RESEARCH REVIEW JOURNALS
Volume-04
Issue-04
April -2019

ISSN: 2455-3085 (Online)
RESEARCH REVIEW International Journal of Multidisciplinary
www.rrjournals.com [UGC Listed Journal]

A Study of Java Application Design and Development of Rich Client Application

[1]G. Shabaz Mohsin & [2]Dr. Harsh Kumar

[1]*Ph.D Research Scholar, Dept. Of. Computer Science, Himalayan Garhwal University, Uttarakhand*
[2]*Associate Professor,Dept. Of. Computer Science, Himalayan Garhwal University, Uttarakhand*

ARTICLE DETAILS
Article History
Published Online: 15 April 2019

Keywords
Development, Java Application, Rich Client Application.

ABSTRACT
This tutorial teaches the creation of rich client applications based on the Eclipse Rich Client Platform (RCP). Soon after its inception, Eclipse was used for building applications outside the tools domain for which Eclipse was originally designed. The Eclipse development team embraced this trend and with Eclipse 3.0 introduced the RCP that facilitates the creation of rich client applications. In this tutorial we describe the overall architecture of an RCP-based application, its specific components, development, packaging, deployment and testing. Participants will learn and perform the steps to build their own RCP application. We will show how to develop a minimal application plug-in, add a feature including custom branding, and package the application for deployment.

1. Introduction

Today, in the time of World Wide Web, schedules should have been redone and refreshed for their expanding significance and use in our day by day life. One should be alarmed for an occasion for example meeting, arrangement or some other significant date. Correspondence through email turns out to be so normal in this age, it is important to make a schedule application, by which one can be alarmed through email. This proposition work mirrors a solid encounter of Java programming language, and learning of associating the applications with database.

The advancement methodology for the application required the accompanying advances:

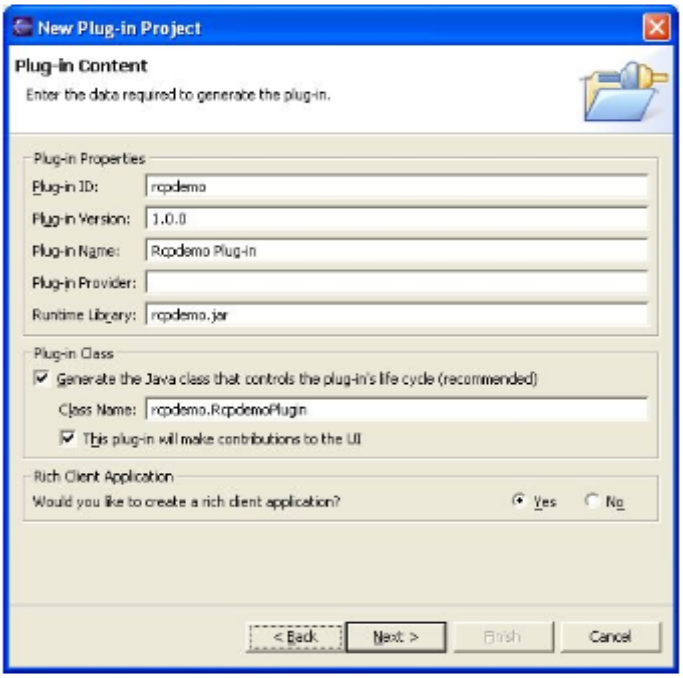
- Layout of the UI
- Coding for the application
- Creation of database for the application
- Connecting the database with application
- Addition of email usefulness in the application

Java is a processing stage and abnormal state programming language, which was grown initially by James Gosling at Sun Microsystems. A group at Sun Microsystems named "green group" began taking a shot at Java language venture in 1991, which was at last discharged in 1995. Java has its linguistic structure like and got from C, C++ dialects. Except if different dialects, in which one needs to either arrange or decipher the code before use it, in Java we need to accumulate and translate code to run it.

Generate the Most Basic RCP Application

Obscuration accompanies a wizard to make a rich customer application. This gives you a skeleton for all your own improvement. This wizard fundamentally plays out the manual advances that Ed Burnette depicts in his RCP tuorial [EclipsePowered]. To make our life more straightforward, we utilize the wizard.

• Create another module venture
• Project name: rcpdemo
• Next
• "Might you want to make a rich customer application?" > Check "Yes"

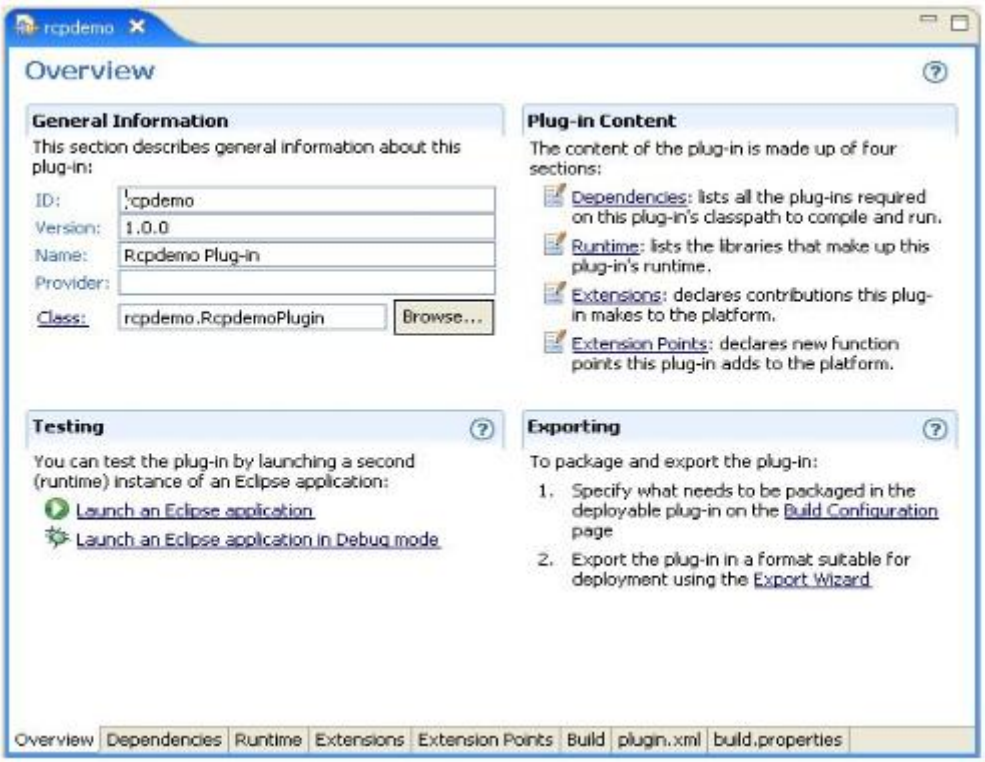


• On the Templates page, check the box and select the “Hello RCP” template.

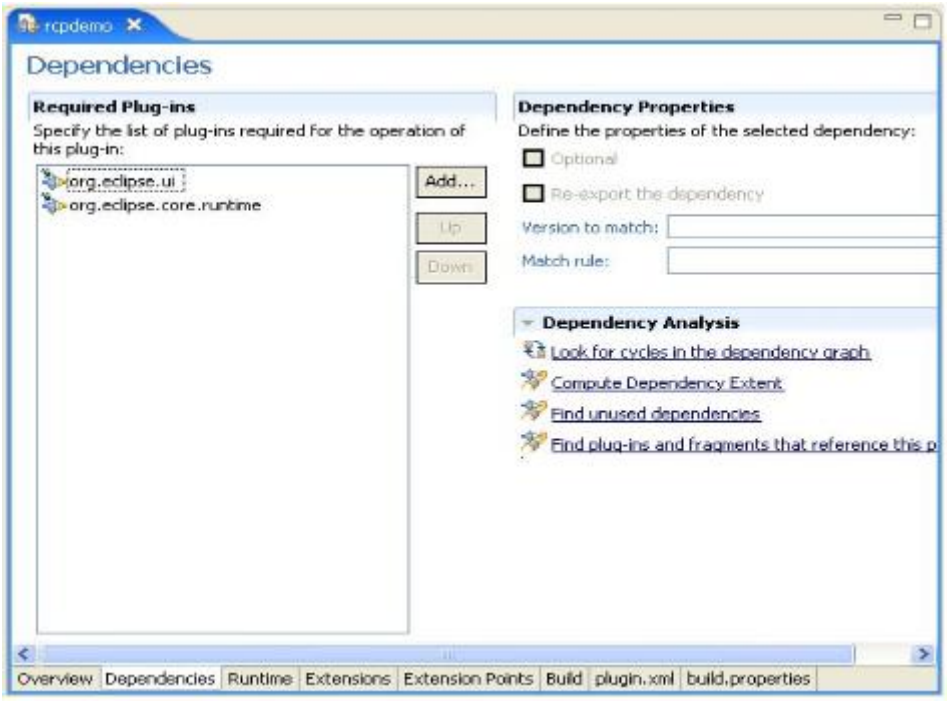
• Click “Finish”

This creates a project that contains the most basic elements of a rich client application. The wizard opens the Plug-in Manifest editor. Let us quickly go through the different tabs.

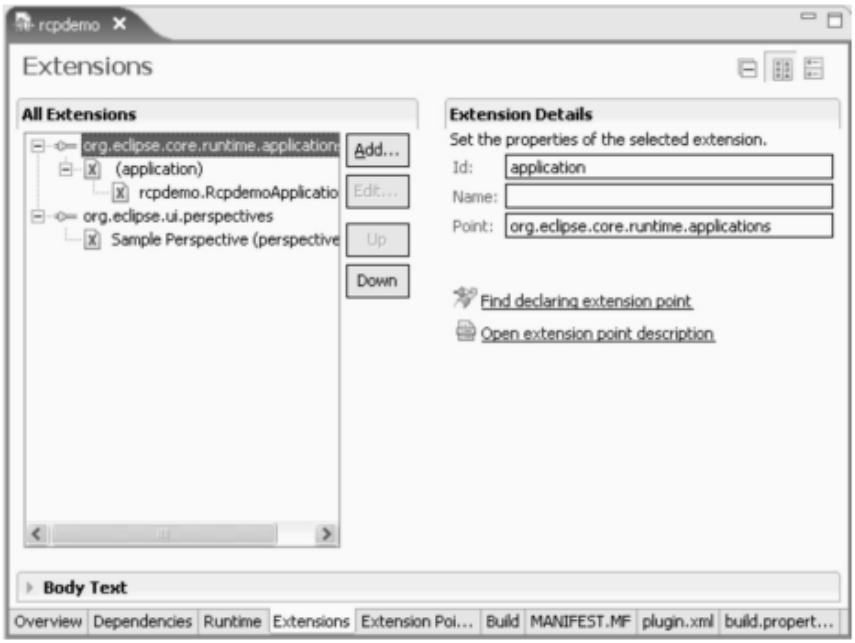
The “Overview” tab shows the plug-in ID and the name of its plug-in class.



The "Conditions" tab just demonstrates the immediate conditions. It doesn't indicate aberrant conditions. For running a rich customer application, it needs to have all immediate and backhanded ward modules accessible at runtime. This degree is significant for application arrangement. We will see later, how the dispatch design wizard can process the entire degree of required modules dependent on this reliance data here.



In the “Extensions” tab we see, that our plug-in defines a new application. It also defines a new perspective. We will revisit the implementing classes in a second.



The new project wizard also created all the necessary source code for the rcpdemo application. Before we review this source code, we want to launch the application.

Starting an RCP Application

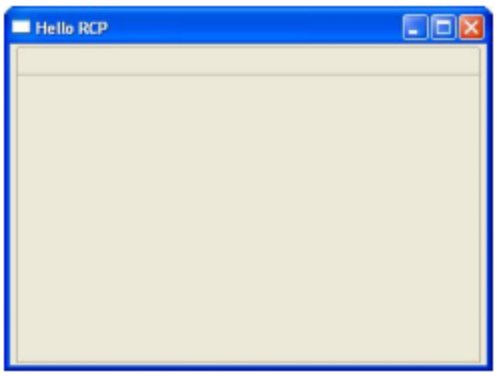
A traditional commitment to Eclipse comprises of a lot of perspectives, editors, points of view, and so on that expand the current workbench. A rich customer application, then again, replaces the outstanding Eclipse workbench. If there should arise an occurrence of rcpdemo it doesn't characterize a view that could be begun inside the traditional workbench.

In this manner, to begin a rich customer application, we have to characterize the arrangement of modules and point to the application that replaces the old style workbench. This is done in the dispatch arrangement. The module show manager gives a helpful method to make the suitable dispatch arrangement (see the screen capture above).

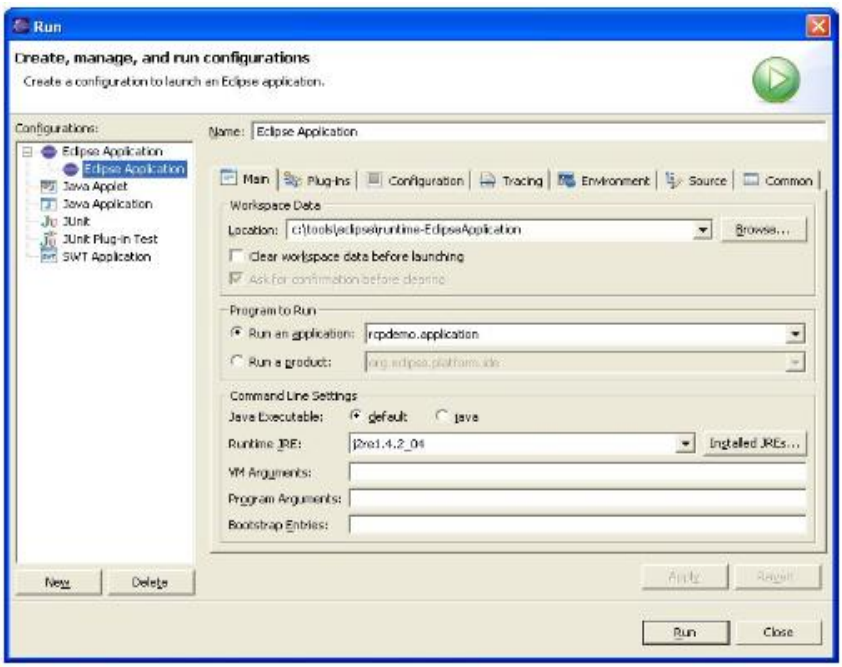
• Switch back to the "Outline" tab.

• Click on "Dispatch an Eclipse application"

Presently you should see the most essential workbench window.

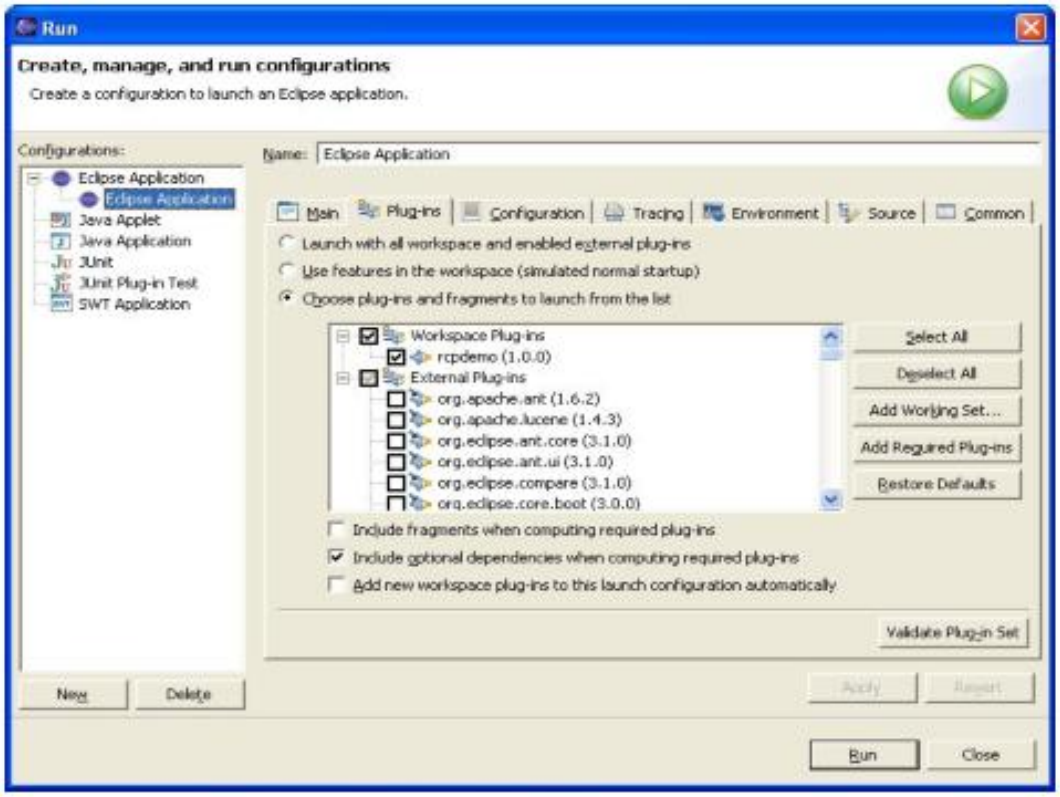


Now review the launch configuration (Run>Run...>Eclipse Application)



The “main” tab specifies a location for workspace data. As “Program to Run”, the rcpdemo. Application is selected. We will discuss later, how to run the rich client application as a product.

The “Plug-ins” tab lists all the necessary plug-ins. Based on the list of direct dependencies as listed in the plug-in manifest, the launch configuration lists all direct and indirect dependencies.



The “configuration” tab specifies the location of configuration information. Review the “config.ini” file that is found in that directory.

Tip: open this file inside Eclipse using File > Open external file... and navigate to “\.metadata\.plugins\org.eclipse.pde.core\Eclipse Application\config.ini”.

The file contains a list of all plug-ins that are listed in the “plug-ins” tab and is generated from that table. The OSGi runtime reads the static list of plug-ins from this config file.

Basic Elements of an RCP Application

Let's review the source code of the rcpdemo application. It consists of the classes

1. **Rcp demo Plugin:** Since each rich customer application is an Eclipse module, it needs a module class. The created execution gives access to the module occasion and loads the asset group.
2. **Application:** This is the bootstrap class for the rich customer application. This class makes and runs the workbench that contains the headliner circle.
3. **Viewpoint:** To mastermind UI components inside the workbench, you in fact need a point of view. Until further notice, it doesn't contain anything intriguing, since the rcpdemo application doesn't contain any view or editorial manager.
4. **Application Action Bar Advisor:** The activity bar consultant gives snares to filling the diverse activity bars like the menu bar and the cool bar. It likewise arranges the status line and makes the activities.
5. **Application Workbench Advisor:** The workbench guide gives various snares around the lifecycle of the workbench itself. One of the most significant occupations is to give the Application Workbench Window Advisor.
6. **Application Workbench Window Advisor:** The workbench window guide gives various snares around the lifecycle of the workbench window. For instance it permits to set the window size before it is open-end.

RCP Application as a Product – rcpmail

Subsequent to acing the initial steps with the most straightforward rich customer application, we need to investigate a progressively complete model.

• In the PDE point of view, make another module venture

• Name: rcpmail

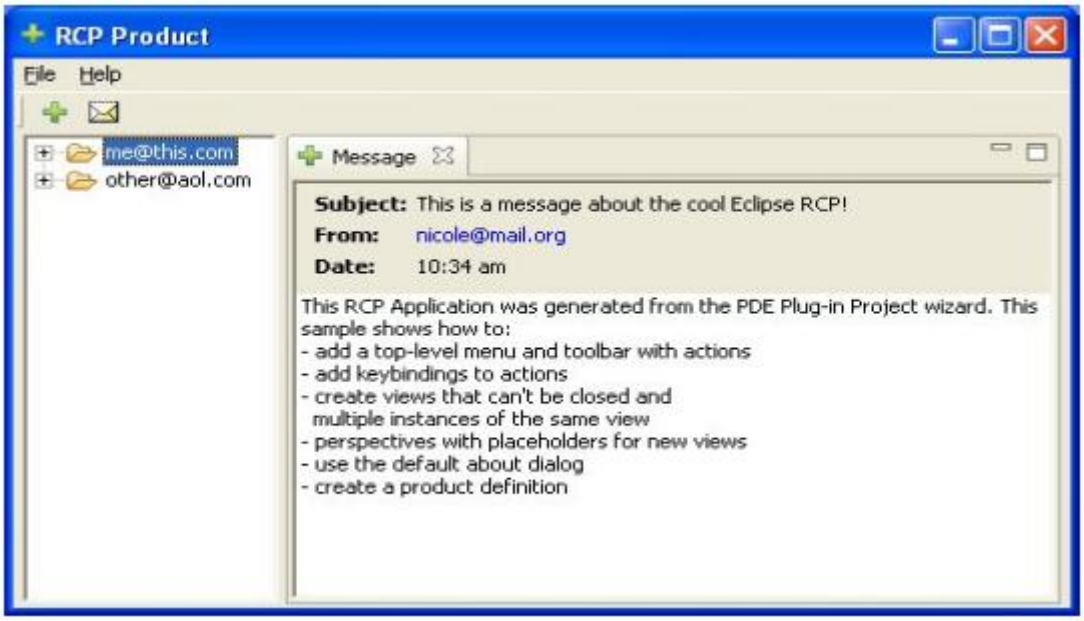
• Next

• "Might you want to make a rich customer application?" > Check "Yes"

• Create a module utilizing the "RCP Mail Template". You should peruse the short depiction of the layout.

• Finish

• Start the new rcpmail application and watch the new includes.



Add Update-Manager to rcpmail

The update chief disseminates refreshes for an Eclipse-based application. For the RCP, this is viewed as a discretionary element. We need to add it to the rcpmail application. The Update Manager requires, that we bundle the modules of our rich customer application as an Eclipse highlight. By show, include venture names end in "...-highlight". Since we make a component for our rcpmail application, we give it the name "rcpmail-highlight). The element ID, in any case, must be "rcpmail" rather than the proposal of the element creation wizard. The method of reasoning behind this is, each element requires a marking module. As a matter of course, the ID of the marking module is thought to be equivalent to the element ID. In the element show you could determine an alternate marking module ID. For our situation, the current rcpmail module is the marking module.

1. Create Product Features

First, create the branding plug-in for the new feature that contain the Eclipse base plug-ins.

- Create a new plug-in project
- Name: rcpmailbase
- Uncheck “Create a Java project”
- Press finish

This branding plug-in is required for the following feature.

- Create an new feature project
- Name: rcpmailbase-feature
- Feature ID: rcpmailbase
- Accept the remaining defaults
- In the list of plug-ins check all plug-ins that you find in the launch configuration except “rcpmail” and “rcphelp”.
- Also add the rcpmailbase plug-in to the list. (While you are at it, you might want to add it to the launch configuration for future reference.)

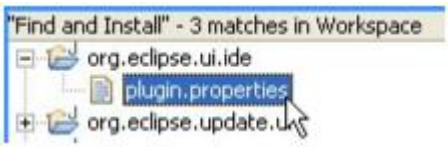
Now we can create the feature that contains our own plug-ins.

- Create an new feature project
- Name: rcpmail-feature
- Feature ID: rcpmail
- Accept the remaining defaults
- In the list of plug-ins check “rcpmail”, and “rcphelp”.
- Press finished
- In the manifest editor on the “Included features” tab add the rcpmailbase feature.
- On the “Dependencies” tab, clear the list.

2. Add Update UI Action

Like for the help system, we have to add the update manager dialog to the menu and add some plug-ins to the launch configuration. Let's see, how the Eclipse SDK adds the menu entry (Help > Software Updates > Find and Install) to the workbench.

- Open the file search dialog
- Containing text: “Find and Install”
- File name pattern: “plugin.properties”
- Scope: Workspace (we assume that you have imported all Eclipse base plugins into the workspace).



- Open the plugin.properties of org.eclipse.ui.ide
- This text is defined for the variable UpdateActionSet.updates.label
- Open the manifest of that plug-in and look for this label.
- You find an action set definition that adds the menu entries.
- Copy this definition and adapt it to the rcpmail application.


```
<extension
      point="org.eclipse.ui.actionSets">
   <actionSet
         label="%UpdateActionSet.label"
         visible="true"
         id="rcpmail.softwareUpdates">
      <menu
            label="%UpdateActionSet.menu.label"
            id="rcpmail.updateMenu">
      </menu>
      <action
            label="%UpdateActionSet.updates.label"
            icon="icons/usearch_obj.gif"
            class="rcpmail.rcp.InstallWizardAction"
            menubarPath="rpcmail.updateMenu"
            id="rcpmail.newUpdates">
      </action>
   </actionSet>
</extension>
```

The show of org.eclipse.ui.ide references the class org.eclipse.ui.internal.ide.update.InstallWizardAction to open the introduce wizard. Since this class lives in a module, that we would prefer not to rely upon, we have to make a neighborhood duplicate of it and reference it from the activity definition. You see this reference to the nearby duplicate in the piece above as of now. Note: What? Reorder programming? You may think, we have lost our psyches. However, hold up a moment. In the event that we need to summon the introduce wizard, we have to give an activity that carries out that responsibility. Such an activity exists in Eclipse, yet inside a module that we would prefer not to add to our necessities list (org.eclipse.ui.ide). Presently, in the event that you audit the duplicated class you will discover, that it contains 5 lines of code that really accomplish something. We adjusted this code duplication against the overhead of including the undesirable module dependecy and found that we made the best decision. We even can hope to get the favors of Erich Gamma and Kent Beck, since we pursued their "Monkey see, monkey do" house rule.

This replicated class references some code from org.eclipse.update.ui module.

• Add it to the conditions rundown of the rcpmail application – not to the venture assemble way. Spare the module show.

• Open the dispatch design and in the modules rundown press "Include required modules"

• Run the dispatch design and open the Install Wizard.

2. Conclusion

Overshadowing is a multi-lingual programming advancement condition, which comprises of coordinated improvement condition (IDE) and additional module framework. An IDE gives a wide range of instruments for example code composing, arranging, running, troubleshooting, record the executives, and documentation all at one single stage. The most clear approach to construct and fare and application from inside Eclipse is accessible straightforwardly in the UI and you have utilized those highlights all through this instructional exercise through the fare wizard. This advantageous usefulness is given by org.eclipse.pde.ui. For construct robotization, be that as it may, the utilization of this usefulness is debilitated. The fabricate forms in org.eclipse.pde.ui run nonconcurrently and in this manner can't be utilized in insect construct content, since it can't get together with other Eclipse forms. Obscuration is one of the IDEs which are generally utilized expertly to create applications and programming arrangements. A preferred position of Eclipse over other expert IDEs is that Eclipse is an open source stage, and subsequently, it is anything but difficult to include new libraries and assets in it. Its greater part is written in Java programming language.

References

1- Wikipedia, Java Latest specifications, Wikipedia the free encyclopedia. Aug, 2012 (online) Available:http://en.wikipedia.org/wiki/Java_version_history#Java_SE_7_.28Jul y_28.2C_2011.29 . [Accessed: 16Aug2012]

2-GUI Components, Paul Deitel & Harvey Deitel, Java How to program 9th Edition.ISBN:0132575663, Available:http://www.deitel.com/Books/Java/JavaHowtoProgram9eEarlyObject sVersion/tabid/3622/Default.aspx.

3- Wikipedia, Swing, Wikipedia the free encyclopedia, (online) Available: http://en.wikipedia.org/wiki/Swing_%28Java%29. [Accessed: 28July, 2012]

4-Oracle, The Java tutorials. Graphical User Interface, Swing. (Online) Available:http://docs.oracle.com/Javase/tutorial/ui/overview/intro.html [Accessed:28 July, 2012]

5-Wikipedia, MySQL, (online) Available: http://en.wikipedia.org/wiki/MySQL [accessed: 22 Aug, 2012]

6- Developers MySQL, Documentation. (Online) Available: http://dev.mysql.com/doc/refman/5.5/en/mysql-nutshell.html [Accessed: 22 Aug, 2012]

7- JDBC, what is JDBC. Roseindia.net. (Online) Available: http://www.roseindia.net/jdbc/what-is-jdbc.shtml [Accessed: 03 Sep 2012]

8- JDBC connection. Work of Java database connectivity. (Online) Available:http://aspen.ucs.indiana.edu/webtech/jdbc/overviewpaper/JDBCconn. html .[Accessed: 03 Sep 2012]

9- Mechanism of JDBC. (Online) Available: http://www.datadirect.com/resources/jdbc/faqs/how-does-jdbcwork.html [Accessed: 04 Sep 2012]

10- Wikipedia, Eclipse Software, Wikipedia the free encyclopedia (Online) Available: http://en.wikipedia.org/wiki/Eclipse_%28software%29 [Accessed: 11 Sep 2012]

11-Java development tools in eclipse, JDT. (Online) Available:http://www.eclipse.org/jdt/ [Accessed: 9 Sep 2012]

12- Developing Java projects in Eclipse. (Online) Available: http://www.vogella.com/articles/Eclipse/article.html.[Accessed: 9 Sep 2012]

