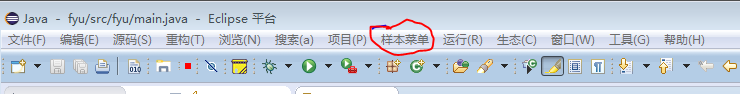
# 菜单类

## 、menu

* 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.

## 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,

## 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。

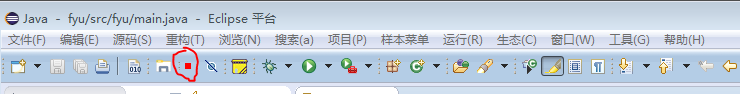
TestHandler 要继承AbstractHandler, public Object execute(ExecutionEvent event) throws ExecutionException {}是系统调入接口.

* 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**;  }    } |

## 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.

## 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> |

## 右键菜单

### 右键任意菜单

|  |
| --- |
| <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

### 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

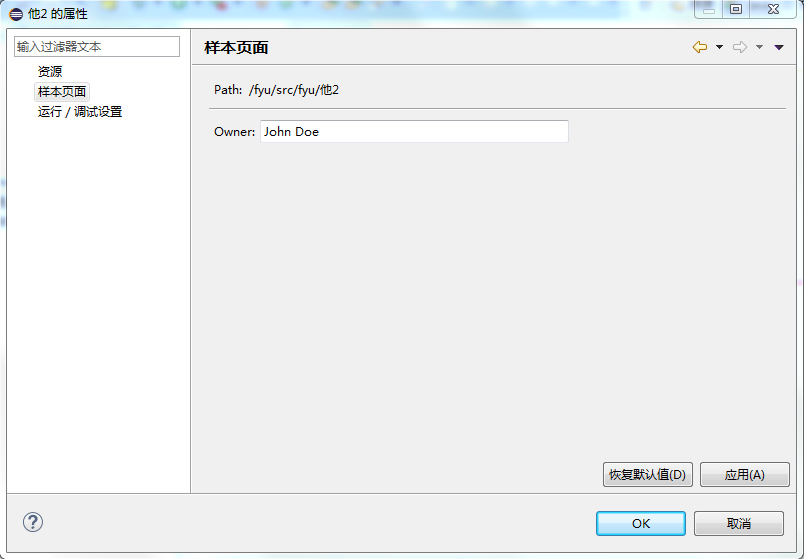
|  |
| --- |
| import org.eclipse.core.internal.runtime.Activator;  import org.eclipse.jface.action.IAction;  import org.eclipse.jface.dialogs.MessageDialog;  import org.eclipse.jface.viewers.ISelection;  import org.eclipse.swt.widgets.Shell;  import org.eclipse.ui.IObjectActionDelegate;  import org.eclipse.ui.IWorkbenchPart;  @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.

### 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

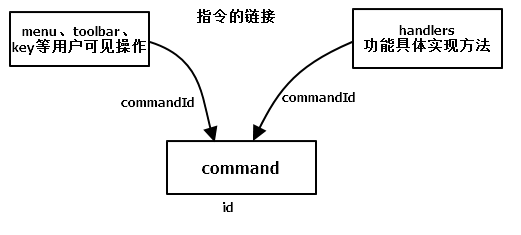
|  |
| --- |
| import org.eclipse.core.resources.IResource;  import org.eclipse.core.runtime.CoreException;  import org.eclipse.core.runtime.Platform;  import org.eclipse.core.runtime.QualifiedName;  import org.eclipse.swt.SWT;  import org.eclipse.swt.layout.GridData;  import org.eclipse.swt.layout.GridLayout;  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;  }  } |

表 1-1

* PropertyPage

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

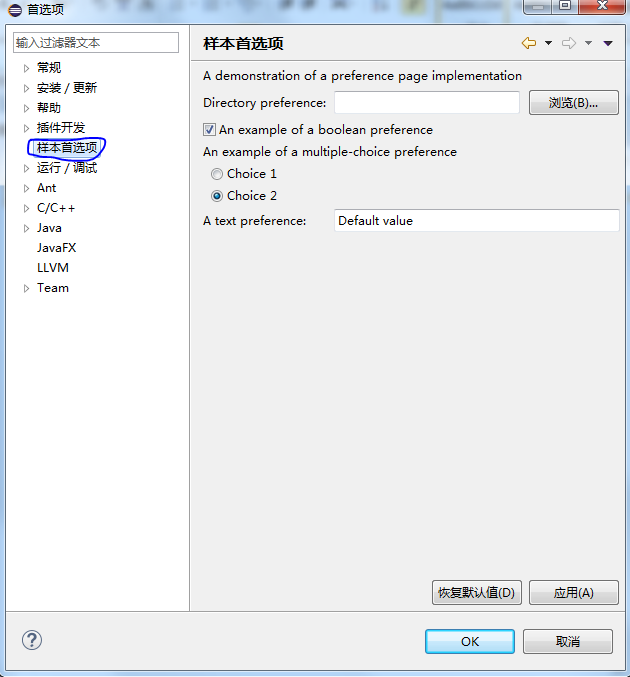
## command的链接作用



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。

## 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) {  }    } |

* b. 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");  }  } |

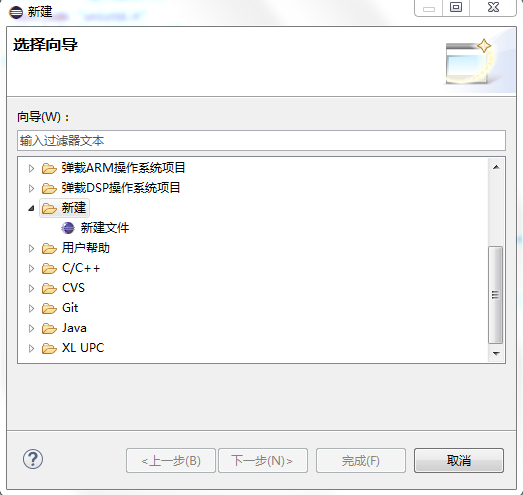
说明:

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

# 文件操作

## 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> |

表 2-1

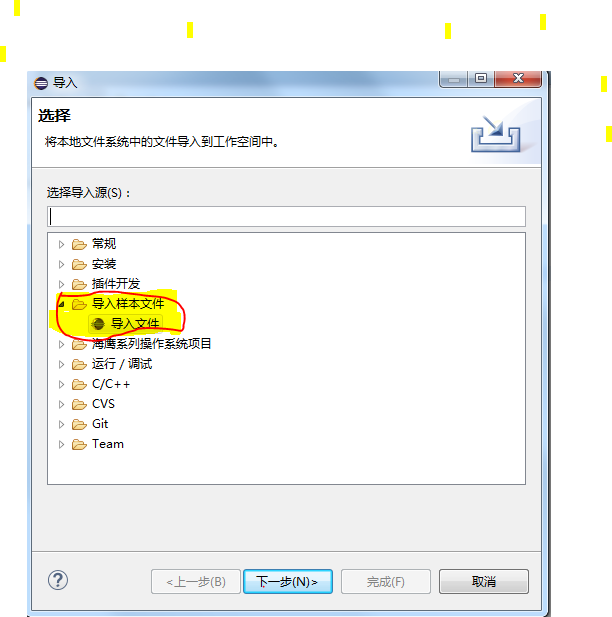
* 说明

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

## 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

## 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

## 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

## 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。

# 其他

## 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。

## 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。

## perspective 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。

## 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().

## propertyTesters属性测试对象

* 用途未知

## 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.

## 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> |

## 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

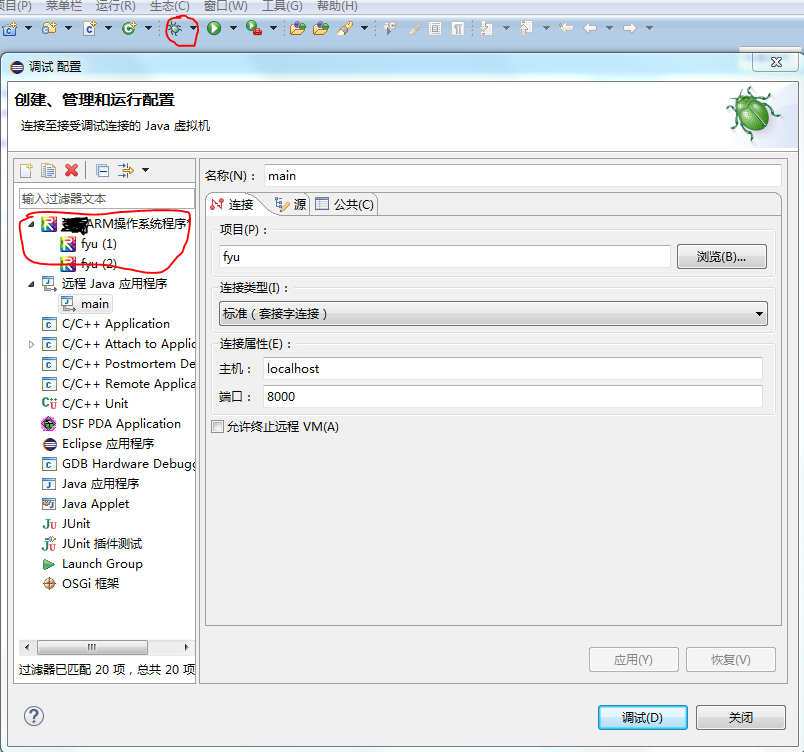
runtime 表示自启插件,随eclipse开启而自启.

builder这里的builder用途未知

# 调试

## launchConfigurationType 启动配置类型

* 交互方式



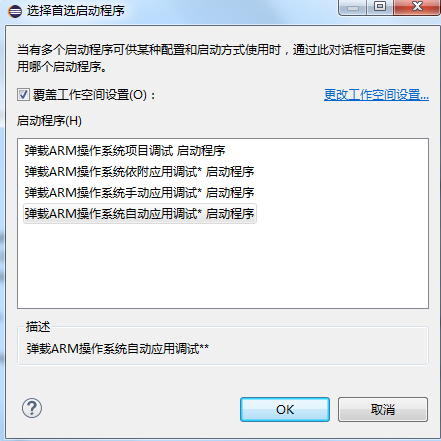
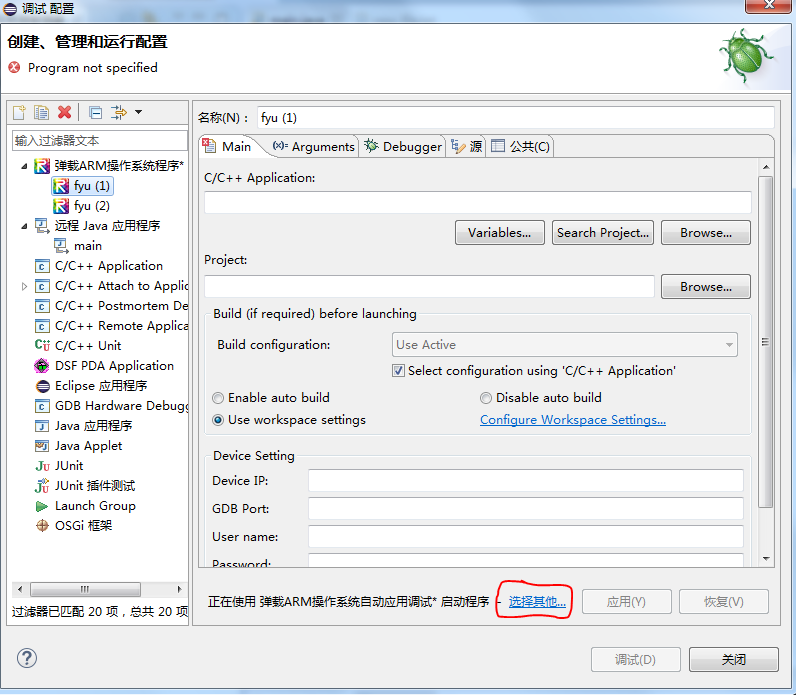
* xml

|  |
| --- |
| <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?

## 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。

## 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。