# Eclipse RCP 入门

**zhlmmc 2006-4-23**

最近做了一个基于 RCP 的项目，感受颇深，觉得 RCP 有希望扭转 Java 桌面应用的颓势。在项目中积累了一点经验与心得，拿出来与大家分享，希望能给 RCP 初学者一点帮助。我研究 Eclipse 插件开发已经有一段时间了，但是我并没有很系统的学习过 Eclipse 的插件开发，往往只是做项目需要临时学的一点，所以讲的东西难免粗陋，请见谅。

一、Eclipse 简介

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。事实上，Eclipse 已经超越了一般 Java IDE 的概念。Eclipse 是一个平台，一个开放的平台，你可以为 Eclipse 添加任何你想要的功能，比如播放音乐，观看电影，聊天……这些不是天方夜谭，而是已经实现的事实。虽然 Eclipse 可以添加很多附加功能，可以编辑 C/C++，可以编辑 Word 文件，可以开发 UML 等等，但是 Eclipse 最基本，也是最强大的功能还是 Java IDE。

二、RCP 简介

RCP 的全称是 Rich Client Platform，可以把它看成是 Eclipse 的骨架，其他的插件是器官与血肉。我们可以把这个骨架拿过来填入自己的器官和血肉，这样就创造了我们自己的“Eclipse”!

使用 RCP 来开发 Java 桌面应用可以把开发的焦点转移到系统的逻辑功能上，而不是界面上。我们自己的程序可以继承 Eclipse 的风格与功能，而不用自己去编写诸如菜单，工具条，子窗口等等的界面元素。甚至我们可以把 Eclipse 本身的功能插件，比如 Console 拿来放在自己的程序里，这样就避免了很多重复开发。

三、知识准备

我写这篇文章并不是面向 Java 的初学者，而是面向有一定 Eclipse 使用基础的开发者。所以我假设你已经具备一下基本知识：

1、 Java 基础

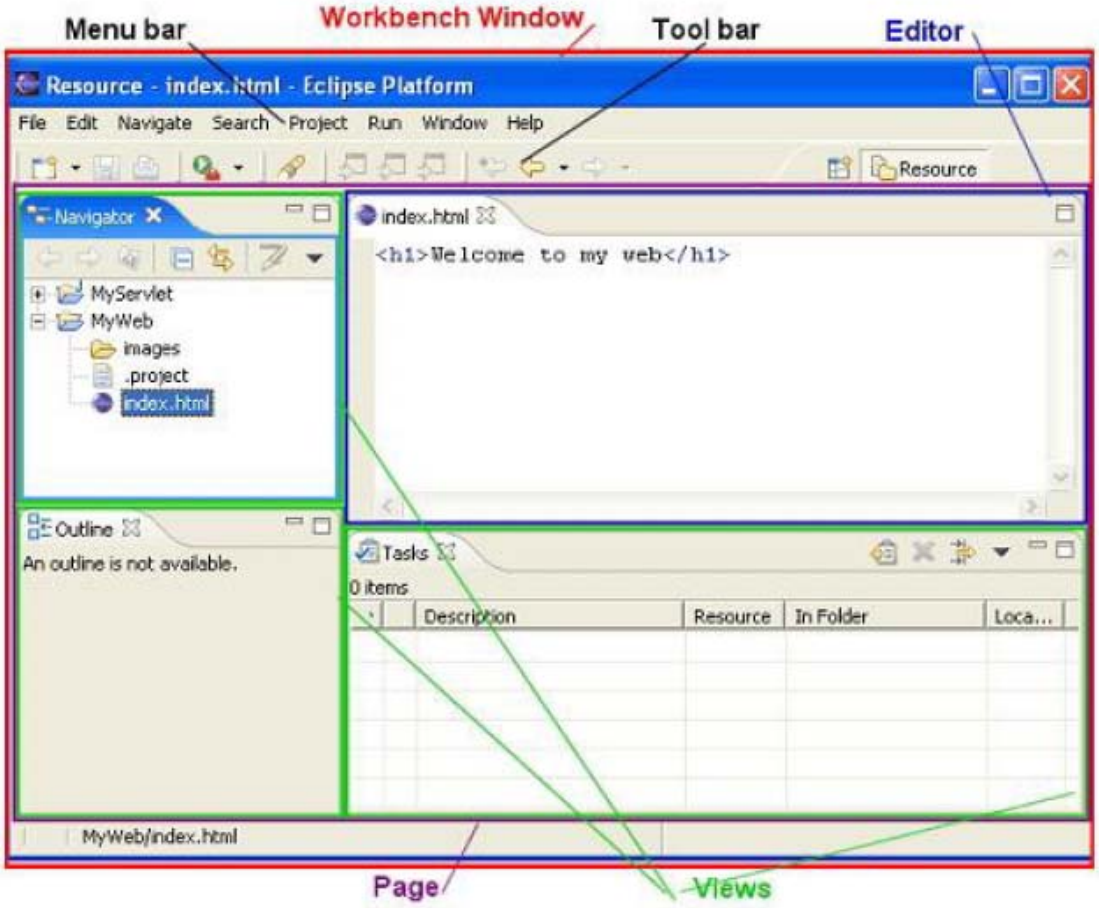
2、 用过 Eclipse 进行开发

3、 SWT/JFace 开发基础（可选）

如果你还不具备上述条件，那么看我的文章你会看的很郁闷，建议你先去学习这些基本知识。

四、Eclipse 组件

在开发 Eclipse 插件（RCP 可以看成是 Eclipse 的插件，只不过是脱离 Eclipse 运行的）之前，得先对 Eclipse 的结构有个了解。这里我简单介绍一下 Eclipse 的基本组件，这些名词可能比较陌生，但这都是开发 Eclipse 插件必须了解的。




如上图所示，我逐一介绍一个各个组件：

1、 Menu bar：这个东西你一定不陌生，每个软件都有的。不过 Eclipse 的菜单栏是动态的，也就是说，根据所编辑的内容不同，显示的菜单也可以不一样。

2、 Tool bar：这个东西也是每个软件都有的，和菜单栏一样，工具栏也是可以根据所编辑的内容不同而不同。

3、 Editor：编辑器，Eclipse 的主要编辑工作是在 Editor 里面完成的。

4、 View：视图，视图是为了方便用户编辑提供一些辅助功能或编辑一些属性。比如最常见的 Outline 视图往往用来提供当前编辑的文档的结构。

5、 Page：页，一个页表示了当前用户的工作状态，包括 View 和 Editor。

6、 Workbench Window：涵盖所有上述组件的组件叫做工作台窗口（这个名词的翻译我没见到过，我这里纯粹是直译，感觉有些词不达意）。Eclipse 是允许创建多个工作台的。通过 Window->New window 菜单可以创建当前工作台的副本。

除了这些组件以外我还要介绍另外两个概念，一个是“Work Space”，在 Eclipse 启动的时候都要求指定一个 Work Space，而且 Work Space 是不能被共用的。也就是说在同一时间，同一个 Work Space 只可以被一个 Eclipse 使用。但是一个 Work Space 是可以被多个 Workbench Window 共享的。很容易联想到，Workbench Window 上面还有一层 Workbench。事实上 Workbench 才是 Eclipse 的 UI 的最高管理者。另外一个概念是“Perspective”，中文翻译是“透视图（或者观察点）”。所谓 Perspective 是指当前 Page 的布局。最常见的是 Java 透视图和 Debug 透视图，可以看到这两个透视图的 Page 排布完全不一样。通过切换透视图可以很方便的切换开发环境以完成不同功能的开发。这里可以看出 View 和 Editor 的区别，Editor 是在不同的透视图中共享的，而 View 不是。

五、开发前的准备

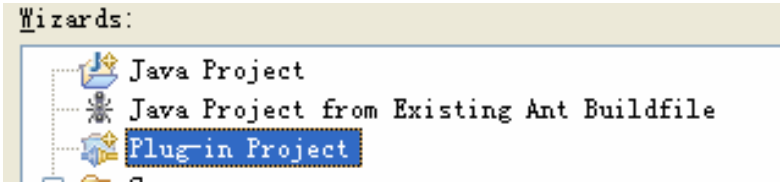
Eclipse 是自带插件开发环境 PDE（Plug-in Develop Environment）的，所以要开发 Eclipse 插件只需要下载一个标准的 Eclipse 即可。我现在用的 Eclipse 版本是 3.1.2，是最新的稳定版，建议下载这个版本进行开发（我用的是英文版，所以下文提到的 Eclipse 相关的选项都是英文描述）。

虽然 Eclipse 生来就是开放的插件平台，但是 Eclipse 插件，特别是 RCP 是从 3.0 开始才走红的。Eclipse 3.0 是一个具有里程碑意义的版本，它对 Eclipse 以前的结构做了一定的改进，并且升级了 PDE，极大的简化了插件开发的配置，基本上实现了插件开发全图形的化操作，使得插件开发人员可以专注于插件功能的开发，而不用去管琐碎的配置文件。

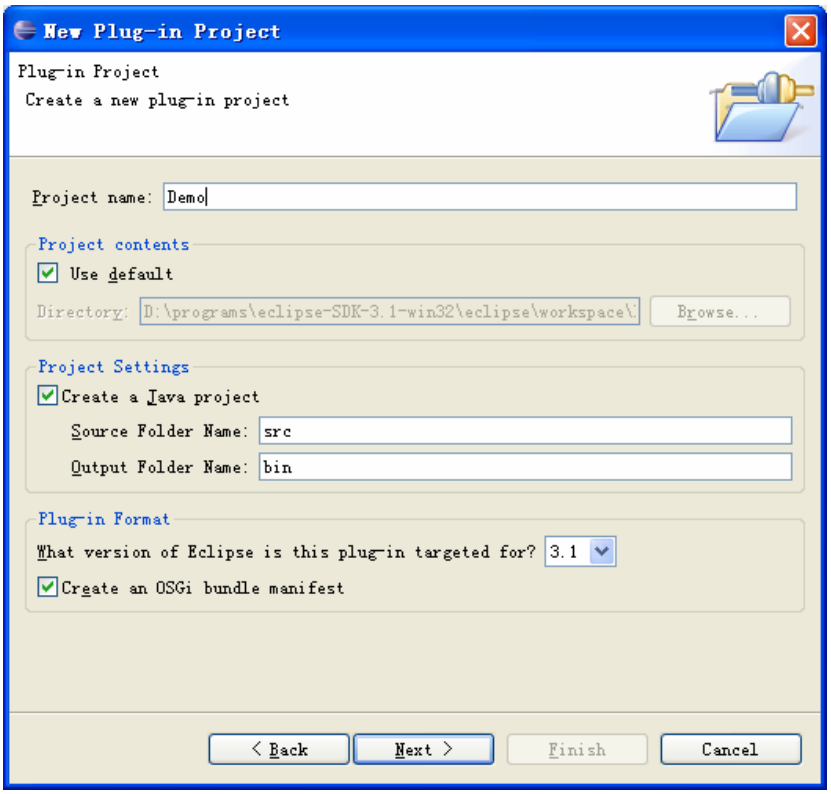
六、第一个 RCP 程序

Eclipse 提供了一些 RCP 程序的模板，通过 PDE 的插件创建向导能直接生成一个可用的 RCP 程序。

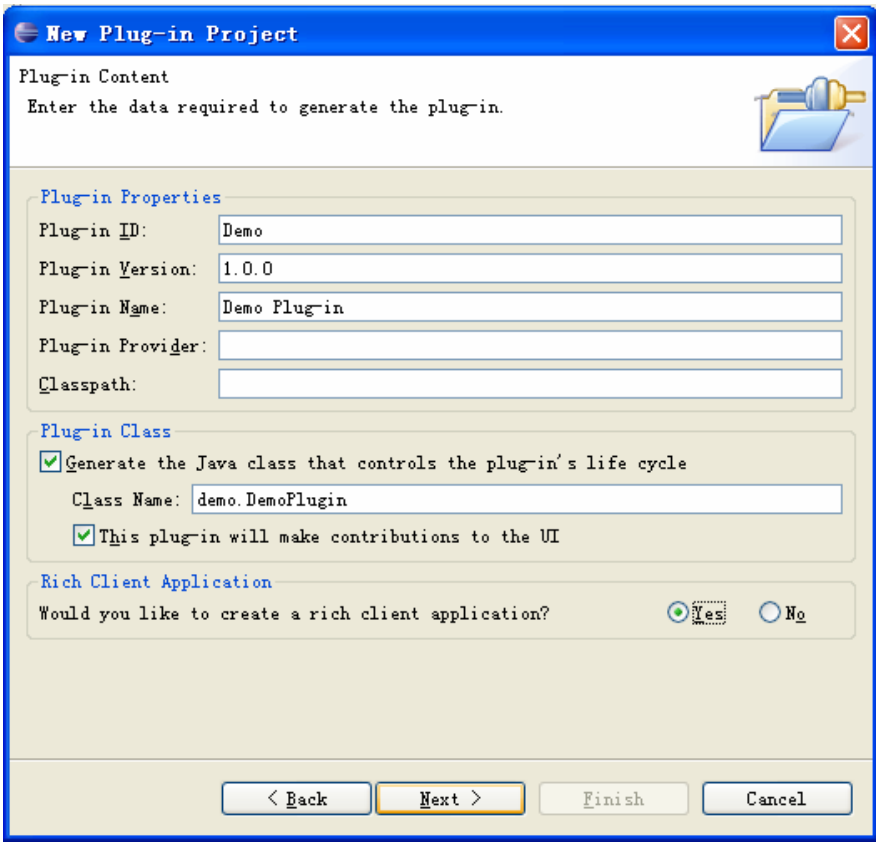
首先要新建一个 Plug-in Project



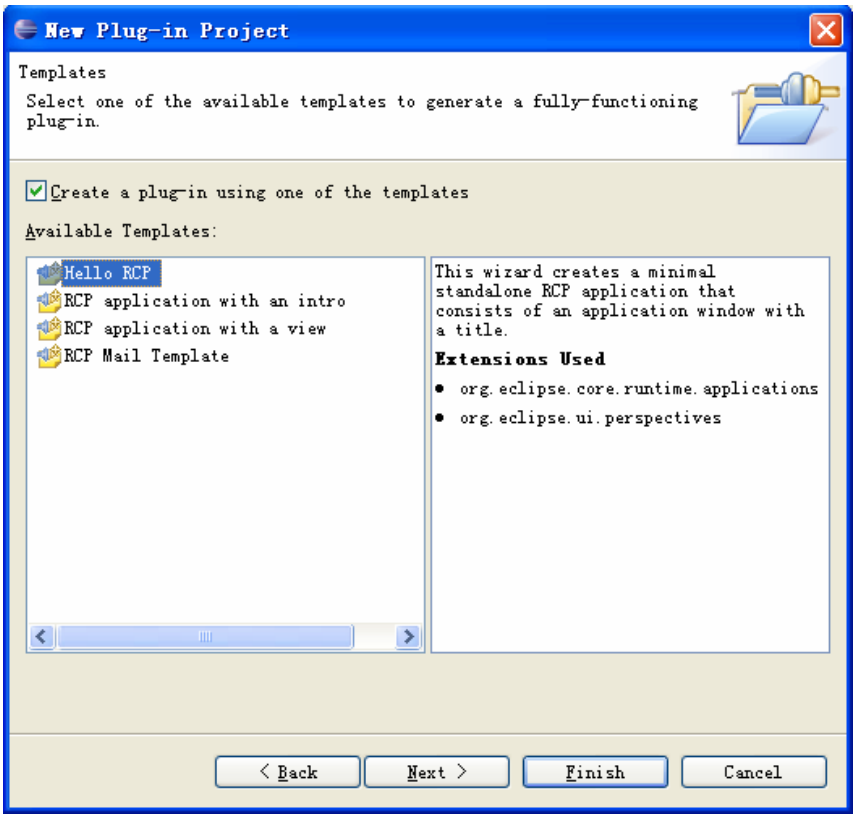

然后输入 Project 名字，其他都用默认选项就行，点击“next”

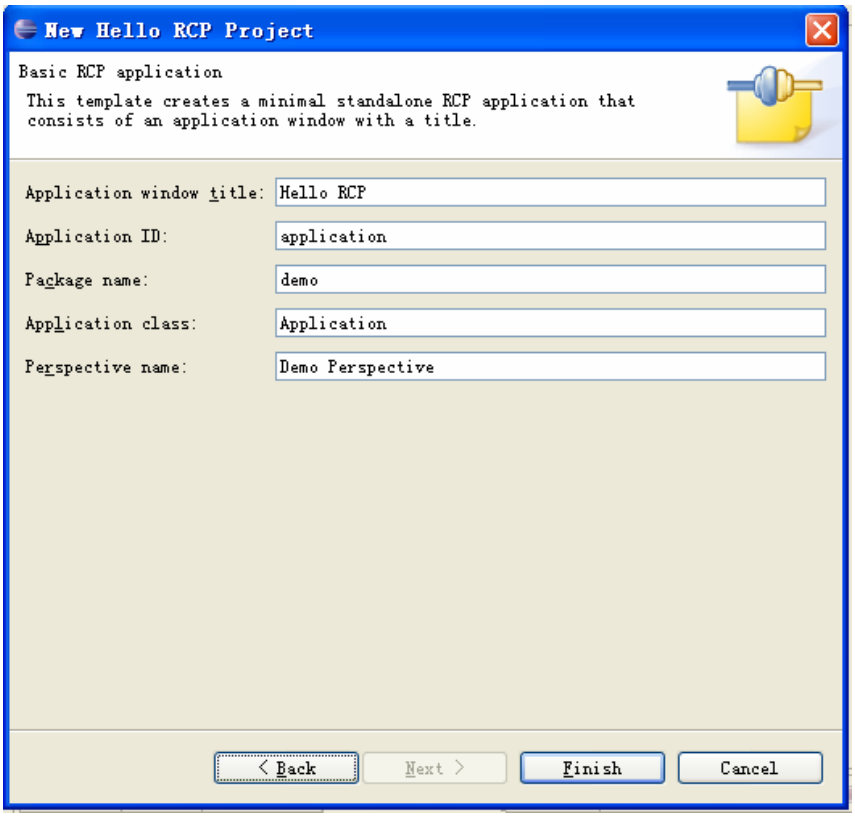



在 Rich Client Application 部分选择“Yes”，点击“Next”



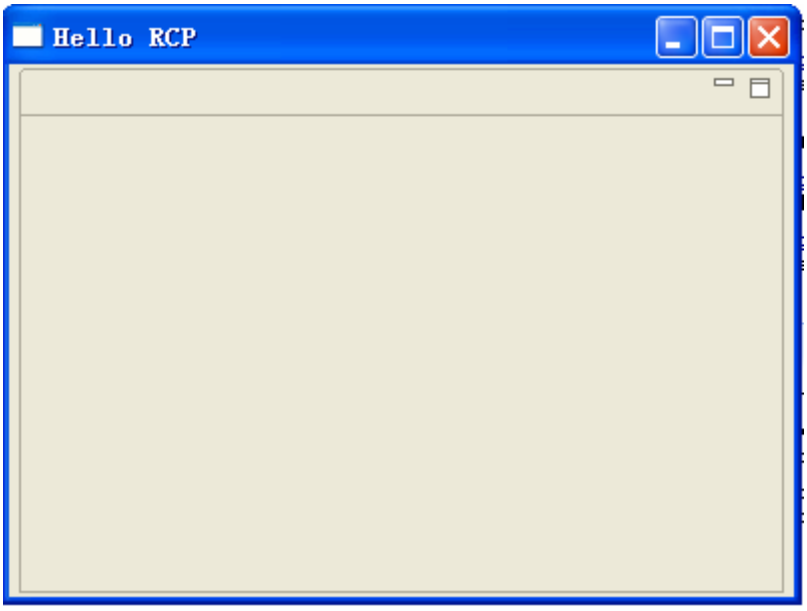

模板选择 Hello RCP，点击“Next”






点击“Finish”将提示转换到 Plug-in development perspective，选择“Yes”。

我们先看一下刚才新建的 RCP 的运行结果，右键点击“Demo”project，选择 Run As，选择 Eclipse Application，就可以看到一个最简单的 RCP 程序窗口。




接下来我先分析一下这个应用程序的结构。在 Demo 包下面有六个类和三个配置文件，下面我解释一下这些元素：

1、 Application：这个类是程序的入口，虽然没有 Main 函数，但是这个类实现了 IPlatformRunnable 接口，当 JVM 完毕，初始化 RCP 框架以后会调用这个类的 run 函数来完成 UI 设置和开始执行我们指定的程序功能。在绝大多

数 RCP 程序中，这个类不用更改。

2、 ApplicationActionBarAdvisor：简单的说这个类是用来配置程序的菜单栏和工具栏的，具体的应用在后面会讲到。

3、 ApplicationWorkbenchAdvisor：这个类是 RCP 程序的 Workbench，RCP 是 Eclipse 的简化，但是所有的组件都是和 Eclipse 一样的。一个 RCP 程序也只能有一个 Workbench。

4、 ApplicationWorkbenchWindowAdvisor ： 这 个 类 是 RCP 的 WorkbenchWindow ， 隶 属 于 当 前 的 Workbench 。 可 以 有 多 个 WorkbenchWindow。

5、 DemoPlugin：这个类代表了我们的插件，因为 RCP 程序也是一个插件，Eclipse 中所有的插件都必须继承 AbstractUIPlugin。这个类为我们提供了很多和插件相关的信息，比如插件的资源，配置等等。

6、 Perspective：是我们新建的 RCP 的默认透视图。可以在这个类中指定 View 和 Editor 的排布。

7、 plugin.xml：这个文件是我们插件的配置文件，包括我们的插件用了哪些其他的插件，具体是怎么配置这些插件的等等。

8、 build.properties：这个文件是用来配置我们插件的编译信息的，用来指定如何编译我们的插件。

9、 MANIFEST.MF：这个文件用来指定我们插件的元数据，比如插件的版本信息。一般来说，这个文件不用手动的去更改。

## 七、添加菜单和工具栏

这一节我演示以下如何为程序创建菜单和工具栏。在创建菜单和工具栏之前我想先介绍一个概念“Action”。开发 Eclipse 插件会经常看到这个东西，它和事件处理有点相似，如果你了解 JFace 的话，这里的 Action 你应该很熟悉，两者是一样的，我们自己定义的 Action 必须继承 org.eclipse.jface.action.Action。程序可以定义当用户执行某项操作时触发某个 Action。比如用户点击工具栏的一个按钮，或者选中了某个菜单项。

下面我们创建一个类 HelloAction

```
public class HelloAction extends Action
{
    private IWorkbenchWindow window;

    public HelloAction(IWorkbenchWindow window)
    {
        this.window = window;
        this.setText("Hello");
        ImageDescriptor imgDes =
            WorkbenchImages.getImageDescriptor(
            IWorkbenchGraphicConstants.IMG_ETOOL_HOME_NAV);
        this.setImageDescriptor(imgDes);
    }
}
```

由于 Action 是一个 UI 元素，所以往往创建一个带 IWorkbenchWindow 参数的构造函数，以便在 Action 内部调用。setText() 是设置 Action 对外显示的名字。setImageDescriptor()是设置 Action 的图标，这里我为了简化，用的是 Eclipse Workbench 自带的图标。

下面增加一个 run 方法，功能是弹出一个对话框显示“Hello World!”

```
public void run()
{
    MessageBox mb = new MessageBox(window.getShell(), SWT.OK);
    mb.setMessage("Hello world!");
    mb.setText("Demo");
    mb.open();
}
```

这里你可以看到构造函数传入 window 的好处了吧。

上述代码都很简单，我不多解释，接下来我们把创建的 Action 关联到菜单栏和工具栏上。我们可以把同一个 Action 同时关联到多个菜单项和工具栏按钮，这和事件处理中的事件处理函数可以被多个事件共享一样。

上文我已经提到过，ApplicationActionBarAdvisor 是用来配置工具栏和菜单栏的，那么我们现在来看以下具体如何操作。修改 ApplicationActionBarAdvisor 如下：

```
public class ApplicationActionBarAdvisor extends ActionBarAdvisor
{
    private HelloAction helloAction;

    public ApplicationActionBarAdvisor(IActionBarConfigurer configurer)
    {
        super(configurer);
    }

    protected void makeActions(IWorkbenchWindow window)
    {
        super.makeActions(window);
        helloAction = new HelloAction(window);
    }

    protected void fillMenuBar(IMenuManager menuBar)
    {
        super.fillMenuBar(menuBar);
        MenuManager demoMenu = new MenuManager("&Demo", "");
        demoMenu.add(helloAction);
        menuBar.add(demoMenu);
    }

    protected void fillCoolBar(ICoolBarManager coolBar)
    {
        IToolBarManager toolbar = new ToolBarManager(SWT.FLAT | SWT.RIGHT);
        coolBar.add(new ToolBarContributionItem(toolbar, "main"));
        toolbar.add(helloAction);
    }
}
```

代码看起来有点晕，我来一一解释。首先是 makeActions()方法，这个方法是用来初始化 action 的，所有要用到的 action 可以都在这里初始化，但是这一步与界面无关。fillMenuBar()是用来设置菜单栏的。先创建一个 MenuManager，然后把 helloAction 加入到这个菜单下面，如果有多个 action 的话，可以加入同一个 MenuManager，这样 demoMenu 相当于一个下拉菜单，而加入的 action 就是菜单项。最后把 demoMenu 加入到菜单栏。当然，一个菜单栏可以加入多个下拉菜单。多级菜单也是可以做的，有兴趣的可以自行研究。fillCoolBar()是用来设置工具栏的，原理和菜单栏是一样的，coolBar 也有分组的，每个 toolbar 就是一组。这里的 toolbar 相当于 demoMenu。

运行一下程序你会发现菜单栏出来了，但是工具栏没有出现。原因是 Hello RCP 默认是不显示工具栏的，我们得改一下设置。打开 ApplicationWorkbenchWindowAdvisor 类，更改 preWindowOpen 方法：

```
public void preWindowOpen()
{
    IWorkbenchWindowConfigurer configurer = getWindowConfigurer();
    configurer.setInitialSize(new Point(400, 300));
    //configurer.setShowCoolBar(false);
    configurer.setShowStatusLine(false);
    configurer.setTitle("Hello RCP");
}
```

以上代码也很简单，从字面就能看出什么意思，我也不多解释，注意看注释掉的那行。preWindowOpen()这个函数在程序的窗口打开之前被调用，可以在这个函数里设置一些初始化的内容。类似的还有 postWindowClose()等等，你可以查阅 API 来了解这些函数的用途。

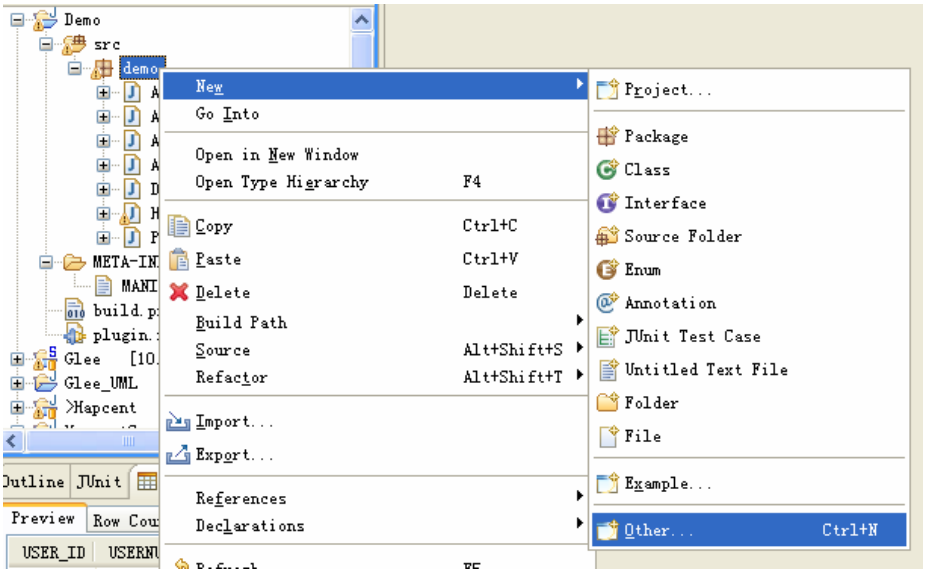
现在我们的程序可以正确的运行了，看一下运行结果：


八、创建 View 和 Editor

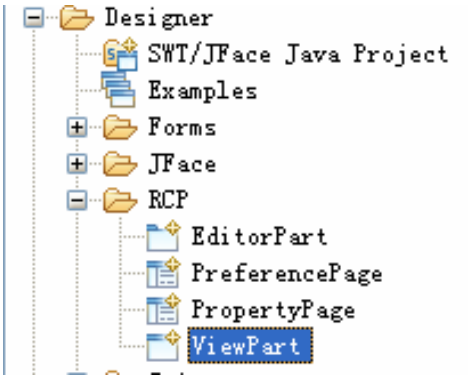
我用 Designer 来编辑 View 和 Editor。我用的版本是 WindowBuilder Pro 4.1.0，建议你也去下载一个，因为这个插件能够支持 SWT，JFace，RCP，Swing 的可视化编辑，极大的缩短了界面开发的时间。不过这个是商业软件，要破解了才能用，网上有很多破解的方法，google 一下就行了。

下面开始创建第一个 View。

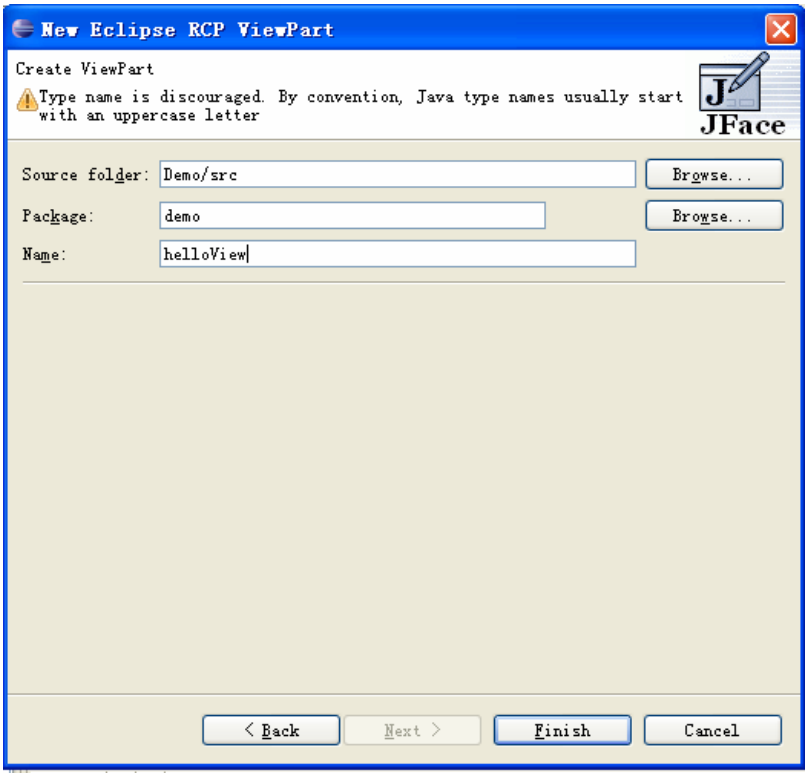
右键点击 demo 包，选择“New”，“Other”




然后选择 Designer->RCP->ViewPart，点击“Next”




输入“HelloView”，点击“Finish”




你可以在 Source 和 Design 之间切换。View 相当于一个窗口，在 View 的界面上可以放任何组件。我拖了一个 button 上去，名字就叫 button。HelloView 类中要添加一个属性 ID，我稍后解释这个属性。

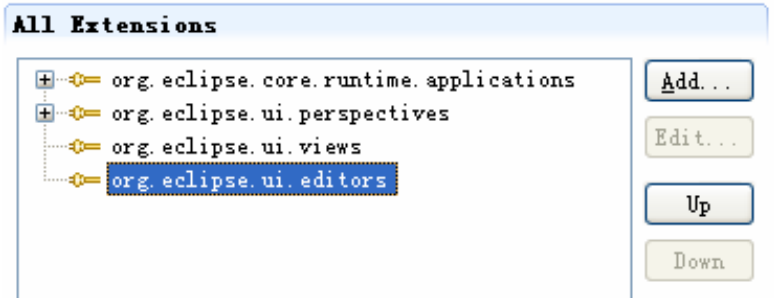
```
public static final String ID="Demo.helloView";
```

可以用同样的方法创建一个 EditorPart，名字就叫 HelloEditor。在 editor 上面放一个 text 组件吧，这样看起来像个 editor，然后把 HelloEditor 的 layout 改成 FillLayout。当然也要为 HelloEditor 添加一个 ID 属性。这里需要改一下 Editor 的 init 方法，否则 editor 无法正常运行。

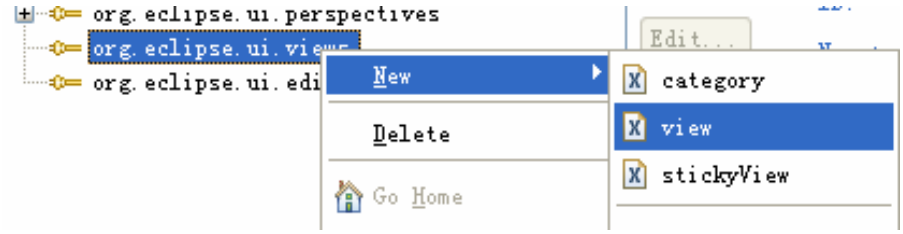
```
public void init(IEditorSite site, IEditorInput input)
        throws PartInitException
{
    this.setSite(site);
    this.setInput(input);
}
```

setSite 是让 editor 能够使用 workbench 所提供的一些功能，EditorSite 可以看成是一个代理，EditorPart 通过 EditorSite 来访问 workbench。EditorSite 是 editor part 和 workbench 之间的一层接口，详细情况你可以去 google 一下或者参考 API。setInput 的意思就很明了，就是让 editor 知道显示什么东西。这样我们就创建好了基本的 View 和 Editor，下面来看怎么把它们加到我们的程序中去。

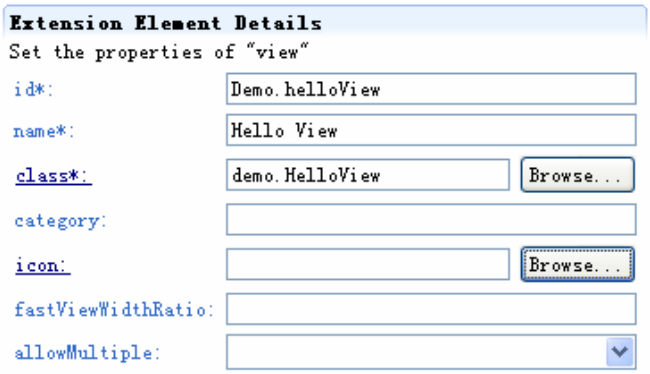
打开 plugin.xml，选择 Extensions 标签页，在 All Extensions 部分点击“Add”按钮，选择 org.eclipse.ui.views, org.eclipse.ui.editors, 两个 extension，结果如下：




右键 org.eclipse.ui.view，选择 New->view




在 Extension Element Details 部分设置 View 的属性。ID 可以随意设置，但是不能重复，我这里设置为 Demo.helloView（虽说是随意设置，但是程序是通过这个 ID 来获得 View 的，所以最好与 HelloView 中的 ID 属性一致，我在 HelloView 中加入这个 ID 属性也是为了方便程序获得 HelloView），Name 也可以随意设置，我这里设置为 Hello View，这个属性会显示在 View 的顶端，就像 Outline View 顶部显示的名称是“Outline”。class 选择刚才创建的 demo.HelloView。icon 是显示在 View 顶部的图标，没有可以不设置。

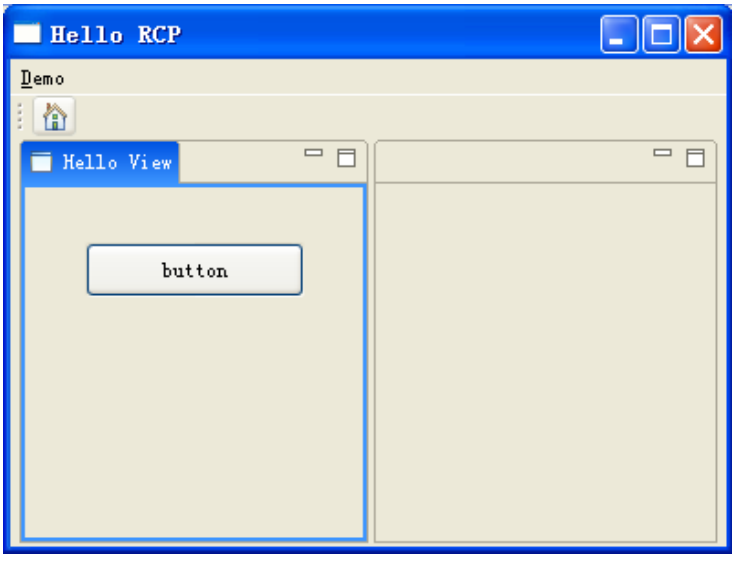



类似的创建 Hello Editor，设置 Hello Editor 的详细属性，设置 id, name, class 三个属性以外，还要设置 icon 属性（Editor 的 Icon 属性是必须的！），随便给个图片文件就行，其他的属性可以不填。这里属性比较多，有兴趣可以去看 Eclipse 的插件开发帮助，里面对每个属性都有解释。


到目前为止，我们的配置工作已经完成，但是要让我们的程序正确的显示 View 和 Editor，我们还要更改默认的 Perspective。打开 Perspective.java，修改 createInitialLayout() 方法（默认这个方法是空的）

```
public void createInitialLayout(IPageLayout layout)
{
    layout.addStandaloneView(HelloView.ID, true,
        IPageLayout.LEFT, .50f, layout.getEditorArea());
    layout.getViewLayout(HelloView.ID).setCloseable(false);
}
```

这个方法实际上就是修改 layout，在这个方法中安排每个 view 的位置和大小。addStandaloneView()是为当前的 Perspective 添加一个独立的 View，所谓独立的 View 就是 View 所占的位置不能和其他 View 共享，你在 Eclipse 里应该看到过几个 View 叠在一起的情况。接下来一行设置是去除了关闭 View 的功能，就是这个 View 的顶部没有关闭按钮。我们来看一下运行结果



也许你正在想怎么打开 Editor，IPageLayout 并不能直接把一个 Editor 加入显示，但是它会预留一块空间给 Editor。如果你仔细观察上面那段代码，你会发现，事实上整个空间都是 Editor 的，View 是在瓜分 Editor 的空间。接下来我将介绍如何打开 Editor。

## 九、设置并打开 Editor

相对 View 来说，Editor 有点麻烦。因为要打开 Editor 的话必须给 Editor 内容，因为 Editor 是个编辑器，你得让它知道要编辑什么东西它才能打开。这里的内容就是 Eclipse 里面的 EditorInput。没有现成合适的 EditorInput 用（一般情况下可以用 FileEditorInput，把某个文件作为 Input 让 Editor 打开，在 Eclipse 里面双击打开某个文件就是这个过程），我这里创建一个 HelloEditorInput 继承 IEditorInput 接口。用 Eclipse 向导创建类的时候，先选择 Interface，然后记得选中“Inherited abstract methods”，这样会自动继承所有 abstract 的方法，不用再去手工创建了。

HelloEditorInput 的代码比较长，我不全部贴出来了，我贴一部分。我贴的两个方法是必须要修改的，这个类的其他代码用 Eclipse 自动生成的就好（做实际开发的时候肯定是要改的，我这里只是演示一个过程，所以尽量简化了）。

```
public String getName()
{
    return "Hello";
}

public String getToolTipText()
{
    return "Demo";
}
```

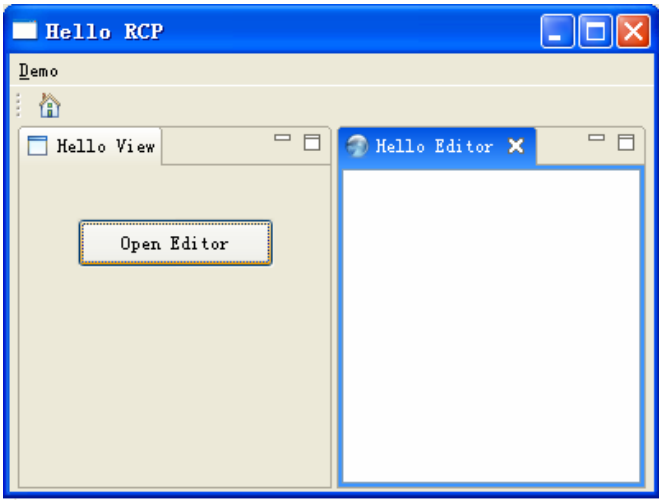
这里的 getName 是返回当前 input 的名字，比如文件名，getToolTipText 是返回当前 input 的提示，当鼠标移动到 editor 的顶部标签上时就会显示。这两个方法都不能返回 null。

好了现在一切准备工作已经就绪，就等打开 Editor 了。我们通过 HelloView 上面的那个按钮来打开 HelloEditor，每按一次按钮打开一个 Editor。首先更改按钮的 Text 属性为“Open Editor”，然后为按钮添加事件，事件代码如下：

```
openEditorButton.addSelectionListener(new SelectionAdapter()
{
    public void widgetSelected(SelectionEvent e)
    {
        try
        {
            getViewSite().getWorkbenchWindow().getActivePage()
                .openEditor(new HelloEditorInput(), HelloEditor.ID);
        }
        catch (Exception ex)
        {
            System.out.println(ex);
        }
    }
});
```

关键代码就一行，就是那个 openEditor 方法，这个函数的两个参数，前面的代表资源，后面的代表工具，就是用后面的 Editor 去打开前面的内容。提醒一下，这里的 Editor ID 必须和 plugin.xml 文件里面的一样，否则会找不到合适的 Editor。可以看到 openEditor 是 Page 的方法，这也和我前面介绍的 Eclipse 的组件结构一致，View 和 Editor 是属于某个 page 的。其实在配置 Perspective 的时候也能看出这个问题，我们都是用 IPageLayout

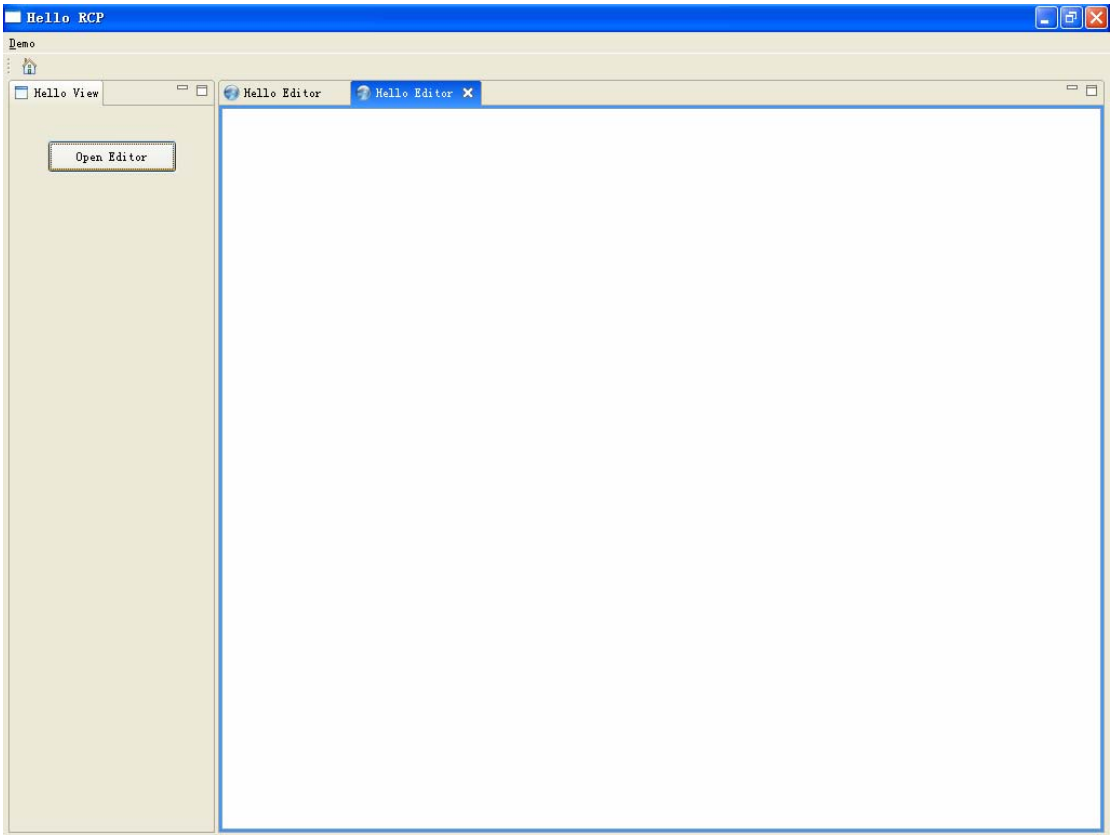
来布置界面。下面运行一下程序，看一下运行结果。




可能你会感觉不太协调，那么我们把窗口默认最大化（似乎能好看点），这就需要修改 ApplicationWorkbenchWindowAdvisor 类中的 postWindowOpen 函数：

```
public void postWindowOpen()
{
    this.getWindowConfigurer().getWindow().getShell().setMaximized(true);
}
```

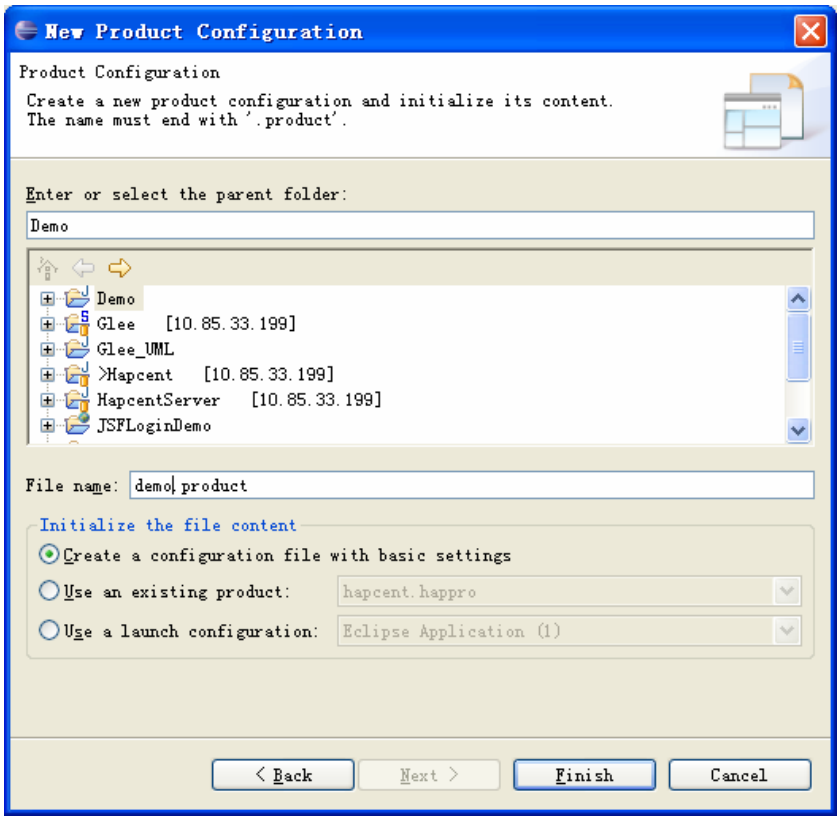
如果你以前接触过 SWT/JFace 的话这段代码应该很熟悉吧，就算没接触过，也应该能大概明白什么意思，所以我不解释了。这句话一定要放在 postWindowOpen 里面，因为在 preWindowOpen 的时候还没有 Window 存在，所以没法设置。我们来看一下最终的运行结果：



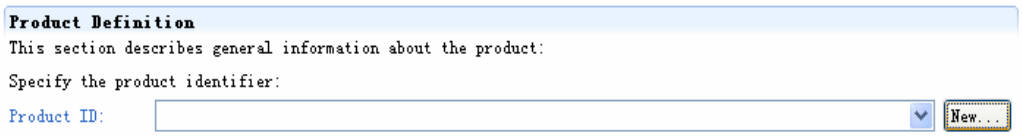

下面我介绍一下如何导出/发布 RCP 程序，使程序能够独立运行。

十、导出 RCP 程序

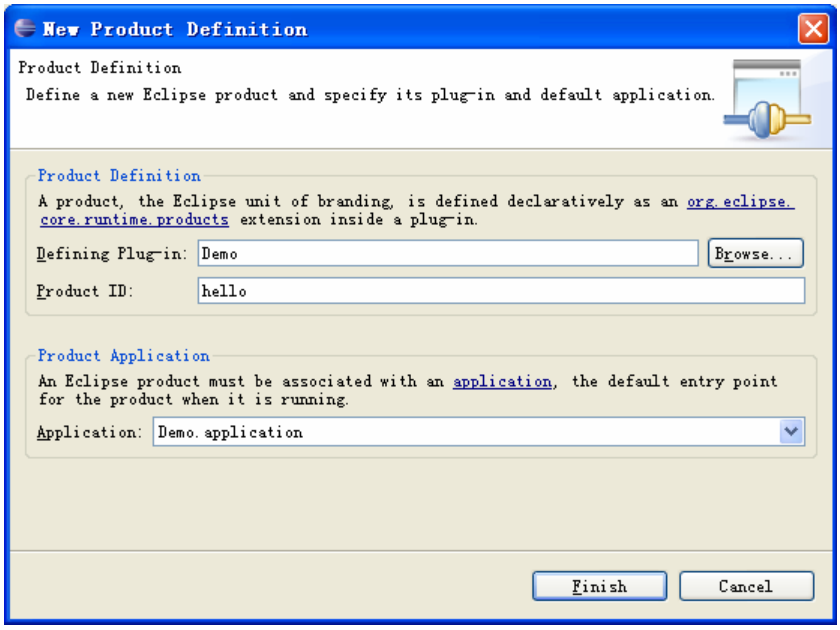
从 Eclipse 3.0 开始，RCP 程序的配置和导出就变得相当方便。RCP 程序又称为 Eclipse Product，我们先为 Demo Project 新建一个 Product 配置文件。右键 Demo Project，选择 New->Other->Product Configuration，然后选择 Demo 文件夹，File name 输入 demo.product，点击 Finish



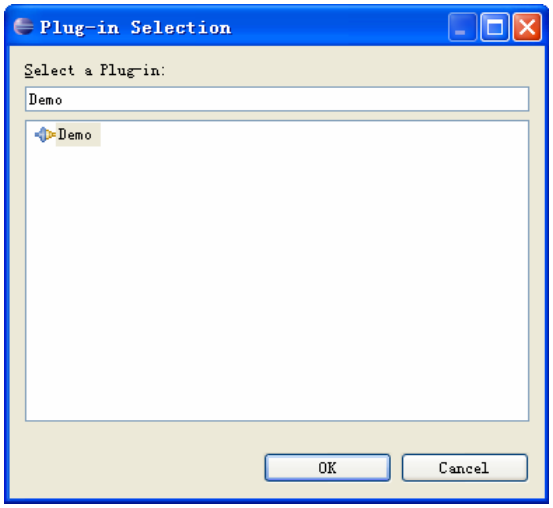

接下来我们要新建一个 Product Id，点击 New



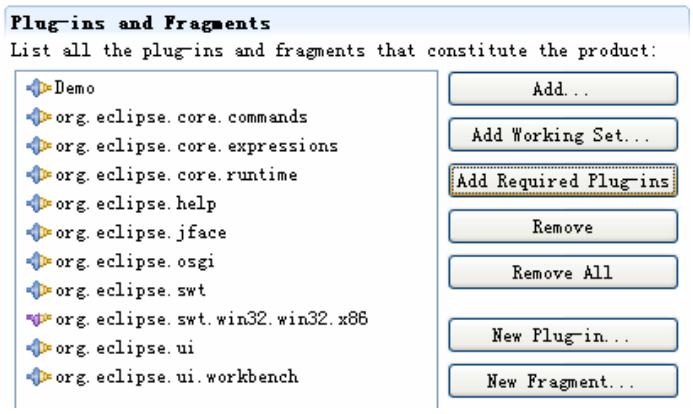

然后点击 Browse 选择 Demo，并输入 Product ID 为 hello，点击 Finish




Overview 标签页的 Product Definition 部分，输入 Product Name 为 Hello Demo。在 Testing 部分点击一下“Synchronize”（最好点一下，有时候不点也没问题），这是为了让我们的 Product 配置和插件的配置保持一致。然后切换到 Configuration 标签页，点击 Add，输入 Demo，点击 OK

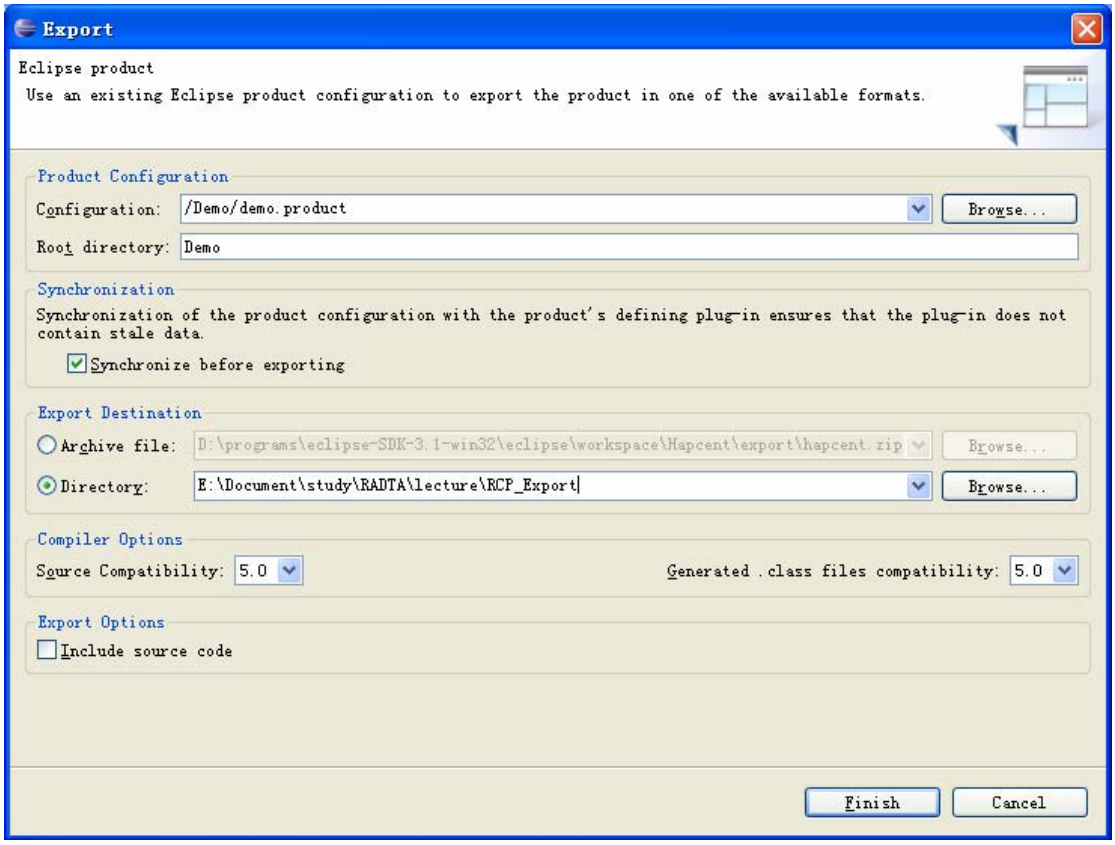



然后点击“Add Required Plug-ins”，这一步是添加我们的 Product 所用到的其他插件，这样才可以脱离 Eclipse 运行。



然后进入 Branding 标签页，这一页是设置 Product 的图标，程序名字等等，可以不设置，系统会自动使用默认值，我这里输入一个 Launcher Name 为 HelloDemo。

现在我们已经完成了导出的配置工作，现在来导出我们的 RCP 程序。进入 Demo.product 的 Overview 标签页，在右下角 Exporting 部分点击“Eclipse Product export wizard”，输入 Root Directory 和 Export Destination，点击 Finish。




到这里为止，我的例子就结束了，你可以到你 export 的地方去找 HelloDemo.exe（文件名根据设置不一样而不一样），然后双击打开就可以看到效果了。我的这个例子只是演示一个 RCP 程序开发的流程，中间有很多细节都省略了，真正做 RCP 开发的时候还要考虑很多问题。如果在开发中遇到什么困难我建议去 http://www.eclipse.org 寻找帮助，那里有很多文章，新闻组的人气也很旺，牛人特别多。