

RCP 客户端升级说明

作者：小东

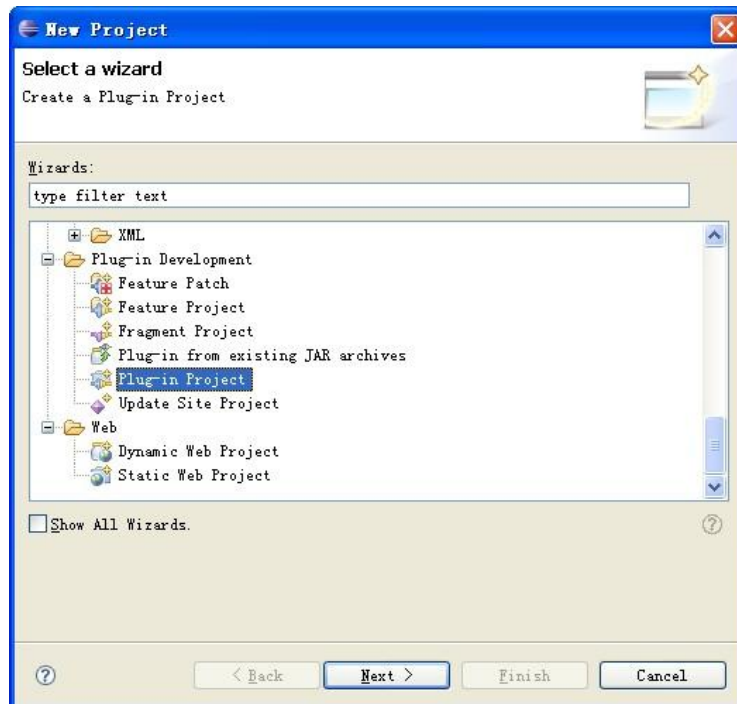
使用工具版本： Eclipse3.2

Designer V6.5

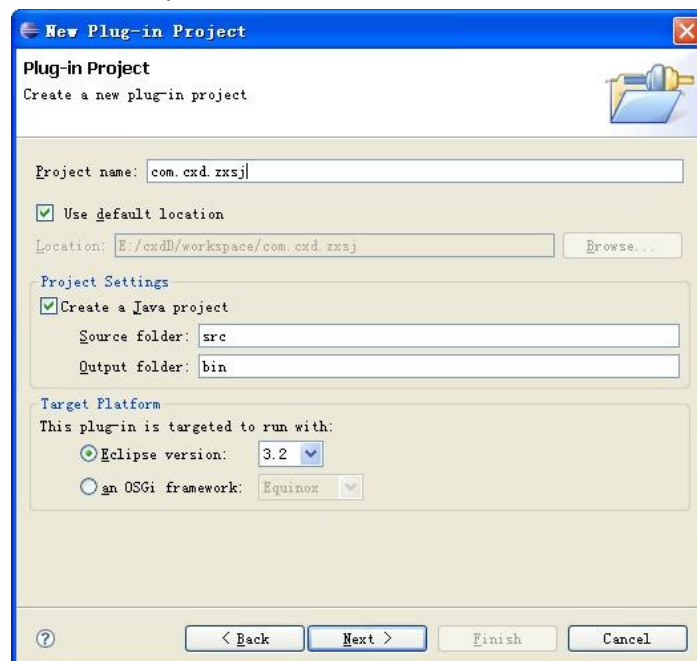
参考书籍：《Eclipse RCP 应用系统开发与实战》 作者:陈冈

制作一个 RCP 客户端

点击左侧空白处，右键选择 New→Other→Plug-in Project，选择创建一个插件工程。点击按钮 Next



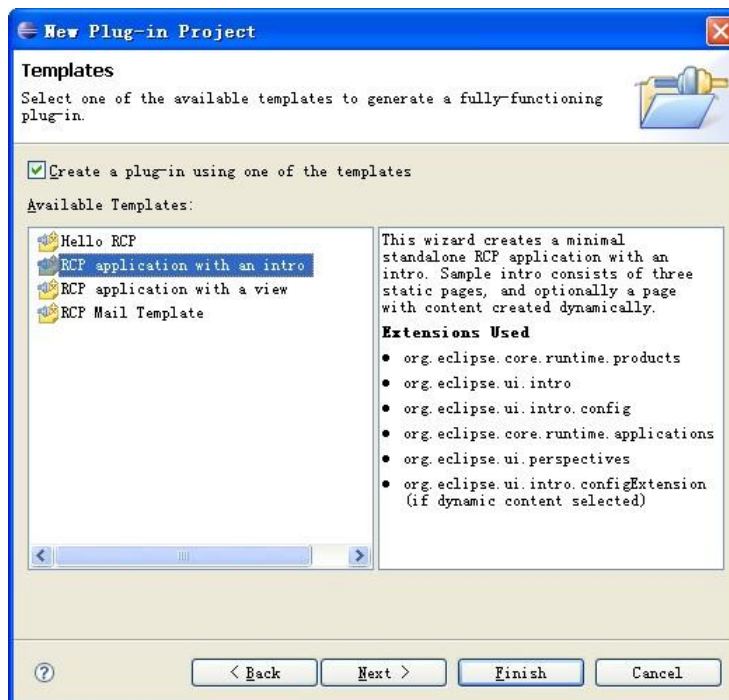
输入你的工程名称为 com.cxd.zxsj (注意：这里自己随便输入自己要定义的工程名称)，点击按钮 Next



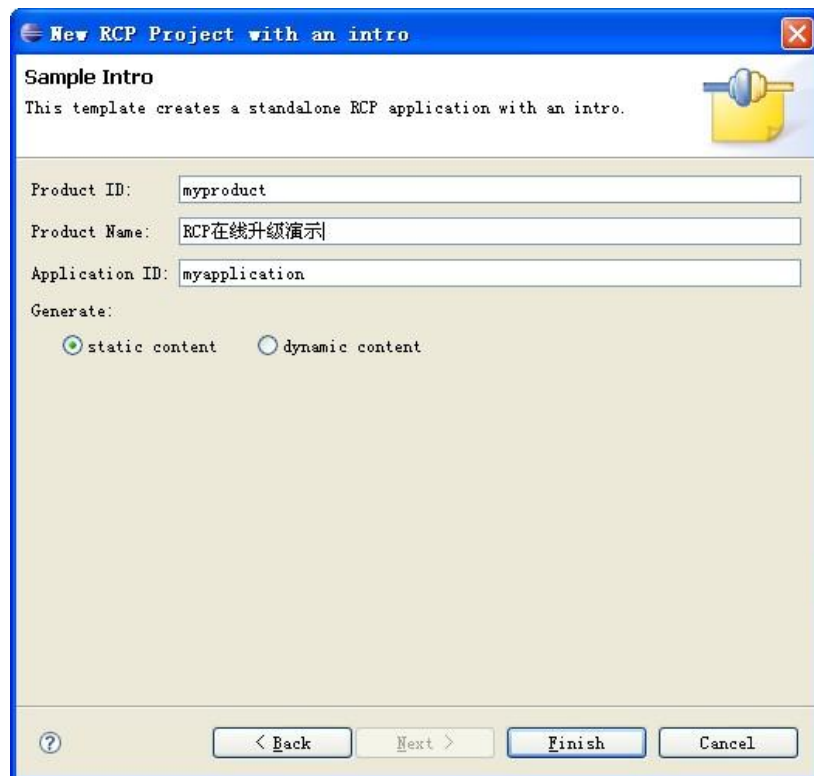
在这里要注意：在 Rich Client Application 里，Would you like to create a rich client application ?选择 Yes



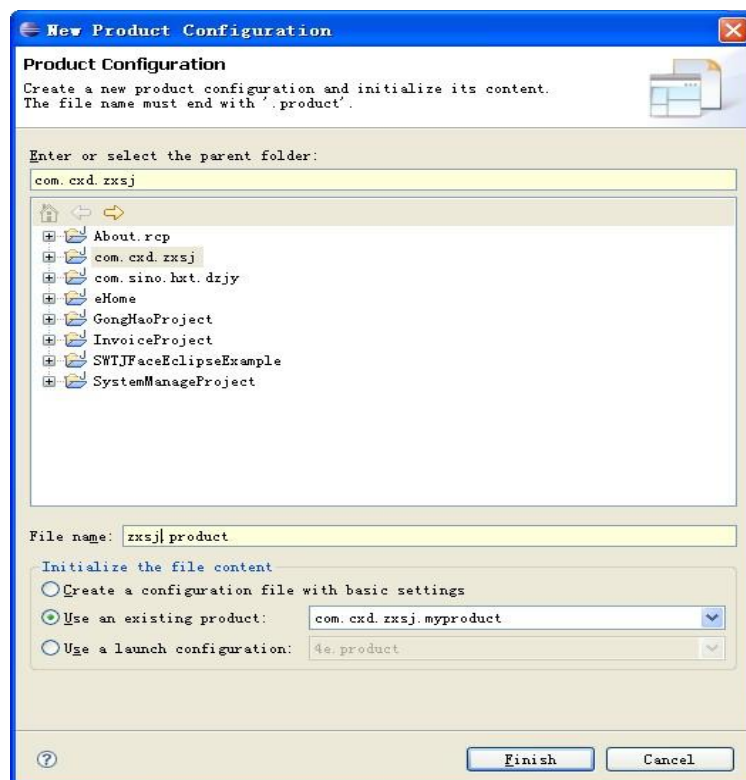
点击 Next,会出现一个选择 RCP 模板的对话框，在这里我们选择 RCP application with an intro 这个模板



然后我们定义导出产品的名称，在这里我们修改 Product ID, Product Name, Application ID 这个三个数据，（注意：在这里修改数据对我们创建*.product 这个文件有帮助,其中 Product ID 是我们创建产品文件时需要用到的，不然会在创建产品后，运行后会出错）



点击 Finish 按钮，我们完成了创建 RCP 客户端。接下来我们首先创建一个*.product 文件。选中该工程，点击右键，依次点击 New→Other→Product Configuration,弹出 New Product Configuration 对话框



在“File name”后输入该产品的名称 zxsj.product，选中 Use an existing product，点击 Finish 结束。然后我们看到在 zxsj.product 这个文件的 Overview 这个选项看里关于改工程的配置情况。然后我们点击 Launch the product 启动这个工程，看看我们的创建的 RCP 客户端。

Overview

Product Definition

This section describes general information about the product.

Specify the name that appears in the title bar of the application:

Product Name: RCP在线升级演示

Specify the product identifier:

Product ID: com.cxd.xxsj.myproduct


Specify the application to run when launching this product:


Application: com.cxd.xxsj.myapplication

The product configuration is based on: ☒ plug-ins ☐ features

Testing

1. Synchronize this configuration with the product's defining plug-in.
2. Test the product by launching a runtime instance of it:

 Launch the product

 Launch the product in Debug mode

Exporting

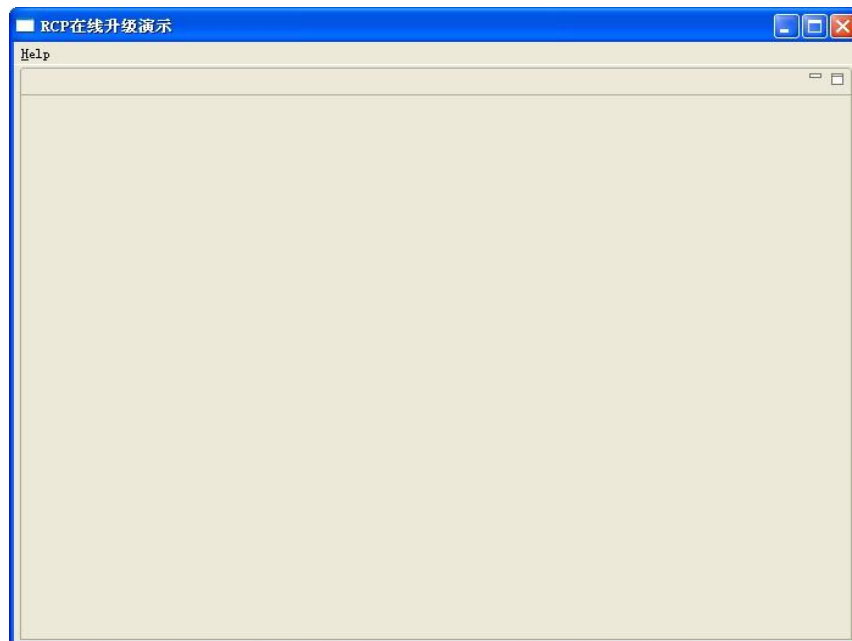
Use the Eclipse Product export wizard to package and export the product defined in this configuration.

To export the product to multiple platforms:

1. Install the RCP delta pack in the target platform.
2. List all the required fragments on the Configuration page.

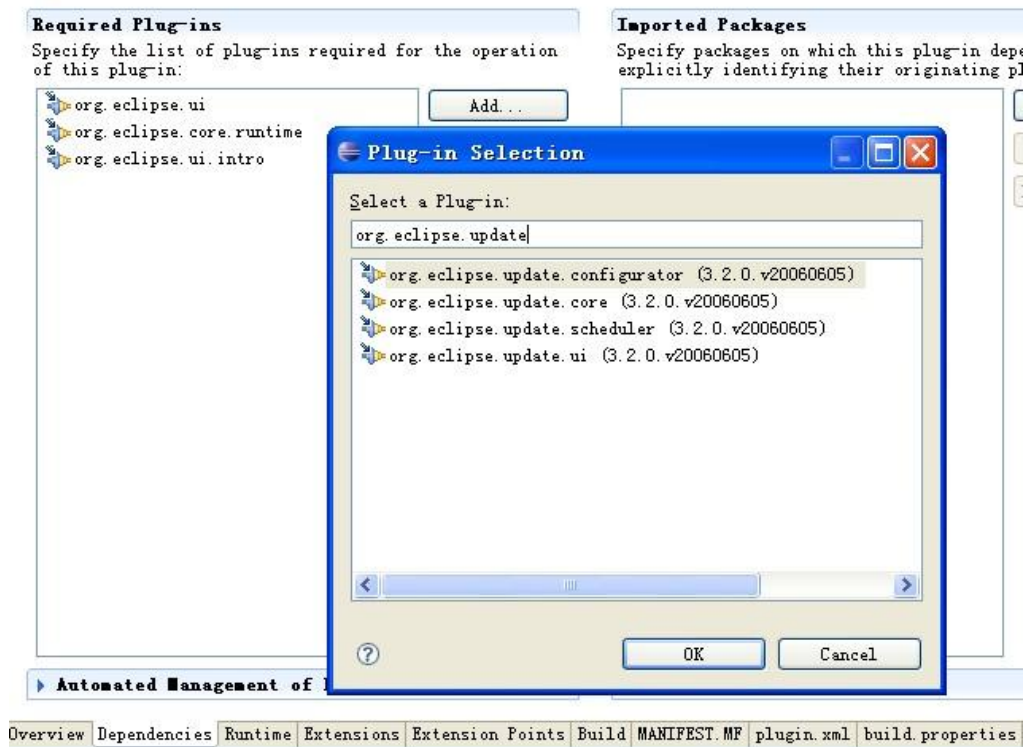
Overview Configuration Launcher Branding

效果如下：



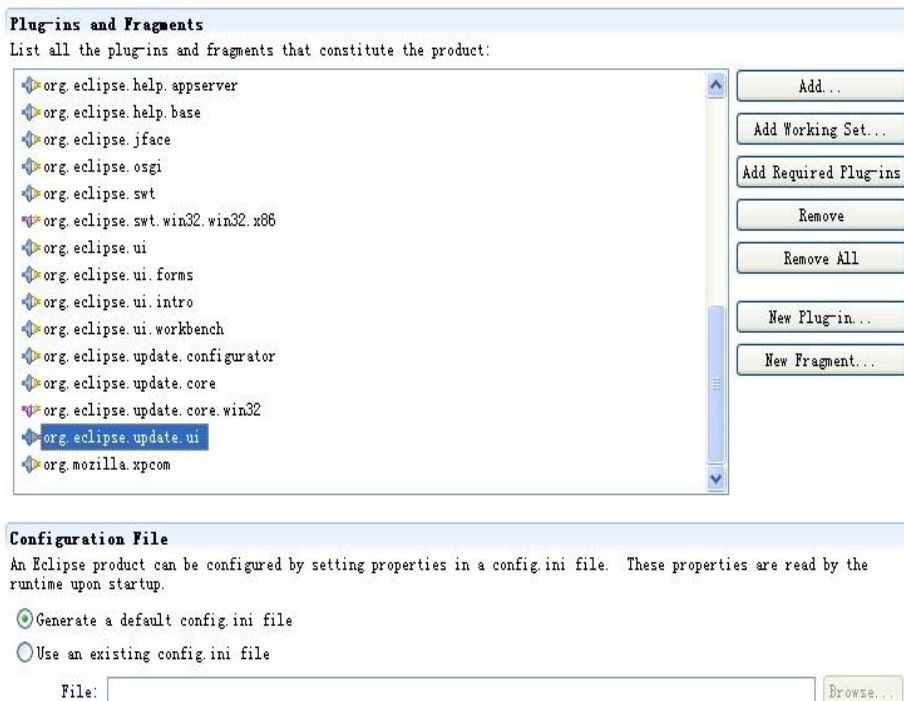
配置 Dependencies 插件依赖项：打开 plugin.xml, 切换到 Dependencies 选项卡，单击 Add 按钮，在弹出的 Plug-in Selection 对话框中选中 org.eclipse.update.configurator, org.eclipse.update.core, org.eclipse.update.ui, 再点击 OK 按钮保存。

Dependencies



然后打开 xsj.product, 切换到 Configuration 选项卡, 单击 Add Required Plug-ins 按钮。你会注意到 Eclipse 自动添 org.eclipse.update.configurator,org.eclipse.update.core,org.eclipse.update.core.win32,org.eclipse.update.ui

Configuration



然后我们开始添加一个在线升级的按钮, 首先我们创建一个 Update 类:

```
package com.cxd.xsj.actions;
```

```

import java.lang.reflect.InvocationTargetException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import org.eclipse.core.runtime.CoreException;
import org.eclipse.core.runtime.IProgressMonitor;
import org.eclipse.core.runtime.OperationCanceledException;
import org.eclipse.jface.action.Action;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.dialogs.ProgressMonitorDialog;
import org.eclipse.jface.operation.IRunnableWithProgress;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.plugin.AbstractUIPlugin;
import org.eclipse.update.configuration.IConfiguredSite;
import org.eclipse.update.configuration.ILocalSite;
import org.eclipse.update.core.IFeature;
import org.eclipse.update.core.IFeatureReference;
import org.eclipse.update.core.SiteManager;
import org.eclipse.update.operations.IInstallFeatureOperation;
import org.eclipse.update.operations.OperationsManager;
import org.eclipse.update.search.BackLevelFilter;
import org.eclipse.update.search.EnvironmentFilter;
import org.eclipse.update.search.IUpdateSearchCategory;
import org.eclipse.update.search.IUpdateSearchResultCollector;
import org.eclipse.update.search.UpdateSearchRequest;
import org.eclipse.update.search.UpdateSearchScope;

import com.cxd.zxsj.Activator;
import com.cxd.zxsj.sys.IAppConstants;
import com.cxd.zxsj.sys.IImageKey;

public class Update extends Action
{
    private List<IInstallFeatureOperation> updateOperation = new
ArrayList<IInstallFeatureOperation>();

    private int updateTag = -1;

    public Update ()
    {
        setId("com.cxd.zxsj.actions.Update");
        setText("在线升级@ALT+O");
    }

```

```

setImageDescriptor(AbstractUIPlugin.imageDescriptorFromPlugin(IAppConstants.
APPLICATION_ID, IImageKey.UPDATE));
}

public void run ()
{
    ProgressMonitorDialog pmd = new ProgressMonitorDialog(null);
    IRunnableWithProgress rwp = new IRunnableWithProgress()
    {
        public void run ( IProgressMonitor monitor )
            throws InvocationTargetException, InterruptedException
        {
            monitor.beginTask("正在进行软件升级。。。",
IPProgressMonitor.UNKNOWN);
            try
            {
                final String homeBase =
System.getProperty("zxsj.homebase");
                monitor.subTask("搜索服务器站点" + homeBase + "。。。");
                // 搜索升级信息
                searchUpdateInfo(homeBase, monitor);
                // 执行更新
                updateTag = installUpdate(updateOperation, monitor);
                monitor.done();
            }
            catch ( CoreException e1 )
            {
                e1.printStackTrace();
            }
            catch ( MalformedURLException e )
            {
                e.printStackTrace();
            }
        }
    };
    try
    {
        pmd.run(true, false, rwp);
    }
    catch ( Exception e )
    {
        e.printStackTrace();
    }
    if ( updateTag == 1 )

```

```

        PlatformUI.getWorkbench().restart();
    else if ( updateTag == 0 )
        MessageDialog.openInformation(null, "提示", "服务器站点没有可供升级
的新版本! ");
    else
        MessageDialog.openError(null, "提示", "升级失败! ");
}

// 搜索升级信息
private void searchUpdateInfo ( String homeBase, IProgressMonitor monitor )
    throws MalformedURLException, OperationCanceledException,
CoreException
{
    // 准备搜索
    IUpdateSearchCategory category =
UpdateSearchRequest.createDefaultSiteSearchCategory();
    UpdateSearchScope searchScope = new UpdateSearchScope();
    URL url = new URL(homeBase);
    searchScope.addSearchSite(homeBase, url, null);
    UpdateSearchRequest searchRequest = new UpdateSearchRequest(category,
searchScope);

    // 过滤掉较低版本
    searchRequest.addFilter(new BackLevelFilter());
    // 过滤掉与当前环境不匹配的
    searchRequest.addFilter(new EnvironmentFilter());
    // 设置搜索相关信息
    IUpdateSearchResultCollector collector = new SearchResultCollector();
    // 执行搜索
    searchRequest.performSearch(collector, monitor);
}

@SuppressWarnings("deprecation")
private int installUpdate ( List<IInstallFeatureOperation> update,
IProgressMonitor monitor )
    throws CoreException
{
    if ( update.size() > 0 )
    {
        monitor.subTask("获取本地软件信息。。。");
        ILocalSite localSite = SiteManager.getLocalSite();
        IConfiguredSite localConfigSite =
localSite.getCurrentConfiguration().getConfiguredSites()[0];
        IFeatureReference [] localFeatures =
localConfigSite.getConfiguredFeatures();
    }
}

```



```

        for ( int i = 0; i < update.size(); i++ )
        {
            IInstallFeatureOperation op = (IInstallFeatureOperation)
update.get(i);

            IFeature feature = op.getFeature();
            for ( int j = 0; j < localFeatures.length; j++ )
            {
                String id = localFeatures
[j].getVersionedIdentifier().getIdentifier();
                String sid =
feature.getVersionedIdentifier().getIdentifier();
                if ( sid.equals(id) )
                {
                    if
( feature.getVersionedIdentifier().getVersion().isGreaterThan(localFeatures
[j].getVersionedIdentifier().getVersion()) )
                    {
                        // 升级
                        localConfigSite.install(feature, null,
monitor);

                        return 1;
                    }
                }
            }
        }
    }
    return 0;
}

// 收集更新内容
class SearchResultCollector implements IUpdateSearchResultCollector
{
    public void accept ( IFeature match )
    {
        IInstallFeatureOperation oper =
OperationsManager.getOperationFactory().createInstallOperation(null, match,
null, null, null);
        updateOperation.add(oper);
    }
}
}

```

然后我们修改 ApplicationActionBarAdvisor 这个类

```
package com.cxd.zxsj.intro;

import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.action.MenuManager;
import org.eclipse.ui.actions.ActionFactory;
import org.eclipse.ui.actions.ActionFactory.IWorkbenchAction;
import org.eclipse.ui.application.ActionBarAdvisor;
import org.eclipse.ui.application.IActionBarConfigurer;
import org.eclipse.ui.IWorkbenchActionConstants;
import org.eclipse.ui.IWorkbenchWindow;

import com.cxd.zxsj.actions.Update;

public class ApplicationActionBarAdvisor extends ActionBarAdvisor
{

    private IWorkbenchAction introAction;

    private Update updateAction;

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

    protected void makeActions ( IWorkbenchWindow window )
    {
        introAction = ActionFactory.INTRO.create(window);
        register(introAction);
        //注册
        updateAction = new Update();
        register(updateAction);
    }

    protected void fillMenuBar(IMenuManager menuBar) {

        MenuManager helpMenu = new MenuManager("&Help",
IWorkbenchActionConstants.M_HELP);
        menuBar.add(helpMenu);

        // Help
```

```

        helpMenu.add(introAction);
        helpMenu.add(updateAction);
    }
}

```

注意: 这里我们添加了一个变量 `updateAction`, 在 `makeActions()` 方法注册这个 `Action`, 然后我们添加菜单 `helpMenu.add(updateAction)`。

效果如下:



接下来我们来配置更新服务器的 URL 地址: 在该项目中创建一个 `configuration` 文件夹, 在其下创建一个 `config.ini` 文件:

文件内容:

#Product Runtime Configuration File

osgi.splashPath=platform:/base/plugins/com.cxd.zxsj

eclipse.product=com.cxd.zxsj.myproduct

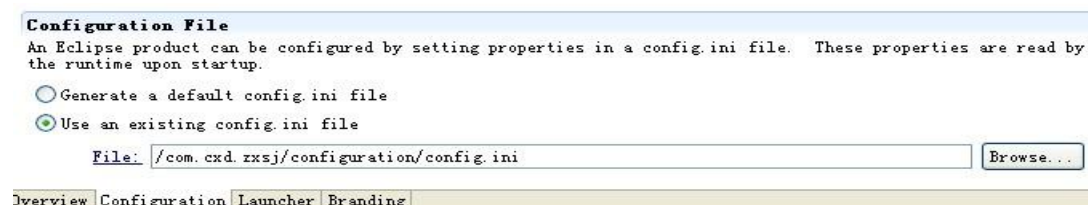
osgi.bundles=org.eclipse.equinox.common@2:start,org.eclipse.update.configurator@3:start,org.eclipse.core.runtime@start

osgi.bundles.defaultStartLevel=4

zxsj.homebase=http://localhost:8080/com.cxd.zxsj.update

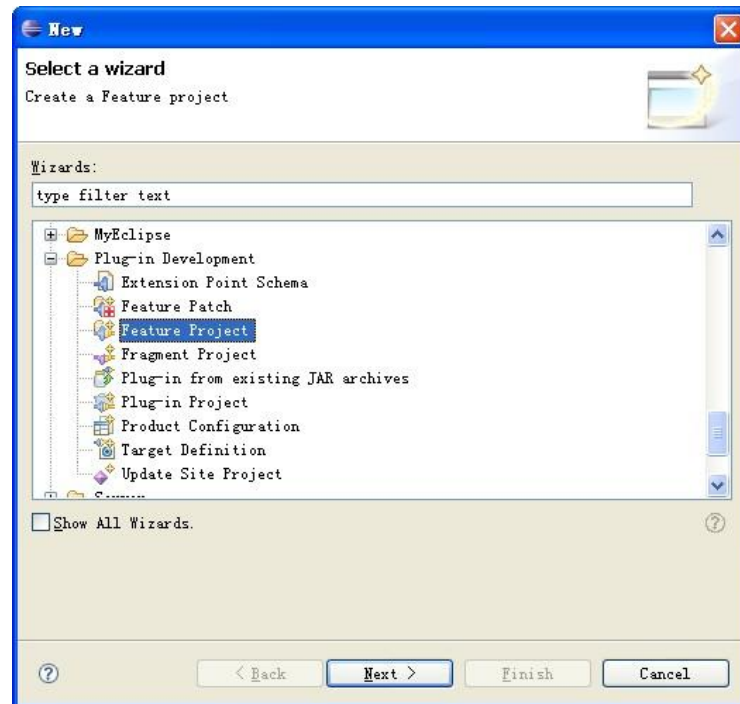
注意: `eclipse.product` 后面的值应该是你的产品 ID, 即你在打开 `zxsj.product` 文件 `Overview` 页面中的 `product ID` 值, `zxsj.homebase` 是我们请求服务器的连接地址 URL;

打开 `zxsj.product`, 切换到 `Configuration`, 在 `Configuration File` 区选择 “use an existing config.ini file”, 点击 `Browse` 浏览, 选中 `Configuration/config.ini`, 如图:

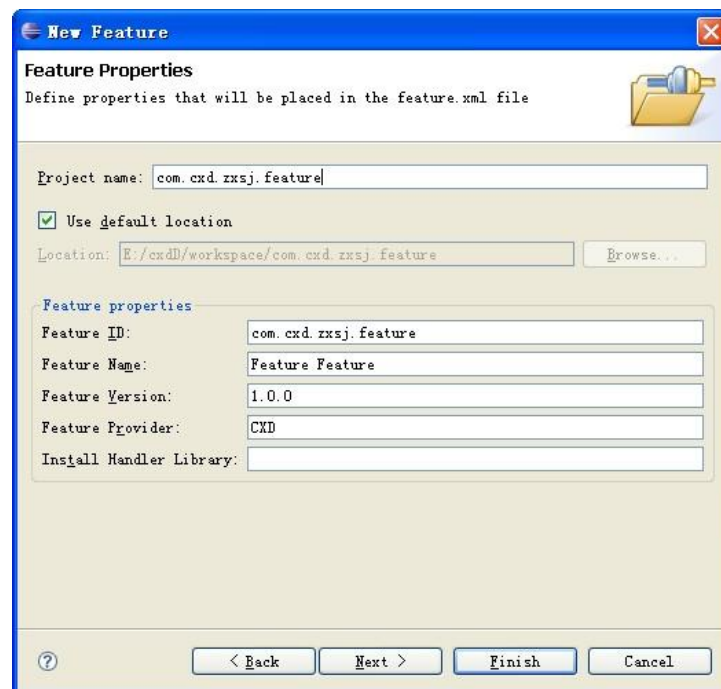


创建 Feature 功能部件项目

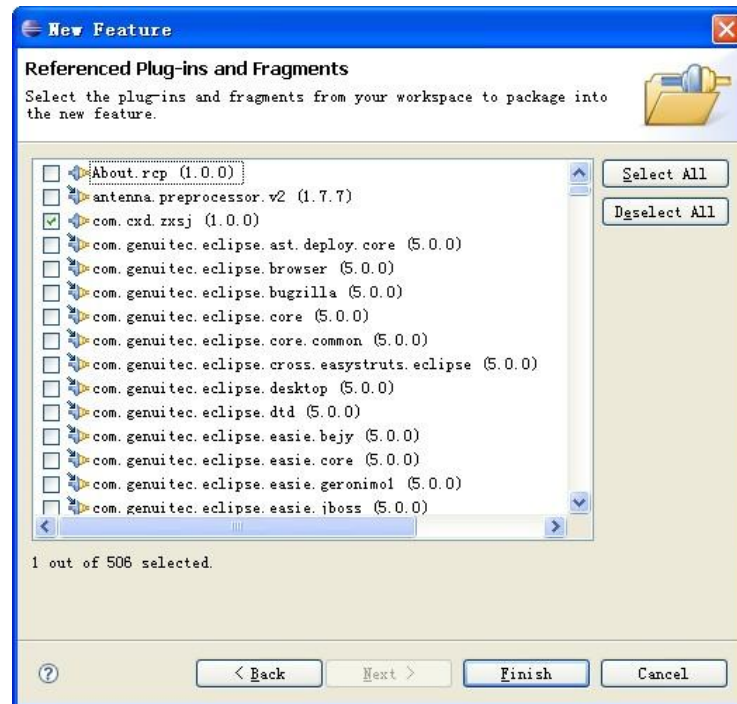
点击菜单 New→Other→Plug-in Project→Feature Project，选择创建一个 Feature Project.



然后点击 NEXT 进行下一步；



在该窗体上输入工程的名字，点击 Next



在 New Feature 对话框里找到我们创建的工程，即 com.cxd.zxsj(1.0.0)勾选中，点击 Finish 结束。在打开 feature.xml 文件，填写 Update Site URL和 Update Site Name 这两个文本框，前者为请求地址，后者为网站名称。

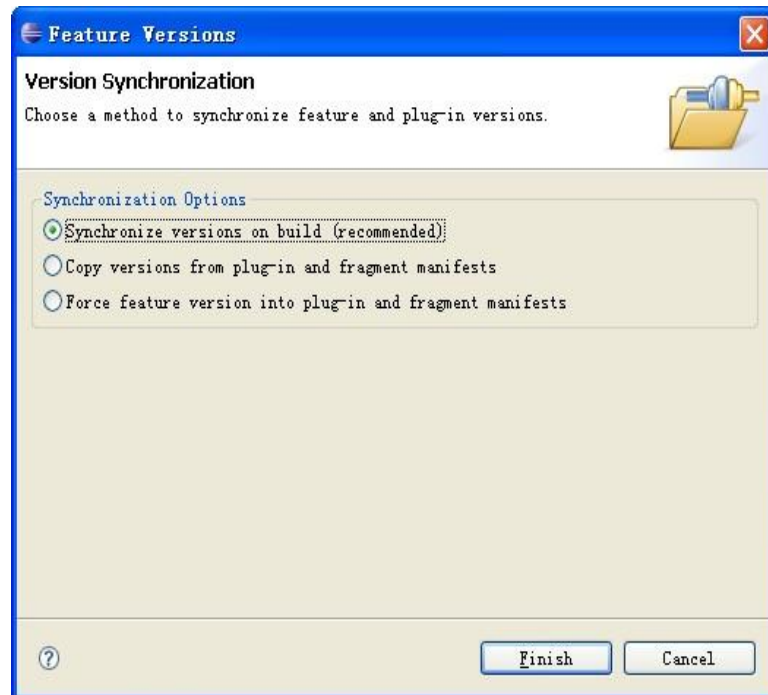
Feature Feature

General Information	
This section describes general information about this feature.	
ID:	com.cxd.zxsj.feature
Version:	1.0.0
Name:	Feature Feature
Provider:	CXD
Branding Plug-in:	<input type="text"/> Browse...
Update Site URL:	http://localhost:8080/com.cxd.zxsj.update
Update Site Name:	RCP客户端在线升级网站

然后找到 Exporting，点击 Synchronize 连接。

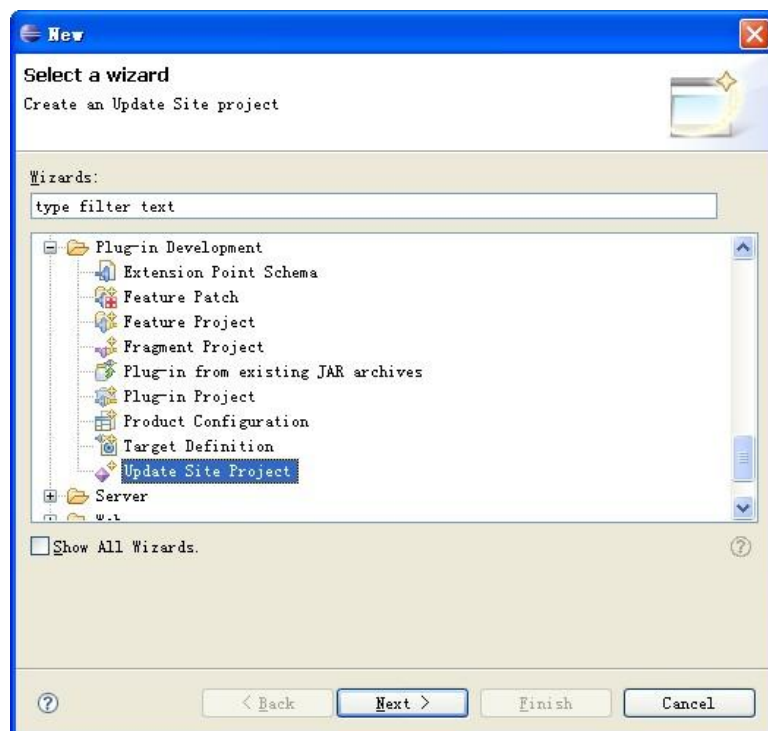
Exporting
To export the feature:
1. Synchronize versions of contained plug-ins and fragments with their version in the workspace
2. Specify what needs to be packaged in the feature archive on the Build Configuration page
3. Export the feature in a format suitable for deployment using the Export Wizard

Feature Versions 对话框，选中第二项 Copy versions from plug-in and fragment manifests, 点击 Finish 结束并保存。



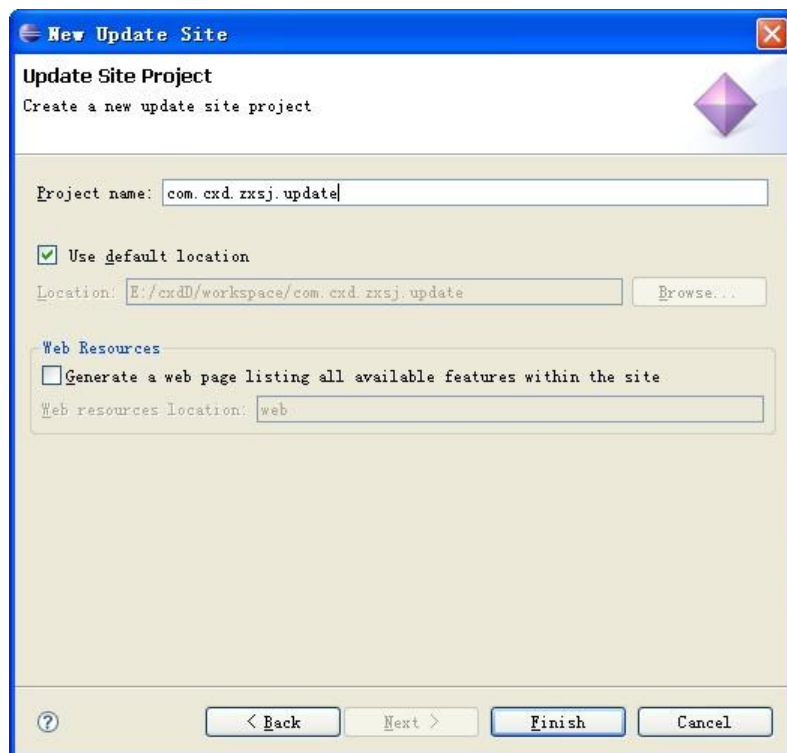
创建更新站点项目

点击菜单 New→Other→Plug-in Project→Update Site Project ,创建一个更新站点项目。

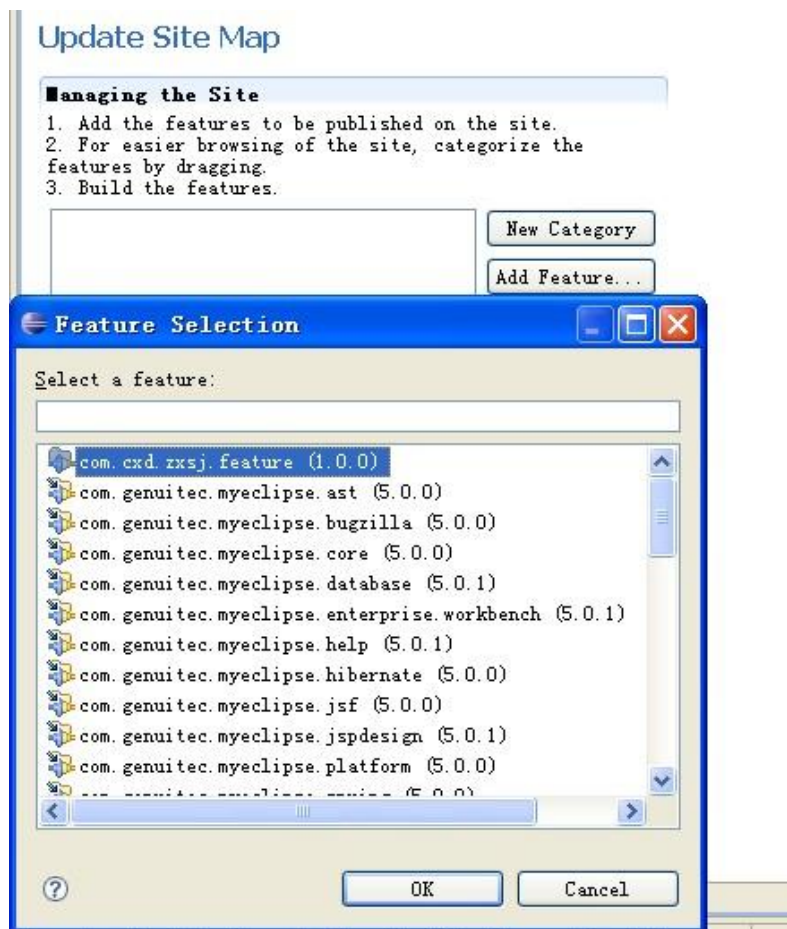


接下来在 New Update Site 对话框里输入工程的名称 com.cxd.zxsj.update ，这里需要把 Web Ressources

里勾选中（如果这里选中，后面就不要再创建 web 文件及其之下的两个文件）点击 Finish 结束。



点击 Add Feature，弹出 Feature Selection 对话框，选中我们创建的 Feature 项目，单击 OK 结束。



打开 site.xml 文件，切换到 Archives 页，输入 URL 和 Description 这两项。前者为网站地址，后者为网站名称。

Site Description
Describe the update site and specify its address:

URL:

http://localhost:8080/com.cxd.zxsj.update/

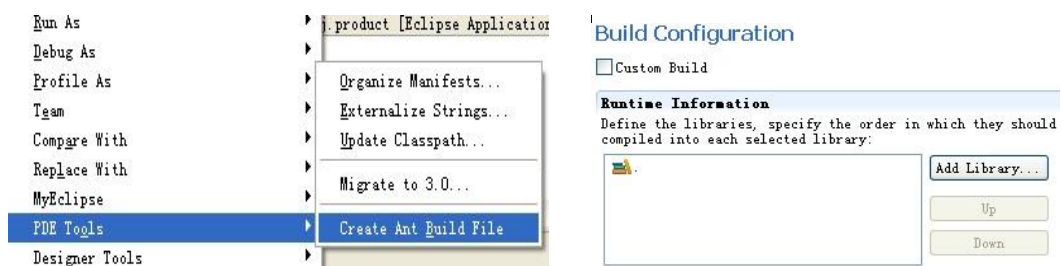
Description:

RCP 客户端在线升级网站

到这里就结束了，我们做升级所需要的工程已经全部创建完成了。

导出产品

首先在 com.cxd.zxsj 项目里打开文件 plugin.xml 文件，在 Build 页，取消 Custom Build 勾选。然后在 Plugin.xml 上单击鼠标右键，在弹出菜单中选择 PDE Tools Create Ant Build,将生成文件 build.xml 文件。在切换到 Plugin.xml 文件的 Build 页，勾选 “Custom Build” ,保存。



打开 build.xml,找到<javac>,添加属性 encoding="GBK"

修改前:

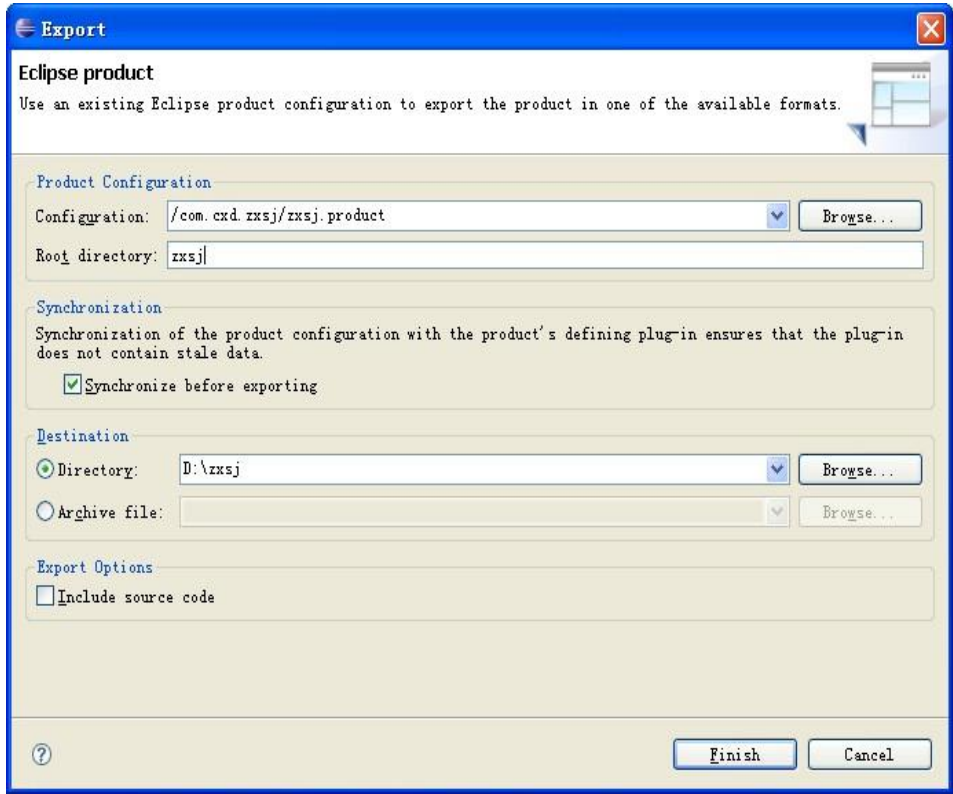
```
<!-- compile the source code -->
<javac destdir="${temp.folder}/@dot.bin" failonerror="${javacFailOnError}" ver
  <compilerarg line="${compilerArg}" compiler="${build.compiler}"/>
  <classpath refid="@dot.classpath" />
  <src path="src/" />
  <compilerarg value="@${basedir}/javaCompiler...args" compiler="org.eclipse
  <compilerarg line="-log '${temp.folder}/@dot.bin${logExtension}'" compile
</javac>
```

修改后:

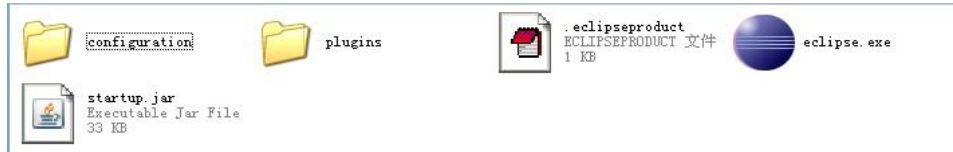
```
<!-- compile the source code -->
<javac encoding="GBK" destdir="${temp.folder}/@dot.bin" failonerror="${javacF
  <compilerarg line="${compilerArg}" compiler="${build.compiler}"/>
  <classpath refid="@dot.classpath" />
  <src path="src/" />
  <compilerarg value="@${basedir}/javaCompiler...args" compiler="org.eclipse
  <compilerarg line="-log '${temp.folder}/@dot.bin${logExtension}'" compile
</javac>
```

打开 zxsj.product 文件，单击 Overview 页“Eclipse Product export wizard”，在弹出对话框中 Directory 后输入

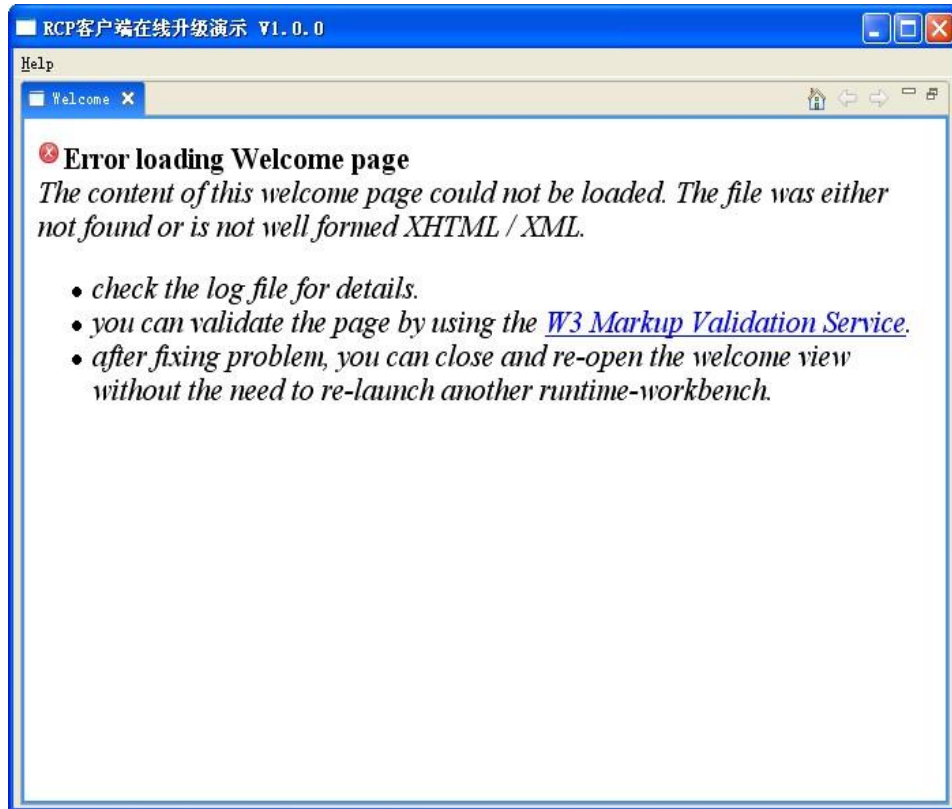
导出的目的地，如图：



点击 Finish 结束。测试下我们导出的文件，双击 eclipse.exe 文件。



启动后如图：



现在构建新的版本

首先在 com.cxd.zxsj 项目里打开文件 plugin.xml 文件，在 Build 页，取消 Custom Build 勾选；在 Overview 页，修改其版本号为 1.0.5，然后修改 IAppConstants 类：

将语句

```
public static final String APPLICATION_TITLE="RCP客户端在线升级演示 V1.0.0";
```

修改为：

```
public static final String APPLICATION_TITLE="RCP客户端在线升级演示 V1.0.5";
```

然后在 Plugin.xml 上单击鼠标右键，在弹出菜单中选择 PDE Tools Create Ant Build,将生成文件 build.xml 文件。同样在 bulid.xml 中添加 encoding="GBK",在切换到 Plugin.xml 文件的 Build 页，勾选“Custom Build”，保存。

然后打开 com.cxd.zxsj.feature 项目 feature.xml 文件，在 Overview 中修改版本为 1.0.3，然后在单击 Synchronize 连接中弹出对话框中的第二项。同步插件。

Overview

General Information

This section describes general information about this plug-in.


ID:	com.cxd.zxsj	
Version:	1.0.3	
Name:	Zxsj Plug-in	
Provider:	CXD	
Platform filter:		
Activator:	com.cxd.zxsj.Activator	Browse...

☒ Activate this plug-in when one of its classes is loaded

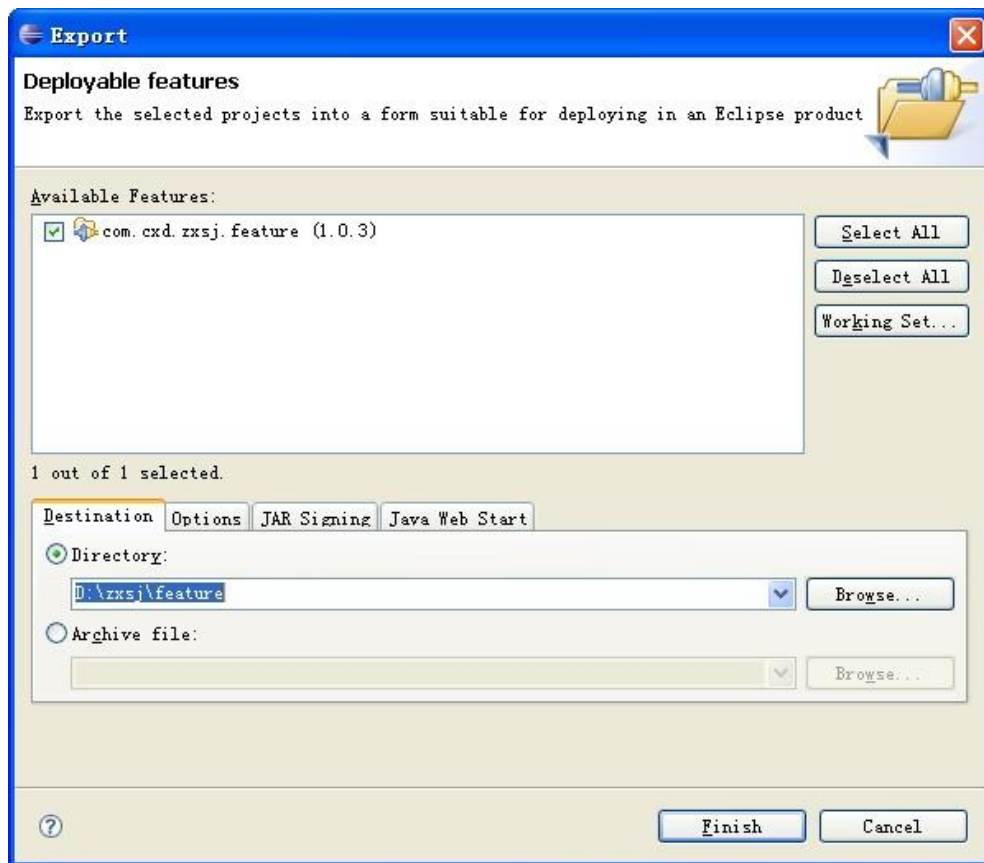
Plug-ins and Fragments

Plug-ins and Fragments

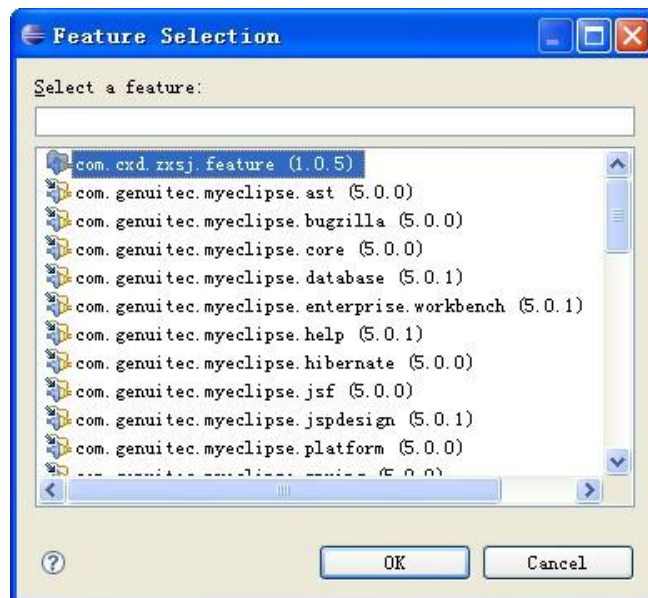
Select plug-ins and fragments that should be packaged in this feature.

 com.cxd.zxsj (1.0.3)	Add...
	Versions...

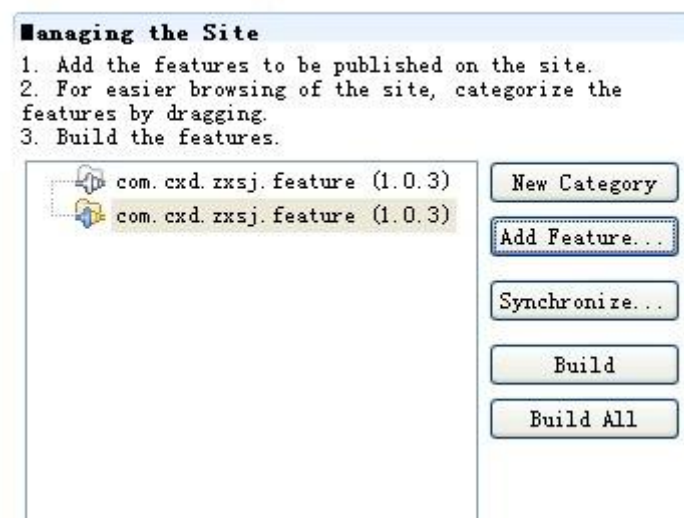
再单击 Overview 页的 Export Wizard，导出目的地，选择自定义文件夹 d:\zxsj\feature.



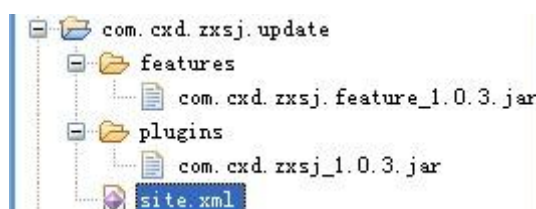
在打开 com.cxd.zxsj.update 项目的 site.xml,单击 Add Feature 按钮,在弹出对话框中选择 com.cxd.zxsj.feature(1.0.3).(注意: 图片不符,以文字为准)



Update Site Map



单击 **Build** 按钮进行构建。构建完成将会出现 **features** 和 **plugins** 文件夹。



在 **com.cxd.zxsj.update** 项目里，我们再添加一个文件夹 **web**，里面包含两个文件，分别是：
文件 **site.css**

```
<STYLE type="text/css">
td.spacer {padding-bottom: 10px; padding-top: 10px;}
.title { font-family: sans-serif; color: #99AACC;}
.bodyText { font-family: sans-serif; font-size: 9pt; color:#000000; }
.sub-header { font-family: sans-serif; font-style: normal; font-weight: bold; font-size: 9pt; color: white;}
.log-text {font-family: sans-serif; font-style: normal; font-weight: lighter; font-size: 8pt; color:black;}
.big-header { font-family: sans-serif; font-style: normal; font-weight: bold; font-size: 9pt; color: white;
border-top:10px solid white;}
.light-row {background:#FFFFFF}
.dark-row {background:#EEEEFF}
.header {background:#99AADD}
#indent {word-wrap : break-word;width :300px;text-indent:10px;}
</STYLE>
```

文件 **site.xml**

```
<xsl:stylesheet version = '1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
xmlns:msxsl="urn:schemas-microsoft-com:xslt">
```

```

<xsl:output method="html" encoding="UTF-8"/>
<xsl:key name="cat" match="category" use="@name"/>
<xsl:template match="/">
  <xsl:for-each select="site">
    <html>
    <head>
      <title>cn.edu.jfcs.update</title>
      <style>@import url("web/site.css");</style>
    </head>
    <body>
      <h1 class="title">cn.edu.jfcs.update</h1>
      <p class="bodyText"><xsl:value-of select="description"/></p>
      <table width="100%" border="0" cellspacing="1" cellpadding="2">
        <xsl:for-each select="category-def">
          <xsl:sort select="@label" order="ascending" case-order="upper-first"/>
          <xsl:sort select="@name" order="ascending" case-order="upper-first"/>
          <xsl:if test="count(key('cat',@name)) != 0">
            <tr class="header">
              <td class="sub-header" width="30%">
                <xsl:value-of select="@name"/>
              </td>
              <td class="sub-header" width="70%">
                <xsl:value-of select="@label"/>
              </td>
            </tr>
            <xsl:for-each select="key('cat',@name)">
              <xsl:sort select="ancestor::feature//@version" order="ascending"/>
              <xsl:sort select="ancestor::feature//@id" order="ascending" case-order="upper-first"/>
              <tr>
                <xsl:choose>
                  <xsl:when test="(position() mod 2 = 1)">
                    <xsl:attribute name="class">dark-row</xsl:attribute>
                  </xsl:when>
                  <xsl:otherwise>
                    <xsl:attribute name="class">light-row</xsl:attribute>
                  </xsl:otherwise>
                </xsl:choose>
                <td class="log-text" id="indent">
                  <xsl:choose>
                    <xsl:when test="ancestor::feature//@label">
                      <a href="{ancestor::feature//@url}"><xsl:value-of
select="ancestor::feature//@label"/></a>
                    </xsl:when>
                    <xsl:otherwise>
                      <br/>
                    </xsl:otherwise>
                  </xsl:choose>
                  <div id="indent">

```

```

                (<xsl:value-of select="ancestor::feature//@id"/> - <xsl:value-of
select="ancestor::feature//@version"/>)
            </div>
        </xsl:when>
        <xsl:otherwise>
            <a href="{ancestor::feature//@url}"><xsl:value-of
select="ancestor::feature//@id"/> - <xsl:value-of select="ancestor::feature//@version"/></a>
        </xsl:otherwise>
        </xsl:choose>
        <br />
    </td>
    <td>
        <table>
            <xsl:if test="ancestor::feature//@os">
                <tr><td class="log-text" id="indent">Operating Systems:</td>
                <td class="log-text" id="indent"><xsl:value-of
select="ancestor::feature//@os"/></td>
                </tr>
            </xsl:if>
            <xsl:if test="ancestor::feature//@ws">
                <tr><td class="log-text" id="indent">Windows Systems:</td>
                <td class="log-text" id="indent"><xsl:value-of
select="ancestor::feature//@ws"/></td>
                </tr>
            </xsl:if>
            <xsl:if test="ancestor::feature//@nl">
                <tr><td class="log-text" id="indent">Languages:</td>
                <td class="log-text" id="indent"><xsl:value-of
select="ancestor::feature//@nl"/></td>
                </tr>
            </xsl:if>
            <xsl:if test="ancestor::feature//@arch">
                <tr><td class="log-text" id="indent">Architecture:</td>
                <td class="log-text" id="indent"><xsl:value-of
select="ancestor::feature//@arch"/></td>
                </tr>
            </xsl:if>
        </table>
    </td>
</tr>
</xsl:for-each>
<tr><td class="spacer"><br/></td><td class="spacer"><br/></td></tr>
</xsl:if>
</xsl:for-each>

```

```

<xsl:if test="count(feature) > count(feature/category)">
<tr class="header">
    <td class="sub-header" colspan="2">
        Uncategorized
    </td>
</tr>
</xsl:if>
<xsl:choose>
<xsl:when test="function-available('msxsl:node-set')">
    <xsl:variable name="rtf-nodes">
        <xsl:for-each select="feature[not(category)]">
            <xsl:sort select="@id" order="ascending" case-order="upper-first"/>
            <xsl:sort select="@version" order="ascending" />
            <xsl:value-of select="."/>
            <xsl:copy-of select="." />
        </xsl:for-each>
    </xsl:variable>
    <xsl:variable name="myNodeSet" select="msxsl:node-set($rtf-nodes)/*"/>
<xsl:for-each select="$myNodeSet">
<tr>
    <xsl:choose>
    <xsl:when test="position() mod 2 = 1">
    <xsl:attribute name="class">dark-row</xsl:attribute>
    </xsl:when>
    <xsl:otherwise>
    <xsl:attribute name="class">light-row</xsl:attribute>
    </xsl:otherwise>
    </xsl:choose>
    <td class="log-text" id="indent">
        <xsl:choose>
        <xsl:when test="@label">
            <a href="{@url}"><xsl:value-of select="@label"/></a>
            <br />
            <div id="indent">
                (<xsl:value-of select="@id"/> - <xsl:value-of select="@version"/>)
            </div>
        </xsl:when>
        <xsl:otherwise>
            <a href="{@url}"><xsl:value-of select="@id"/> - <xsl:value-of select="@version"/></a>
        </xsl:otherwise>
        </xsl:choose>
        <br /><br />
    </td>
</tr>

```



```

<table>
  <xsl:if test="@os">
    <tr><td class="log-text" id="indent">Operating Systems:</td>
      <td class="log-text" id="indent"><xsl:value-of select="@os"/></td>
    </tr>
  </xsl:if>
  <xsl:if test="@ws">
    <tr><td class="log-text" id="indent">Windows Systems:</td>
      <td class="log-text" id="indent"><xsl:value-of select="@ws"/></td>
    </tr>
  </xsl:if>
  <xsl:if test="@nl">
    <tr><td class="log-text" id="indent">Languages:</td>
      <td class="log-text" id="indent"><xsl:value-of select="@nl"/></td>
    </tr>
  </xsl:if>
  <xsl:if test="@arch">
    <tr><td class="log-text" id="indent">Architecture:</td>
      <td class="log-text" id="indent"><xsl:value-of select="@arch"/></td>
    </tr>
  </xsl:if>
</table>
</td>
</tr>
</xsl:for-each>
</xsl:when>
<xsl:otherwise>
<xsl:for-each select="feature[not(category)]">
<xsl:sort select="@id" order="ascending" case-order="upper-first"/>
<xsl:sort select="@version" order="ascending" />
<tr>
  <xsl:choose>
    <xsl:when test="count(preceding-sibling::feature[not(category)]) mod 2 = 1">
      <xsl:attribute name="class">dark-row</xsl:attribute>
    </xsl:when>
    <xsl:otherwise>
      <xsl:attribute name="class">light-row</xsl:attribute>
    </xsl:otherwise>
  </xsl:choose>
  <td class="log-text" id="indent">
    <xsl:choose>
      <xsl:when test="@label">
        <a href="{@url}"><xsl:value-of select="@label"/></a>
      </xsl:when>
    </xsl:choose>
  </td>
</tr>

```

```

        <div id="indent">
            (<xsl:value-of select="@id"/> - <xsl:value-of select="@version"/>)
        </div>
    </xsl:when>
    <xsl:otherwise>
        <a href="{@url}"><xsl:value-of select="@id"/> - <xsl:value-of select="@version"/></a>
    </xsl:otherwise>
</xsl:choose>
<br /><br />
</td>
<td>
    <table>
        <xsl:if test="@os">
            <tr><td class="log-text" id="indent">Operating Systems:</td>
            <td class="log-text" id="indent"><xsl:value-of select="@os"/></td>
            </tr>
        </xsl:if>
        <xsl:if test="@ws">
            <tr><td class="log-text" id="indent">Windows Systems:</td>
            <td class="log-text" id="indent"><xsl:value-of select="@ws"/></td>
            </tr>
        </xsl:if>
        <xsl:if test="@nl">
            <tr><td class="log-text" id="indent">Languages:</td>
            <td class="log-text" id="indent"><xsl:value-of select="@nl"/></td>
            </tr>
        </xsl:if>
        <xsl:if test="@arch">
            <tr><td class="log-text" id="indent">Architecture:</td>
            <td class="log-text" id="indent"><xsl:value-of select="@arch"/></td>
            </tr>
        </xsl:if>
    </table>
</td>
</tr>
</xsl:for-each>
</xsl:otherwise>
</xsl:choose>
</table>
</body>
</html>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>

```

在创建一个 index.html 文件，在工程目录下（注意：不是 web 文件夹下）

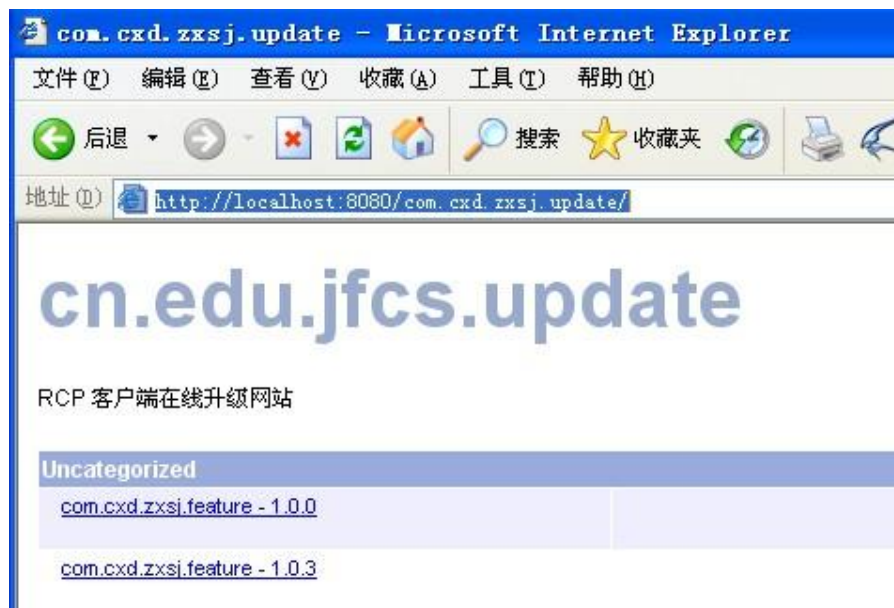
```
<html>
<head>
<title>com.cxd.zxsj.update</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<style>@import url("web/site.css");</style>
<script type="text/javascript">
var returnval = 0;
var stylesheet, xmlFile, cache, doc;
function init(){
    // NSCP 7.1+ / Mozilla 1.4.1+ / Safari
    // Use the standard DOM Level 2 technique, if it is supported
    if (document.implementation && document.implementation.createDocument) {
        xmlFile = document.implementation.createDocument("", "", null);
        stylesheet = document.implementation.createDocument("", "", null);
        if (xmlFile.load){
            xmlFile.load("site.xml");
            stylesheet.load("web/site.xsl");
        } else {
            alert("Document could not be loaded by browser.");
        }
        xmlFile.addEventListener("load", transform, false);
        stylesheet.addEventListener("load", transform, false);
    }
    //IE 6.0+ solution
    else if (window.ActiveXObject) {
        xmlFile = new ActiveXObject("msxml2.DOMDocument.3.0");
        xmlFile.async = false;
        xmlFile.load("site.xml");
        stylesheet = new ActiveXObject("msxml2.FreeThreadedDOMDocument.3.0");
        stylesheet.async = false;
        stylesheet.load("web/site.xsl");
        cache = new ActiveXObject("msxml2.XSLTemplate.3.0");
        cache.stylesheet = stylesheet;
        transformData();
    }
}
// separate transformation function for IE 6.0+
function transformData(){
    var processor = cache.createProcessor();
    processor.input = xmlFile;
    processor.transform();
    data.innerHTML = processor.output;
```

```

}
// separate transformation function for NSCP 7.1+ and Mozilla 1.4.1+
function transform(){
    returnval+=1;
    if (returnval==2){
        var processor = new XSLTProcessor();
        processor.importStylesheet(stylesheets);
        doc = processor.transformToDocument(xmlFile);
        document.getElementById("data").innerHTML = doc.documentElement.innerHTML;
    }
}
</script>
</head>
<body onload="init();">
<!--[insert static HTML here]-->
<div id="data"><!-- this is where the transformed data goes --></div>
</body>
</html>

```

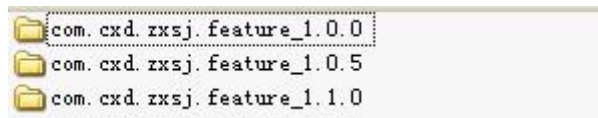
然后将整个工程文件复制到 D:\tomcat-6.0\webapps 文件下,启动服务,我们访问网站如下:



然后我们启动客户端程序,单击菜单中的在线升级,他会出现如下问题:



这是什么原因呢，我在看下我们客户端程序的目录，原来他缺少了一个文件创建 features\com.cxd.zxsj.feature_1.0.0\feature.xml 文件，使我们疏忽，在没有记录我们使用版本的文件。（注意：具体这里怎么出现这个文件我也不是很清楚，但是你要是缺少了这个文件就会提示这个问题，features 文件下的文件要与服务器上的存在的 jar 文件的版本要一致，我们升级成功，会自动添加使用版本的文件，就不需要我们操心了）



Feature.xml 的内容

```
<?xml version="1.0" encoding="UTF-8"?>
<feature
    id="com.cxd.zxsj.feature"
    label="Feature Feature"
    version="1.0.0"
    provider-name="SINO">

    <description url="http://www.example.com/description">
        [Enter Feature Description here.]
    </description>

    <copyright url="http://www.example.com/copyright">
        [Enter Copyright Description here.]
    </copyright>

    <license url="http://www.example.com/license">
        [Enter License Description here.]
    </license>

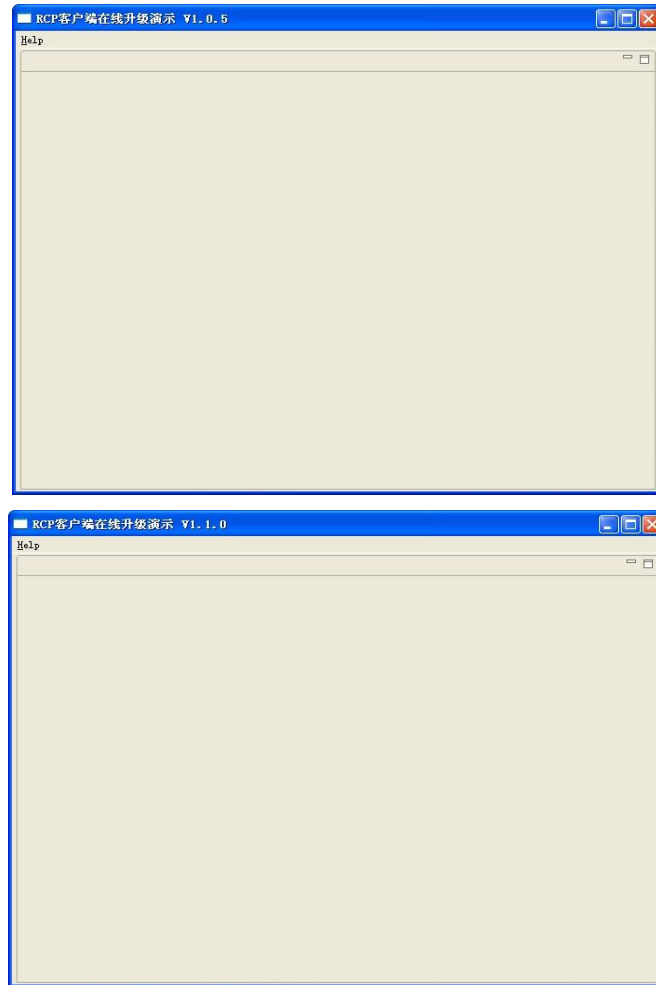
    <url>
        <update label="RCP 客户端在线升级" url="http://localhost:8080/com.cxd.zxsj.update"/>
    </url>

    <plugin
        id="com.sino.hxt.dzjy"
        download-size="0"
        install-size="0"
```

```
version="1.0.0"  
unpack="false"/>
```

</feature>

然后我们再实验，就不会出现了刚才出现的问题。然后我们多次实验下，修改版本，再次升级，都不会出现问题了。



最后我在这里说明下，如果我们的客户端更新次数比较多的话，我们的客户端文件会越来越多，越来越大，但还是有一点好处就是，如果出错，我们可以删除最新的文件，我们就可以用以前的版本了。

本人没有用上面的升级方法，我的方法是：向服务器传递当前使用的版本，事发后需要更新，然后是利用 FTP 下载最新的程序包，程序包是利用 NSIS 打包，将所有更新文件打包一个 exe 文件，执行它，就替换掉我们的主架包，替换的时候客户端需要关闭，你需要另外写个升级程序，来调用它。这样就保持了文件不会一直增大。