The Three-Page Tutorial [tm] Series

Creating Cheat Sheets in Eclipse 3.0

Carlos Valcarcel, Director of Technology

Trivera Technologies LLC

The Three Page Tutorial – Creating Cheat Sheets in Eclipse 3.0

This tutorial will create a simple cheat sheet with one manual step and one step that includes a cheat sheet action. For extra credit you can add a Cheat Sheet listener that will be called during the Cheat Sheet's lifecycle. Print the following three pages as a reference.

The Steps

1. Start **Eclipse**.
2. Press **Ctrl+N** to open the **New** dialog. Select **Plug-in Project**. Click **Next**.
3. Enter a **Project Name** of **com.triveratech.cheatsheet**. Click **Next**.
4. Leave the information in the **Plug-in Content** page alone. Click **Finish**.
5. The **Plug-in Manifest Editor** will open on **plugin.xml** for **com.triveratech.cheatsheet** . Click the **Extensions** tab.
6. Click **Add**. When the **New Extension** dialog opens *uncheck* **Show Only Extension Points from the Required Plug-ins**. From the **Available Extension Points** list scroll down and select **org.eclipse.ui.cheatsheets.cheatSheetContent**. Click **Finish**.
7. If the extension is not selected in the **All Extensions** list to the left select it. In the Extensions Detail area to the right enter:

   Id: com.triveratech.cheatsheet.example
   Name: TriveraTech Example Cheat Sheet

8. Save the file by pressing **Ctrl+S**.
9. Right-click to open the popup menu. Select **New->Category**. In the **Extensions Element Detail** area to the right enter:

   Id: com.triveratech.cheatsheet.category1
   Name: A Trivera Technologies Cheat Sheet

   The **Id** field should already contain the proper information. Leave **parentCategory** blank.

10. Right-click to open the popup menu. Select **New->Cheatsheet**. In the **Extensions Element Detail** area to the right enter:

    Id: com.triveratech.cheatsheet.cheatsheet1
    Name: My First Cheat Sheet
    Category: com.triveratech.cheatsheet.category1
    contentFile: cheatsheets/MyFirstCheatSheet.xml

11. In the **Package Explorer** view create a folder under the **com.triveratech.cheatsheet** project named **cheatsheets**. Press **Ctrl+N** to open

the **New** dialog, select **Simple->Folder** and click **Next**. Select the project and in the **Folder Name** field enter **cheatsheets**. Click **Finish**.

12. In the **Package Explorer** view select the new **cheatsheets** folder and create a file named **MyFirstCheatSheet.xml** after pressing **Ctrl+N** and selecting **Simple->File**.
13. Enter the following into the file **MyFirstCheatSheet.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>
<cheatsheet title="My First Cheat Sheet">
  <intro>
    <description>
      This is the introduction to your first cheat sheet.
    </description>
  </intro>
  <item title="Step 1">
    <description>
      This is the description of how to execute the first step.
    </description>
  </item>
</cheatsheet>
```

14. Run the **Runtime Workbench** (**Run->Run As->Runtime Workbench**) and select **Help->Cheat Sheets** from the **Runtime Workbench**. Click **OK** to see the Cheat Sheet open in the view to the right.
15. Exit the **Runtime Workbench**.
16. Select the **src** directory of the **com.triveratech.cheatsheet** project from the **Package Explorer** view.
17. Press **Ctrl+N**, select **Class** and click **Next**.
18. In the **Java Class** page enter:

    Package: com.triveratech.cheatsheet.action
    Name: SimpleAction

19. Click **Browse** next to **Superclass**. When the **Superclass Selection** dialog opens select **Action** from **org.eclipse.jface.action** and click **OK**.
20. Click **Add** next to **Interfaces**. When the **Implemented Interfaces Selection** dialog opens select **ICheatSheetAction** and click **OK**.
21. Click **Finish** to close the **New Java Class** dialog.
22. In **SimpleAction** add the **run()** method declared in **ICheatSheetAction**:

```
public void run(String[] params, ICheatSheetManager manager) {
  MessageDialog.openInformation(PlatformUI.getWorkbench()
    .getActiveWorkbenchWindow().getShell(),
    "My First Cheat Sheet Action",
    "The incoming parameter is " + params[0]);
}
```

23. Enter the following after the last item, but before the closing **cheatsheet** tag:

```
<item title="Step 2">
  <action pluginId="com.triveratech.cheatsheet"
     class="com.triveratech.cheatsheet.action.SimpleAction"
     param1="This is my action parameter."
  <description>
    This is a step that includes a Cheat Sheet Action.
  </description>
</item>
```

24. Restart the Runtime Workbench and select your cheat sheet from **Help->Cheat Sheets**. The **Introduction** step has a **right-pointing arrow** that will take you to **Step 1**, **Step 1** has a **check** so that you can let it know when you **complete** that step and **Step 2** has a **right-pointing arrow** that will execute the **SimpleAction**.

Creating Cheat Sheets in Eclipse 3.0

Carlos Valcarcel, Director

Level: **Intermediate**

Summary

Cheat Sheets are a technology that made its first introduction in the IBM-branded version of Eclipse, WebSphere Application Developer. Cheat Sheets allow you to explicitly walk the user through an explanation of a task as well as execute specific steps in that task. This tutorial will discuss the creation of a Cheat Sheet, how to define steps that are pure explanation and steps that are attached to Eclipse actions. Cheat Sheet variables will not be discussed.

Required Software

Eclipse 3.0.x
JDK 1.4.x

Creating Cheat Sheets in Eclipse 3.0

An Eclipse Cheat Sheet is just another plug-in. If you have never implemented a plug-in then I would highly recommend some of the books listed at the end of this tutorial. A plug-ins is not difficult to implement, but is different than the average application since it works within the Eclipse framework so is not considered a standalone application.

The following Cheat Sheet will not focus on the explanation of a particular task. There are certain steps you must execute to activate Cheat Sheet functionality. We will look at:

- Defining a Cheat Sheet plug-in
- Defining a Cheat Sheet category
- Adding an introduction and a single element to the Cheat Sheet
- Adding an additional element with an Eclipse action to the Cheat Sheet
- Adding a Cheat Sheet listener to the plug-in definition and implementing a class that receives Cheat Sheet life cycle events. This particular functionality was not demonstrated in the 3-page tutorial.

Defining a Cheat Sheet plug-in

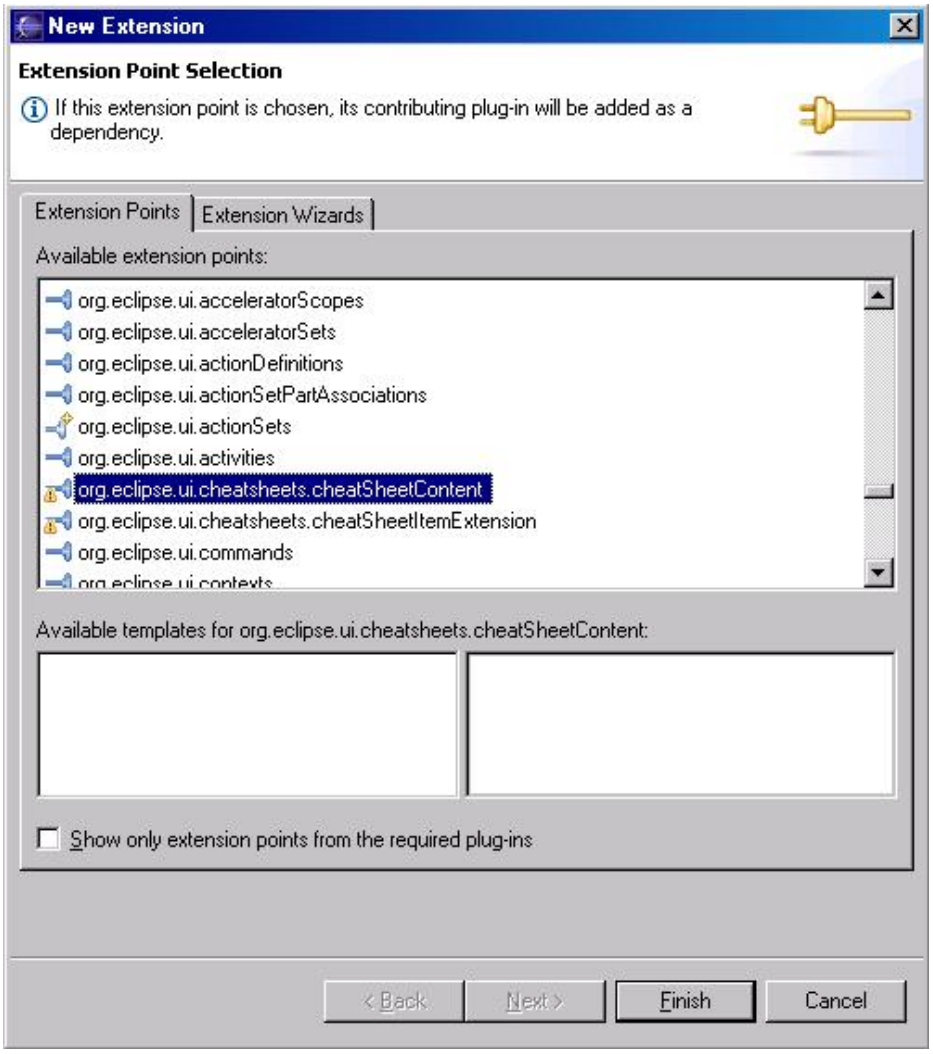
Start Eclipse. Since the Cheat Sheet is a plug-in you need to create a **Plug-in Project** where the plug-in will be implemented. Press **Ctrl+N** to open the **New** dialog. When the **New** dialog opens select **Plug-in Project** from the top batch of wizards and click **Next**. In the **Plug-in Project** page enter a **Project Name** of **com.triveratech.cheatsheet**. If you want to save the project somewhere other than the default directory then uncheck **Use Default** and enter, or browse, to a new directory. Click **Next**.

training@triveratech.com

The **Plug-in Content** page can be left alone. You can change the Plug-in ID, version, name, provider, jar file name or plug-in class name for your next Cheat Sheet. Click **Finish**.

When the plug-in completes its initialization the **Plug-in Manifest Editor** will open **plugin.xml** for **com.triveratech.cheatsheet**, which is the label that will appear in the editor tab. The next thing you need to do is list which extension point you are going to extend and any sub-nodes that need to be defined to based on the extension point. Click the **Extensions** tab.

Click **Add**. When the **New Extension** dialog opens *uncheck* **Show Only Extension Points from the Required Plug-ins**. From the **Available Extension Points** list scroll down and select **org.eclipse.ui.cheatsheets.cheatSheetContent**. Click **Finish**.



Once the **New Extension** dialog closes the **All Extensions** window in the **Plug-in Manifest** editor will display the **org.eclipse.ui.cheatsheets.cheatSheetContent** node in

the list to the left. If the extension is not selected in the **All Extensions** list to the left then select it. In the **Extensions Details** area to the right enter:

Id: com.triveratech.cheatsheet.example
Name: TriveraTech Example Cheat Sheet

Save the file by pressing **Ctrl+S**. The plug-in is now ready for accept a Cheat Sheet category and Cheat Sheet.

Defining a Cheat Sheet category

Cheat Sheets cannot exist in isolation. They will be placed into a category named **Other** or they can go into your own custom category. For this tutorial you will create a brand new category for the Cheat Sheet named **A Trivera Technologies Cheat Sheet**.

In the Manifest Editor, return to the **Extensions** tab. Right-click on the **org.eclipse.ui.cheatsheets.cheatSheetContent** node in the **All Extensions** list to the left to open the popup menu. Select **New->Category**. Enter in the **Extensions Element Detail** area to the right:

Id: com.triveratech.cheatsheet.category1
Name: A Trivera Technologies Cheat Sheet

The **Id** field should already contain the above information. Leave **parentCategory** blank.

Once again, right-click on **org.eclipse.ui.cheatsheets.cheatSheetContent** in the **All Extensions** list to the left to open the popup menu. Select **New->Cheatsheet**. In the **Extensions Element Detail** area to the right enter:

Id: com.triveratech.cheatsheet.cheatsheet1
Name: My First Cheat Sheet
Category: com.triveratech.cheatsheet.category1
contentFile: cheatsheets/MyFirstCheatSheet.xml

Save the **plugin.xml** file by pressing **Ctrl+S**.

At this point you can check your work by going to the main menu and select **Run->Run As->Runtime Workbench**. From the **Runtime Workbench** select **Help->Cheat Sheets**. The **Cheat Sheet Selection** dialog should list your new Cheat Sheet as the first category (Figure 1).

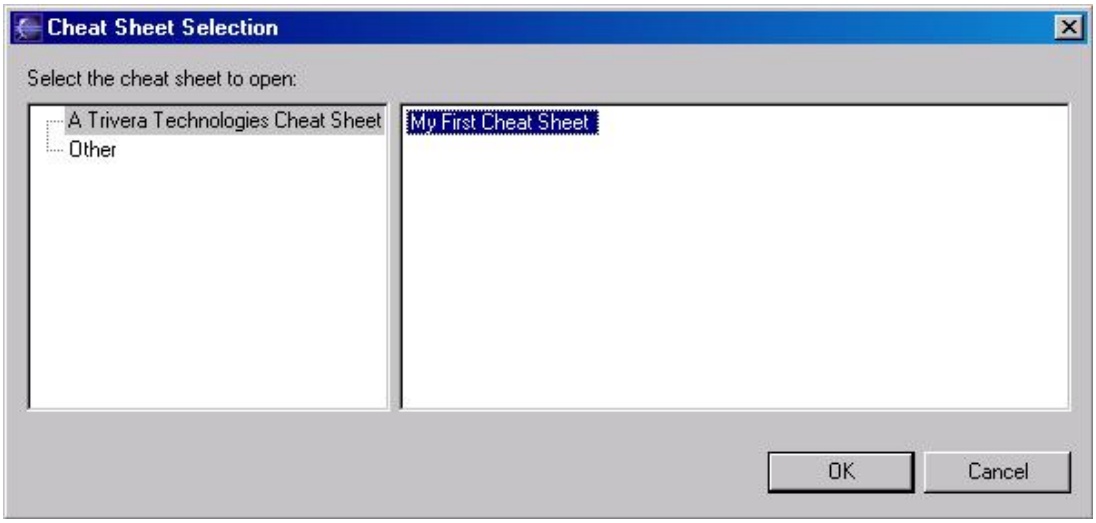


Figure 1 - The Cheat Sheet Selection dialog displaying the new Cheat Sheet.

Close the dialog and the Runtime Workbench.

Adding an introduction and a single element to the Cheat Sheet

Cheat Sheets have a minimum required part count: they must have a declared introduction, even if it is blank, and a single cheat sheet item. Both are declared in the XML file that contains the cheat sheet content. The next steps will define an introduction and an item.

In the **Package Explorer** view create a folder under the **com.triveratech.cheatsheet** project named **cheatsheets**. Press **Ctrl+N** to open the **New** dialog, select **Simple->Folder** and click **Next**. Select the project and in the **Folder Name** field enter **cheatsheets**. Click **Finish**.

In the **Package Explorer** view select the new **cheatsheets** folder and create a file named **MyFirstCheatSheet.xml** after pressing **Ctrl+N** and selecting **Simple->File**. Click **Next** and enter **MyFirstCheatSheet.xml** in the **File Name** field. Click **Finish**.

Enter the following into the file **MyFirstCheatSheet.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>
<cheatsheet title="My First Cheat Sheet">
  <intro>
    <description>
      This is the introduction to your first cheat sheet.
    </description>
  </intro>
  <item title="Step 1">
    <description>
      This is the description of how to execute the first step.
    </description>
  </item>
</cheatsheet>
```

The above XML is the minimum needed for a Cheat Sheet to display itself: a **cheatsheet** element containing an **intro** element and an **item** element. Run the **Runtime Workbench** (**Run->Run As->Runtime Workbench**) and select **Help->Cheat Sheets** from the **Runtime Workbench**. Click **OK** to see the Cheat Sheet open in the view to the right (Figure 2). If your get any kind of error message about the Cheat Sheet the easiest way to reset Eclipse's configuration is to delete the **.metadata** directory under the **runtime-workbench-workspace** directory under the **same directory** where your **workspace** is located. If you need to delete the **.metadata** directory first exit the Runtime Workbench, delete the directory and then restart the **Runtime Workbench**.

Figure 2 - The Cheat Sheet view with the Introduction and its only step.

Open **Step 1** to see the text from the **<description>** element. Exit the **Runtime Workbench**. As long as your task can be explained with individual steps and no direct interaction with Eclipse is needed the above steps are all you will need. If you need to interact with Eclipse, for example opening a perspective, then you need to declare the use of a Cheat Sheet action. Let's look at adding a Cheat Sheet action next.

Adding an additional element with an Eclipse action to the Cheat Sheet

Cheat Sheet actions are only slightly different than Eclipse actions. All you need to do is declare a class that inherits from **Action** and implements **ICheatSheetAction,** and then declare the class for use in **MyFirstCheatSheet.xml**.

Start by selecting the **src** directory of the **com.triveratech.cheatsheet** project from the **Package Explorer** view. Press **Ctrl+N**, select **Class** and click **Next**. In the **Java Class** page enter:

Package: com.triveratech.cheatsheet.action
Name: SimpleAction

The parent class must be changed and an interface added. Click **Browse** next to the **Superclass** field. When the **Superclass Selection** dialog opens enter and select **Action** from the **org.eclipse.jface.action** package and click **OK**. The **org.eclipse.jface.action.Action** class is now the **Superclass** for this action.

Next, Click **Add** next to **Interfaces**. When the **Implemented Interfaces Selection** dialog opens enter and select **ICheatSheetAction** and click **OK**. The Java Class page should look like Figure 3.

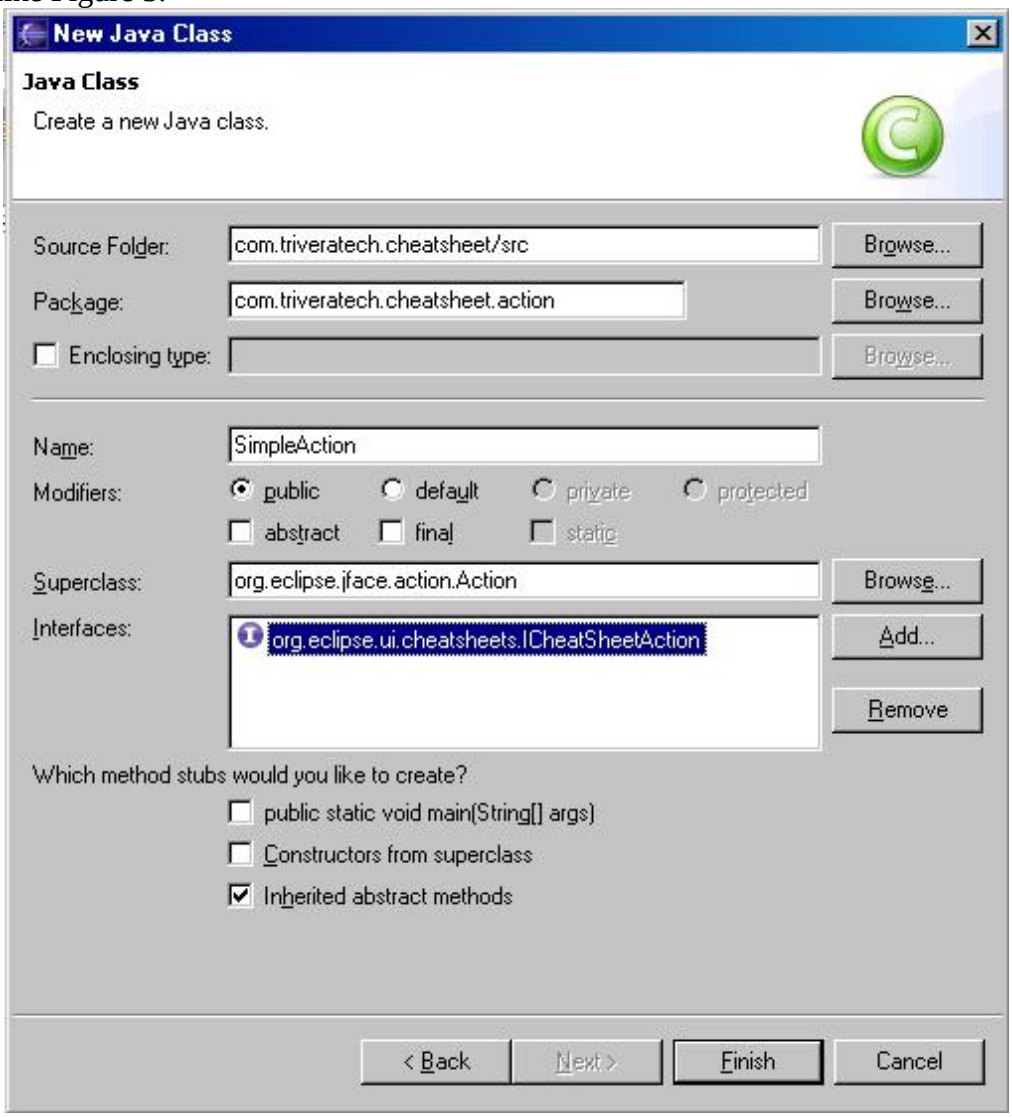


Figure 3 - The New Java Class dialog displaying the Cheat sheet action information.

Click **Finish** to close the **New Java Class** dialog and open the Java editor on the new class.

In **SimpleAction** add the code listed below in **bold** to the **run()** method which is declared in **ICheatSheetAction** and must be implemented by this class.

```
public void run(String[] params, ICheatSheetManager manager) {
  MessageDialog.openInformation(PlatformUI.getWorkbench()
    .getActiveWorkbenchWindow().getShell(),
    "My First Cheat Sheet Action",
    "The incoming parameter is " + params[0]);
}
```

If you encounter any compile errors it will be because the **MessageDialog** and **PlatformUI** classes are not defined. Press **Ctrl+Shift+O** to automatically enter the appropriate imports.

Return to the **MyFirstCheatSheet.xml** file. Enter the following after the last item, but before the closing **cheatsheet** tag:

```
<item title="Step 2">
  <action pluginId="com.triveratech.cheatsheet"
     class="com.triveratech.cheatsheet.action.SimpleAction"
     param1="This is my action parameter." />
  <description>
    This is a step that includes a Cheat Sheet Action.
  </description>
</item>
```

The entire file should contain (the new section is in **bold**):

```
<?xml version="1.0" encoding="UTF-8"?>
<cheatsheet title="My First Cheat Sheet">
    <intro>
        <description>
            This is the introduction to your first cheat sheet.
         </description>
    </intro>
    <item title="Step 1">
        <description>
            This is the description of how
            to execute the first step.
        </description>
    </item>

    <item title="Step 2">
        <action pluginId="com.triveratech.cheatsheet"
                class="com.triveratech.cheatsheet.action.SimpleAction"
                param1="This is my action parameter." />
        <description>
            This is a step that includes a Cheat Sheet Action.
        </description>
    </item>

</cheatsheet>
```

Restart the **Runtime Workbench** and select your cheat sheet from **Help->Cheat Sheets**. The **Introduction** step has a **right-pointing arrow** (Figure 2). Click the arrow to go from the **Introduction** to **Step 1**.

Step 1 has a **check** (Figure 4) so that you can let Eclipse know when you **complete** that step. Notice also the **blue** check next to **Introduction**. This is just another visible sign that you completed that particular item. Click the arrow to go to the next step.

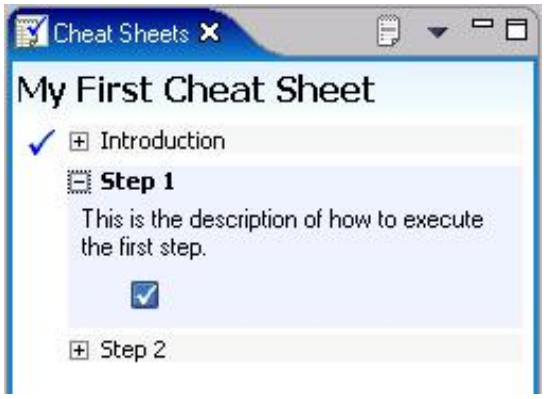


Figure 4 - **Step 1**'s check icon should be clicked upon completion of the step.

Step 2 has a **right-pointing arrow** that will execute the **SimpleAction** (Figure 5). Click the right-pointing arrow to execute **run()** which will open the **MessageDialog** (Figure 6).

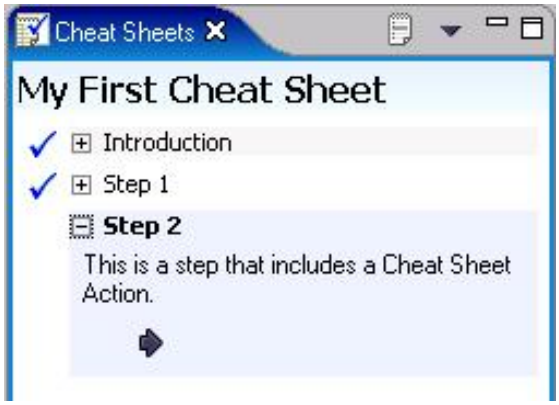


Figure 5 - **Step 2** with its right-arrow icon.

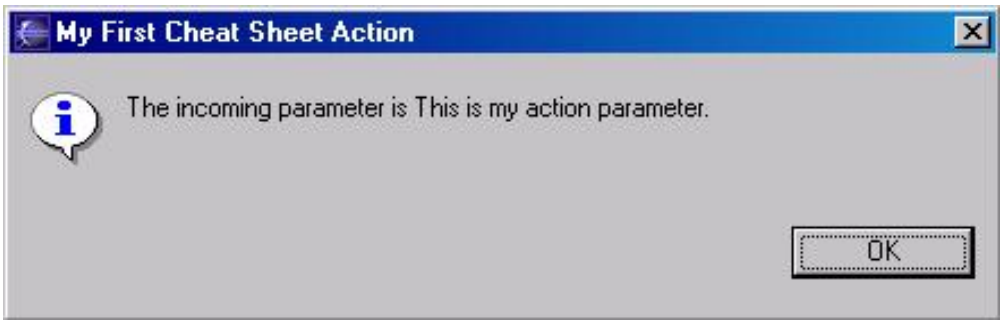


Figure 6 - The fruits of your **SimpleAction**.

When you click **OK** to close the dialog the Cheat sheet will close **Step 2** and return to the **Introduction**.

Adding a Cheat Sheet listener

Cheat Sheet listeners are Java classes that you declare in the plug-in manifest file **plugin.xml**.

Adding a **Cheat Sheet listener** is quite trivial. The real question should be: When would you want to use one? Because the nature of Cheat Sheets is dependent on input from the user of the Cheat Sheet a listener just affords an extra level of granularity to make sure the user doesn't make a misstep. Let's build one and observe its output.

Open the New dialog and select to create a new Java class. In the **Java Class** page enter a package name of **com.triveratech.cheatsheet.listener** and a class name of **CheatSheetListener**. Click **Browse** next to the **Superclass** field and select **CheatSheetListener**. Click **Finish** to close the **New** dialog and create the class.

When the **CheatSheetListener** opens in the Java editor add the following instance **String** array field and add the following code to **cheatSheetEvent()**:

```
private String [] TYPE = {"Opened", "Closed", "Started",
                          "Restarted", "Completed", "Restored"};

public void cheatSheetEvent(ICheatSheetEvent event) {
    int eType = event.getEventType();

    System.out.println("CheatSheet Event Type: " + TYPE[eType]);
}
```

In the **Package Explorer** view double-click on **plugin.xml** to open the **Plug-in Manifest Editor** if it is not already open. Click the **Extensions** tab and open the **org.eclipse.ui.cheatsheet.cheatSheetContent** node to the left. Select the **My First Cheat Sheet** node.

In the **Extension Element Details** section to the right enter in the **listener** field **com.triveratech.cheatsheet.listener.CheatSheetListener**, the name of the class you just implemented. Save the file.

Restart the **Runtime Workbench**. When it opens look at the **Console** view of the **development workbench** (not the **Runtime Workbench**) to see the output from the listener. The Cheat Sheet listener output should look like:

```
CheatSheet Event Type: Opened
CheatSheet Event Type: Restored
CheatSheet Event Type: Completed
```

Open the Cheat Sheet and **click on the various arrows and checkboxes**. When you click the first arrow the Cheat Sheet the output will append:

```
CheatSheet Event Type: Restarted
```

When you click on the **arrow of the last step** the output will include:

```
CheatSheet Event Type: Completed
```

Close the Cheat Sheet view in the **Runtime Workbench**. The full output from the development workbench should read:

```
CheatSheet Event Type: Opened
CheatSheet Event Type: Restored
CheatSheet Event Type: Completed

CheatSheet Event Type: Restarted


CheatSheet Event Type: Completed

CheatSheet Event Type: Closed
```

The number of blank lines between the various messages may vary.

Conclusion

In this tutorial we covered:

- The creation of the most basic Cheat Sheet.
- The addition of an Eclipse Action to the Cheat Sheet.
- The addition of a Cheat Sheet listener.

Existing Cheat Sheets are a great place to search for existing functionality to allow you to control the behavior of Eclipse through it s API. The opening of perspectives, editors, and view are just a few of the many things you can find in the existing Eclipse API.

Resources

The **Eclipse Help** pages have an entry on contributing Cheat Sheets at:

Platform Plug-in Developer Guide->Programmer's Guide->Advanced Workbench Concepts->Guiding the User Through Tasks->Cheat Sheets.

Books

Eclipse: Building Commercial-Quality Plug-ins (Eclipse Series) by Eric Clayberg, Dan Rubel.

Publisher: Addison-Wesley Professional
ISBN: 0321228472

Java(TM) Developer's Guide to Eclipse, The (2nd Edition) by Jim D'Anjou, Scott Fairbrother, Dan Kehn, John Kellerman, Pat McCarthy

Publisher: Addison-Wesley Professional; 2 edition
ISBN: 0321305027

Eclipse Kick Start by Carlos Valcarcel

Publisher: Sams
ISBN: 0672326108

About the Author

Carlos Valcarcel is the Director of Technology at Trivera Technologies LLC and has been involved in Java and distributed systems since 1996. He has done numerous presentations at domestic and international conventions. In addition to his work as an architect, developer and mentor, he has also presented Java courses for various companies including Sun and IBM. He can be reached at carlos@triveratech.com.