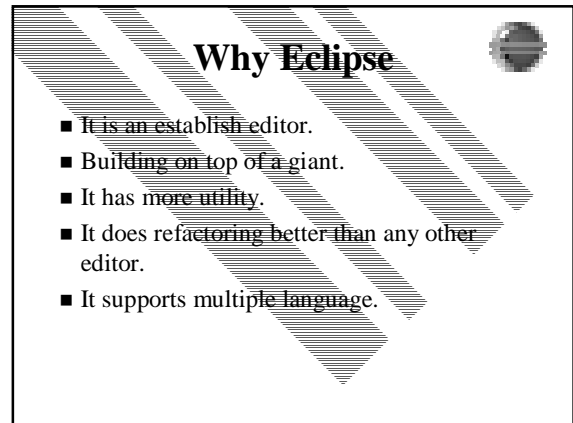


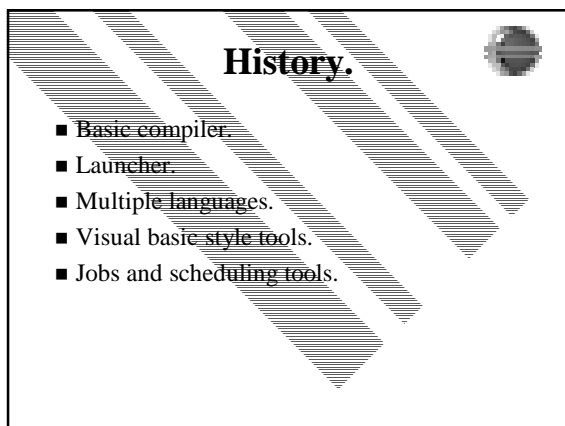
## Eclipse Plug-in Architecture

by  
Chun ping Wang.



## Why Eclipse

- It is an established editor.
- Building on top of a giant.
- It has more utility.
- It does refactoring better than any other editor.
- It supports multiple languages.



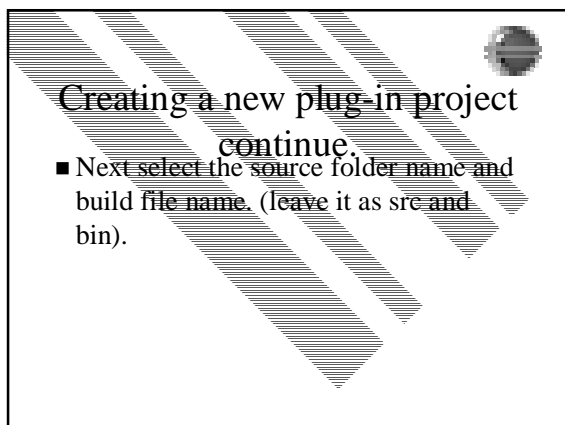
## History.

- Basic compiler.
- Launcher.
- Multiple languages.
- Visual basic style tools.
- Jobs and scheduling tools.



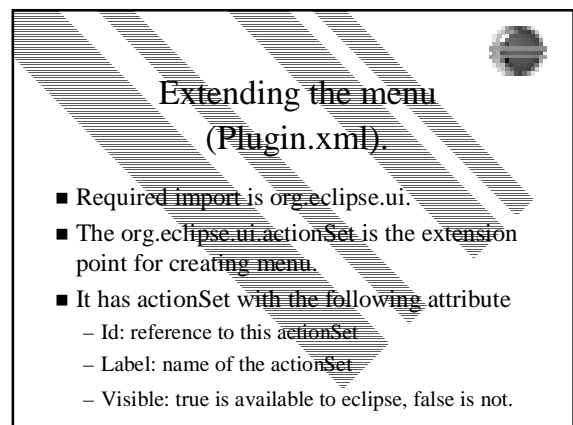
## Creating a new plug-in

- Go to file, new, plug-in project.
- To avoid conflict, unselect use default directory, and create a directory with the same name as your project title inside eclipse.
- Example c:/eclipse/cs491project and title to be cs491 project.



## Creating a new plug-in project continue.

- Next select the source folder name and build file name. (leave it as src and bin).



## Extending the menu (Plugin.xml).

- Required import is org.eclipse.ui.
- The org.eclipse.ui.actionSet is the extension point for creating menu.
- It has actionSet with the following attribute
  - Id: reference to this actionSet
  - Label: name of the actionSet
  - Visible: true is available to eclipse, false is not.

## Extending the menu (Plugin.xml) sub-xml tag of actionSets.

- Menu with the following attribute
  - id: reference to this menu
  - label: the name you want to display in eclipse menu
  - path: path of this menu

## Extending the menu (Plugin.xml) sub-xml tag of actionSets

- Action
  - id: the reference to this action.
  - Label: name of the action
  - class: (class must implement IWorkbenchWindowActionDelegate)
  - toolbarpath: (where do you want it to appear in the toolbar, you can specify any arbitrary name)
  - menubarPath: (menu id/name of the file)
  - Tooltip: information to help user know what this thing does.

## Sample actionSet in plugin.xml

- <actionSet
- label = "helloworld.actionSet"
- visible = "true"
- id = "helloworld.actionSet"
- >
- </actionSet>

## Sample menu in plugin.xml

- <menu
- label = "helloworld"
- id = "helloworld">
- </menu>

## Sample action in plugin.xml

- <Action
- Id="helloworld.action"
- Visible="true"
- Class="helloworld.hello"
- Menubarpath="helloworld/hello">
- </Action>
- You can view the whole plugin in the example files.

## Implementing IWorkbenchWindowActionDelegate

Run (IAction action):

Triggers the program to run inside eclipse. The parameter action is the non-UI side of the command which handles presentation portion of the action.

selectionChanged(IAction action, Iselection selection);

Notifies this action delegate that the selection in the workbench has changed.

You can basically notify presentation properties.

## Implementing IWorkbenchWindowActionDelegate

Dispose();  
dispose this action delegate.  
Suggested idea is to create an empty  
IWorkbenchWindow and close it on  
dispose().  
Init(IWorkbenchWindow window)  
initialize with the window it will work on.

## Extending the view(Plugin.xml)

Org.eclipse.ui.views is the extension point for  
creating eclipse views for Windows → Show view  
in the menu bar.

You create the view with org.eclipse.swt.widgets

It has the <category> element with the following  
attribute

parentCategory: note you have to reference to the parentCategory's id  
AND not name.  
name: (how you want it to appear in eclipse).  
id : the reference to this category.

## Extending the view(Plugin.xml).

It has the <view> element with the following  
attribute

Category: what category reference does it belong  
to.

Id: the reference to this view.

Name: Name of this view.

Class: any class that implements  
ViewPart(Default, suggested for beginner) or  
IViewPart.

Icons: any icons you want to represent this view.

## Sample category in Plugin.xml

```
<category  
  name="Sample helloworld view"  
  id="helloworldview">  
</category>
```

## Sample view in Plugin.xml

```
<view  
  name="Hello View"  
  category="helloworldview"  
  class="helloworld.helloworldv  
  iew"  
  
  id="helloworld.Helloworldview  
  ">  
</view>
```

## Implementing ViewPart interface

createPartControl(Composite Parent): creates  
the view control. It is suggested that you use  
eclipse own widgets library instead of the  
standard awt library.

setFocus(); This function is to accept focus.