PROJECT: BOGOR – JAVA ENVIRONMENT FOR ECLIPSE

DELIVERABLE: COMPONENT DESIGN

Date: December 5, 2005

Prepared by: Yong Peng
Major Professor: Robby
Kansas State University
# Table of Content

1. **INTRODUCTION** .................................................................................................................. 3

2. **INCREMENTAL COMPILATION PLUG-IN** .......................................................................... 3
   2.1. **OVERVIEW** ......................................................................................................................... 3
   2.2. **PACKAGE - EDU.KSU.CIS.PROJECTS.BOGOR.INCREMENTAL** ........................................... 3
       2.2.1. **Class – BytecodeToBirIncrementalPlugin** ................................................................. 3
   2.3. **PACKAGE - EDU.KSU.CIS.PROJECTS.BOGOR.INCREMENTAL.BUILDER** ......................... 4
       2.3.1. **Class – BytecodeToBirBuilder** ..................................................................................... 5
       2.3.2. **Class – BytecodeToBirNature** ..................................................................................... 7
       2.3.3. **Class – ToggleNatureAction** ..................................................................................... 7
   2.4. **PACKAGE - EDU.KSU.CIS.PROJECTS.BOGOR.VM.TRANSLATOR** ................................... 8
       2.4.1. **Class – BirHandler** .................................................................................................... 9

3. **BOGOR VM VIEWER PLUG-IN** .............................................................................................. 9
   3.1. **OVERVIEW** ......................................................................................................................... 9
   3.2. **PACKAGE - EDU.KSU.CIS.PROJECTS.BOGOR.UTIL** ......................................................... 10
       3.2.1. **Class – EclipseUtils** .................................................................................................. 10
       3.2.2. **Class – JdtUtils** ......................................................................................................... 10
   3.3. **PACKAGE - EDU.KSU.CIS.PROJECTS.BOGOR.VIEWS** ..................................................... 11
       3.3.1. **Class – BytecodeToBirPlugin** ...................................................................................... 12
       3.3.2. **Class – BytecodeToBirView** ...................................................................................... 13
       3.3.3. **Class – EditorListener** ............................................................................................. 15
   3.4. **PACKAGE - EDU.KSU.CIS.PROJECTS.BOGOR.VM.TRANSLATOR** ................................... 17
       3.4.1. **Class – BBVisitor** ...................................................................................................... 17
       3.4.2. **Class – GeneratedBir** ............................................................................................. 17

4. **LAUNCHER PLUG-IN** ............................................................................................................. 18
   4.1. **OVERVIEW** ......................................................................................................................... 18
   4.2. **PACKAGE - EDU.KSU.CIS.PROJECTS.BOGOR.LAUNCHCONFIGURATIONS** ..................... 19
       4.2.1. **Class – BogorVMMConfigTab** .................................................................................... 19
       4.2.2. **Class – BogorVMMainTab** ......................................................................................... 19
   4.3. **PACKAGE - EDU.KSU.CIS.PROJECTS.BOGOR.LAUNCHING** ........................................... 21
       4.3.1. **Class – BogorVMLaunchConfigurationDelegate** ....................................................... 22
       4.3.2. **Class – BogorVMTabGroup** ....................................................................................... 22
       4.3.3. **Class – LauncherMessages** ....................................................................................... 23
       4.3.4. **Class – LaunchingPlugin** ....................................................................................... 24

5. **ERROR TRACE PLUG-IN** ....................................................................................................... 25
1. Introduction

The purpose of this document is to provide a component design for the Bogor – Java Environment for Eclipse. This component design document presents the internal design for each component.

2. Incremental Compilation Plug-in

2.1. Overview

This plug-in has three packages. Figure 2.1.1 is a component diagram that shows the relationships among three packages.

![Figure 2.1.1 incremental plug-in component diagram](image)

2.2. Package - edu.ksu.cis.projects.bogor.incremental

BytecodeToBirIncrementalPlugin is the only class in this package. This is the main plug-in class to be used on desktop. It extends the AbstractUIPlugin class, which is provided by the Eclipse API. This class handles the life cycle for the plug-
in. It provides the error and log method to log errors. Eclipse uses this class and the plug-in configuration XML to register this plug-in into eclipse environment. The configuration plug-in XML file contains all information required for this plug-in, such as plug-in ID, version number, dependencies to other plug-in, etc.

2.2.1. Class – BytecodeToBirIncrementalPlugin

![Figure 2.2.1 BytecodeToBirIncrementalPlugin class](image)

**Methods:**

- `start(BundleContext):` This method is called upon plug-in activation.
- `stop(BundleContext):` This method is called when the plug-in is stopped.
- `getDefault():` returns the shared instance.
- `getImageDescriptor(path : Logical View::java::lang::String) : ImageDescriptor` returns an image descriptor for the image file at the given plug-in relative path.
- `getShell():` returns the workspace instance.
- `error(messageID : Logical View::java::lang::String, error : Throwable) : void` writes error message into log and display it.
- `log(error : Throwable, statusID : int) : void` writes error message into log.
2.3. Package - edu.ksu.cis.projects.bogor.incremental.builder

This package includes the builder classes. The classes in this package are the core for this plug-in.

2.3.1. Class – BytecodeToBirBuilder

This class defines the incremental builder. It extends the IncrementalProjectBuilder class, which is provided by the Eclipse API. This class handles full or delta build. There are two helper methods to create and delete BIR by calling methods from class BirHandler. Figure 2.3.1.1 present the sequence of build process.
1. full build
   - eclipse listens the action: full build
   - 1. if it's a full build
      - 1.1. fullBuild(monitor : IProgressMonitor)
      - 1.2. add resource visitor
      - 1.3. createBIR(resource : IResource)
      - 1.4. writeBIR(resource : IResource, monitor : IProgressMonitor)

2. delta build
   - eclipse listens the action: delta build
   - 2. if it's a delta build
      - 2.1. incrementalBuild(delta : IResourceDelta, monitor : IProgressMonitor)
      - 2.2. add delta visitor
      - 2.3. create/delete BIR
      - 2.4. if delete: deleteBIR(resource : IResource, monitor : IProgressMonitor)
      - 2.5. if write: writeBIR(resource : IResource, monitor : IProgressMonitor)

Figure 2.3.1.1 builder sequence diagram

Figure 2.3.1.2 BytecodeToBirBuilder class
Attributes:

- BUILDER_ID: unique ID for this builder

Methods:

- build(…): this method is invoked if any changes made.
- fullBuild(…): this method deals with full build. It accepts ResourceVisitor to the current project.
- incrementalBuild(…): this method deals with delta build. It accepts DeltaVisitor to the current project.
- createBir(…): this method calls BirHandler.writeBir() to create BIR file.
- deleteBir(…): this method calls BirHandler.deleteBir() to delete BIR file.

2.3.2. Class – BytecodeToBirNature

This is the project nature’s configure class. It handles add or remove nature from menu. This class implements the eclipse API interface IProjectNature.

![BytecodeToBirNature Class](image)

**Figure 2.3.2 BytecodeToBirNature class**

Attributes:

- NATURE_ID: unique ID for this project nature

Methods:
configure(): This method configures this nature for its project. The workspace calls this method when natures are added to the project.

deConfigure(): This method de-configures this nature for its project. The workspace calls this method when natures are removed from the project.

getProject(): This method returns the project to which this nature applies.

setProject(…): This method sets the project to which this nature applies.

2.3.3. Class – ToggleNatureAction

This class toggles the nature on a project. It implements the Eclipse API interface ObjectActionDelegate.

Figure 2.3.2 ToggleNatureAction class

Methods:

- run(…): This method is called when the action has been triggered.
- selectionChanged(…): This method notifies action delegate that the selection in the workbench has changed.
- setActivePart(…): This method sets the active part for the delegate.
- toggleNature(…): toggles Byte code to BIR nature on a project.
2.4. Package - edu.ksu.cis.projects.bogor.vm.translator

2.4.1. Class – BirHandler

This is the helper class to delete and write BIR. This class is used by the BytecodeToBirBuilder to create and delete BIR from file system. There is a helper method to help create and delete method.

![BirHandler class diagram]

Methods:

- `getClassFullyQualifiedName(resource: IResource): Logical View::java::lang::String`:
  Returns the full qualified name for given resource.

- `writeBIR(resource: IResource, monitor: IProgressMonitor): void`:
  Write compiled BIR file into “bir” directory.

- `deleteBIR(resource: IResource, monitor: IProgressMonitor): void`:
  Delete compiled BIR file from “bir” directory.

3. Bogor VM Viewer Plug-in

3.1. Overview

This plug-in has three packages. Figure 3.1.1 is a component diagram that shows the relationships.
Figure 3.1.1 Bogor VM Viewer plug-in component diagram

3.2. Package - edu.ksu.cis.projects.bogor.util

3.2.1. Class – EclipseUtils

This is a helper class. BytecodeToBirView class uses this class to highlight Java code in Java editor. BytecodeToBirView class also uses it to obtain Java editors, Java elements, selected texts in Java editor, and Java package names.

Figure 3.2.1 EclipseUtils class

Methods:

- `getActiveEditor()`: returns current active editor in workbench.
- `getJavaInput(IEditorPart)`: returns editor input as IJavaElement.
- `selectInEditor(ITextEditor, int, int)`: makes selection in editor.
- `getSelection(ISelectionProvider)`: returns TextSelection or null if provider does not provide TextSelection's
- `getJavaPackageName(IJavaElement)`: returns full package name

### 3.2.2. Class – JdtUtils

This is a helper class. BytecodeToBirView class uses this class to obtain the enclosing type for given Java elements, to create input stream from Java elements, and to obtain the Java elements from given offset.

**Figure 3.2.2 JdtUtils class**

**Methods:**

- `createMethodSignature(IMethod)`: creates method signature.
- getEnclosingType(IJavaElement): returns first ancestor with IJavaElement TYPE, element type, or null
- getElementAtOffset(IJavaElement, ITextSelection): returns IJavaElement for the given offset.
- getElementName(IJavaElement): returns element name for a given Java element.
- getByteCodePath(IJavaElement): returns full operation system specific file path for a given Java element.
- createInputStream(IJavaElement): returns new generated input stream for given Java element.
- getFullBytecodeName(IClassFile): returns full qualified byte code name for the given Java class.
- getClassLoader(IJavaElement): returns class loader for a given Java element.
- isAbstractOrInterface(IJavaElement): check if Java element is an interface or abstract method.

3.3. Package - edu.ksu.cis.projects.bogor.views

3.3.1. Class – BytecodeToBirPlugin

This is the main plug-in class to be used in desktop. It extends the AbstractUIPlugin class, which is provided by the Eclipse API. This class handles the life cycle for the plug-in. It provides the error and log method to log errors. It also equips methods to retrieve values from resource bundlers. Eclipse uses this class and the plug-in configuration XML to
register this plug-in into eclipse environment. The configuration plug-in XML file contains all information required for this plug-in, such as plug-in ID, version number, dependencies to other plug-in, etc.

![BytecodeToBirPlugin class](image)

**Figure 3.3.1 BytecodeToBirPlugin class**

**Methods:**

- `start(BundleContext):` This method is called upon plug-in activation.
- `Stop(BundleContext):` This method is called when the plug-in is stopped.
- `getDefault():` returns the shared instance.
- `getResourceString(…):` returns the string from the plugin resource bundle, or 'key' if not found.
- `getResourceBundle():` Returns the plugin resource bundle.
- `getImageDescriptor(…):` Returns an image descriptor for the image file at the given plug-in relative path.
- `getShell():` Returns the workspace instance.
- `error(…):` write error message into log and display it.
- log(...): write error message into log.

3.3.2. Class – BytecodeToBirView

This class creates the viewer. It extends the ViewPart class, which is provided by the Eclipse API. This class builds the user interface to display BIR. It makes use class EditorListener to catch the user or eclipse event and handles them accordingly.
Figure 3.3.2 BytecodeToBirView class

Attributes:

- textViewer: TextViewer instance
- textControl: StyledText instance
- doLinkWithEditor: a Boolean value used to decide if view linked with editor.
Methods:

- init(IViewSite): is to initial view part.
- createPartControl(Composite): creates part and setup all listeners.
- setFocus(): sets focus to this view.

3.3.3. Class – EditorListener

This class listens to all events. This class implements eclipse API interface
ISelectionChangedListener,ISelectionListener, IFileBufferListener,
IPartListener2.
Methods:

- `dispose()`: cleans view reference
- `selectionChanged(IWorkbenchPart, ISelection)`: is to call view method `handleSelectionChanged`.
- `dirtyStateChanged(IFileBuffer, boolean)`: to call view method `handleBufferIsDirty`.
- `partClosed(IWorkbenchPartReference)`: to call view method `handlePartHidden`.
- `partHidden(IWorkbenchPartReference)`: to call view method `handlePartHidden`.
- `partOpened(IWorkbenchPartReference)`: to call view method `handlePartVisible`.
- `partVisible(IWorkbenchPartReference)`: to call view method `handlePartVisible`.

3.4. Package - edu.ksu.cis.projects.bogor.vm.translator

3.4.1. Class – BBVisitor

BytecodeToBirView uses this class to obtain `GeneratedBir`.

![BBVisitor](image)

**Figure 3.3.3 EditorListener class**

Methods:

- `getGeneratedBir(type : IJavaElement) : GeneratedBir`
3.4.2. Class – `GeneratedBir`

This class contains the information of BIR. `BytecodeToBirView` uses this class to retrieve BIR information, such as to get source line, to get offset in BIR.

```
GeneratedBir
(from translator)

getSourceLine() : int
getBIROffset() : int
getText() : String
```

**Figure 3.4.2 GeneratedBir class**

**Methods:**

- `getSourceLine()`: returns source line number
- `getBIROffset()`: returns BIR offset
- `getText()`: returns BIR text as a string

4. Launcher Plug-in

4.1. Overview

This plug-in has two packages. **Figure 4.1.1** is a component diagram that shows the relationships.
4.2. Package - edu.ksu.cis.projects.bogor.launchConfigurations

4.2.1. Class – BogorVMConfigTab

This class creates the configure tab for launcher. It builds the interface for
config tab, listens to user’s actions, and performs the task accordingly.

This class extends Eclipse AbstractLaunchConfigurationTab.

Attributes:

- fParametersTable: table instance
- fParametersAddButton: button instance
- fParametersRemoveButton: button instance
- fParametersEditButton: button instance
- fParametersLoadDefaultButton: button instance

Figure 3.3.2 BogorVMConfigTab class
resourceBundle: resource bundle instance

Methods:

- getImage(): returns image
- activated(ILaunchConfigurationWorkingCopy): returns a notification that this tab has become the active tab in the launch configuration dialog.
- deactivated(ILaunchConfigurationWorkingCopy): returns a notification that this tab is no longer the active tab in the launch configuration dialog.
- setDefaults(ILaunchConfigurationWorkingCopy): initializes the given launch configuration with default values for this tab
- initializeFrom(ILaunchConfiguration): initializes this tab's controls with values from the given launch configuration.
- getDefaultConfig(): gets default values from property file.
- performApply(ILaunchConfigurationWorkingCopy): copies values from this tab into the given launch configuration.
- getName(): returns the tab name

4.2.2. Class – BogorVMMainTab

This class creates the main tab for launcher. It builds the interface for main tab, listens to user’s actions, and performs the task according to user’s action. This class extends eclipse JavaLaunchConfigurationTab.
Figure 3.3.2 BogorVMMainTab class

Attributes:

- fProjText: text widget instance
- fProjButton: button instance
- fMainText: text widget instance
- fSearchButton: button instance

Methods:

- createControl(Composite): creates tab
- isValid(ILaunchConfiguration): validates configuration
- setDefaults(ILaunchConfigurationWorkingCopy): sets default values
- getImage(): returns image
- activated(ILaunchConfigurationWorkingCopy): returns a notification that this tab has become the active tab in the launch configuration dialog.
4.3. Package - edu.ksu.cis.projects.bogor.launching

4.3.1. Class – BogorVMLaunchConfigurationDelegate

This configuration delegate receives information from tabs and invokes Bogor. This class implements the ILaunchConfigurationDelegate interface, which is provided by Eclipse API. The launch method retrieves configuration from work copy and then invokes Bogor to do model checking. BogorVMLaunchConfigurationDelegate performs launching for a BogorVM type of launch configuration.

![BogorVMLaunchConfigurationDelegate class](image)

**Figure 3.3.2 BogorVMLaunchConfigurationDelegate class**

**Methods:**

- deactivated(ILaunchConfigurationWorkingCopy): returns a notification that this tab is no longer the active tab in the launch configuration dialog.
- initializeFrom(ILaunchConfiguration): initializes this tab's controls with values from the given launch configuration.
- performApply(ILaunchConfigurationWorkingCopy): copies values from this tab into the given launch configuration.
- getName(): returns the tab name
- `launch(ILaunchConfiguration, String, ILaunch, IProgressMonitor)`: launches the program
- `getProperties(ILaunchConfiguration, Properties)`: returns properties from launcher pad
- `checkModel(Properties, System, String)`: check Bogor
- `getSymbolTable(Properties, IBogorConfiguration, String, System)`: returns symbol table.
- `printErrors(Collection<FileMessage>, PrintWriter)`: print errors in eclipse console

4.3.2. Class – BogorVMTabGroup

This class groups application tabs into launcher. This class extents AbstractLaunchConfigurationTabGroup. The `createTabs` method creates all required tabs for Bogor VM launcher.

```java
BogorVMTabGroup
createTabs(dialog : ILaunchConfigurationDialog, mode : Logical View::java::lang::String) : void
```

**Figure 3.3.2 BogorVMTabGroup class**

**Methods:**

- `createTabs`: creates main, config, and common tabs.

4.3.3. Class – LauncherMessages

This class contains all messages used in launcher. It reads the information from LauncherMessages.properties. It extends the NLS class, which is provided by the Eclipse API.
Figure 3.3.2 BytecodeToBirView class

Attributes:

- **BogorVMLauncher_ArgumentTab_Parameters_Button_Add_Text**: adds button text
- **BogorVMLauncher_ArgumentTab_Parameters_Button_Edit_Text**: edit button text
- **BogorVMLauncher_ArgumentTab_Parameters_Button_Remove_Text**: remove button text
- All other attributes are similar to the above three attributes.

4.3.4. Class – LaunchingPlugin

This is the main plug-in class to be used on desktop. It extends the AbstractUIPlugin class, which is provided by the Eclipse API. It handles the life cycle of launching plug-in. Eclipse uses this class and the plug-in configuration XML to register this plug-in into eclipse environment. The configuration plug-in XML file contains all information required for this
plug-in, such as plug-in ID, version number, dependencies to other plug-in, etc.

![LaunchingPlugin](from launching)

### Methods:

- `start(BundleContext):` This method is called upon plug-in activation.
- `Stop(BundleContext):` This method is called when the plug-in is stopped.
- `getDefault():` returns the shared instance.
- `getImageDescriptor(path : Logical View::java::lang::String) : ImageDescriptor`:

![Figure 4.3.4](methods)

5. **Error Trace Plug-in**

   This error trace plug-in reuses Bogor UI plug-in and makes changes on Java class “CounterExampleView” to highlight Java code instead of highlighting Bogor trails file. Figure 5 presents the methods changed to implement the highlight function.
CounterExampleView
(from views)

- openJavaFile(path : String, typeName : String) : EditorPart
- getJavaTypeName(location : String) : String
- getJavaPath(location : String) : String
- isReturnNull(location : String) : boolean
- getJavaSourceLineNumber(location : String) : int
- replaceWithPattern(input : String) : String
- formatLineNumber(input : String) : String
- highlightLoc(reset : boolean, lineNumber : int, path : String, typeName : String) : void
- updateLoc() : void
- updateTrees() : void

Figure 5

Methods:

- openJavaFile(...): This method is used to open an editor for selected Java file.
- getJavaTypeName(...): returns type name for given location.
- getJavaPath(): returns Java path for given location.
- isReturnNull(): checks if it needs to return null for given location.
- getJavaSourceLineNumber(String): returns Java source line number for given location.
- replaceWithPattern(String) : returns a new string by applying a pattern on the given string.
- formatLine_number(String): formats the given string.