

## 目录

<b>ECLIPSE插件开发.....</b>	<b>4</b>
<b>一、 插件开发概述.....</b>	<b>4</b>
1. 简述 .....	4
2. 基本步骤 .....	4
<b>二、 新建插件工程.....</b>	<b>5</b>
1. 新建工程 .....	5
2. 工程名称 .....	5
3. 插件属性 .....	6
4. 选择模板 .....	7
5. 设置视图属性 .....	8
6. 运行插件 .....	9
7. 运行结果 .....	10
8. 异常处理 .....	11
<b>三、 PLUG-IN.XML说明 .....</b>	<b>12</b>
1. OVERVIEW .....	12
2. DEPENDENCIES.....	13
3. RUNTIME .....	13
4. EXTENSIONS.....	14
5. EXTENSION POINTS.....	15
6. BUILD .....	15
7. MANIFEST.MF .....	15
8. PLUGIN.XML.....	15
9. BUILD.PROPERTIES.....	16
<b>四、 视图.....</b>	<b>17</b>
1. 概述 .....	17
2. 打开视图 .....	17
3. CATEGORY .....	18
4. VIEW .....	18
5. STICKYVIEW.....	18
6. 视图对应的类 .....	19
7. 新建视图 .....	19
8. 创建视图界面 .....	19
9. 运行结果 .....	20
<b>五、 透视图 .....</b>	<b>21</b>
1. 概述 .....	21
2. 扩展已有透视图 .....	21
1) 透视图扩展点 .....	21

2)	添加视图.....	21
3.	打开透视图操作.....	22
1)	新建Action。.....	22
2)	修改按钮绑定的Action。.....	23
3)	运行结果。.....	23
4.	新建透视图.....	24
1)	添加透视图扩展点.....	24
2)	新建透视图.....	24
3)	类的实现。.....	25
4)	运行结果。.....	25
<b>六、</b>	<b>编辑器.....</b>	<b>26</b>
1.	概述.....	26
2.	扩展编辑器.....	26
1)	声明编辑器扩展点.....	26
2)	实现com.galaxy.tutorial.editor.SampleEditor。.....	27
3)	运行结果.....	30
3.	扩展多页编辑器.....	30
1)	声明编辑器扩展点.....	31
2)	引入依赖的插件.....	31
3)	实现com.galaxy.tutorial.editor.SampleMultiPageEditor。.....	31
4)	运行结果.....	32
<b>七、</b>	<b>向导.....</b>	<b>32</b>
1.	概述.....	32
1)	向导.....	32
2)	向导页.....	33
3)	向导容器.....	33
2.	扩展向导.....	33
1)	声明向导扩展点.....	33
2)	新建向导类别.....	34
3)	新建向导.....	34
4)	实现向导类.....	35
5)	运行结果.....	37
<b>八、</b>	<b>上下文操作.....</b>	<b>38</b>
1.	声明上下文菜单扩展点.....	38
2.	对象上下文操作.....	39
1)	新建对象操作.....	39
2)	新建操作.....	39
3)	实现操作的类.....	40
4)	运行结果.....	42
3.	视图上下文操作.....	42
1)	新建视图操作.....	42
2)	新建一个菜单.....	43

3)	新建操作.....	43
4)	实现操作的类.....	44
5)	运行结果.....	46
4.	编辑器上下文操作 .....	46
<b>九、</b>	<b>操作集 .....</b>	<b>46</b>
1.	WORKBENCH菜单栏和工具栏操作集 .....	46
1)	声明workbench 操作集扩展点 .....	46
2)	新建操作集 .....	46
3)	新建菜单.....	47
4)	新建操作.....	47
5)	实现类 .....	48
6)	运行结果.....	50
2.	视图菜单栏和工具栏操作集 .....	51
1)	声明视图操作集扩展点 .....	51
2)	新建viewContribution.....	51
3)	新建菜单.....	52
4)	新建操作.....	52
5)	实现类 .....	52
6)	运行结果.....	54
3.	编辑器菜单栏和工具栏操作集 .....	54
1)	声明编辑器操作集扩展点.....	54
2)	新建editorContribution.....	55
3)	新建操作.....	55
4)	实现类 .....	56
5)	运行结果.....	57
<b>十、</b>	<b>首选项 .....</b>	<b>57</b>
1.	声明首选项扩展点 .....	58
2.	扩展首选项 .....	58
1)	新建首选项页.....	58
2)	实现类 .....	58
3)	运行结果.....	61
<b>十一、</b>	<b>属性视图 .....</b>	<b>61</b>
1.	声明属性页扩展点 .....	61
2.	扩展属性页 .....	62
1)	新建属性页 .....	62
2)	实现类 .....	62
3)	运行结果.....	64

# Eclipse 插件开发

## 一、 插件开发概述

### 1. 简述

Eclipse 由一个很小的核心和核心之上的大量插件组成。有些插件仅仅是供其它插件使用的库。其中存在很多你可以利用的工具。所有插件使用的基础库是：

**标准 Widget 工具包 (SWT)：**Eclipse 中处处使用的图形化组件：按钮，图像、光标、标签等等。布局管理类。通常这个库被用于代替 Swing。

**JFace：**菜单、工具条、对话框、参数选择、字体、图像、文本文件的类和向导基类。

**插件开发环境 (PDE)：**辅助数据操作、扩展、建立过程和向导的类。

**Java 开发者工具包 (JDT)：**用于编程操作 Java 代码的类。

上面的每一个类都有自己专有的功能，其中一些还可以单独使用（尽管它们内在地依赖于其它类）。例如，SWT 不仅仅只用于插件；它还可以被用于建立非 Eclipse 的、独立的应用程序。

还有一些其它的库没有被列举出来。下图显示了 Eclipse 不同层次之间的关系。

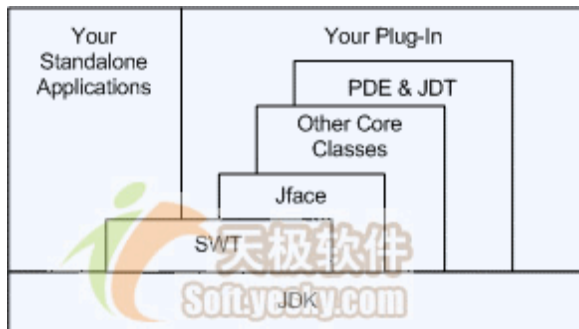


图 1：分层的类库

### 2. 基本步骤

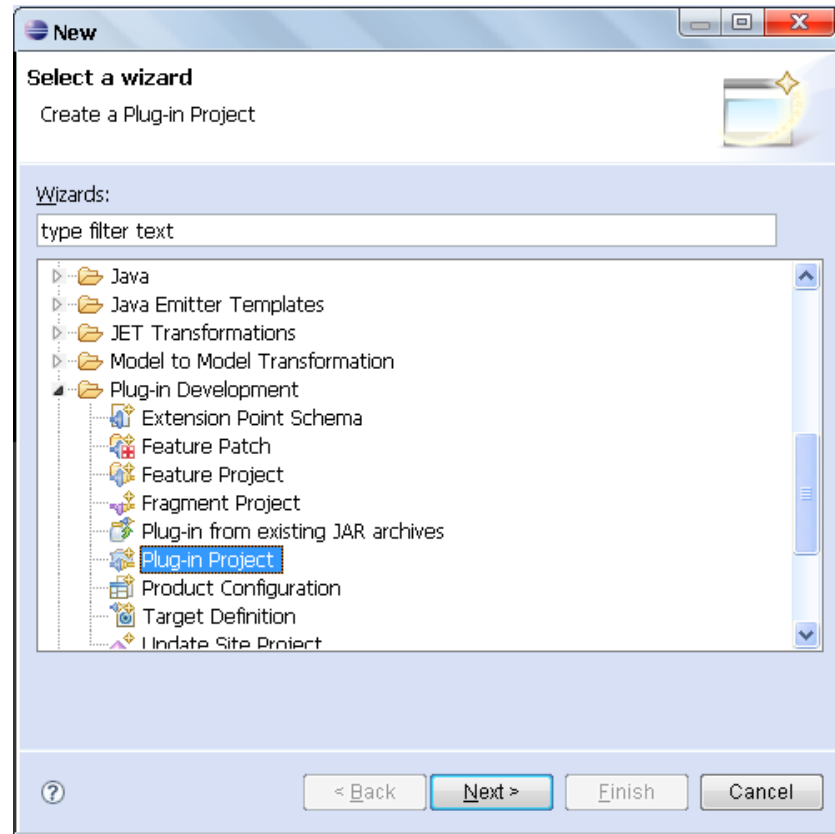
插件通过添加到预定义的扩展点来向平台添加功能。要将程序代码变成插件，需要：

- 决定插件如何与平台集成
- 标识需要进行添加的扩展点以便与开发者的插件进行集成
- 根据扩展点的规范来实现这些扩展
- 提供清单文件 `plugin.xml`，它描述开发者正在提供的扩展以及代码的封装
- 测试插件
- 对插件进行封装

## 二、新建插件工程

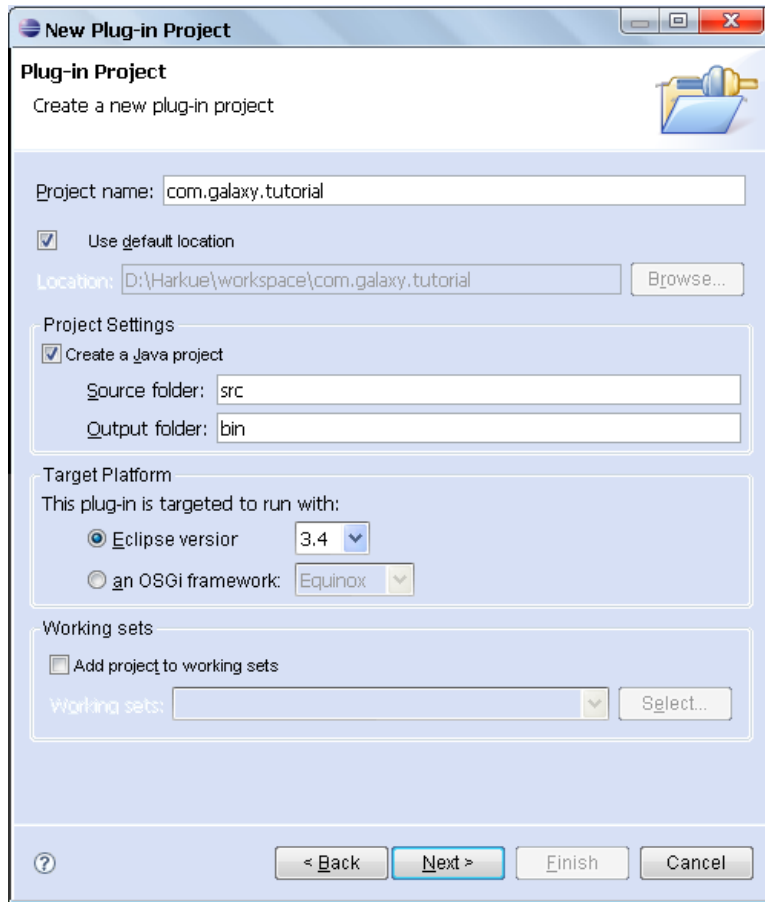
### 1. 新建工程

File→New→Project。选择 Plug-in Project，点击 Next。



### 2. 工程名称

填写工程名称 com.galaxy.tutorial，其他默认，点击 Next.



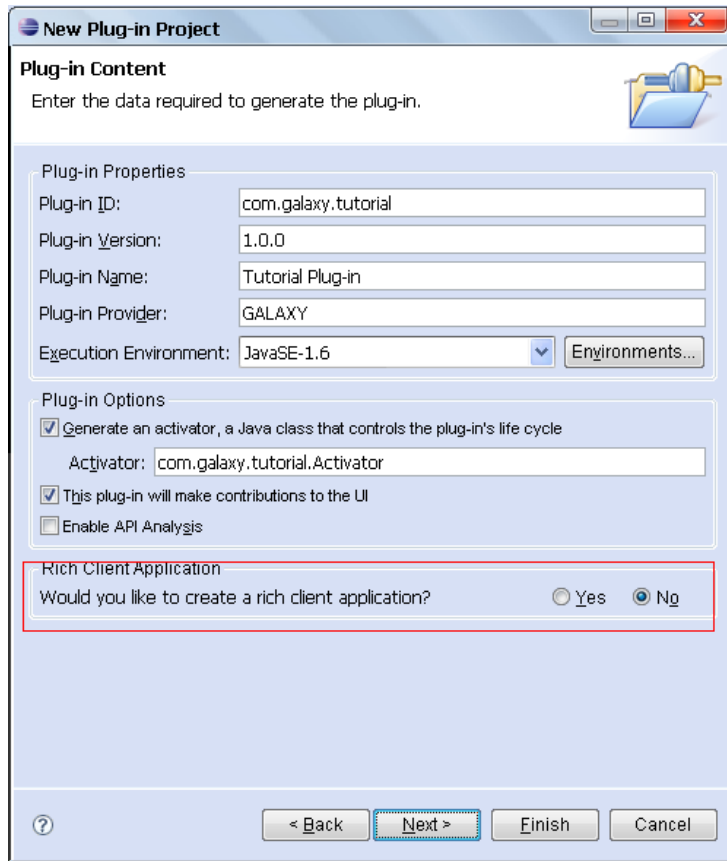
### 3. 插件属性

默认各项属性，点击 **Next**。

**Rich Client Application** 选项

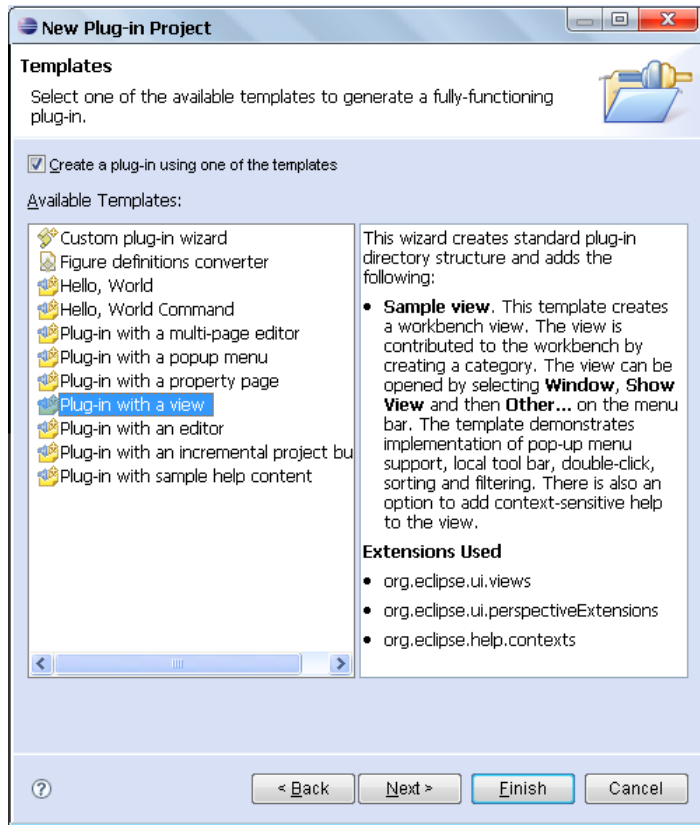
**Yes**。程序将以富客户端的方式即应用程序运行。

**No**。程序将以插件的形式绑定 **Eclipse** 平台运行。



#### 4. 选择模板

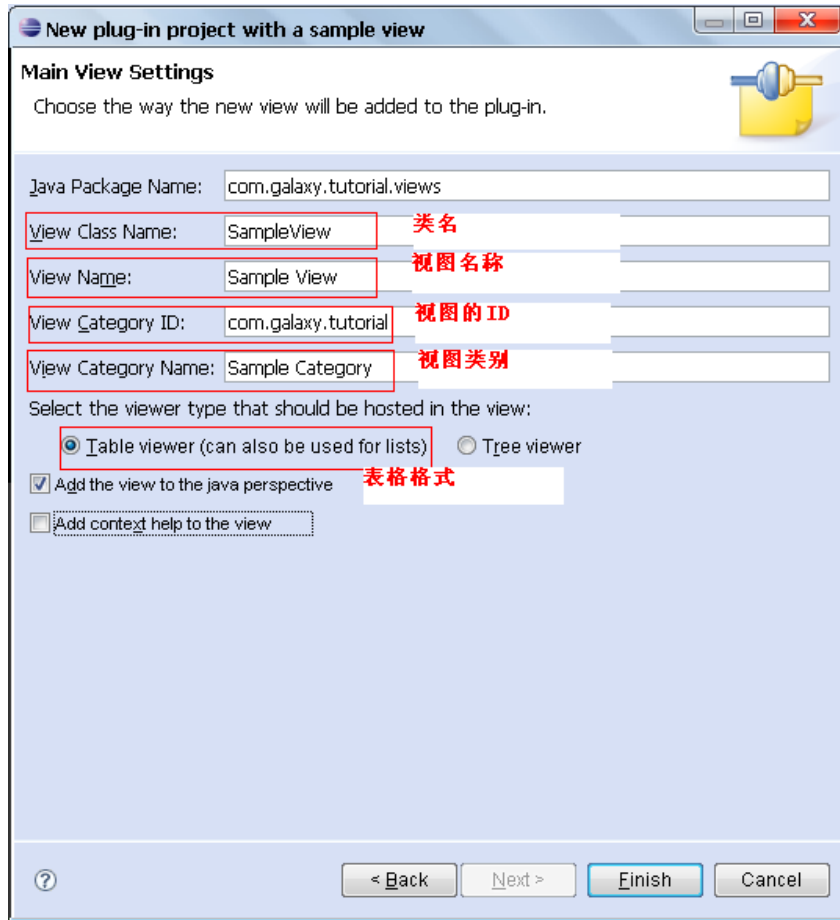
选择一个模板 Plug-in with a view，点击 Next。




## 5. 设置视图属性

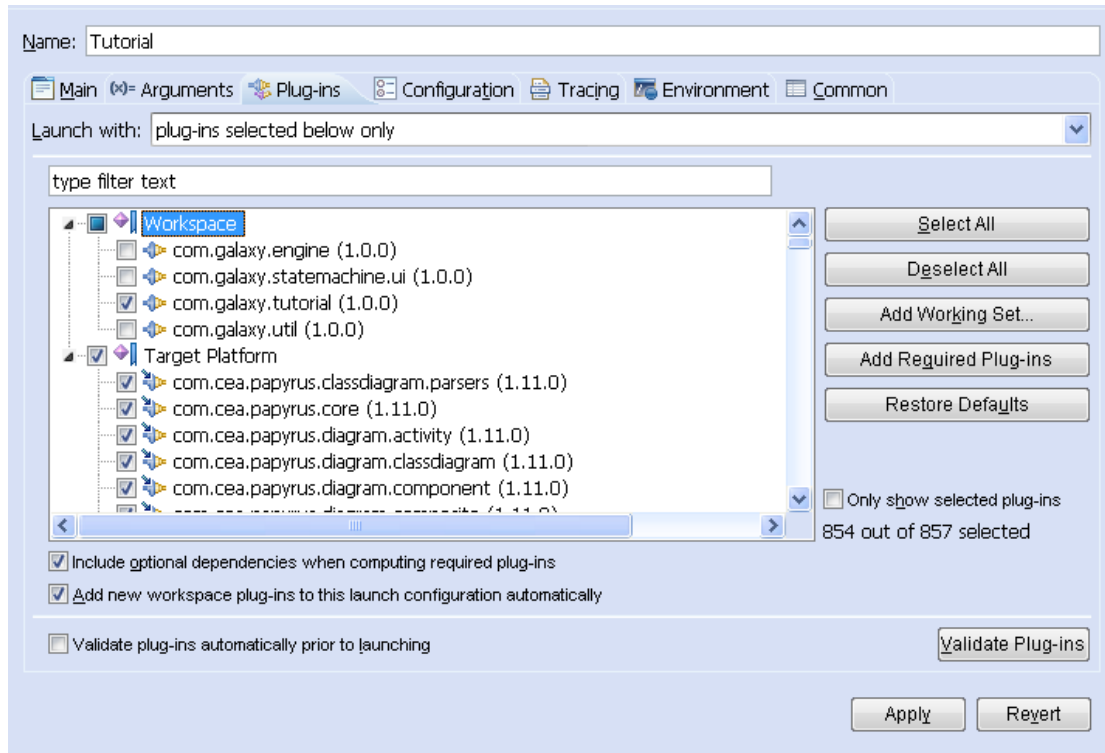
默认，点击 Finish。





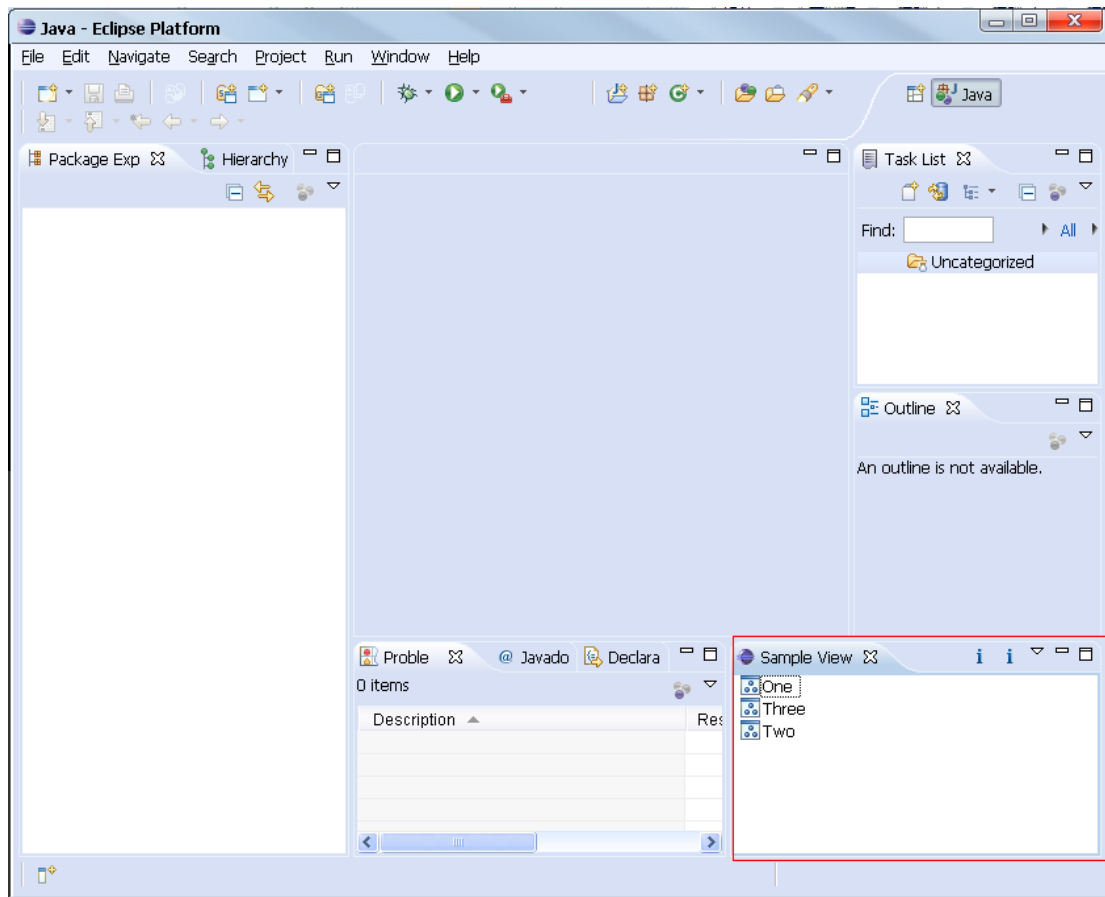
## 6. 运行插件

点击  下三角，选择 Run Configurations..，双击 Eclipse Application，新建一个 Run Configuration。修改 Name: Tutorial，切换到 Plug-in 选项页，修改 Launch with 选项，去掉列表中不需要运行的插件。点击 Apply，再点击 Run。



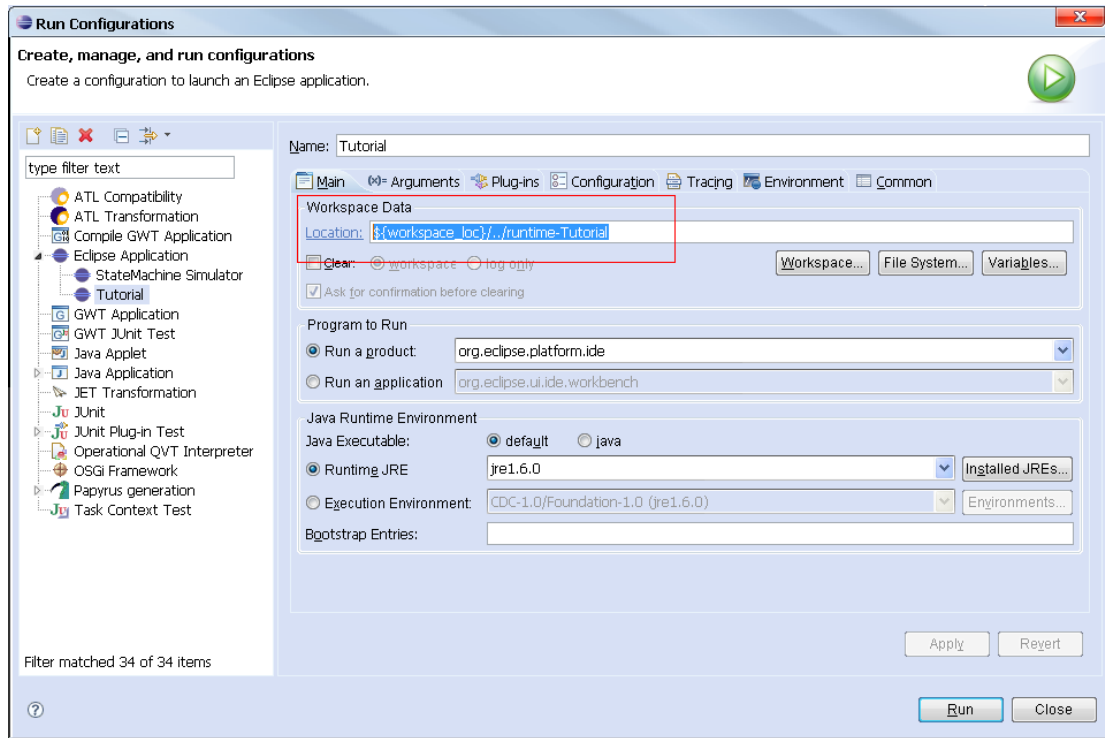
## 7. 运行结果

红色部分就是新构建的视图。



## 8. 异常处理

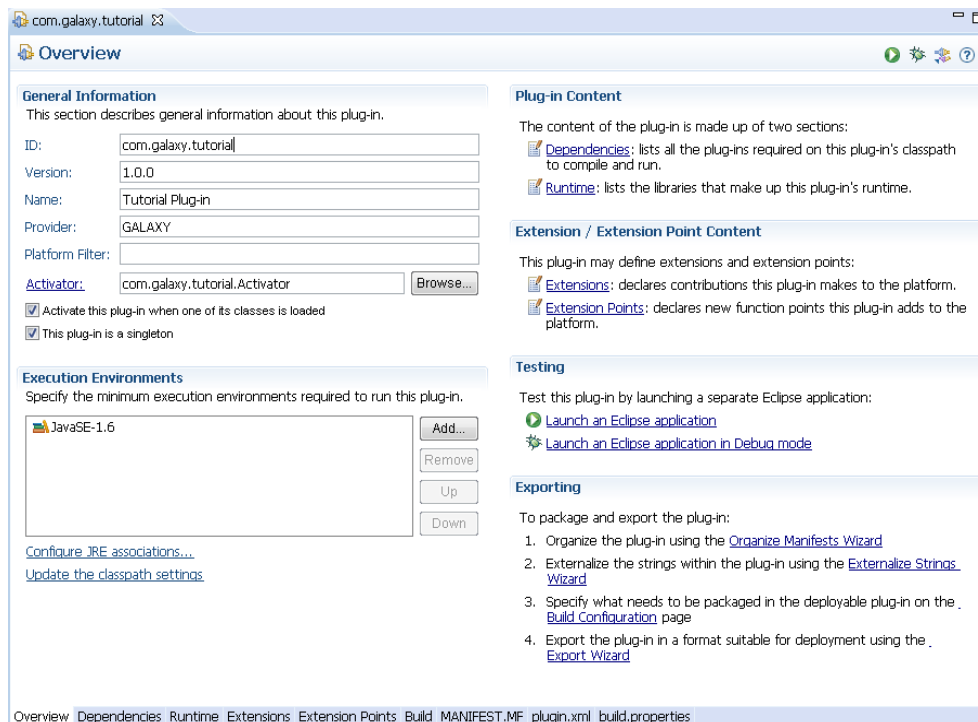
假如报错，到下图指定的 Location 目录下，打开.metadata 文件夹，查看日志.log 文件。若出现 `org.osgi.framework.BundleException: The activator com.galaxy.tutorial.Activator for bundle com.galaxy.tutorial is invalid` 错误，见第三章 `plug-in.xml` 文件说明之 Build 部分。



### 三、 Plug-in.xml 说明

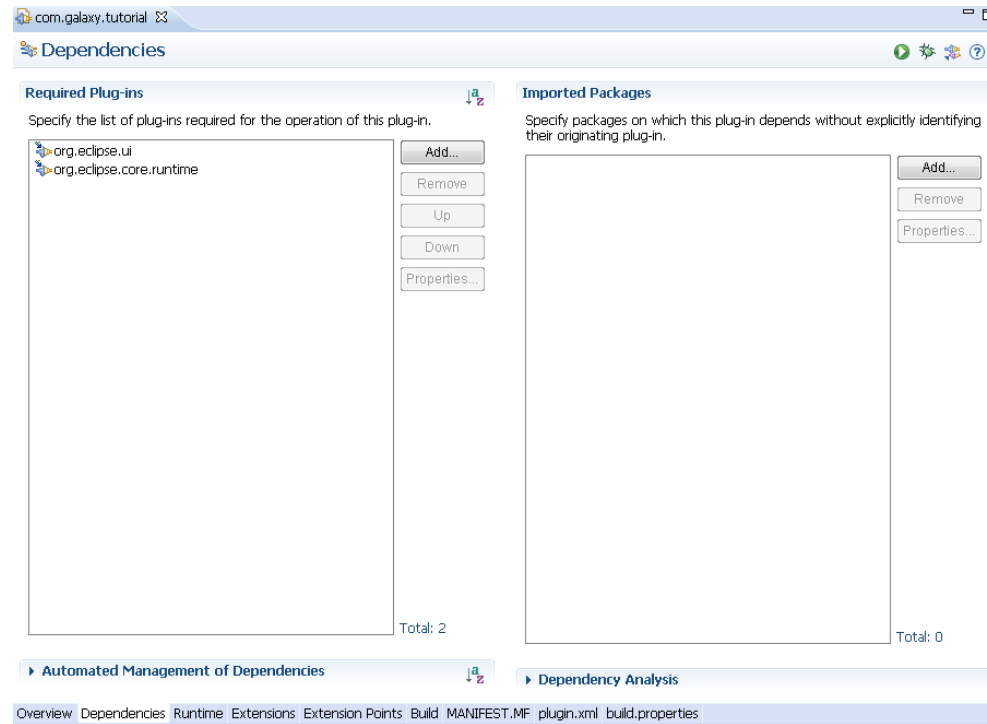
#### 1. Overview

插件的概述，其中包括插件的 ID、名称、版本等信息。



## 2. Dependencies

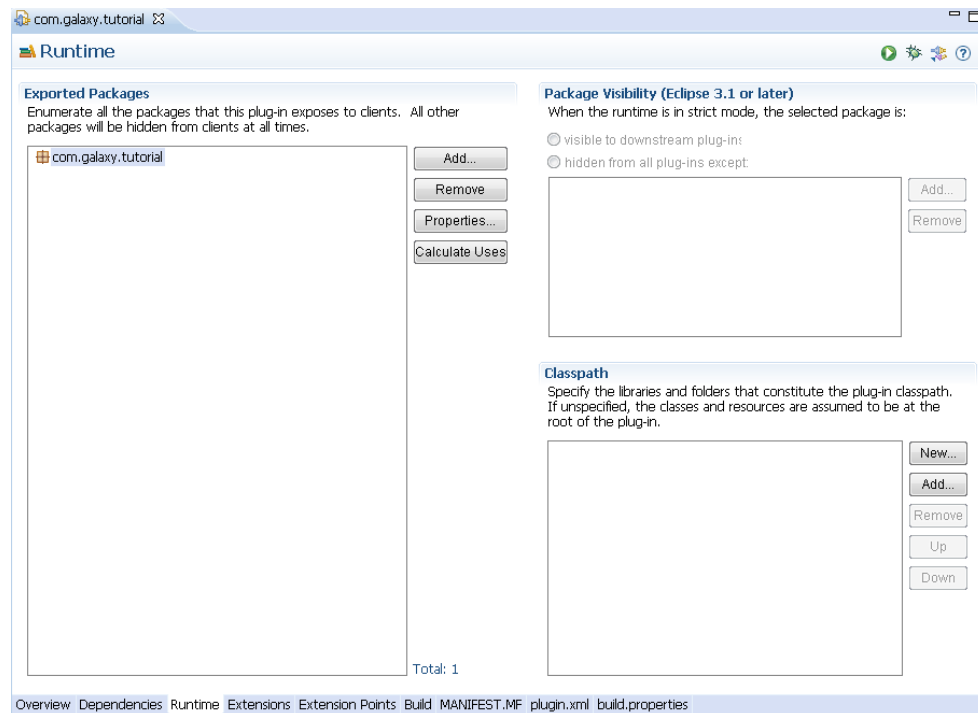
指定工程中所需要的插件依赖以及包依赖。例如工程中要用到 `org.eclipse.uml2.uml` 包，在左侧点击 Add，输入 `org.eclipse.uml2.uml`，点击 OK，引入 UML2 所需要的 Jar 包。



## 3. Runtime

Runtime 指定插件的运行时。

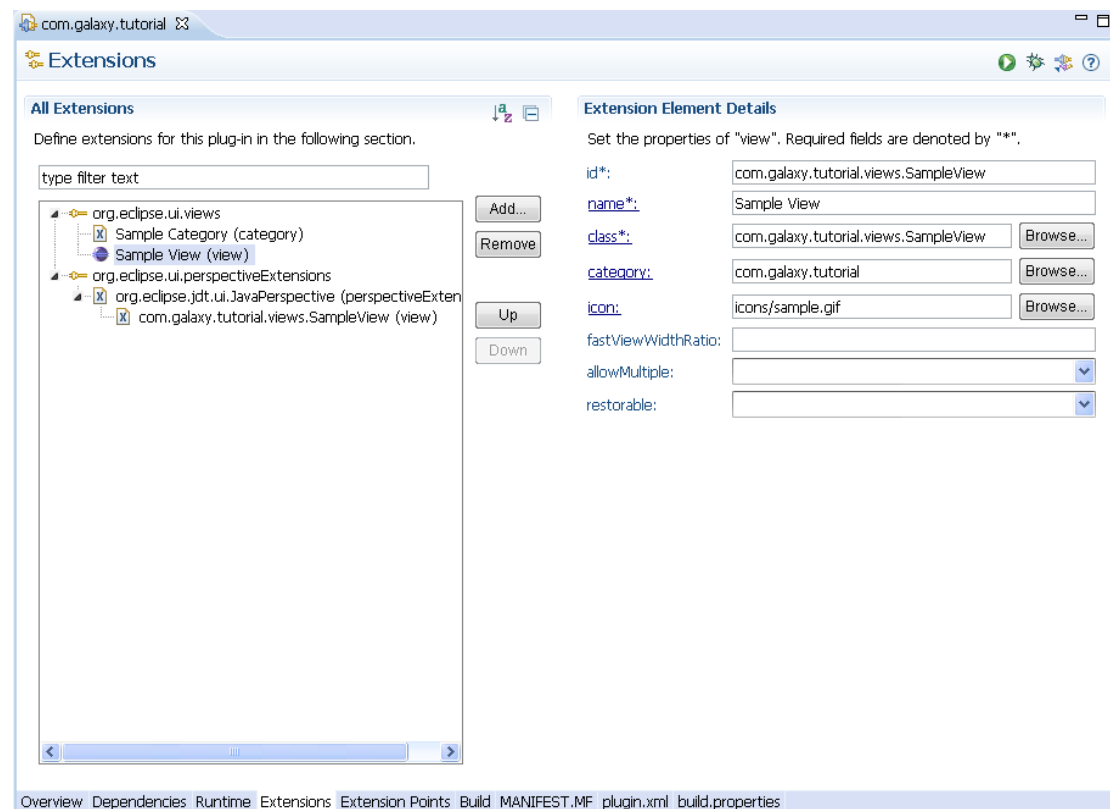
**Exported Package:** 导出被其他工程引用的包。如果工程 ProjectA 引用了该工程中的类，需要先在 ProjectA 的 Dependencies 中引入该工程的插件，并且在 Runtime 下 Exported Package 导出所引用的包。



## 4. Extensions

插件扩展。

- `org.eclipse.ui.views`: 视图的扩展点。
- `org.eclipse.ui.perspectiveExtensions`: 透视图的扩展点。

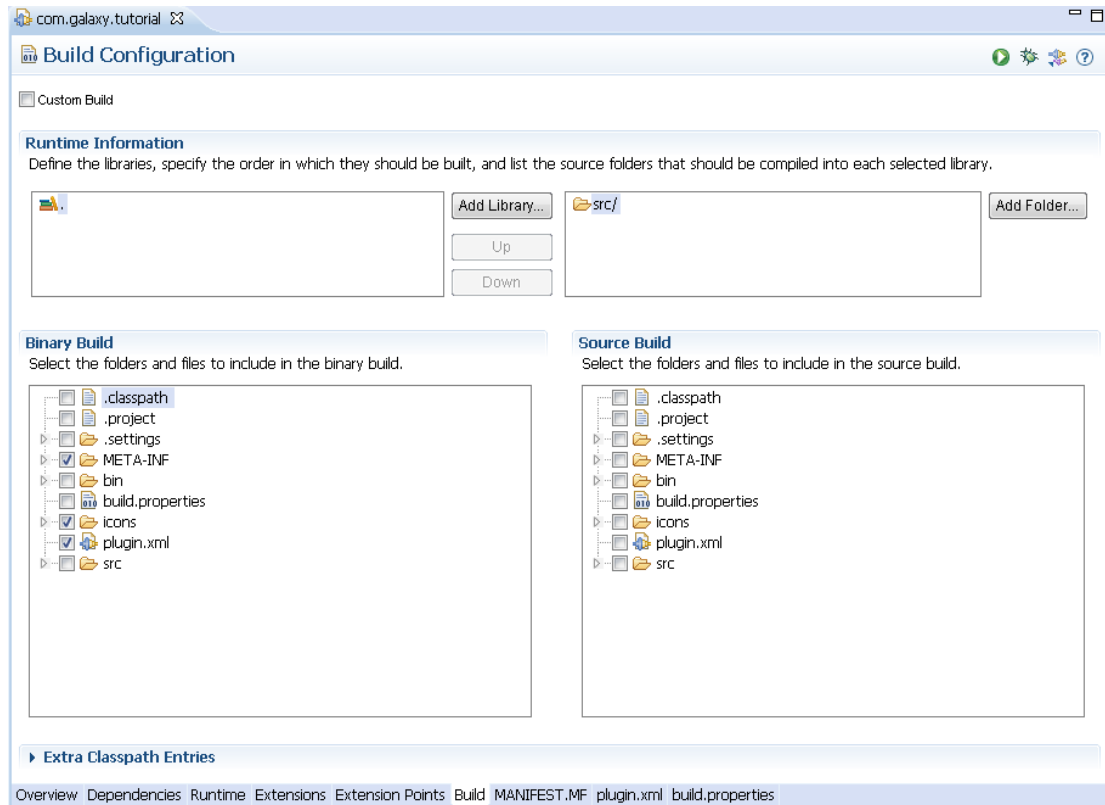


## 5. Extension Points

为其他项目提供插件开发的插入点。

## 6. Build

要构建的项目。如果出现第二章的错误[8]，尝试在 Runtime Information 下点击 Add Library，输入“.”，点击 Add Folder，选择 src 文件夹。



## 7. MANIFEST.MF

绑定的项目或者插件。

## 8. Plugin.xml

以 XML 文件的方式记录配置信息。

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
  <extension
    point="org.eclipse.ui.views">
    <category
```

```

        id="com.galaxy.tutorial"
        name="Sample Category">
    </category>
    <view
        name="Sample View"
        icon="icons/sample.gif"
        category="com.galaxy.tutorial"
        class="com.galaxy.tutorial.views.SampleView"
        id="com.galaxy.tutorial.views.SampleView">
    </view>
</extension>
<extension
    point="org.eclipse.ui.perspectiveExtensions">
    <perspectiveExtension
        targetID="org.eclipse.jdt.ui.JavaPerspective">
        <view
            ratio="0.5"
            relative="org.eclipse.ui.views.TaskList"
            relationship="right"
            id="com.galaxy.tutorial.views.SampleView">
        </view>
    </perspectiveExtension>
</extension>
</plugin>

```

## 9. Buildproperties

导出插件的配置信息。例如设置导出的编码格式 UTF-8，在文件最后追加：  
 javacDefaultEncoding.. = UTF-8 即可。

```

output.. = bin/
bin.includes = plugin.xml,\
    META-INF/, \
    icons/
jars.compile.order = .
source.. = src/
javacDefaultEncoding.. = UTF-8

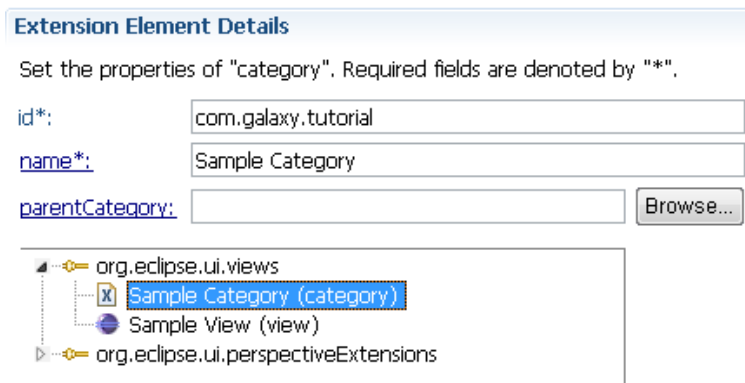
```



## 四、 视图

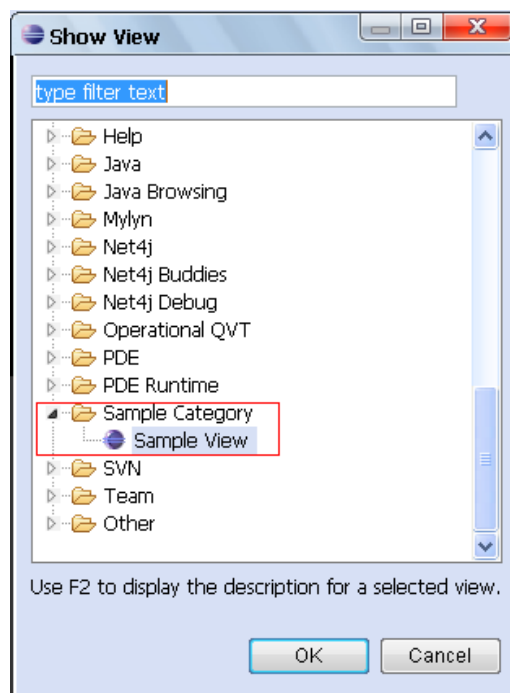
### 1. 概述

打开 plugin.xml 文件，切换到 Extension 选项页。左侧的列表中会有 org.eclipse.ui.views 的扩展点。如果没有，则点击 Add，输入 org.eclipse.ui.views，点击 Finish 添加。如下图所示，在 org.eclipse.ui.views 会有 Sample Category 和 Sample View 两个节点。右键 org.eclipse.ui.views 扩展点点击 New 有 3 个选项：Category、View、Stickyview。



### 2. 打开视图

在运行插件后，点击 Windows→Show View→Other，如下图所示：Sample View 对应视图类别为 Sample Category。



### 3. Category

Category 是视图的类别，默认为 Other 视图类别。

- Id: 视图类别的唯一标识。如需指定某个视图的类别，即可通过该 Id 关联。
- Name: 视图类别的名称。
- ParentCategory: 视图的父类别（暂时不知道怎么用，如果该项指定已有的 Category，Sample Category 会覆盖原有的 Category。）

### 4. View

视图的各项属性：

- Id: 视图的唯一标识。
- Name: 视图名称
- Class: 视图对应的类
- Category: 视图所在的视图类别。如果没有指定，默认该视图在 Other 视图类别目录下。
- Icon: 图标
- fastViewWidthRatio: 视图宽度的比例大小（0.05-0.95）
- allowMultiple: 是否允许多个视图的实例存在。默认为 false。
- restorable: 是否保存视图的状态，即再次打开 Eclipse 时，使用最后一次关闭 Eclipse 前该视图的状态。默认为 true，即自动记录该视图的状态。

#### Extension Element Details

Set the properties of "view". Required fields are denoted by "\*\*".

id*:	<input type="text" value="com.galaxy.tutorial.views.SampleView"/>
name*:	<input type="text" value="Sample View"/>
class*:	<input type="text" value="com.galaxy.tutorial.views.SampleView"/> Browse...
category:	<input type="text" value="com.galaxy.tutorial"/> Browse...
icon:	<input type="text" value="icons/sample.gif"/> Browse...
fastViewWidthRatio:	<input type="text" value=""/>
allowMultiple:	<input type="checkbox"/>
restorable:	<input checked="" type="checkbox"/>

### 5. Stickyview

Stickyview 是在无论打开任何透视图，该视图都会打开的视图。最好不要轻易使用该视图，一般用得也不多。

- Id: 视图的 ID。
- Location: 视图所在的方位，默认为 right。
- Closeable: 是否可关闭，默认为 true。（貌似没什么作用）
- Moveable: 是否可移动，默认为 true。（貌似没什么作用）

Extension Element Details

Set the properties of "stickyView". Required fields are denoted by "\*\*".

id\*:

org.eclipse.ui.console.ConsoleView

Browse...

location:

RIGHT

▼

closeable:

▼

moveable:

▼

## 6. 视图对应的类

在以上过程中设置对应的 class 为 `com.galaxy.tutorial.views.SampleView`。`SampleView` 必须继承于 `ViewPart`（实现视图的基础类）。在类中负责创建视图界面的方法为 `public void createPartControl(Composite parent){.....}`。如果需要自定义界面，只需要重写该方法即可。

## 7. 新建视图

右键 `org.eclipse.ui.views`→`New`→`View`，创建一个新视图。各项属性如下图所示：

Extension Element Details

Set the properties of "view". Required fields are denoted by "\*\*".

id\*:

com.galaxy.tutorial.views.SampleView

name\*:

Sample View

class\*:

com.galaxy.tutorial.views.SampleView

Browse...

category:

com.galaxy.tutorial

Browse...

icon:

icons/sample.gif

Browse...

fastViewWidthRatio:

allowMultiple:

▼

restorable:

▼

## 8. 创建视图界面

其中 `com.galaxy.tutorial.views.SampleViewII` 对应的内容，修改以下代码：

```
public class SampleViewII extends ViewPart {
    private Group group;
    private Button testButton; // 测试按钮

    public SampleViewII() {
        // TODO Auto-generated constructor stub
    }
}
```

```

@Override
public void createPartControl(Composite parent) {
    // TODO Auto-generated method stub
    group = new Group(parent, SWT.BORDER);
    group.setBounds(10, 10, 208, 322);
    group.setText("测试视图");

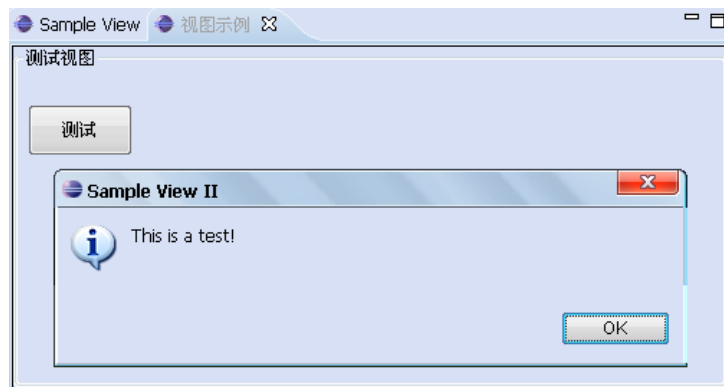
    testButton = new Button(group, SWT.PUSH);
    testButton.setText("测试");
    testButton.setBounds(10, 40, 73, 36);
    testButton.addSelectionListener(new SelectionAdapter() {
        public void widgetSelected(SelectionEvent e) {
            MessageDialog.openInformation(null, "Sample View II",
                "This is a test!");
        }
    });
}

@Override
public void setFocus() {
    // TODO Auto-generated method stub
    testButton.setFocus();
}
}

```

## 9. 运行结果

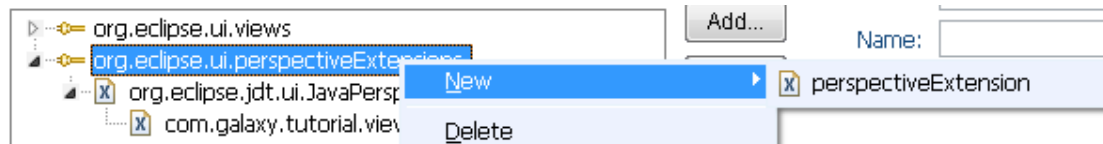
由于 Sample View II 尚未加入 Java 透视图，打开视图 Sample View II，点击测试按钮，得到以下结果：



## 五、 透视图

### 1. 概述

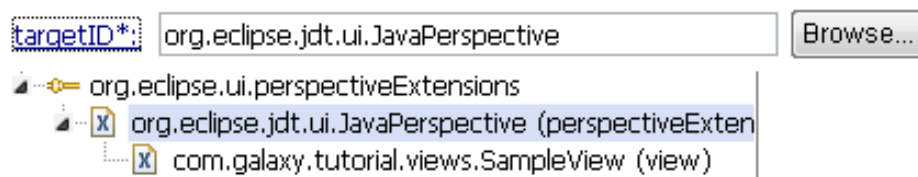
透视图是视图和操作的集合，将用户常用的视图拼接在一起。打开 `plugin.xml` 文件，切换到 Extension 选项页。左侧的列表中会有 `org.eclipse.ui.perspectiveExtension` 的扩展点。如果没有，则点击 Add，输入 `org.eclipse.ui.perspectiveExtension`，点击 Finish 添加。



### 2. 扩展已有透视图

#### 1) 透视图扩展点

右键 `org.eclipse.ui.perspectiveExtension`，新建一个 `perspectiveExtension`，`targetID` 填写对应的目标透视图 ID，这里为 `org.eclipse.jdt.ui.JavaPerspective`，即 Java 透视图的 ID。



#### 2) 添加视图

右键 `org.eclipse.jdt.ui.JavaPerspective`，新建一个 view。

- Id: 视图的 ID。
- Relationship: 与 relative 指定视图的相对位置。
- Relative: 相对视图。
- Ratio: 视图的比例 (0.05-0.95)。
- Visible: 初始化透视图时，该视图是否可见。
- Moveable、closeable 与之前一样。
- Standalone: 是否可单独显示。
- showTitle: 是否显示视图标题。
- minimized: 初始化透视图时，是否最小化该视图，默认 false。

### Extension Element Details

Set the properties of "view". Required fields are denoted by "\*\*".

<b>id*:</b>	<input type="text" value="com.galaxy.tutorial.views.SampleView"/>	<input type="button" value="Browse..."/>
<b>relationship*:</b>	<input type="text" value="right"/>	<input type="button" value="v"/>
<b>relative:</b>	<input type="text" value="org.eclipse.ui.views.TaskList"/>	<input type="button" value="Browse..."/>
<b>ratio:</b>	<input type="text" value="0.5"/>	
<b>visible:</b>	<input type="text" value="true"/>	<input type="button" value="v"/>
<b>closeable:</b>	<input type="text"/>	<input type="button" value="v"/>
<b>moveable:</b>	<input type="text"/>	<input type="button" value="v"/>
<b>standalone:</b>	<input type="text"/>	<input type="button" value="v"/>
<b>showTitle:</b>	<input type="text"/>	<input type="button" value="v"/>
<b>minimized:</b>	<input type="text"/>	<input type="button" value="v"/>

## 3. 打开透视图操作

在 Sample View II 视图中，设置按钮打开 Java 透视图。

### 1) 新建 Action。

创建 com.galaxy.tutorial.action.OpenPerspectiveAction，内容如下：

```
public class OpenPerspectiveAction extends SelectionAdapter {
    /**
     * Document Start
     * 打开Java透视图
     * Document End
     * Author: 黄露
     * Time: 2009-7-8 上午11:32:14
     * @see
     org.eclipse.swt.events.SelectionAdapter#widgetSelected(org.eclips
     e.swt.events.SelectionEvent)
     */
    public void widgetSelected(SelectionEvent e) {
        System.out.println("try to open perspective.");
        try {
            PlatformUI.getWorkbench().showPerspective(
                "org.eclipse.jdt.ui.JavaPerspective", //Java透视图
                的ID
            );
            PlatformUI.getWorkbench().getActiveWorkbenchWindow();
        } catch (WorkbenchException we) {
```

```

        ProgressDialog.openInformation(null, "Sample View II",
            "Open Java perspective failed.");
        we.printStackTrace();
    }
}
}

```

## 2) 修改按钮绑定的 Action。

在 com.galaxy.tutorial.views.SampleViewII 中，修改方法如下：

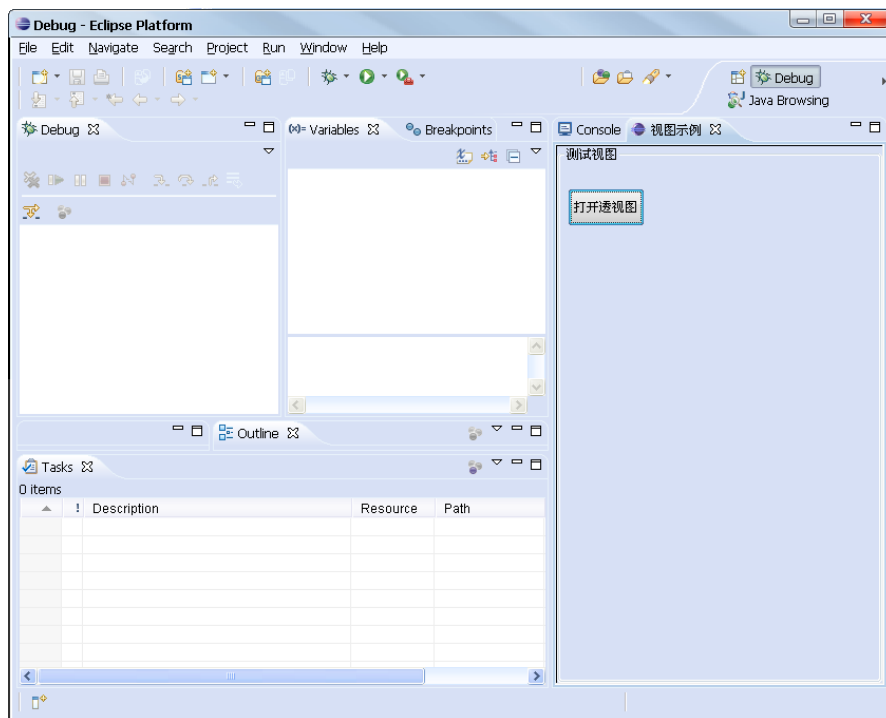
```

public void createPartControl(Composite parent) {
    // TODO Auto-generated method stub
    group = new Group(parent, SWT.BORDER);
    group.setBounds(10, 10, 208, 322);
    group.setText("测试视图");
    testButton = new Button(group, SWT.PUSH);
    testButton.setText("打开透视图");
    testButton.setBounds(10, 40, 73, 36);
    testButton.addSelectionListener(new OpenPerspectiveAction());
}

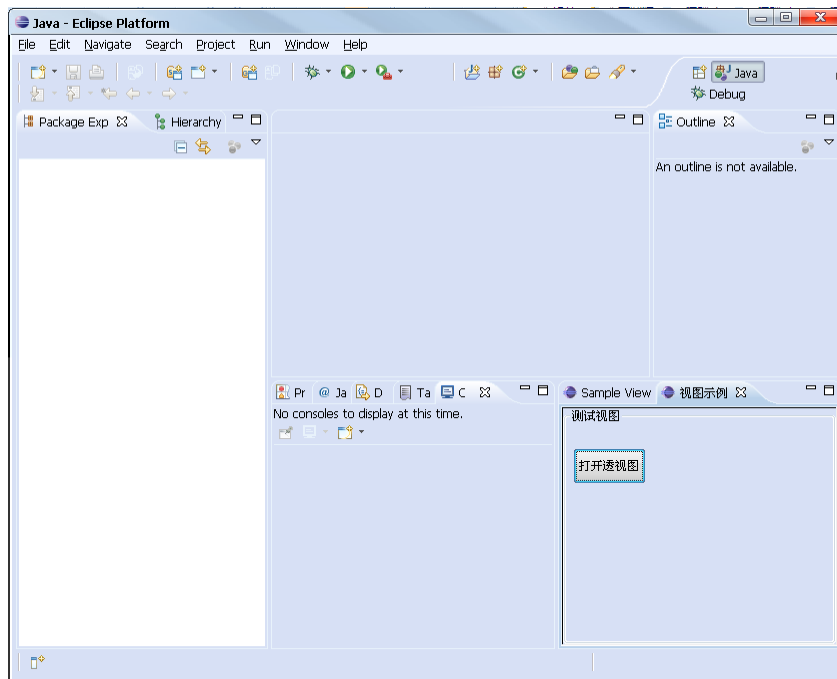
```

## 3) 运行结果。

先将透视图切换到 Debug 透视图，打开 Sample View II 视图，点击“打开透视图”按钮。



打开之前



打开之后

## 4. 新建透视图

### 1) 添加透视图扩展点

打开 plugin.xml 文件, 切换到 Extension 选项页。点击 Add, 输入 org.eclipse.ui.perspectives, 点击 Finish 完成。

### 2) 新建透视图

右键 org.eclipse.ui.perspectives, New → perspective。填写以下内容:

- Id: 透视图的唯一标识。
- Name: 透视图名称
- Class: 创建透视图的类。
- Icon: 图标。
- Fixed: 是否固定, 默认 false。



### Extension Element Details

Set the properties of "perspective". Required fields are denoted by "\*".

id\*:

name\*:

class\*:

icon:

fixed:

org.eclipse.ui.perspectives  
透视图实例 (perspective)

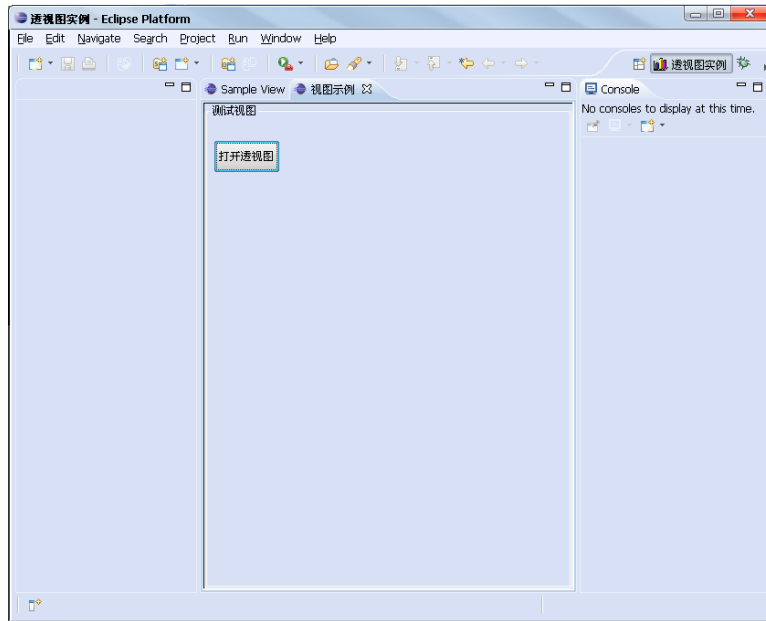
### 3) 类的实现。

透视图对应的类 `com.galaxy.tutorial.perspective.SamplePerspectiveFactory` 实现如下：

```
public class SamplePerspectiveFactory implements IPerspectiveFactory {
    /**
     * Document Start
     * 设置透视图的布局
     * Document End
     * Author: 黄露
     * Time: 2009-7-9 下午05:31:26
     * @see
     org.eclipse.ui.IPerspectiveFactory#createInitialLayout(org.eclipse.ui
     .IPageLayout)
     */
    @Override
    public void createInitialLayout(IPageLayout layout) {
        // 添加视图
        layout.addView("com.galaxy.tutorial.SampleViewII", // 要添加的视
图ID
            IPageLayout.RIGHT, // 方位
            0.5f, // 视图比例
            "com.galaxy.tutorial.views.SampleView"); // 依附的视图ID
        // 添加视图的快捷方式
        layout.addShowViewShortcut("com.galaxy.tutorial.SampleViewII");
    }
}
```

### 4) 运行结果。


视图示例已经打开，但是 `Sample View` 可能不会打开。



## 六、 编辑器

### 1. 概述

通常情况下，编辑器被认为仅仅是一个用来输入文本的窗口。然而，编辑器与普通窗口的重要区别在于编辑器所能完成的任务，以及怎样把编辑器的内容显示给用户。简单的说，编辑器通常用来修改某种类型的逻辑输入，并且这个任务可以持续很长时间。

学习编辑器的插件开发，可以新建一个插件工程，选择  **Plug-in with an editor** 模板，查看 Eclipse 自带的例子。

### 2. 扩展编辑器

#### 1) 声明编辑器扩展点

打开 `plug-in.xml` 文件，切换到 **Extensions** 选项页。点击 **Add**，输入 `org.eclipse.ui.editors`，点击 **Finish**，添加编辑器的扩展点。这样会自动新建一个编辑器，填写以下内容：

- **Id**: 编辑器的唯一标识。
- **Name**: 编辑器名称
- **Icon**: 图标。
- **Extensions**: 文件后缀名。
- **Class**: 编辑器对应的类。
- **Command**: 启动外部编辑器的命令。可执行命令必须位于系统路径或者插件的目录中。该属性的类、命令和启动程序是互斥的。

- **Launcher:** 实现 `org.eclipse.ui.IEditorLauncher` 并打卡外部编辑器的类的名字。
- **ContributorClass:** 实现 `org.eclipse.ui.IEditorActionBarContributor` 和在工作台菜单与工具栏添加新操作的类的完全限定名反映了编辑器类型的特征。
- **Default:** 是否作为该类型的默认编辑器。
- **FileNames:** 有逗号分隔的文件名组成的字符串，指示编辑器所理解的文件名。例如：理解插件和片断清单文件的编辑器可以注册 `plugin.xml`、`fragment.xml`。
- **SymbolicFontName:** 编辑器的字体。
- **MatchingStrategy:** 自定义匹配文件后缀名的策略。

**Extension Element Details**

Set the properties of "editor". Required fields are denoted by "\*".

<code>id*:</code>	<input type="text" value="com.galaxy.tutorial.sampleEditor"/>
<code>name*:</code>	<input type="text" value="Sample Editor"/>
<code>icon:</code>	<input type="text" value="icons/editor.ico"/> <input type="button" value="Browse..."/>
<code>extensions:</code>	<input type="text" value="test"/>
<code>class:</code>	<input type="text" value="com.galaxy.tutorial.editor.SampleEditor"/> <input type="button" value="Browse..."/>
<code>command:</code>	<input type="text"/>
<code>launcher:</code>	<input type="text"/> <input type="button" value="Browse..."/>
<code>contributorClass:</code>	<input type="text"/> <input type="button" value="Browse..."/>
<code>default:</code>	<input type="text" value="false"/> <input type="button" value="v"/>
<code>filenames:</code>	<input type="text"/>
<code>symbolicFontName:</code>	<input type="text"/>
<code>matchingStrategy:</code>	<input type="text"/> <input type="button" value="Browse..."/>

org.eclipse.ui.editors  
Sample Editor (editor)

## 2) 实现 `com.galaxy.tutorial.editor.SampleEditor`。

编辑器的实现类既可以实现 `IEditorPart` 接口，也可以继承于 `EditorPart`。在 `EditorPart` 中基本的方法：

- `createPartControl (Composite)`：创建组成编辑器的控件。
- `doSave (IProgressMonitor)`：保存该编辑器的内容。
- `doSaveAs ()`：调用它打开“另存为”对话框。
- `gotoMarker (IMarker)`：为给定标记指定的编辑器设置光标和选择状态。
- `init (IEditorSite, IEditorInput)`：使用给定的编辑器站点和输入初始化该编辑器。
- `isDirty ()`：返回自从上次保存操作之后编辑器的内容是否更改。

`com.galaxy.tutorial.editor.SampleEditor` 的实现如下：（由于是个例子，所以非常简单，可能会存在一些错误。其中很多的功能还需要自己添加，这里不作详细叙述。）

```
/**
 * Document Start
 * 实现编辑器
 * Document End
 * Author: 黄露
 * Time: 2009-7-10 上午09:04:05
 */
```

```

public class SampleEditor extends EditorPart {
    /**
     * Document Start
     * 保存
     * Document End
     * Author: 黄露
     * Time: 2009-7-10 上午09:04:06
     * @see
     org.eclipse.ui.part.EditorPart#doSave(org.eclipse.core.runtime.IProgr
     essMonitor)
     */
    @Override
    public void doSave(IProgressMonitor monitor) {
        // TODO Auto-generated method stub
        System.out.println("Save");
    }

    /**
     * Document Start
     * 另存为
     * Document End
     * Author: 黄露
     * Time: 2009-7-10 上午09:04:06
     * @see org.eclipse.ui.part.EditorPart#doSaveAs()
     */
    @Override
    public void doSaveAs() {
        // TODO Auto-generated method stub
    }

    /**
     * Document Start
     * 初始化
     * Document End
     * Author: 黄露
     * Time: 2009-7-10 上午09:04:06
     * @see
     org.eclipse.ui.part.EditorPart#init(org.eclipse.ui.IEditorSite,
     org.eclipse.ui.IEditorInput)
     */
    @Override
    public void init(IEditorSite site, IEditorInput input)
        throws PartInitException {

```

```

        setSite(site);
        setInput(input);
    }

    /**
     * Document Start
     *
     * Document End
     * Author: 黄露
     * Time: 2009-7-10 上午09:04:06
     * @see org.eclipse.ui.part.EditorPart#isDirty()
     */
    @Override
    public boolean isDirty() {
        return false;
    }

    /**
     * Document Start
     *
     * Document End
     * Author: 黄露
     * Time: 2009-7-10 上午09:04:06
     * @see org.eclipse.ui.part.EditorPart#isSaveAsAllowed()
     */
    @Override
    public boolean isSaveAsAllowed() {
        return false;
    }

    /**
     * Document Start
     * 创建编辑器的控件
     * Document End
     * Author: 黄露
     * Time: 2009-7-10 上午09:04:06
     * @see
     org.eclipse.ui.part.WorkbenchPart#createPartControl(org.eclipse.swt.w
idgets.Composite)
     */
    @Override
    public void createPartControl(Composite parent) {
        Text text = new Text(parent, SWT.BORDER);
        text.setText("Sample Editor!");
    }

```

```

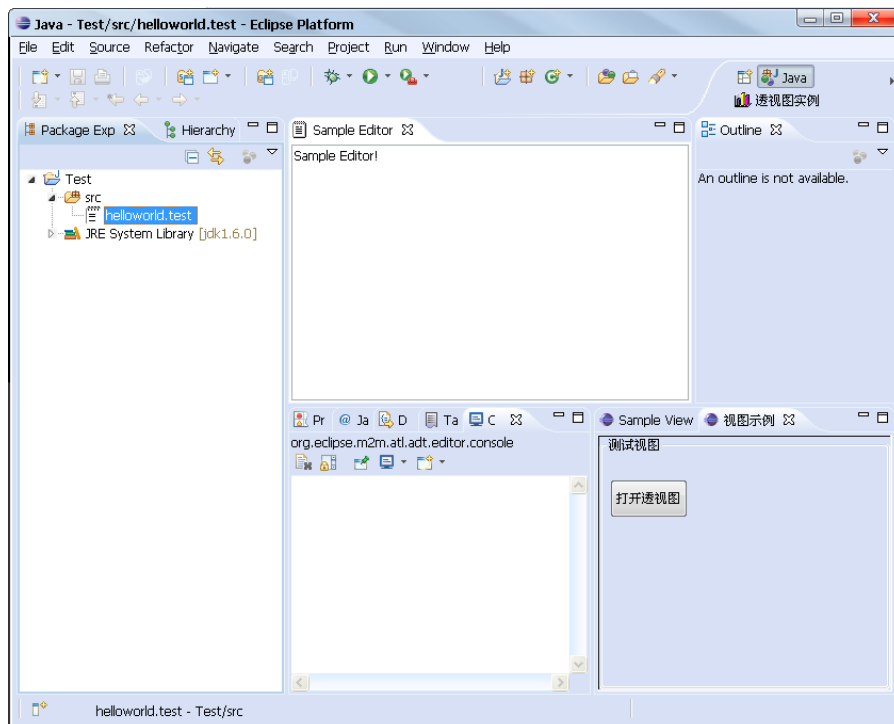
    }

    /**
     * Document Start
     * 设置光标
     * Document End
     * Author: 黄露
     * Time: 2009-7-10 上午09:04:06
     * @see org.eclipse.ui.part.WorkbenchPart#setFocus()
     */
    @Override
    public void setFocus() {
        // TODO Auto-generated method stub
    }
}

```

### 3) 运行结果

运行插件，新建一个工程，在 src 文件夹下新建一个 helloworld.test，点击 Finish 后出现下面的结果：



### 3. 扩展多页编辑器

多页编辑器的扩展与编辑器类似，只是类的实现不一样。

## 1) 声明编辑器扩展点

右键 `org.eclipse.ui.editors`，`New→editor` 新建一个编辑器。填写以下内容：

**Extension Element Details**

Set the properties of "editor". Required fields are denoted by "\*\*".

id*:	<code>com.galaxy.tutorial.sampleMultiPageEditor</code>
name*:	<code>Sample MultiPageEditor</code>
icon:	<code>icons/multipageeditor.ICO</code> <span>Browse...</span>
extensions:	<code>smpe</code>
class:	<code>com.galaxy.tutorial.editor.SampleMultiPage</code> <span>Browse...</span>
command:	
launcher:	
contributorClass:	
default:	<code>false</code> <span>▼</span>
filenames:	
symbolicFontName:	
matchingStrategy:	

org.eclipse.ui.editors

- Sample Editor (editor)
- Sample MultiPageEditor (editor)**

## 2) 引入依赖的插件

切换至 `Dependencies`，在 `Required Plug-ins` 中添加 `org.eclipse.ui.editors`, `org.eclipse.core.resources`, `org.eclipse.ui.ide`, `org.eclipse.jface.text` 四个插件，在分页编辑器的实现类要引入这些包的接口和类。

**Required Plug-ins**

Specify the list of plug-ins required for the operation of this plug-in.

- org.eclipse.ui
- org.eclipse.core.runtime
- org.eclipse.ui.editors (3.4.0)**
- org.eclipse.core.resources (3.4.1)**
- org.eclipse.ui.ide (3.4.1)**
- org.eclipse.jface.text (3.4.1)**

Overview Dependencies Runtime Extensions

Add... Remove Up Down Properties...

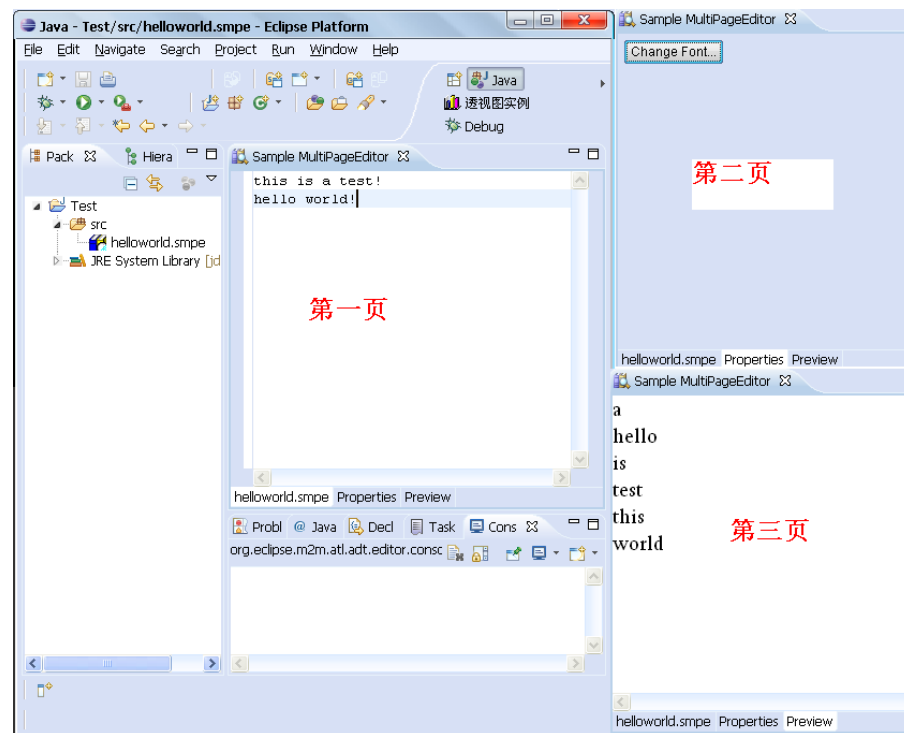
## 3) 实现 `com.galaxy.tutorial.editor.SampleMultiPageEditor`。

`SampleMultiPageEditor` 继承于 `MultiPageEditorPart` 类，实现分页编辑器的功能；实现 `IResourceChangeListener` 接口，实现分页之间资源同步的功能。

`SampleMultiPageEditor` 的具体实现如下：

## 4) 运行结果

新建一个 helloworld.smpe 文件，打开后输入图中的语句，出现以下的结果：



## 七、 向导

### 1. 概述

#### 1) 向导

向导的任务是创建和初始化它包含的页面、处理页面间所有特定的定制流和信息，并在单击“完成”按钮时执行操作。向导通常是实现了 `org.eclipse.jface.wizard.IWizard` 接口的 `org.eclipse.jface.wizard.Wizard` 的子类，其主要方法：

- `addPages ()`：添加适当的页。
- `getContainer ()`：返回向导容器，该向导将会在此容器中显示。
- `getNextPage (IWizardPage)`：下一个页面。
- `getPreviousPage (IWizardPage)`：上一个页面。
- `getStartingPage ()`：向导的第一个页面。
- `performCancel ()`：取消向导，点击 Cancel。
- `performFinish ()`：完成向导，点击 Finish。
- `setWindowTitle (String)`：设置窗口标题。



## 2) 向导页

向导页的任务是给用户展示页面信息、验证用户在该页面输入的所有信息，并为向导提供存取方式来收集输入的信息。向导页通常会创建 `org.eclipse.jface.wizard.WizardPage` 类的子类，而不是实现 `org.eclipse.jface.wizard.IWizardPage` 接口。`WizardPage` 主要的方法有：

- `createControl (Composite)`：创建组成向导页的控件。
- `getContainer ()`：返回容纳该向导页的向导容器。
- `getDialogSettings ()`：返回该向导页的对话框设置。
- `getWizard ()`：返回容纳该向导页的向导。
- `setDescription (String)`：设置显示在向导标题区的描述性文本。
- `setErrorMessage (String)`：为该页面设置或清空出错信息。
- `setMessage (String)`：为该页面设置或清空信息。
- `setPageComplete (boolean)`：设置该页是否完整。这是确定是否启用“下一步”和“完成”按钮的依据。

## 3) 向导容器

向导使用向导容器与显示向导的上下文进行通信。向导容器通常采用实现 `org.eclipse.jface.wizard.IWizardContainer` 接口的方式，其主要的方法有：

- `getCurrentPage ()`：返回正在显示的当前页面。
- `run (boolean, boolean, IRunnableWithProgress)`：运行向导对话框上下文中给定的可运行程序。
- `showPage (IWizardPage)`：显示指定的向导页面。
- `updateButtons ()`：调整“上一步”、“下一步”和“完成”按钮的启用状态，来反映该容器中活动页面的状态。
- `updateMessage ()`：更新显示在消息行的消息，来反映该容器中活动页面的状态。
- `updateTitleBar ()`：更新标题栏（标题、描述和图像），来反映该容器中活动页面的状态。
- `updateWindowTitle ()`：更新窗口标题，来反映该向导的状态。

# 2. 扩展向导

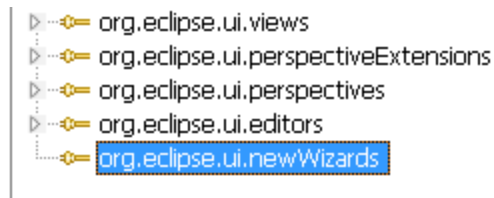
## 1) 声明向导扩展点

针对不同的向导，可以声明不同的向导扩展点：

- `org.eclipse.ui.exportWizards`——在“Export”向导中添加一个嵌套向导，通过选择“File→export”菜单可以显示该向导。与该扩展点关联的向导类别必须实现 `org.eclipse.ui.IExportWizard` 接口。
- `org.eclipse.ui.importWizards`——在“Import”向导中添加一个嵌套向导，通过选择“File→import”菜单可以显示该向导。与该扩展点关联的向导类别必须实现 `org.eclipse.ui.IImportWizard` 接口。

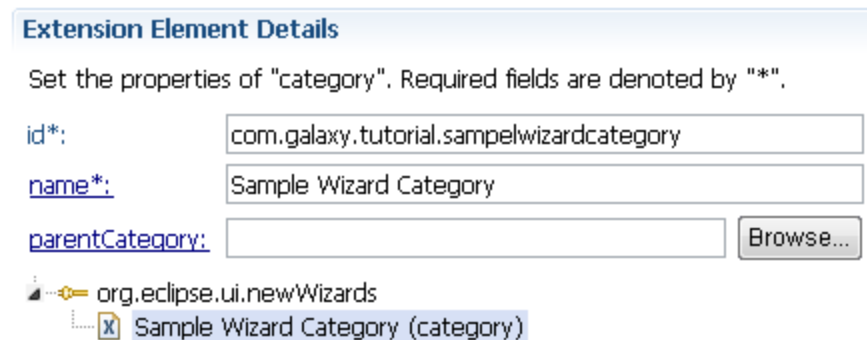
- `org.eclipse.ui.newWizards`——在“New”向导中添加一个嵌套向导，通过选择“File→New→Other”菜单可以显示该向导。与该扩展点关联的向导类别必须实现 `org.eclipse.ui.INewWizard` 接口。

打开 `plug-in.xml` 文件，切换到 Extensions 选项页。点击 Add，输入 `org.eclipse.ui.newWizards`，点击 Finish。



## 2) 新建向导类别

右键 `org.eclipse.ui.newWizards`，选择 `New→category`。与视图类型相似，不再赘述。输入 id 和 name，如下图所示：



## 3) 新建向导

右键 `org.eclipse.ui.newWizards`，选择 `New→wizard`。向导的各项属性说明如下：

- `id`: 向导的唯一标识。
- `name`: 向导的名称。
- `class`: 向导对应的实现类。
- `icon`: 向导的图标。
- `category`: 向导所在的类别。
- `project`: 是否在新建项目中显示该向导。
- `finalPerspective`: 当工程建立时，所打开的透视图的 ID。
- `preferredPerspectives`: 当工程建立时，作为第二选项打开的透视图的 ID。
- `helpHref`: 连接帮助文档的链接。
- `descriptionImage`: 描述向导的图片。
- `canFinishEarly`: 是否可以提前完成向导。
- `hasPages`: 向导是否提供页面显示。

### Extension Element Details

Set the properties of "wizard". Required fields are denoted by "\*\*".

id*:	com.galaxy.tutorial.sampleWizard	
name*:	Sample Wizard	
class*:	com.galaxy.tutorial.wizard.SampleNew\	Browse...
icon:	icons/wizard.gif	Browse...
category:	com.galaxy.tutorial.sampelwizardcategory	
project:		
finalPerspective:		Browse...
preferredPerspectives:		
helpHref:		
descriptionImage:		Browse...
canFinishEarly:		
hasPages:		

org.eclipse.ui.newWizards
 

- Sample Wizard Category (category)
- Sample Wizard (wizard)**

#### 4) 实现向导类

com.galaxy.tutorial.wizard.SampleNewWizard 继承于 Wizard 并且实现 INewWizard 接口，其具体实现如下：

```

package com.galaxy.tutorial.wizard;

import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.jface.wizard.Wizard;
import org.eclipse.ui.INewWizard;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.dialogs.WizardNewFileCreationPage;

/**
 * Document Start
 * 向导的实现类
 * Document End
 * Author: 黄露
 * Time: 2009-7-16 下午02:36:52
 */
public class SampleNewWizard extends Wizard implements INewWizard {
    private IStructuredSelection selection;
    private WizardNewFileCreationPage newFileWizardPage;

```

```

/**
 * Document Start
 * 构造函数
 * Document End
 * Author: 黄露
 * Time: 2009-7-16 下午02:36:53
 */
public SampleNewWizard() {
    System.out.println("New a sample wizard!");
}

/**
 * Document Start
 * 添加向导页
 * Document End
 * Author: 黄露
 * Time: 2009-7-16 下午07:30:43
 * @see org.eclipse.jface.wizard.Wizard#addPages()
 */
@Override
public void addPages() {
    // 向导标题栏文字
    setWindowTitle("New a File");

    // 添加向导页
    newFileWizardPage = new WizardNewFileCreationPage("Page name",
        selection);
    addPage(newFileWizardPage);
}

/**
 * Document Start
 * 点击Finish
 * Document End
 * Author: 黄露
 * Time: 2009-7-16 下午02:36:53
 * @see org.eclipse.jface.wizard.Wizard#performFinish()
 */
@Override
public boolean performFinish() {
    System.out.println("Finish");
    return true;
}

```

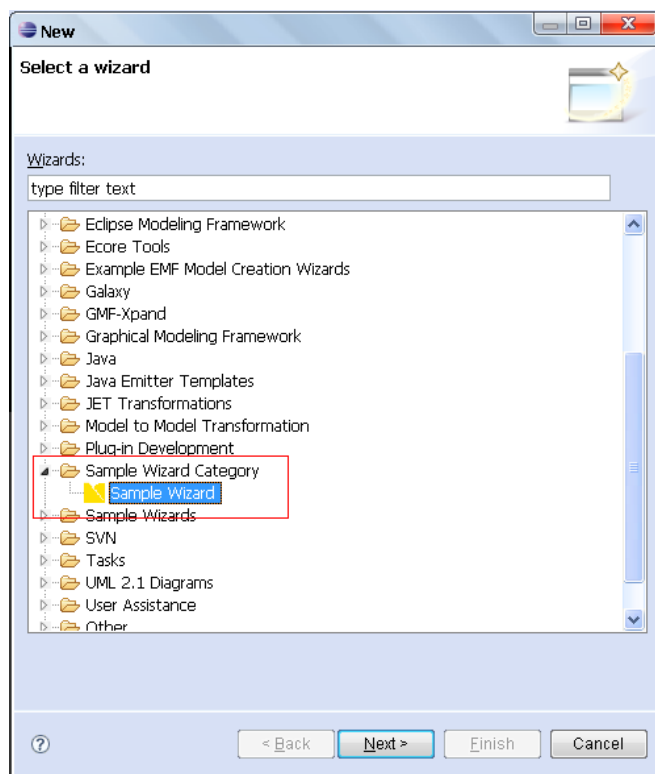
```

/**
 * Document Start
 * 初始化
 * Document End
 * Author: 黄露
 * Time: 2009-7-16 下午02:36:53
 * @see
org.eclipse.ui.IWorkbenchWizard#init(org.eclipse.ui.IWorkbench,
org.eclipse.jface.viewers.IStructuredSelection)
 */
@Override
public void init(IWorkbench workbench, IStructuredSelection
selection) {
    this.selection = selection;
}
}

```

## 5) 运行结果

运行插件，File→New→Other，结果如下：



点击 Next，出现新建文件向导页：



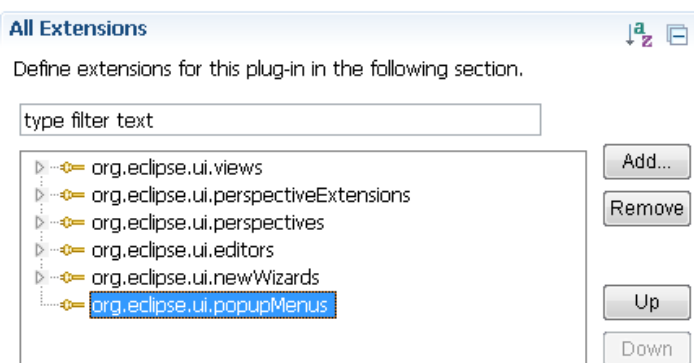
当然，也可以自己实现向导页，只需要继承 `org.eclipse.jface.wizard.WizardPage` 类，覆盖主要的方法即可。

## 八、 上下文操作

在视图或者编辑器的操作中，我们希望选择与这个操作兼容的对象时，操作才出现在上下文的菜单中。因此，用户需要该操作时，就可以为对象添加功能了。

### 1. 声明上下文菜单扩展点

打开 `plug-in.xml` 文件，切换到 Extensions 选项页。点击 Add，输入 `org.eclipse.ui.popupMenus`，点击 Finish。

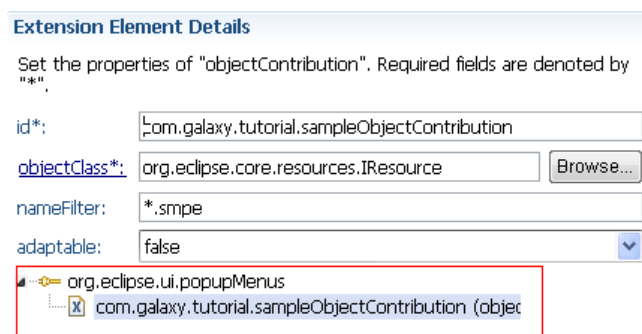


## 2. 对象上下文操作

### 1) 新建对象操作

右键 `org.eclipse.ui.popupMenus`，`new`→`objectContribution`，新建一个对象操作。对象操作的具体属性如下：

- `id`：对象操作的唯一标识。
- `objectClass`：对象操作要指定的对象类型。
- `nameFilter`：对象的后缀名。
- `adaptable`：指出适配 `IResource` 的对象是可接受的目标。



### 2) 新建操作

右键 `com.galaxy.tutorial.sampleObjectContribution`，`new`→`action`，新建一个操作。其属性说明如下：

- `id`：操作的唯一标识。
- `label`：操作的显示名称。
- `class`：操作对应的实现类。
- `definitionId`：
- `menubarPath`：操作所在目录位置。`additions` 表示在上下文菜单最外层，没有任何目录包含。
- `icon`：图标。
- `helpContextId`：帮助文档的 `Id`。
- `style`：定义操作的可视形式的属性。其中 `pullDown` 不适用于对象添加。
- `state`：
- `enableFor`：指出操作何时启用的表达式。
  - !——选中 0 个项。
  - ?——选中 0 个或 1 个项。
  - +——选中 1 个或者更多项。
  - multiple, 2+——选中 2 个或者更多项。
  - \*——选中任何数量的项。
- `overrideActionId`：指定操作覆盖的操作的标识。
- `tooltip`：操作的提示信息。

**Extension Element Details**

Set the properties of "action". Required fields are denoted by "\*".

id*:	<input type="text" value="com.galaxy.tutorial.objectActionTest"/>
label*:	<input type="text" value="Sample Object Operation"/>
class*:	<input type="text" value="com.galaxy.tutorial.action.SampleObjectActi"/> <input type="button" value="Browse..."/>
definitionId:	<input type="text"/> <input type="button" value="Browse..."/>
menubarPath:	<input type="text" value="additions"/>
icon:	<input type="text" value="icons/ico.ico"/> <input type="button" value="Browse..."/>
helpContextId:	<input type="text"/>
style:	<input type="text"/> <input type="button" value="v"/>
state:	<input type="text"/> <input type="button" value="v"/>
enablesFor:	<input type="text" value="1"/>
overrideActionId:	<input type="text"/> <input type="button" value="Browse..."/>
tooltip:	<input type="text" value="This is a Sample Object Operation!"/>

- org.eclipse.ui.popupMenus
  - com.galaxy.tutorial.sampleObjectContribution (object)
    - Sample Object Operation (action)

### 3) 实现操作的类

对象操作对应的类 `com.galaxy.tutorial.action. SampleObjectAction`，实现如下：

```
package com.galaxy.tutorial.action;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IObjectActionDelegate;
import org.eclipse.ui.IWorkbenchPart;

/**
 * Document Start
 * 对象操作的实现类
 * Document End
 * Author: 黄露
 * Time: 2009-7-16 下午08:06:27
 */
public class SampleObjectAction implements IObjectActionDelegate {

    /**
     * Document Start
     * 构造函数
     * Document End
     * Author: 黄露
     * Time: 2009-7-16 下午08:06:27
     */
}
```



```

    public SampleObjectAction() {
        // TODO Auto-generated constructor stub
    }

    /**
     * Document Start
     * 设置活动部件
     * Document End
     * Author: 黄露
     * Time: 2009-7-16 下午08:06:27
     * @see
     org.eclipse.ui.IObjectActionDelegate#setActivePart(org.eclipse.jface.
     action.IAction, org.eclipse.ui.IWorkbenchPart)
     */
    @Override
    public void setActivePart(IAction action, IWorkbenchPart targetPart)
    {
        // TODO Auto-generated method stub
    }

    /**
     * Document Start
     * 对象操作所执行的事件
     * Document End
     * Author: 黄露
     * Time: 2009-7-16 下午08:06:27
     * @see
     org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
     */
    @Override
    public void run(IAction action) {
        MessageDialog.openInformation(null, "提示信息",
            "This is a object operation!");
    }

    /**
     * Document Start
     * 选中目标
     * Document End
     * Author: 黄露
     * Time: 2009-7-16 下午08:06:27
     * @see
     org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.act

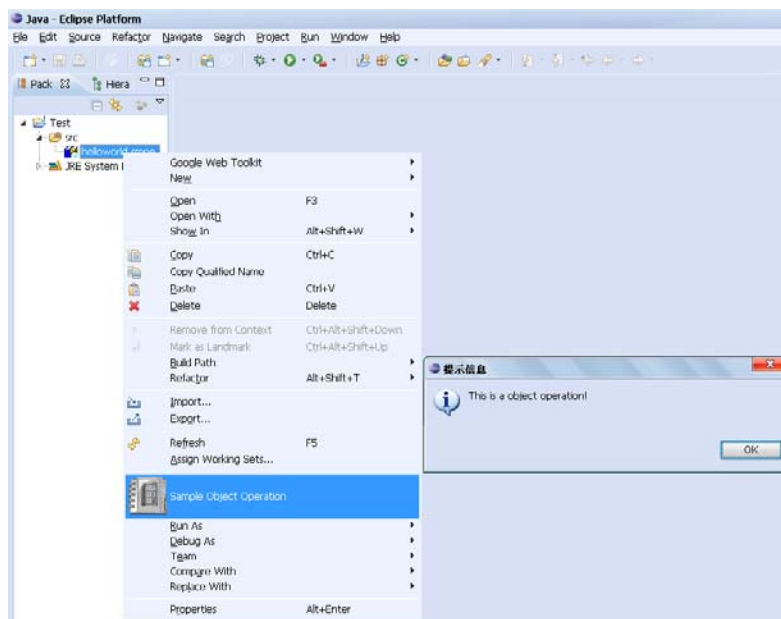
```

```
ion.IAction, org.eclipse.jface.viewers.ISelection)

    */
    @Override
    public void selectionChanged(IAction action, ISelection selection) {
        // TODO Auto-generated method stub
    }
}
```

#### 4) 运行结果

运行插件后，选中后缀名为.smpe 的文件，右键点击 Sample Object Operation，出现以下结果：



### 3. 视图上下文操作

#### 1) 新建视图操作

右键 `org.eclipse.ui.popupMenus`，`new`→`viewerContribution`，新建一个视图操作。视图操作的具体属性如下：

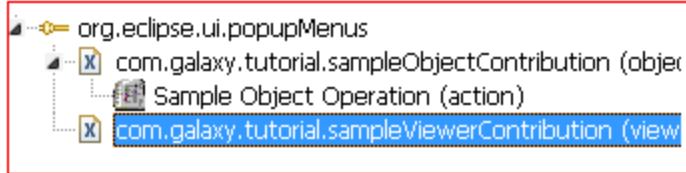
- **id**：视图操作的唯一标识。
- **targetID**：所要定义操作的目标视图。（一定要保证目标视图的插件被工程引入了！）

注：如何获取一个视图的ID以及一个对象的类型？在Eclipse3.4 版本之后，添加了 **Alt+Shift+F1** 的快捷键，查询选中对象的基本信息，其中包括对象所对应的类、视图的ID、当前活动的ID 等等一系列信息。

### Extension Element Details

Set the properties of "viewerContribution". Required fields are denoted by "\*\*".

id\*:   
 targetID\*:



## 2) 新建一个菜单

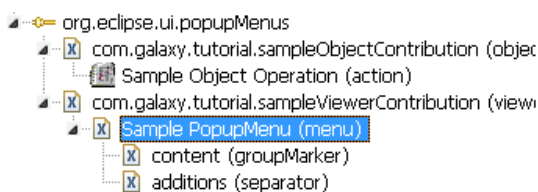
右键 `com.galaxy.tutorial.sampleViewerContribution`, `new`→`menu`, 新建一个菜单。其属性说明如下:

- id: 菜单的唯一标识。
- label: 菜单的显示名称。
- path: 菜单的相对位置。

### Extension Element Details

Set the properties of "menu". Required fields are denoted by "\*\*".

id\*:   
 label\*:   
 path:



右键 `menu`, 新建一个 `groupMarker` 和 `separator`, `name` 属性分别填写为 `content` 和 `additions`。  
`content` 是 `menu` 的子目录, `additions` 是一个分隔符。

## 3) 新建操作

右键 `com.galaxy.tutorial.sampleViewerContribution`, `new`→`action`, 新建一个操作。其属性说明同对象操作的属性说明。

Extension Element Details

Set the properties of "action". Required fields are denoted by "\*".

id\*:

com.galaxy.tutorial.viewerActionTest

label\*:

Sample Viewer Operation

class\*:

com.galaxy.tutorial.action.SampleViewerAct

Browse...

definitionId:

Browse...

menubarPath:

com.galaxy.tutorial.samplePopupMenu/content

icon:

icons/medal.gif

Browse...

helpContextId:

style:

▼

state:

▼

enablesFor:

overrideActionId:

Browse...

tooltip:

This is a Sample Viewer Operation!

org.eclipse.ui.popupMenus

com.galaxy.tutorial.sampleObjectContribution (object)

Sample Object Operation (action)

com.galaxy.tutorial.sampleViewerContribution (viewer)

Sample PopupMenu (menu)

content (groupMarker)

additions (separator)

Sample Viewer Operation (action)

#### 4) 实现操作的类

对象操作对应的类 `com.galaxy.tutorial.action.SampleViewerAction`，实现如下：

```
package com.galaxy.tutorial.action;
```

```
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IViewActionDelegate;
import org.eclipse.ui.IViewPart;
```

```
/**
 * Document Start
 * 视图操作的实现类
 * Document End
 * Author: 黄露
 * Time: 2009-7-16 下午08:47:54
 */
```

```
public class SampleViewerAction implements IViewActionDelegate {
```

```
/**
 * Document Start
 * 初始化
```

```

    * Document End
    * Author: 黄露
    * Time: 2009-7-16 下午08:47:54
    * @see
org.eclipse.ui.IViewActionDelegate#init(org.eclipse.ui.IViewPart)
    */
    @Override
    public void init(IViewPart view) {
        // TODO Auto-generated method stub

    }

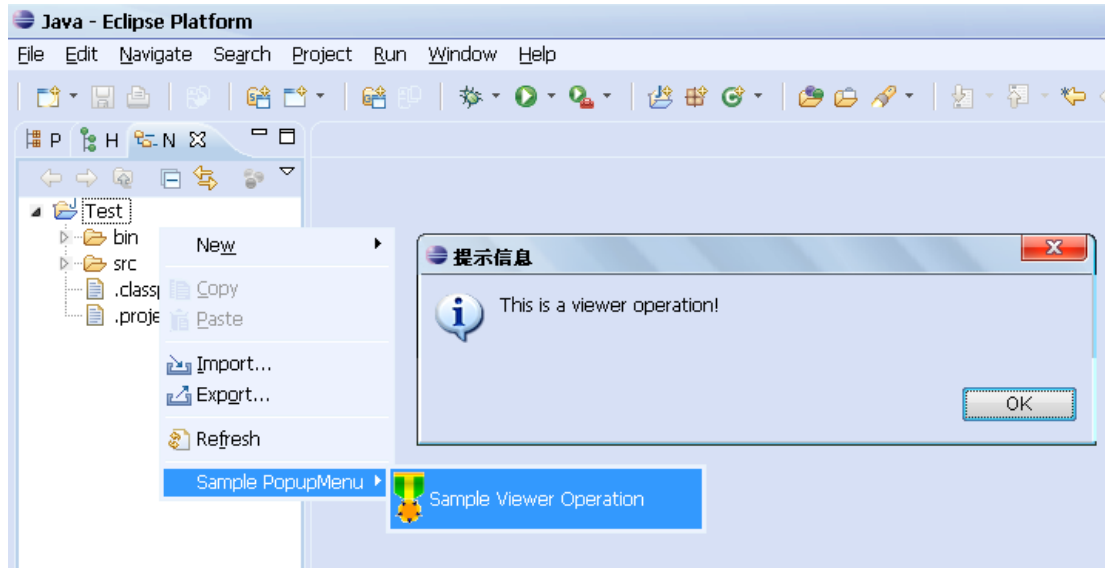
    /**
    * Document Start
    * 视图操作所执行的事件
    * Document End
    * Author: 黄露
    * Time: 2009-7-16 下午08:47:54
    * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
    */
    @Override
    public void run(IAction action) {
        MessageDialog.openInformation(null, "提示信息",
            "This is a viewer operation!");
    }

    /**
    * Document Start
    * 选中
    * Document End
    * Author: 黄露
    * Time: 2009-7-16 下午08:47:54
    * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.act
ion.IAction, org.eclipse.jface.viewers.ISelection)
    */
    @Override
    public void selectionChanged(IAction action, ISelection selection)
    {
        // TODO Auto-generated method stub
    }
}

```

## 5) 运行结果

运行插件后，在 Navigator 视图中右键，选择 Sample PopupMenu→Sample Viewer Operation，出现以下结果：



## 4. 编辑器上下文操作

# 九、 操作集

## 1. Workbench 菜单栏和工具栏操作集

### 1) 声明 workbench 操作集扩展点

打开 plug-in.xml 文件，切换到 Extensions 选项页。点击 Add，输入 org.eclipse.ui.actionSets，点击 Finish。

### 2) 新建操作集

右键 org.eclipse.ui.actionSets，选择 New→actionSet。id 和 label 不再赘述。

- visible: 操作集是否可见。
- description: 操作集的描述。

**Extension Element Details**

Set the properties of "actionSet". Required fields are denoted by "\*".

id\*:

label\*:

visible:

description:

org.eclipse.ui.actionSets

Sample ActionSet (actionSet)

### 3) 新建菜单

右键 Sample ActionSet，选择 New→menu。id 和 label 不再赘述。

➤ path: 操作集分组的 id。

同上面的操作，添加 groupMarker 和 separator，name 分别为 content 和 additions。

**Extension Element Details**

Set the properties of "menu". Required fields are denoted by "\*".

id\*:

label\*:

path:

Sample ActionSet (actionSet)

Tutorial (menu)

content (groupMarker)

additions (separator)

### 4) 新建操作

右键 Sample ActionSet，选择 New→action。

Extension Element Details

Set the properties of "action". Required fields are denoted by "\*".

id\*:

com.galaxy.tutorial.sampleWorkbenchAction

label\*:

Sample Workbench Action

accelerator:

definitionId:

Browse...

menubarPath:

com.galaxy.tutorial.tutorialMenu/content

toolbarPath:

additions

icon:

icons/CLASS.ICO

Browse...

disabledIcon:

icons/TortoiseLocked.ico

Browse...

hoverIcon:

Browse...

tooltip:

Sample Workbench Action

helpContextId:

style:

push

state:

pulldown:

class:

com.galaxy.tutorial.action.SampleWorkbenchAction

Browse...

retarget:

allowLabelUpdate:

enablesFor:

mode:

org.eclipse.ui.actionSets

Sample ActionSet (actionSet)

Tutorial (menu)

Sample Workbench Action (action)

## 5) 实现类

com.galaxy.tutorial.action.SampleWorkbenchAction 类的实现如下：

```

/**
 * Document Start
 * workbench操作集的实现类
 * Document End
 * Author: 黄露
 * Time: 2009-7-20 下午02:45:55
 */
public class SampleWorkbenchAction implements
IWorkbenchWindowActionDelegate {

    /**
     * Document Start
     *
     * Document End
     * Author: 黄露
    
```



```

* Time: 2009-7-20 下午02:45:55
* @see org.eclipse.ui.IWorkbenchWindowActionDelegate#dispose()
*/
@Override
public void dispose() {
    // TODO Auto-generated method stub

}

/**
 * Document Start
 *
 * Document End
 * Author: 黄露
 * Time: 2009-7-20 下午02:45:55
 * @see
org.eclipse.ui.IWorkbenchWindowActionDelegate#init(org.eclipse.ui.IWo
rkbenchWindow)
*/
@Override
public void init(IWorkbenchWindow window) {
    // TODO Auto-generated method stub

}

/**
 * Document Start
 *
 * Document End
 * Author: 黄露
 * Time: 2009-7-20 下午02:45:55
 * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
*/
@Override
public void run(IAction action) {
    MessageDialog.openInformation(null, "提示信息",
        "This is a workbench action!");
}

/**
 * Document Start
 *
 * Document End

```

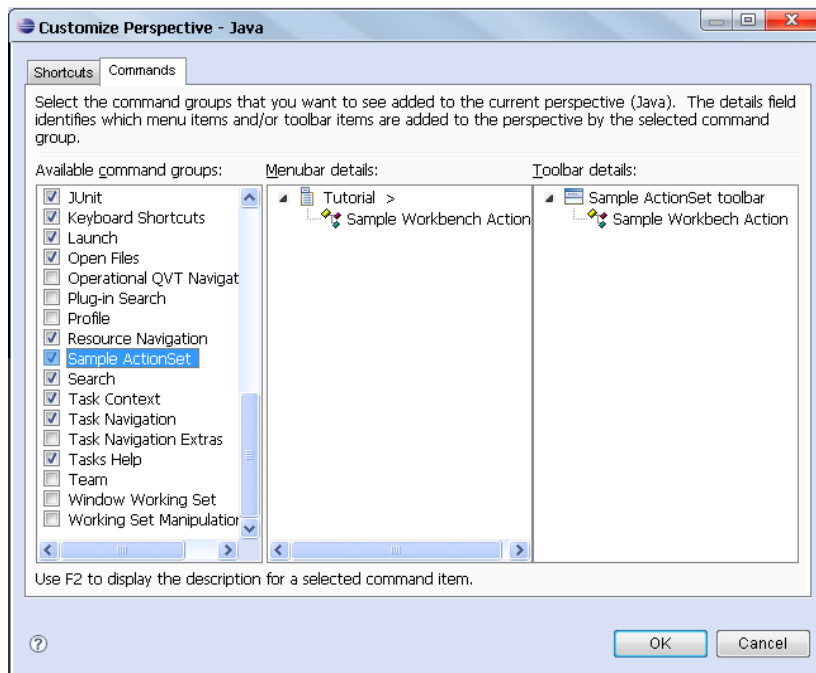
```

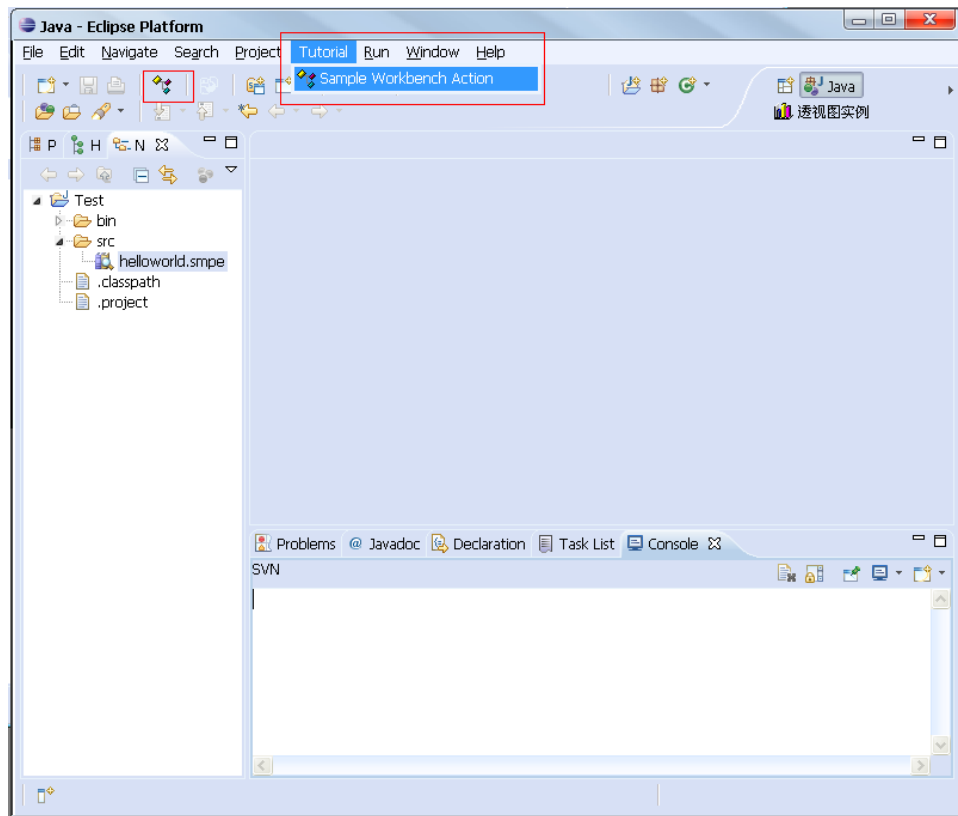
* Author: 黄露
* Time: 2009-7-20 下午02:45:55
* @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.action.IAction, org.eclipse.jface.viewers.ISelection)
*/
@Override
public void selectionChanged(IAction action, ISelection selection)
{
    // TODO Auto-generated method stub
}
}

```

## 6) 运行结果

运行插件，在菜单栏没有发现 Tutorial 的菜单。点击 Window→Customize perspective..., 切换至 Command 选项页，在 Available command groups 栏，选中 Sample ActionSet，点击 OK，即可在菜单栏显示 Tutorial 的菜单，在工具栏亦可看见 Sample Workbench Action 的图标。





## 2. 视图菜单栏和工具栏操作集

### 1) 声明视图操作集扩展点

打开 `plug-in.xml` 文件，切换到 `Extensions` 选项页。点击 `Add`，输入 `org.eclipse.ui.viewActions`，点击 `Finish`。

### 2) 新建 `viewContribution`

右键 `org.eclipse.ui.viewActions`，`new`→`viewContribution`。

- `id`: 唯一标识。
- `targetID`: 目标视图的 ID。

**Extension Element Details**

Set the properties of "viewContribution". Required fields are denoted by "\*".

`id*`:

`targetID*`:

`org.eclipse.ui.viewActions`  
`com.galaxy.tutorial.sampleViewContribution (viewContrit`

### 3) 新建菜单

同上面的步骤，右键 `com.galaxy.tutorial.sampleViewContribution`，新建菜单。

**Extension Element Details**

Set the properties of "menu". Required fields are denoted by "\*\*".

id*:	<input type="text" value="com.galaxy.tutorial.sampleViewIIMenu"/>
label*:	<input type="text" value="Sample ViewII Menu"/>
path:	<input type="text" value="additions"/>

### 4) 新建操作

右键 `com.galaxy.tutorial.sampleViewContribution`，new→action。类似的填写 action 的各项属性。

**Extension Element Details**

Set the properties of "action". Required fields are denoted by "\*\*".

id*:	<input type="text" value="com.galaxy.tutorial.sampleViewAction"/>
label*:	<input type="text" value="Sample View Action"/>
class*:	<input type="text" value="com.galaxy.tutorial.action.SampleViewAction"/> <input type="button" value="Browse..."/>
menubarPath:	<input type="text" value="com.galaxy.tutorial.sampleViewIIMenu/content"/>
toolbarPath:	<input type="text" value="additions"/>
icon:	<input type="text" value="icons/pdf.ico"/> <input type="button" value="Browse..."/>
disabledIcon:	<input type="text" value="icons/TortoiseDeleted.ico"/> <input type="button" value="Browse..."/>
hoverIcon:	<input type="text"/> <input type="button" value="Browse..."/>
tooltip:	<input type="text" value="Sample View Action"/>
helpContextId:	<input type="text"/>
style:	<input type="text" value="push"/> <input type="button" value="v"/>
state:	<input type="text"/> <input type="button" value="v"/>
enablesFor:	<input type="text"/>
mode:	<input type="text"/> <input type="button" value="v"/>

### 5) 实现类

`com.galaxy.tutorial.action.SampleViewAction` 类的实现：

```
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IViewActionDelegate;
import org.eclipse.ui.IViewPart;

/**
 * Document Start
 * 视图菜单栏和工具栏操作的实现类
 * Document End
 * Author: 黄露
 * Time: 2009-7-20 下午03:31:27
 */
public class SampleViewAction implements IViewActionDelegate {

    /**
     * Document Start
     * 初始化
     * Document End
     * Author: 黄露
     * Time: 2009-7-20 下午03:31:27
     * @see
     org.eclipse.ui.IViewActionDelegate#init(org.eclipse.ui.IViewPart)
     */
    @Override
    public void init(IViewPart view) {
        // TODO Auto-generated method stub
    }

    /**
     * Document Start
     * 执行
     * Document End
     * Author: 黄露
     * Time: 2009-7-20 下午03:31:27
     * @see
     org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
     */
    @Override
    public void run(IAction action) {
        MessageDialog.openInformation(null, "提示信息",
            "This is a view action!");
    }
}
```

```

    }

    /**
     * Document Start
     * 选中
     * Document End
     * Author: 黄露
     * Time: 2009-7-20 下午03:31:27
     * @see
     org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.act
     ion.IAction, org.eclipse.jface.viewers.ISelection)
     */
    @Override
    public void selectionChanged(IAction action, ISelection selection)
    {
        // TODO Auto-generated method stub

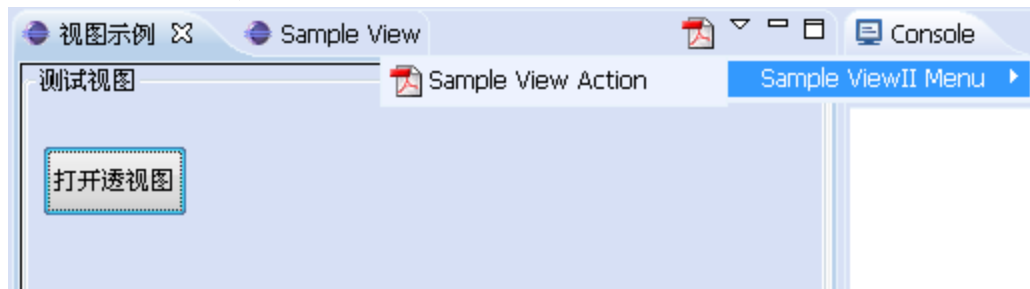
    }

}

```

## 6) 运行结果

运行插件，打开 Sample ViewII 对应的视图，在视图的工具栏可以看到刚才新建的操作，点击下三角符号，出现以下结果：



## 3. 编辑器菜单栏和工具栏操作集

### 1) 声明编辑器操作集扩展点

打开 plug-in.xml 文件，切换到 Extensions 选项页。点击 Add，输入 org.eclipse.ui.editorActions，点击 Finish。

## 2) 新建 editorContribution

右键 org.eclipse.ui.editorActions, new→editorContribution, 类似的填写各项属性。

- id: 唯一标识。
- targetID: 编辑器的 ID。

**Extension Element Details**

Set the properties of "editorContribution". Required fields are denoted by "\*".

id\*:

targetID\*:

org.eclipse.ui.editorActions

com.galaxy.tutorial.sampleEditorContribution (editorCon

## 3) 新建操作

右键 com.galaxy.tutorial.sampleEditorContribution, new→action, 填写属性如下:

**Extension Element Details**

Set the properties of "action". Required fields are denoted by "\*".

id\*:

label\*:

class\*:

accelerator:

definitionId:

menubarPath:

toolbarPath:

icon:

disabledIcon:

hoverIcon:

tooltip:

helpContextId:

style:

state:

enablesFor:

actionID:

mode:

org.eclipse.ui.editorActions

com.galaxy.tutorial.sampleEditorContribution (editorCon

Sample Editor Action (action)

#### 4) 实现类

com.galaxy.tutorial.action.SampleEditorAction 类的实现如下:

```
/**
 * Document Start
 * 编辑器操作集的实现类
 * Document End
 * Author: 黄露
 * Time: 2009-7-20 下午03:49:48
 */
public class SampleEditorAction implements IEditorActionDelegate {

    /**
     * Document Start
     * 设置活动的编辑器
     * Document End
     * Author: 黄露
     * Time: 2009-7-20 下午03:49:48
     * @see
     org.eclipse.ui.IEditorActionDelegate#setActiveEditor(org.eclipse.jfac
     e.action.IAction, org.eclipse.ui.IEditorPart)
     */
    @Override
    public void setActiveEditor(IAction action, IEditorPart targetEditor)
    {
        // TODO Auto-generated method stub
    }

    /**
     * Document Start
     * 执行事件
     * Document End
     * Author: 黄露
     * Time: 2009-7-20 下午03:49:48
     * @see
     org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
     */
    @Override
    public void run(IAction action) {
        MessageDialog.openInformation(null, "提示信息", "This is a editor
        action!");
    }
}
```



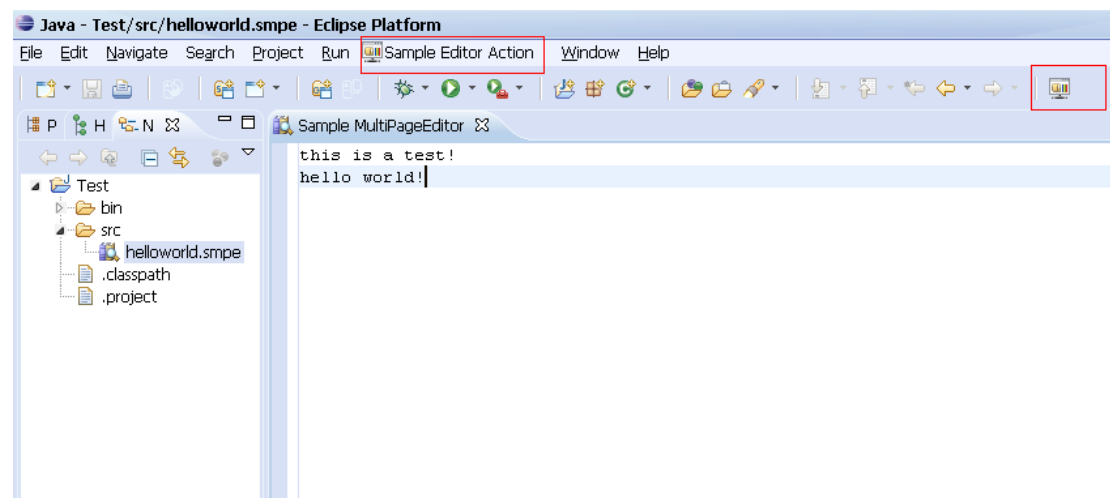
```

/**
 * Document Start
 * 选中
 * Document End
 * Author: 黄露
 * Time: 2009-7-20 下午03:49:48
 * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.act
ion.IAction, org.eclipse.jface.viewers.ISelection)
 */
@Override
public void selectionChanged(IAction action, ISelection selection)
{
    // TODO Auto-generated method stub
}
}

```

## 5) 运行结果

运行插件，打开一个 xx.smpe 文件，出现以下结果：



## 十、 首选项

可配置的首选项用来控制插件执行和显示信息，它在多个 Eclipse 会话间保存选项值。

## 1. 声明首选项扩展点

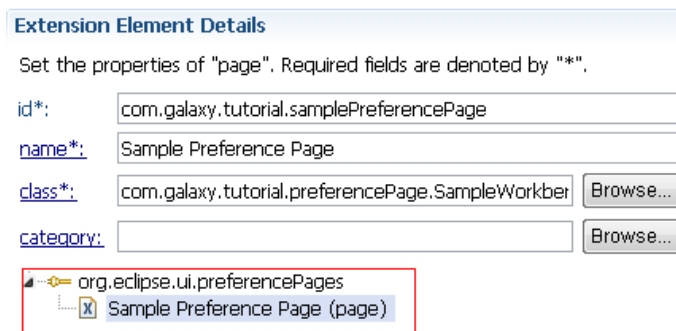
打开 `plug-in.xml` 文件，切换到 `Extensions` 选项页。点击 `Add`，输入 `org.eclipse.ui.preferencePages`，点击 `Finish`。

## 2. 扩展首选项

### 1) 新建首选项页

右键 `org.eclipse.ui.preferencePages`，`new`→`page`，新建一个首选项页。首选项页的具体属性如下：

- `id`：首选项页的唯一标识。
- `name`：首选项页名称。
- `class`：首选项页对应的实现类。
- `category`：首选项页对应的类别。



### 2) 实现类

首选项页必须实现 `org.eclipse.ui.IWorkbenchPreferencePage` 接口。而 `org.eclipse.jface.PreferencePage` 和 `org.eclipse.jface.preference.FieldEditorPreferencePage` 抽象类提供了实现该接口的很多基础工作。只要根据需实现其中最重要的方法 `createContents`（`Composite`）创建完成首选项页的控件即可。

`com.galaxy.tutorial.preferencePage.SampleWorkbenchPreferencePage` 类的实现如下：

```
package com.galaxy.tutorial.preferencePage;

import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.preference.PreferencePage;
import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.widgets.Button;
```

```

import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchPreferencePage;

/**
 * Document Start
 * 实现首选项页
 * Document End
 * Author: 黄露
 * Time: 2009-7-17 上午09:26:59
 */
public class SampleWorkbenchPreferencePage extends PreferencePage
implements
    IWorkbenchPreferencePage {

    /**
     * Document Start
     * 构造方法
     * Document End
     * Author: 黄露
     * Time: 2009-7-17 上午09:27:00
     */

    public SampleWorkbenchPreferencePage() {
        // TODO Auto-generated constructor stub
    }

    /**
     * Document Start
     * 带标题参数的构造方法
     * Document End
     * Author: 黄露
     * Time: 2009-7-17 上午09:27:00
     * @param title
     */

    public SampleWorkbenchPreferencePage(String title) {
        super(title);
    }

    /**
     * Document Start
     * 带标题和图片参数的构造方法

```

```

* Document End
* Author: 黄露
* Time: 2009-7-17 上午09:27:00
* @param title
* @param image
*/

public SampleWorkbenchPreferencePage(String title, ImageDescriptor
image) {
    super(title, image);
}

/**
* Document Start
* 创建首选项页的控件
* Document End
* Author: 黄露
* Time: 2009-7-17 上午09:26:59
* @see
org.eclipse.jface.preference.PreferencePage#createContents(org.eclips
e.swt.widgets.Composite)
*/
@Override
protected Control createContents(Composite parent) {
    // 添加一个按钮
    Button button = new Button(parent, SWT.PUSH);
    button.setText("首选项页");
    button.addSelectionListener(new SelectionAdapter() {
        public void widgetSelected(SelectionEvent e) {
            MessageDialog.openInformation(null, "Title Name",
                "This is a preferencepage!");
        }
    });
    return parent;
}

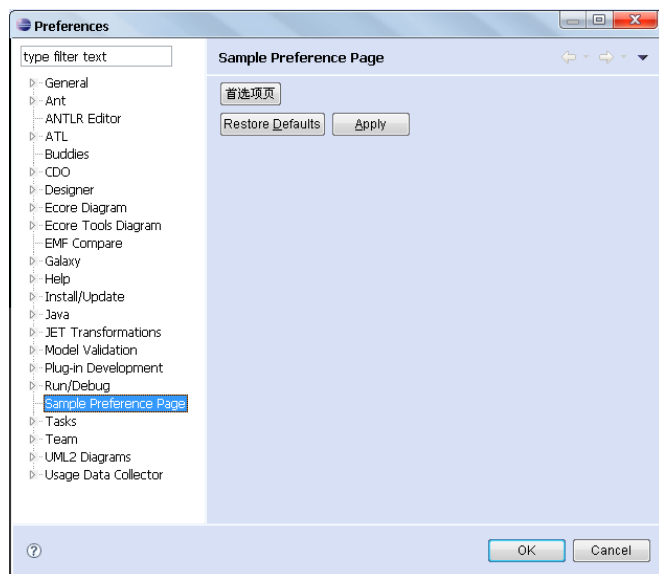
/**
* Document Start
* 初始化
* Document End
* Author: 黄露
* Time: 2009-7-17 上午09:26:59
* @see
org.eclipse.ui.IWorkbenchPreferencePage#init(org.eclipse.ui.IWorkbench

```

```
h)
    */
    @Override
    public void init(IWorkbench workbench) {
        System.out.println("init");
    }
}
```

### 3) 运行结果

运行插件后，选择 Window→Preferences，打开首选项对话框，选择 Sample Preference Page，出现以下结果：



## 十一、 属性视图

属性应用于出现在 Eclipse 环境中的资源或者其他对象。访问对象属性的一种典型方式，从上下文菜单中选择“属性”命令打开“属性”对话框。另一种方式是打开“属性”视图，它将显示所选择对象的属性。

### 1. 声明属性页扩展点

打开 plug-in.xml 文件，切换到 Extensions 选项页。点击 Add，输入 org.eclipse.ui.propertyPages，点击 Finish。

## 2. 扩展属性页

### 1) 新建属性页



右键 `org.eclipse.ui.propertyPages`, `new`→`page`, 新建一个属性页。属性页的具体属性如下：

- `id`: 属性页的唯一标识。
- `name`: 属性页的名称。
- `class`: 属性页对应的实现类。
- `icon`: 图标。
- `objectClass`: 属性页所对应的对象。例如 `org.eclipse.core.resources.IFile` 表示是文件对象。
- `nameFilter`: 对象的后缀名。
- `adaptable`: 是否可适配为 `IResource` 类型。
- `category`: 属性页的分组类别。

**Extension Element Details**

Set the properties of "page". Required fields are denoted by "\*".

<code>id*</code> :	<input type="text" value="com.galaxy.tutorial.samplePropertyPage"/>
<code>name*</code> :	<input type="text" value="Sample Property Page"/>
<code>class*</code> :	<input type="text" value="com.galaxy.tutorial.propertyPage.SampleWorkber"/> <input data-bbox="826 969 922 999" type="button" value="Browse..."/>
<code>icon</code> :	<input type="text" value="icons/property.ico"/> <input data-bbox="826 1014 922 1043" type="button" value="Browse..."/>
<code>objectClass</code> :	<input type="text" value="org.eclipse.core.resources.IFile"/> <input data-bbox="826 1059 922 1088" type="button" value="Browse..."/>
<code>nameFilter</code> :	<input type="text" value="*.smpe"/>
<code>adaptable</code> :	<input type="button" value="v"/>
<code>category</code> :	<input type="text"/> <input data-bbox="826 1193 922 1223" type="button" value="Browse..."/>

 `org.eclipse.ui.propertyPages`  
 `Sample Property Page (page)`

### 2) 实现类

同首选项页，实现最主要的方法 `createContents`（`Composite`）即可。

`com.galaxy.tutorial.propertyPage.SampleWorkbenchPropertyPage` 类的实现如下：

```
package com.galaxy.tutorial.propertyPage;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.ui.IWorkbenchPropertyPage;
import org.eclipse.ui.dialogs.PropertyPage;
```

/\*\*

```

* Document Start
* 属性页的实现类
* Document End
* Author: 黄露
* Time: 2009-7-17 上午 09:57:26
*/

public class SampleWorkbenchPropertyPage extends PropertyPage implements
    IWorkbenchPropertyPage {

    /**
     * Document Start
     * 构造方法
     * Document End
     * Author: 黄露
     * Time: 2009-7-17 上午 09:57:26
     */

    public SampleWorkbenchPropertyPage() {
        // TODO Auto-generated constructor stub
    }

    /**
     * Document Start
     * 创建属性页的控件
     * Document End
     * Author: 黄露
     * Time: 2009-7-17 上午 09:57:26
     *
     * @see
     org.eclipse.jface.preference.PreferencePage#createContents(org.eclipse.swt.widgets.Composite)
     */
    @Override
    protected Control createContents(Composite parent) {
        // 添加一个按钮
        Button button = new Button(parent, SWT.PUSH);
        button.setText("属性页");
        button.addSelectionListener(new SelectionAdapter() {
            public void widgetSelected(SelectionEvent e) {
                MessageDialog.openInformation(null, "属性示例",
                    "This is a propertypage!");
            }
        });
        return parent;
    }
}

```

```
}
```

### 3) 运行结果

运行插件，选中工程中的 helloworld.smpe 文件，右键选择 Properties，弹出属性对话框，出现以下结果：

