Chapter 6: System Development

6.1 Introduction

A proper design process has been explained in the previous chapter. Now, the real implementation of developing this project has to be carried out based on the UML diagrams and designs constructed during the analysis and design stage.

In this chapter, topics covered are the system development environment, software development kit used, setup process, development process and testing process involved in realizing the OODA System for Microsoft Excel Files. Finally, a conclusion is presented to summarise and conclude the outcome of the system development process.

6.2 System Development Environment

System development platform that is chosen for this project is Microsoft Windows platform. Microsoft Excel program could be installed and used without any conflicts occurred under this platform. There are few types of Microsoft Windows platform, namely Windows 3.*, Windows 95, Windows 98, Windows Me, Windows NT, Windows 2000, Windows 2000 Advanced Server and Windows XP.

Among these platforms, Microsoft Windows XP Professional Edition is chosen for developing this project. The reason for choosing Microsoft Windows XP Professional Edition is the stability and performance of the operating system. Microsoft Windows XP

Professional Edition has a better improvement on its stability and performance compared to previous version of Windows O/S such as Windows 98 and Windows 95. It is proven by the Microsoft Corporation (Microsoft's Challenge: Competing With Itself, 2003).

Besides that, Microsoft Excel version 2000 is needed for completing this project. The dynamic link library (DLL) for Microsoft Excel program has to be placed in the windows system directory. The DLL has to be used by the java classes that were designed properly during the O-O design stage for manipulating Excel files.

6.3 Software Development Kit

It is important to use appropriate tools before starting any system development. For developing this project, Java 2 SDK (J2SDK) provided by Sun Microsystems has been chosen. J2SDK includes useful tools for developing and testing programs which are written in Java programming language and run on Java platform. These tools are designed to be used from the command line. There is no graphical user interface provided for using these tools. A proper installation of J2SDK has to be carried out in order to utilise these tools. Section 6.3.2 explains the installation instructions. However, a detailed checking on the system requirements for utilising J2SDK must be carried out first for deciding whether the current system platform can be used or some upgrades need to be done. Section 6.3.1 below will first explain the system requirements for J2SDK.

6.3.1 System Requirements

In order to be able to use the J2SDK tools (windows version), a machine must be equipped with either one of the following operating systems. These operating systems are shown in Table 6.1.

Table 6.1 Operating Systems accepted by J2SDK

No	Operating System Accepted
1	Microsoft Windows 95
2	Microsoft Windows 98 (1st or 2nd Edition)
3	NT 4.0 with Service Pack 6
4	Windows ME
5	Windows 2000 Professional
6	Windows 2000 Server
7	Windows 2000 Advanced Server
8	Windows XP Operating Systems running on Intel hardware

Besides that, there are certainly some minimum hardware requirements that must be followed in order to utilise the J2SDK tools. Generally, a Pentium 166MHz or faster processor is needed for better system performance. A 32MB of Random Access Memory (RAM) is needed as well. Finally, there must be at least 70 MB of hard disk space for developing a J2SDK complaint system.

6.3.2 J2SDK Installation Instructions

In order to install J2SDK, a self-installing executable program has to be run to unpack the J2SDK bundle. J2SDK can be downloaded from the Java Sun Microsystems's website at http://java.sun.com. Figure 6.1 below shows the file structure of J2SDK after downloaded from the website.

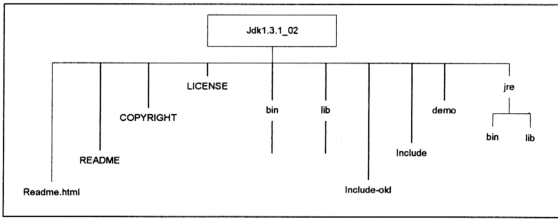

Figure 6.1 J2SDK File Structure

After executing the exe file (j2sdk-1_3_1_02-windows-i586.exe), J2SDK has been installed properly in the machine. Reboot system is required in order to complete windows systems registry modifications. A process of setting the environment variable is required in order to enable the use of the J2SDK tools under any physical file directories. Use the command showed below for setting the environment variable.

```
Set PATH=%PATH%;C:\JDK131\BIN
```

The above command should be written in a system file called autoexec.bat.

When all of these actions were done accordingly, the system development process proceeds to the setting of a web server tool called Tomcat.

6.4 Setup Process

As the installation of the J2SDK tools has been done completely, the next step would be setting up an interface engine by utilising a server tool called Tomcat. The purpose of choosing Tomcat is to be able to construct the Java Servlet pages. These Java Servlet pages act as the interface layer objects which are responsible for the interaction between the OODA System for Microsoft Excel Files and its users.

Tomcat is a Java Servlet container and the JavaServer Pages (JSP) implementation. It may be used stand alone, or in conjunction with several popular web servers such as Apache, version 1.3 or later, Microsoft Internet Information Server version 4.0 or later and Microsoft Personal Web Server version 4.0 or later. It needs a Java Runtime Environment (JRE) version 1.1 or later. Tomcat also requires a Java compiler, which is already provided by the J2SDK tools. For this project, Tomcat developed by Apache Software Foundation is chosen and the installation process is described in section 6.4.1.

6.4.1 Tomcat Server Setup Process

Firstly, Tomcat that is a free open source program could be downloaded from the Apache Software Foundation website at http://jakarta.apache.org/. The downloaded file is in the format of zip or tar.gz. Table 6.2 explains the steps involved when setting up the Tomcat server.

Table 6.2 Tomcat Server Installation Steps

No.	Steps
1	Unzip the file into a directory (say foo). This should create a new subdirecotry named "jakarta-tomcat-3.3.1". Move this subdirectory to the root directory (say c drive) and named it tomcat33.
2	Set a new environment variable called TOMCAT_HOME to point to the root directory of the tomcat. On Win32, type "set TOMCAT_HOME=c:\tomcat33" in the autoexec.bat file.
3	Set the environment variable JAVA_HOME to point to the root directory of the JDK tools, then add the JDK131\bin\tools.jar and JDK131\bin\rt.jar to the PATH environment variable. All these steps are done on the autoexec.bat file (Win32 system).

As all of the steps mentioned in Table 6.2 are carried out successfully, the Tomcat Server can now be used for executing the Java Servlet pages. For ensuring this, go to the command prompt and change the directory to c:\tomcat33\bin. This step is shown in Figure 6.2.

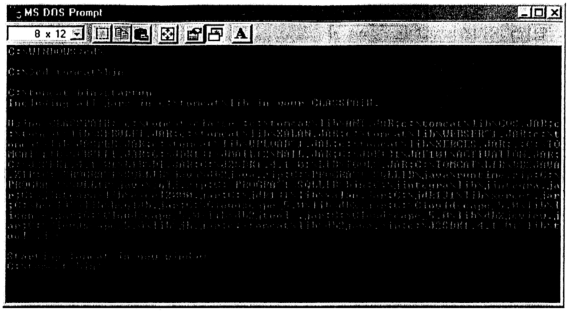

Figure 6.2 Tomcat Start up command

The startup command run under the c:\tomcat33\bin directory will automatically invoke the Tomcat Server application. If the command window shown in Figure 6.3 pop up automatically, then it gives a sign showing that the Tomcat Server has been installed successfully.

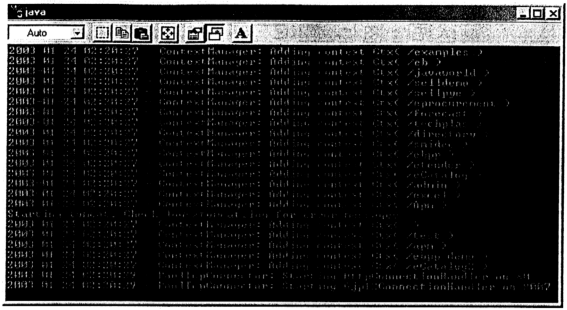

Figure 6.3 Tomcat Start Up Successfully

To shut down the Tomcat Server application, just go back to the command prompt window, change the directory to c:\tomcat33\bin again and type shutdown. This should be able to completely terminate the Tomcat Server application. The shutdown step is shown in Figure 6.4.

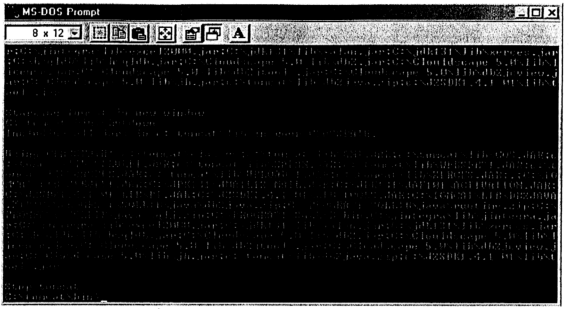

Figure 6.4 Tomcat Shutdown command

In order to ensure that the Tomcat Server could be used to execute Java Servlet pages, the following steps need to be carried out. These steps are shown in Table 6.3.

Table 6.3 Testing Java Servlet Pages

No.	Steps
1	Go to command prompt and change directory to c:\tomcat33\bin
2	Type startup at the command prompt
3	Open an internet browser application (Internet Explorer or Netscape Communicator) and go to http://localhost/index.html.
4	Click on the servlet examples link and execute few examples of servlet pages provided.
5	Type shutdown at the command prompt to stop the Tomcat Server application.

After all the above steps have been carried out and tested with a few of the Java Servlet pages, the Tomcat Server is ready for the real system development of this project.

Figure 6.5 shown below is the first screen seen when reaching http://localhost/index.html using Microsoft Internet Explorer.

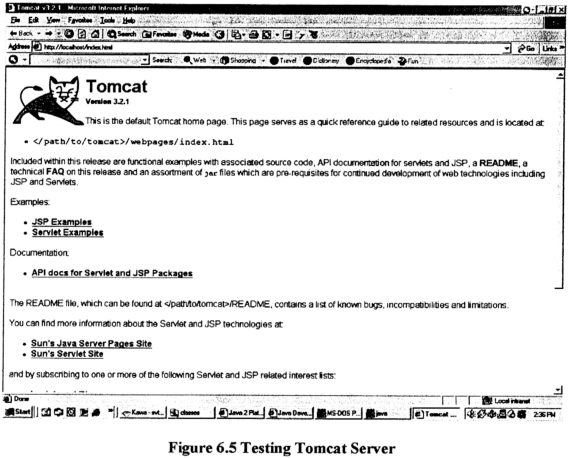

Figure 6.5 Testing Tomcat Server

6.5 Development Process

After setting up the JDK tools and the Tomcat Server, the development process has now reached the coding process. In order to fasten the development process, a Java program text editor has been chosen to write the Java program. The text editor application chosen was Kawa version 3.51.

The Kawa version 3.51 is used for constructing Java files (with the extension *.java). The Java files are then compiled by using the JDK1.3.1 tools. Hence, the class files (with the extension *.class) are produced. These class files are then need to be placed in the TOMCAT_HOME\webapps (TOMCAT_HOME is the environment variables that shows the physical directory of Tomcat application server) directory correctly in order to be invoked while processing the Java Servlet pages. Figure 6.6 shows the processes involved in developing this project.

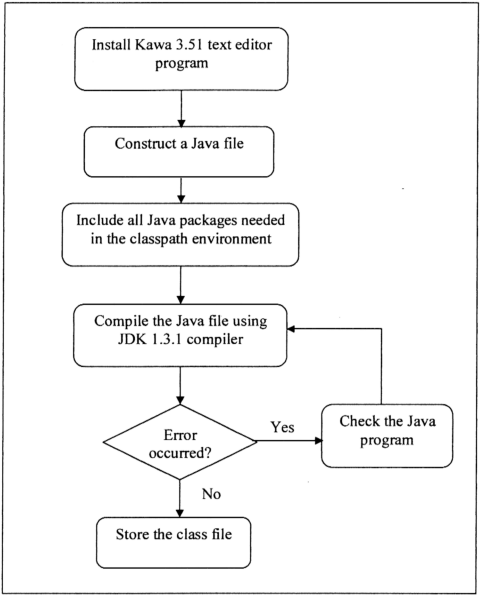

Figure 6.6 Process Involved in Creating Java file and Class file

Next, the Java Servlet pages have to be created as well. The Java Servlet pages act as the interface layer objects for the interaction between OODA system for Microsoft Excel Files and the system users. The differences between a Java Servlet page development and the normal Java program development are shown in Table 6.4.

Table 6.4 Differences between a Java Servlet page and a normal Java program

Java Servlet	Normal Java program
Import javax.servlet.* package	Doesn't need to import javax.servlet.* package
Import javax.servlet.http.* package	Doesn't need to import javax.servlet.http.* package
Object extends HttpServlet class	Never extends HttpServlet class
doGet(HttpServletRequest req, HttpServletResponse res) function has to be created	Doesn't need doGet function
dPost(HttpServletRequest req, HttpServletResponse res) function has to be created	Doesn't need doPost function
Use HttpServletResponse instance to get the PrintWriter object for writing output	Use System.out.println() function for writing output

The JDK1.3.1 compiler which has been installed earlier is needed in the process of compiling the Java Servlet pages. This is the similarity characteristics between a Java Servlet page and a normal Java coding page.

Finally, all class files generated after compilations have to be placed in a proper directory so that the Java Servlet Pages could be executed successfully. The Java Servlet container, Tomcat Server must be configured properly.

Figure 6.7 shows the process of creating a proper directory to place the class files. Table 6.5 explains the configuration involved in order to create a virtual path to access the Java Servlet pages from the Internet browser.

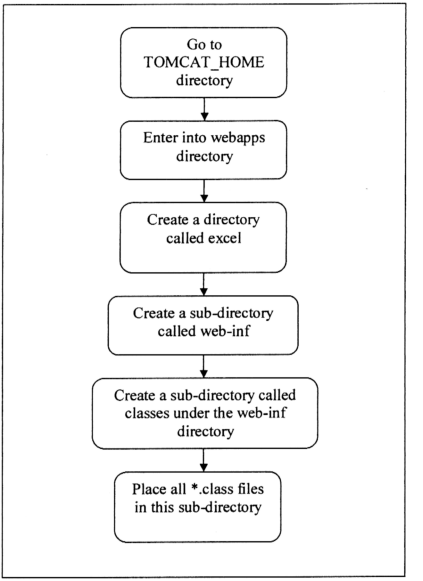

Figure 6.7 Creating Directory in Tomcat Servlet Application

Table 6.5 Tomcat Configuration Steps

No	Steps
1	Go to TOMCAT_HOME\conf directory
2	Search for server.xml file and open it for editing

3	Add the XML tag showed below in the server.xml file
	
	`<Context path="/excel"`
	`docBase="webapps/excel"`
	`crossContext="true"`
	`debug="0"`
	`reloadable="true"`
	`trusted="false" >`
	`</Context>`
	
	These XML statements have to be added within the <ContextManager> tag.

4	Save the server.xml file

6.6 Java Codes Testing Process

Each compiled Java page (either Java Servlet or the normal java page) has to be tested to ensure the correctness of the system functions and minimize program errors. To begin the testing process, Tomcat Server has to be started first. Open a command prompt window and go to the TOMCAT_HOME directory. Type startup at the command prompt and wait for the invocation of the Tomcat Server application. Once the server was started up successfully, the Java Servlet pages can be executed.

Open Internet Explorer and type the URL http://localhost/excel/servlet/svtSeparateExcel in the location bar. This will present the first page for the entry point of segregating an Excel file. Test with an Excel file and segregate it to different portions by entering the exact phrases found in the file.

Once the Excel segregation part has been tested successfully, continue the testing process by typing the URL http://localhost/excel/servlet/svtCombineExcel in the location bar. Test with combining several Excel files either horizontally or vertically.

6.7 Conclusion

As the conclusion of this chapter, the most important step in system development is the testing on each of the Java codes. This is to ensure that the functionality of each Java object developed meet the requirements set during the system analysis and design stage that were discussed previously.

In the next chapter, an overall explanation of testing for this project will be presented.