eclipse插件开发节点说明

zjl

2018-10-08T00:00:00.000Z

### 一级节点 节点功能

* 视图说明
* 节点说明

#### 节点关系和继承

import 关系图 #### 节点解析 {#节点解析 } ##### 子节点1 {#子节点1 }

##### 子节点2

###### 节点 - class 实现类

import 代码

# 节点详细

一段描述

## 系统菜单、工具条和弹出菜单

一段描述一下有几个节点，或者分个类

### org.eclipse.ui.menus 菜单栏和工具条

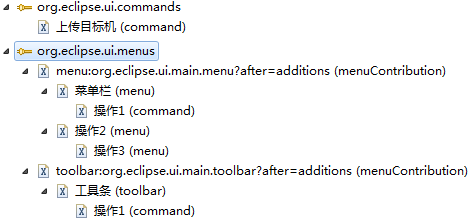
* 视图说明



* 节点说明 此节点包含 菜单栏 和工具条， 都是通过自子节点[**menuContribution**]的[*locationURI*]来控制的。详情见下文

#### 节点关系和继承

* 关系图



* 关系说明
  + [*menuContribution*] 可设置子节点是菜单栏或工具条
  + [*menu*] 菜单栏节点，可多层**嵌套**
  + [*toolbar*] 工具条节点，实现类和 *command* 一样。

#### 节点解析

##### 子节点:menuContribution

* 节点参数

locationURI\*://表示菜单位置  
// menu:org.eclipse.ui.main.menu?after=additions 表示此节点为菜单栏，after的值是表示菜单的位置。  
// toolbar:org.eclipse.ui.main.toolbar?after=additions 表示此节点为工具条，具体信息见视图说明  
class:这里的class不是必要参数，一般不设置

##### 子节点: menu

* 节点参数

lable\*: 菜单 //这里的lable就是显示在菜单栏的具体菜单名，用户通过此进行交互  
id: com.xxx.menu // 非必须  
commandId: // 默认command 链接，非必须

##### 子节点: toolbar

* 节点参数

lable: 工具 //这里的lable就是显示在工具栏的具体菜单名，用户通过此进行交互  
id\*: com.xxx.toolbar // 必须id

##### 子节点: command

* 节点参数

|  |  |  |
| --- | --- | --- |
| 节点标识 | 类型 | 说明 |
| **commandId** | 工具 | 这里的lable就是显示在工具栏的具体菜单名，用户通过此进行交互 |

**commandId:** 工具 //

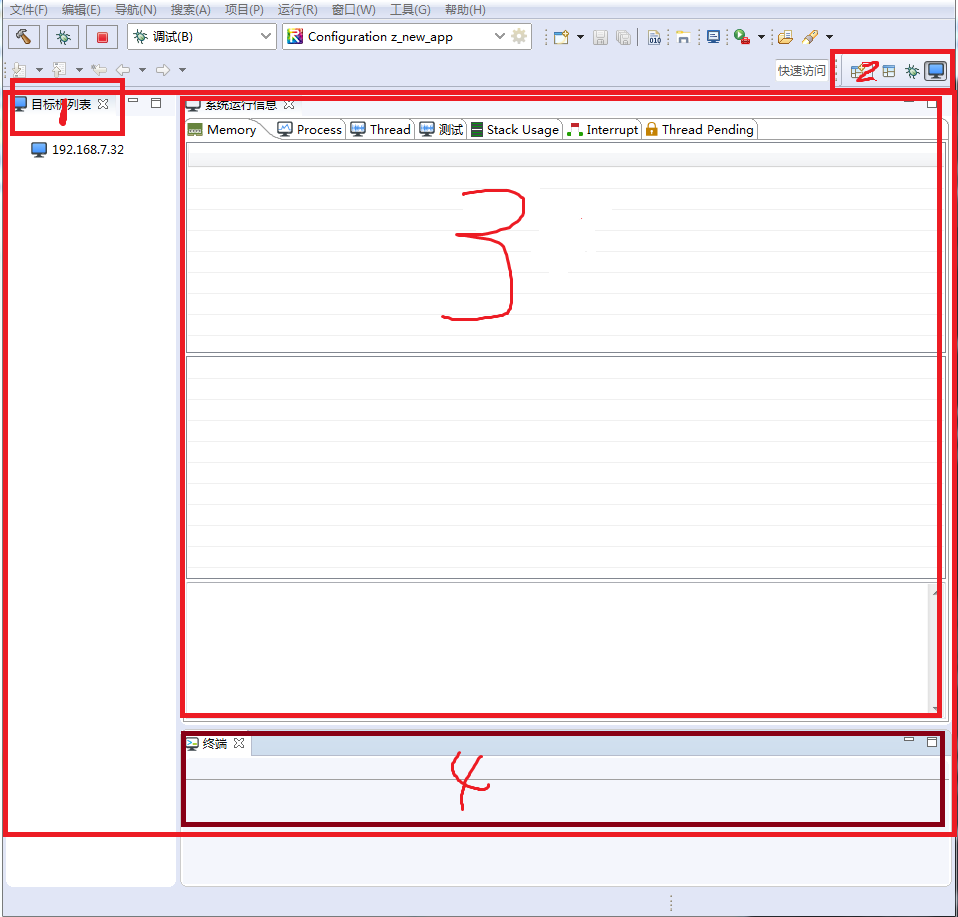
commandId: 工具 //这里的lable就是显示在工具栏的具体菜单名，用户通过此进行交互  
id\*: com.xxx.toolbar // 必须id

###### 节点 - class 实现类

import 代码

### 弹出菜单

## 工作台视图

* 视图说明 

### org.eclipse.ui.perspective\* 工作台位置布局

#### 子节点:perspective

* 节点说明： 这是一个后台管理类节点，负责管理需要实现的具体视图的对应Layout，初始化透视图。如例图所示。1、3、4位置是独立的子view，并在节点[**org.eclipse.ui.views**]\*中实现。所有子View都通过2实现处按钮实现快速切换功能。
* 关键参数：

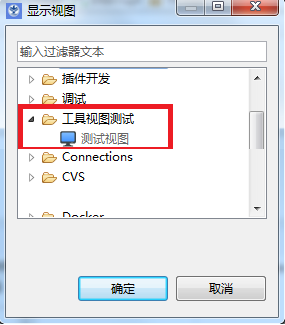
id\*:com.packagename.DemoPerspectives // 建议和class名一样  
name\*:测试视图   
class\*:com.packagename.DemoPerspectives  
icon:icons/device.png // 建议在工程目录下建立一个 [icons] 文件夹，存储所有图标文件。建议像素大小为16x16

* class实现类: DemoPerspectives.java

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* 文 件 名: Perspectives.java  
 \* 创 建 人: macro  
 \* 文件创建日期: 2018 年 1 月 12 日  
 \* 描 述: org.eclipse.ui.perspective扩展点perspective属性调用该功能，初始化透视图  
 \* 文件修改记录：  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
package com.xxx;  
   
import org.eclipse.ui.IFolderLayout;  
import org.eclipse.ui.IPageLayout;  
import org.eclipse.ui.IPerspectiveFactory;  
import org.eclipse.ui.wizards.newresource.BasicNewFileResourceWizard;  
import org.eclipse.ui.wizards.newresource.BasicNewFolderResourceWizard;  
   
public class Perspectives implements IPerspectiveFactory {  
   
 public Perspectives() {  
 super();  
 }  
   
 @Override  
 public void createInitialLayout(IPageLayout layout) {  
 defineLayout(layout); // 增加透视图的布局和视图  
 // defineActions(layout); // 增加透视图的动作和快捷  
 }  
   
 // 增加透视图的布局和视图  
 public void defineLayout(IPageLayout paramIPageLayout) {  
 /\* paramIPageLayout 是方位布局，即上中下左右\*/  
   
 // 设置设置编辑器区域可见不可见  
 paramIPageLayout.setEditorAreaVisible(false);  
 // 获取编辑器区域  
 String strEditorArea = paramIPageLayout.getEditorArea();  
   
 /\* 子view 布局 \*/  
 // 左边 区域 布局  
 IFolderLayout leftFolderLayout = paramIPageLayout.createFolder("leftFolder", 1, 0.18F,  
 strEditorArea); // left  
 // 添加子view的节点 id (这里是此插件节点的布局，而不是class的类名)  
 paramIPageLayout.addView("com.acce.hyide.product.gadget.navigator.view", 3, 0.65F,  
 "leftFolder"); // left-bottom  
   
 // 底部 区域布局  
 IFolderLayout bottomIFolderLayout = paramIPageLayout.createFolder("bottomFolder", 4, 0.75F,  
 strEditorArea); // bottom  
 // 添加子view的节点 id  
 bottomIFolderLayout.addView("com.acce.hyide.product.gadget.terminal.view");  
 bottomIFolderLayout.addView("com.acce.hyide.product.gadget.test.view");  
   
 // 中间区域 布局  
 IFolderLayout middleIFolderLayout = paramIPageLayout.createFolder("MiddleFolder", 2, 0.75F,  
 strEditorArea); // Middle  
 // 添加子view的节点 id  
 middleIFolderLayout.addView("com.acce.hyide.product.gadget.information.view");  
 middleIFolderLayout.addView("com.acce.hyide.product.gadget.ftp.view");  
   
 }  
   
 // 增加透视图的动作和快捷  
 public void defineActions(IPageLayout layout) {  
   
 }  
   
}

### org.eclipse.ui.views

* 视图说明

打开方式： 菜单栏：窗口>显示视图>其他>category名   
#### 节点关系 {#节点关系 }

category 是分类文件夹

  
view 是视图实现节点，被category分类 #### 节点解析 {#节点解析-2 }

##### 子节点 category：

id\*:com.xxx.view.category  
name\*:工具视图测试

##### 子节点 view：

id\*:com.packagename.DemoRemoteSystemInfoView  
name\*:工具视图测试  
class\*:com.packagename.DemoRemoteSystemInfoView  
category:com.xxx.view.category // 添加到分类category  
icon:

###### view - class实现类

DemoRemoteSystemInfoView.java

package com.xxx;  
   
import java.io.File;  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.List;  
   
import org.eclipse.jface.dialogs.MessageDialog;  
import org.eclipse.jface.viewers.TableViewer;  
import org.eclipse.swt.SWT;  
import org.eclipse.swt.custom.CTabFolder;  
import org.eclipse.swt.custom.CTabItem;  
import org.eclipse.swt.custom.SashForm;  
import org.eclipse.swt.custom.StyleRange;  
import org.eclipse.swt.custom.StyledText;  
import org.eclipse.swt.events.MenuEvent;  
import org.eclipse.swt.events.MenuListener;  
import org.eclipse.swt.events.SelectionAdapter;  
import org.eclipse.swt.events.SelectionEvent;  
import org.eclipse.swt.events.SelectionListener;  
import org.eclipse.swt.layout.GridData;  
import org.eclipse.swt.layout.GridLayout;  
import org.eclipse.swt.widgets.Button;  
import org.eclipse.swt.widgets.Combo;  
import org.eclipse.swt.widgets.Composite;  
import org.eclipse.swt.widgets.Control;  
import org.eclipse.swt.widgets.DirectoryDialog;  
import org.eclipse.swt.widgets.Display;  
import org.eclipse.swt.widgets.Event;  
import org.eclipse.swt.widgets.Group;  
import org.eclipse.swt.widgets.Listener;  
import org.eclipse.swt.widgets.Menu;  
import org.eclipse.swt.widgets.MenuItem;  
import org.eclipse.swt.widgets.MessageBox;  
import org.eclipse.swt.widgets.Shell;  
import org.eclipse.swt.widgets.Table;  
import org.eclipse.swt.widgets.TableColumn;  
import org.eclipse.swt.widgets.TableItem;  
import org.eclipse.swt.widgets.Text;  
import org.eclipse.swt.widgets.Tree;  
import org.eclipse.swt.widgets.TreeItem;  
import org.eclipse.ui.part.ViewPart;  
   
public class RemoteSystemInfoView2 extends ViewPart {  
 // 分页标签的主页面  
 public CTabFolder tabFolder;  
 // 页面子标签项  
 public CTabItem tbtmNewItem;  
 public CTabItem tbtmNewItem\_1;  
 // 子标签视图  
 public TableViewer tableViewer;  
 public TableViewer tableViewer\_1;  
 public TableViewer tableViewer\_2;  
 // 标签组件  
 // 组件 ： 列表  
 public Table table;  
 public Table table\_1;  
 public Table table\_2;  
 // 组件 ： 文本框  
 public Text text;  
   
 // 布局 ：Group，可作为一个容器或页面，添加其他组件、布局等  
 public Group group;  
 // 组件 ： 下拉框  
 public Combo combo;  
 // 组件 ： 按键  
 public Button button;  
 // 组件,和 Text 差不多，和设置字体和颜色  
 public StyledText styledText;  
   
 // 树状图  
 public Tree tree;  
 private TreeItem treeItem;  
 private TreeItem treeItem2;  
 private MenuItem menuItem1;  
 private MenuItem menuItem2;  
   
 // 简单消息对话框  
 MessageBox messagebox;  
   
   
 @Override  
 public void createPartControl(Composite parent) {  
 // 容器初始化 (所有Tab页面都在此容器下)  
 tabFolder = new CTabFolder(parent, SWT.BORDER);  
 tabFolder.setSimple(false);  
   
 // 添加 UI  
 addTableView\_1();  
 addTableView\_2();  
 addGroupView\_3(parent);  
 addTreeView\_4(parent);  
   
 // 添加右键菜单  
 addMenu(table);  
   
 }  
 // 添加标签页1 方式1  
 // 布局1  
 void addTableView\_1()  
 {  
 // 标签1  
 tbtmNewItem = new CTabItem(tabFolder, SWT.NONE);  
 //tbtmNewItem.setImage(DeviceUtil.getImage("icons/Memory.png")); //$NON-NLS-1$  
 tbtmNewItem.setText("标签1");  
   
 // 标签的容器，在此容器内添加组件  
 SashForm sashForm = new SashForm(tabFolder, SWT.VERTICAL);  
 tbtmNewItem.setControl(sashForm);  
   
 // (列表视图)标签页面的组件  
 tableViewer = new TableViewer(sashForm, SWT.BORDER | SWT.FULL\_SELECTION);  
 table = tableViewer.getTable();  
 table.setHeaderVisible(true);  
 table.setLinesVisible(true);  
 // 添加列表项的监听  
 table.addListener(SWT.MouseDoubleClick, new Listener() {  
 @Override  
 public void handleEvent(org.eclipse.swt.widgets.Event event) {  
 TableItem tableitm;  
 try {  
 tableitm = table.getSelection()[0];// 获取选中的行  
 putMsg("" + tableitm.getText());  
 } catch (Exception e) {  
 return;  
 }  
 File fitem = (File) tableitm.getData();  
 if (fitem.toString().equals("..")) { //$NON-NLS-1$  
 File file = new File(strlocpath);  
 file = file.getParentFile();  
 listLocalFiles(file, table);  
 return;  
 }  
 if (fitem.isDirectory()) {  
 try {  
 listLocalFiles(fitem, table);  
 } catch (Exception e) {  
 // 遇到无权限的文件夹则返回他上级目录  
 MessageDialog.openInformation(Display.getCurrent().getActiveShell(),  
 "title",  
 "无权限的文件夹");  
 File file = fitem.getParentFile();  
 listLocalFiles(file, table);  
 return;  
 }  
 } else {  
 //if (!rimplementation.isListener) {  
 // return;  
 //}  
 //logMessage(Messages.RemoteSystemFtpImplementation\_Startuploading);  
 //rimplementation.Upload(filelist);  
 }  
 }  
 });  
   
 tableViewer\_2 = new TableViewer(sashForm, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_2 = tableViewer\_2.getTable();  
 table\_2.setToolTipText("table\_2");  
 table\_2.setHeaderVisible(false);  
 table\_2.setLinesVisible(true);  
 // (文本框视图)标签页面的组件  
 text = new Text(sashForm, SWT.BORDER | SWT.WRAP | SWT.V\_SCROLL | SWT.READ\_ONLY);  
 //text.setFont(DeviceUtil.getFont());  
 text.setBackground(Display.getCurrent().getSystemColor(SWT.COLOR\_WHITE));  
 text.setEditable(true);  
   
 // sashForm 的权重  
 sashForm.setWeights(new int[] { 161, 170, 103 });  
 }  
 // 添加标签页2 方式2  
 // 布局2  
 void addTableView\_2()  
 {  
 /\* 方式2 \*/  
 Composite compsoite = new Composite(tabFolder, SWT.NONE);  
 tbtmNewItem\_1 = new CTabItem(tabFolder, SWT.NONE);  
 //tbtmNewItem\_1.setImage(DeviceUtil.getImage("icons/Process.png"));  
 tbtmNewItem\_1.setText("标签2");  
 // 组件1  
 tableViewer\_1 = new TableViewer(compsoite, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_1 = tableViewer\_1.getTable();  
 // 组件布局方式  
 GridData gd\_tablea = new GridData(SWT.FILL, SWT.FILL, true, true, 1, 1);  
 gd\_tablea.heightHint = 220;  
 // 添加布局到 table  
 table\_1.setLayoutData(gd\_tablea);  
 table\_1.setHeaderVisible(true);  
 table\_1.setLinesVisible(true);  
 // 添加UI内容到 CTabItem  
 tbtmNewItem\_1.setControl(compsoite);  
 GridLayout layout = new GridLayout();  
 layout.marginWidth = 0;  
 layout.marginHeight = 0;  
 // 添加布局到 Composite  
 compsoite.setLayout(layout);  
   
 }  
 // 布局3 另起一组  
 void addGroupView\_3(Composite parent)  
 {  
 group = new Group(parent, SWT.NULL);  
 group.setText("Group 布局");  
 GridLayout gridLayout = new GridLayout(2, true);  
 gridLayout.marginHeight = 0;  
 group.setLayout(gridLayout);  
 group.setLayoutData(new GridData(GridData.FILL\_HORIZONTAL));  
   
 /\* 下拉框 \*/  
 combo = new Combo(group, SWT.FLAT);  
 combo.setLayoutData(new GridData(SWT.FILL, SWT.CENTER, true, false, 1, 1));  
 combo.setText("默认数据");  
 combo.add("下拉数据1", 0);//添加到最前  
 combo.add("下拉数据2", 0);//不会覆盖  
 // String[] data = combo.getItems()获得所有项  
 // 为该下拉框选中事件  
 combo.addSelectionListener(new SelectionAdapter() {  
 public void widgetSelected(SelectionEvent e) {  
 String strcomnocontent = combo.getText().trim();// 选择的数据  
 MessageDialog  
 .openInformation(  
 Display.getCurrent().getActiveShell(),  
 "标题",  
 "内容:" + strcomnocontent);  
 }  
 });  
   
 /\* 按键 \*/  
 button = new Button(group, SWT.PUSH);  
 //button.setImage(DeviceUtil.getImage("icons/up.png"));  
 button.addSelectionListener(new SelectionAdapter() {  
 @Override  
 public void widgetSelected(SelectionEvent e) {  
 // 打开文件资源管理器  
 DirectoryDialog dialog = new DirectoryDialog(parent.getShell());  
 // 返回选中的文件路径  
 String strpath = dialog.open();  
 if (strpath == null) {  
 return;  
 }  
 File file = new File(strpath);  
 if (file.getParentFile() == null) {  
 //buttonUpLocalDir.setEnabled(false);  
 } else {  
 listLocalFiles(file, table);  
 putStyledText(file.toPath().toString());  
 }  
 }  
 });  
 //  
 styledText = new StyledText(group, SWT.BORDER | SWT.V\_SCROLL);  
 styledText.setSize(200, 400);  
 styledText.setEditable(false);  
 }  
   
 // 给tab页面添加右键菜单  
 // 添加 Tree 视图  
 void addTreeView\_4(Composite parent)  
 {  
 this.tree = new Tree(parent, SWT.NONE);  
 // 添加双点击事件  
 this.tree.addSelectionListener(new SelectionListener() {  
 // 双击时  
 @Override  
 public void widgetDefaultSelected(SelectionEvent e) {  
 putMsg("TreeView点击响应" + tree.getSelection()[0].getData().toString() );// 获取选中信息  
 }  
   
 // 获得(单击)焦点时  
 @Override  
 public void widgetSelected(SelectionEvent e) {  
 //putMsg("TreeView点击响应widgetSelected");  
 }  
 });  
   
 // 添加右键菜单  
 addMenu(tree);  
   
 treeItem = new TreeItem(tree, SWT.None);  
   
 treeItem.setText("subItem");  
 treeItem.setData("subItemData");  
 //treeItem.setImage(DeviceUtil.getImage("icons/connect.png")); //$NON-NLS-1$  
   
 treeItem2 = new TreeItem(treeItem, SWT.None);  
 treeItem2.setText("subItem2");  
 treeItem2.setData("subItemData2");  
 //treeItem2.setImage(DeviceUtil.getImage("icons/connect.png")); //$NON-NLS-1$  
   
 }  
   
 // 添加右键菜单 给Table、Tree等继承Control的类都行  
 private void addMenu(Control table) {  
 // 添加右键菜单  
 Menu memorymenu = new Menu(table);  
   
 // 添加操作项  
 MenuItem memoryItem = new MenuItem(memorymenu, SWT.PUSH);  
 // 操作项名称  
 memoryItem.setText("操作项1");  
 // 添加操作响应监听  
 memoryItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread memorymenu = new Thread() {  
 public void run() {  
 //操作响应  
 // 列表头  
 String[] strtitle = {"strtitle1","strtitle2"};  
 // 列表数据行  
 String[][] strcontent = {{"strcontent1", "strcontent2"},{"strcontent3", "strcontent4"}};  
 setTableData\_table(strtitle, strcontent);  
 }  
 };  
 memorymenu.setName("memorymenu");  
 memorymenu.start();  
 }  
 });  
   
 // 创建一个分割线  
 MenuItem separator1 = new MenuItem(memorymenu, SWT.SEPARATOR);  
   
 // 添加其他操作项  
 MenuItem connectItem = new MenuItem(memorymenu, SWT.PUSH);  
 connectItem.setText("操作项2");  
 connectItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
   
 }  
 });  
   
 // 菜单监听  
 memorymenu.addMenuListener(new MenuListener() {  
 // 菜单显示前  
 @Override  
 public void menuShown(MenuEvent e) {  
 // 设置操作项不可显示  
 memoryItem.setEnabled(true);  
 connectItem.setEnabled(false);  
 // 这里可以设置规则，控制操作项的显示逻辑  
 // \*\*\*  
 }  
   
 // 菜单隐藏后  
 @Override  
 public void menuHidden(MenuEvent e) {  
 }  
 });  
 table.setMenu(memorymenu);  
   
 /\* MenuItem 可重复添加 \*/  
 }  
   
 /\*\*  
 \* StyledText 显示数据  
 \* @param message  
 \*/  
 public void putStyledText(String message) {  
 styledText.getDisplay().asyncExec(new Runnable() {  
 @Override  
 public void run() {  
 StyleRange styleRange1 = new StyleRange();  
 styleRange1.start = styledText.getCharCount();  
 styleRange1.length = message.length();  
 styleRange1.fontStyle = SWT.BOLD;  
 Shell shell = new Shell();  
 styleRange1.foreground = shell.getDisplay().getSystemColor(SWT.COLOR\_RED);// Display.getCurrent().getActiveShell()  
 styledText.append(message + "\r\n"); //$NON-NLS-1$  
 styledText.setStyleRange(styleRange1);  
 styledText.setSelection(styledText.getCharCount());  
 }  
 });  
 }  
   
 public static long kb = 1024;  
 public static long mb = kb \* 1024;  
 public static long gb = mb \* 1024;  
 public static DecimalFormat df = new DecimalFormat("#,###"); //$NON-NLS-1$  
 String strlocpath = "";// 当前地址  
 /\*  
 \* 在table显示 本地路径 数据  
 \*/  
 private void listLocalFiles(File selDisk, Table table) {  
 if (selDisk == null || selDisk.isFile()) {  
 return;  
 }  
 strlocpath = selDisk.toString();  
 combo.setText(selDisk.toString());// 当前地址  
 table.removeAll();// 移除旧数据  
 File[] listFiles = selDisk.listFiles(); // 获取磁盘文件列表  
 List<File> fileList = new ArrayList<File>();  
 for (File f : listFiles) {  
 fileList.add(f);  
 }  
 Collections.sort(fileList, new Comparator<File>() {  
 @Override  
 public int compare(File o1, File o2) {  
 if (o1.isDirectory() && o2.isFile()) {  
 return -1;  
 }  
 if (o1.isFile() && o2.isDirectory()) {  
 return 1;  
 }  
 return o1.getName().compareTo(o2.getName());  
 }  
 });  
 // 上层目录  
 if (selDisk.getParentFile() != null) {  
   
 File file = new File(".."); //$NON-NLS-1$  
 TableItem itema = new TableItem(table, SWT.NONE);  
 //itema.setImage(DeviceUtil.getImage("icons/newfolder.png")); //$NON-NLS-1$  
 itema.setText(new String[] { "..", "", "" }); //$NON-NLS-1$ //$NON-NLS-2$ //$NON-NLS-3$  
 itema.setData(file);  
 } else {  
   
 }  
   
 // 遍历磁盘根文件夹的内容，添加到表格中  
 for (File file : fileList) {  
 TableItem itemsa = new TableItem(table, SWT.NONE);  
 String length = ""; // 获取文件大小 //$NON-NLS-1$  
 long lsize = file.length();  
 if (lsize >= gb) {  
 length = String.format("%.1fGB", (float) lsize / gb) + " (" + df.format(lsize) + ")"; //$NON-NLS-1$ //$NON-NLS-2$ //$NON-NLS-3$  
 } else if (lsize >= mb) {  
 float f = (float) lsize / mb;  
 length = String.format(f > 100 ? "%.0fMB" : "%.1fMB", f) + " (" + df.format(lsize) + ")"; //$NON-NLS-1$ //$NON-NLS-2$ //$NON-NLS-3$ //$NON-NLS-4$  
 } else if (lsize >= kb) {  
 float f = (float) lsize / kb;  
 length = String.format(f > 100 ? "%.0fKB" : "%.1fKB", f) + " (" + df.format(lsize) + ")"; //$NON-NLS-1$ //$NON-NLS-2$ //$NON-NLS-3$ //$NON-NLS-4$  
 } else {  
 length = String.format("%dB", lsize); //$NON-NLS-1$  
 }  
 if (file.isDirectory()) { // 显示文件夹标志  
 length = ""; //$NON-NLS-1$  
 //itemsa.setImage(DeviceUtil.getIcon(file));  
 } else {  
 //itemsa.setImage(DeviceUtil.getIcon(file));  
 }  
 // 获取文件的最后修改日期  
 //SimpleDateFormat myFmt = new SimpleDateFormat(  
 //ConstantClass.RemoteSystemFtpView\_DateFormat);  
 //String modifDate = myFmt.format(new Date(file.lastModified()));  
 itemsa.setText(new String[] { file.getName(), length, "日期" });  
 itemsa.setData(file);  
 }  
 }  
   
 // 添加 数据奥 table 列表  
 /\*添加数据到 tab 列表中\*/  
 // 给table 添加数据  
 public void setTableData\_table(String[] strtitle, String[][] strcontent) {  
 setTableData(table, tableViewer, strcontent, strtitle);  
 }  
   
 /\*\*  
 \* 给table添加信息公共方法  
 \*  
 \* @param table  
 \* @param tableViewer  
 \*/  
 void setTableData(Table table, TableViewer tableViewer, String[][] strcontent,  
 String[] strtitle) {  
 Display.getDefault().syncExec(new Runnable() {  
 @Override  
 public void run() {  
 // 记录table当前位置  
 int iscrollbarlocation = table.getTopIndex();  
 int iSelection = table.getSelectionIndex();  
 // 清空table内容  
 table.removeAll();  
   
 // 重新加载table内容  
 // if(tableViewer.getTable().getColumnCount()==0){  
 TableColumn[] columns = table.getColumns(); // 清除表格原有表头  
 for (TableColumn tc : columns) {  
 tc.dispose();  
 }  
 // 添加表头  
 for (int i = 0; i < strtitle.length; i++) {  
 TableColumn TableContent = new TableColumn(table, SWT.NONE, i);  
 TableContent.setWidth(140);  
 TableContent.setText(strtitle[i]);  
 }  
 // }  
   
 for (int j = 0; j < strcontent.length; j++) {  
 // 添加数据  
 TableItem TableTitle = new TableItem(table, SWT.NONE);  
 String strcont[] = new String[strcontent[j].length];  
 for (int k = 0; k < strcontent[j].length; k++) {  
 strcont[k] = strcontent[j][k];  
 }  
 TableTitle.setText(strcont);  
 }  
 // 设置当前选中信息位置  
 table.setTopIndex(iscrollbarlocation);  
 }  
 });  
 }  
   
 @Override  
 public void setFocus() {  
 // TODO Auto-generated method stub  
 }  
 void putMsg(String msg)  
 {  
 MessageDialog  
 .openInformation(  
 Display.getCurrent().getActiveShell(),  
 "标题",  
 msg);  
 }  
   
}

* 说明 还有部分节点未涉及，暂不说明

## 新建工程

## 调试核心

## 目标程序下载

## 集成TFTP工具

## 串口访问工具

## 目标机查看和管理工具

## 可视化寄存器修改

## 系统级调试

## 任务级调试

## 代码规则检查工具

## 错误解析

## 通用界面相关

## 通用工具相关

### 1 菜单类

#### 1.1、menu

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* 中国航天科工集团三院三部  
 \* 海鹰集成开发环境  
 \* 文 件 名: RemoteSystemInfoView.java  
 \* 创 建 人:  
 \* 文件创建日期: 2018 年 1 月 12 日  
 \* 描 述:  
 \* 文件修改记录：  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
package com.acce.hyide.product.gadget.infoview;  
   
import java.io.File;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.LinkedHashMap;  
import java.util.List;  
import java.util.Map;  
import java.util.Map.Entry;  
   
import org.eclipse.core.resources.IFile;  
import org.eclipse.core.resources.IFolder;  
import org.eclipse.core.resources.IProject;  
import org.eclipse.core.resources.IResource;  
import org.eclipse.core.resources.ResourcesPlugin;  
import org.eclipse.core.runtime.IStatus;  
import org.eclipse.core.runtime.Status;  
import org.eclipse.jface.dialogs.Dialog;  
import org.eclipse.jface.dialogs.IDialogConstants;  
import org.eclipse.jface.dialogs.MessageDialog;  
import org.eclipse.jface.viewers.ISelectionChangedListener;  
import org.eclipse.jface.viewers.SelectionChangedEvent;  
import org.eclipse.jface.viewers.TableViewer;  
import org.eclipse.jface.viewers.Viewer;  
import org.eclipse.jface.viewers.ViewerFilter;  
import org.eclipse.swt.SWT;  
import org.eclipse.swt.custom.CTabFolder;  
import org.eclipse.swt.custom.CTabItem;  
import org.eclipse.swt.custom.SashForm;  
import org.eclipse.swt.events.MenuEvent;  
import org.eclipse.swt.events.MenuListener;  
import org.eclipse.swt.events.SelectionAdapter;  
import org.eclipse.swt.events.SelectionEvent;  
import org.eclipse.swt.events.SelectionListener;  
import org.eclipse.swt.graphics.Color;  
import org.eclipse.swt.layout.GridData;  
import org.eclipse.swt.layout.GridLayout;  
import org.eclipse.swt.widgets.Composite;  
import org.eclipse.swt.widgets.Display;  
import org.eclipse.swt.widgets.Event;  
import org.eclipse.swt.widgets.Listener;  
import org.eclipse.swt.widgets.Menu;  
import org.eclipse.swt.widgets.MenuItem;  
import org.eclipse.swt.widgets.Shell;  
import org.eclipse.swt.widgets.Table;  
import org.eclipse.swt.widgets.TableColumn;  
import org.eclipse.swt.widgets.TableItem;  
import org.eclipse.swt.widgets.Text;  
import org.eclipse.ui.PlatformUI;  
import org.eclipse.ui.dialogs.ElementTreeSelectionDialog;  
import org.eclipse.ui.dialogs.ISelectionStatusValidator;  
import org.eclipse.ui.model.WorkbenchContentProvider;  
import org.eclipse.ui.model.WorkbenchLabelProvider;  
import org.eclipse.ui.part.ViewPart;  
import org.eclipse.ui.views.navigator.ResourceComparator;  
   
import com.acce.hyide.product.util.CommonUtil;  
import com.acce.hyide.product.util.ProjectCfg;  
   
import com.acce.hyide.product.gadget.Activator;  
import com.acce.hyide.product.gadget.Messages;  
import com.acce.hyide.product.gadget.util.AttachDebugAction;  
import com.acce.hyide.product.gadget.util.DeviceUtil;  
   
public class RemoteSystemInfoView extends ViewPart {  
 public CTabFolder tabFolder;  
 public CTabItem tbtmNewItem;  
 public CTabItem tbtmNewItem\_1;  
 public CTabItem tbtmNewItem\_2;  
 public CTabItem tbtmNewItem\_test;  
 public CTabItem tbtmNewItem\_stackusage;  
 public CTabItem tbtmNewItem\_tp;  
 public CTabItem tbtmNewItem\_ints;  
 public Table table;  
 public Table table\_1;  
 public Table table\_2;  
 public Table table\_3;  
 public Table table\_test;  
 public Table table\_stackusage;  
 public Table table\_tp;  
 public Table table\_ints\_nesting;  
 public Table table\_ints\_vector;  
 public Text text;  
 public TableViewer tableViewer;  
 public TableViewer tableViewer\_1;  
 public TableViewer tableViewer\_2;  
 public TableViewer tableViewer\_3;  
 public TableViewer tableViewer\_stackusage;  
 public TableViewer tableViewer\_tp;  
 public TableViewer tableViewer\_ints\_nesting;  
 public TableViewer tableViewer\_ints\_vector;  
 public int TPSelection = -1;  
 RemoteSystemDataInfo rDataInfo;  
 private String strpttid = ""; //$NON-NLS-1$//保存死锁总tid  
 TableItem itemi = null; // 保存临时被替换颜色的TableItem  
 Color col = new Color(Display.getDefault(), 255, 255, 0);  
 Color Red = new Color(Display.getDefault(), 255, 0, 0);  
 Color Lime = new Color(Display.getDefault(), 0, 255, 0);  
 Color coldefault;  
   
 public RemoteSystemInfoView() {  
 rDataInfo = new RemoteSystemDataInfo();  
 }  
   
 @Override  
 public void createPartControl(Composite parent) {  
   
 tabFolder = new CTabFolder(parent, SWT.BORDER);  
 tabFolder.setSimple(false);  
   
 tbtmNewItem = new CTabItem(tabFolder, SWT.NONE);  
 tbtmNewItem.setImage(DeviceUtil.getImage("icons/Memory.png")); //$NON-NLS-1$  
 tbtmNewItem.setText(Messages.RemoteSystemInofView\_Memory);  
 SashForm sashForm = new SashForm(tabFolder, SWT.VERTICAL);  
 tbtmNewItem.setControl(sashForm);  
 tableViewer = new TableViewer(sashForm, SWT.BORDER | SWT.FULL\_SELECTION);  
 table = tableViewer.getTable();  
 table.setHeaderVisible(true);  
 table.setLinesVisible(true);  
 tableViewer\_3 = new TableViewer(sashForm, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_3 = tableViewer\_3.getTable();  
 table\_3.setToolTipText("table\_3");  
 table\_3.setHeaderVisible(true);  
 table\_3.setLinesVisible(true);  
 // 添加一个文本框  
 text = new Text(sashForm, SWT.BORDER | SWT.WRAP | SWT.V\_SCROLL | SWT.READ\_ONLY);  
 text.setFont(DeviceUtil.getFont());  
 text.setBackground(Display.getCurrent().getSystemColor(SWT.COLOR\_WHITE));  
 text.setEditable(true);  
   
 sashForm.setWeights(new int[] { 161, 170, 103 });  
   
 /\* ps \*/  
 Composite compsoite2 = new Composite(tabFolder, SWT.NONE);  
 tbtmNewItem\_1 = new CTabItem(tabFolder, SWT.NONE);  
 tbtmNewItem\_1.setImage(DeviceUtil.getImage("icons/Process.png")); //$NON-NLS-1$  
 tbtmNewItem\_1.setText(Messages.RemoteSystemInofView\_Process);  
 tableViewer\_1 = new TableViewer(compsoite2, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_1 = tableViewer\_1.getTable();  
 GridData gd\_tablea = new GridData(SWT.FILL, SWT.FILL, true, true, 1, 1);  
 gd\_tablea.heightHint = 220;  
 table\_1.setLayoutData(gd\_tablea);  
 table\_1.setHeaderVisible(true);  
 table\_1.setLinesVisible(true);  
 tbtmNewItem\_1.setControl(compsoite2);  
 GridLayout layout2 = new GridLayout();  
 layout2.marginWidth = 0;  
 layout2.marginHeight = 0;  
 compsoite2.setLayout(layout2);  
   
 /\* ts \*/  
 Composite compsoite3 = new Composite(tabFolder, SWT.NONE);  
 tbtmNewItem\_2 = new CTabItem(tabFolder, SWT.NONE);  
 tbtmNewItem\_2.setImage(DeviceUtil.getImage("icons/Thread12.png")); //$NON-NLS-1$  
 tbtmNewItem\_2.setText(Messages.RemoteSystemInofView\_Thread);  
 tableViewer\_2 = new TableViewer(compsoite3, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_2 = tableViewer\_2.getTable();  
 GridData gd\_tablea1 = new GridData(SWT.FILL, SWT.FILL, true, true, 1, 1);  
 gd\_tablea1.heightHint = 220;  
 table\_2.setLayoutData(gd\_tablea1);  
 table\_2.setHeaderVisible(true);  
 table\_2.setLinesVisible(true);  
 tbtmNewItem\_2.setControl(compsoite3);  
 GridLayout layout3 = new GridLayout();  
 layout3.marginWidth = 0;  
 layout3.marginHeight = 0;  
 compsoite3.setLayout(layout3);  
   
 /\* test \*/  
 Composite compsoite\_test = new Composite(tabFolder, SWT.NONE);  
 tbtmNewItem\_2 = new CTabItem(tabFolder, SWT.NONE);  
 tbtmNewItem\_2.setImage(DeviceUtil.getImage("icons/Thread12.png")); //$NON-NLS-1$  
 tbtmNewItem\_2.setText("测试");  
 tableViewer\_2 = new TableViewer(compsoite\_test, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_test = tableViewer\_2.getTable();  
 GridData gd\_tablea\_test = new GridData(SWT.FILL, SWT.FILL, true, true, 1, 1);  
 gd\_tablea\_test.heightHint = 220;  
 table\_test.setLayoutData(gd\_tablea\_test);  
 table\_test.setHeaderVisible(true);  
 table\_test.setLinesVisible(true);  
 tbtmNewItem\_2.setControl(compsoite\_test);  
 GridLayout layout\_test = new GridLayout();  
 layout3.marginWidth = 0;  
 layout3.marginHeight = 0;  
 compsoite\_test.setLayout(layout\_test);  
   
 /\* ss \*/  
 Composite compsoitecupusage = new Composite(tabFolder, SWT.NONE);  
 tbtmNewItem\_stackusage = new CTabItem(tabFolder, SWT.NONE);  
 tbtmNewItem\_stackusage.setImage(DeviceUtil.getImage("icons/stackusage.png")); //$NON-NLS-1$  
 tbtmNewItem\_stackusage.setText(Messages.RemoteSystemInofView\_StackUsage);  
 tableViewer\_stackusage = new TableViewer(compsoitecupusage, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_stackusage = tableViewer\_stackusage.getTable();  
 GridData gd\_tablecupusage = new GridData(SWT.FILL, SWT.FILL, true, true, 1, 1);  
 gd\_tablecupusage.heightHint = 220;  
 table\_stackusage.setLayoutData(gd\_tablecupusage);  
 table\_stackusage.setHeaderVisible(true);  
 table\_stackusage.setLinesVisible(true);  
 tbtmNewItem\_stackusage.setControl(compsoitecupusage);  
 GridLayout layoutcupusage = new GridLayout();  
 layoutcupusage.marginWidth = 0;  
 layoutcupusage.marginHeight = 0;  
 compsoitecupusage.setLayout(layoutcupusage);  
   
 /\* ints \*/  
 tbtmNewItem\_ints = new CTabItem(tabFolder, SWT.NONE);  
 tbtmNewItem\_ints.setImage(DeviceUtil.getImage("icons/stop.png")); //$NON-NLS-1$  
 tbtmNewItem\_ints.setText(Messages.RemoteSystemInofView\_Interrupt);  
 SashForm sashForm\_ints = new SashForm(tabFolder, SWT.VERTICAL);  
 tbtmNewItem\_ints.setControl(sashForm\_ints);  
 tableViewer\_ints\_vector = new TableViewer(sashForm\_ints, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_ints\_vector = tableViewer\_ints\_vector.getTable();  
 table\_ints\_vector.setHeaderVisible(true);  
 table\_ints\_vector.setLinesVisible(true);  
 tableViewer\_ints\_nesting = new TableViewer(sashForm\_ints, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_ints\_nesting = tableViewer\_ints\_nesting.getTable();  
 table\_ints\_nesting.setHeaderVisible(true);  
 table\_ints\_nesting.setLinesVisible(true);  
 sashForm\_ints.setWeights(new int[] { 161, 103 });  
   
 /\* tp \*/  
 Composite compsoitetp = new Composite(tabFolder, SWT.NONE);  
 tbtmNewItem\_tp = new CTabItem(tabFolder, SWT.NONE);  
 tbtmNewItem\_tp.setImage(DeviceUtil.getImage("icons/lock.png")); //$NON-NLS-1$  
 tbtmNewItem\_tp.setText(Messages.RemoteSystemInofView\_ThreadPending);  
 tableViewer\_tp = new TableViewer(compsoitetp, SWT.BORDER | SWT.FULL\_SELECTION);  
 table\_tp = tableViewer\_tp.getTable();  
 GridData gd\_tabletp = new GridData(SWT.FILL, SWT.FILL, true, true, 1, 1);  
 gd\_tabletp.heightHint = 220;  
 table\_tp.setLayoutData(gd\_tabletp);  
 table\_tp.setHeaderVisible(true);  
 table\_tp.setLinesVisible(true);  
 tbtmNewItem\_tp.setControl(compsoitetp);  
 GridLayout layouttp = new GridLayout();  
 layouttp.marginWidth = 0;  
 layouttp.marginHeight = 0;  
 compsoitetp.setLayout(layouttp);  
   
 ThreadPendingMenu();  
 business();  
   
 tabFolder.setSelection(tbtmNewItem);  
 MemoryMenu();  
 ProcessMenu();  
 ThreadMenu();  
 StackUsageMenu();  
 interruptMenu();  
 TestMenu();  
 }  
   
 /\*\*  
 \* 内存选项页菜单  
 \*/  
 private void MemoryMenu() {  
 Menu memorymenu = new Menu(table);  
 MenuItem memoryItem = new MenuItem(memorymenu, SWT.PUSH);  
 memoryItem.setText(Messages.RemoteSystemInofView\_Refresh);  
 memoryItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread memorymenu = new Thread() {  
 public void run() {  
 rDataInfo.RefreshMems();  
 }  
 };  
 memorymenu.setName("memorymenu"); //$NON-NLS-1$  
 memorymenu.start();  
 }  
 });  
   
 MenuItem connectItem = new MenuItem(memorymenu, SWT.PUSH);  
 connectItem.setText(Messages.RemoteSystemInofView\_Reconnect);  
 connectItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
   
 memorymenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 memoryItem.setEnabled(false);  
 connectItem.setEnabled(false);  
 if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 memoryItem.setEnabled(true);  
 } else if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && !RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 connectItem.setEnabled(true);  
 }  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
 }  
 });  
 table.setMenu(memorymenu);  
   
 Menu memoryfreemenu = new Menu(table\_3);  
   
 MenuItem memoryfreeItem = new MenuItem(memoryfreemenu, SWT.PUSH);  
 memoryfreeItem.setText("Refresh.."); //$NON-NLS-1$  
 memoryfreeItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread memoryfreemenu = new Thread() {  
 public void run() {  
 rDataInfo.RefreshMems();  
 }  
 };  
 memoryfreemenu.setName("memoryfreemenu"); //$NON-NLS-1$  
 memoryfreemenu.start();  
 }  
 });  
   
 MenuItem connectfreeItem = new MenuItem(memoryfreemenu, SWT.PUSH);  
 connectfreeItem.setText("Reconnect.."); //$NON-NLS-1$  
 connectfreeItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
   
 memoryfreemenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 memoryfreeItem.setEnabled(false);  
 connectfreeItem.setEnabled(false);  
 if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 memoryfreeItem.setEnabled(true);  
 } else if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && !RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 connectfreeItem.setEnabled(true);  
 }  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
 }  
 });  
 table\_3.setMenu(memoryfreemenu);  
 }  
   
 /\*\*  
 \* 进程选项页菜单  
 \*/  
 private void ProcessMenu() {  
 Menu processmenu = new Menu(table\_1);  
   
 MenuItem KillprocessItem = new MenuItem(processmenu, SWT.PUSH);  
 KillprocessItem.setText(Messages.RemoteSystemInofView\_KillProcess);  
 KillprocessItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 String strprocess = table\_1.getSelection()[0].getText(TableColumns.getIps\_pid());  
 rDataInfo.KillSendcommands(strprocess);  
 }  
 });  
   
 MenuItem refreshprocessItem = new MenuItem(processmenu, SWT.PUSH);  
 refreshprocessItem.setText(Messages.RemoteSystemInofView\_Refresh);  
 refreshprocessItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread processmenu = new Thread() {  
 public void run() {  
 rDataInfo.RefreshProcess();  
 }  
 };  
 processmenu.setName("processmenu"); //$NON-NLS-1$  
 processmenu.start();  
   
 }  
 });  
 /\*----------------------未处理（包含菜单的显示以及判断修改进程核数的pid）-------------------------\*/  
 MenuItem CorrelationprocessItem = new MenuItem(processmenu, SWT.PUSH);  
 CorrelationprocessItem.setText(Messages.RemoteSystemInofView\_Correlation);  
 CorrelationprocessItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Shell shell = PlatformUI.getWorkbench().getActiveWorkbenchWindow().getShell();  
 String strprocessname = table\_1.getSelection()[0].getText(TableColumns  
 .getIps\_name());  
 String strprocess = table\_1.getSelection()[0].getText(TableColumns.getIps\_pid())  
 .trim();  
 String strcpuactive = "NULL";  
 RelevanceDialog dialog = new RelevanceDialog(shell, strprocessname, TableColumns  
 .getIcpucores(), strcpuactive);  
 if (dialog.open() == Dialog.OK) {  
 Thread psaffinitytid = new Thread() {  
 public void run() {  
 rDataInfo.telnetclient  
 .sendCommand("affinity " + strprocess + " " + dialog.getstrCPUActive()); //$NON-NLS-1$ //$NON-NLS-2$  
 }  
 };  
 psaffinitytid.setName("psaffinity"); //$NON-NLS-1$  
 psaffinitytid.start();  
 }  
 }  
 });  
 /\*----------------------未处理-------------------------\*/  
   
 // 创建一个debug按钮，该按钮可以获取当前选中的进程id(pid)，选中需要被调试的app工程后，并由此启动一个debug调试 ，  
 // guoenjing@acoinfo.com  
 MenuItem debugItem = new MenuItem(processmenu, SWT.PUSH);  
 debugItem.setText(Messages.RemoteSystemInofView\_Debug);  
 debugItem.addSelectionListener(new SelectionAdapter() {  
 @Override  
 public void widgetSelected(SelectionEvent event) {  
 if (table\_1.getSelectionCount() != 1) {  
 return;  
 }  
 ElementTreeSelectionDialog dialog = new ElementTreeSelectionDialog(processmenu  
 .getShell(), new WorkbenchLabelProvider(), new WorkbenchContentProvider());  
 dialog.setAllowMultiple(false);  
 dialog.setHelpAvailable(false);  
 dialog.setTitle(Messages.RemoteSystemInofView\_SelectionDialogMsg); //$NON-NLS-1$  
 dialog.setMessage(Messages.RemoteSystemInofView\_SelectionDialogMsg); //$NON-NLS-1$  
 dialog.setInput(ResourcesPlugin.getWorkspace().getRoot());  
 dialog.setComparator(new ResourceComparator(ResourceComparator.NAME));  
 dialog.addFilter(new ViewerFilter() {  
 @Override  
 public boolean select(Viewer viewer, Object parentElement, Object element) {  
 if (element instanceof IResource) {  
 if (element instanceof IProject) {  
 IProject project = (IProject) element;  
 String strProjectPath = project.getLocation().toOSString();  
 if (!"SylixOSAppProject".equals(ProjectCfg.GetProjectCfg(strProjectPath).GetProjectType()) || CommonUtil.IsOlderVersionProject(strProjectPath)) { //$NON-NLS-1$  
 // 排除非app工, 排除老工程  
 return false;  
 }  
 } else if (element instanceof IFolder) {  
 IFolder folder = (IFolder) element;  
 if (!"Debug".equals(folder.getName())) {//$NON-NLS-1$  
 return false;  
 }  
 } else if (element instanceof IFile) {  
 IFile file = (IFile) element;  
 if (!"Debug".equals(file.getParent().getName())) {//$NON-NLS-1$  
 return false;  
 }  
 }  
 return true;  
 } else {  
 return false;  
 }  
 }  
 });  
   
 dialog.setValidator(new ISelectionStatusValidator() {  
 @Override  
 public IStatus validate(Object[] selection) {  
 if (selection != null && selection.length > 0) {  
 if (selection[0] instanceof IProject) {  
 return new Status(IStatus.OK, Activator.PLUGIN\_ID, "");  
 } else if (selection[0] instanceof IFile) {  
 return new Status(IStatus.OK, Activator.PLUGIN\_ID, "");  
 }  
 }  
 return new Status(IStatus.ERROR, Activator.PLUGIN\_ID,  
 Messages.RemoteSystemInofView\_SelectionDialogErrorMsg);  
 }  
 });  
 if (dialog.open() == IDialogConstants.OK\_ID) {  
 Object result = dialog.getFirstResult();  
 if (result == null) { //$NON-NLS-1$  
 return;  
 }  
 IProject project;  
 String programName;  
 String strprocess = table\_1.getSelection()[0].getText(TableColumns.getIps\_pid());  
 if (result instanceof IProject) {  
 project = (IProject) result;  
 programName = "Debug" + File.separator + project.getName();  
 } else if (result instanceof IFile) {  
 IFile executableFile = (IFile) result;  
 project = executableFile.getProject();  
 programName = "Debug" + File.separator + executableFile.getName();  
 } else {  
 return;  
 }  
 new AttachDebugAction().launch(project, programName, strprocess);  
 }  
 }  
 });  
   
 MenuItem ConnectprocessItem = new MenuItem(processmenu, SWT.PUSH);  
 ConnectprocessItem.setText(Messages.RemoteSystemInofView\_Reconnect);  
 ConnectprocessItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
   
 processmenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 isDebugMenuShow(debugItem);  
 isShowMenu(KillprocessItem, refreshprocessItem, ConnectprocessItem,  
 CorrelationprocessItem, table\_1);  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
 }  
 });  
 table\_1.setMenu(processmenu);  
 }  
   
 /\*\*  
 \* 线程选项页菜单  
 \*/  
 private void ThreadMenu() {  
 // 创建一个Thread 右键菜单  
 Menu threadmenu = new Menu(table\_2);  
 MenuItem killthreadItem = new MenuItem(threadmenu, SWT.PUSH);  
 killthreadItem.setText(Messages.RemoteSystemInofView\_KillThread);  
 killthreadItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 String strthreadtid = table\_2.getSelection()[0].getText(TableColumns.getIts\_tid())  
 .trim();  
 rDataInfo.KillSendcommands(strthreadtid);  
 }  
 });  
   
 MenuItem refreshthreadItem = new MenuItem(threadmenu, SWT.PUSH);  
 refreshthreadItem.setText(Messages.RemoteSystemInofView\_Refresh);  
 refreshthreadItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread threadmenu = new Thread() {  
 public void run() {  
 rDataInfo.RefreshThread();  
 }  
 };  
 threadmenu.setName("threadmenu"); //$NON-NLS-1$  
 threadmenu.start();  
 }  
 });  
   
 MenuItem CorrelationthreadItem = new MenuItem(threadmenu, SWT.PUSH);  
 CorrelationthreadItem.setText(Messages.RemoteSystemInofView\_Correlation);  
 CorrelationthreadItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 String strcpuactive = "NULL";  
 String strthreadname = table\_2.getSelection()[0]  
 .getText(TableColumns.getIts\_name()).trim();  
 String strthreadtid = table\_2.getSelection()[0].getText(TableColumns.getIts\_tid())  
 .trim();  
 if (rDataInfo.telnetclient.isTelnetConnected()) {  
 String str = rDataInfo.telnetclient.sendCommand("affinity"); //$NON-NLS-1$  
 rDataInfo.rDataParsing.getaffinity(str, 2);  
 LinkedHashMap<String, String> mapLink = rDataInfo.rDataParsing.getaffinitymap();  
 Iterator<Entry<String, String>> iter = mapLink.entrySet().iterator();  
 while (iter.hasNext()) {  
 Map.Entry entry = (Map.Entry) iter.next();  
 if (entry.getKey().equals(strthreadtid)) {  
 strcpuactive = (String) entry.getValue();  
 }  
 }  
 Shell shell = PlatformUI.getWorkbench().getActiveWorkbenchWindow().getShell();  
 RelevanceDialog dialog = new RelevanceDialog(shell, strthreadname, TableColumns  
 .getIcpucores(), strcpuactive);  
 if (dialog.open() == Dialog.OK) {  
 Thread affinitytid = new Thread() {  
 public void run() {  
 rDataInfo.telnetclient  
 .sendCommand("affinity " + strthreadtid + " " + dialog.getstrCPUActive()); //$NON-NLS-1$ //$NON-NLS-2$  
 }  
 };  
 affinitytid.setName("affinity"); //$NON-NLS-1$  
 affinitytid.start();  
 }  
 }  
 }  
 });  
   
 MenuItem ConnectthreadItem = new MenuItem(threadmenu, SWT.PUSH);  
 ConnectthreadItem.setText(Messages.RemoteSystemInofView\_Reconnect);  
 ConnectthreadItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
 threadmenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 isShowMenu(killthreadItem, refreshthreadItem, ConnectthreadItem,  
 CorrelationthreadItem, table\_2);  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
   
 }  
 });  
 table\_2.setMenu(threadmenu);  
 }  
   
 /\*\*  
 \* Test选项页菜单  
 \*/  
 private void TestMenu() {  
 // 创建一个Thread 右键菜单  
 Menu threadmenu = new Menu(table\_test);  
   
 // 在右键菜单添加操作项  
 MenuItem killthreadItem = new MenuItem(threadmenu, SWT.PUSH);  
 killthreadItem.setText(Messages.RemoteSystemInofView\_KillThread);  
 killthreadItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 String strthreadtid = table\_test.getSelection()[0].getText(TableColumns.getIts\_tid())  
 .trim();  
 rDataInfo.KillSendcommands(strthreadtid);  
 }  
 });  
   
 MenuItem refreshthreadItem = new MenuItem(threadmenu, SWT.PUSH);  
 refreshthreadItem.setText(Messages.RemoteSystemInofView\_Refresh);  
 refreshthreadItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread threadmenu = new Thread() {  
 public void run() {  
 rDataInfo.RefreshThread();  
 }  
 };  
 threadmenu.setName("Testmenu"); //$NON-NLS-1$  
 threadmenu.start();  
 }  
 });  
   
 MenuItem CorrelationthreadItem = new MenuItem(threadmenu, SWT.PUSH);  
 CorrelationthreadItem.setText(Messages.RemoteSystemInofView\_Correlation);  
 CorrelationthreadItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 String strcpuactive = "NULL";  
 String strthreadname = table\_test.getSelection()[0]  
 .getText(TableColumns.getIts\_name()).trim();  
 String strthreadtid = table\_test.getSelection()[0].getText(TableColumns.getIts\_tid())  
 .trim();  
 if (rDataInfo.telnetclient.isTelnetConnected()) {  
 String str = rDataInfo.telnetclient.sendCommand("affinity"); //$NON-NLS-1$  
 rDataInfo.rDataParsing.getaffinity(str, 2);  
 LinkedHashMap<String, String> mapLink = rDataInfo.rDataParsing.getaffinitymap();  
 Iterator<Entry<String, String>> iter = mapLink.entrySet().iterator();  
 while (iter.hasNext()) {  
 Map.Entry entry = (Map.Entry) iter.next();  
 if (entry.getKey().equals(strthreadtid)) {  
 strcpuactive = (String) entry.getValue();  
 }  
 }  
 Shell shell = PlatformUI.getWorkbench().getActiveWorkbenchWindow().getShell();  
 RelevanceDialog dialog = new RelevanceDialog(shell, strthreadname, TableColumns  
 .getIcpucores(), strcpuactive);  
 if (dialog.open() == Dialog.OK) {  
 Thread affinitytid = new Thread() {  
 public void run() {  
 rDataInfo.telnetclient  
 .sendCommand("affinity " + strthreadtid + " " + dialog.getstrCPUActive()); //$NON-NLS-1$ //$NON-NLS-2$  
 }  
 };  
 affinitytid.setName("affinity"); //$NON-NLS-1$  
 affinitytid.start();  
 }  
 }  
 }  
 });  
   
 MenuItem ConnectthreadItem = new MenuItem(threadmenu, SWT.PUSH);  
 ConnectthreadItem.setText(Messages.RemoteSystemInofView\_Reconnect);  
 ConnectthreadItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
   
 // 菜单显示规则  
 threadmenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 isShowMenu(killthreadItem, refreshthreadItem, ConnectthreadItem,  
 CorrelationthreadItem, table\_test);  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
   
 }  
 });  
 // 添加到标签页  
 table\_test.setMenu(threadmenu);  
 }  
   
 /\*\*  
 \* 堆栈选项页菜单  
 \*/  
 private void StackUsageMenu() {  
 Menu stackusagemenu = new Menu(table\_stackusage);  
   
 MenuItem refreshssItem = new MenuItem(stackusagemenu, SWT.PUSH);  
 refreshssItem.setText("Refresh.."); //$NON-NLS-1$  
 refreshssItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread stackusagemenu = new Thread() {  
 public void run() {  
 rDataInfo.RefreshCPUutilization();  
 }  
 };  
 stackusagemenu.setName("stackusagemenu"); //$NON-NLS-1$  
 stackusagemenu.start();  
 }  
 });  
   
 MenuItem connectssItem = new MenuItem(stackusagemenu, SWT.PUSH);  
 connectssItem.setText("Reconnect.."); //$NON-NLS-1$  
 connectssItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
   
 stackusagemenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 refreshssItem.setEnabled(false);  
 connectssItem.setEnabled(false);  
 if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 refreshssItem.setEnabled(true);  
 } else if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && !RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 connectssItem.setEnabled(true);  
 }  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
 }  
 });  
 table\_stackusage.setMenu(stackusagemenu);  
 }  
   
 /\*\*  
 \* 选项页菜单  
 \*/  
 private void interruptMenu() {  
 Menu interruptmenu = new Menu(table\_ints\_nesting);  
   
 MenuItem refreshItem = new MenuItem(interruptmenu, SWT.PUSH);  
 refreshItem.setText("Refresh.."); //$NON-NLS-1$  
 refreshItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread interruptMenu = new Thread() {  
 public void run() {  
 rDataInfo.Refreshints();  
 }  
 };  
 interruptMenu.setName("interruptRefresh"); //$NON-NLS-1$  
 interruptMenu.start();  
 }  
 });  
   
 MenuItem connectItem = new MenuItem(interruptmenu, SWT.PUSH);  
 connectItem.setText("Reconnect.."); //$NON-NLS-1$  
 connectItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
   
 interruptmenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 refreshItem.setEnabled(false);  
 connectItem.setEnabled(false);  
 if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 refreshItem.setEnabled(true);  
 } else if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && !RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 connectItem.setEnabled(true);  
 }  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
 }  
 });  
 table\_ints\_nesting.setMenu(interruptmenu);  
   
 Menu ints\_vectortmenu = new Menu(table\_ints\_vector);  
   
 MenuItem refreshintsItem = new MenuItem(ints\_vectortmenu, SWT.PUSH);  
 refreshintsItem.setText("Refresh.."); //$NON-NLS-1$  
 refreshintsItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread interruptMenu = new Thread() {  
 public void run() {  
 rDataInfo.Refreshints();  
 }  
 };  
 interruptMenu.setName("interruptRefresh"); //$NON-NLS-1$  
 interruptMenu.start();  
 }  
 });  
   
 MenuItem connectintsItem = new MenuItem(ints\_vectortmenu, SWT.PUSH);  
 connectintsItem.setText("Reconnect.."); //$NON-NLS-1$  
 connectintsItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
   
 ints\_vectortmenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 refreshintsItem.setEnabled(false);  
 connectintsItem.setEnabled(false);  
 if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 refreshintsItem.setEnabled(true);  
 } else if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && !RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 connectintsItem.setEnabled(true);  
 }  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
 }  
 });  
 table\_ints\_vector.setMenu(ints\_vectortmenu);  
 }  
   
 /\*\*  
 \* 线程阻塞选项页菜单  
 \*/  
 private void ThreadPendingMenu() {  
 Menu tpmenu = new Menu(table\_tp);  
   
 MenuItem refreshtpItem = new MenuItem(tpmenu, SWT.PUSH);  
 refreshtpItem.setText("Refresh.."); //$NON-NLS-1$  
 refreshtpItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Thread tpmenu = new Thread() {  
 public void run() {  
 rDataInfo.Refreshtp();  
 }  
 };  
 tpmenu.setName("tpmenu"); //$NON-NLS-1$  
 tpmenu.start();  
   
 }  
 });  
   
 MenuItem connecttpItem = new MenuItem(tpmenu, SWT.PUSH);  
 connecttpItem.setText("Reconnect.."); //$NON-NLS-1$  
 connecttpItem.addListener(SWT.Selection, new Listener() {  
 @Override  
 public void handleEvent(Event event) {  
 Reconnect();  
 }  
 });  
   
 tpmenu.addMenuListener(new MenuListener() {  
 @Override  
 public void menuShown(MenuEvent e) {  
 refreshtpItem.setEnabled(false);  
 connecttpItem.setEnabled(false);  
 if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 refreshtpItem.setEnabled(true);  
 } else if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient != null  
 && !RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 connecttpItem.setEnabled(true);  
 }  
 }  
   
 @Override  
 public void menuHidden(MenuEvent e) {  
 }  
 });  
 table\_tp.setMenu(tpmenu);  
 }  
   
 /\*\*  
 \* 添加信息公共方法  
 \*  
 \* @param table  
 \* @param tableViewer  
 \*/  
 void setcreatePart(Table table, TableViewer tableViewer, String[][] strcontent,  
 String[] strtitle) {  
 Display.getDefault().syncExec(new Runnable() {  
 @Override  
 public void run() {  
 // 记录table当前位置  
 int iscrollbarlocation = table.getTopIndex();  
 int iSelection = table.getSelectionIndex();  
 // 清空table内容  
 table.removeAll();  
   
 // 重新加载table内容  
 // if(tableViewer.getTable().getColumnCount()==0){  
 TableColumn[] columns = table.getColumns(); // 清除表格原有表头  
 for (TableColumn tc : columns) {  
 tc.dispose();  
 }  
 // 添加表头  
 for (int i = 0; i < strtitle.length; i++) {  
 TableColumn TableContent = new TableColumn(table, SWT.NONE, i);  
 TableContent.setWidth(140);  
 TableContent.setText(strtitle[i]);  
 }  
 // }  
   
 for (int j = 0; j < strcontent.length; j++) {  
 // 添加数据  
 TableItem TableTitle = new TableItem(table, SWT.NONE);  
 String strcont[] = new String[strcontent[j].length];  
 for (int k = 0; k < strcontent[j].length; k++) {  
 strcont[k] = strcontent[j][k];  
 }  
 TableTitle.setText(strcont);  
 }  
 if (table.equals(table\_tp)) {  
 // 获取table所有行  
 List<TableItem> list = new ArrayList<TableItem>();  
 TableItem[] items = table\_tp.getItems();  
 for (int i = 0; i < items.length; i++) {  
 if (!items[i].getText(6).trim().equals("")) { //$NON-NLS-1$  
 items[i].setBackground(col);  
 list.add(items[i]);  
 }  
 }  
 List<TableItem> listred = new ArrayList<TableItem>();  
 for (int x = 0; x < list.size(); x++) {  
 for (int y = 0; y < list.size(); y++) {  
 if ((list.get(y).getText(6).trim()  
 .equals(list.get(x).getText(1).trim()))  
 && (list.get(y).getText(1).trim().equals(list.get(x).getText(6)  
 .trim()))) {  
 listred.add(list.get(y));  
 }  
 if (list.get(x).getText(6).trim().equals(list.get(y).getText(1).trim())) {  
 listred.add(list.get(y));  
 }  
 }  
 }  
 for (int x = 0; x < listred.size(); x++) {  
 listred.get(x).setBackground(Red);  
 }  
 itemi = null;  
 }  
 if (!table.equals(table\_tp) && iSelection != -1) {  
 table.setSelection(iSelection);  
 }  
   
 if (TPSelection != -1) {  
 TableItem[] items = table\_tp.getItems();  
 for (int i = 0; i < items.length; i++) {  
 if (items[i].getText(1).equals(strpttid)) {  
 TPSelection = i;  
 }  
 }  
 if (TPSelection >= items.length) {  
 return;  
 }  
 table\_tp.setSelection(TPSelection);  
 TableItem item = table\_tp.getItem(TPSelection);  
 String strowner = item.getText(6);  
 if (!strowner.isEmpty()) {  
 for (int i = 0; i < items.length; i++) {  
 if (items[i].getText(1).equals(strowner)) {  
 itemi = items[i];  
 itemi.setBackground(Lime);  
 }  
 }  
 }  
 }  
 // 设置当前选中信息位置  
 table.setTopIndex(iscrollbarlocation);  
 }  
 });  
 }  
   
 /\*\*  
 \* 商业版中tptableViewer 选中的行  
 \*/  
 public void business() {  
 tableViewer\_tp.addSelectionChangedListener(new ISelectionChangedListener() {  
 public void selectionChanged(SelectionChangedEvent event) {  
 if (itemi != null) {  
 if (coldefault == null) {  
 coldefault = table\_tp.getBackground();  
 }  
 itemi.setBackground(coldefault);  
 }  
 TPSelection = table\_tp.getSelectionIndex();  
 if (TPSelection != -1) {  
 TableItem item = table\_tp.getItem(TPSelection);  
 strpttid = item.getText(1);  
 String strowner = item.getText(6);  
 if (!strowner.isEmpty()) {  
 TableItem[] items = table\_tp.getItems();  
 for (int i = 0; i < items.length; i++) {  
 if (items[i].getText(1).equals(strowner)) {  
 coldefault = items[i].getBackground();  
 itemi = items[i];  
 itemi.setBackground(Lime);  
 }  
 }  
 }  
 }  
 }  
 });  
 }  
   
 public void setmenry(String[] strtitle, String[][] strcontent) {  
 setcreatePart(table, tableViewer, strcontent, strtitle);  
 }  
   
 public void setProcess(String[] strtitle, String[][] strcontent) {  
   
 setcreatePart(table\_1, tableViewer\_1, strcontent, strtitle);  
 }  
   
 public void setThread(String[] strtitle, String[][] strcontent) {  
 setcreatePart(table\_2, tableViewer\_2, strcontent, strtitle);  
 }  
   
 public void setCPUutilization(String[] strtitle, String[][] strcontent) {  
 setcreatePart(table\_stackusage, tableViewer\_stackusage, strcontent, strtitle);  
 }  
   
 public void settp(String[] strtitle, String[][] strcontent) {  
 setcreatePart(table\_tp, tableViewer\_tp, strcontent, strtitle);  
 }  
   
 public void setfree(String[] strtitle, String[][] strcontent, String strtextcontent) {  
 setcreatePart(table\_3, tableViewer\_3, strcontent, strtitle);  
 text.getDisplay().syncExec(new Runnable() {  
 @Override  
 public void run() {  
 text.setText(""); //$NON-NLS-1$  
 text.setText(strtextcontent);  
 }  
 });  
 }  
   
 public void dispose() {  
 RemoteSystemDataInfo.DisConnect();  
 RemoteSystemDataInfo.rDeviceEntrity = null;  
 RemoteSystemDataInfo.telnetclient = null;  
 DeviceUtil.RefreshView();  
 }  
   
 private void isShowMenu(MenuItem KillItem, MenuItem RefreshItem, MenuItem ConnectItem,  
 MenuItem CorrelationItem, Table table) {  
 KillItem.setEnabled(false);  
 RefreshItem.setEnabled(false);  
 ConnectItem.setEnabled(false);  
 CorrelationItem.setEnabled(false);  
 if (RemoteSystemDataInfo.rDeviceEntrity != null) {  
 if (RemoteSystemDataInfo.telnetclient.isTelnetConnected()) {  
 RefreshItem.setEnabled(true);  
   
 if (table.getSelectionIndex() != -1) {  
 TableItem items = table.getItem(table.getSelectionIndex());  
 if (items == null) {  
 return;  
 }  
 if (table.equals(table\_1)  
 && !items.getText(TableColumns.getIps\_pid()).trim().equals("0")) { //$NON-NLS-1$  
 KillItem.setEnabled(true);  
 CorrelationItem.setEnabled(true);  
 }  
 if (table.equals(table\_2)  
 && !items.getText(TableColumns.getIts\_pid()).trim().equals("0")) { //$NON-NLS-1$  
 KillItem.setEnabled(true);  
 }  
 if (TableColumns.getIcpucores() == 1) {  
 CorrelationItem.setEnabled(false);  
 } else if (table.equals(table\_2)  
 && !items.getText(TableColumns.getIts\_name()).trim().startsWith("t\_idle")) { //$NON-NLS-1$  
 CorrelationItem.setEnabled(true);  
 }  
 }  
   
 } else {  
 ConnectItem.setEnabled(true);  
 }  
 }  
 }  
   
 /\*\*  
 \* 判断debug按钮是否显示  
 \*  
 \* @author guoenjing@acoinfo.com  
 \* @param debugItem  
 \*/  
 private void isDebugMenuShow(MenuItem debugItem) {  
 debugItem.setEnabled(false);  
 if (RemoteSystemDataInfo.rDeviceEntrity != null  
 && RemoteSystemDataInfo.telnetclient.isTelnetConnected()  
 && table\_1.getSelectionIndex() != -1) {  
 TableItem items = table\_1.getItem(table\_1.getSelectionIndex());  
 if (items == null) {  
 return;  
 }  
 if (!items.getText(TableColumns.getIps\_pid()).trim().equals("0")) { //$NON-NLS-1$  
 debugItem.setEnabled(true);  
 }  
 }  
 }  
   
 private void Reconnect() {  
 rDataInfo.ismessagebox = true;  
 if (rDataInfo.Reloginthread != null && rDataInfo.Reloginthread.isAlive()) {  
 return;  
 }  
 rDataInfo.Relogin();  
 }  
   
 @Override  
 public void setFocus() {  
 // TODO Auto-generated method stub  
 }  
}

* 交互方式；
* plugin.xml标签

<extension  
 point="org.eclipse.ui.menus">  
 <menuContribution  
 allPopups="false"  
 locationURI="menu:org.eclipse.ui.main.menu?after=additions">  
 <menu  
 label="菜单栏">  
 <command  
 commandId="test\_01.command1"  
 label="菜单栏1.1"  
 style="push"  
 tooltip="测试">  
 </command>  
 <command  
 commandId="test\_01.command3"  
 label="菜单栏1.2"  
 style="push"  
 tooltip="保存全部c">  
 </command>  
 <menu  
 commandId="test\_01.command2"  
 label="菜单栏1.3">  
 </menu>  
 </menu>  
 </menuContribution>  
 </extension>

* 关键参数说明 point 表示menuContribution 标签会将插件添加到头菜单或工具栏位置。 locationURI 是菜单位置 menu 表示是菜单,menu可以嵌套多个menu实现多级菜单 command 表示一个链接指令，这里的id是非必须得 commandId 是链接要的执行的插件ID,此插件将实现具体功能。 由于menu也是通过commandId访问类的,所以链接的类都要遵循handler类规则,继承AbstractHandler. #### 1.2 command的声明 {#command的声明 }
* plugin.xml标签

<extension  
 point="org.eclipse.ui.commands">  
 <category  
 name="样本类别"  
 id="Test\_01.commands.category">  
 </category>  
 <command  
 name="样本命令"  
 categoryId="Test\_01.commands.category"  
 defaultHandler="test.handlers.TestHandler "  
 id="Test\_01.commands.sampleCommand">  
 </command>  
 </extension>

* 关键参数说明 category 提供一个分类，其他组件通过categoryId来导入进分类菜单 command 是一个链接指令，没有实方法，而是要通过hander标签来实现。这里的id是必须的，所有链接都通过commandId来寻找对应点。 defaultHandler默认指向的实现类.可以不填,并通过handler从新指向, #### 1.3 command的实现-handlers {#command的实现-handlers }
* xml标签

<extension  
 point="org.eclipse.ui.handlers">  
 <handler  
 commandId="Test\_01.commands.sampleCommand"  
 class="test.handlers.TestHandler">  
 </handler>  
</extension>

* 关键参数说明 通过绑定command后，指定class。来将command和class绑定，实现指定功能，方便拓展。 TestHandler 要继承AbstractHandler, public Object execute(ExecutionEvent event) throws ExecutionException {}是系统调入接口.
* 样例代码

import org.eclipse.core.commands.AbstractHandler;  
import org.eclipse.core.commands.ExecutionEvent;  
import org.eclipse.core.commands.ExecutionException;  
import org.eclipse.core.runtime.Platform;  
import org.eclipse.jface.action.IAction;  
import org.eclipse.jface.dialogs.MessageDialog;  
import org.eclipse.swt.widgets.Shell;  
import org.eclipse.ui.IObjectActionDelegate;  
import org.eclipse.ui.IWorkbenchPart;  
   
public class GetWorkSpacePath extends AbstractHandler  
{  
   
 private Shell shell;  
   
 // 这里可以进行一些初始化操作,在super();后  
 public GetWorkSpacePath() {  
 super();  
 }  
   
 /\*\*  
 \* @see IObjectActionDelegate#setActivePart(IAction, IWorkbenchPart)  
 \*/  
 public void setActivePart(IAction action, IWorkbenchPart targetPart) {  
 shell = targetPart.getSite().getShell();  
 }  
   
 // 通过右键任意菜单,调用的类,都会自动调用此方法  
 @Override  
 public Object execute(ExecutionEvent event) throws ExecutionException {  
 MessageDialog.openInformation(  
 shell,  
 "当前WS路径",  
 Platform.getLocation().toString());  
 return null;  
 }  
   
}

#### 1.4 toobar

* 交互方式
* xml标签

<extension  
 <menuContribution  
 locationURI="toolbar:org.eclipse.ui.main.toolbar?after=additions">  
 <toolbar  
 id="Test\_01.toolbars.sampleToolbar">  
 <command  
 commandId="Test\_01.commands.sampleCommand"  
 icon="icons/sample.gif"  
 tooltip="向大家问好"  
 id="Test\_01.toolbars.sampleCommand">  
 </command>  
 </toolbar>  
 </menuContribution>  
</extension>

* 关键参数说明 toolbar 表示一个工具栏位置 tooltip 是鼠标停留时的文字提示 和menu一样,都是通过commond链接的,这里就不重复了.

#### 1.5快捷键

* xml标签

<extension  
 point="org.eclipse.ui.bindings">  
 <key  
 commandId="Test\_01.commands.sampleCommand"  
 contextId="org.eclipse.ui.contexts.window"  
 sequence="M1+6"  
 schemeId="org.eclipse.ui.defaultAcceleratorConfiguration">  
 </key>  
 </extension>

#### 1.6 右键菜单

##### 1.6.1 右键任意菜单 PopupMenu

<extension  
 <menuContribution  
 allPopups="false"  
 locationURI="popup:org.eclipse.ui.popup.any?after=additions">  
 <menu  
 label="右键菜单"  
 mnemonic="M"  
 id="Test\_01.menus.samplePopupMenu">  
 <command  
 commandId="Test\_01.commands.command2"  
 label="工作空间路径"  
 mnemonic="S"  
 id="Test\_01.popup.sampleCommand">  
 </command>  
 </menu>  
 </menuContribution>  
</extension>

locationURI: 菜单位置,表示任意右键菜单 command 链接的handler的类需要 继承 AbstractHandler, public Object execute(ExecutionEvent event) {}是系统调入接口. \* 样例源码

test\_01.popup.actions.GetWorkSpacePath.java  
import org.eclipse.core.commands.AbstractHandler;  
import org.eclipse.core.commands.ExecutionEvent;  
import org.eclipse.core.commands.ExecutionException;  
import org.eclipse.core.runtime.Platform;  
import org.eclipse.jface.action.IAction;  
import org.eclipse.jface.dialogs.MessageDialog;  
import org.eclipse.swt.widgets.Shell;  
import org.eclipse.ui.IObjectActionDelegate;  
import org.eclipse.ui.IWorkbenchPart;  
   
public class GetWorkSpacePath extends AbstractHandler  
{  
   
 private Shell shell;  
   
 // 这里可以进行一些初始化操作,在super();后  
 public GetWorkSpacePath() {  
 super();  
 }  
   
 /\*\*  
 \* @see IObjectActionDelegate#setActivePart(IAction, IWorkbenchPart)  
 \*/  
 public void setActivePart(IAction action, IWorkbenchPart targetPart) {  
 shell = targetPart.getSite().getShell();  
 }  
   
 // 通过右键任意菜单,调用的类,都会自动调用此方法  
 @Override  
 public Object execute(ExecutionEvent event) throws ExecutionException {  
 MessageDialog.openInformation(  
 shell,  
 "当前WS路径",  
 Platform.getLocation().toString());  
 return null;  
 }  
   
}

command -Test\_01.commands.command2需要一个handler,指向test\_01.popup.actions.GetWorkSpacePath.java

##### 1.6.2右键资源菜单

* xml

<extension  
 point="org.eclipse.ui.popupMenus">  
 <objectContribution  
 id="Test\_01.contribution1"  
 objectClass="org.eclipse.core.resources.IFile">  
 <menu  
 id="Test\_01.menu1"  
 label="新的子菜单"  
 path="additions">  
 <separator  
 name="group1">  
 </separator>  
 </menu>  
 <action  
 class="test\_01.popup.actions.newClass"  
 enablesFor="1"  
 id="Test\_01.newAction"  
 label="新的操作"  
 menubarPath="Test\_01.menu1/group1">  
 </action>  
 </objectContribution>  
 </extension>

如果objectClass的值为org.eclipse.core.internal.resources.Project,则此菜单针对项目显示 实现类test\_01.popup.actions.newClass要 implements IobjectActionDelegate，并实现run (IAction action) menubarPath 是菜单位置 \* 样例源码 test\_01.popup.actions.新的操作.java

import org.eclipse.core.internal.runtime.Activator;  
import org.eclipse.jface.action.IAction;  
import org.eclipse.jface.dialogs.MessageDialog;  
import org.eclipse.jface.viewers.ISelection;  
import org.eclipse.swt.widgets.Shell;  
import org.eclipse.ui.IObjectActionDelegate;  
import org.eclipse.ui.IWorkbenchPart;  
   
@SuppressWarnings("restriction")  
public class 新的操作 implements IObjectActionDelegate  
{  
 private Shell shell;  
 public 新的操作() {  
 super();  
 }  
 public void setActivePart(IAction action, IWorkbenchPart targetPart) {  
 shell = targetPart.getSite().getShell();  
 }  
 // 通过资源右键菜单实例化后,会自动调用此方法  
 public void run(IAction action) {  
 String path = Activator.getDefault().getInstallLocation().toString();//((IResource) getElement()).getFullPath().toString();  
 MessageDialog.openInformation(  
 shell,  
 "新的操作",  
 path);  
 }  
 public void selectionChanged(IAction action, ISelection selection) {  
 }  
}

test\_01.popup.actions.新的操作.java竟然可以用中文命名类!!! 同一个类,可以被右键任意菜单和资源菜单分别调用,只是其(继承)调用方法不同,一个是run,一个是execute.

##### 1.6.3 右键资源属性菜单

* 交互方式
* xml

<extension  
 point="org.eclipse.ui.propertyPages">  
 <page  
 class="test\_01.properties.SamplePropertyPage"  
 id="test\_01.properties.samplePropertyPage"  
 name="样本页面"  
 nameFilter="\*.\*" >  
 <enabledWhen>  
 <instanceof  
 value="org.eclipse.core.resources.IFile">  
 </instanceof>  
 </enabledWhen>  
 </page>  
</extension>

* 说明 test\_01.properties.SamplePropertyPage 要继承PropertyPage nameFilter 资源过滤器 instanceof value 表示资源对象是文件类型,也就是说,只有右键某一具体文件如text文件时,才会加载其属性页.当value = org.eclipse.core.internal.resources.Project时,此项对项目响应.
* 样例源码 test\_01.properties.SamplePropertyPage.java

import org.eclipse.core.resources.IResource;  
import org.eclipse.core.runtime.CoreException;  
import org.eclipse.core.runtime.Platform;  
import org.eclipse.core.runtime.QualifiedName;  
import org.eclipse.swt.SWT;  
import org.eclipse.swt.layout.GridData;  
import org.eclipse.swt.layout.GridLayout;  
import org.eclipse.swt.widgets.Composite;  
import org.eclipse.swt.widgets.Control;  
import org.eclipse.swt.widgets.Label;  
import org.eclipse.swt.widgets.Text;  
import org.eclipse.ui.dialogs.PropertyPage;  
   
public class SamplePropertyPage extends PropertyPage {  
   
 private static final String PATH\_TITLE = "Path:";  
 private static final String OWNER\_TITLE = "&Owner:";  
 private static final String OWNER\_PROPERTY = "OWNER";  
 private static final String DEFAULT\_OWNER = "John Doe";  
   
 private static final int TEXT\_FIELD\_WIDTH = 50;  
   
 private Text ownerText;  
   
 /\*\*  
 \* Constructor for SamplePropertyPage.  
 \*/  
 public SamplePropertyPage() {  
 super();  
 }  
   
 private void addFirstSection(Composite parent) {  
 Composite composite = createDefaultComposite(parent);  
   
 // 路径标签  
 Label pathLabel = new Label(composite, SWT.NONE);  
 pathLabel.setText(PATH\_TITLE);  
   
 // 路径值  
 Text pathValueText = new Text(composite, SWT.WRAP | SWT.READ\_ONLY);  
 // 文件路径 工作空间路径 + 文件相对路径  
 String path = Platform.getLocation().toString()  
 + ((IResource) getElement()).getFullPath().toString();  
 pathValueText.setText(path);  
   
 }  
   
 // 添加分割线  
 private void addSeparator(Composite parent) {  
 Label separator = new Label(parent, SWT.SEPARATOR | SWT.HORIZONTAL);  
 GridData gridData = new GridData();  
 gridData.horizontalAlignment = GridData.FILL;  
 gridData.grabExcessHorizontalSpace = true;  
 separator.setLayoutData(gridData);  
 }  
   
 // 自定义UI和数据可视化  
 private void addSecondSection(Composite parent) {  
 Composite composite = createDefaultComposite(parent);  
   
 // 标签1  
 Label ownerLabel = new Label(composite, SWT.NONE);  
 ownerLabel.setText(OWNER\_TITLE);  
   
 // 输入框  
 ownerText = new Text(composite, SWT.SINGLE | SWT.BORDER);  
 GridData gd = new GridData();  
 gd.widthHint = convertWidthInCharsToPixels(TEXT\_FIELD\_WIDTH);  
 ownerText.setLayoutData(gd);  
   
 // 获取数据  
 try {  
 // 获取 字段为OWNER\_PROPERTY的数据  
 String owner = ((IResource) getElement())  
 .getPersistentProperty(new QualifiedName("", OWNER\_PROPERTY));  
 // 填充数据到输入框  
 ownerText.setText((owner != null) ? owner : DEFAULT\_OWNER);  
 } catch (CoreException e) {  
 ownerText.setText(DEFAULT\_OWNER);  
 }  
 }  
   
 /\*\*  
 \* @see 属性偏向页面，创建内容(Composite)  
 \*/  
 protected Control createContents(Composite parent) {  
 Composite composite = new Composite(parent, SWT.NONE); // 获取组合模式  
 GridLayout layout = new GridLayout(); // 网格布局  
 composite.setLayout(layout);  
 GridData data = new GridData(GridData.FILL);  
 data.grabExcessHorizontalSpace = true;  
 composite.setLayoutData(data);  
   
 addFirstSection(composite);  
 addSeparator(composite);  
 addSecondSection(composite);  
 return composite;  
 }  
   
 private Composite createDefaultComposite(Composite parent) {  
 Composite composite = new Composite(parent, SWT.NULL);  
 GridLayout layout = new GridLayout();  
 layout.numColumns = 2;  
 composite.setLayout(layout);  
   
 GridData data = new GridData();  
 data.verticalAlignment = GridData.FILL;  
 data.horizontalAlignment = GridData.FILL;  
 composite.setLayoutData(data);  
   
 return composite;  
 }  
   
 // 恢复默认值响应  
 protected void performDefaults() {  
 super.performDefaults();  
 // Populate the owner text field with the default value  
 ownerText.setText(DEFAULT\_OWNER);  
 }  
   
 // 用户确认响应  
 public boolean performOk() {  
 // store the value in the owner text field  
 try {  
 // 保存数据，类似keyValue的操作  
 ((IResource) getElement()).setPersistentProperty(new QualifiedName(  
 "", OWNER\_PROPERTY), ownerText.getText());  
 } catch (CoreException e) {  
 return false;  
 }  
 return true;  
 }  
   
}

* PropertyPage 在PropertyPage中,可通过getElement()).getFullPath().toString()获得资源文件路径

#### 1.7 command的链接作用

1、 command声明一个id后，handlers通过commandId实现将class和command绑定； 2、 menu、toolbar、快捷键等用户操作即可、通过指定commandId链接到对应的实现class。 3、 通过command，可为同一个功能（插件）提供多种用户入口，提高软件的交互友好性。

#### 1.8 PreferencePage

* 交互方式
* xml

<extension  
 point="org.eclipse.ui.preferencePages">  
 <page  
 category="test\_02"  
 class="test\_01.preferences.SamplePreferencePage"  
 id="test\_01.preferences.SamplePreferencePage"  
 name="样本首选项">  
 <keywordReference  
 id="test\_01.keywordReference1">  
 </keywordReference>  
 </page>  
 </extension>  
 <extension  
 point="org.eclipse.core.runtime.preferences">  
 <initializer  
 class="test\_01.preferences.PreferenceInitializer">  
 </initializer>  
 </extension>

* 说明 preferencePages是首选项页面 preferences是对应首选项的默认数据

test\_01.preferences.SamplePreferencePage要继承 FieldEditorPreferencePage并implements IworkbenchPreferencePage.这是个很有意思很强大的类,可以自动UI布局和数据的适配,读取都简化了. test\_01.preferences.PreferenceInitializer是,首选项点击恢复默认时的调用.要继承AbstractPreferenceInitializer \* 样例源码 a.test\_01.preferences.SamplePreferencePage.java

import org.eclipse.jface.preference.\*;  
import org.eclipse.ui.IWorkbenchPreferencePage;  
import org.eclipse.ui.IWorkbench;  
import test\_01.Activator;  
   
public class SamplePreferencePage  
 extends FieldEditorPreferencePage  
 implements IWorkbenchPreferencePage {  
   
 public SamplePreferencePage() {  
 super(GRID);  
 setPreferenceStore(Activator.getDefault().getPreferenceStore());  
 setDescription("这里添加自定义功能描述或提示文本");  
 }  
   
 public void createFieldEditors() {  
 // 添加一个文件路径选择  
 addField(new DirectoryFieldEditor(PreferenceConstants.P\_PATH,   
 "&这里添加文件路径选择描述", getFieldEditorParent()));  
 // 添加一个单选框  
 addField(  
 new BooleanFieldEditor(  
 PreferenceConstants.P\_BOOLEAN,  
 "&这里添加单选框描述",  
 getFieldEditorParent()));  
 // 复单选框  
 addField(new RadioGroupFieldEditor(  
 PreferenceConstants.P\_CHOICE,  
 "这里添加复单选框描述",  
 1,  
 new String[][] { { "&Choice 1", "choice1" }, {  
 "C&hoice 2", "choice2" }  
 }, getFieldEditorParent()));  
   
 // 输入框  
 addField(  
 new StringFieldEditor(PreferenceConstants.P\_STRING, "这里添加输入框描述", getFieldEditorParent()));  
 }  
 // 这里是自动调用的初始化方法  
 public void init(IWorkbench workbench) {  
 }  
   
}

* 1. PreferenceInitializer.java

import org.eclipse.core.runtime.preferences.AbstractPreferenceInitializer;  
import org.eclipse.jface.preference.IPreferenceStore;  
   
import test\_01.Activator;  
   
public class PreferenceInitializer extends AbstractPreferenceInitializer {  
   
 public void initializeDefaultPreferences() {  
 // 获取默认配置  
 IPreferenceStore store = Activator.getDefault().getPreferenceStore();  
 // 通过字段，设置响应的值  
 store.setDefault(PreferenceConstants.P\_BOOLEAN, true);  
 store.setDefault(PreferenceConstants.P\_PATH, "DefaultPath");//路径可以省略不写  
 store.setDefault(PreferenceConstants.P\_CHOICE, "choice1");  
 store.setDefault(PreferenceConstants.P\_STRING,  
 "Default value");  
 }  
   
}

* 说明: PreferenceConstants.P\_BOOLEAN此类都是字段,通过字段,配置相应的UI和数据操作.

### 2 视图(view)

### 2.1 一级节点 org.eclipse.ui.views

#### 2.2 分类文件夹 category

* xml

<category  
 id="xxx.xxx.category.xxx"  
 name="视图名称">  
 </category>

* 说明 此节点在：窗口->显示视图->“视图名称” 进行可视操作.
* 具体UI将在 此节点下的子标签： view 下设置 #### 2.3 view {#view }
* 右键新建 view 标签，填写各项参数
* class

### 3 文件操作

#### 2.1 newWizards

* 交互方式
* xml

<extension  
 point="org.eclipse.ui.newWizards">  
 <category  
 id="test\_01"  
 name="新建">  
 </category>  
 <wizard  
 category="test\_01"  
 class="test\_01.wizards.SampleNewWizard"  
 icon="icons/sample.gif"  
 id="test\_01.wizards.SampleNewWizard"  
 name="新建文件">  
 </wizard>  
 </extension>

* 说明 category 是新建类型的分类文件夹,如果不设,则下面新建则自动归档在”其他”文件夹下  
  test\_01.wizards.SampleNewWizard 要继承Wizard,并implements InewWizard

#### 2.2 importWizards

* 交互方式
* xml

<extension  
 point="org.eclipse.ui.importWizards">  
 <category  
 id="test\_01.importWizards.sampleCategory"  
 name="导入样本文件">  
 </category>  
 <wizard  
 category="test\_01.importWizards.sampleCategory"  
 class="test\_01.importWizards.ImportWizard"  
 icon="icons/sample.gif"  
 id="test\_01.importWizards.ImportWizard"  
 name="导入文件">  
 <description>  
 将本地文件系统中的文件导入到工作空间中。  
 </description>  
 </wizard>  
 </extension>

* 说明 category 导入分类文件夹 wizard 文件导入操作 description 导入说明 test\_01.importWizards.ImportWizard 要继承Wizard,并IimportWizard

#### 2.3 renameParticipant 重命名

* xml

<extension  
 point="org.eclipse.ltk.core.refactoring.renameParticipants">  
 <renameParticipant  
 class="com.acce.hyide.project.core.participant.ProjectRenameParticipant"  
 id="com.acce.hyide.project.core.projectrenameParticipant"  
 name="ProjectRenameParticipant">  
 <enablement>  
 <with  
 variable="element">  
 <instanceof  
 value="org.eclipse.core.resources.IProject">  
 </instanceof>  
 </with>  
 </enablement>  
 </renameParticipant>  
</extension>

* 说明 enablement 的 with标签,是对资源类型的过滤, enablement->with-> instanceof-.> value 表示对项目起作用,

<adapt  
 type="org.eclipse.core.resources.IFile">  
 </adapt>为对文件响应

com.acce.hyide.project.core.participant.ProjectRenameParticipant要继承RenameParticipant

#### 2.4 deleteParticipant 删除

* xml

<extension  
 point="org.eclipse.ltk.core.refactoring.deleteParticipants">  
 <deleteParticipant  
 class="com.acce.hyide.project.core.participant.mkFileDeleteParticipant"  
 id="com.acce.hyide.project.core.mkfiledeleteParticipant"  
 name="mkFileDeleteParticipant">  
 <enablement>  
 <with  
 variable="element">  
 <adapt  
 type="org.eclipse.core.resources.IFile">  
 </adapt>  
 </with>  
 </enablement>  
 </deleteParticipant>  
 </extension>

* 说明 com.acce.hyide.project.core.participant.mkFileDeleteParticipant要继承DeleteParticipant,其他参数和renameParticipant意义一样

#### 2.5 navigatorContent

* xml

<extension  
 point="org.eclipse.ui.navigator.navigatorContent">  
 <commonWizard  
 associatedExtensionId="org.eclipse.cdt.ui.navigator.content"  
 menuGroupId="org.eclipse.cdt.ui.newProject"  
 type="new"  
 wizardId="com.acce.hyide.project.arm.wizard\_cproject\_base">  
 <enablement>  
 <or>  
 <with  
 variable="activeWorkbenchWindow.activePerspective">  
 <equals  
 value="com.acce.hyide.project.arm.perspective">  
 </equals>  
 </with>  
 </or></enablement>  
 </commonWizard>  
<extension>

* 说明 内容扩展提供可由导航器内容服务使用的内容和标签提供程序。 navigatorContent 扩展定义除了扩展所了解的元素类型之外的内容提供程序、标签提供程序和操作提供程序的特定类。 与newWizards补充的内容，具体未知。 ### 3 其他 {#其他 } #### 3.1 actionSets {#actionsets }
* xml

<extension  
 point="org.eclipse.ui.actionSets">  
 <actionSet  
 id="com.acce.hyide.project.core.actionSetPrjSylixOS"  
 label="%HYIDE"  
 visible="true">  
 <action  
 class="com.acce.hyide.project.core.ActionUpload"  
 definitionId="com.acce.hyide.project.core.commandUpload"  
 id="com.acce.hyide.project.core.actionUpload"  
 label="%Upload"  
 menubarPath="project/com.acce.hyide.project.core.prjSeaEagle/Upload"  
 style="push">  
 </action>  
 </actionSet>  
 </extension>

* 说明 用来将菜单、菜单项和工具栏按钮添加到“工作台”窗口中的公共区域.但存在较大局限性, 定义整个要在操作集中引用的菜单结构十分重要。因此，如果另一个操作集定义了名为“example”的菜单，就不可靠使用现有的“example”。有必要在每个需要使用“example”菜单的操作集中重新定义它。不推荐使用。没有直接使用menu等节点方便。

#### 3.2 perspectiveExtensions 透视图扩展

* xml 暂时无详细示例

<extension  
 point="org.eclipse.ui.perspectiveExtensions">  
 <perspectiveExtension  
 targetID="\*">  
 <perspectiveShortcut  
 id="org.eclipse.debug.ui.DebugPerspective">  
 </perspectiveShortcut>  
 </perspectiveExtension>  
 </extension>

* 说明 此扩展点用来扩展其他插件注册的透视图。透视图定义窗口操作栏（菜单和工具栏）的初始内容和工作台页面内的一组初始视图及其布局。其他插件可将在选择透视图时出现的操作或视图添加至透视图。其他插件的可选添加项被追加至初始定义。

#### 3.3perspective 透视图

* xml

<extension  
 point="org.eclipse.ui.perspectives">  
 <perspective  
 class="com.acce.hyide.project.arm.ARMPerspective"  
 id="com.acce.hyide.project.arm.perspective"  
 name="%ARM\_Project">  
 </perspective>  
 </extension>

* 说明 此扩展点用来将透视图工厂添加至工作台。透视图工厂用来定义透视图的初始布局和可视操作集。用户可通过调用“窗口”菜单的“打开透视图”子菜单来选择透视图。class是继承IperspectiveFactory的实现类。

#### 3.4 startup

* xml

<extension  
 point="org.eclipse.ui.startup">  
 <startup  
 class="test\_01.StartUp"></startup>  
 </extension>

* 样例代码

import org.eclipse.ui.IStartup;  
   
import Model.Log;  
   
public class StartUp implements IStartup{  
   
 @Override  
 public void earlyStartup() {  
   
 Log.i("插件 项目 测试 启动");  
 }  
}

此项随插件启动而启动, StartUp需要implements Istartup,eclipse会自动调用earlyStartup().

#### 3.5 propertyTesters属性测试对象

* 用途未知 #### 3.6actionSet操作集 {#actionset操作集 }
* xml

<extension  
 point="org.eclipse.ui.actionSets">  
 <actionSet  
 description="功能说明\*上传目标机\*"  
 id="com.acce.hyide.project.core.actionSetPrjSylixOS"  
 label="HYIDE\*"  
 visible="true">  
 <action  
 class="com.acce.hyide.project.core.ActionUpload"  
 definitionId="com.acce.hyide.project.core.commandUpload"  
 id="com.acce.hyide.project.core.actionUpload"  
 label="上传目标机\*"  
 menubarPath="project/com.acce.hyide.project.core.prjSeaEagle/Upload"  
 style="push">  
 </action>  
 </actionSet>  
</extension>

* 说明 definitionId 值是一个commnd的Id.官方说明是: 此扩展点用来将菜单、菜单项和工具栏按钮添加到“工作台”窗口中的公共区域。将这些添加项统称为操作集并由定制透视图的用户安排出现在“工作台”窗口中。但在界面中,并未找到相关UI.

#### 3.7 builders

<extension  
 id="SeaEagleBuilder"  
 name="SeaEagleBuilder"  
 point="org.eclipse.core.resources.builders">  
 <builder  
 callOnEmptyDelta="true"  
 hasNature="true"  
 isConfigurable="true">  
 <run  
 class="com.acce.hyide.project.core.SeaEagleBuilder">  
 </run>  
 </builder>  
 </extension>

#### 3.8 natures

* xml

<extension  
 id="SeaEagleNature"  
 name="SeaEagleNature"  
 point="org.eclipse.core.resources.natures">  
 <runtime>  
 <run  
 class="com.acce.hyide.project.core.SeaEagleNature">  
 </run>  
 </runtime>  
 <builder  
 id="com.acce.hyide.project.core.SeaEagleBuilder">  
 </builder>  
</extension>

* 说明 runtime 表示自启插件,随eclipse开启而自启. builder这里的builder用途未知

### 4 调试

#### 4.1 launchConfigurationType 启动配置类型

* 交互方式
* xml

<extension  
 point="org.eclipse.debug.core.launchConfigurationTypes">  
 <launchConfigurationType  
 id="com.acce.hyide.project.debug.launchConfigurationType"  
 name="ARM操作系统程序"  
 public="true">  
 </launchConfigurationType>  
 </extension>

这里没有指定的类,数据是如何可视化的?

#### 4.2 launchDelegates启动代表

* 交互方式
* xml

<extension  
 point="org.eclipse.debug.core.launchDelegates">  
 <launchDelegate  
 delegate="com.acce.hyide.project.debug.delegate.ProjectDebugDelegate"  
 delegateDescription="海鹰ARM操作系统项目调试\*\*"  
 id="com.acce.hyide.project.debug.projectDebugDelegate"  
 modes="debug,run"  
 name="%launchDelegate.name"  
 sourceLocatorId="org.eclipse.cdt.debug.core.sourceLocator"  
 sourcePathComputerId="org.eclipse.cdt.debug.core.sourcePathComputer"  
 type="com.acce.hyide.project.debug.launchConfigurationType">  
 </launchDelegate>  
 <launchDelegate  
 delegate="org.eclipse.cdt.dsf.gdb.launching.GdbLaunchDelegate"  
 delegateDescription="弹载ARM操作系统手动应用调试\*\*"  
 id="com.acce.hyide.project.debug.launchManuDelegate"  
 modes="debug"  
 name="弹载ARM操作系统手动应用调试\*"  
 sourceLocatorId="org.eclipse.cdt.debug.core.sourceLocator"  
 sourcePathComputerId="org.eclipse.cdt.debug.core.sourcePathComputer"  
 type="com.acce.hyide.project.debug.launchConfigurationType">  
 </launchDelegate>  
 ~~~  
</extension>

* 说明 此处的type，是launchConfigurationType的id。 delegate是对应的delegate元素实现类，要继承GdbLaunchDelegate。要实现目标机IP、端口号等配置。

#### 4.3 launchConfigurationTabGroups启动配置标签组

* xml

<extension  
 point="org.eclipse.debug.ui.launchConfigurationTabGroups">  
 <launchConfigurationTabGroup  
 class="org.eclipse.cdt.debug.internal.ui.launch.PlaceHolderLaunchConfigurationTabGroup"  
 id="com.acce.hyide.project.debug.remoteApplicationLaunchTabGroup"  
 type="com.acce.hyide.project.debug.launchConfigurationType">  
 <launchMode  
 mode="debug">  
 </launchMode>  
 </launchConfigurationTabGroup>  
 <launchConfigurationTabGroup  
 class="org.eclipse.cdt.debug.internal.ui.launch.PlaceHolderLaunchConfigurationTabGroup"  
 id="com.acce.hyide.project.debug.remoteApplicationRunLaunchTabGroup"  
 type="com.acce.hyide.project.debug.launchConfigurationType">  
 <launchMode  
 mode="run">  
 </launchMode>  
 </launchConfigurationTabGroup>  
 </extension>

* 说明 此扩展点提供了一种机制来将一组选项卡添加至某种类型的启动配置的启动配置对话框 class 是继承AbstractLaunchConfigurationTabGroup的实现方法，一般通过以下方式实现。 @Override public void createTabs(ILaunchConfigurationDialog dialog, String mode) { setTabs(new ILaunchConfigurationTab[0]); } 即设置launchDelegates到launchConfigurationType中。