Metropolia

Mesfin Tegegne

Server Side Collaboration of eLearning Platform and Integrated Development Environment

Helsinki Metropolia University of Applied Sciences

Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Thesis	
Date	18 May 2011

Author(s) Title Number of Pages Date	Mesfin Tegegne Server side coolaburation of eLearning platform and integrated development environment 53 pages 18 May 2011
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	Advanced Application Development
Instructor(s)	Mika Lackman(CEO) , Project Manager Kari Aaltonen, Principal Lecturer (Advisor)

The purpose of this thesis project was to design a plug-in, which integrates a locally installed Eclipse IDE with a remote exercise management system. The client of this project was Viope Solutions OY, an online based eLearning service provider company located in Helsinki Finland. The user groups of the service are students and individuals.

This project was carried out using Eclipse plug-in Development Environment (PDE) and different programming technologies such as Java technology, Standard Widget Toolkit (SWT), and XML parsing.

In this project there are three plug-in solutions such as: login, chapter and exercise selector, and exercise sender. Each plug-in solution has been integrated with client's already existing system, and tested using dummy data files.

As a result, the requirements set for this project have been accomplished. Also additional features, such as enabling users to access exercises from Eclipse IDE main tool bar, were realized.

The plug-ins designed may be useful in facilitating the client's existing exercise handling process. However, only these solutions may not be enough to increase the productivity of the company. Therefore, it is recommended that additional plug-ins with multiple functionalities should be added in future.

Keywords	Eclipse, IDE, Eclipse Plug-in, eLearning

Table of Contents

Abstract

Terminologies and Abbreviations

1 Introduction 5

2 Integrated Development Environments 6

2.1 Examples of Integrated Development Environment 7

2.2 Eclipse IDEs and its Plug-in Based Architecture 9

2.3 Plug-in Overview 14

2.4 NetBeans IDE 16

2.5 Comparison Between Integrated Development Environments 23

2.6 Web Services as a Tool for eLearning 24

3 Client Application 26

3.1 The Existing eLearning System 26

3.2 Requirements and Specification 28

3.3 Overall Description 29

3.4 Plug-in Based Development Architecture 30

4 Development Methods and Technologies 37

4.1 Standard Widget Toolkit 37

4.2 Tree-Based XML Parsing 39

4.3 Java Technology 39

4.4 Other Supportive Libraries 40

5 Testing and Deploying Solutions 42

5.1 Testing Solution 42

5.2 Deploying to Final Destination 43

5.3 Licensing and Related Legal Issues 50

6 Conclusion 51

References 52

Appendix

Terminologies and Abbreviations

IDEs	Integrated Development Environments
RCP	Rich Client Platform
Eclipse platform	The core framework and services upon which all plug-in extensions are created. It also provides the runtime in which they are loaded, integrated, and executed. The primary purpose of the Platform subproject is to enable other tool developers to easily build and deliver integrated tools.
OSGi	OSGi Alliance is a worldwide consortium of technology innovators that advances a proven and mature process to create open specifications that enable the modular assembly of software built with Java technology.
Workbench	Desktop Development Environment
Stand-alone Application	An application that does not need to interact with any other application, which is working on its own independently from the existing application.
Monolithic Application	Single-tiered software application which designed without modularity, where user interface and data manipulation code is merged together into a single program.
EPL	Eclipse Public License is a legal agreement that governs the rights granted to material licensed under it.
CDDL	Common Development and Distribution License
SWT	Standard Widget Tool kit

1 Introduction

eLearning technology is one common way of providing online education to people irrespective of their physical presence. The innovation of eLearning technology is becoming more complex and competitive. An effective eLearning process must have easier interactivity and be easily understandable by different users group.

Integrating an eLearning system with common development tools, is an important development area to make the eLearning process more effective and dynamic.

Viope Solutions is an eLearning service provider which is based in Helsinki, Finland. Its main activity is to offer a dynamic eLearning solution for users. The company has played a great role in the past in introducing efficient eLearning materials over the internet for students as well as teachers to use in their daily teaching process. However, the company's online exercise management system has not been very user friendly and dynamic in providing efficient service to the learners. Specifically the code editor part has a number of limitations; therefore, the introduction of new exercise management system was necessary in order to reduce the current burden and to facilitate the overall eLearning process. Using the most common development environment and integrating with the company's eLearning system to tackle the current limitations, was the objective of this study.

The project was divided into two theses studies. The first study *"Client Side Collaboration eLearning Platform and Integrated Development Environment"* deals with the integration of the eLearning materials and user interfaces with the application. The present study is the second part of the project.

The goal of this project was to design an Eclipse plug-in that connects the Viope solution's exercise management system with the user's locally installed development environment. More specifically this study focuses on how a particular plug-in is designed, tested and deployed for its destination.

2 Integrated Development Environments

Integrated development tools are most commonly composed of:

- Code editor
- Compiler
- Debugger and User interface(GUI)

These tools may come up as a standalone application or as part of the existing application. They are designed to make the programmer's life easier and to make them more productive by providing comprehensive components during their development process. However, to work smoothly and to use the highest support from the IDE requires a lengthy learning process which requires skills.

Integrated development tools are designed for specific programming languages, though there are some multiple-language programs, such as Eclipse, recent versions of Net Beans, Microsoft Visual Studio etc. These applications and their advantages will be discussed in this study.

Integrated Development tools are designed as a collection of multifunctional programs which integrate all the pieces of development units or tools to be harmonically linked to each other. The most common benefits are their capacity of compiling, authoring, deploying and debugging all in one. They are also designed to reduce the amount of unnecessary pieces of configurations. Theoretically, this reduces the time to learn the programming language and improves the efficiency of the developers. [1]

2.1 Examples of Integrated Development Environment

Visual Studio

.NET Framework is a software framework that uses windows platform to run its applications. It is supporting plenty of programming languages. Developers can write code, build, and execute within the Framework. NET Framework is composed of code libraries which are called as a Base Class Library (BCL).Their main purpose is to solve common programming tasks such as: file manipulation, graphic rendering, xml processing, and database integration. NET framework is a combination of layers in which each program is executed sequentially. The runtime of the .NET Framework is called the Common Language Runtime (CLR) which is in charge of code compilation during runtime. Also it is responsible for managing memory and error handling. All programs developed in .NET languages are needed to be compiled primarily into an intermediate language called Common Intermediate Language (CIL). [2]

What can be done by using .Net Framework?

Visual studio IDE can serve as a tool for many .NET based applications, apart from supporting for designing, development, debugging, and deploy web applications. It is also important to develop applications such as presented below.

ASP.Net

ASP.NET is a web application platform which is a run on .NET development environment. It allows developers to design dynamic web applications, Web services, and web sites.

Microsoft Azure

This is a cloud computing or cloud service operating system which provides services such as hosting, service management for applications designed for cloud computing. Azure provides services for developers such as administration and scalability. When it comes to designing applications with azure, developers are only worrying about their business logic and the rest will be done by the framework itself.

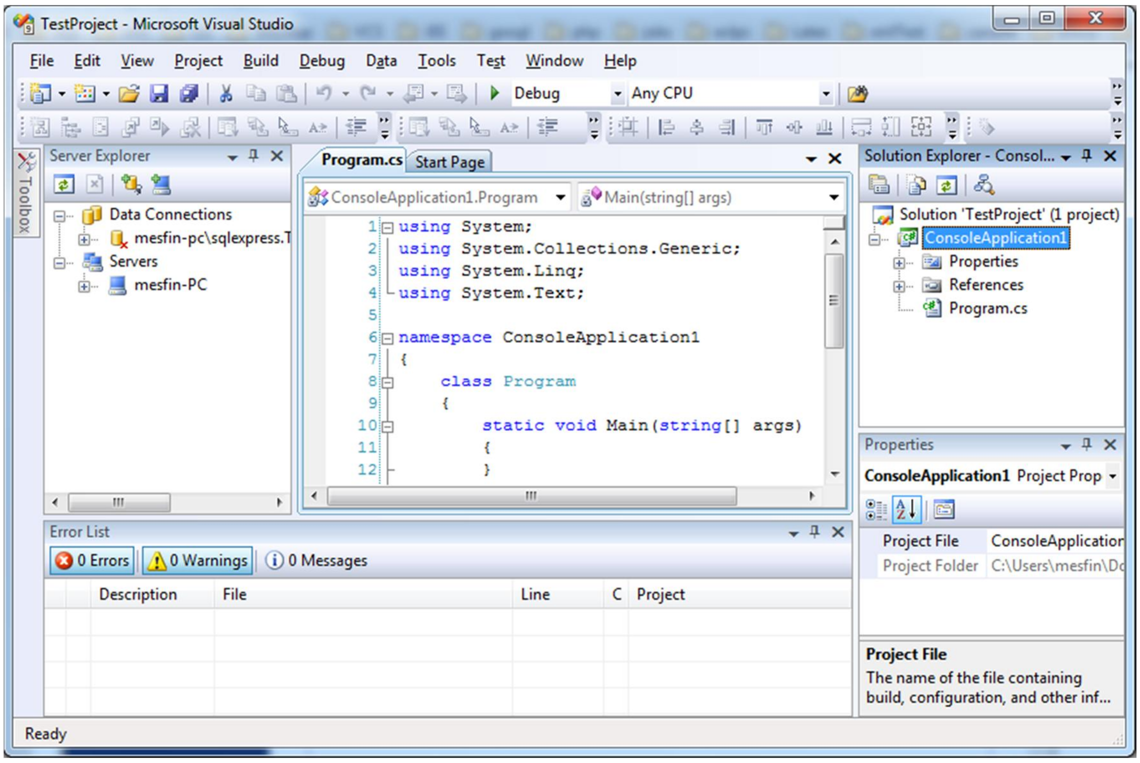
Microsoft SharePoint

This is a web content management and document management system. It is a multipurpose platform which allows managing intranet, extranet and documents. It is highly scalable and can support multiple organizations from a single Web server.

Microsoft Xna

Xna is a Microsoft game development framework and it includes a standard development tool kit (Xbox). After the release of XNA studio, the framework is becoming more robust and any potential developer can create different games. [3]

Figure 1 shows Microsoft Visual Studio 2008, which includes various application development tools, specifically it is designed to support Web and desktop applications development.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
```

Project File	ConsoleApplicatior
Project Folder	C:\Users\mesfin\Dc

Project File
The name of the file containing build, configuration, and other inf...

Figure 1. Visual Studio 2008

Visual Studio supports multiple programming languages which developers can choose according to their expertise. Figure 2 shows the three most common programming languages such as: Vb, C#, C++. In addition, Visual Studio has a feature of being customizable based on user preferences. It is also possible to change the settings of the layout or the behavior of the keyboards. Visual Studio comes with a built-in browser that helps the developer to browse the internet without launching any other application; it is also helpful in accessing online help files and source codes.

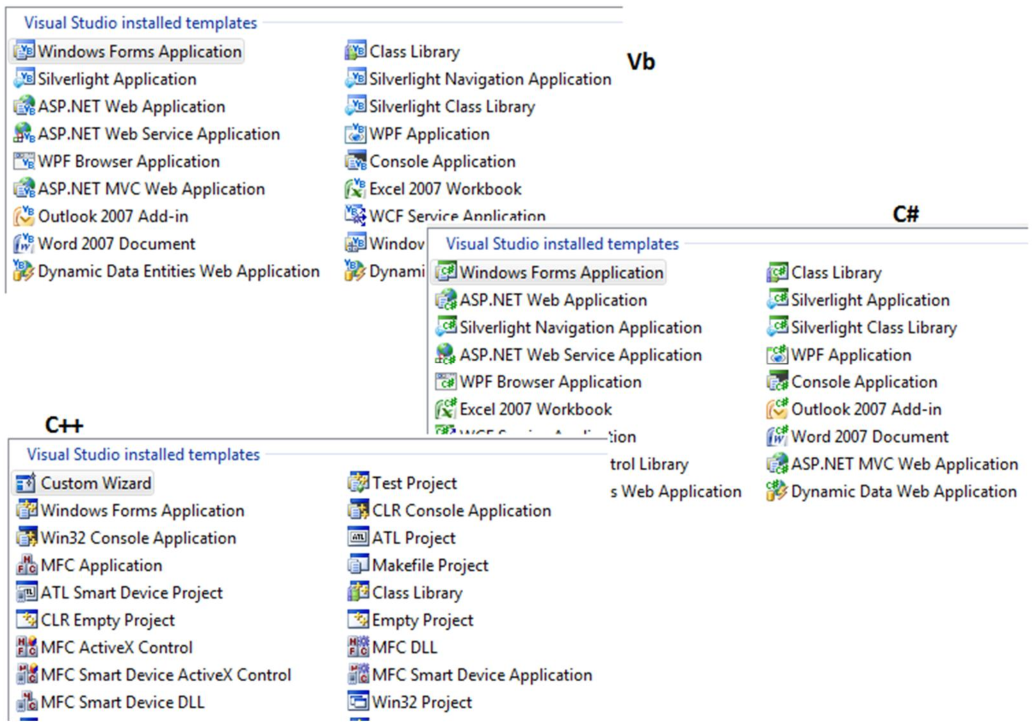


Figure 2. Languages supported by .NET framework

The significant difference of Visual Studio from other Java based development tools such as Eclipse and NetBeans is its monolithic application nature. Its architecture is mainly based on a single-tier application, in which the user interface and data access and are combined all together at once.

2.2 Eclipse IDEs and its Plug-in Based Architecture

The Infrastructure

This platform is a freeware universal development tool. It is supporting multiple languages and resources such as Java, Html, C, and JSP. It is a Java based development tool which is a collection of unique plug-ins produced by third-party developers.

Eclipse by itself does not offer anything special to end-users. What makes it so interesting is the availability of plug-ins to support additional functionalities and services. This pluggable architecture offers the end user a variety of tools.

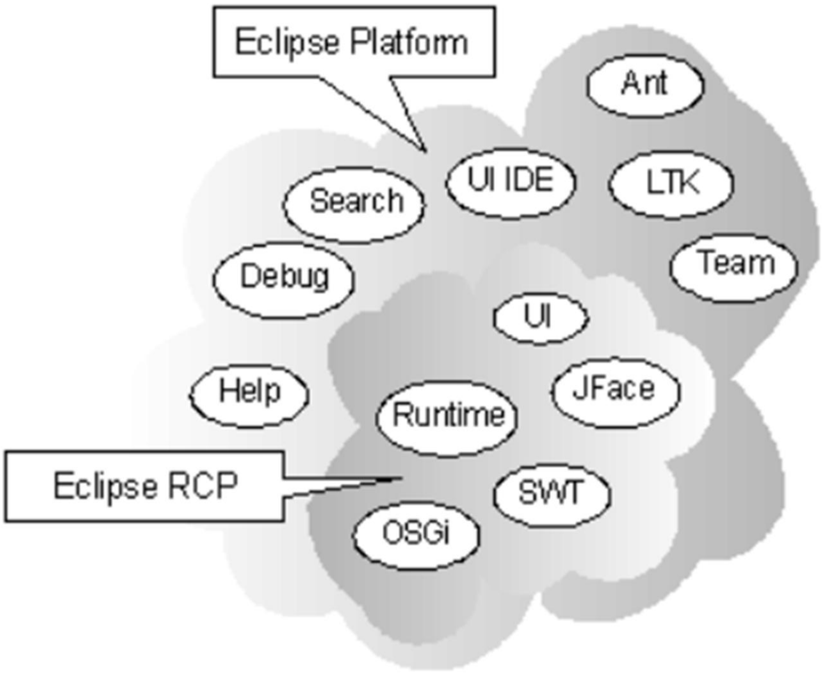


Figure 3. Platform is a composition of components [4]

Eclipse platform itself is composed of distinct components, which are integrated smoothly and form a solid platform. By using these components, it is possible to create any application.

The Rich Client Platform (RCP) is one of the sub components as shown in Figure 3, and it provides a platform for building a general purpose application. Rich Client Component Platform itself is constituted from the following elements:

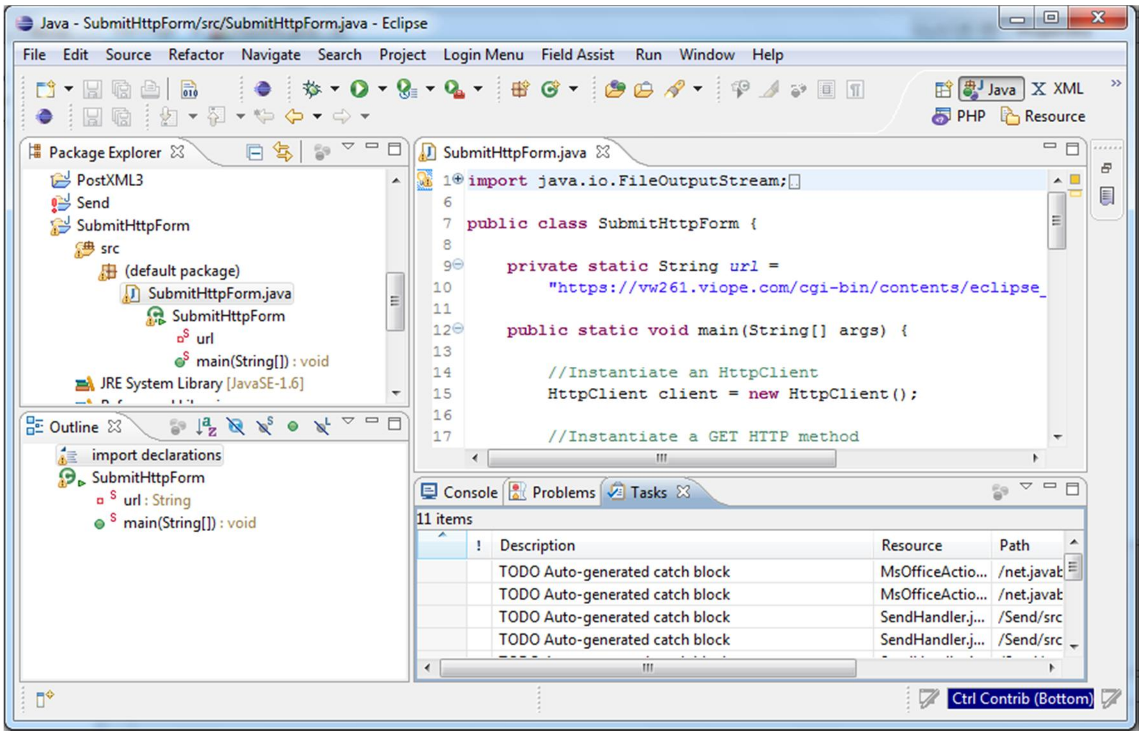
- Standard Widget Tool kit
- Core Platform
- JFace
- Workbench
- Equinox OSGi

The platform is capable of switching into any language based on the users need. For example, a developer can easily turn to C/C++ platform by simply adding a development component such as CDT. [4]

Eclipse Platform UI

As described in the previous section, the platform user interface is composed of two UI components: Jface and Workbench. These two components together form the Rich Client Platform which is providing a basic user experience and interactivity.

JFace: is a UI toolkit which has several classes to handle multiple UI related programming tasks. JFace works with SWT (Stand-alone Widget toolkit) . The most common UI tools such as text, images, font registries, dialogs are handled by JFace.



```java
import java.io.FileOutputStream;

public class SubmitHttpForm {

    private static String url =
        "https://vw261.viope.com/cgi-bin/contents/eclipse_

    public static void main(String[] args) {

        //Instantiate an HttpClient
        HttpClient client = new HttpClient();

        //Instantiate a GET HTTP method
```

11 items

	!	Description	Resource	Path
		TODO Auto-generated catch block	MsOfficeActio...	/net.javab
		TODO Auto-generated catch block	MsOfficeActio...	/net.javab
		TODO Auto-generated catch block	SendHandler.j...	/Send/src
		TODO Auto-generated catch block	SendHandler.j...	/Send/src

Figure 4. Its platform UI

User Interface components include for example entry fields, push buttons, tables, and tree views. The platform also performs many complex tasks such as management of window life cycle, docking views and editors, and drags and drop features. Figure 4 shows screen shoots of the workbench window with all basic UI components.

The top left corner shows the current project files in the user's workspace; the main text editor shows the content of the file which is currently edited; below the editor view, we get the list of tasks or to-dos (bottom right) in tasks view; the problem view shows the list of errors and their possible causes and the location of errors. Console view is dedicated to display the output of the running program. Outline view shows the outline of the content being edited.

Workbench

Eclipse Workbench is designed on the concept of a perspective, and it is very important to understand this concept before trying to use it effectively. Perspective here means the ability of providing different window arrangements based on user's preferred task selection. Whenever users perform a task such as writing any arbitrary Java program, all the necessary tools for coding and debugging will be automatically arranged by a window according to the current programming language. Such task specific window arrangements are named as perspectives.

Each perspective has specific tools such as menus and buttons that are appropriate to the given perspective. Figure 4 shows the current perspective which is *Java* .Based on this selected perspective we have views, such as the navigation view and outline view.

As Figure 5 illustrates, the Workbench enables the overall structure of the user interface and presents it as an extensible UI to the end user. Workbench typically facilitates the smooth interaction of tools and defines the extension points of these tools. [5, 1-2]

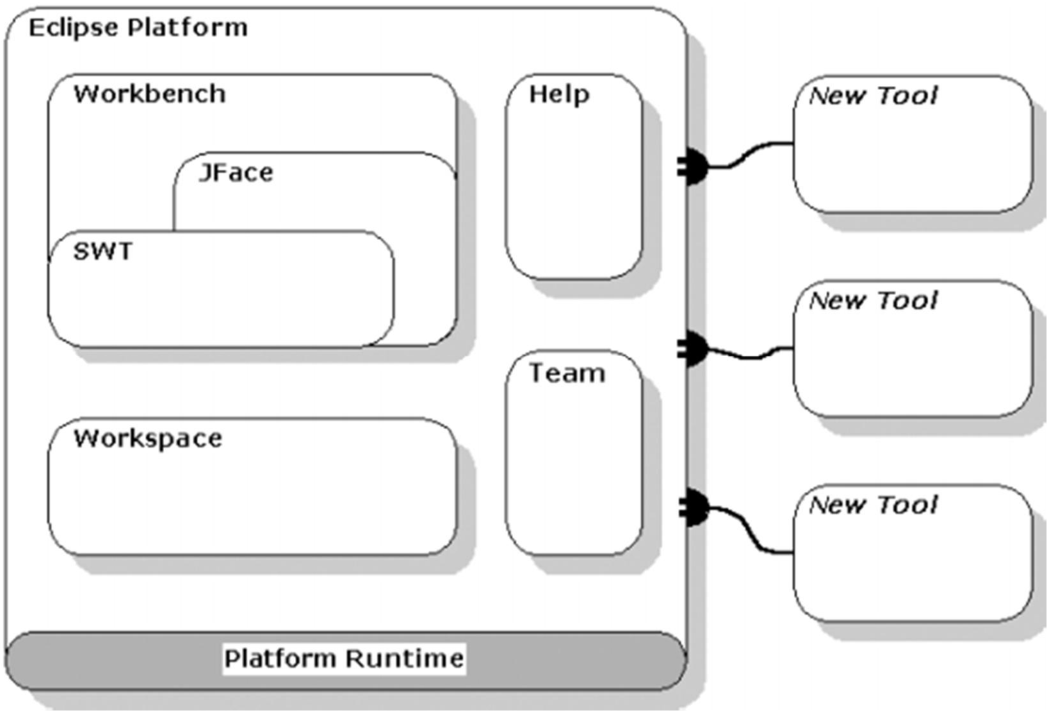


Figure 5. Platforms Architecture [2]

The Workbench enables the UI personal views.

Workspace

Workspace manages all the resources of the project which users are working on at the current directory. Whenever users create a project, it will be given a unique folder in the Workspace where all files and related resources are stored.

During code development the workspace is responsible for managing the overall resources associated with the project and adjust the files after they are updated.

According to the new releases and additional features, users can also create a project or point to a project outside the workspaces .This flexibility is very helpful when it comes to projects from different environments.

Users are also free to use multiple Workspaces as they want. It is possible to store projects that are totally different in nature with other projects and save them into separate workspaces as illustrated in Figure 5. [6,129-131]

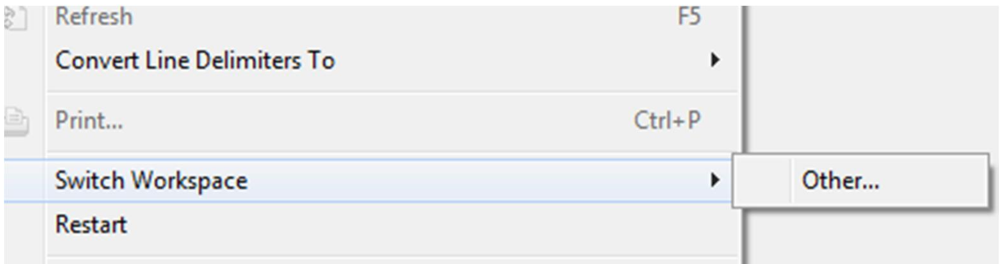


Figure 6. Switching to a different Workspace

2.3 Plug-in Overview

Eclipse is an open source that has been designed for the use of integrated web application development tool. However, the platform is nothing by itself, but rather it is a combination of hundreds of plug-ins into one. Figure 7, shows a sample plug-in, that defines how each plug-in is interacted and offers their services to other plug-ins.

Plug-ins are integrated with each other to offer a chunk of functionality and make users to have a multitasking application development tool. [7,129 - 131]

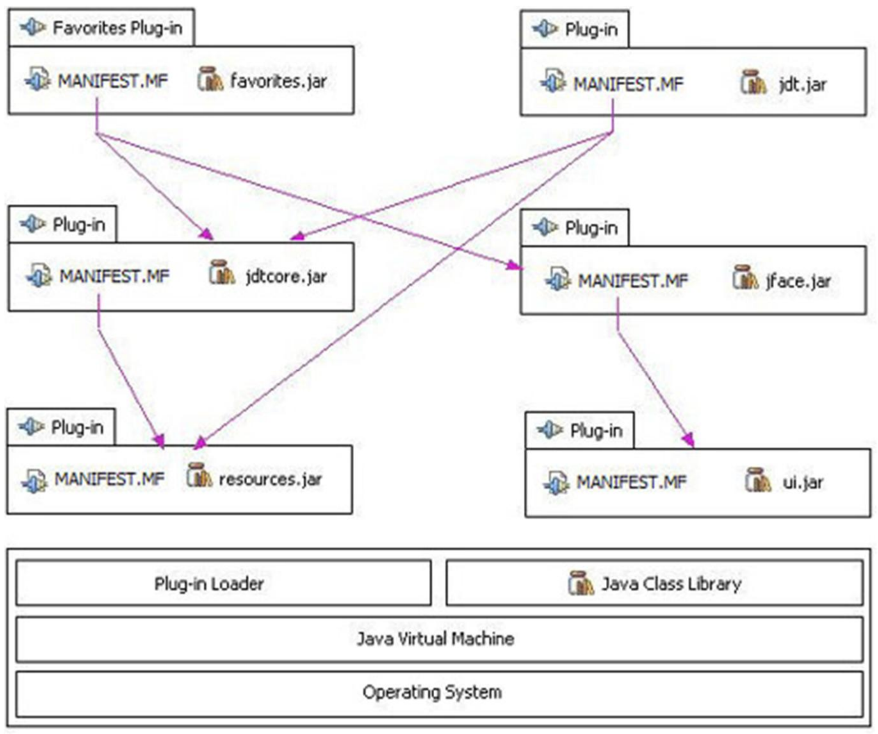


Figure 7. Combinations of plug-ins inside Eclipse [6]

Plug-in-based Architecture

One of the unique features of Eclipse is its extendibility based application development environment. Each distinctive application inside it is developed and deployed separately.

As shown in Figure 8, the role of Workbench and the Workspace is just creating a conducive environment for the new plug-in by providing an extension point to the existing IDE.

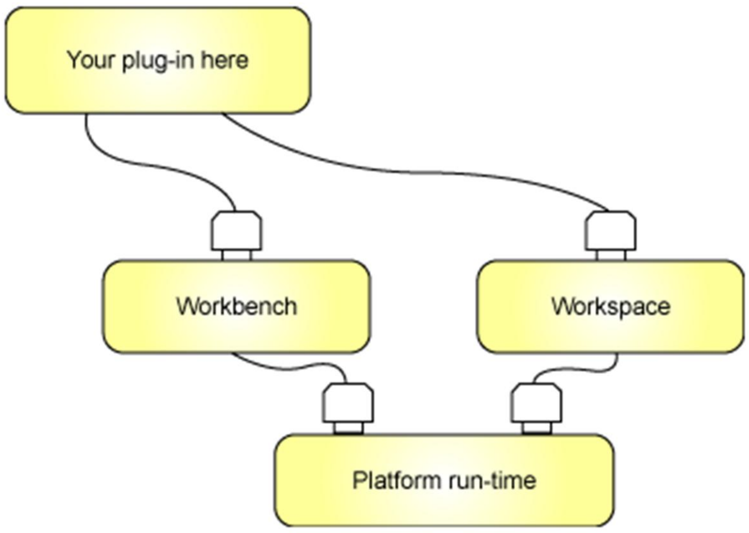


Figure 8. Workbench and workspace [8]

There are other components that can be extended by other plug-ins. For example a debugging component is the most common component which enables the program to run and deal with errors, even though it is not needed by most applications. [8, 5]

Eclipse extendibility for Other Applications

Extendibility of Eclipse is mainly due to its nature of architecture, where a small application can be added on the top of the existing one in order to enhance the functionality and to maintain extra services. Therefore, apart from compiling or debugging, it is also useful for applications such as networking, database management, and project management solutions such as version control.

2.4 NetBeans IDE

NetBeans is another common IDE development tool. This chapter will discuss the architecture of NetBeans and the relationship between an application created by a user and the NetBeans platform itself.

NetBeans Platform Architecture

The modern application development environments are becoming more complex and dynamic. Their complexity increases their flexibility and extendibility. Extendibility makes it possible for the applications to be divided into distinctive and independent applications. As a result, each part will become a building block of the general modular architecture.

These modular parts are used by other parts in the same application. Other parts must be able to use it and extend from it and vice versa. The division of applications into logically independent parts is the central concept of NetBeans. [8, 7-8]

Runtime Container

The fundamental building block of NetBeans Platform is a module. A module is a group of functionally related classes or components. Their integration method is specified as to how one module exposes itself to integrate with another module to achieve smooth functionality.

Modules are loaded dynamically and automatically by the NetBeans Platform, inside the NetBeans runtime container. This is the execution environment for the modules. As shown in Figure 9, the default five (5) NetBeans modules are included in the run time container.

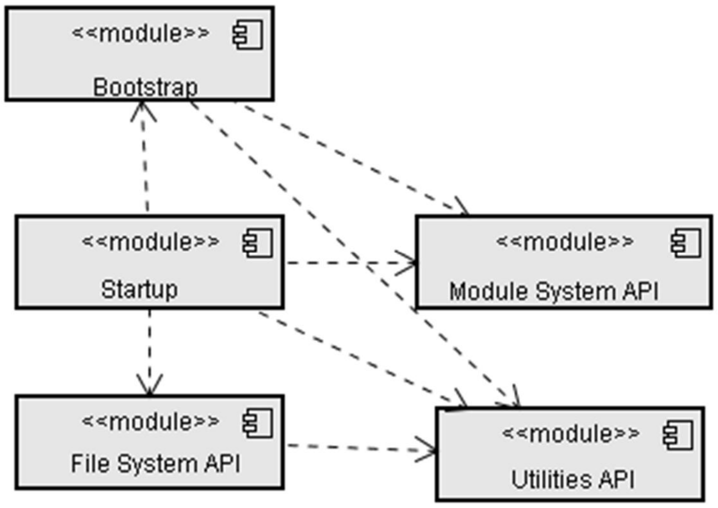


Figure 9. NetBeans runtime container [9]

`Bootstrap:`	This module enables the runtime container to understand the forth coming modules and prepares the boot class holder to load the startup module.
`Startup:`	This module provides necessary applications and codes to initialize the module system and file system modules.
`Module System:`	It lets the users to manage each module's life cycle in your application.
`File System:`	Provides module resources to the application.
`Utilities:`	This module provides miscellaneous components such as communication services between modules.

The container API provides the most basic services and functionalities to build applications on top of NetBeans platform. The five boxes are representing distinct modules that make up runtime container. NetBeans is a typical example of a rich client application which is composed of a set of modules as shown in Figure 10. NetBeans represents different core modules required by an application. The platform makes all the needed APIs and services available which is a considerable advantage for developers because this makes the development process considerably easier. [10,122 - 123]

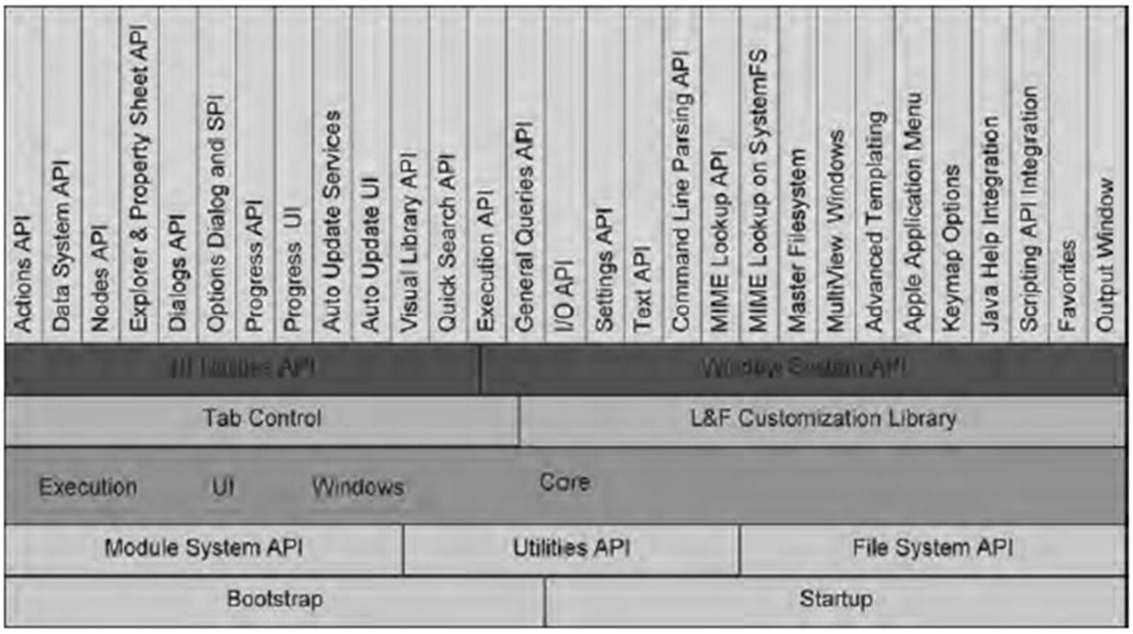


Figure 10. The structure of NetBeans IDE [8, 9]

Creating NetBeans Modules

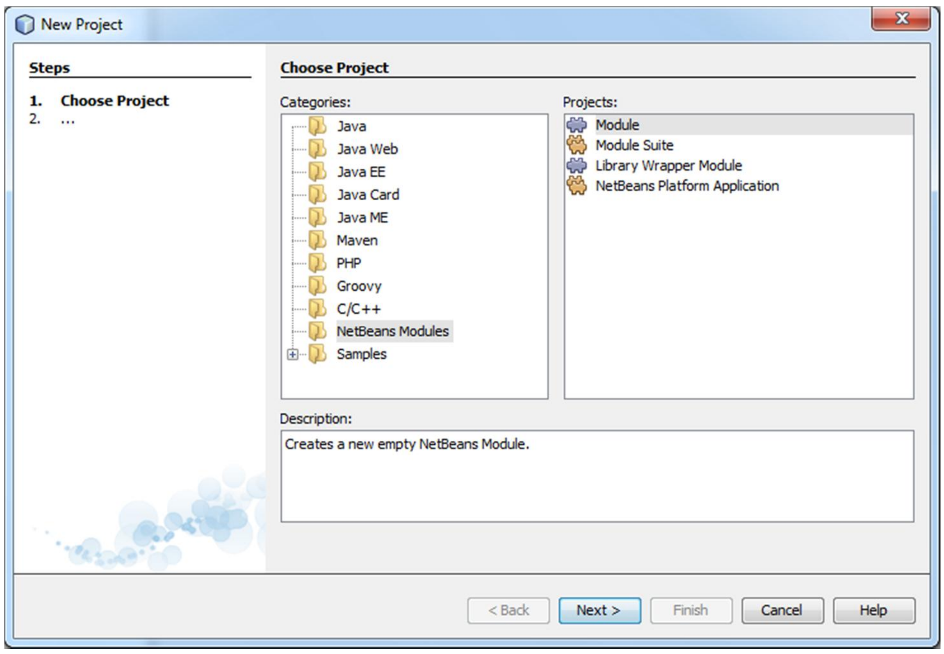
The next section focuses on how to design a NetBeans module by creating a sample module which shows the basic steps to accomplish a project.

Setting up the Module Project

When developing a NetBeans module the developer always needs to know if all the necessary project files exist and that they are structured properly. NetBeans provides essential steps and necessary files to create a Module.

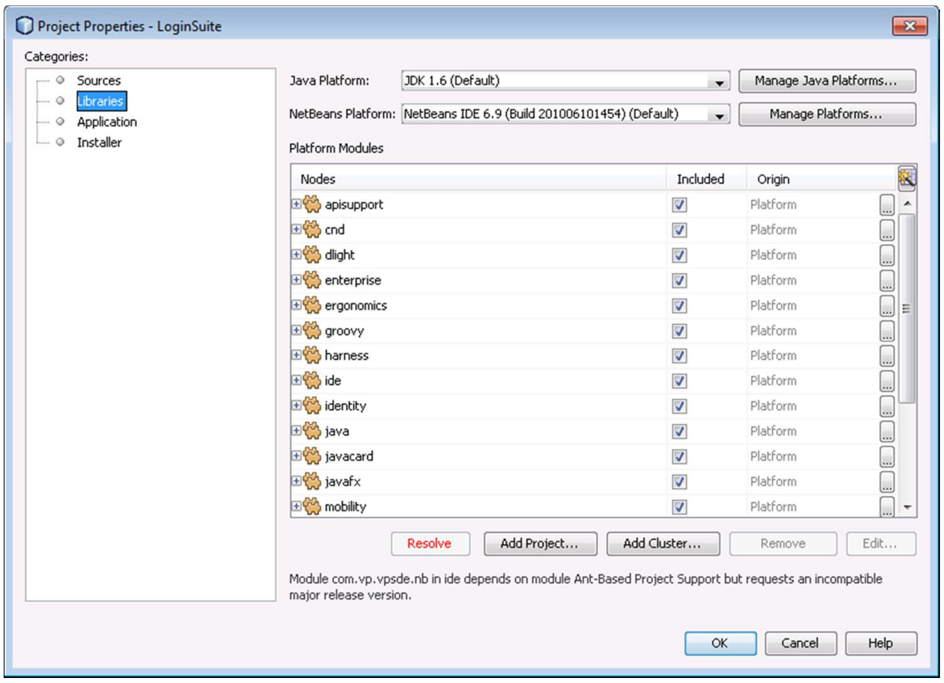
1. Start the local NetBeans and select File->New Project (see Figure 11).
2. In the next panel give a name for the module, such as `Hello_Viope`, and set the location,
3. Now the IDE is created. You can even run the application by `right-click` the module and select `run.` The whole IDE will be launched as a standalone application. However, we do not need the complete IDE features. The application can be made faster by reducing the unwanted libraries through the following steps.

4. `right-click` on the module suite name, choose properties, select libraries, Now you see the list of platform modules in the right side (see Figure 12). Deselect those modules you are not working with `Click` **`OK.`**



Creates a new empty NetBeans Module.

Figure 11. Creating a new NetBeans platform application project



Java Platform: JDK 1.6 (Default)

NetBeans Platform: NetBeans IDE 6.9 (Build 201006101454) (Default)

Nodes	Included	Origin
apisupport	✓	Platform
cnd	✓	Platform
dlight	✓	Platform
enterprise	✓	Platform
ergonomics	✓	Platform
groovy	✓	Platform
harness	✓	Platform
ide	✓	Platform
identity	✓	Platform
java	✓	Platform
javacard	✓	Platform
javafx	✓	Platform
mobility	✓	Platform

Module com.vp.vpsde.nb in ide depends on module Ant-Based Project Support but requests an incompatible major release version.

Figure 12. Deselect the unwanted module

Reducing the unwanted libraries will help the system to run faster.

If we run the plug-in right now, we will see nothing, since we do not have any items or actions defined on the plug-in yet as illustrated in Figure 13.

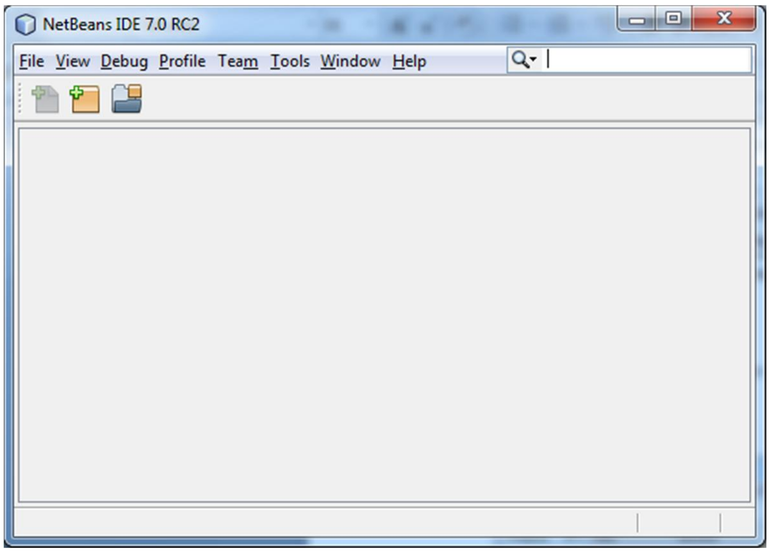


Figure 13. Blank NetBeans Module

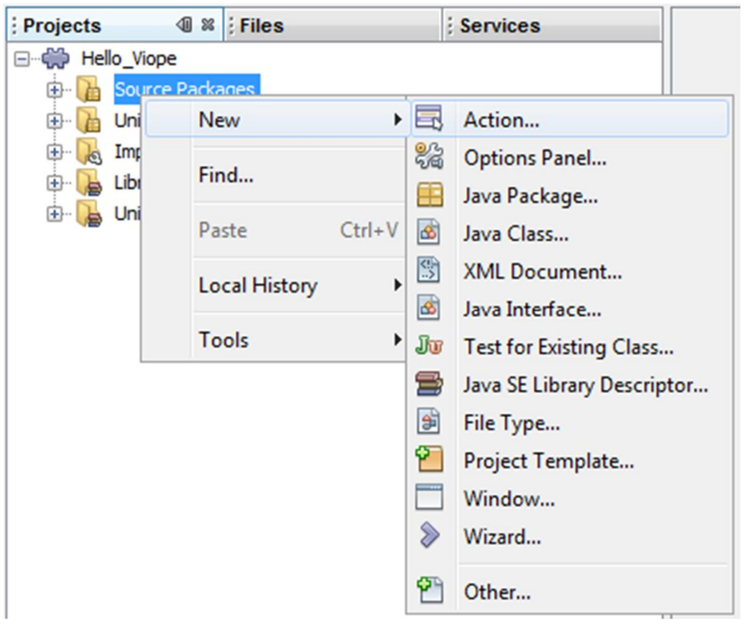
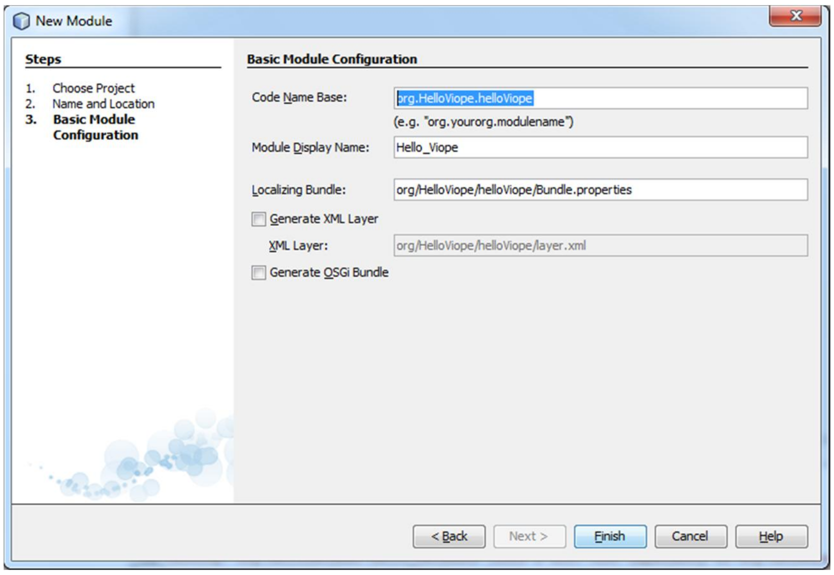


Figure 14. Creating actions for module

Now we need some actions to function in our plug-in, therefore, `right click` on the module (see Figure 14). Select NetBeans modules. Now select the project type NetBeans platform application on the right side, set up the code name such as org.HelloViope.helloViope.


Basic Module Configuration

Code Name Base: org.HelloViope.helloViope
(e.g. "org.yourorg.modulename")

Module Display Name: Hello_Viope

Localizing Bundle: org/HelloViope/helloViope/Bundle.properties

☐ Generate XML Layer

XML Layer: org/HelloViope/helloViope/layer.xml

☐ Generate OSGi Bundle

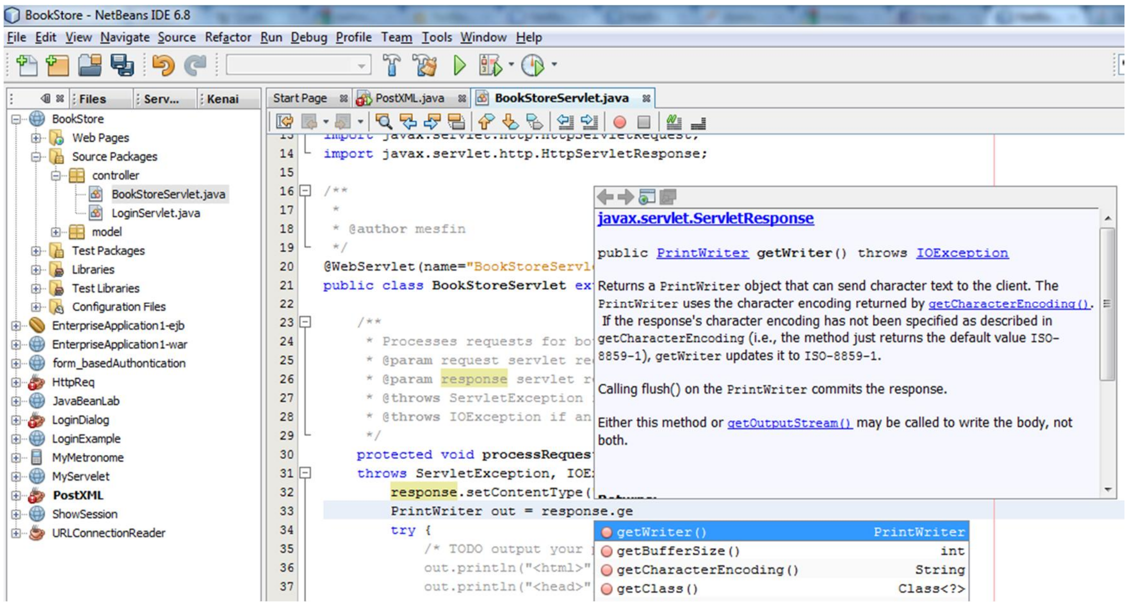
Figure 15. Setting New Module

Finally set up the screen *Name, Icon and Location* (see Figure 15). In class name choose `Hello Viope`.

NetBeans IDE Features

Source Code Editor

NetBeans supports several programming languages including Java, PHP, C++, XHTML, HTML, and JSP .It is possible to choose other languages by adding extra module to support the specific language.


```
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author mesfin
 */
@WebServlet(name="BookStoreServl
public class BookStoreServlet ex

    /**
     * Processes requests for bo
     * @param request servlet re
     * @param response servlet r
     * @throws ServletException
     * @throws IOException if an
     */
    protected void processReques
    throws ServletException, IOE
        response.setContentType(
        PrintWriter out = response.ge
        try {
            /* TODO output your
            out.println("<html>"
            out.println("<head>"
```

javax.servlet.ServletResponse

`public PrintWriter getWriter() throws IOException`

Returns a `PrintWriter` object that can send character text to the client. The `PrintWriter` uses the character encoding returned by `getCharacterEncoding()`. If the response's character encoding has not been specified as described in `getCharacterEncoding` (i.e., the method just returns the default value `ISO-8859-1`), `getWriter` updates it to `ISO-8859-1`.

Calling flush() on the `PrintWriter` commits the response.

Either this method or `getOutputStream()` may be called to write the body, not both.

```
getWriter()                 PrintWriter
getBufferSize()                     int
getCharacterEncoding()           String
getClass()                     Class<?>
```

Figure 16. NetBeans editor

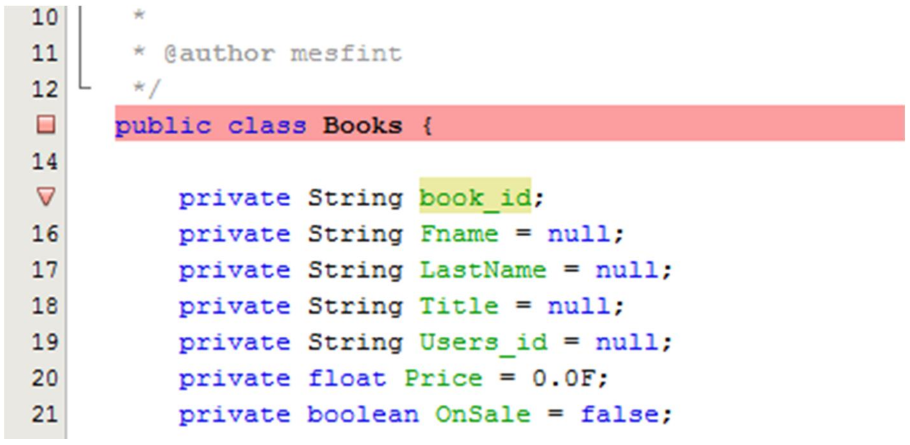
The Editor Parses the source code as it written on the editor. The editor also checks errors and highlights the occurrences, providing code hints, warnings, and quick fixing which are the most common features that NetBeans editor offers to programmers.

Refactoring codes means restructuring the source code without altering its functionality.
Reasons for Code refactoring are:

- To make code readability easier.
- To reduce repetitions.
- To make the code easy to understand.

Figure 16, shows fast code completions. The editor supports multiple languages. As the code is typed on the editor, a list of possible code completions such as package names, classes, interfaces, and common methods will be provided regardless of which language has been selected.

Debugging is a method to check if the application has any problems or to see the sequential work of functions defined in the program. Debugging can be applied by setting breakpoints and watches in NetBeans, and then running the application in the debugger. NetBeans will provide essential information regarding the overall situation of the application to discover the errors.



```
 *
 * @author mesfint
 */
public class Books {

    private String book_id;
    private String Fname = null;
    private String LastName = null;
    private String Title = null;
    private String Users_id = null;
    private float Price = 0.0F;
    private boolean OnSale = false;
```

Figure 17. NetBeans editor debugger

As shown above in Figure 17, there are two breakpoints. The first red square refers to a line breakpoint, which tells the debugger to stop further execution of the code. When a program stops on a breakpoint, you can perform actions such as analyzing variables if they are working as intended. The second red triangle represents a field of breakpoint. Users can customize breakpoints based on their interest. [11, 87]

2.5 Comparison between integrated development environments

The purpose of this chapter is not to evaluate each IDEs tools. Rather, it is to compare the most common development environments as to how they are capable of providing services and their accessibility to the users as shown in Table 1.

IDEs	License	Multiple Language Support	Windows	Linux	Mac Os	Code Auto complete	Plug-in Support	Other Platforms
Visual Studio	Proprietary	✓	✓	✗	✗	✓	✗	✗
Eclipse	EPL	✓	✓	✓	✓	✓	✓	JVM
NetBeans	CDDL	✓	✓	✓	✓	✓	✓	Solaris

Table 1. Comparison of integrated development environments

Visual studio is provided under proprietary license (PL) commercial software which restricts further modification towards the software. However, Express Edition is a freeware, whereas Both Eclipse (EPL) and NetBeans (CDDL) are allowing third parties to use the source code of the framework, and to modify copy, and distribute. The modified version provided by developers must adhere to the state-of–the-art of the framework before release. [8,280-281]

All the three IDEs are supporting multiple languages. In addition, NetBeans and Eclipse have additional plug-ins or modules which are designed for a specific language that needs to be installed on top of the existing framework. Visual studio is a windows operating system dependent; however, the former tools are also running on Linux and Mac OS. [12, 13, 5-7]

2.6 Web Services as a tool for eLearning

Countries have been devoted to enhance their educational system and electronic education in recent years, due to the fact that contents and tool support for education have become widely affordable. Many developed countries widen their educational system towards online learning and virtualized learning system. Based on the current development arena, eLearning systems use internet as the distribution channel for their contents quite efficiently. Even most remote areas are able to have access to the system which is controlled by an administrator.

These systems are mainly designed based on peer to peer architecture. In a typical eLearning system environment, course administrator and learners are the main group of users. eLearning system basically needs to cope with a heterogeneous group of users with various backgrounds and levels of understanding.

Web Services Architecture for eLearning System

Web services are distinct software applications which have unique URI addresses and are designed to accomplish a certain task or a group of tasks that use internet for communication and composition structure. Web services have to be independent of operating systems. Web services are commonly based on conventional standards; the most common ones are:

- XML based Specification SOAP (Simple Object Access Protocol)
- UDDL (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

As shown in Figure 18, the web service architecture is divided into two parts: a client part and web services part provided by companies. The client part is mainly a collection of software that supports the learners to access to the eLearning material. On the other hand, the web service part is implemented on a distributed server. Specifically they are dedicated to author eLearning contents, tracking users and performing evaluations based on their performance towards the materials provided.

The web services can also be integrated with portable devices such as mobile, as long as the client is implemented to that particular device. [14]

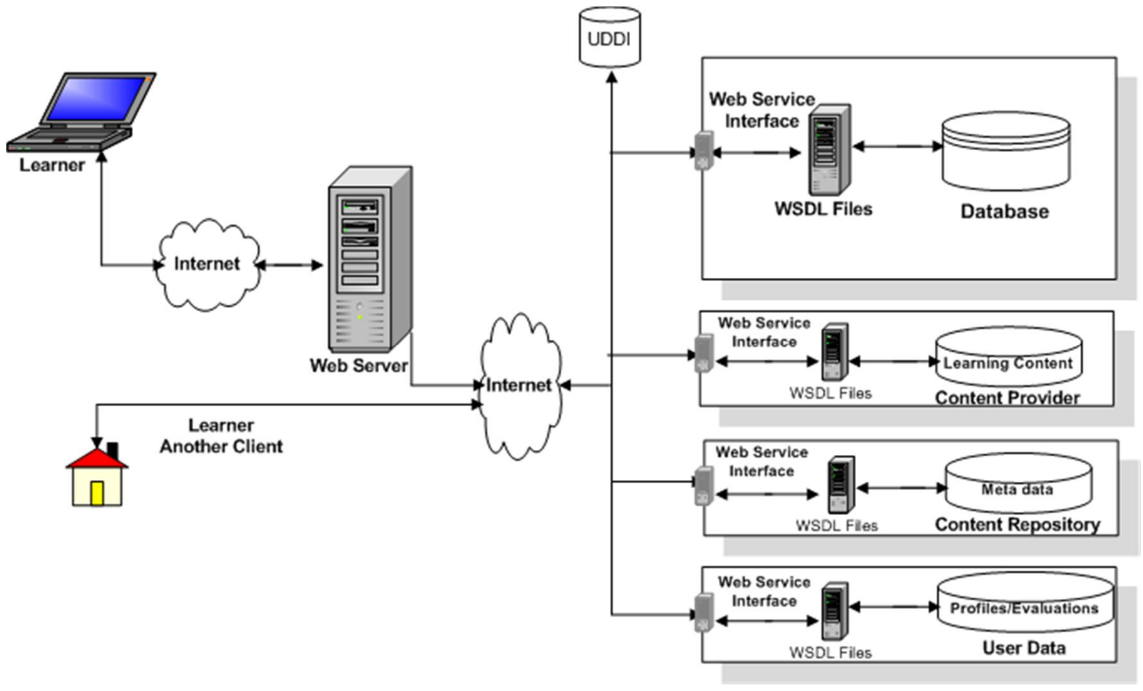


Figure 18. High-level Web service Architecture for eLearning [14]

3 Client Application

3.1 The Existing eLearning System

Viope Solutions OY is one of the pioneer companies in the eLearning industry and it was founded in 2001. The company mainly provides online programming and education materials in mathematics. The major clients in Finland are high schools, technical colleges and universities. The company has also managed to reach international customers overseas. Currently the company is running its official web site in three languages: Chinese, Finnish and English. [15]

Online Courses

The company's online courses are designed for self-learning and as a support material for teachers. The course materials are meant to be accessed on the internet. The courses include programming languages: c, c++, Java, Python, Ruby, Sql and Php.The programming courses are arranged in an easily understandable way to promote user interactivity. Each course is divided into chapters and each chapter has both theoretical and practical lessons. At the end of every lesson students are expected to do certain programming tasks which are designed to help them to anticipate possible questions, review what they have learned and to begin putting their knowledge into practice.

Service Users

The eLearning system is manipulated by the students or self-learners and the course administrator or instructor. As shown in Figure 19, below indicates, each user has a specific role and access right. Students are eligible to access the courses and their corresponding exercises which they are already enrolled to, whereas instructors are course administrators who are able to manage courses, exercises and evaluate students work among other things.

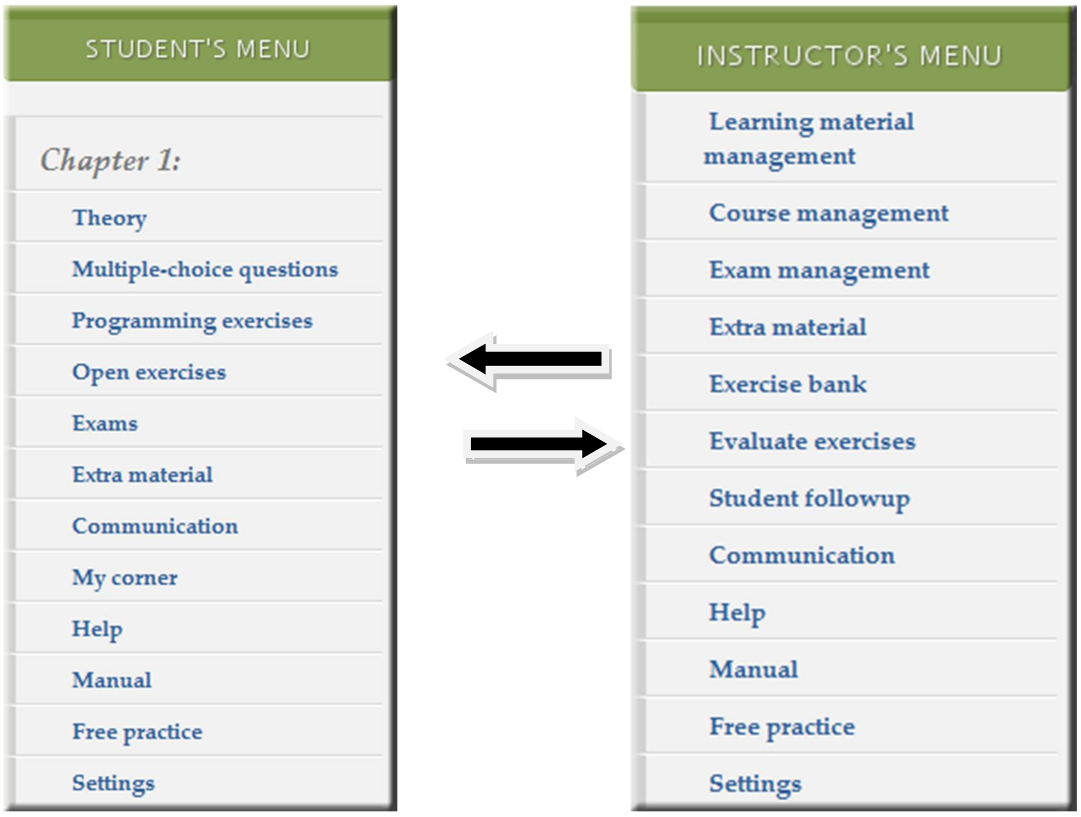
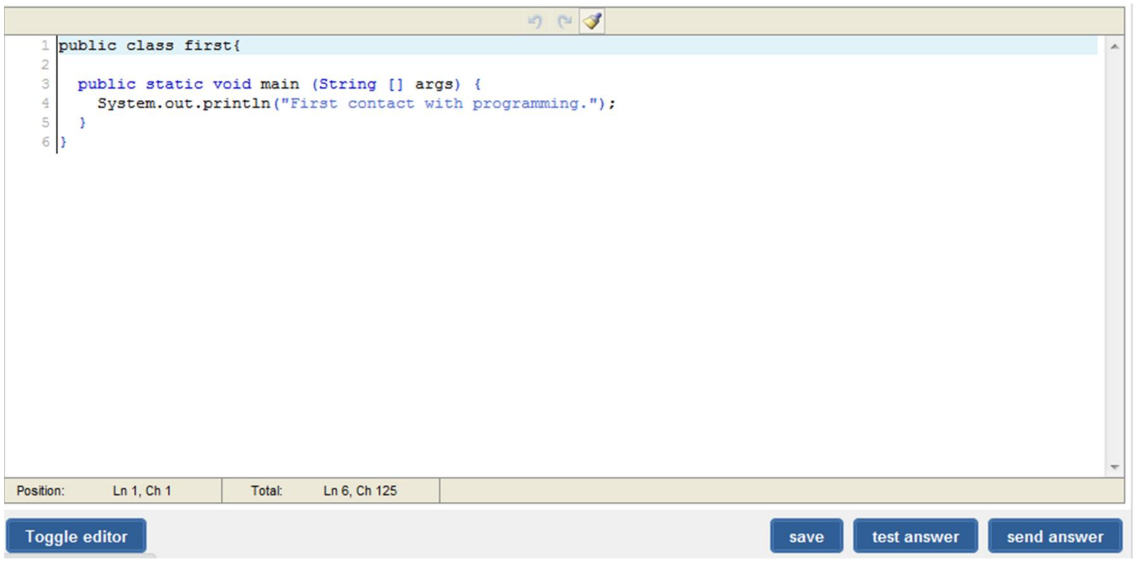


Figure 19. Students and instructors menu [16, 17]

Exercise Management System

Viope has an existing exercise management system which is a form input based system. It receives a code from learners and sends it to a remote server where the final evaluation is taking place, as illustrated below in Figure 20.



```
public class first{

  public static void main (String [] args) {
    System.out.println("First contact with programming.");
  }
}
```

Figure 20. Viope's exercise editors

The server performs tasks such as handling the code and gives feedback based on user's submitted solution. However, the existing editor is not a functionally complete editor to consider the solution as a complete tool for writing codes: some limitations that the current system faces are listed below.

Limitations of Client Editor

- No instant code feed back
- Codes are only checked by the server(over load to server)
- Generic error feedback
- No quick fix hint to users
- Fixed class names

3.2 Requirements and Specification

Purpose

The purpose of the project is to design an Eclipse plug-in which integrates the existing Viope eLearning system (exercise management portion) with a local IDE. The users interact with Viope's remote server from their locally installed integrated development tool.

Scope

The final output of this project is expected to communicate learners with Viope's exercise management server, so that they are able to send their solutions to the server across our solution.

Since users are able to work directly with Eclipse IDE, they will have all support from it such as instant code error feedback. Eclipse IDE also provides help before sending their solution to the exercise handling server. This minimizes the considerable work load of the system.

3.3 Overall Description

System Environment

The Exercise management system has two active actors. The learners and the exercise administrator. The actors have two distinct tasks as shown in Figure 21.

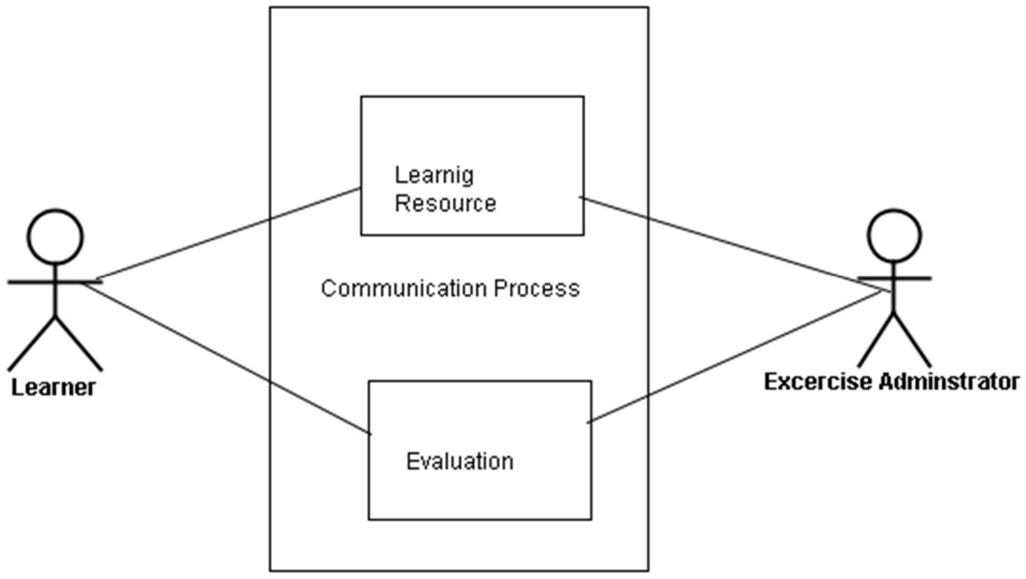


Figure 21. System Environments

The Exercise management server has been divided into two task operations the exercise storage and the user profile. The server analyzes the solutions provided by learners and issues feedback based on their answers.

Functional Requirements Specification

This section outlines the sequential activities of a learner discussing how users are interacting with the system and receiving feedback from the remote server after they have accomplished their task. As shown in Figure 22, users are able to fetch assignments from the exercise bank which includes chapters, exercises and update the code in the given exercise. Finally, they send them to the dedicated server for final evaluation.

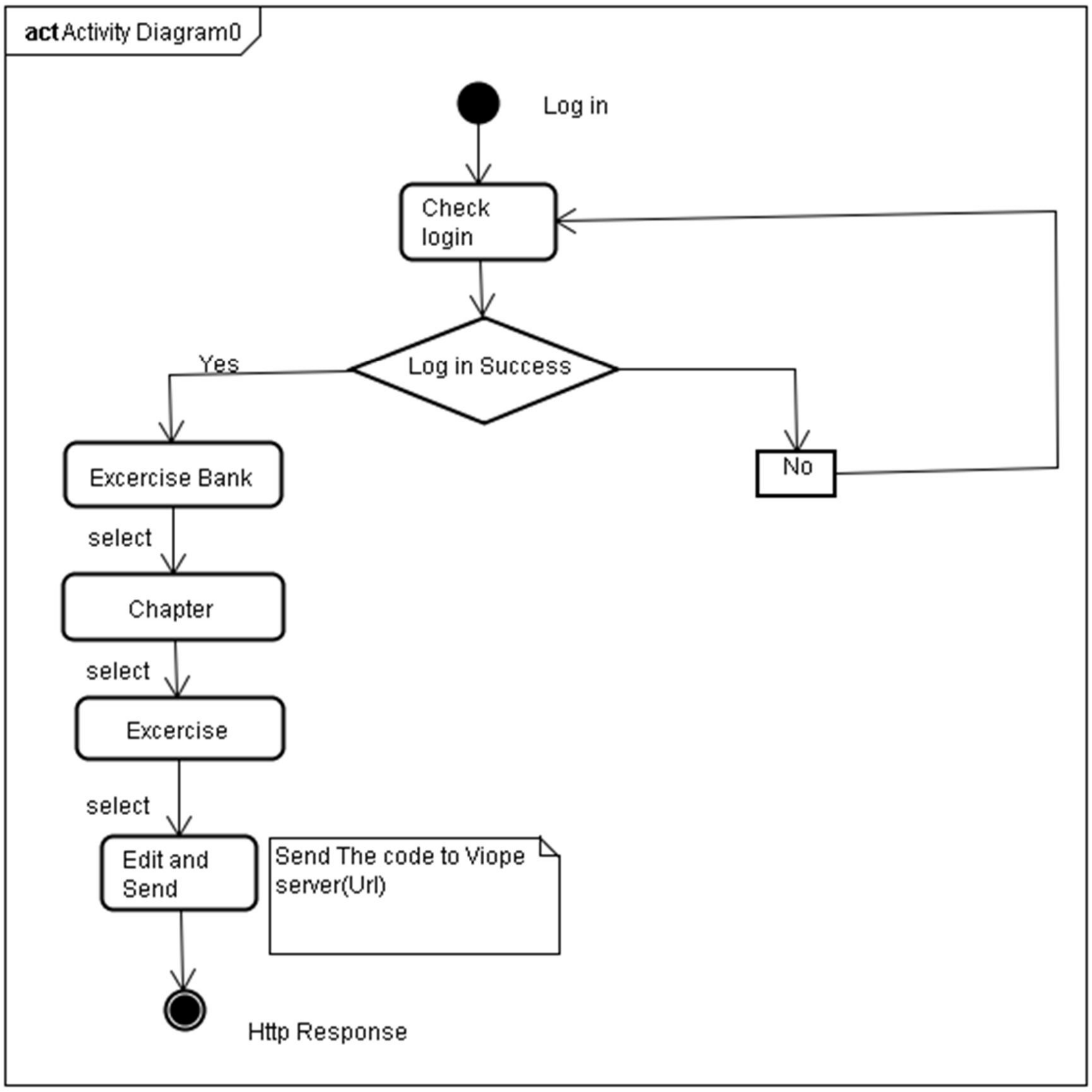


Figure 22. Functional requirement specifications

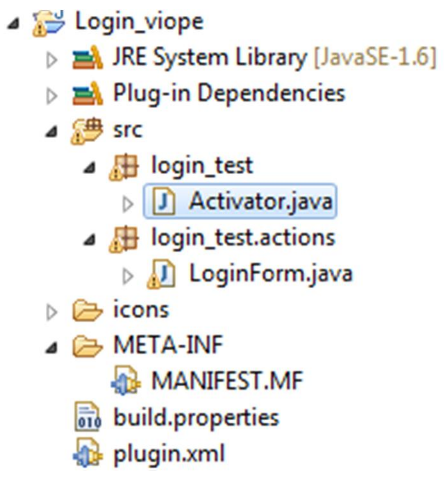
3.4 Plug-in Based Development Architecture

Creating an Eclipse plug-in is not a hard job to accomplish for an experienced coder. However, for a newbie it is a challenge. No matter what the output looks like, it is always a combination of different tools and methods that are merged harmoniously and make things work as intended.

As mentioned in the previous section, Eclipse is the collection of plug-ins and each of them is contributing a distinct functionality. Their service is monitored by the runtime which is activating when there is a demand for it. All plug-ins in the system are collected in a subdirectory named Eclipse/plug-ins in the installation directory.

Solution #1-Login_Viope

Login_Viope plug-in is the first application we developed; its functionality is to accept user name and password from registered Learners. They are authenticated if correct credentials are inserted. Figure 23 illustrates the project directories and the supporting files.



```
Login_viope
  JRE System Library [JavaSE-1.6]
  Plug-in Dependencies
  src
    login_test
      Activator.java
    login_test.actions
      LoginForm.java
  icons
  META-INF
    MANIFEST.MF
  build.properties
  plugin.xml
```

Figure 23. Login_viope Project files

Plug-in Directory

LoginForm.Java: This is the main Java file. This file contains methods and logics that govern the plug-in functionality. The file can be given any name as long as it is described in the project configuration file. The code can be found in Appendix 1.

Icons: All image or icon files that are referenced in the plug-in .xml file are placed in this file.

META-INF: This directory contains program configuration files and runtime information.

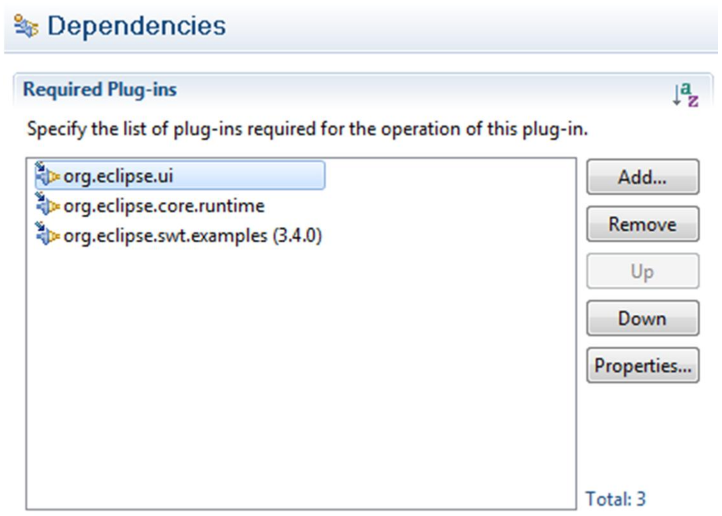
Plug-in .xml: This is an XML format file that describes the login viope plug-in and how it integrates with Eclipse. See the file in Appendix 2.

Manifest Editor

As shown in the above Figure 23, there is only one manifest file per plug-in describing the various sections and components the plug-in have. The file contains four main sections such as:

- Dependencies
- Run Time
- Extensions and Extension Points. [6,132-133]

Dependencies



Dependencies

Required Plug-ins

Specify the list of plug-ins required for the operation of this plug-in.

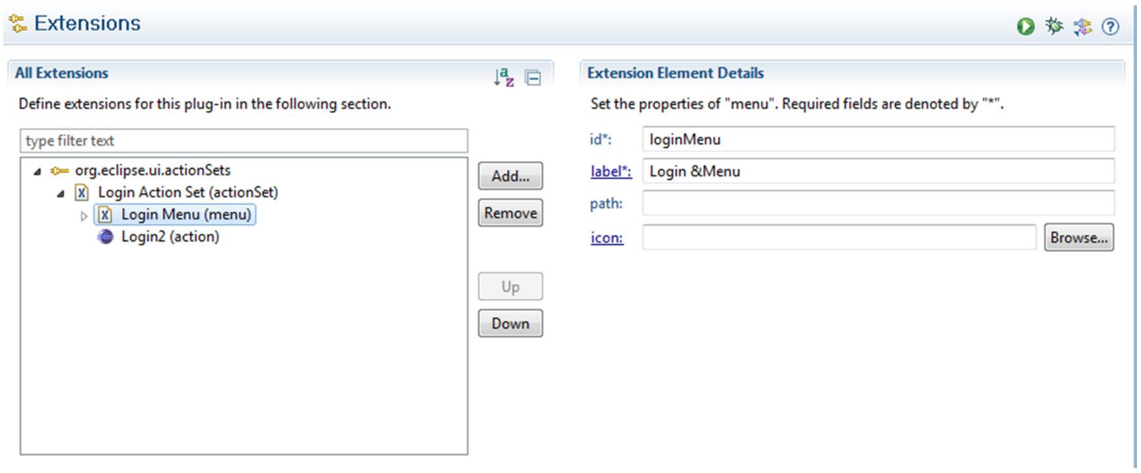
- org.eclipse.ui
- org.eclipse.core.runtime
- org.eclipse.swt.examples (3.4.0)

Figure 24. Plug-in dependencies

Menus that are attached on the Eclipse toolbar are extensions of library called *org.Eclipse.ui* .All menu items are dependent on this library for their functionality. The library is also responsible for the integration of the user with a development tool. As illustrated in Figure 24, the second package *org.Eclipse.core.runtime provides* a support for the runtime platform, core utility methods and extension registry.

The last dependency library required by this solution is *org.Eclipse.swt.*examples.It is a very important component that is related to a standard widget tool. It is useful for manipulation and integration of the widget with the system.

Extensions


Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

- org.eclipse.ui.actionSets
  - Login Action Set (actionSet)
    - Login Menu (menu)
    - Login2 (action)

Extension Element Details

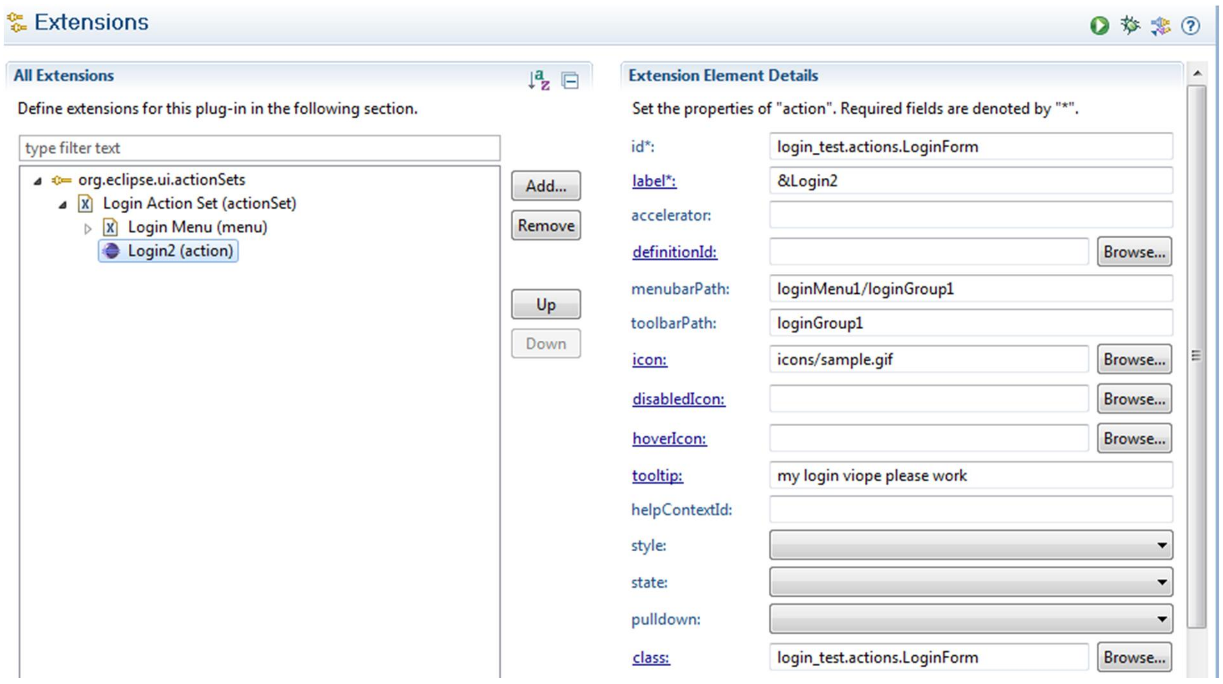
Set the properties of "menu". Required fields are denoted by "*".

id*:	loginMenu
label*:	Login &Menu
path:	
icon:	

Figure 25. Extensions of plug-in

Extensions are the additional feature contributors to the system in which the extension points are welcoming the extensions by providing a space for additional extensions without interfering the overall system function.

Menu items are extensions which have unique functions individually or as a group. They are generally collections of action sets. In this solution the login menu is extended by org.Eclipse.ui.actionSets as shown in Figure 25.The extension point defines its structure and related components as well as some sections to evaluate the new extension (login menu) .The login menu has an action set which specifies its internal identifier and visibility in the workbench.


Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

- org.eclipse.ui.actionSets
  - Login Action Set (actionSet)
    - Login Menu (menu)
    - Login2 (action)

Add... Remove Up Down

Extension Element Details

Set the properties of "action". Required fields are denoted by "*".

id*:	login_test.actions.LoginForm
label*:	&Login2
accelerator:	
definitionId:	
menubarPath:	loginMenu1/loginGroup1
toolbarPath:	loginGroup1
icon:	icons/sample.gif
disabledIcon:	
hoverIcon:	
tooltip:	my login viope please work
helpContextId:	
style:	
state:	
pulldown:	
class:	login_test.actions.LoginForm

Figure 26. Extension, actual action of the menu

As Figure 26 displays, login2 (action) element defines the login menu item. The right side displays all the details of the menu.

Extension Points

Declaring extension points in every plug-in manifest is a common practice throughout Eclipse. Defining extension point means exposing an entire plug-in interface for other plug-ins to use. In response, the other plug-in declares extensions to that extension point. All information about the available extension point and the supplied extensions are registered in “Extension registry” as illustrated below in Figure 27.

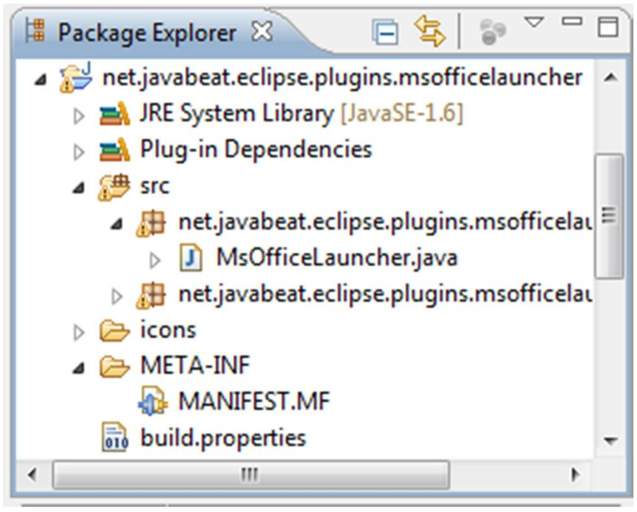
```
<plug-in >
   <extension
         point="org.eclipse.ui.actionSets">
      <actionSet
            label="Login Action Set"
            visible="true"
            id="Login.actionSet">
         <menu
               label="Login &amp;Menu"
               id="loginMenu">
            <separator
                  name="loginGroup1">
            </separator>
         </menu>
      </actionSet>
   </extension>

</plug-in >
```

Figure 27. Sample extension point declaration

Extension points are defined in XML schema which is referred to as plug-in.xml .This file defines the various components and contracts of the extension point. The above code snippet shows the menu extension's different elements that define how it will be presented on the workbench. [18,477]

Solution #2-Accessing Courses

The purpose of the second solution is to allow users to access exercises dynamically from Viope's system while they are working on their local Eclipse editor. To ensure this service is exists, first users must have logged in successfully. After that they will be provided a unique session id and identified as a registered user. Finally, this id allows them to accessing the course exercise as shown in Figure 28.


Package Explorer

```
net.javabeat.eclipse.plugins.msofficelauncher
    JRE System Library [JavaSE-1.6]
    Plug-in Dependencies
    src
        net.javabeat.eclipse.plugins.msofficelau
            MsOfficeLauncher.java
        net.javabeat.eclipse.plugins.msofficelau
    icons
    META-INF
        MANIFEST.MF
    build.properties
```

Figure 28. Solution #2 project files

This solution uses dummy exercises because the courses we have designed have not been launched yet. The extension point and the contributor extension have been defined in the xml schema attached in Appendix 3. Like the first plug-in, we have also used program dependencies or libraries, which provide extra services and functionalities to the new solution. Since we also have menu items in this solution, the use of UI libraries is very crucial. As shown in Figure 29, the top-level element of this extension is an org.Eclipse.ui, which is in charge of the menu items located on the tool bar, as well as their label and visibility. The second extension is in charge of the items runtime activity.

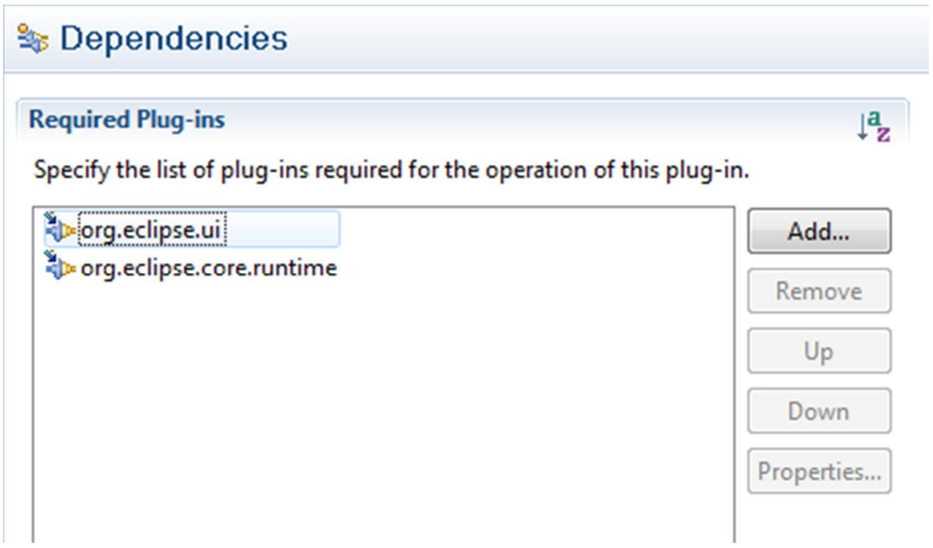


Figure 29. Plug-in dependencies

The courses menu is attached to the main tool bar. Only registered users are able to access these courses and the corresponding exercises.

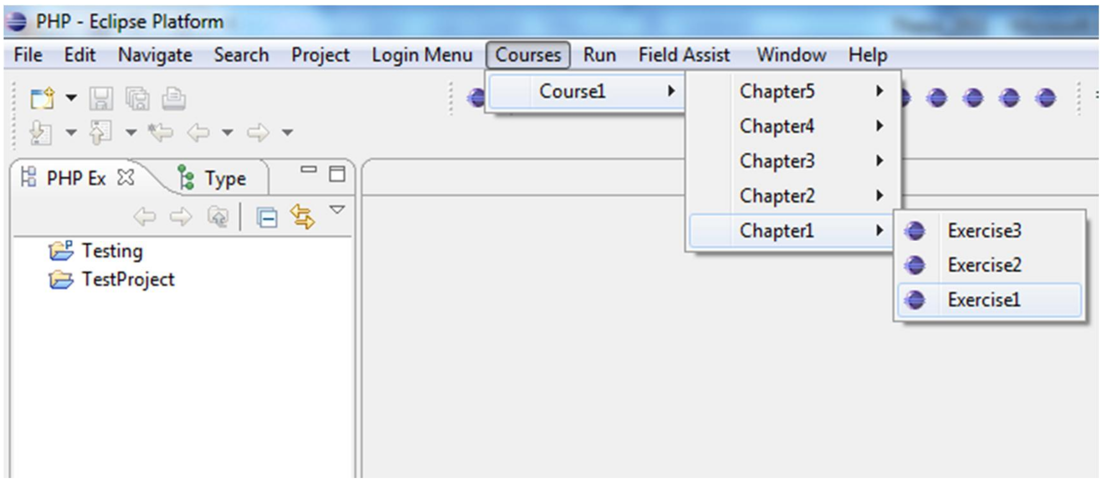


Figure 30. Course application attached on toolbar menu

As shown in Figure 30, the course plug-in is already included in the main workbench. Each course has chapters and exercises, which are populated dynamically from the remote exercise management system.

4 Development Methods and Technologies

4.1 Standard Widget Toolkit

Standard Widget Toolkit (SWT) is an open source widget and a building block of Graphical User Interface (GUI) for Eclipse. More specifically it controls text fields, buttons, and check boxes. In most cases it is useful using this toolkit working wizards, dialogs and property pages.

SWT includes dozens of rich features (Java packages) and it is possible to write different applications using it. Therefore, the concept of SWT including core systems, widgets, layouts, and events is essential.

Basic SWT application structure

SWT is an inbuilt application with an Eclipse for its own use; it can also be downloaded from the internet to use it in different Java applications. All native applications designed by standard widget have the same structure. When working with SWT we need to use two classes: Display and Shell.

The Display facilitates the connection between SWT and the GUI system of the operating system. Displays are generally the back bone of SWT implementation that is used to manage the platform events and to control UI threads. The presence of the display is a must before creating any windows, and it should be disposed when the shell is closed. Display is crucial particularly for multi-tasking applications. A shell is a "window" managed by the operating system as shown in Figure 30 [19, 92 - 93].

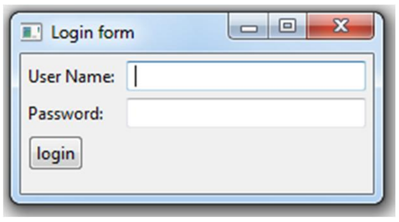
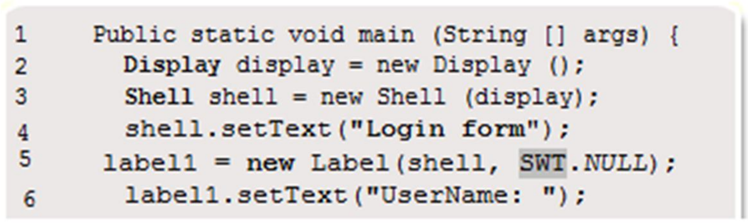
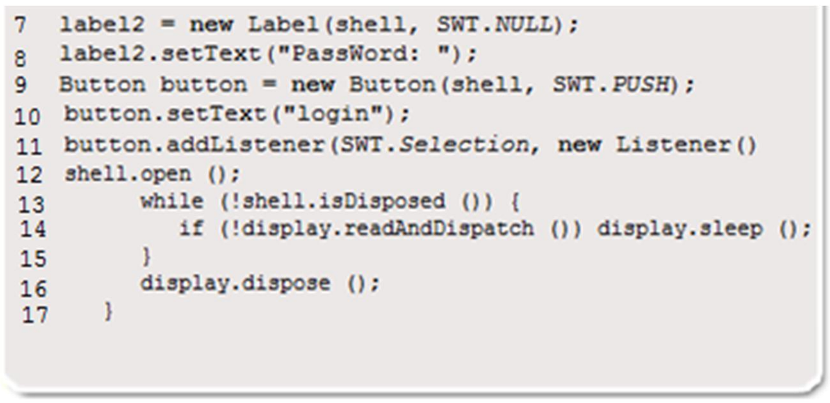


Figure 30. Sample SWT shell and button application



```
Public static void main (String [] args) {
  Display display = new Display ();
  Shell shell = new Shell (display);
  shell.setText("Login form");
 label1 = new Label(shell, SWT.NULL);
  label1.setText("UserName: ");
```



```
7   label2 = new Label(shell, SWT.NULL);
8   label2.setText("PassWord: ");
9   Button button = new Button(shell, SWT.PUSH);
10  button.setText("login");
11  button.addListener(SWT.Selection, new Listener()
12  shell.open ();
13        while (!shell.isDisposed ()) {
14           if (!display.readAndDispatch ()) display.sleep ();
15        }
16        display.dispose ();
17     }
```

Figure 31. Applying SWT in the project

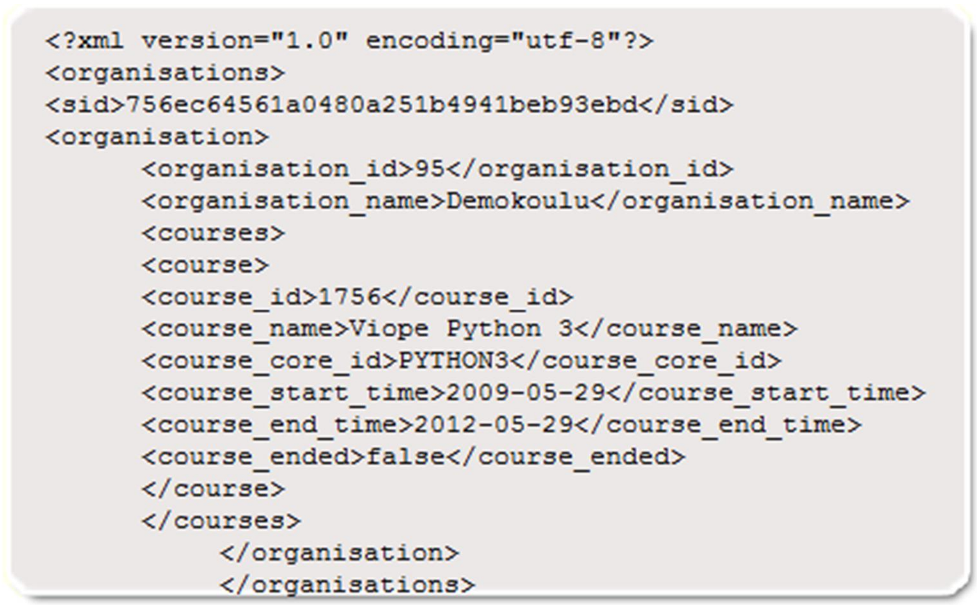
The Workbench is handling all the necessary start up functions while we are working on the User Interface using the standard widget. Basically it is running as a standalone application.

Steps to design application using SWT: as illustrated in Figure 31.

- Line 2: create **Display** and **Shell** Objects.
- Line 3: the shell object is connected to the display.
- Line 4: assign a name for the login window.
- Line 5-8: assigning label names to the login form.
- Line 9-11: creating button using shell, assigning “push or clickable” feature and define event listener to it.
- Line 12: Opens the shell/login form once the login menu item is pressed.
- Line 13 – 15: assign an event dispatching the while loop statement.The purpose of the loop is to react with events triggered by users if they are exist. Otherwise the display will be idle, in order to conserve resources.
- Line 16: finally, release the connection.

4.2 Tree-Based XML Parsing

Literally xml parsing means reading the xml data from a file or from the application itself and converting it into a human readable format. The parsed xml file will be displayed in the console. A sample code was used in this project to pars xml data as shown below in Figure 32.



```
<?xml version="1.0" encoding="utf-8"?>
<organisations>
<sid>756ec64561a0480a251b4941beb93ebd</sid>
<organisation>
      <organisation_id>95</organisation_id>
      <organisation_name>Demokoulu</organisation_name>
      <courses>
      <course>
      <course_id>1756</course_id>
      <course_name>Viope Python 3</course_name>
      <course_core_id>PYTHON3</course_core_id>
      <course_start_time>2009-05-29</course_start_time>
      <course_end_time>2012-05-29</course_end_time>
      <course_ended>false</course_ended>
      </course>
      </courses>
           </organisation>
           </organisations>
```

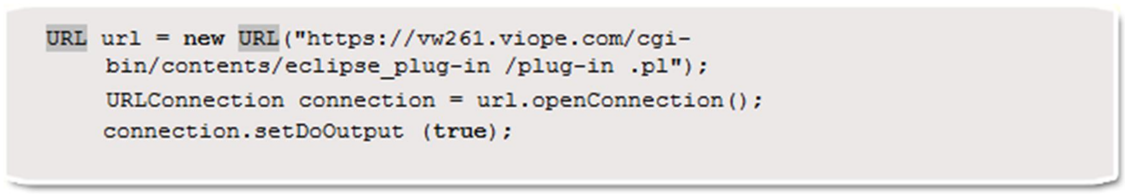
Figure 32. Sample XML file used in the project

The above xml code snippet is parsed using a Java xml parser. The xml is dynamically generated from a remote server where it is managed centrally. The xml file is generated by the remote server as a response. However it was originally mixed with some other html syntax codes, therefore, it was necessary to use a Java command to filter out all unnecessary codes inside the xml file.

4.3 Java Technology

Java is the main programming language used in this project. It is extremely difficult to experiment Eclipse, without using Java. This is because the development environment itself is a Java based tool, and consequently all the libraries and classes are Java based.

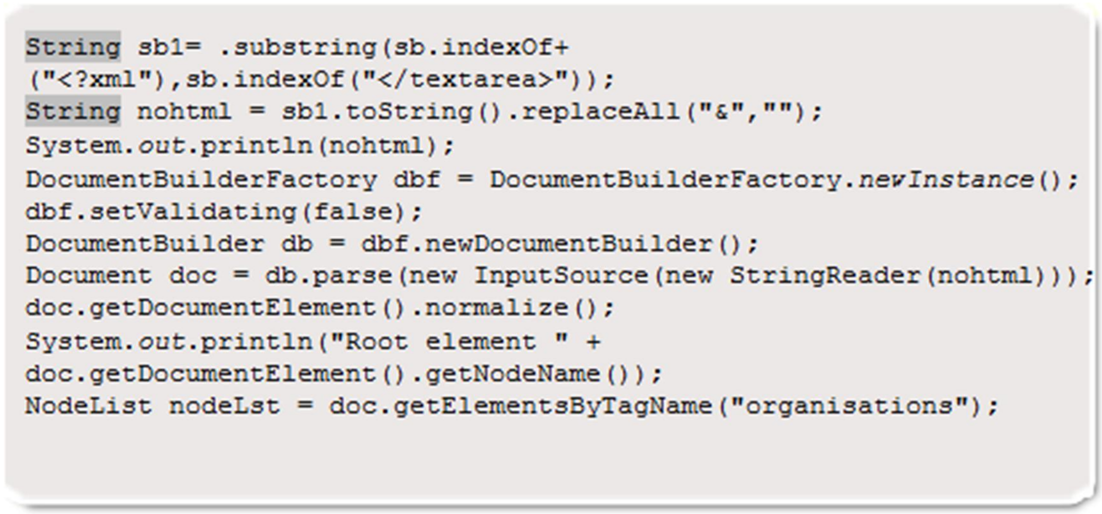
In this project the Java programming has been implemented for the following functionalities: connecting the solution to the company's server.



```
URL url = new URL("https://vw261.viope.com/cgi-
    bin/contents/eclipse_plug-in /plug-in .pl");
    URLConnection connection = url.openConnection();
    connection.setDoOutput (true);
```

Figure 33. Connecting the plug-in and the client server

As shown in Figure 33, in the Java code, the first two lines are try to connect the application with the remote server. After being connected, the server will be opened and ready to be accessed. Reading or parsing the xml files from a specific url address.



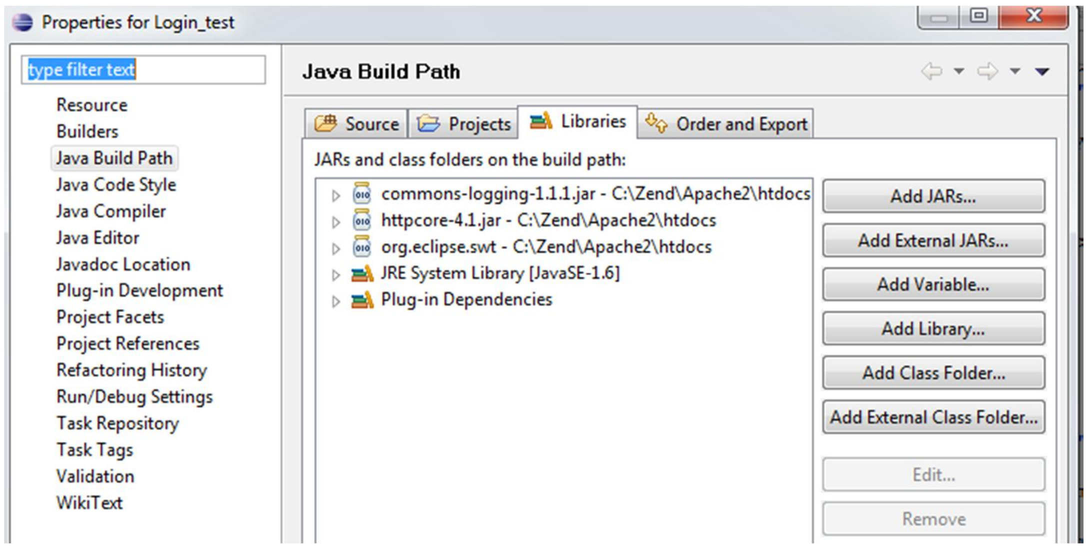
```
String sb1= .substring(sb.indexOf+
("<?xml"),sb.indexOf("</textarea>"));
String nohtml = sb1.toString().replaceAll("&","");
System.out.println(nohtml);
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setValidating(false);
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse(new InputSource(new StringReader(nohtml)));
doc.getDocumentElement().normalize();
System.out.println("Root element " +
doc.getDocumentElement().getNodeName());
NodeList nodeLst = doc.getElementsByTagName("organisations");
```

Figure 34. Reading and formatting xml file using Java programming

As illustrated in Figure 34, the first three lines are parsing and formatting the xml file into readable format. Every unwanted character will be removed or replaced by empty a space.

4.4 Other Supportive Libraries

These libraries are crucial when we come to run a Java application since they are providing runtime support to the main Java application.


Properties for Login_test

type filter text

Resource
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Plug-in Development
Project Facets
Project References
Refactoring History
Run/Debug Settings
Task Repository
Task Tags
Validation
WikiText

Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

- commons-logging-1.1.1.jar - C:\Zend\Apache2\htdocs
- httpcore-4.1.jar - C:\Zend\Apache2\htdocs
- org.eclipse.swt - C:\Zend\Apache2\htdocs
- JRE System Library [JavaSE-1.6]
- Plug-in Dependencies

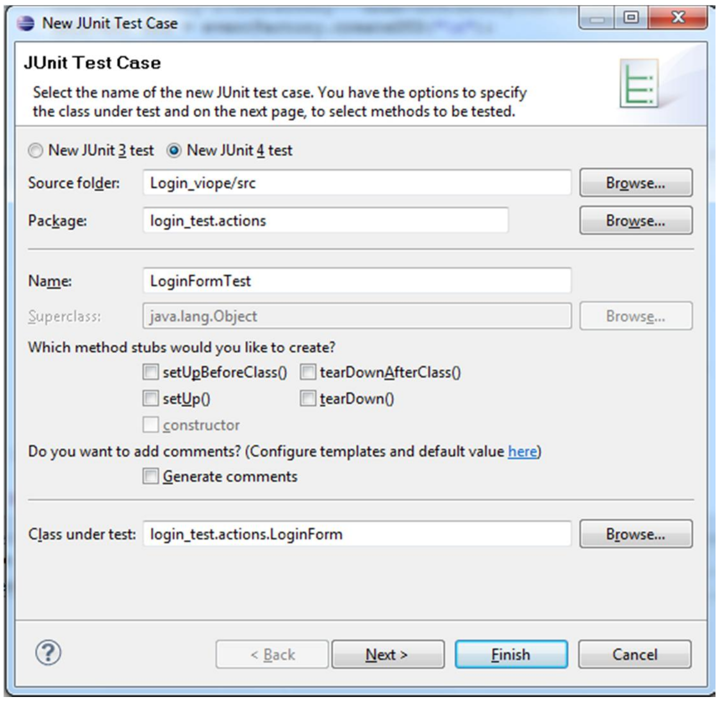
Figure 35. Project supportive Libraries

As shown in Figure 35, the Java class library has 3 (three) sets of dynamically loadable jar files. We used these libraries because Java platform is not dependent on any specific operating system. Therefore, the Java platform offers a comprehensive set of libraries that support the application to run smoothly during runtime. Even in a normal deploying process, these dependent libraries should always be moved together; otherwise the chance of application to break down will increase.

5 Testing and Deploying Solutions

5.1 Testing solution

The Eclipse platform provides an environment to test and debug applications before deployed to their final destinations. As shown in Figure 36, the plug-in is in testing environment and under testing process.


New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

New JUnit 3 test [✓ New JUnit 4 test]

Source folder:	Login_viope/src	Browse...
Package:	login_test.actions	Browse...
Name:	LoginFormTest	
Superclass:	java.lang.Object	Browse...

Which method stubs would you like to create?

☐ setUpBeforeClass() ☐ tearDownAfterClass()
☐ setUp() ☐ tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Class under test:	login_test.actions.LoginForm	Browse...

< Back | Next > | Finish | Cancel

Figure 36. JUnit test

If we run the application, the runtime workbench will appear within seconds. A new window will appear, containing Login menu. When this menu item is clicked, a login form pops up (see Figure 33) to receive the user's username and password.

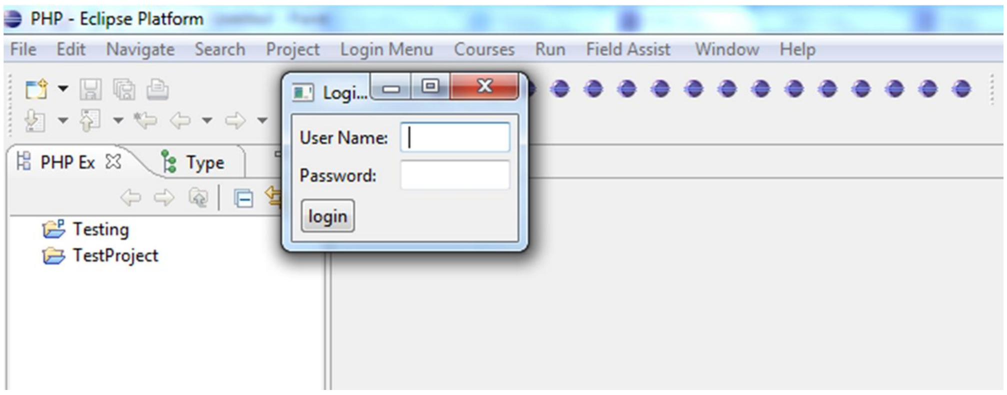

PHP - Eclipse Platform
File Edit Navigate Search Project Login Menu Courses Run Field Assist Window Help
PHP Ex
Type
Testing
TestProject
Logi...
User Name:
Password:
login

Figure 37. Testing applications

5.2 Deploying to final Destination

The last step in the development cycle is packaging and deploying the application into Eclipse based products. Therefore, packaging the readymade plug-in with suitable format such as *jar file* is common practice.

The process of packaging a project means mapping of source folder to jars. However, which files are included or excluded is determined in a key–value pair.

Steps of deployment

1. `Right Click` on the project you need to deploy

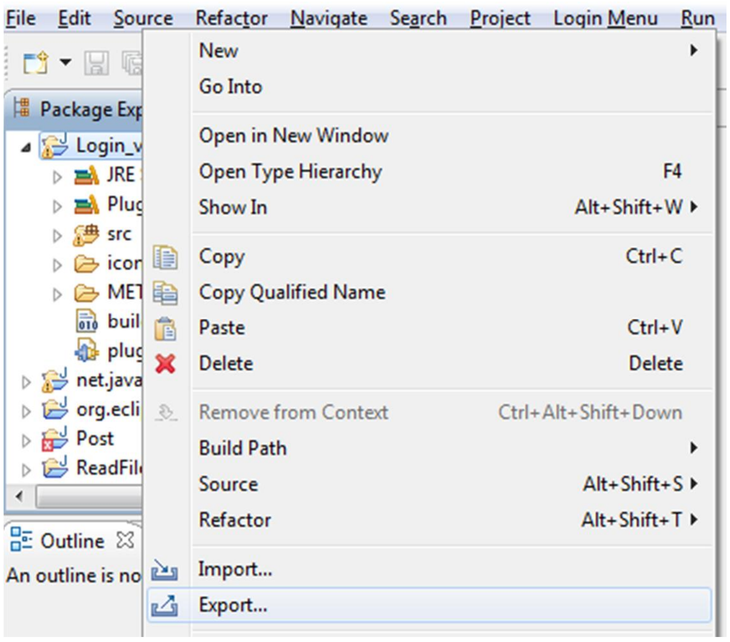


Figure 38. Exporting/deploying projects

2. Select `deployable project and fragments (`see Figure 38`)` then click on `Next`.

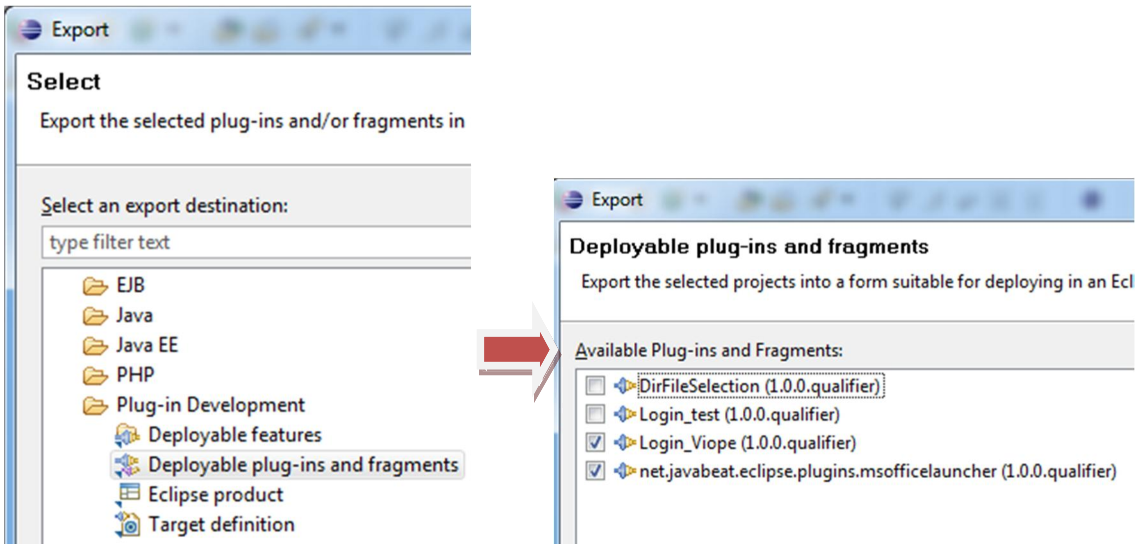

Export
Select
Export the selected plug-ins and/or fragments in
Select an export destination:
type filter text
EJB
Java
Java EE
PHP
Plug-in Development
Deployable features
Deployable plug-ins and fragments
Eclipse product
Target definition
Export
Deployable plug-ins and fragments
Export the selected projects into a form suitable for deploying in an Ecl
Available Plug-ins and Fragments:
DirFileSelection (1.0.0.qualifier)
Login_test (1.0.0.qualifier)
Login_Viope (1.0.0.qualifier)
net.javabeat.eclipse.plugins.msofficelauncher (1.0.0.qualifier)

Figure 39. Select the deployable project from the list

3. Check the deployable plug-in from the List as illustrated in Figure 39.
4. Finally, click on `finish`.
5. Copy the exported jar files and paste it in `dropin` (see Figure 40) directory inside the main installation directory.

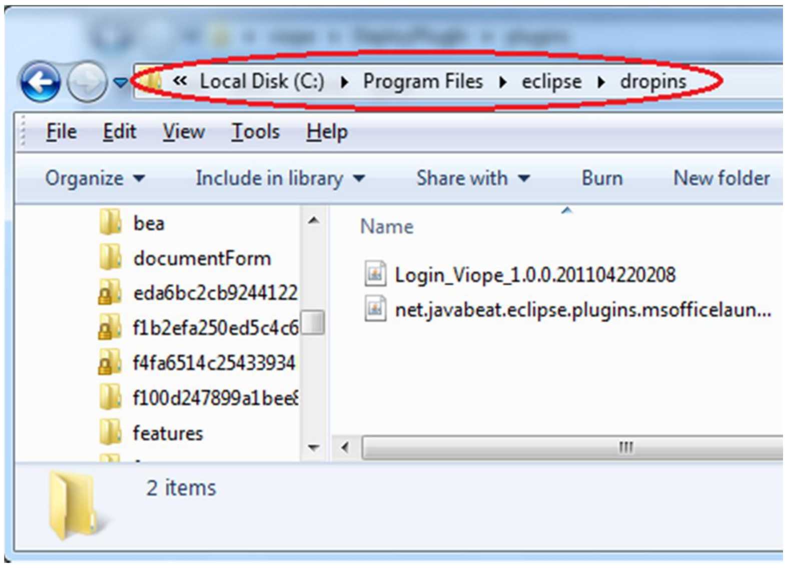

« Local Disk (C:) ▸ Program Files ▸ eclipse ▸ dropins
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Burn New folder
bea
documentForm
eda6bc2cb9244122
f1b2efa250ed5c4c6
f4fa6514c25433934
f100d247899a1bee
features
Name
Login_Viope_1.0.0.201104220208
net.javabeat.eclipse.plugins.msofficelaun...
2 items

Figure 40. Deploying the jar files inside dropin folder

At this point, the new application becomes part of the main development tool. Next we will find out the operation of the new solution associated with the existing environment. [20,302-303]

Our Solution on Action with Viope System

Once the deployment process is done, it is time to make sure the solution is performing as it is intended. The new application has to be displayed on the main development environment after the first reboot of the system. Let us see how the application is working using a real example exercise.

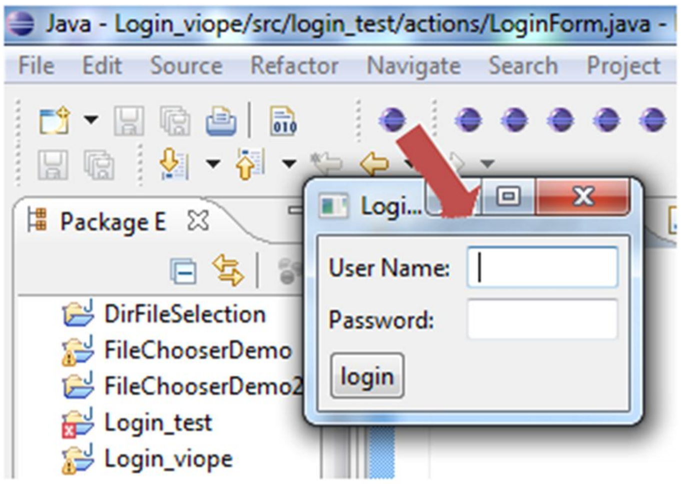
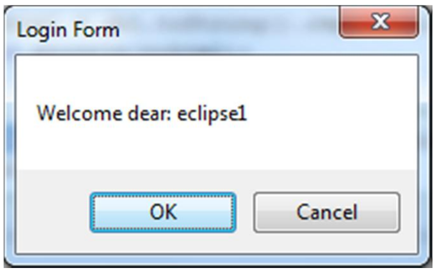


Figure 41. Login form inaction

The first thing that users need to do is to login with a registered username and password as shown in Figure 41. Successful users would be confirmed by a welcome pop up message as shown below in Figure 42. Now the users are identified with a unique session ID which enables them to stay in the application for one (1) hour.

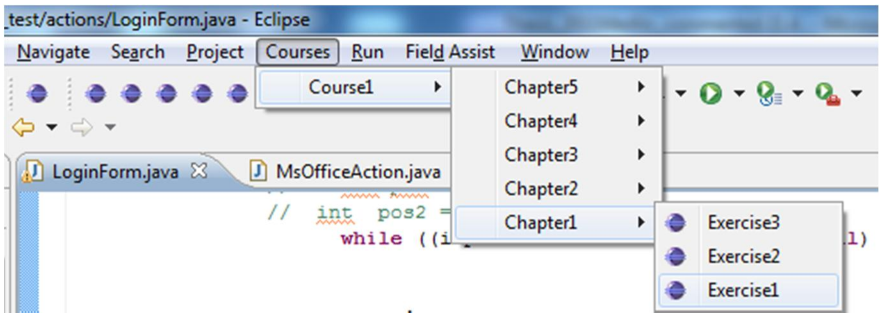

Login Form

Welcome dear: eclipse1

OK Cancel

Figure 42. Welcoming message

The purpose of accessing the chapters and exercises is to find the right IDs, which is helpful when sending the modified exercises latter.



```
// int pos2 =
   while ((i
```

Figure 43. Chapters and exercise plug-in on action

The course and chapter selector is embedded in the main tool bar as shown above in Figure 43.The real contents of the exercise can be accessed from Viope's web site. Once the student has finished working on the exercise, s/he can send the results directly to the remote server from the Eclipse editor, as shown in Figure 43.Though users have all the support of Eclipse for checking their code, they still need to send it for extra evaluation by the exercise management system.

Sample Exercise from Viope's Exercise Bank

The next step is to send an exercise solution to the company system straight from our application, as shown in the sample exercise below.

Tehtävä/Task:

1. Fixing variable definitions and initializations

Kysymys/Question:

Copy the source code below into the text box. Locate the errors in code and fix them.

Hint: What values need to be used for the given integer variable in order to give the specified output?

The program is written to a class called InvalidVariables.

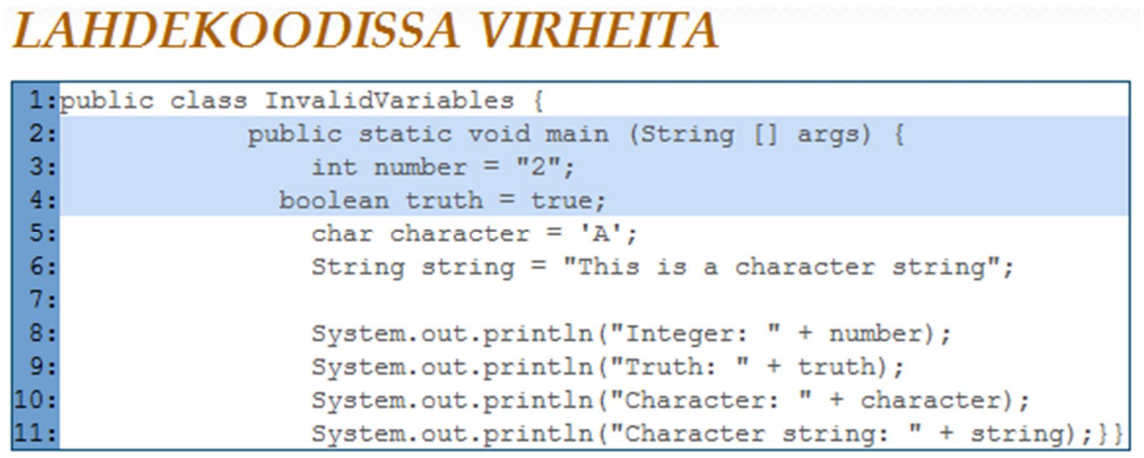
```
public class InvalidVariables {
    public static void main (String [] args) {
        int number = "2";
        boolean truth = TRUE;
        char character = "A";
        String string = 'This is a character string';
        System.out.println("Integer: " + number);
        System.out.println("Truth: " + truth);
     System.out.println("Character: " + character);
     System.out.println("Character string: " + string);
  }}
```

Figure 44. Sample exercise

Esimerkkitulostus/Output:

```
Integer: 2
Truth: true
Character: A
Character string: This is a character string
```

As shown above in Figure 44, the sample exercise code has a syntax error on line three (3), `int number = "2"`, the integer variable is quoted under quotation mark. If we send this exercise using the company's existing system, we will get an error message as shown below in Figure 45.The message indicates where exactly the error has happened.



LAHDEKOODISSA VIRHEITA

```
 1:public class InvalidVariables {
 2:            public static void main (String [] args) {
 3:                int number = "2";
 4:              boolean truth = true;
 5:                char character = 'A';
 6:                String string = "This is a character string";
 7:
 8:                System.out.println("Integer: " + number);
 9:                System.out.println("Truth: " + truth);
10:                System.out.println("Character: " + character);
11:                System.out.println("Character string: " + string);}}
```

Figure 45. Error code sent message

If we follow the same procedure using our new solution, we will find the same result. Also the error code will be notified by the IDE before it is sent to the remote system.

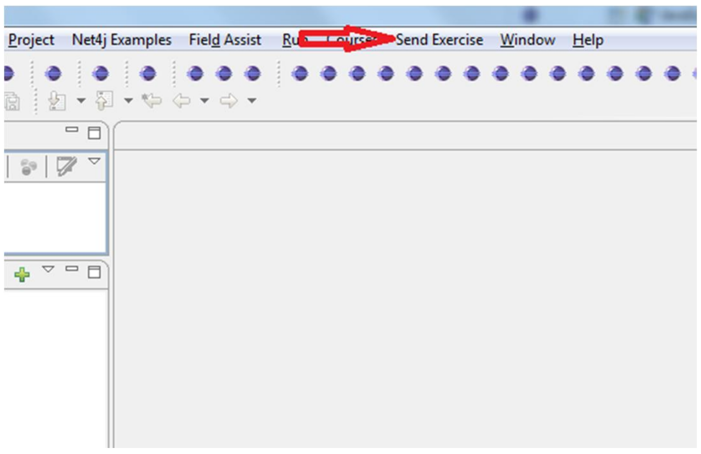


Figure 46. Sending solutions to a remote server from the local IDE

Before sending the code by using our solution, two things need to be considered. The exercise code has to be located in the project **scr** directory (eg.sample.txt), and secondly, the send button on the menu tool bar must be pressed (see Figure 46). This file browser pop ups, when users press the send button on the main editor (see Figure 47).

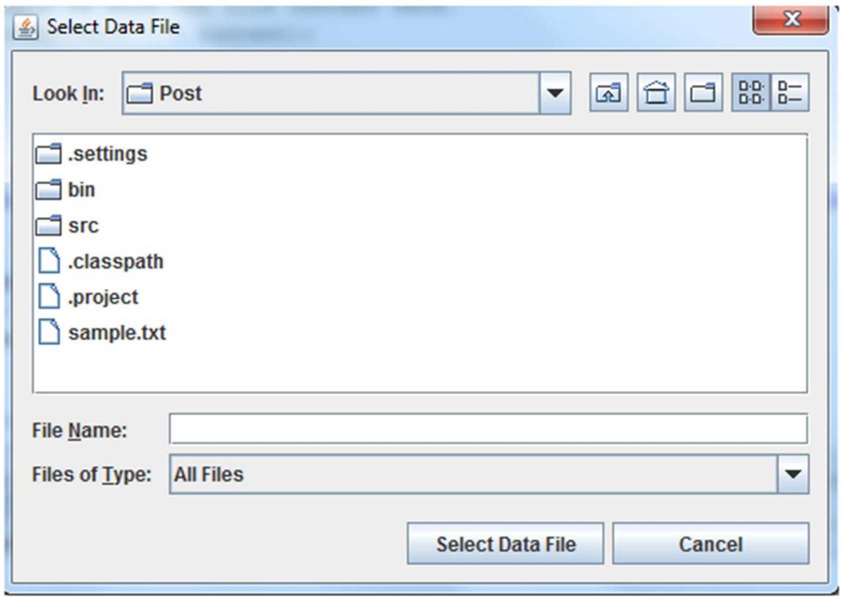
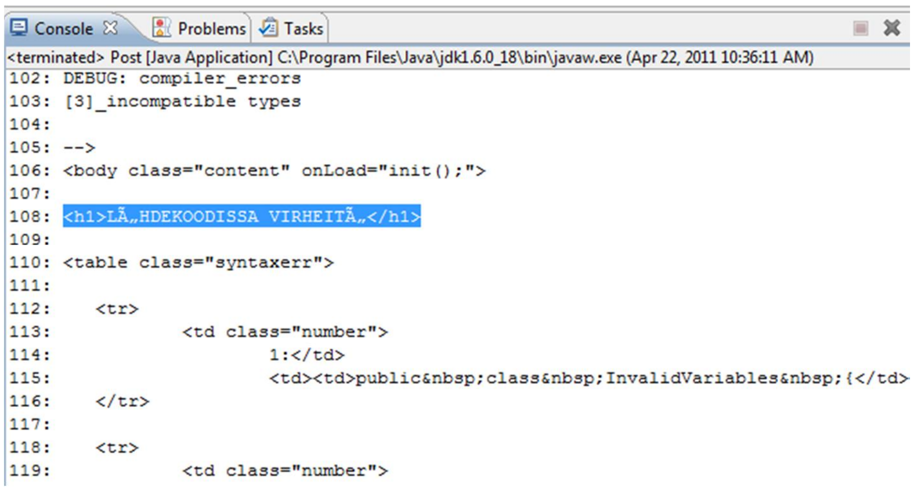


Figure 47. Java file browser

In this case the same exercise code is saved in the `sample.txt` for testing purpose. If we send this file, we will see the result in the editor console as shown in the Figure 48.

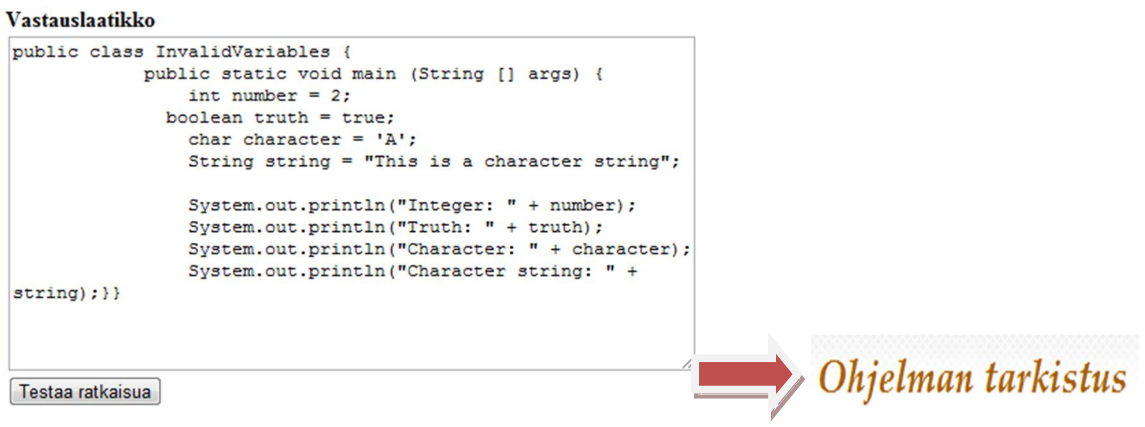


```
<terminated> Post [Java Application] C:\Program Files\Java\jdk1.6.0_18\bin\javaw.exe (Apr 22, 2011 10:36:11 AM)
102: DEBUG: compiler_errors
103: [3]_incompatible types
104:
105: -->
106: <body class="content" onLoad="init();">
107:
108: <h1>LÃ„HDEKOODISSA VIRHEITÃ„</h1>
109:
110: <table class="syntaxerr">
111:
112:    <tr>
113:            <td class="number">
114:                    1:</td>
115:                    <td><td>public&nbsp;class&nbsp;InvalidVariables&nbsp;{</td>
116:    </tr>
117:
118:    <tr>
119:            <td class="number">
```

Figure 48. Error message at local console

Sending the right solution to the system results in different feedback message in both the company's solution and in our new solution; let us examine the feedback using the client's existing system. This time the quotation of the integer variable mentioned above is removed as follows;

```
int number = 2;
```



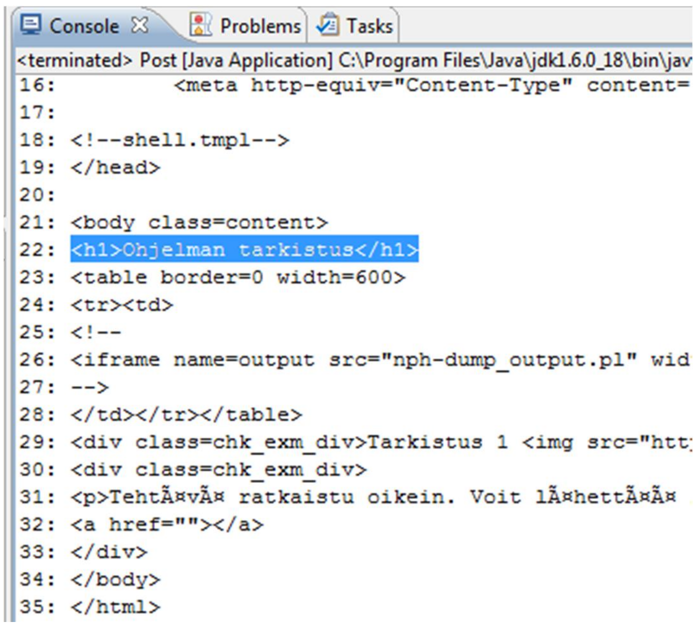
Vastauslaatikko

```
public class InvalidVariables {
            public static void main (String [] args) {
                int number = 2;
              boolean truth = true;
                char character = 'A';
                String string = "This is a character string";

                System.out.println("Integer: " + number);
                System.out.println("Truth: " + truth);
                System.out.println("Character: " + character);
                System.out.println("Character string: " +
string);}}
```

Figure 49. Right messages confirmation from remote system

If this exercise has been executed from Viope's editor as shown in Figure 49, the result is confirmed as it is correct.

The same result on the console will be generated as shown in Figure 50.This feedback is directly received from the remote server which is the one handling the exercises and responses sent by the users.



```
<terminated> Post [Java Application] C:\Program Files\Java\jdk1.6.0_18\bin\jav
16:         <meta http-equiv="Content-Type" content=
17:
18: <!--shell.tmpl-->
19: </head>
20:
21: <body class=content>
22: <h1>Ohjelman tarkistus</h1>
23: <table border=0 width=600>
24: <tr><td>
25: <!--
26: <iframe name=output src="nph-dump_output.pl" wid
27: -->
28: </td></tr></table>
29: <div class=chk_exm_div>Tarkistus 1 <img src="htt
30: <div class=chk_exm_div>
31: <p>TehtÃ¤vÃ¤ ratkaistu oikein. Voit lÃ¤hettÃ¤Ã¤
32: <a href=""></a>
33: </div>
34: </body>
35: </html>
```

Figure 50. Correction message displayed on local console.

The feedback helps the users to know their answers have met the requirement given in the question.

5.3 Licensing and related Legal issues

Eclipse is a free software, which grants users an open privilege for accessing, modifying and redistributing source codes. Eclipse Public License (EPL) is designed based on a business-friendly free software license, which enforces the developers to allow the end users to modify it without any limits. This is called a *copy left (*contrary to Copyright).

Figure 51. Left Copy right

As displayed in Figure 51, Eclipse has adapted copy left by providing exactly the opposite right to users in which the users receive a copy of work permission to reproduce, adapt or redistribute. The conditions that are stipulated under the copy left licenses are: the source code has to be available for the general public use and it must be prepared and licensed for distribution and extra derivation work for free.

The physical parties of EPL are "Contributors" and "Recipients." Contributors are parties who have created the first code or application and distributed it under the terms of EPL.A program is any application that is ready to be distributed or has already been released under same terms and conditions. Recipients or receivers are parties who have received the programs or source code under EPL [21].

6 Conclusion

Integrating Eclipse IDE with an already existing eLearing system, provides enormous benefits to the company such as: reducing the burden of dedicated server, checking syntaxes instantly and providing quick fixes for errors.

The goal of this project was to integrate the locally installed Eclipse, with Viope's platform. This ensures that the users are able to submit their course exercises from a local Eclipse source code editor.

As a result, three (3) plug-ins were developed: the Login plug-in, the Chapter Selector plug-in and the Submit plug-in. With the help of these plug-ins,both the teacher's management of exercises and the student's submission of those exercises can be successfully carried out.

The solutions developed in this study are also applicable in other integrated development environments such as in NetBeans IDE.The present solutions are also portable for different platforms.

References

1 Integrated development environment: article and information. [Online] http://www.fuhz.com/Integrated_development_environment. Accessed March 15, 2011.

2 Integrating MaxScript and .NET Systems. [Online] http://www.gamasutra.com/view/feature/3780/integrating_maxscript_and_ne t_.php?print=1. Accessed March 16, 2011.

3 Cawood S, McGee P, Microsoft XNA game studio,creators guide, Second Edition USA 2009.[Online]
http://www.microsoft.com/en-us/default.aspx. Accessed 22nd April, 2011.

4 Migrating to Eclipse, A developer's guide to evaluating. [Online] http://www.ibm.com/developerworks/library/os-ecjbuild/.Accessed 2nd April, 2011.

5 Holzner S. Essential Eclipse, Programming Java Applications, 1st Edition, publisher O'REILLY.

6 Clayberg E, Rubel D,Eclipse Building Commercial Quality Plug-ins,2nd Edition, Forwards by Skip Mcgaughey and Simon Anchor.

7 DevelopingEclipseplug-ins.[Online] http://www.ibm.com/developerworks/Java/library/os-ecplug/.
Accessed March21, 2011.

8 Bock H. The Definitive Guide to NetBeans Platform; Covers NetBeans Platform 6.5,2009

9 NetBeans Platform Runtime Containers [Online].
http://platform.netbeans.org/tutorials/nbm-runtime-container.html.Accessed 26 March, 2011.

10 Tulach J, Founder of NetBeans platform, Practical API Design, Confessions of Java Framework Architect,NewYork.

11 Myatt A, Leonard B, Wielenga G: Pro NetBeans IDE6,Rich Client Platform Edition ,New York,2008

12 NetBeans IDE 6.9, Features [online]. http://netbeans.org/features/IDE/.Accessed 2nd April, 2011.

13 Microsoft Volume Licensing, Reference Guide December 2010.

14 WesterKamp P, University of Munster; Institute fur Wirtschaftsinformatik

Leonardo-Campus 3D-48149 Munster, Germany-Learning As a web service.

15 Viope solutions OY, Viope solutions ltd founded 2011 [online]. http://www.viope.com/yritys.Accessed 2nd Feb , 2011.

16 Viope solutions OY, Student Menu [online]. https://vw26.viope.com/cgi-bin/login_logout/frameset.pl?course_id=1964. Accessed 7th Feb, 2011.

17 Viope solutions OY, Teachers Menu [online]. https://vw26.viope.com/cgi-binn/login_logout/frameset.pl?course_id=1964. Accessed 7th Feb, 2011.

18 Abott D. Embedded linux development using Eclipse, USA; 1994

19 Northover S, Wilson M; SWT, The Standard Widget Toolkit Volume1, Boston;2004.

20 Hozner S, Eclipse Cookbook, O'Reilly, USA; 2004.

21 The Eclipse Foundation Legal terms and Licensing [Online]. http://www.Eclipse.org/org/documents/epl-v10.php. Accessed, 22nd April , 2011.

22 D'Anjou J, Fairbrother S, Kehn D, Kellerman J, McCarthy P:The Java Develop-ers's Guide to Eclipse,Second Edtion,Boston,2004

Appendix

1 The Main Login form class(LoginForm.Java)

```
public class LoginForm implements IWorkbenchWindowActionDelegate {
           private IWorkbenchWindow window;
           Shell shell = new Shell();
           Label label1, label2;
           Text page;
           Text account;
           Text password;
           Text text;
           public static String sid,me;
           public String configFile;

           /**
            * The constructor.
            */
           public LoginForm() {
                      shell.setLayout(new GridLayout(2, false));
                      shell.setText("Login form");
                      label1 = new Label(shell, SWT.NULL);
                      label1.setText("User Name: ");

account = new Text(shell, SWT.SINGLE | SWT.BORDER);
                      account.setText("");
                      account.setTextLimit(30);

                      label2 = new Label(shell, SWT.NULL);
                      label2.setText("Password: ");

password = new Text(shell, SWT.SINGLE |     SWT.BORDER);
                      System.out.println(password.getEchoChar());
                      password.setEchoChar('*');
                      password.setTextLimit(30);

                      Button button = new Button(shell, SWT.PUSH);
                      button.setText("login");
button.addListener(SWT.Selection, new Listener() {
```

2 Configuration File(Plug-in .xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<?Eclipse version="3.4"?>
<plug-in >
   <extension
         point="org.Eclipse.ui.actionSets">
      <actionSet
            label="Login Action Set"
            visible="true"
            id="Login.actionSet">
         <menu
               label="Login &amp;Menu"
               id="loginMenu">
            <separator
                  name="loginGroup1">
            </separator>
         </menu>
         <action
               class="login_test.actions.LoginForm"
               icon="icons/sample.gif"
               id="login_test.actions.LoginForm"
               label="&amp;Login2"
               menubarPath="loginMenu1/loginGroup1"
               toolbarPath="loginGroup1"
               tooltip="my login viope please work">
         </action>
      </actionSet>
   </extension>

</plug-in >
```

3 Accessing Cources(Configuration XML file)

```
<?xml version="1.0" encoding="UTF-8"?>
<?Eclipse version="3.4"?>
<plug-in >

   <extension
         point="org.Eclipse.ui.actionSets">
      <actionSet
            label="MS Office Action Set"
            visible="true"
            id="net.Javabeat.Eclipse.plug-in
s.msofficelauncher.actionSet">
         <menu
               label="Courses"
               id="MsOfficeMenu">
            <separator
                  name="MsOfficeGroup">
            </separator>
         </menu>
          <menu
               id="Course1"
               label="Course1"
```

```
              path="MsOfficeMenu/MsOfficeGroup">
              <separator
                 name="MsOfficeGroup">
            </separator>
                     </menu>
                      <menu
              id="Course2"
              label="Course2"
              path="MsOfficeMenu/MsOfficeGroup">
              <separator
                 name="MsOfficeGroup">
            </separator>
                     </menu>
                      <menu
              id="Course3"
              label="Course3"
              path="MsOfficeMenu/MsOfficeGroup">
              <separator
                 name="MsOfficeGroup">
            </separator>
                     </menu>
                      <menu
              id="Course4"
              label="Course4"
              path="MsOfficeMenu/MsOfficeGroup">
              <separator
                 name="MsOfficeGroup">
            </separator>
                     </menu>
                      <menu
              id="Course5"
              label="Course5"
              path="MsOfficeMenu/MsOfficeGroup">
              <separator
                 name="MsOfficeGroup">
            </separator>
                     </menu>
```