A Comparative Study of Web Application Design Models Using the Java Technologies

Budi Kurniawan and Jingling Xue

School of Computer Science and Engineering
University of New South Wales
Sydney, NSW 2052, Australia

Abstract. The Servlet technology has been the most widely used technology for building scalable Web applications. In the events, there are four design models for developing Web applications using the Java technologies: Model 1, Model 2, Struts, and JavaServer Faces (JSF). Model 1 employs a series of JSP pages; Model 2 adopts the Model-View-Controller pattern; Struts is a framework employing the Model 2 design model; and JSF is a new technology that supports ready-to-use components for rapid Web application development. Model 1 is not recommended for medium-sized and large applications as it introduces maintenance nightmare. This paper compares and evaluates the ease of application development and the performance of the three design models (Model 2, Struts, and JSF) by building three versions of an online store application using each of the three design models, respectively.

1 Introduction

Today, Web applications are the most common applications for presenting dynamic contents. There are a number of technologies for building Web applications, the most popular of which is the Servlet technology [5]. This technology gains its popularity from its superiority over other technologies such as CGI and PHP [2], [3], [13].

Servlets are cumbersome to develop, however, because sending HTML tags requires the programmer to compose them into a String object and send this object to the browser. Also, a minor change to the output requires the servlet to be recompiled. To address this issue, Sun Microsystems invented JavaServer Pages (JSP) [4]. JSP allows HTML tags to be intertwined with Java code and each page is translated into a servlet. A JSP page is a servlet. However, compilation occurs automatically when the page is first requested. As a result, changing the output does not need recompilation. In addition, JSP enables the separation of presentation from the business logic through the use of JavaBeans and custom tag libraries. The norm now in developing Java-based Web applications is to use servlets along with JavaServer Pages.

In the later development, there are a number of design models for building servlet/JSP applications: Model 1, Model 2, Struts [12], and JSF [6]. Model 1 and Model 2 were first mentioned in the early specifications of JSP. Model 1 strictly uses JSP pages, with no servlets, and Model 2 uses the combination of both servlets and JSP pages. The terms of Model 1 and Model 2 have been used ever since. Model 1 is

J.X. Yu, X. Lin, H. Lu, and Y. Zhang (Eds.): APWeb 2004, LNCS 3007, pp. 711–721, 2004.

suitable for prototypes and very small applications, and Model 2 is the recommended design model for medium sized and large applications.

As Model 2 gained more acceptances in the industry, an open source initiative to build the Struts Framework was initiated. Struts perfects Model 2 by providing the controller part of the Model-View-Controller of Model 2. In addition, Struts provides better page navigation management and several custom tag libraries for more rapid development. Despite its steep learning curve and the fact that it was never defined in any specification, Struts has been gaining popularity as the alternative to Model 2.

JavaServer Faces [6] is built under the Java Community Process under JSR-127. Sun Microsystems proposed this technology in the hope that JSF will be the ultimate model for building Java Web applications. The most important feature of JSF is the availability of ready-to-use components such as extensible UI components, easy page navigation, input validators, data converters and JavaBeans management.

The problem facing servlet/JSP programmers are to choose the most appropriate design model. Clearly, JSF provides a better solution in regard to development time. However, some people are not sanguine to adopt this technology for fear of performance penalty due to the overhead of the JSF implementation.

We build three versions of an online store application named BuyDirect using Model 2, Struts and JSF. The parameters compared are the number of lines of code, the number of classes, and the performance measurement results. We investigate which of the design models allows the most rapid development process. We evaluate the performances of the applications built upon these models. We provide some suggestions to perfect the existing design models to make development more rapid.

The rest of the paper is organised as follows. Section 2 discusses the issues in Web development. Section 3 explains how the three design models address these development issues. Section 4 provides the details of the hardware and software used in these experiments. Section 5 presents the experiment results and analysis. Section 6 reviews the related work. Section 7 concludes by offering some suggestions to improve the existing design models.

2 Java Web Development Issues

All Java Web development uses the Servlet technology as the underlying technology. As such, all Java Web applications have certain issues that need to be addressed:

- **User Interface.** The user interface is what the client browser renders as HTML tags. Any server-side component used in the application must be encoded into the corresponding HTML elements. Besides for displaying the content and data, the user interface is also responsible in receiving input from the user.
- **Input Validation.** User input needs to be validated. There are two types of input validation, server-side and client-side. As the name implies, the server-side input validation is performed on the server after the input reaches the server. Client-side input validation is done on the browser, usually by using JavaScript or other scripting languages. The advantages of using client-side input validation are prompt response and reducing the server workload. The server-side input validation should always be performed regardless the presence of client-side validation because there is no guarantee the user browser's scripting feature is being on and malicious users can easily work around client-side validation.

- **Model Objects.** Model objects in Java-based Web applications are in the forms of JavaBeans. Model objects make up the Model part of the MVC based design model. A model object can be used to bind a component value to be used at a later stage. In addition, it can encapsulate business logic required for processing.
- **Page Navigation.** Almost all Web applications have multiple pages that the user can navigate from one to another. All MVC-based design models use a servlet as the Controller part. This servlet also acts as the sole entry point to the application. Which page to be displayed after the current request is determined by the value of a specified request parameter. Managing page navigation is critically important.

3 Web Application Design Models

The Model 2 design model is based on the Model-View-Controller (MVC) design pattern. As explained by Burbeck [1], there are three main modules in MVC, the Controller, the View, and the Model. The Controller acts as the central entry point to the application. All user interactions go through this controller. The View contains the presentation part of the application, and the Model stores data or encapsulates business logic of the application. In the later development, the Struts Framework provides a common framework to easily build Model 2 applications. Then, the last initiative is the JavaServer Faces, which also employs the MVC design pattern.

In the following sections, we discuss these three design models and explain how each design model addresses the development issues specified in the previous section.

3.1 Model 2

A Java Web application that is based on the Model 2 design model has one servlet (called the Controller servlet) that serves as the Controller part. All requests are first handled by this servlet, which immediately dispatches the requests to the appropriate views using RequestDispatcher objects. Views in the Model 2 design model are represented by JSP pages. To store data, a Model 2 application uses JavaBeans, which are the Model part of the application. In addition to storing data, the JavaBeans also encapsulate business logic. Each HTTP request carries an action parameter that indicates which view to dispatch this request to. The programmer must code the HTML tags for user interface in all JSP pages in the application and write input validation code. In addition, the model objects are managed by individual JSP pages.

3.2 Struts

The Struts Framework is an improvement of the Model 2 design model. It provides a default Controller servlet so that the user does not have to write and compile one. Struts alleviates the task of page navigation by allowing navigation rules to be present in its application configuration file (an XML document). Changes to the navigation rules do not require recompilation of a Java servlet class. In addition to easier page navigation, Struts provides custom tag libraries that define tags representing HTML elements. One of these tags is used for error handling and Struts is therefore capable

of displaying localized error messages in support for internationalization. Struts applications use JavaBeans as their models, just like the Model 2 design model. In addition, Struts programmers have to write their own input validation code.

3.3 JSF

JSF also employs a controller servlet that is called FacesServlet. This servlet is the only entry point to a JSF application. JSF also uses JSP pages as its views and JavaBeans as its model objects. Unlike Model 2 and Struts, however, JSF provides ready-to-use user interface components that can be written on JSP pages. Upon an invocation of a page of a JSF application, the FacesServlet constructs a component tree that represents the JSP page being requested. Some of the components can also trigger events, making JSF event-driven. For page navigation, JSF uses an approach similar to Struts, i.e., by allowing navigation rules to be defined in an application configuration file (again, an XML document).

What distinguishes a JSF application from non-JSF servlet/JSP application is that JSF applications are event-driven. The user interface of a JSF application is one or many JSP pages that host Web components such as forms and input boxes. These components are represented by JSF custom tags and can hold data. A component can be nested inside another, and it is possible to draw a tree of components. Just as in normal servlet/JSP applications, you use JavaBeans to store the data the user entered.

4 Experimental Setup

The software and hardware details for our experiments are described below.

4.1 The Online Store Application

The online store application in this research comes in three versions: Model 2, Struts, and JSF. All of them are named BuyDirect, an online store that sells electronics goods. The application has the following features:

- Search for certain products based on product names or descriptions.
- Browse the list of products by category.
- View a product's details
- Put a product into the shopping cart.
- View the shopping cart
- Check out and place an order.

This application represents the most common Web application that provides the following functionality:

- searching for certain information in the database
- browsing the data in the database,
- performing database transactions.

Data is stored in a MySQL database. The tables used and the relationship among them are depicted in Figure 1.

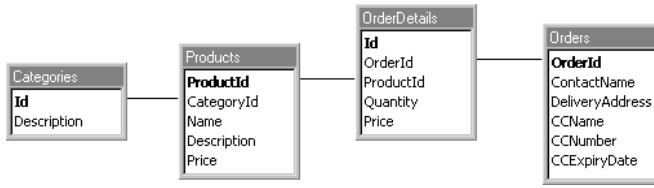

Fig. 1. The tables and relationships among them

4.2 The Servlet Container

A Java Web application runs in a servlet container, which is the engine that processes the incoming HTTP requests for the resources in the application. For this research project, we use Tomcat, an open source servlet container from the Apache Software Foundation. The version we use is 5.02 [11].

Basically, a servlet container processes a servlet by performing the following tasks:

- Creating the HttpRequest Object
- Creating the HttpResponse Object
- Calling the service method of the Servlet interface, passing the HttpRequest and HttpResponse objects.

4.3 Testing Clients

For performance testing, we emulate multiple users using JMeter 1.9 [9], also from the Apache Software Foundation. JMeter allows the user to choose the number of threads to perform testing. Each thread emulates a different user. JMeter also lets us choose how many times a test will be done. To test a Web application using JMeter, you direct requests to certain IP address, context path, and port number. You can also specify request parameters to be included in each HTTP request. As the output, JMeter notifies the response time of the server in milliseconds for a test. From the response time, we derive the number of hits/seconds the server is capable of serving.

4.4 Hardware

We use different computers for running the applications and for testing, so as to obtain maximum performance measurement accuracy. The computer running the application is a Linux machine having the following hardware specifications: Intel Pentium III 750MHz CPU with 256MB RAM. The computer running the testing clients is a Windows 2000 machine running JMeter. The computer has the following specifications: Intel Pentium III 850MHz CPU with 256MB RAM.

5 Experimental Results

We obtain experimental results in two categories: the ease of development and performance. The ease of development category compares the number of classes and the number of lines of code. These numbers indicate how easy it is to develop an application by following a certain design model. An application with the fewer number of classes or the number of lines of code indicates that the application is relatively easier to build. The application with the more number of classes indicates that the application takes more time to develop.

The performance measurement results are obtained by comparing three operations in each version of the online store application: Search, Browse, and Shopping. The Search operation is the most common operation in such an application. The Browse operation displays products by category, and the Shopping operation is the most complex operation of all. It includes filling in the Order form and inserting products in the shopping cart to the database. The database is locked during the product insertion, so either all shopping items are stored in the database or none of them is.

5.1 Ease of Application Development

As Table 1 shows, it takes the most effort to implement the Model 2 design model. Using Struts alleviates the problem a bit, and the best saving in the development comes if one uses JSF.

Table 1. The number of classes and the number of lines for the applications under study

		Model 2	Struts	JSF
Servlet	#Classes	1	0	0
	#Lines	74	0	0
Bean	#Classes	9	9	9
	#Lines	348	348	348
JSP	#Classes	9	9	9
	#Lines	809	733	534
Others	#Classes	12	10	3
	#Lines	590	430	271
Total	#Classes	31	28	21
	#Lines	1821	1511	1153

The Model 2 design model is characterised by the presence of a Controller servlet and a number of JavaBeans classes (as the Model) and JSP pages (as the Views). The Controller servlet is responsible for page navigation rules that employ a series of if statements. Model 2 application programmers must also code for the input validation that in this research is implemented inside a number of custom tag libraries. The other classes in the Model 2 design model are custom tag library and the tag library descriptors responsible for input validation and data display. In fact, input validation takes 590 lines of code, or almost 30% of the total amount of code.

In the Struts application, the Controller servlet is provided by the framework, therefore a Struts programmer saves time for not having to write one. However, he/she still needs to write page navigation rules in the Application Configuration file, which is easier than writing a servlet because the Application Configuration file can

be edited using a text editor and no compilation is necessary. Input validation must still be done manually, even though the Struts Framework provides an error handling mechanism. The number of classes and the number of lines of code for input validation are almost similar to the Model 2 application. In Struts, the other classes are Action classes to which the default Controller servlet dispatches requests.

In JSF input validation comes free through the availability of validator component. As a result, a JSF application developer can skip this task. In addition, page navigation takes the same course as Struts, i.e. by utilising an Application Configuration file. The other classes in JSF are a ContextListener, an ActionListener, and a Database utility class.

5.2 Performance Measurement

For each operation, we measure the server response time (in milliseconds) for 1 to 10 concurrent users. The number of users is specified by setting the number of threads in Jmeter. Each test is conducted 10 times and the average is taken. Each operation is discussed further is the following sub-sections.

5.2.1 Search Operation

The Search operation retrieves all products whose name or description matches the keyword. There is one SQL SELECT statement performed. Figure 2 compares the three versions of applications for the Search operation.

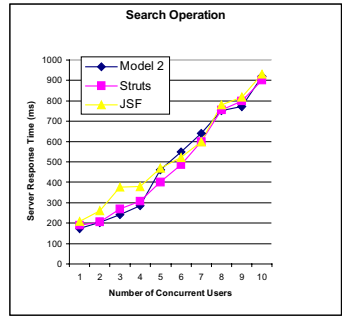

Fig. 2. The performance comparison for the Search operation

For the Model 2 application, the average server response time for one user is 173 ms and for 10 users is 919 ms. For the Struts application, these numbers are 189 ms and 900 ms, respectively. For the application built using JSF, the average server response time is 210 ms for one user and 932ms for 10 users. The increase of the response time is proportional to the increase of the number of concurrent users, which means that the server is still able to cope with the load.

The Model 2 application has the least overhead, therefore the average performance should be better than the Struts and JSF applications. However, the Struts application performs as well as the Model 2 application. This is because the server has enough memory to load all Struts libraries required to run Struts. Also, note that page navigation rules in Struts are loaded and stored in an object called ActionMapping.

Therefore, given an action request parameter, the next page of navigation is obtained through a look-up. On the other hand, the Model 2 application uses a series of if statements to find the next page of navigation, given the action request parameter.

The JSF application performs slightly worse than the other applications in almost all numbers of concurrent users. This could be due to the time taken by the JSF implementation to construct a component tree for each page requested. However, the difference in server response time between JSF and other applications is not that significant.

5.2.2 Browse Operation

The Browse operation retrieves all products belonging to the specified category for the three versions of applications. Like the Search operation, there is one SQL SELECT statement performed. Figure 3 gives the test results for this operation.

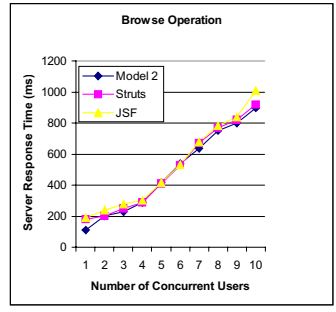

Fig. 3. The performance comparison for the Browse operation

On average, the Model 2 application performs the best because it has the least overhead. The average server response time is 111 ms for one user and 899 ms for 10 users. The Struts application has comparable performance, with one user average server response time of 180 ms and 10 user response time of 920 ms. The JSF lacks a bit behind the two applications with these numbers being 190 and 1009 ms respectively.

The increase of the server response time is proportional to the increase of the number of concurrent users, which means the server is able to serve those users well.

The average performance measurement results of the Browse operation are very similar to the ones for the Search operation because the database operations of both operations are also similar.

5.2.3 Shopping Operation

This operation includes a database transaction with an insert into the Orders table and multiple inserts into the OrderDetails table. The transaction either succeeds or fails as a whole. Figure 4 shows the test results for this operation.

The Model 2 application results in an average server response time of 230 ms for one user and 2088 ms for 10 users. The Struts application scores similar results with 238 ms and 2033 ms for both one user and 10 concurrent users. The JSF application takes an average of 240 ms to server one user and 2227 ms for 10 concurrent users.

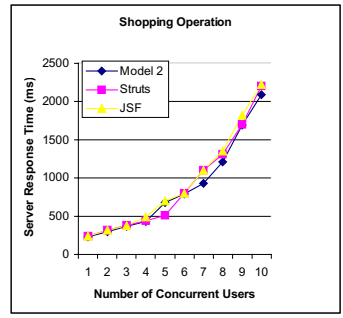

Fig. 4. The performance comparison for the Shopping operation

Figure 4 shows that in all applications, a linear increase in the number of concurrent users causes an almost exponential increase in the average server response time. This is due to the lock in the database during the database transaction that causes subsequent requests to be queued until the database lock is released.

Performance comparison for the Model 2, Struts, and JSF applications for the Shopping operation is almost the same as the Search and Browse operations. Model 2 and Struts perform similarly, while the JSF application is worse. However, the difference between the JSF application and the other two is not significant.

6 Related Work

Wu *et al* [13] compare the performance of database-based Web applications using Java servlets, PHP version 3, and Common Gateway Interface (CGI). After a series of benchmark tests that performs data retrieval from a MySQL database, they find that the solution of Java servlets with persistent database connection has the best performance. PHP3 using persistent database connections performs fairly well when compared to the CGI solution. They also mention the advantages of using Java servlets. According to these authors. Java servlets are an excellent choice to meet the requirement of e-commerce (such as online shopping) applications and are able to handle client requests in a highly interactive mode. However, Wu et al. do not provide analysis of the architectures of the system they are testing. Nor do they study the ease of development and ease of maintenance aspects of those technologies.

Cecchet *et al* [2] conduct similar research, this time comparing PHP 4, Java servlets, and Enterprise JavaBeans. They measure the performance of these three architectures using two applications: an online bookstore and an auction site. The online bookstore stresses the server back-end, whereas the auction site represents an application with most workload on the server front end. Their study reveals that PHP4 is more efficient than Java servlets, and the EJBs perform even worse than servlets. However, they note that servlets, being part of the Java solution, provides the flexibility of being able to be ported to another system with a different operating system. This research too does not compare design models of the same technology, as we do. Neither does it offer an insight into the underlying code of the technologies.

In a similar study, Cecchet *et al* [3] evaluate the performance and scalability of EJB applications using two different open source J2EE containers, JBoss 2.4 [7] and JOnAS 2.4.4 [9], as well as the performance of the EJB applications with servlet-based solutions. They find that the servlets-only application they build performs the best due to the fewer number of layer communications in the server. They find that JOnAS 2.4.4 outperforms JBoss 2.4 because of the more efficient design of the J2EE application server. This study is different from ours because it compares the efficiency of the infrastructure software (the J2EE containers) as opposed to the design models of the applications.

Also worth mention is the white paper from Sun Microsystems [10] that presents the functionality comparison of Java servlets, PHP, and CGI.

7 Conclusion

We find that it is most rapid to build Web applications using JSF. Model 2 applications are the least rapid but give the best performance. Struts applications sit in the middle of the other two design models in both comparisons.

We make some suggestions that could improve the Servlets technology in general and enhance the performance of applications based on both design models.

- **Struts.** Struts is not based on any specification and there is no documentation that discusses its internal working. Therefore, it is hard to know what have been implemented and what could be improved.
- **The Servlets Technology**. The Servlet 2.3 Specification does not define any caching mechanism. There is no mention of caching in the upcoming Servlet 2.4 Specification either. Despite the dynamic nature of the content of a Web application, some contents do not change very often. For example, the categories of products that a user can browse in an online store application probably only change once in a month. If those semi-static contents must be generated from the database every time they are requested, a lot of programming resources will be wasted. Servlet programmers get around the absence of caching by writing an object that caches certain content. However, since there is no standard for caching, many programmers write the same piece of code again and again.
- **Model 2**.The main drawback is that the page navigation rules are hard-coded in the Controller servlet. This means any minor change to the program flow will require the Controller servlet to be re-compiled. The solution to this problem is to provide a mapper that reads the page navigation rules when the application starts. The code could be conveniently written in the init method of the Controller servlet. This method is only executed once, i.e. the first time the servlet is loaded into memory. If the properties file needs to be re-read every time it changes, the programmer can check the timestamp of the properties file for each request, and compares it with the previous read of this file. If the timestamp is more current than the previous read, the mapper can be re-constructed. This feature can be enabled and disabled by using an initial parameter in the Context object. At the development phase, this feature should be enabled. At deployment, this feature should be off. The use of the properties file to store the page navigation rules also makes it possible to avoid a series of if statements in the Controller Servlet, which can be time-consuming for

every request. Instead, a HashMap can be used, with action request parameters as keys and the next JSP pages as values. The other disadvantage of this design model is the absence of standard components for input validation and user interface. However, this has been solved in JSF.

– **JSF.** JSF provides solutions to common problems in Web development, such as page navigation management, UI components and input validators. However, because this technology is still very young, there are not too many UI components available, forcing programmers to combine JSF with non-JSF servlets/JSP pages. JSF is event-driven. JSF programmers determine the behavior of a JSF application by writing event listeners, just like those listeners in a Swing application. In JSF version 1.0, there are currently two types of events that can be triggered: ActionEvent and ValueChangedEvent. However, this is good enough to provide sufficient level of interactivity between the application and its users. Adding more types of events will definitely make JSF more appealing.

References

1. Burbeck, S., Applications Programming in Smalltalk-80: How to use Model-View-Controller (MVC), http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html, 1987.
2. Cecchet, E., Chanda A., Elnikety S., Marguerite J., Zwaenepoel W.: Performance Comparison of Middleware Architectures for Generating Dynamic Web Content. *Proceeding of the 4th International Middelware* Conference, 2003.
3. Cecchet, E., Marguerite, J., and Zwaenepoel, W.: Performance and Scalability of EJB Applications. *Proceedings of OOPSLA'02*, 2002.
4. Java Servlet 2.3 and JavaServer Pages 1.2 Specification (JSR-053), http://jcp.org/aboutJava/communityprocess/final/jsr053/.
5. Java Servlet 2.4 Specification (Proposed Final Draft 3), http://jcp.org/aboutJava/communityprocess/first/jsr154/index3.html.
6. JavaServer Faces Technology, http://java.sun.com/j2ee/javaserverfaces/.
7. JBoss EJB server, http://jboss.org.
8. JMeter, http://jakarta.apache.org/jmeter/.
9. JOnAS: Java Open Application Server, http://www.objectweb.org/jonas.
10. Sun Microsystems, Comparing Methods for Server-Side Dynamic Content, http://java.sun.com/products/jsp/jspservlet.html, 2000.
11. The Apache Software Foundation, http://www.apache.org.
12. The Struts Framework, http://jakarta.apche.org/struts/.
13. Wu, A., Wang, H., and Wilkins, D.: Performance Comparison of Alternative Solutions for Web-To-Database Applications. Proceedings of the Southern Conference on Computing, the University of Southern Mississippi, 2000.