ORIGIN
Product Lifecycle Management
Consulting Service

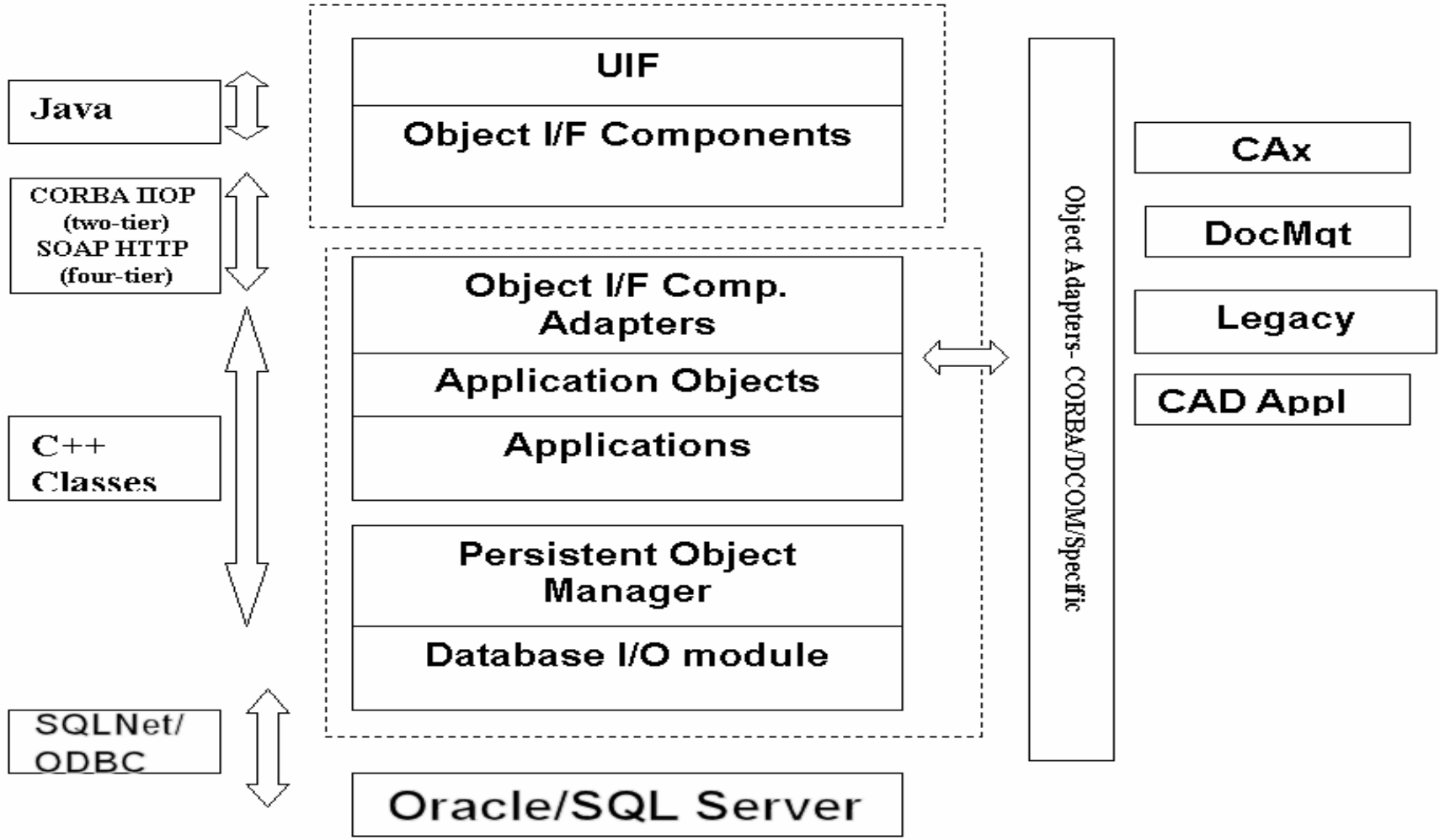
Teamcenter客户化开发（一）

目录

- Teamcenter体系结构
- RCP开发原理
- AWT/Swing和SWT/Jface介绍
- Teamcenter现有类结构
- 开发环境安装与部署
- 开发一个简单插件工程

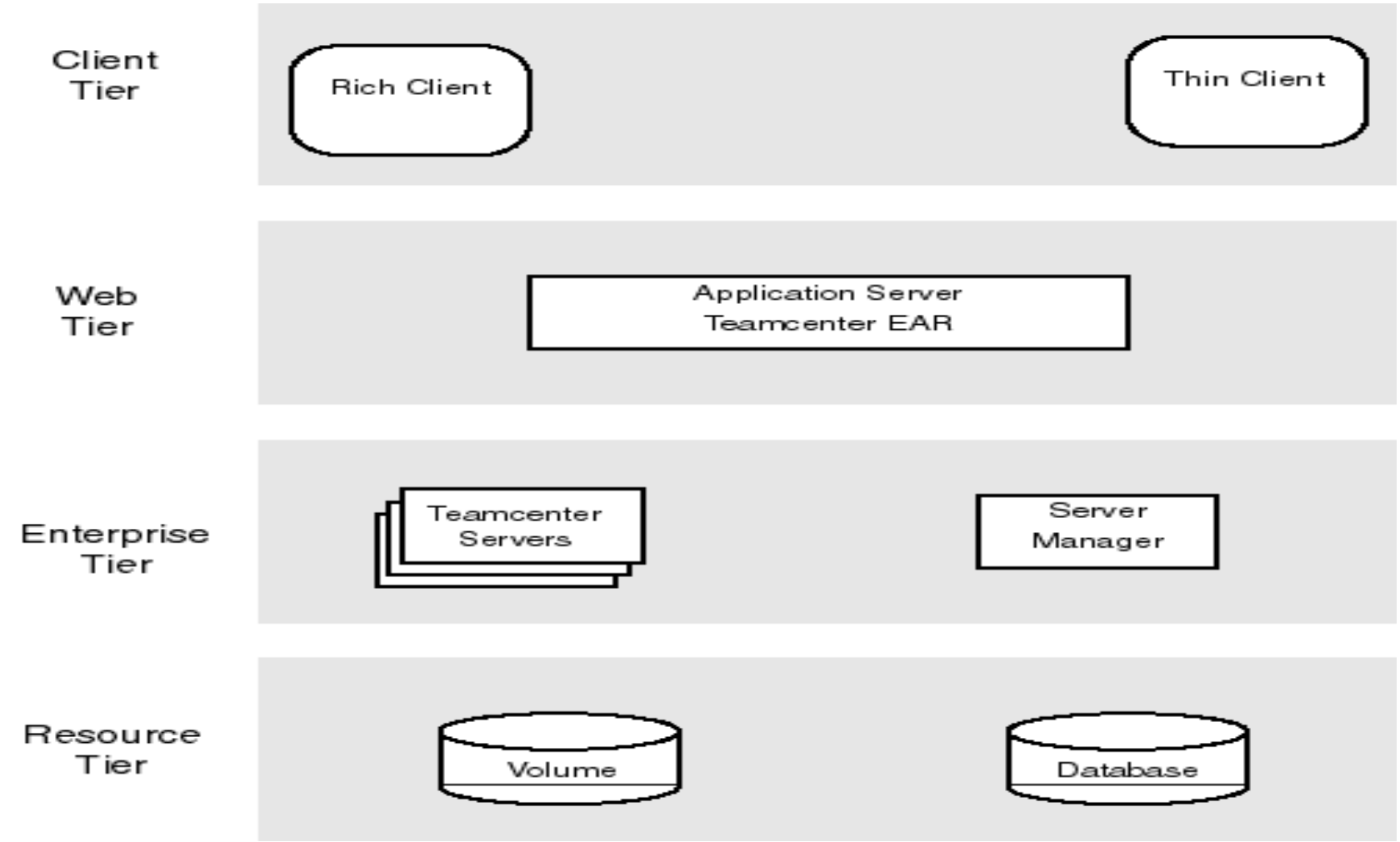
Teamcenter体系结构

Ø两层结构



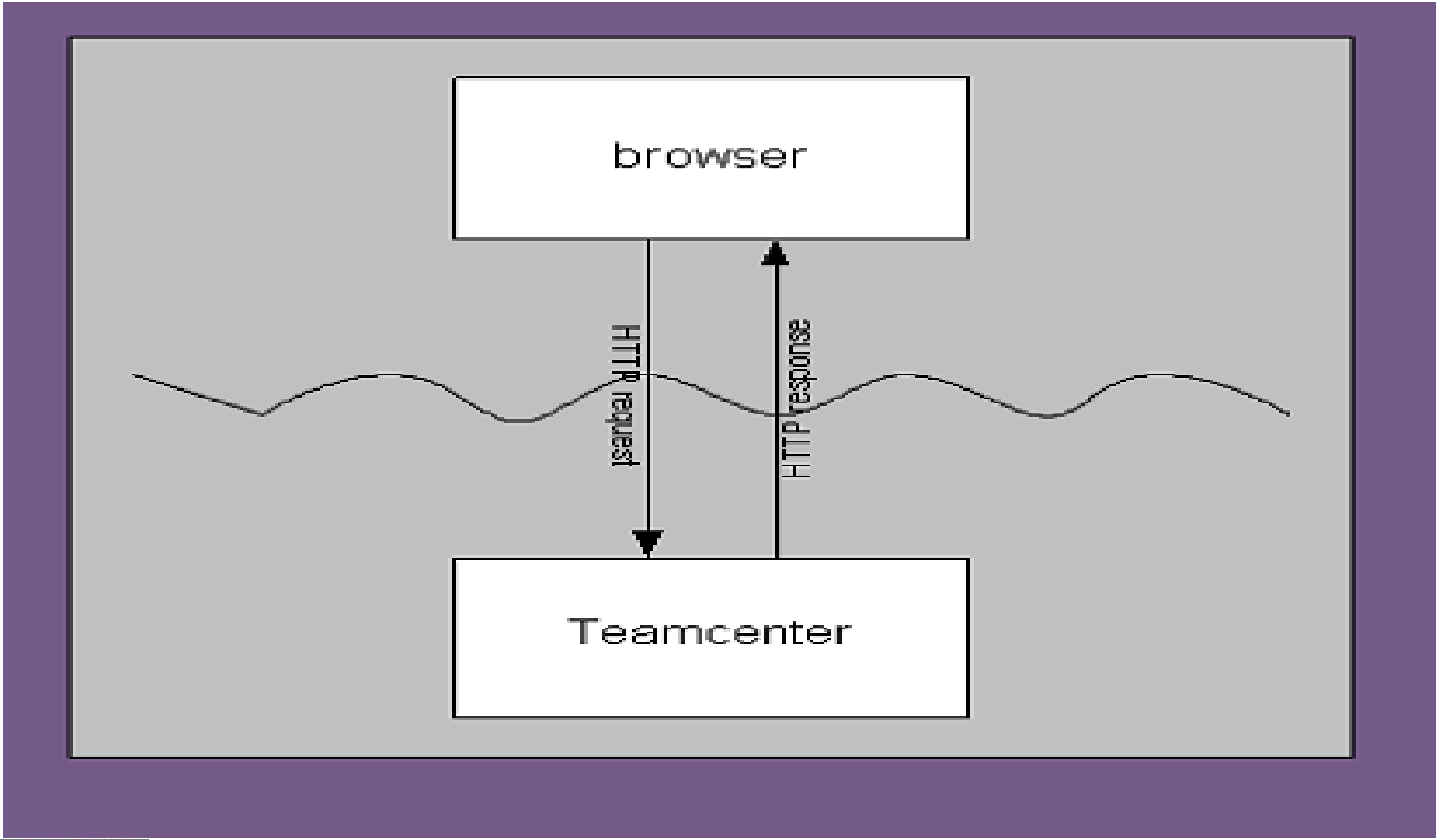
Teamcenter体系结构

Ø四层结构



Teamcenter体系结构

Ø瘦客户端结构



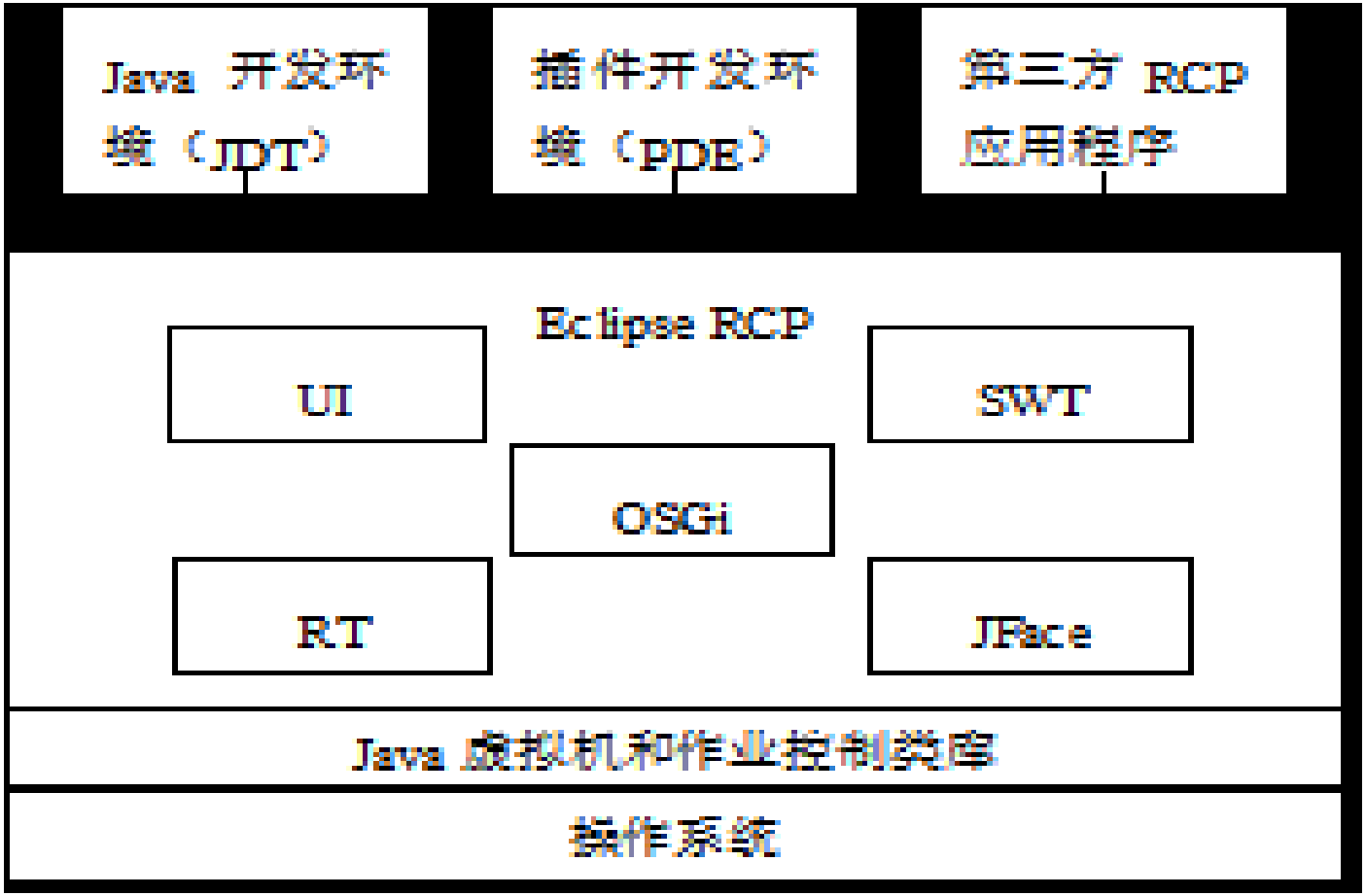
RCP开发原理

ØRCP简介

RCP是Rich Client Platform的缩写。其实就是利用Eclipse核心平台和一些有用的插件。进行应用系统的构建。只要新的代码符合插件的代码结构，将代码放入工作空间后，Eclipse平台会自动加载新代码，构成新的Eclipse平台。

RCP开发原理

ØRCP体系结构

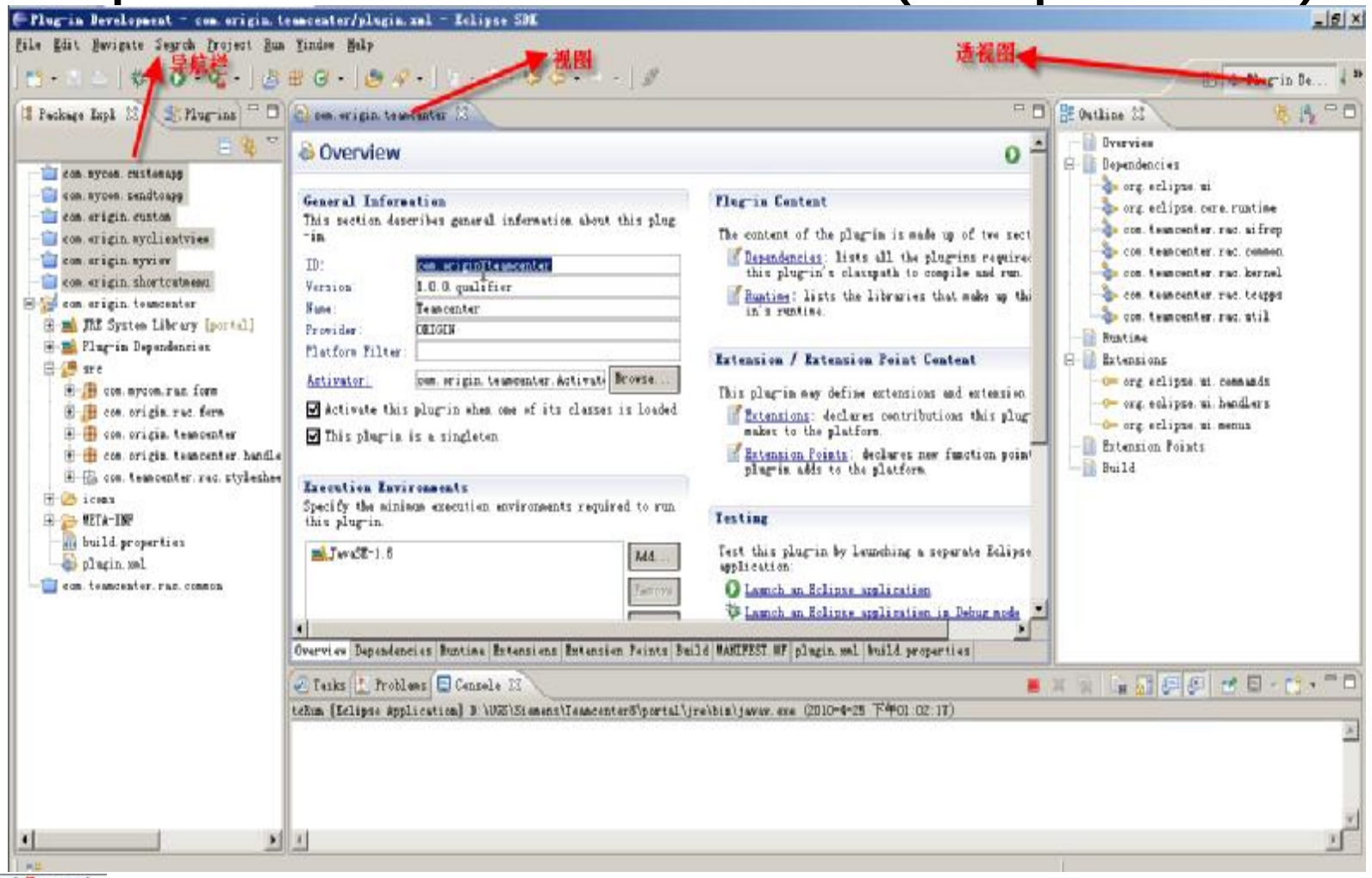


RCP开发原理

- Eclipse平台在文件系统中的目录结构

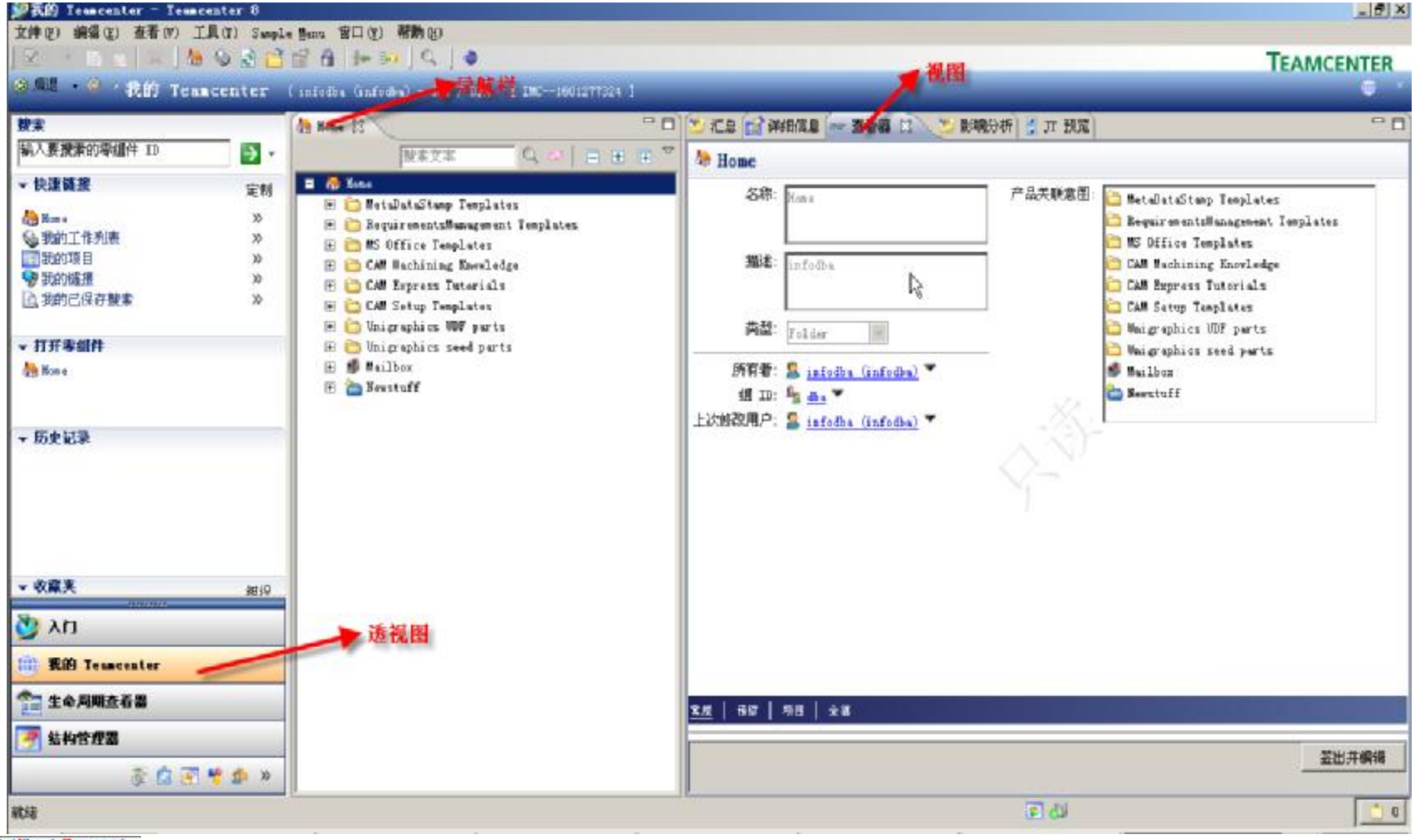
RCP开发原理

- Eclipse平台应用运行的界面(Eclipse为例)



RCP开发原理

- RCP应用运行的界面(Teamcenter为例)



RCP开发原理

ØRCP中视图和透视图说明

RCP中的每个视图都对应每个应用中的各个界面构件。每个透视图对应一个应用组合，如我的Teamcenter，结构管理器等。

RCP开发原理

- Eclipse RCP开发具有的优点：

组件化：基于Eclipse的系统设计由被称为plug-ins的插件构成，可以通过扩展点进行配置，也可以被不同应用程序共享。

便利性:Eclipse RCP对各个平台下的产品包装提供了强有力的支持，其开发的RCP甚至可以在嵌入式设备、掌上电脑上运行。Sun公司对Java的口号是“write once, run everywhere”，Eclipse也可以说是“RCP run everywhere”。

智能安装和升级：Eclipse提供了专门的Update组件，可以实现通过HTTP、Web站点、复制等多种方式进行安装和更新，一扫早期富客户端应用 部署升级的麻烦。

RCP开发原理

可扩展性：Eclipse基于插件进行扩展的思想使得用户可以方便地搭建各种规模、类型和用途的应用程序。按照Eclipse官方的说法，Eclipse RCP一开始就被设计为可扩展的。

本地观感及使用体验：Eclipse为各种操作系统提供了本地图形接口包。当RCP运行时，Eclipse首先直接调用本机窗口组件，只有没有本机所需组 件时才进行模拟。无论RCP在哪种操作系统上运行，都可以保持与本机一致的外观和行为。一个设计优良的富客户端，可以提供诸如拖曳操作、剪切板、导航等 UI元素。UI设计者也可以利用各种界面工具，轻松设计出完美的用户界面。

AWT/Swing和SWT/Jface介绍

ØAWT 概述

Abstract Windows Toolkit（AWT）是最原始的 Java GUI 工具包。AWT 的主要优点是，它在 Java 技术的每个版本上都成为了一种标准配置，包括早期的 Web 浏览器中的 Java 实现；另外它也非常稳定。这意味着我们不需要单独安装这个工具包，在任何一个 Java 运行环境中都可以使用它，这一点正是我们所希望的特性。

AWT 是一个非常简单的具有有限 GUI 组件、布局管理器和事件的工具包。这是因为 Sun 公司决定为 AWT 使用一种最小公分母（LCD）的方法。因此它只会使用为所有 Java 主机环境定义的 GUI 组件。最终的结果非常不幸，有些经常使用的组件，例如表、树、进度条等，都不支持。对于需要更多组件类型的应用程序来说，我们需要从头开始创建这些组件。 这是一个很大的负担。

AWT/Swing和SWT/Jface介绍

ØSwing 概述

Java Swing 是 Java Foundation Classes（JFC）的一部分，它是试图解决 AWT 缺点的一个尝试。在 Swing 中，Sun 开发了一个经过仔细设计的、灵活而强大的 GUI 工具包。不幸的是，这意味着我们又要花一些时间来学习 Swing 了，对于常见的情况来说，Swing 有些太复杂了。

Swing 是在 AWT 组件基础上构建的。所有 Swing 组件实际上也是 AWT 的一部分。Swing 使用了 AWT 的事件模型和支持类，例如 Colors、Images 和 Graphics。Swing 组件、布局管理器以及事件总结如下。正如您可以看到的一样，这些组件集比 AWT 提供的组件集更为广泛，与 SWT 组件集相比也毫不逊色。

AWT/Swing和SWT/Jface介绍

ØSWT 概述

与 AWT 的概念相比，SWT 是一个低级的 GUI 工具包。JFace 是一组用来简化使用 SWT 构建 GUI 的增强组件和工具服务。SWT 的构建者从 AWT 和 Swing 实现中学习了很多经验，他们试图构建一个集二者优点于一体而没有二者的缺点的系统。从很多方面来说，他们已经成功了。

SWT 也是基于一个对等体实现的，在这一点上它与 AWT 非常类似。它克服了 AWT 所面临的 LCD 的问题，方法如下：定义了一组控件，它们可以用来构建大部分办公应用程序或开发者工具，然后可以按照逐个主机的原则，为特定主机所没有提供的控件创建模拟 控件（这与 Swing 类似）。对于大部分现代主机来说，几乎所有的控件都是基于本地对等体的。这意味着基于 SWT 的 GUI 既具有主机外观，又具有主机的性能。这样就避免了使用 AWT 和 Swing 而引起的大部分问题。特定的主机具有一些低级功能控件，因此 SWT 提供了扩充（通常是模拟的）版本（通常使用 “C” 作为名字中的第一个字母），从而可以产生更一致的行为。

AWT/Swing和SWT/Jface介绍

在对等体工作方式上，SWT 与 AWT 不同。在 SWT 中，对等体只是主机控件上的一些封装程序而已。在 AWT 中，对等体可以提供服务来最小化主机之间的差异（就是在这里，AWT 碰到了很多行为不一致的问题）。这意味着 SWT 应用程序实际上就是一个主机应用程序，它必然会全部继承主机的优点和缺点。这还意味着 SWT 不能完全实现 WORE 的目标；它更像是一种 WOTE 解决方案。这就是说，SWT 尽管不如 Swing 那么优秀，但是它在创建可移植解决方案方面是很杰出的。

AWT/Swing和SWT/Jface介绍

在大部分情况中，都是使用 Swing 与结合了 JFace 的 SWT 一起构建GUI的。通常来说，每个工具包都非常完整且功能强大，足以构建功能完善的 GUI，但是 Swing 通常要比单独使用 SWT（不使用 JFace 时）更好。Swing 具有内嵌于 Java 技术的优点，是完全可移植的，无可争议地是一种更好的架构。Swing 也具有高级图形应用程序所需要的优点。SWT 具有可以作为本地应用程序实现的优点，这可以提高性能，并利用基于 SWT 的 GUI 来实现本地兼容性。

如果只为一种平台来开发系统，那么 SWT 就具有主机兼容性方面的优点，包括与主机特性的集成，例如在 Windows 上对 ActiveX 控件的使用。

Teamcenter2007以上都是采用SWT与Swing结合进行客户端构建的。

Ø 客户化开发中常用到的插件

① com.teamcenter.rac.aifrcp

Teamcenter 基础客户化插件，一些主要的接口以及抽象类，入口类等都在该插件中进行了定义，如：

AbstractAIFApplication

AbstractAIFCommand

AbstractAIFDialog

AbstractAIFOperation

AIFDesktop

AIFPortal

AbstractAIFAction

② com.teamcenter.rac.common

Teamcenter的一些动作和菜单都在该插件包中进行了定义，如菜单栏，工具栏，以及右键菜单等。首先要说的是，界面上所有菜单，以及一些公共组件及Form的顶级实现。一般的菜单动作都在com.teamcenter.rac.common.actions中进行了定义，如新建Item为例：

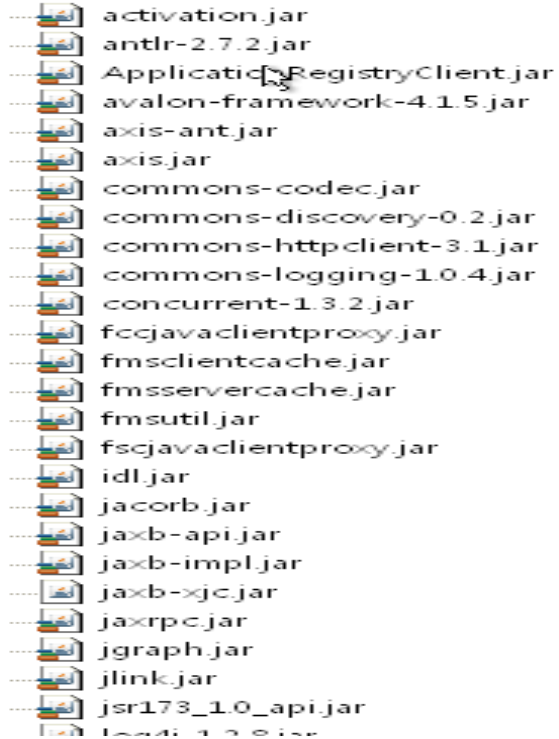
1. 在该包中建立NewItemAction类，并集成AbstractAIFAction类，实现 public void run()方法。
2. 在action.properties文件中进行了注册，如下所示：

```
# File -> New -> Item
# ----------------------
newItemAction=com.teamcenter.rac.common.actions.NewItemAction
newItemAction.ICON=images/newitem_16.png
newItemAction.COMMAND=newItemCommand
newItemAction.ACCELERATOR=ctrl pressed T
newItemCommand=com.teamcenter.rac.commands.newitem.NewItemCommand
```

3. 相应的，在该插件中对应一个com.teamcenter.rac.commands.newitem包，创建Item的业务逻辑都在该包中进行了实现。

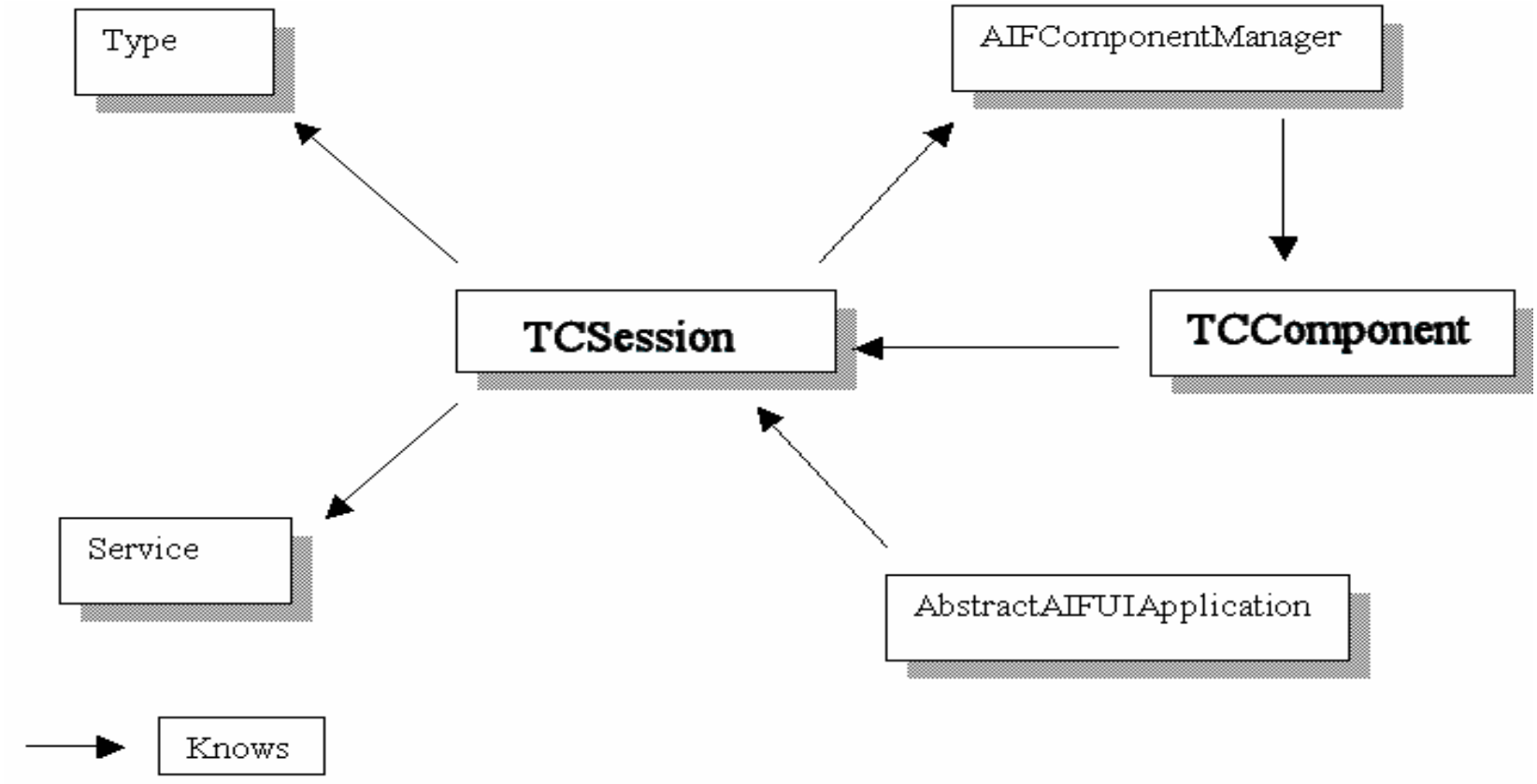
③ com.teamcenter.rac.external

Teamcenter插件中引用到的第三方类大部分都在该插件中进行了集成，可以说该插件为资源性插件。如下图：



③ com.teamcenter.rac.kernal

Teamcenter核心插件包,下图基本上显示了会话的获取方式：



该插件上基本上对业务对象进行了组件的定义，基本上系统中每个业务对象都在该插件中有相应的类去实现。如Folder对象，系统中为文件夹对象，该对象对应的系统类是：

1.TCComponentFolder 该类集成了TCComponent类。扩展定义了 Folder的获取以及和属性的修改方法。基本上所有的业务对象都集成于TCComponent类。相应的我们可以联想到，Item对应的组件类为TCComponentItem；Dataset对应的组件类为TCComponentDataset。

2.一个业务类对应的组件类相应的也会对应一个业务类型类，如Folder对应的业务类型组件类为TCComponentFolderType。该类集成于TCComponentType类。相应的我们可以联想到，Item对应的组件类型类为TCComponentItemType；Dataset对应的组件类型类为TCComponentDatasetType。该类主要扩展定义了相应的业务对象的创建以及另存为等方法。

接下来，我们可以看看怎么通过业务对象对应的类型组件类去创建业务对象，还是以Folder为例：

```
TCComponentFolderType t = (TCComponentFolderType)

session.getTypeComponent("Folder");

TCComponentFolder f = t.create("My Folder Name",

    "My Folder Description", "My Folder Type");
```

④ com.teamcenter.rac.tcapps

Teamcenter中部分应用的基础实现都在该类中进行了实现，这个主要是遗留问题，虽然现在的应用大部分都是以相应的插件进行了区别与分类，但是大部分应用的业务逻辑都在该插件中进行实现，还有一些公共组件等。如：

com.teamcenter.rac.cme.mpp 对MSE应用进行了实现

com.teamcenter.rac.pse 对结构管理器进行了实现

com.teamcenter.rac.querybuilder 对查询构建器进行了实现

com.teamcenter.rac.explorer 对MyTeamcenter进行了实现

⑤ com.teamcenter.rac.util

1.该插件是Teamcenter对大部分Swing组件进行了继承与重写。如：iTextField继承于JTextField；iComboBox类为Teamcenter自己定义的组件。

2. 该插件中还定义了一些公共工具类，如：TcLogger日志控制类，Registry注册控制类等。

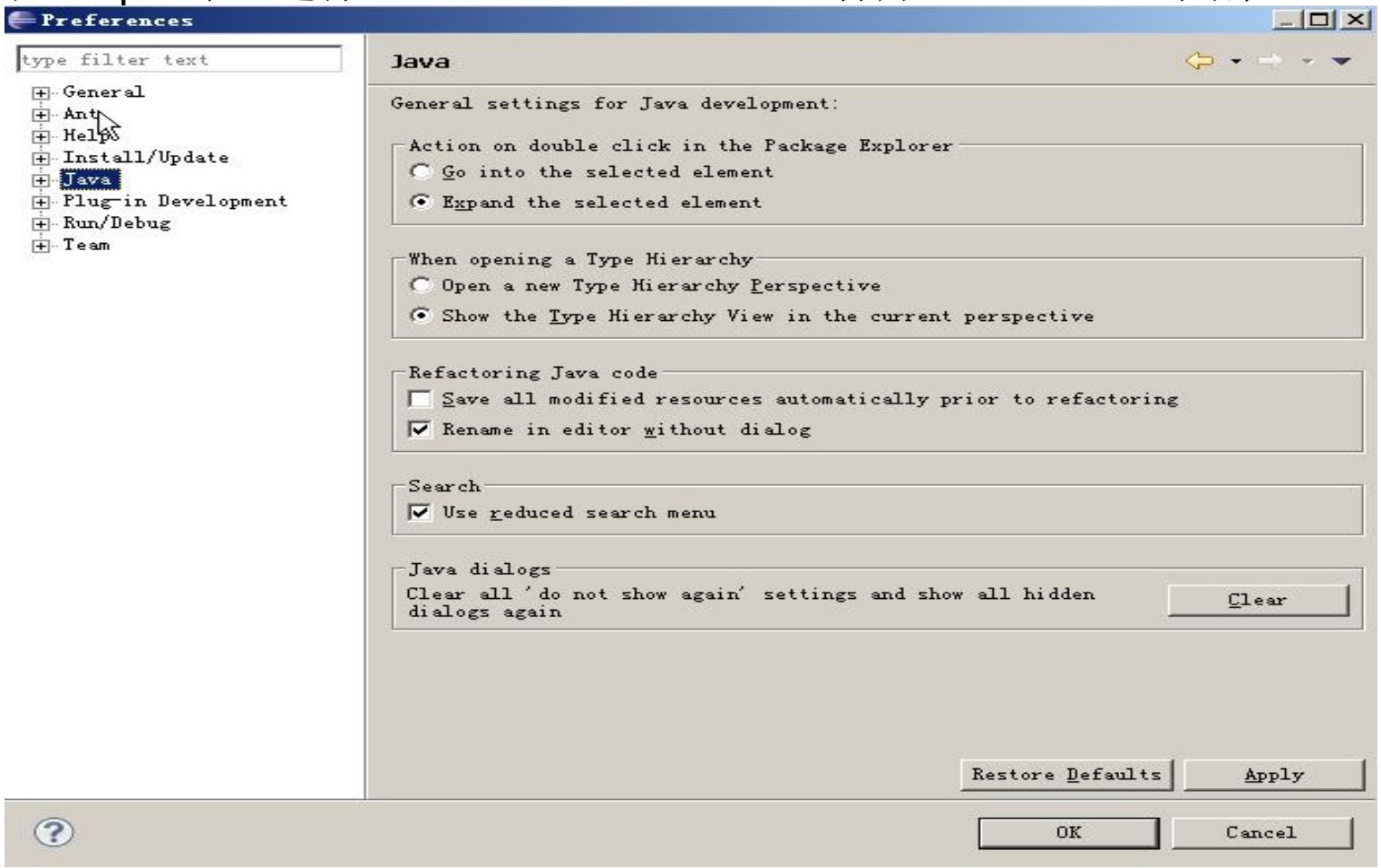
Teamcenter现有类结构讲解

- 客户化开发中常用到Teamcenter组件（待定）

开发环境安装与部署

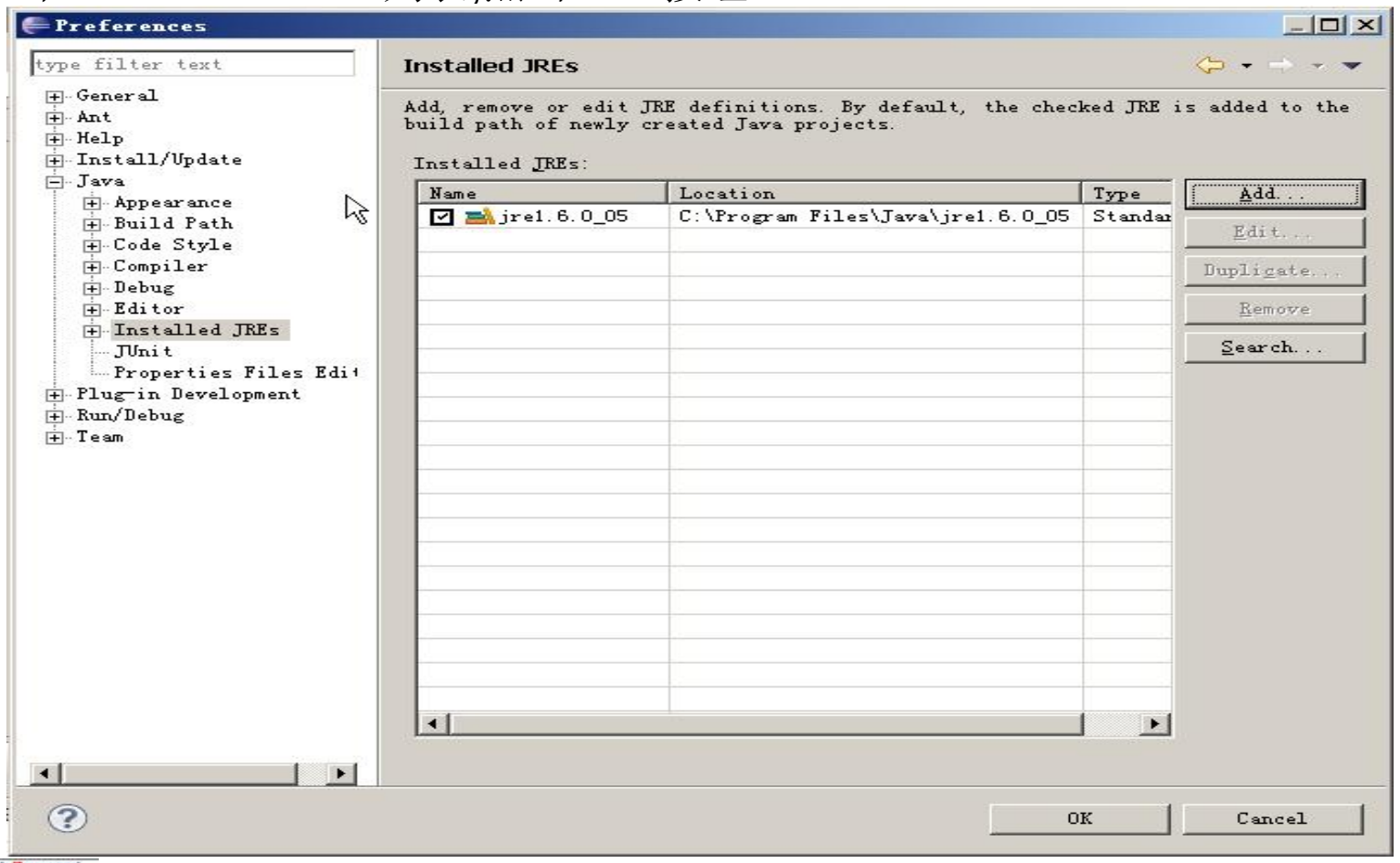
- 开发环境配置

① 在Eclipse中，选择Window→Preferences，打开Preferences对话框。

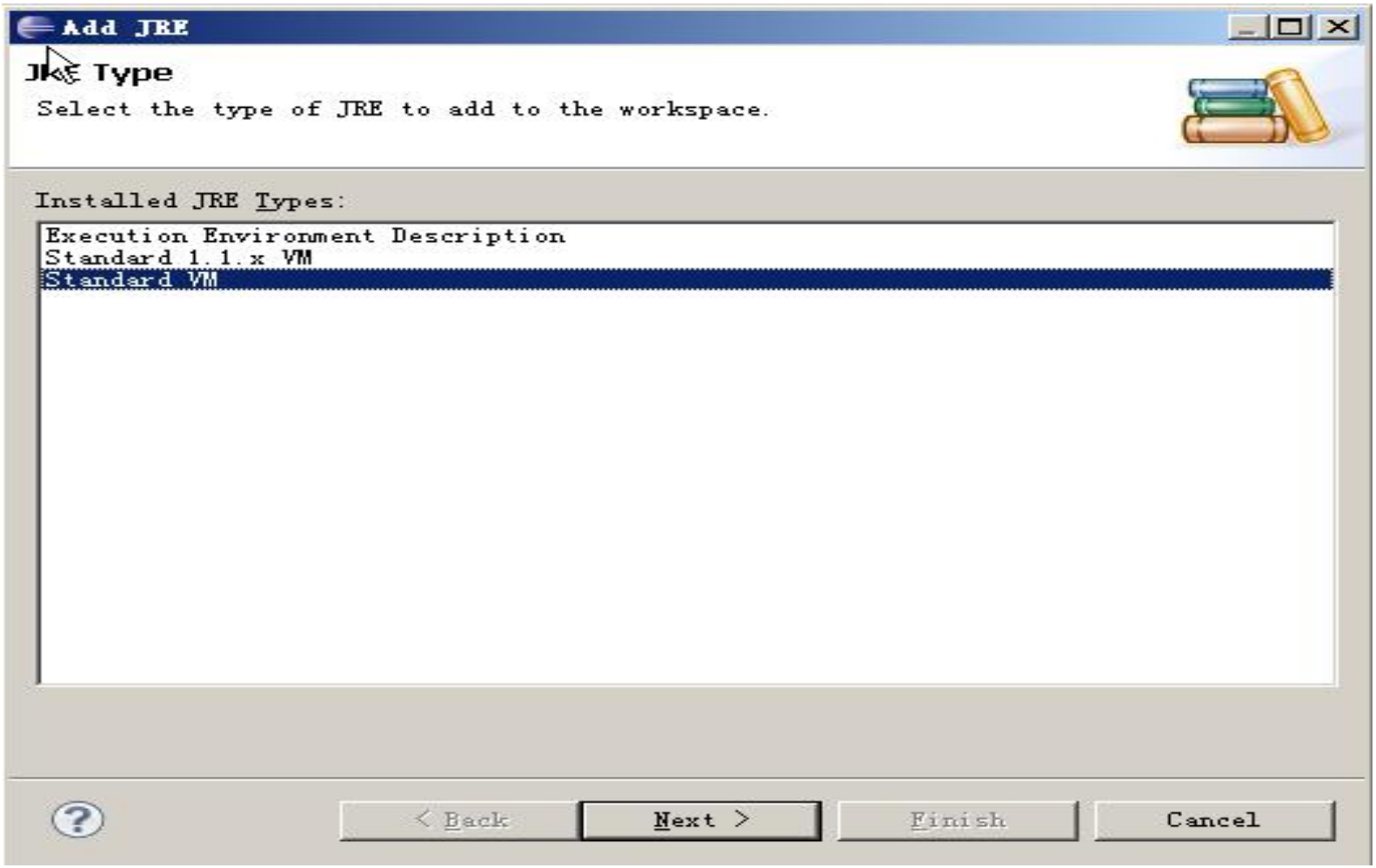


②在左边的树列表中，双击Java，选择Installed JREs。

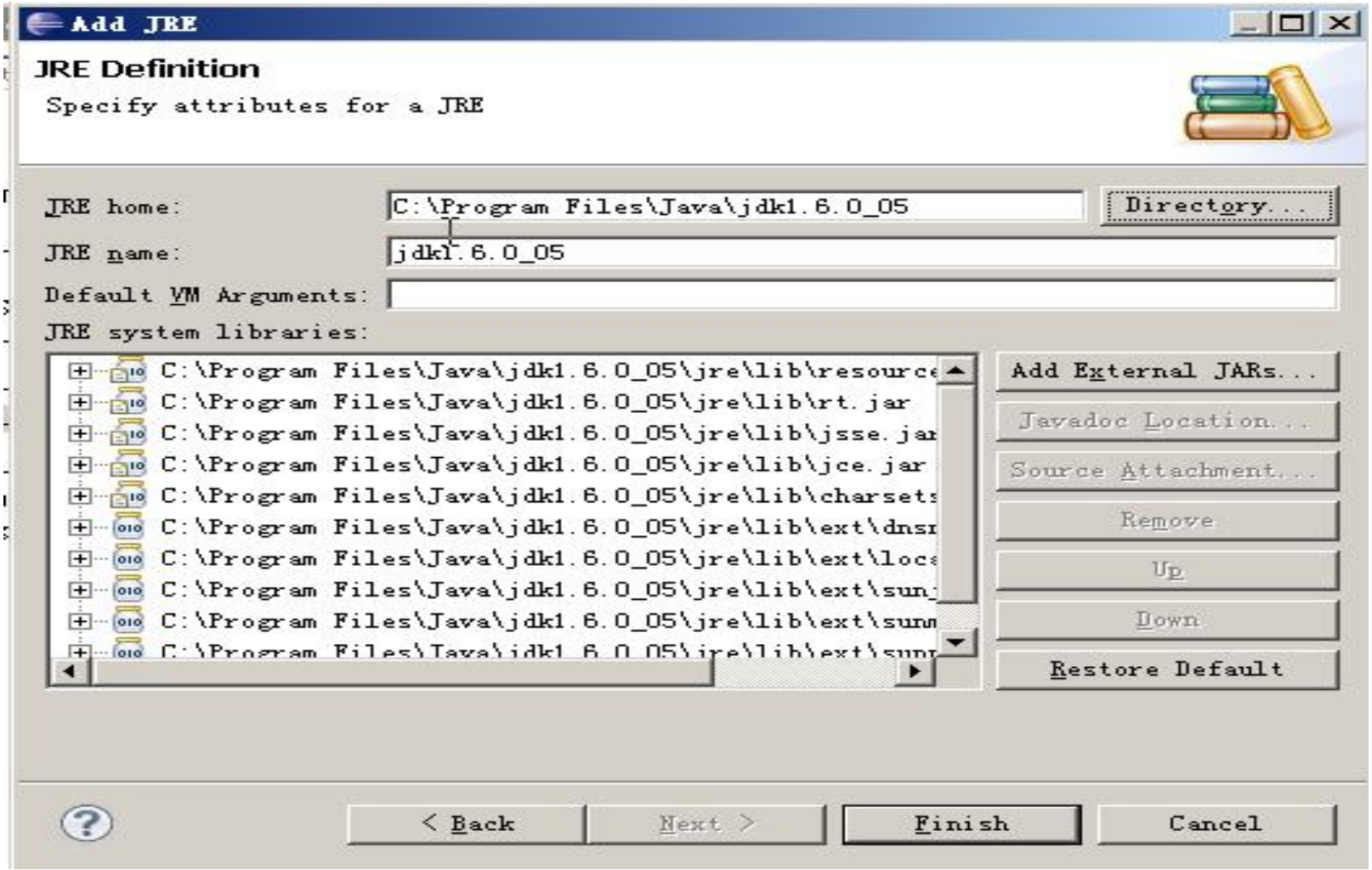
a. 在 Installed JREs 列表,点击Add 按钮。



b.在 JRE Type 对话框, 选择 Standard VM点击 Next.

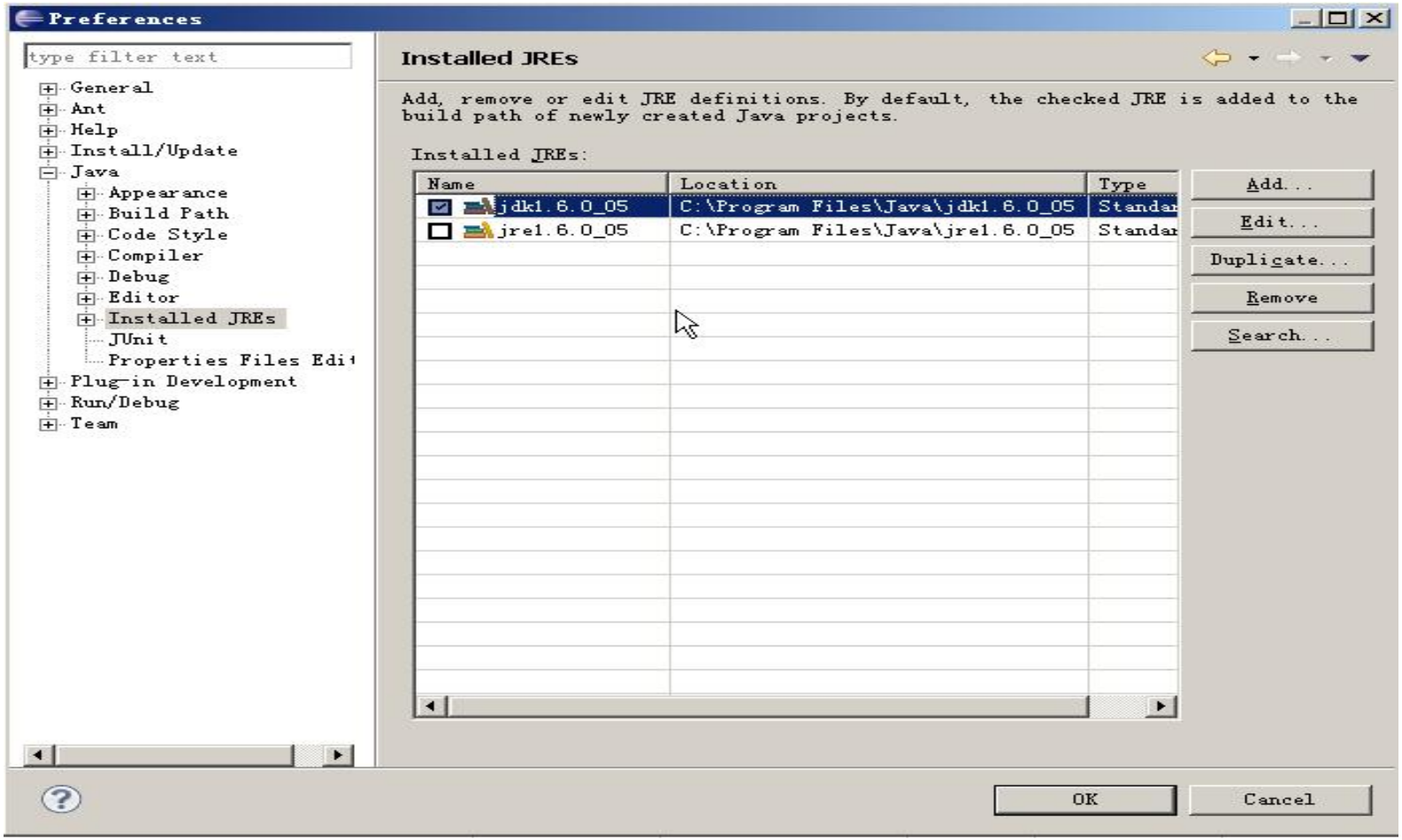


c.在 **JRE Definition** 对话框,指定到JDK的安装目录。

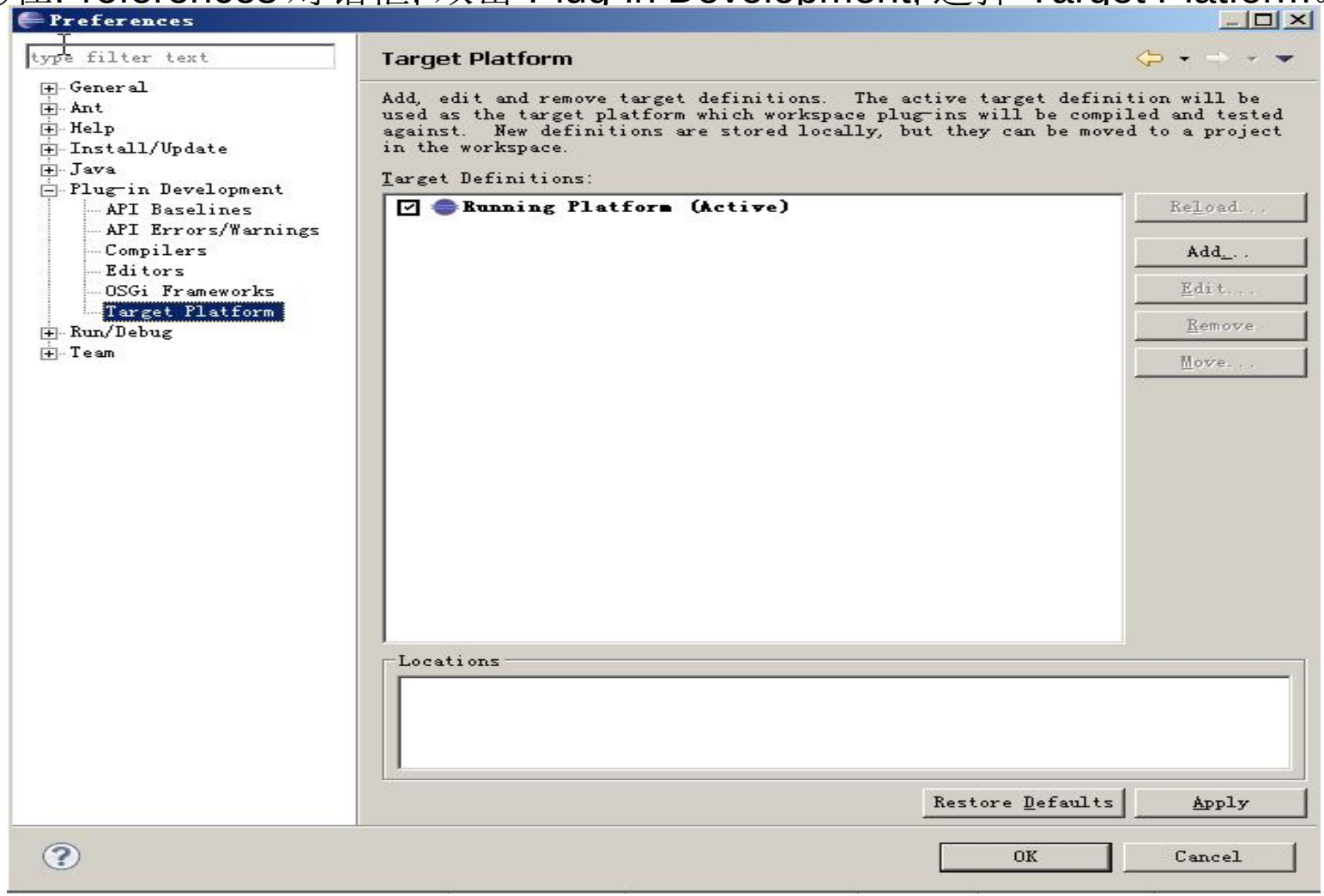


d.点击 **Finish**。

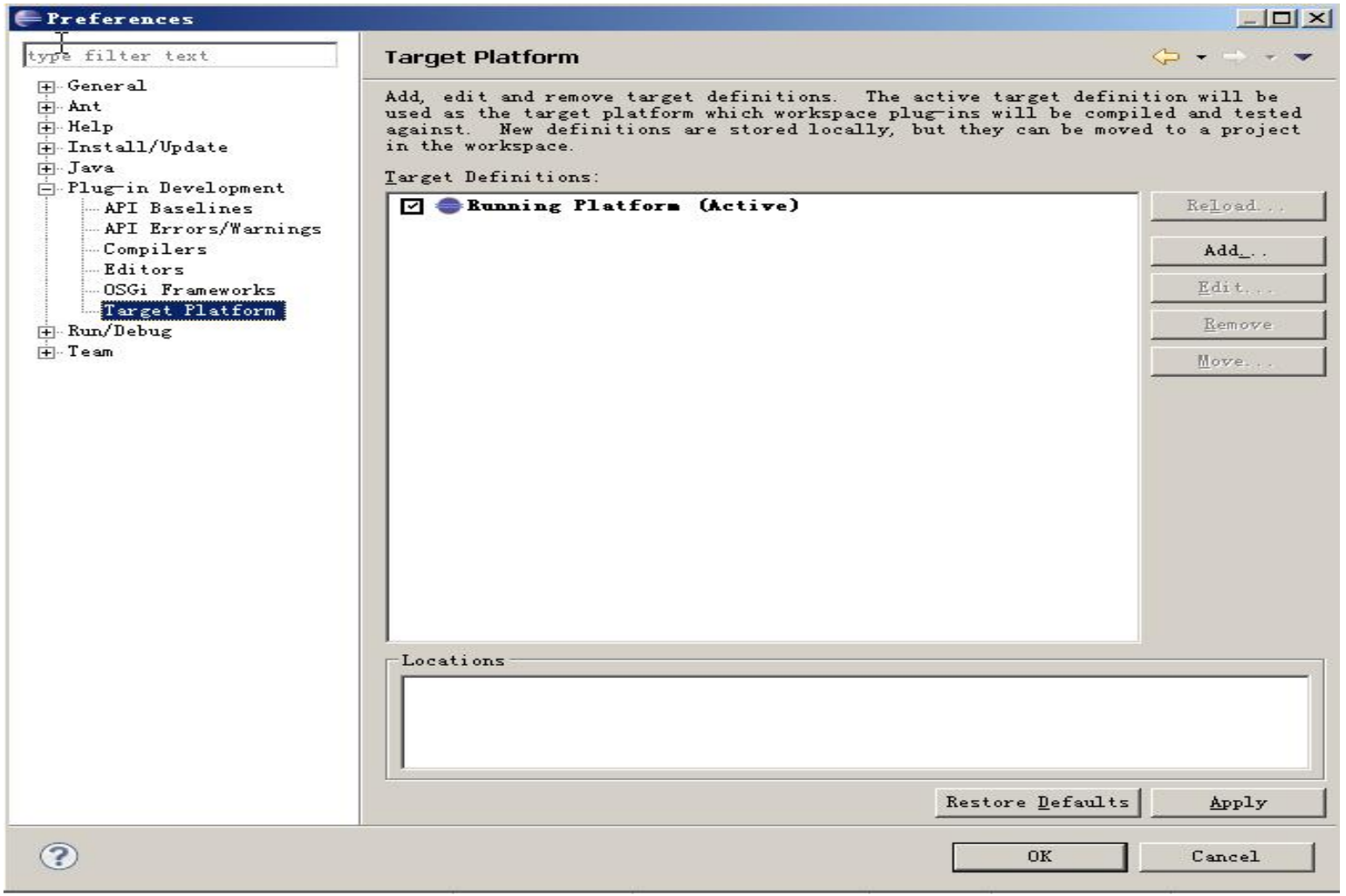
e.在**Preferences**对话框,选择新定义的JRE。



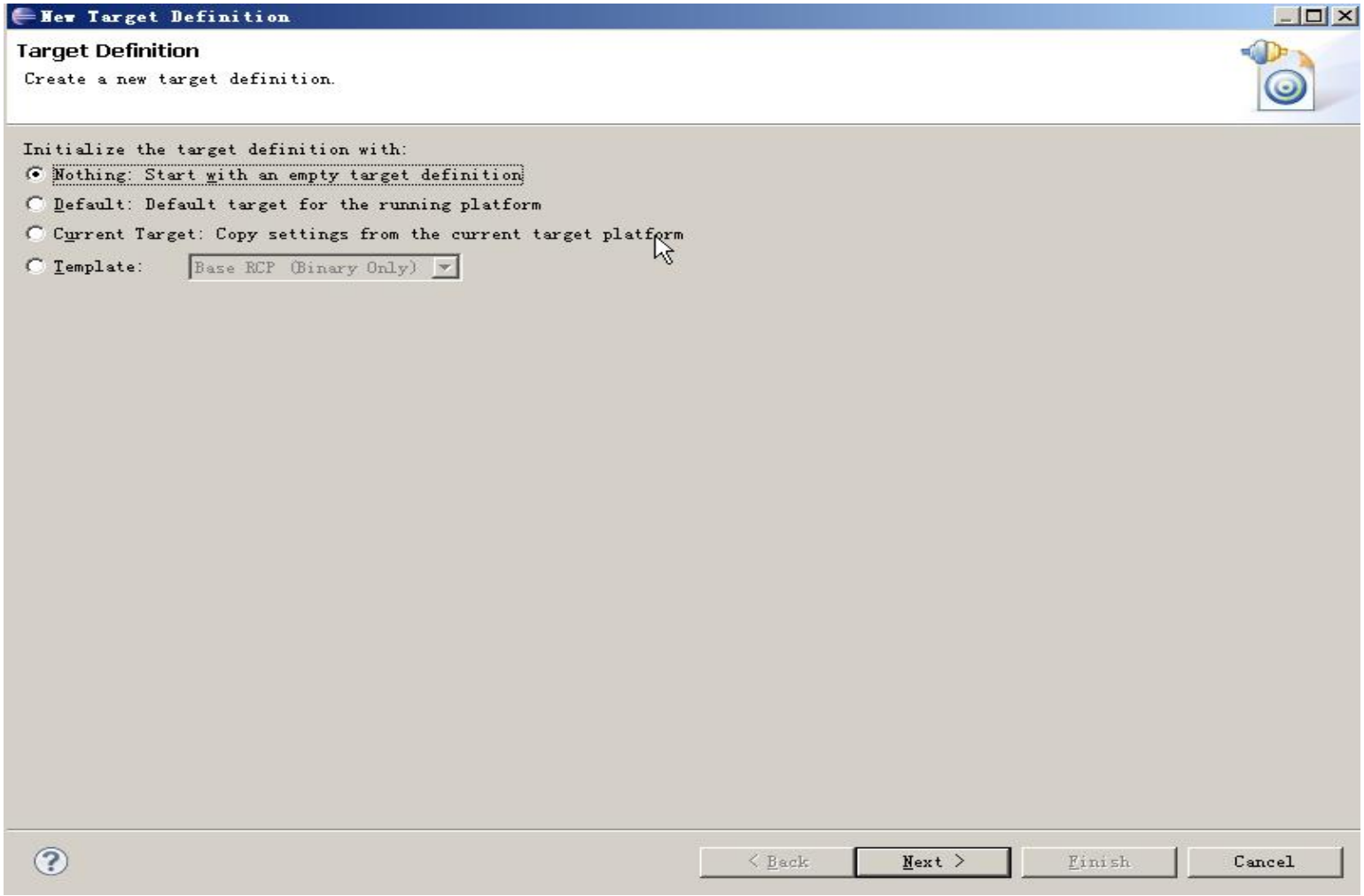
③在Preferences 对话框, 双击 Plug-in Development, 选择 Target Platform。



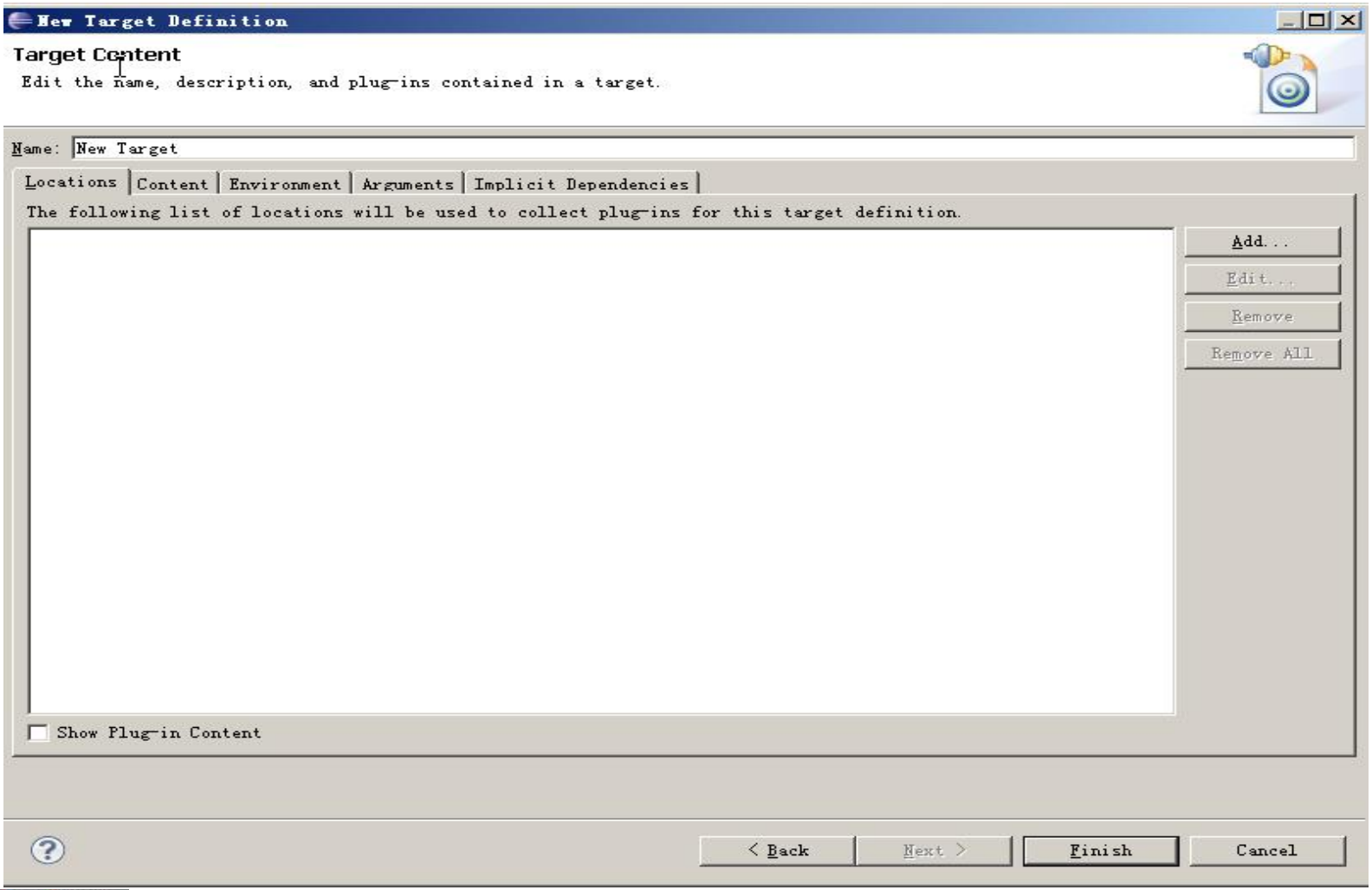
④在 Target Platform 对话框, 点击 Add。



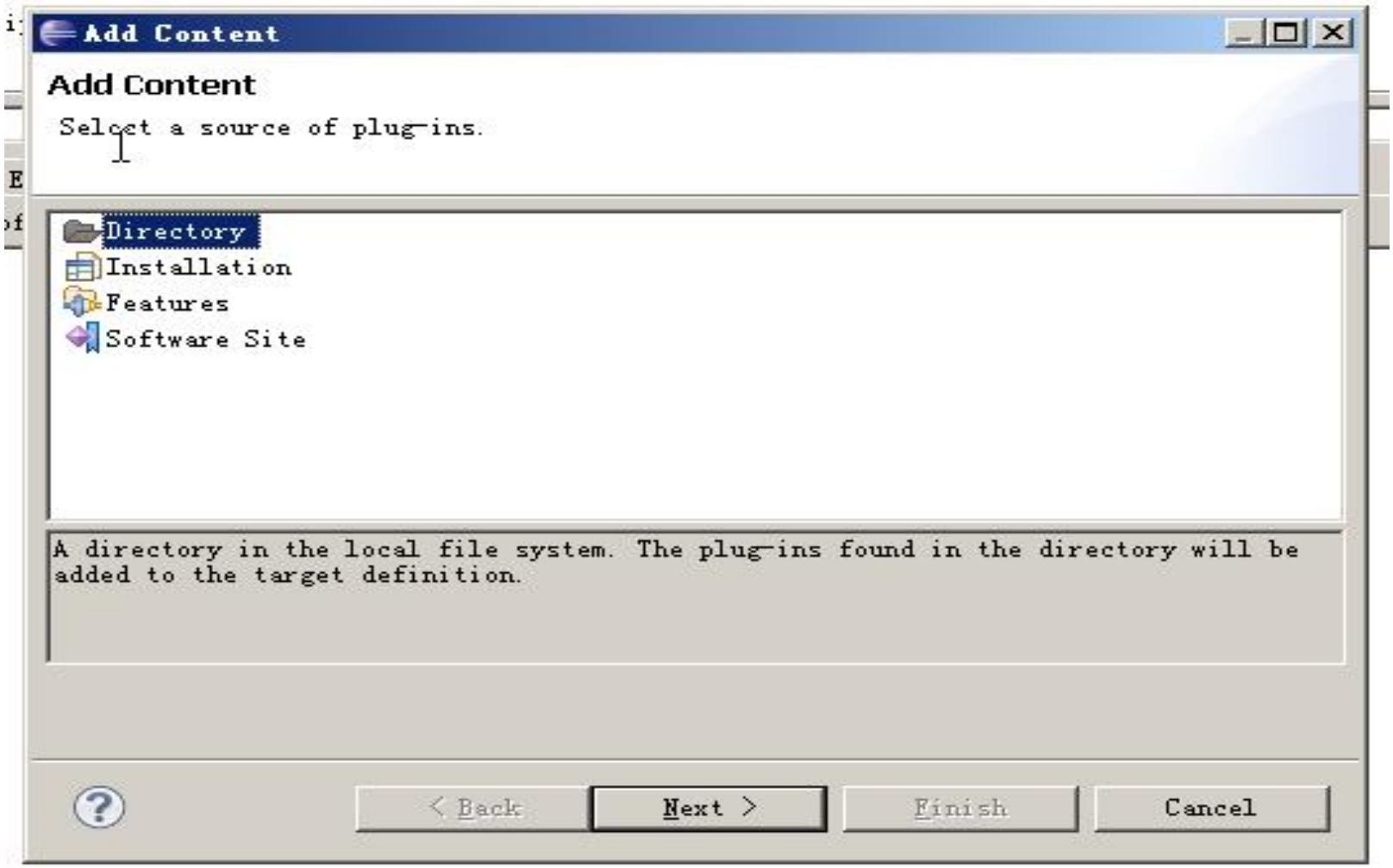
⑤在**Target Definition**框, 确定 **Nothing**是选择的，并点击**Next**。



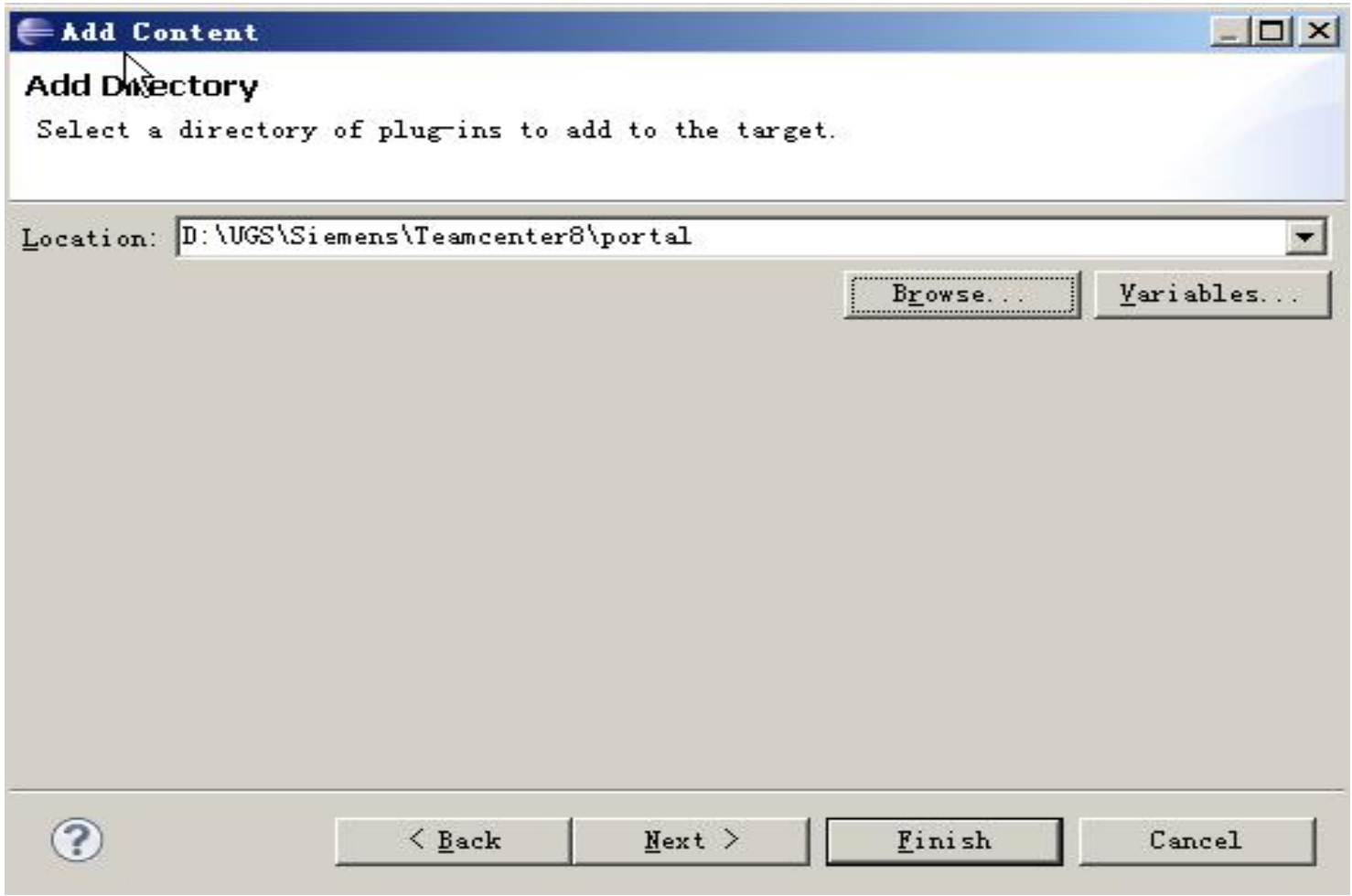
⑥在Target Content 对话框,修改名称为TcTarget并点击 Add 。



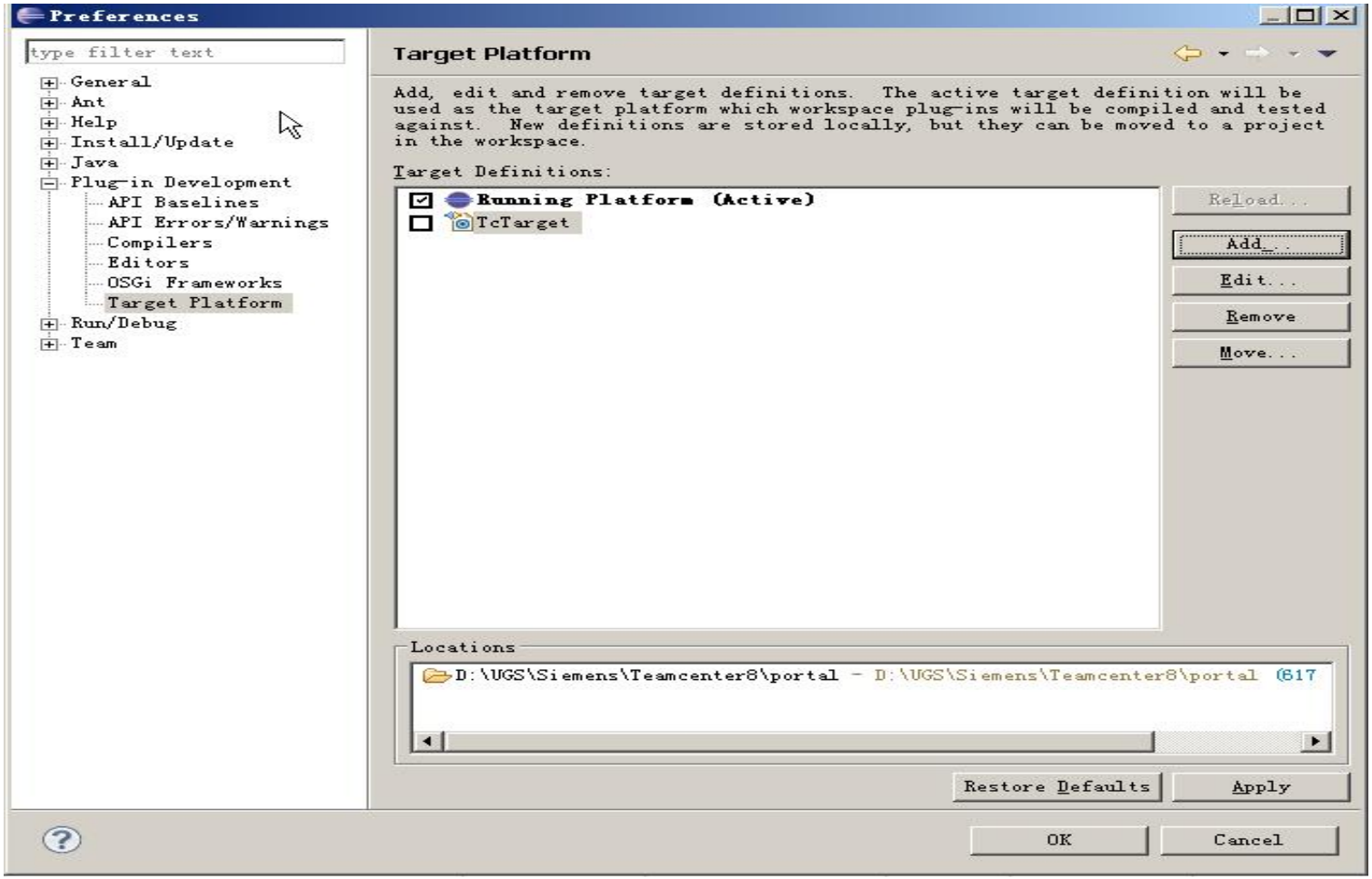
⑦在 Add Content 对话框 选择 Directory 并点击 Next.



⑧指定到TC_ROOT\portal 目录并点击完成Finish。



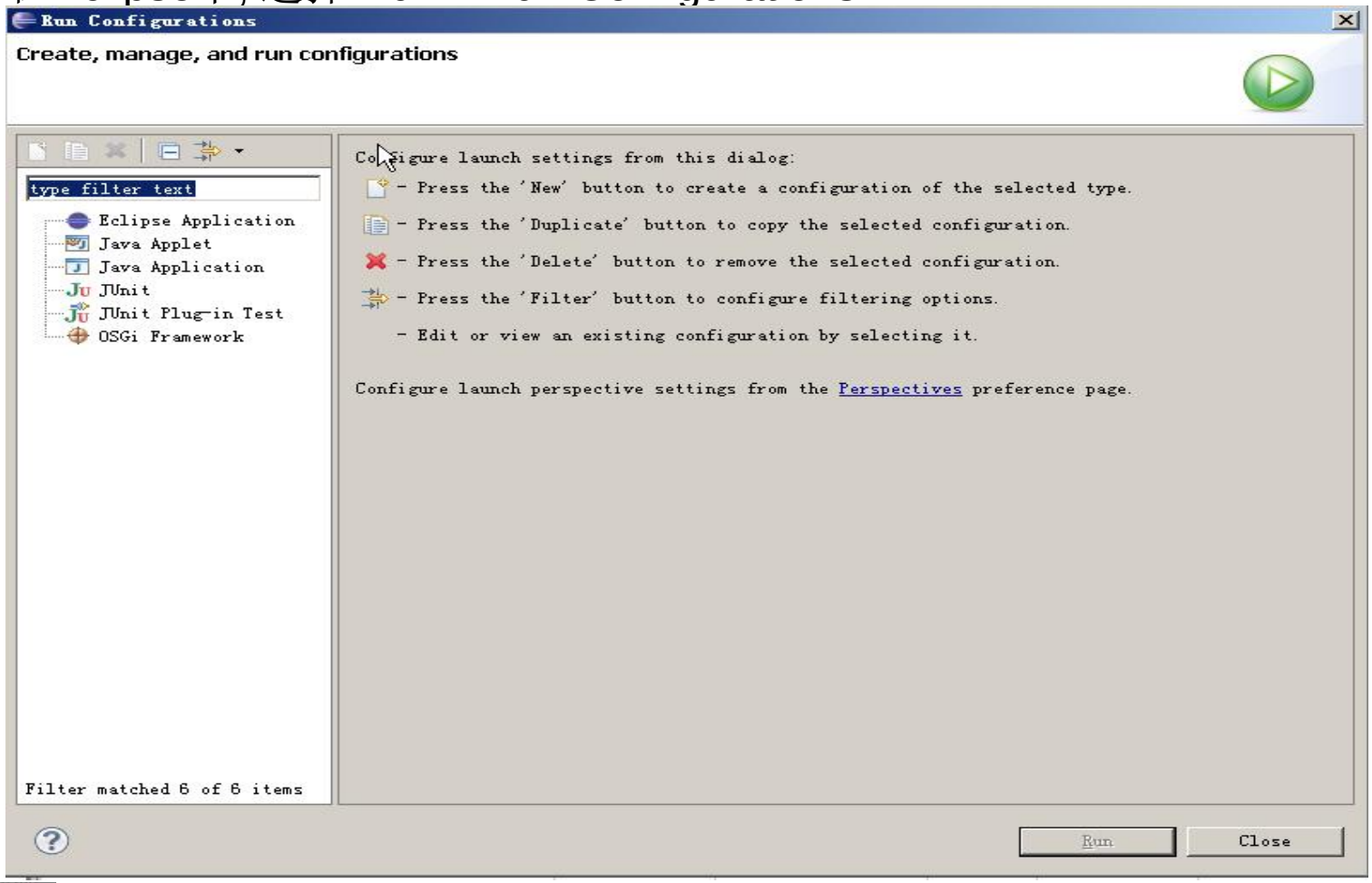
⑨点击Finish 按钮。然后选择刚才添加的平台TcTarget，并点击**OK**.



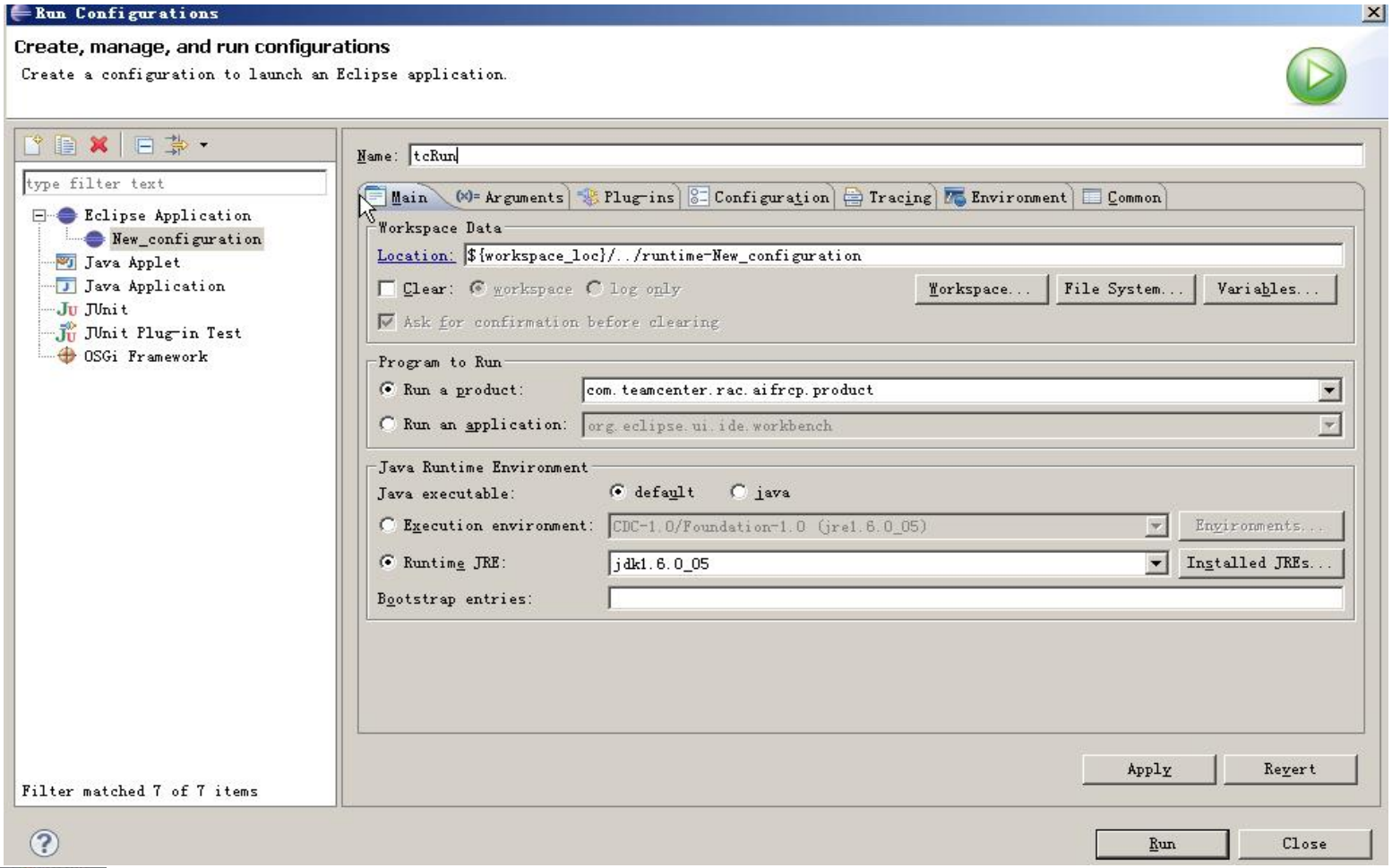
开发环境安装与部署

- 通过Eclipse去运行Rich Client。

① 在Eclipse中, 选择 Run→Run Configurations.

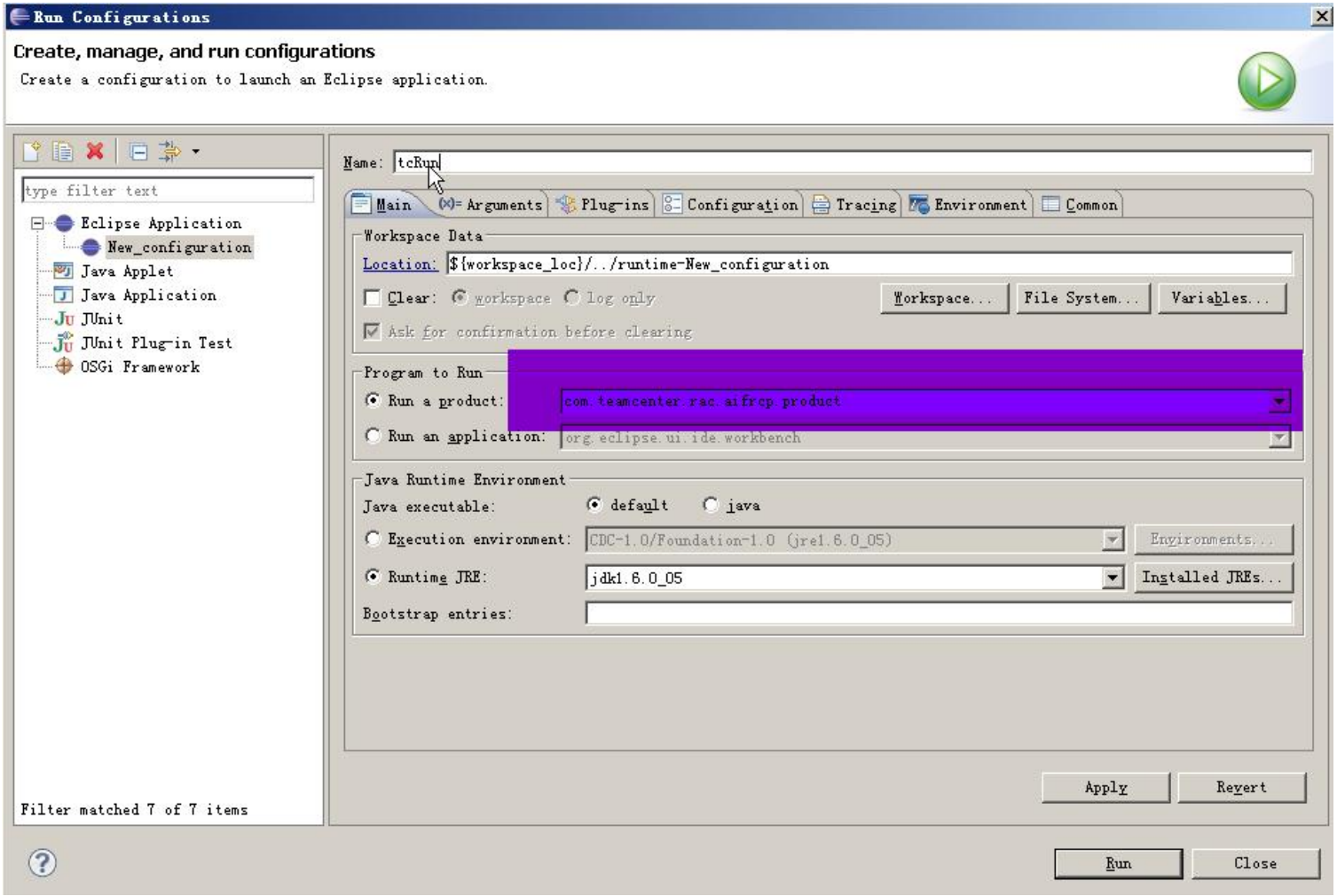


②在 Create, manage, and run configurations 对话框的左边，双击Eclipse Application, 然后选择 New_configuration，并且修改名称为tcRun。

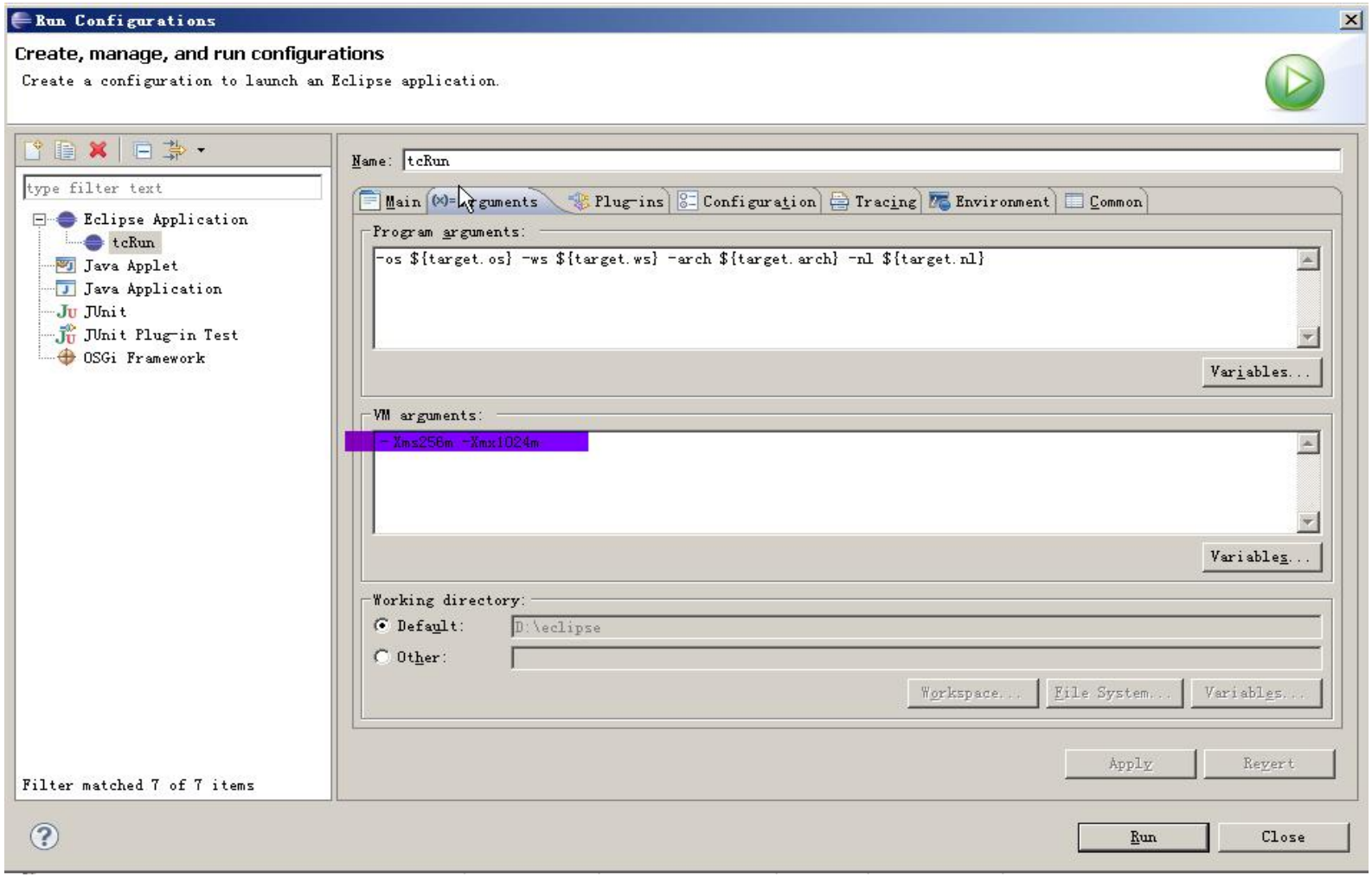


开发环境安装与部署

③确保Run a product 列表选择为com.teamcenter.rac.aifrcp.product。



④点击 **Arguments**页，并在**VM arguments** 框中输入: -Xms256m -Xmx1024m

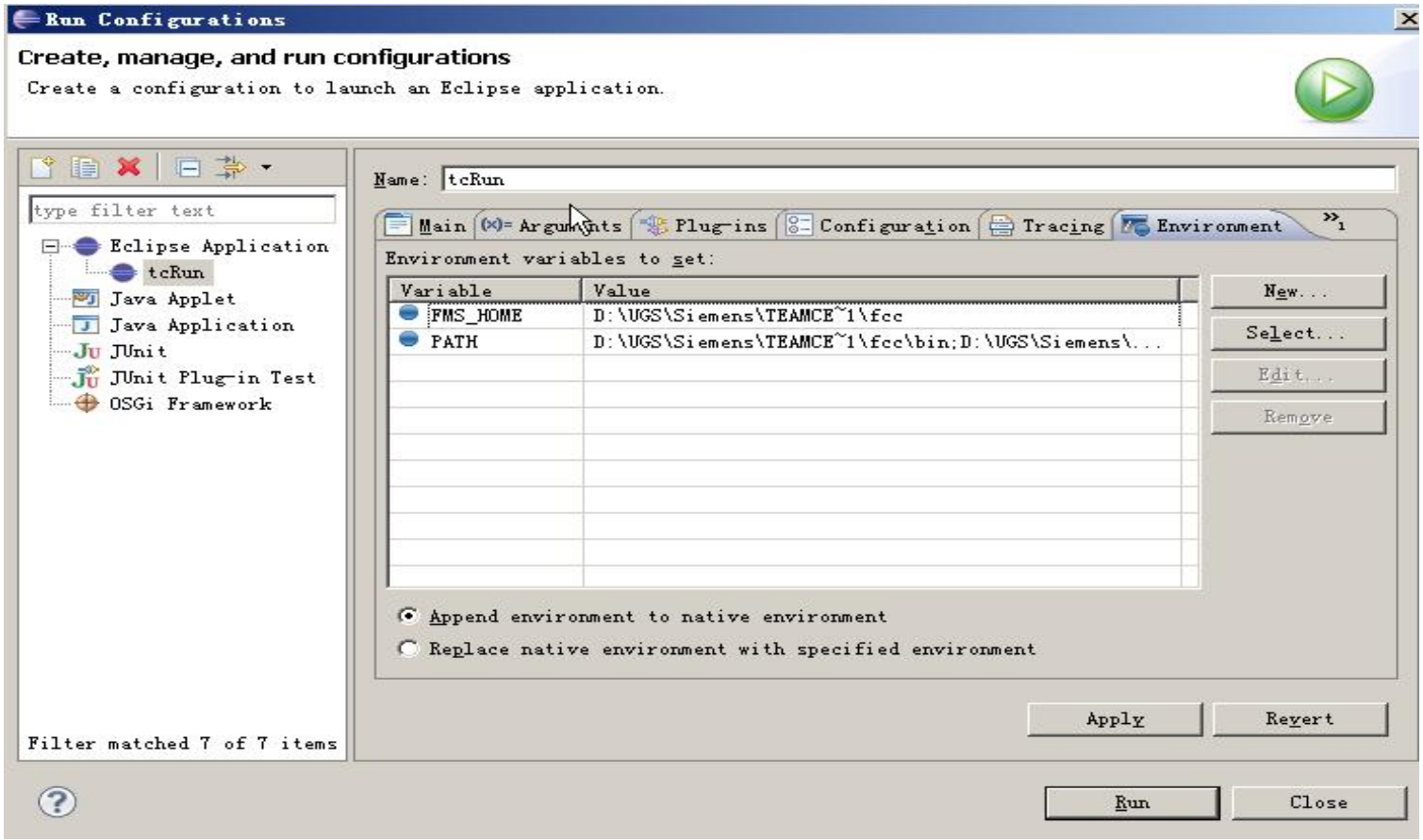


开发环境安装与部署

⑤点击Envirement页面，新建PATH和FMS_HOME选项

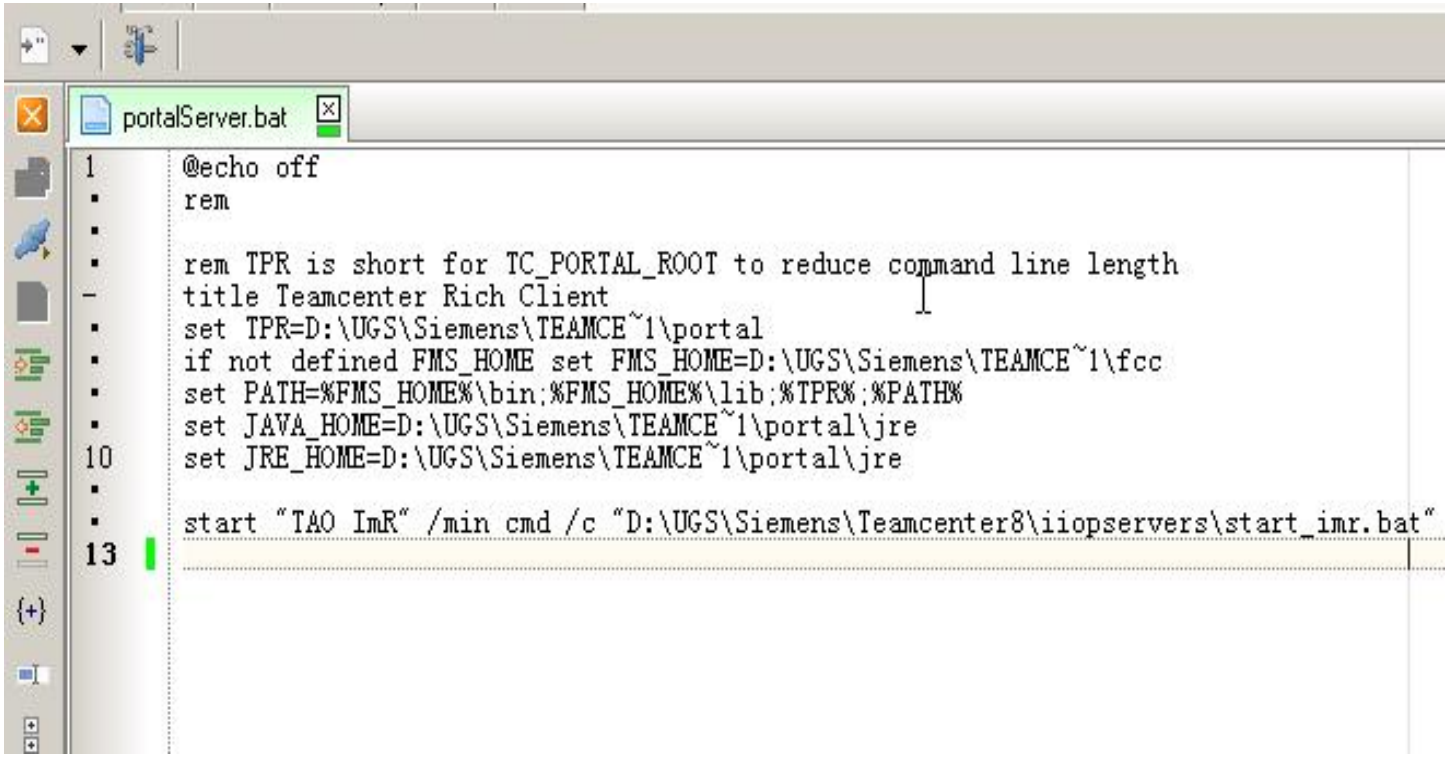
FMS_HOME=TC_ROOT\fcc

PATH=%FMS_HOME%\bin;%FMS_HOME%\lib;%TPR%;%PATH%



Ø 编写服务端部署脚本。

1.在TC_Root\portal 目录中，打开portal.bat，修改脚本为:



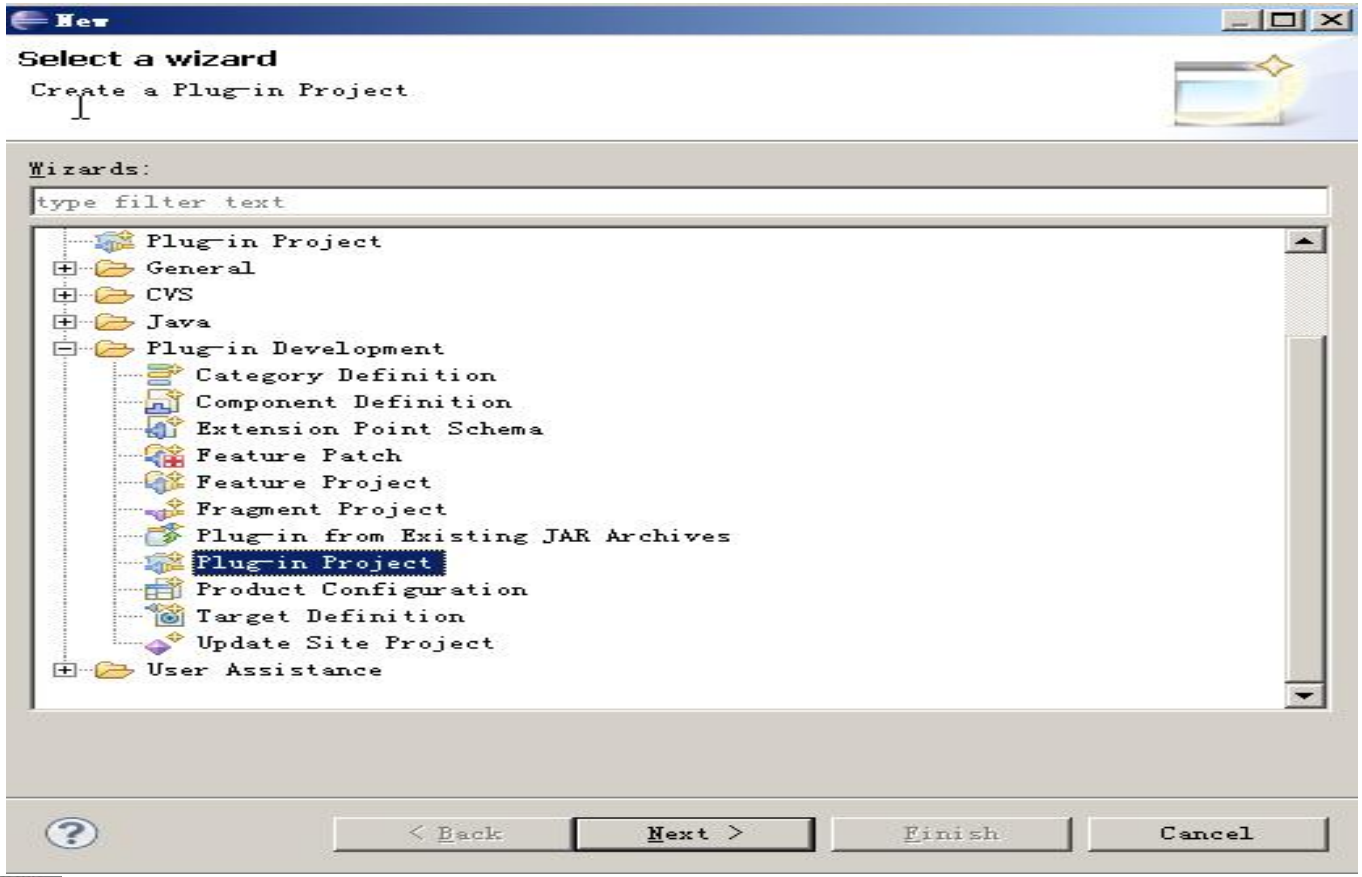
2.另存该脚本为portalServer.bat，通过Eclipse启动TC客户端时，必须通过该脚本先启动TAO控制台。

菜单-工具栏-右键菜单的客户化

- 添加菜单命令和工具栏

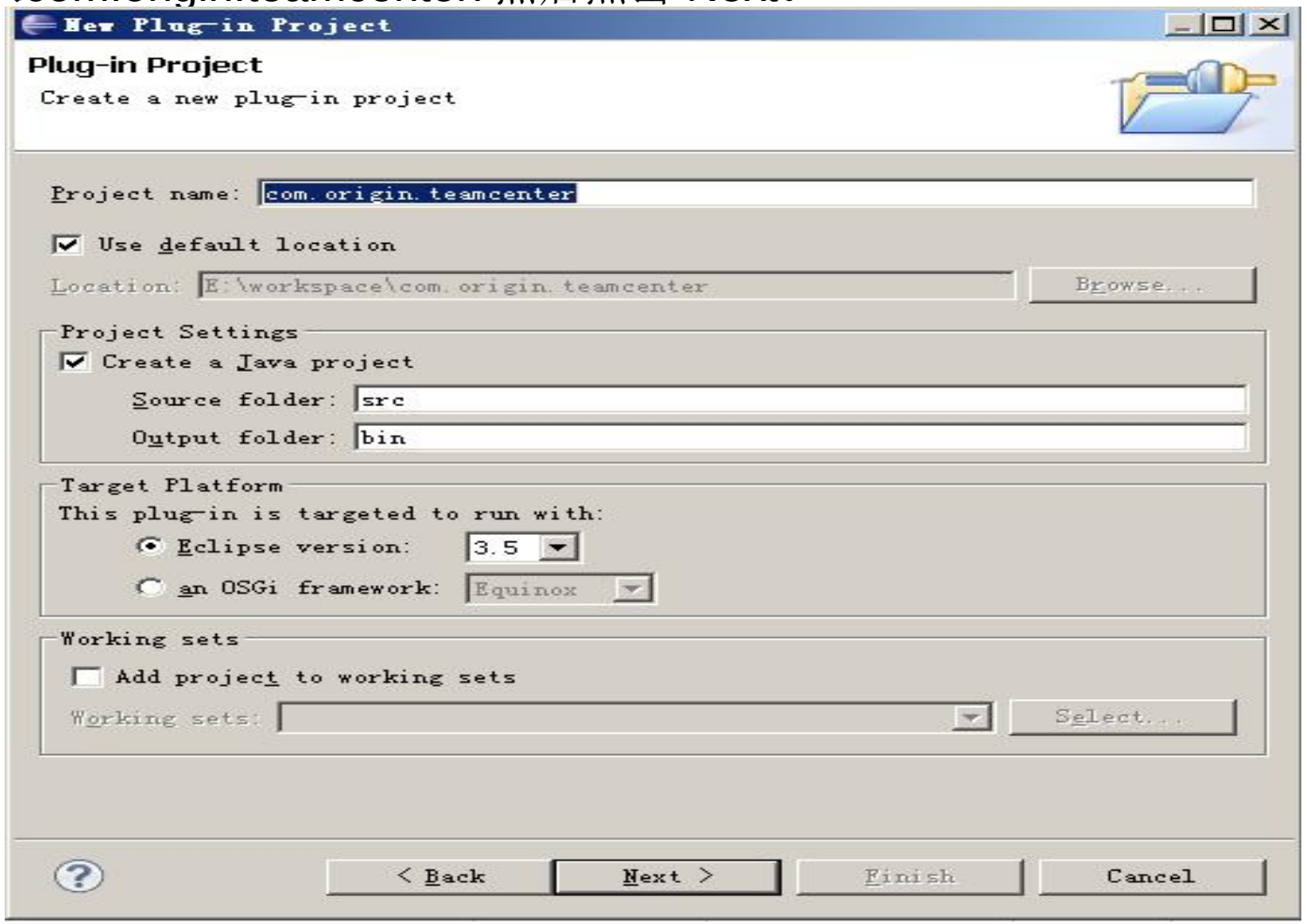
  ①在Eclipse中, 选择 File→New→Project。

  ②在New Project 对话框中, 选择 Plug-in Project然后点击Next。



菜单-工具栏-右键菜单的客户化

③在 New Plug-in Project 对话框的 Plug-in Project 面板,在 Project name 框中输入com.origin.teamcenter. 然后点击 Next。

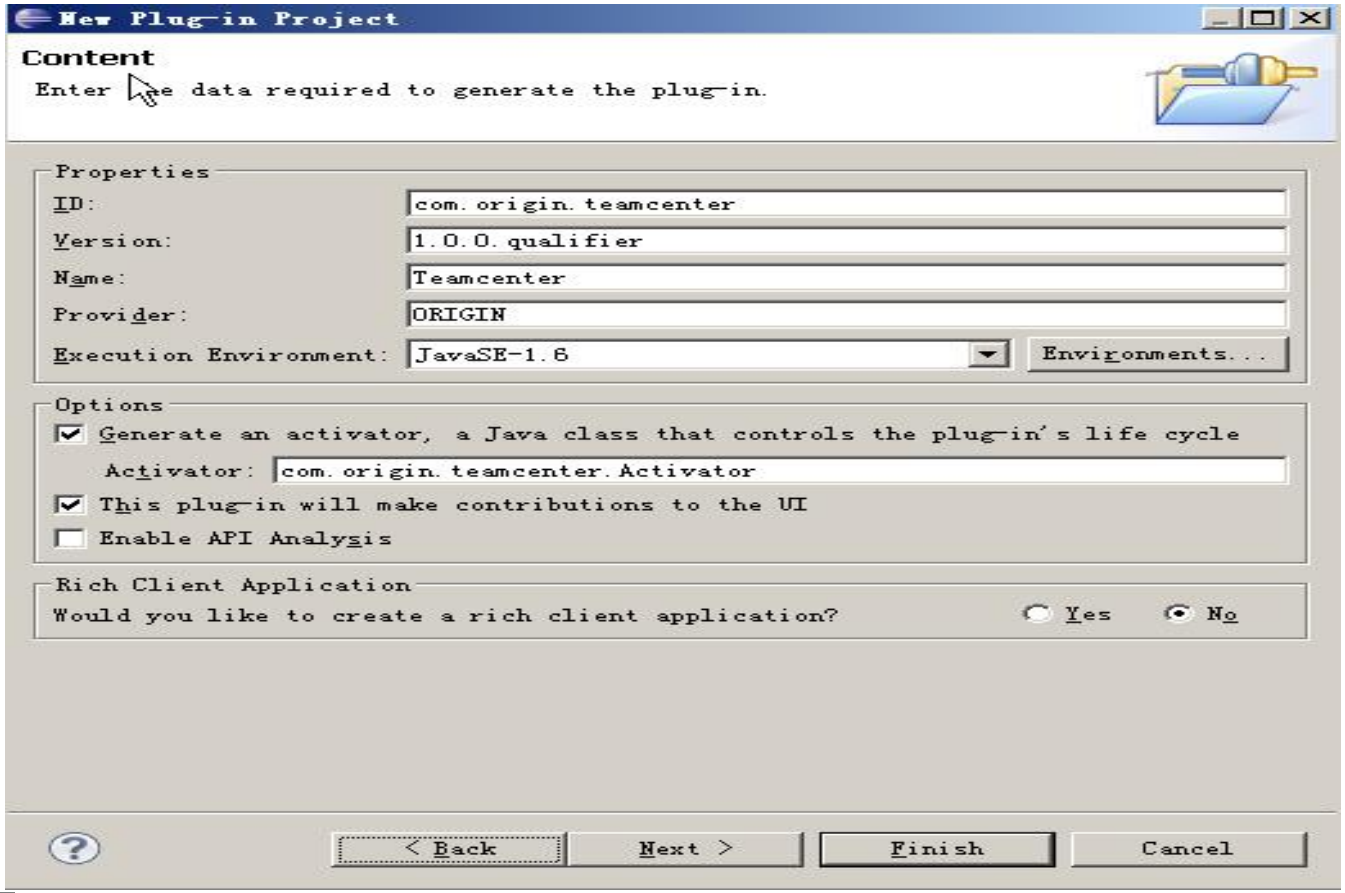


菜单-工具栏-右键菜单的客户化

④在 **New Plug-in Project** 对话框 **Content** 面板, 进行如下操作:

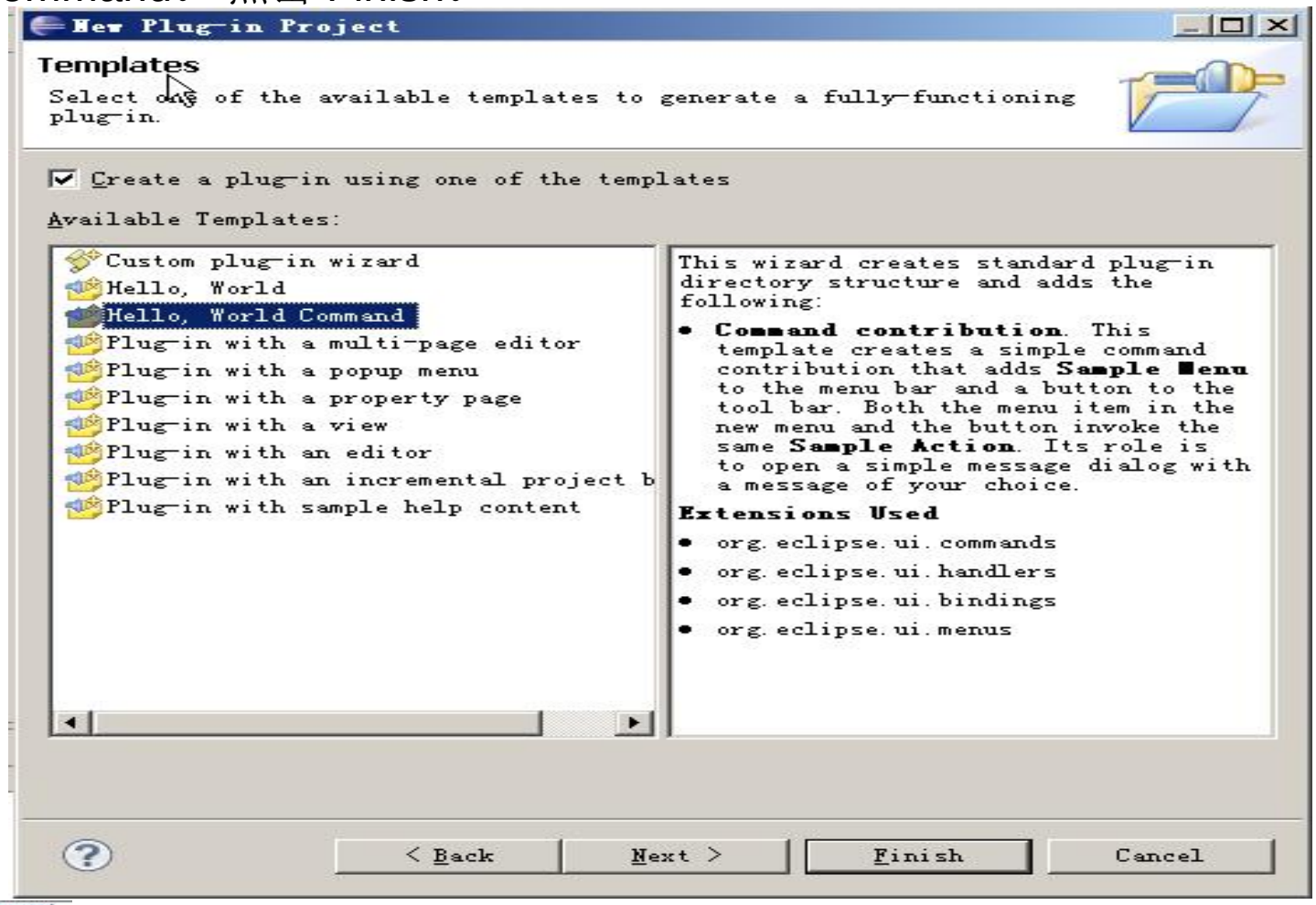
1.在 Options, 确保This plug-in will make contributions to the UI已经选择。

2.确保Would you like to create a rich client application 选择为No 。点击 Next 。



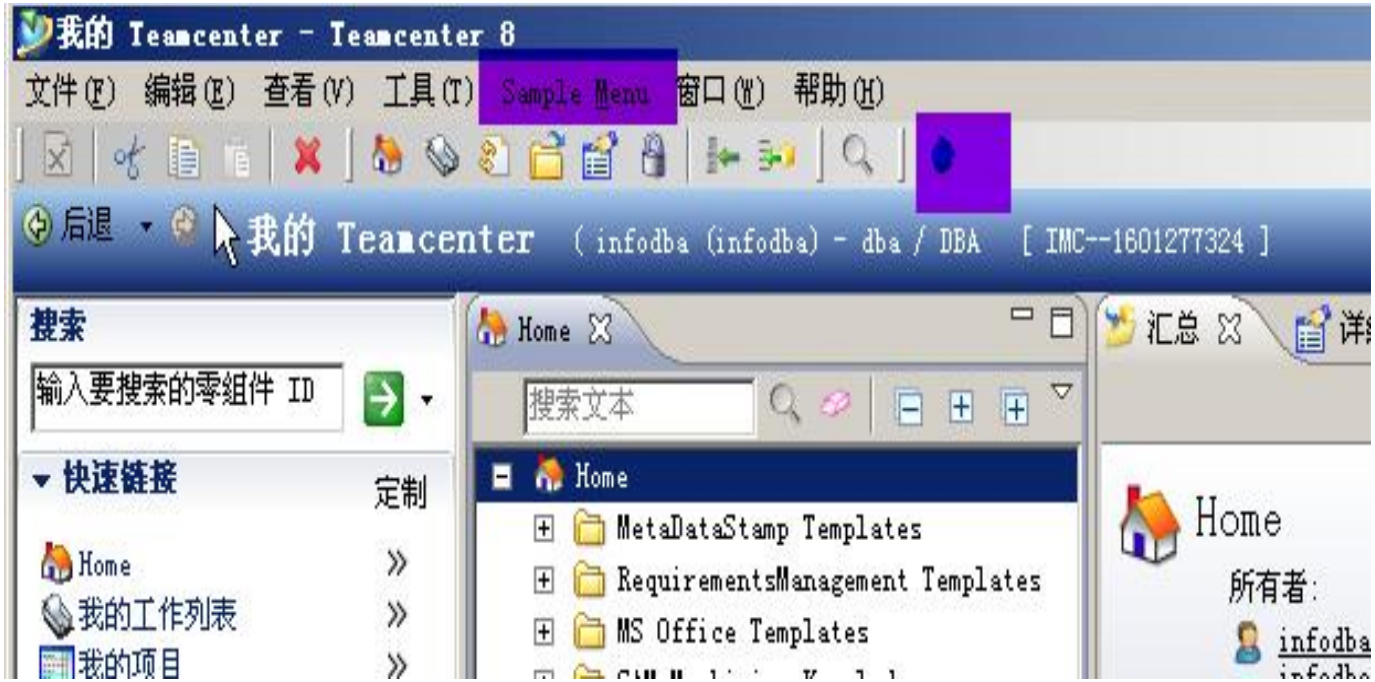
菜单-工具栏-右键菜单的客户化

⑤确保 Create a plug-in using one of these templates 中选择Hello, World Command。 点击 Finish。



菜单-工具栏-右键菜单的客户化

通过Eclipse启动，进入Teamcenter可以看到如下菜单和工具栏。



菜单-工具栏-右键菜单的客户化

⑥编辑 com.origin.teamcenter 的plugin.xml ， 就可以看到菜单栏和工具栏的添加位置菜单的整体添加过程如下：

a) 扩展**org.eclipse.ui.commands** 插件，扩展代码如下所示：

```
<command name="Sample Command"

categoryId="com.origin.teamcenter.commands.category"

id="com.origin.teamcenter.commands.sampleCommand">
</command>
```

菜单-工具栏-右键菜单的客户化

扩展org.eclipse.ui.handlers 插件，该扩展点定义相关的扩展类，并给该类一个全局ID。扩展代码如下：

```
<handler
       commandId="com.origin.teamcenter.commands.sampleCommand"
       class="com.origin.teamcenter.handlers.SampleHandler">
</handler>
```

类SampleHandler继承AbstractHandler类，并重写

public Object execute(ExecutionEvent event)方法，样例代码如下：

```
public Object execute(ExecutionEvent event) throws
ExecutionException {
                IWorkbenchWindow window =
HandlerUtil.getActiveWorkbenchWindowChecked(event);
                MessageDialog.openInformation(
                                window.getShell(),
                                "Teamcenter",
                                "Hello, Eclipse world");
                return null;
        }
```

菜单-工具栏-右键菜单的客户化

c)添加定义好的操作到菜单和工具栏，该操作都在org.eclipse.ui.menus中进行扩展，扩展代码如下图所示：

添加操作到菜单

```
<menuContribution    locationURI="menu:org.eclipse.ui.main.menu?after=additions">
         <menu
               label="Sample Menu"
               mnemonic="M"
               id="com.origin.teamcenter.menus.sampleMenu">
            <command
                  commandId="com.origin.teamcenter.commands.sampleCommand"
                  mnemonic="S"
                  id="com.origin.teamcenter.menus.sampleCommand">
            </command>
         </menu>
</menuContribution>
```

菜单-工具栏-右键菜单的客户化

添加操作到工具栏

```
<menuContribution

locationURI="toolbar:org.eclipse.ui.main.toolbar?after=additions">
         <toolbar
               id="com.origin.teamcenter.toolbars.sampleToolbar">
            <command
                  commandId="com.origin.teamcenter.commands.sampleCommand"
                  icon="icons/sample.gif"
                  tooltip="Say hello world"
                  id="com.origin.teamcenter.toolbars.sampleCommand">
            </command>
         </toolbar>
</menuContribution>
```

菜单-工具栏-右键菜单的客户化

d)定义操作菜单和工具栏的使用范围，代码如下：

```
<visibleWhen>
      <reference
         definitionId="com.teamcenter.rac.pse.inMainPerspective">
      </reference>
</visibleWhen>
```

菜单-工具栏-右键菜单的客户化

Reference节点定义添加到应用的范围。如添加到PSE com.teamcenter.rac.pse.inMainPerspective，MyTeamcenter为com.teamcenter.rac.ui.inMainPerspective。下面，就以添加菜单到MyTeamcenter为例，org.eclipse.ui.menus中扩展的代码应该进行如下定义：

```
<menuContribution
            locationURI="menu:org.eclipse.ui.main.menu?after=additions">
         <menu
               label="Sample Menu"
               mnemonic="M"
               id="com.origin.teamcenter.menus.sampleMenu">
            <command
                  commandId="com.origin.teamcenter.commands.sampleCommand"
                  mnemonic="S"
                  id="com.origin.teamcenter.menus.sampleCommand">
            <visibleWhen>
                <reference
             definitionId="com.teamcenter.rac.ui.inMainPerspective">
                                    </reference>
                           </visibleWhen>
            </command>
         </menu>
      </menuContribution>
```

菜单-工具栏-右键菜单的客户化

Ø 在系统现有结构中添加右键菜单

1.创建插件工程，步骤和上一个样例一样。

2.扩展org.eclipse.ui.commands，代码如下：

```
<extension point="org.eclipse.ui.commands">
      <command
         name="Sample Command"
         id="com.origin.shortcutmenu.commands.sampleCommand">
      </command>
</extension>
```

3.扩展org.eclipse.ui.handlers，代码如下：

```
<extension point="org.eclipse.ui.handlers">
    <handler
         commandId="com.origin.shortcutmenu.commands.sampleCommand"
         class="com.origin.shortcutmenu.handlers.SampleHandler">
    </handler>
 </extension>
```

菜单-工具栏-右键菜单的客户化

4.编写SampleHandler类，代码如下所示:

```
public class SampleHandler extends AbstractHandler {
            /**
             * The constructor.
             */
            public SampleHandler() {
            }
            /**
             * the command has been executed, so extract extract the needed information
             * from the application context.
             */
             public Object execute(ExecutionEvent event) throws ExecutionException{
                    IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
                        MessageDialog.openInformation(
                                                window.getShell(),
                                                "Shortcutmenu",
                                                "Hello, Eclipse world");
                            return null;
            }
}
```

菜单-工具栏-右键菜单的客户化

5.通过扩展org.eclipse.ui.menus，添加到右键菜单.扩展代码如下:

```
<extension point="org.eclipse.ui.menus">
<menuContribution
        locationURI="popup:org.eclipse.ui.popup.any?after=additions">
         <command
           commandId="com.origin.shortcutmenu.commands.sampleCommand"
            mnemonic="S"
            icon="icons/sample.gif"
            id="com.origin.shortcutmenu.menus.sampleCommand">
            <visibleWhen>
               <reference
               definitionId="com.teamcenter.rac.ui.inMainPerspective">
               </reference>
            </visibleWhen>
         </command>
      </menuContribution>
</extension>
```

菜单-工具栏-右键菜单的客户化

6.通过**Eclipse**启动TC客户端，可以显示我们添加的右键菜单，如下所示：