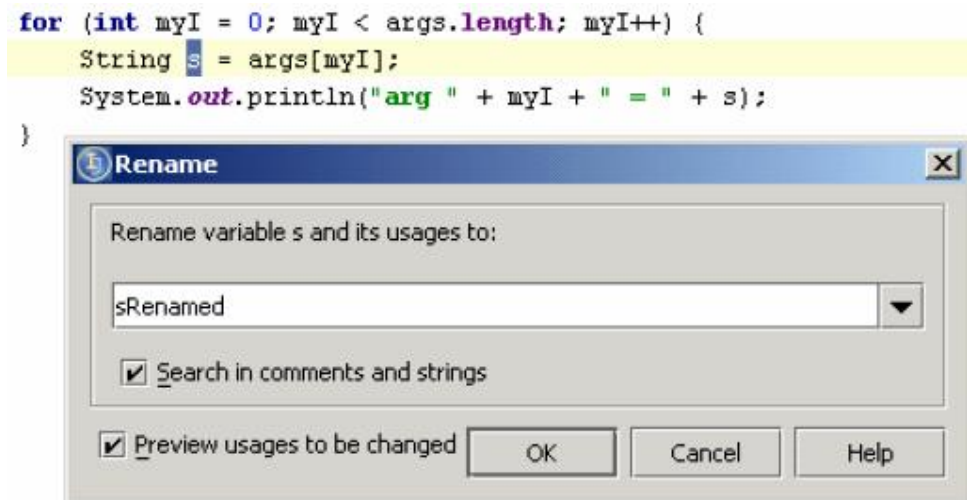


图2.47 重命名变量

单击“OK”，重构预览窗口将出现。

单击“Do Refactor”，变量将被重命名。请注意，“Do Refactor”按钮是进行所有的重构操作的入口按钮。

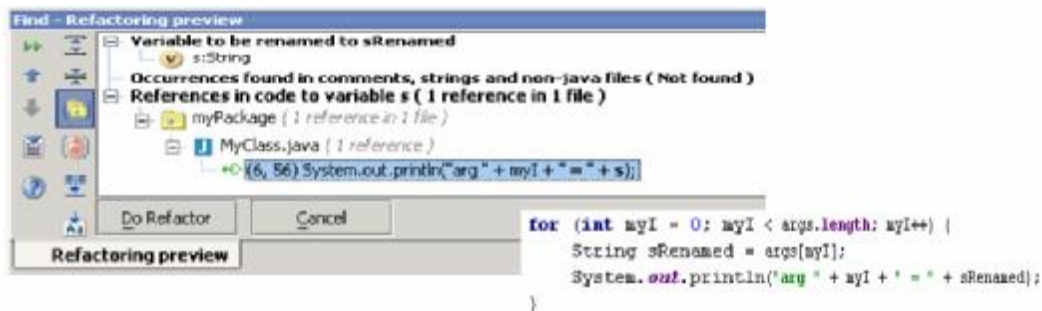


图2.48 重构预览和结果

2.7. 使用版本控制

选择“Tools|Local VCS|Show History”，将出现文件的历史窗口。
单击“Renaming variable s to sRenamed”的动作项。

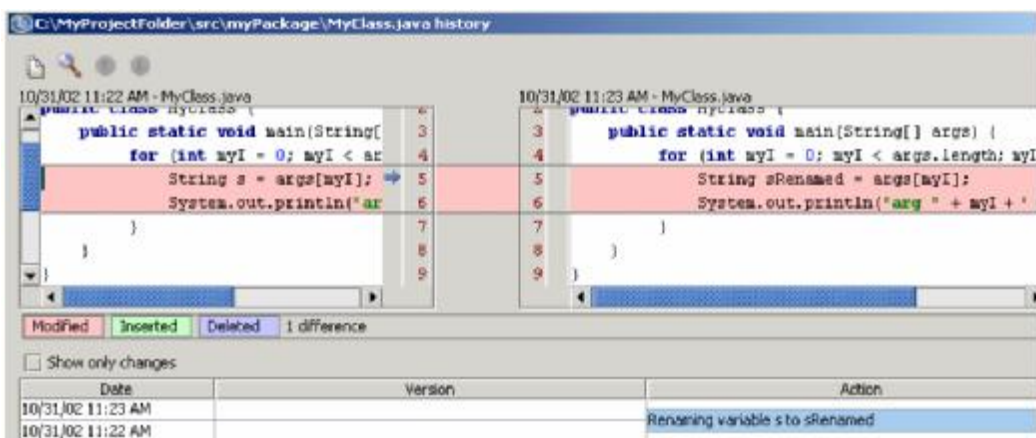


图2.49 历史项中的重命名动作项

右击动作项，将弹出“Rollback”，单击“Rollback”，将出现确认对话框，询问你是否回滚到上一版本，单击“OK”，所作的重命名操作将被回滚。

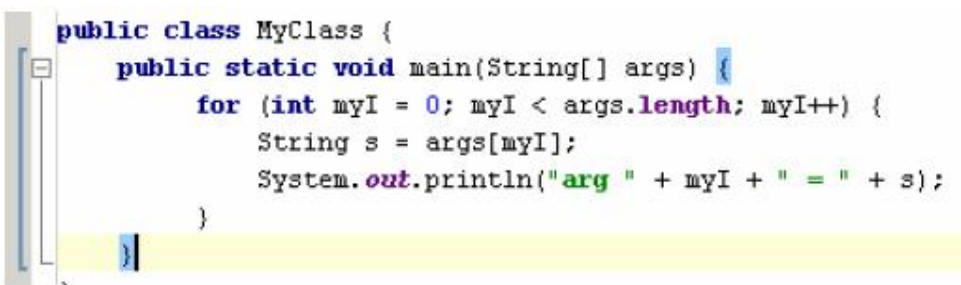


图2.50 回滚指定的操作

2.8. 查看JavaDoc

IDEA的javadoc功能使得创建和操作javadoc变得更加容易。

```
public static void main(String[] args) {
```

键入 “/**” 。按下回车键，javadoc的tag将自动完成。

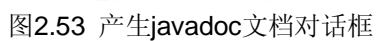
图2.51 javadoc方法的tag

```
public class MyClass {
    /**
     *
     * @param args
     */
    public static void main(String[] args) {
        for myPackage.MyClass
    }
}
```

Parameters:

args -

选择“Tools|Generate JavaDoc”，将出现“Generate JavaDoc”对话框，在“Output directory”选项中键入“c:\MyProjectFolder”。



单击“Start”，javadoc文档将被生成。

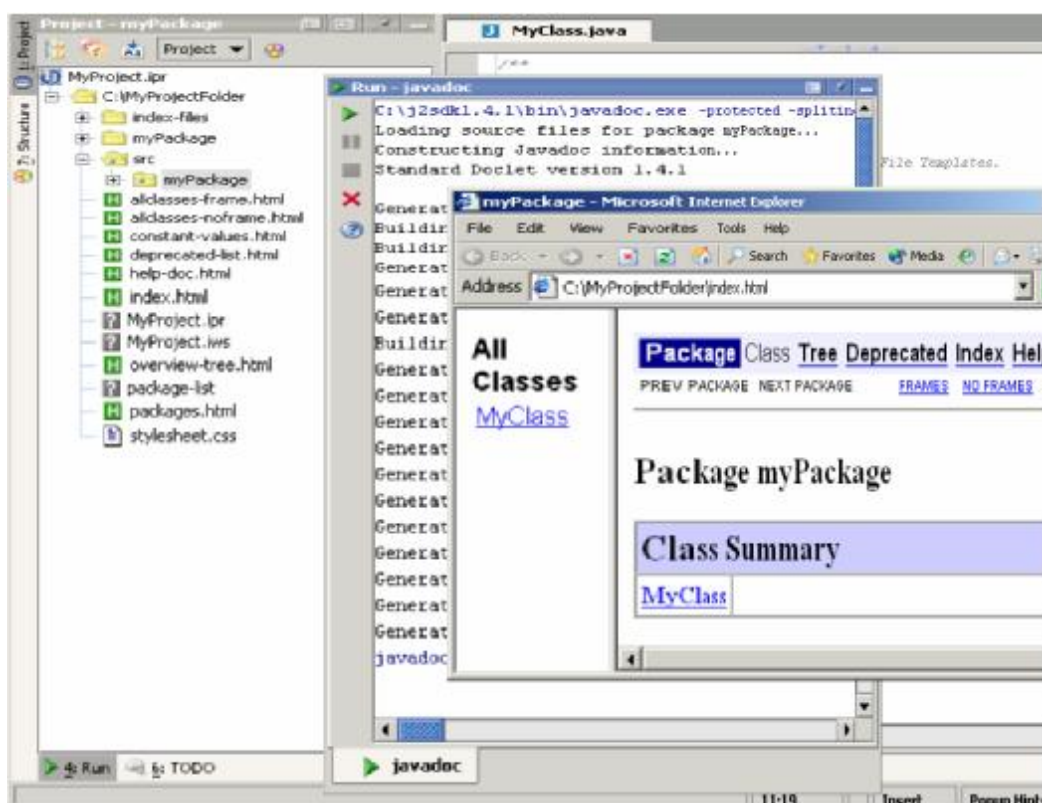


图2.54 MyClass的javadoc文档

2.9. 编译

编辑MyClass（添加一个错和deprecated方法）。

```
System.out.println("arg " + myI + " = " + s);  
Thread.currentThread().resume();
```

从主菜单中选择“Select | compile MyClass.java”或按下“CRTL+SHIFT+F9”，将会弹出一个消息窗口，显示编译中的错误和警告。

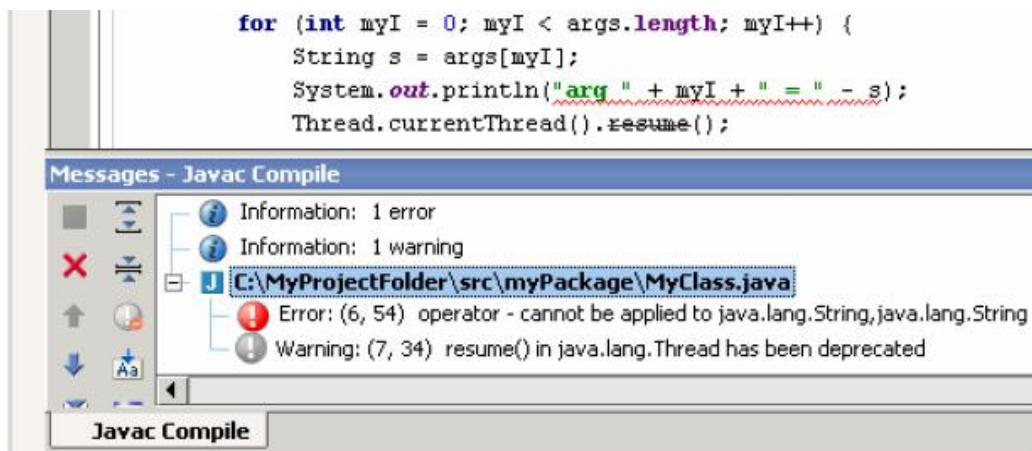


图2.55 显示警告和错误信息消息窗口

纠正错误（将“-”该为“+”），删除以下这行。

```
Thread.currentThread().resume();
```

重新编译。“C:\MyProjectFolder\MyOutputFolder\myPackage\MyClass.class”文件将被创建。



图2.56 编译文件

2.10. 调试

现在你可以准备调试你的程序。调试程序，你需要：

- 2.10.1. 创建应用配置 (44)
- 2.10.2. 运行配置 (46)
- 2.10.3. 设置断点 (46)
- 2.10.4. 程序中的step (47)

2.10.1. 创建应用配置


在主菜单中选择“Run|Run”或“Shift+F10”，将出现“run”对话框，选择“application”标签。单击，将除出现“Unnamed”应用。



图2.57 “Unnamed”应用

将名称更改为“MyApplication”，在“Main class”中单击省略号按钮，将会出现“Choose Main Class”窗口。

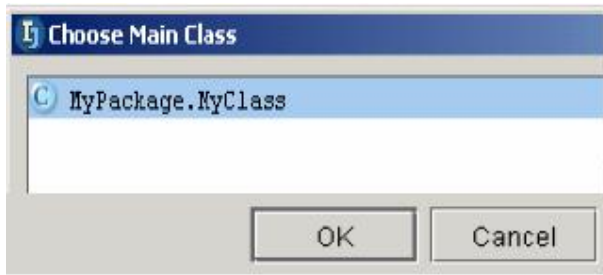


图2.58 选择运行的主java类

选择“myPackage.MyClass”，然后单击“OK”。

在“programe parameters”中输入“param1 param2 param3”。

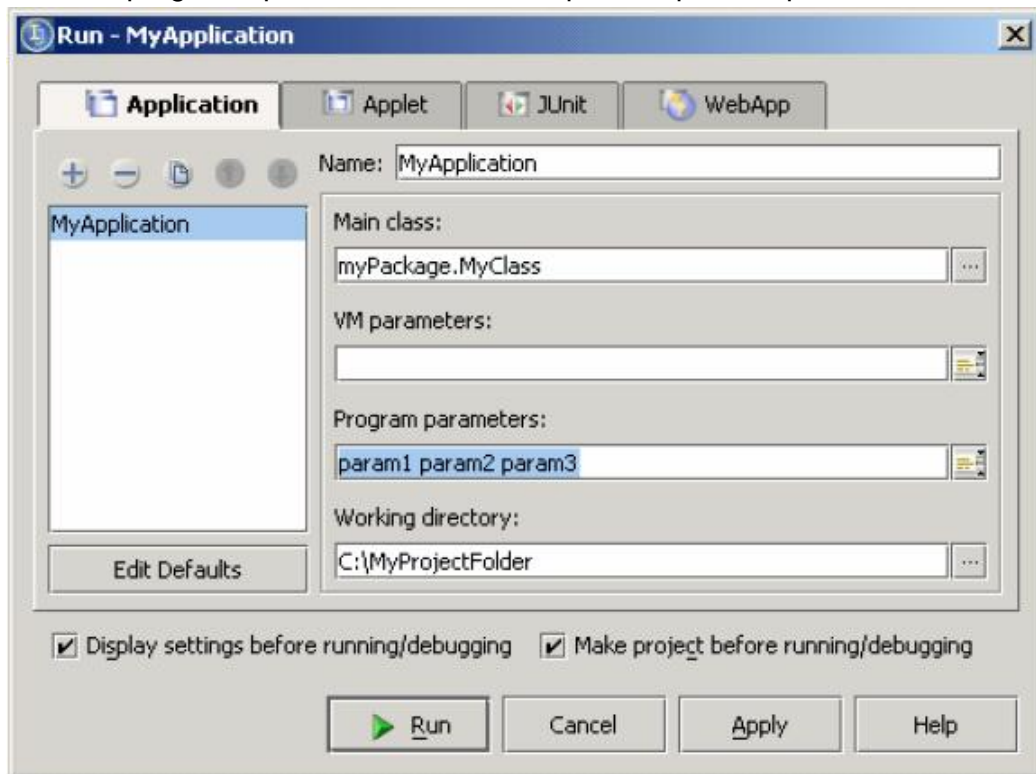


图2.59 应用设定

2.10.2. 运行指定配置

单击“Run”，将出现“Run”工具窗口，程序的输出消息全部将显示在“Run”工具窗口。

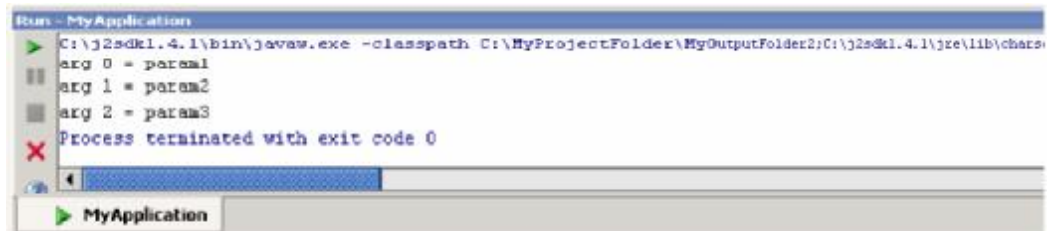


图2.60 “Run” 工具窗口捕获的输出

2.10.3. 设置断点

在 “System.out.println...” 语句前的竖条中将出现一个行断点标识。

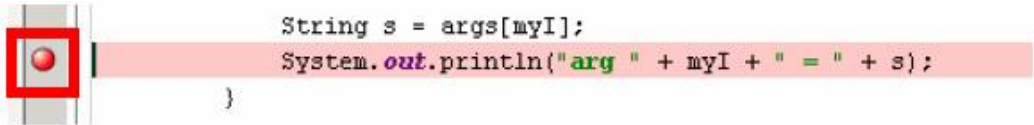


图2.61 设置行的断点

右击断点标识，在弹出的菜单中选择 “Properties”，将出现 “BreakPoints” 对话框。

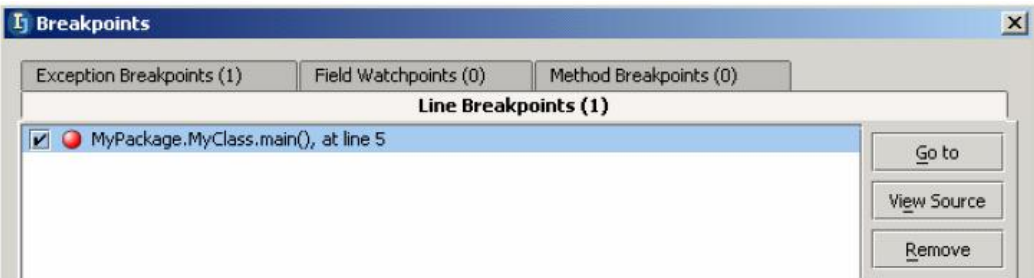



图2.62 “Breakpoints” 对话框

注意：可以从 “Run|View Breakpoints” 或 “CTRL+SHIFT+F8” 打开此对话框。

2.10.4. 程序调试跟踪

单击调试按钮, 将会出现 “Debug – MyApplication” 对话框。单击 “Debug” 按钮开始调试程序，此时将出现调试的工具窗口，程序在断点处暂停运行。

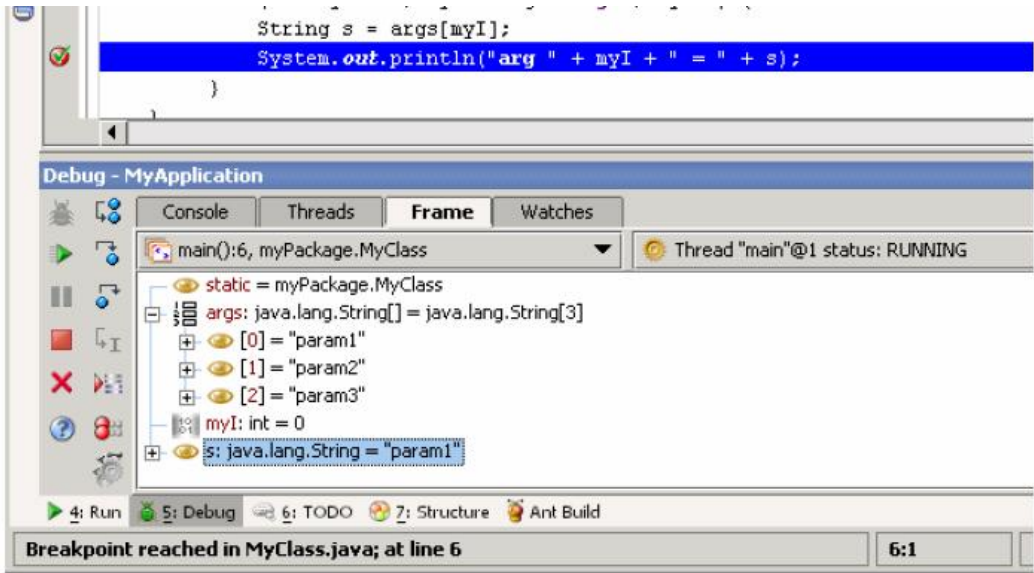



图2.63 调试的工具窗口

单击继续运行按钮，程序继续运行，并在下一个断点处暂停运行。右击断点标识，在弹出的菜单中选择“Disable”，断点将被取消。

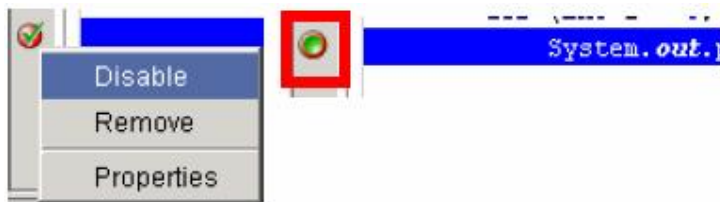


图2.64 取消断点

单击“Resume”按钮，程序运行完成。