2009 IEEE International Advance Computing Conference (IACC 2009)
Patiala, India, 6–7 March 2009

High Quality Web-Application Development on Java EE Platform

Harshad B. Prajapati
Information Technology Department
Dharmsinh Desai University
Nadiad, India
harshad.b.prajapati@gmail.com

Vipul K. Dabhi
Information Technology Department
Dharmsinh Desai University
Nadiad, India
vipul.k.dabhi@gmail.com

***Abstract*—Quality of web-applications plays major role in its success. And the high quality web-application is really possible by following high quality web engineering process. The use of strong web-application architecture with strong development platform not only make web-applications robust and of high quality but also give web-application an ability to meet changing and demanding customer requirements in efficient manner. A Model View Controller (MVC) design pattern has remained fundamental architectural design pattern even after great evolution in architectures of user interactive applications. In this paper, we discuss the support for achieving quality attributes for the web-application, the support for web-application development process and the support for meeting demanding features of web application on Java EE platform. This contribution will help a lot for small scale as well as large scale web-application development and for moulding web-application into a future's high quality finished product from inception phase itself.**

Keywords-MVC; Web Application; Java EE; Quality Attributes; High Quality Web Development.

I. INTRODUCTION

Along with the great demand of web-applications in the era of pervasive and ubiquitous computing, a number of issues related to quality of web-applications [1] have also gained a great attention in web-application development process. Great competitions in the web-application business compel web-application developers more quality conscious. It will be of great value if a web-application could survive against demanding and changing customer requirements, and changing business requirements.

Building high quality web-application is really a difficult and challenging task. But the support for right development process, methods, tools, and people really make possible to achieve high quality of web-application. As the development platform influences associated development process, methods, tools, and people, it really plays a major role for making development process simple, efficient, and robust and for achieving high quality of web-application.

Java EE platform [7] is open, standard based [8], hardware and OS independent platform on which distributed enterprise applications can be developed and run. As the applications targeted on Java EE platform are vendor neutral, the organization does not face the problem of vendor lock-in. The Java EE platform based web-applications use Model/View/Controller (MVC) [2] design pattern [3] for three architectural components: presentation logic, controller logic, and entity/business logic. We discuss MVC design pattern used in traditional GUI based applications and also discuss how it is adapted in the architecture of Java EE platform based web-applications. We analyze the Java EE platform as per (i) quality attributes requirements [1] of web-applications (ii) related support in development process (iii) related support to involved people during development, and present our results in quick digestible form. Our results indicate that web-application development on Java EE platform has a great value in providing high quality to web-application, satisfying changing customer requirements and satisfying changing business requirements.

The presented work, in this paper, concentrates on achieving high quality for web-application developed on Java EE platform from different perspective of overall development process. The related work in the category is as follows. The Web-application development using the MVC design pattern can be found in [4]. The MVC design pattern in ASP.NET and JSP frameworks can be found in [5]. The quality characteristics and attributes for web sites can be found in [1].

The Section II discusses the MVC design pattern in traditional GUI application architecture and its adaptation in web-application architecture. The Section III presents web-application development process and available support for it on Java EE platform. The Section IV presents available valuable features for web-application on Java EE platform. Section V presents the result of Java EE platform analysis for achieving high quality of web-application and its components. And finally the Section VI presents the conclusions and direction for future research work.

II. THE WEB-APPLICATION ARCHITECTURE ON JAVA EE PLATFORM USING MVC DESIGN PATTERN

An MVC [2] design pattern [3] has remained fundamental architectural design pattern even after great evolution in the architectures of user interactive applications. So, we would like to provide short introduction on it to the readers before we discuss its usage in web-application architecture on Java EE platform [7].

978-1-4244-1888-6/08/$25.00 © 2008 IEEE

A. *The MVC Design Pattern*

The MVC design pattern is widely used by programmer, software designer, and GUI component developer to architect the widget they are developing. The MVC design pattern, as shown in Figure 1, consists of three kinds of objects: Model, View, and Controller, which handle three basic responsibilities of any widget: entity (data), boundary (presentation), and control (behavior) respectively. The model encapsulates application data and business logic; the view handles rendering of application data and visual interface to user; and the controller handles user's interaction with the application.

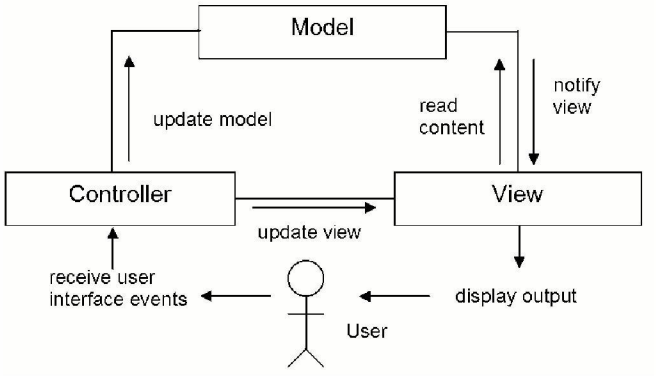

Figure 1. MVC Design Pattern

The MVC design pattern separates views and models by establishing subscribe/notify protocol between them. A view object must ensure that its appearance reflects the state of the model. The model object is independent of both view and controller objects, so it is possible to have multiple views (presentations) of same model (data). All associated views can subscribe with model and model notifies them about its state change.

When user interacts with the MVC design pattern based GUI form or page, all fired events are captured by controller object. The controller then decides whether the fired event is related to change in state of model or change in state of view. As an example, when the user fires event related to changing value in text field, the controller calls the method of model to change its content. But if the user performs horizontal or vertical scroll-up or scroll-down, the model content does not change and only the view should be notified to reflect the changes in its appearance.

B. *Web-Application Architecture in Java EE platform*

In a stand-alone application, generally the model, view, and controller live on the same machine. But in a distributed web-application, the application architecture is different; however, the MVC design pattern is so general that it can still be applied to its architecture. The Java EE platform based web-applications utilize MVC based architecture as shown in Figure 2.

In Java EE platform based web-application architectures, the servlet [10] component is used as a controller; the JavaBean component is used as a model; and the Java Server Pages [11] (JSP) page is used as a view template. The Enterprise Java Bean (EJB) [12] can be used as a model, which can be located in distributed environment unlike JavaBean. The JSP technology is used for creating view; and the JSP page is considered as a view template. The execution of this JSP page generates the view – HTML [13] content.

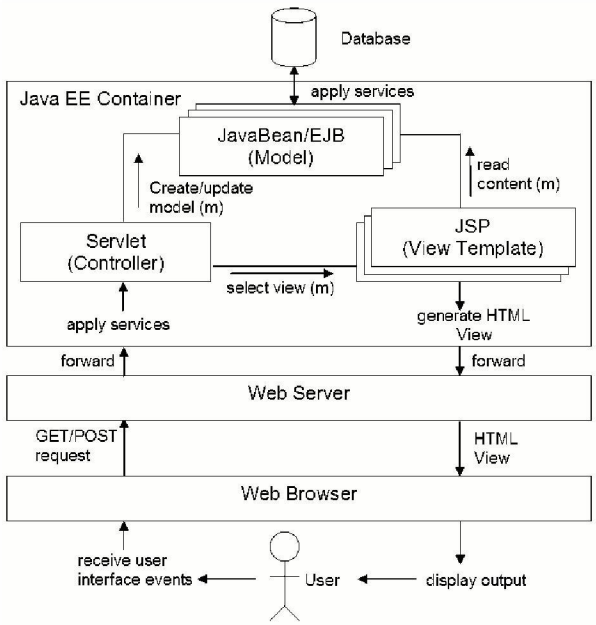

Figure 2. Use of the MVC Design Pattern in the Java EE Platform based web-application architecture

The Java EE platform provides many system services to web-application components. Java EE platform provides support for security, authentication, authorization, transaction. The database connection management is handled by the Java EE platform and it is configurable externally in deployment descriptor, so the model component does not need to worry about those details. Thus, the responsibility of model component is just to handle only business data and business logic. The authentication and authorization service is also provided by the Java EE platform to servlet and this service is also configurable externally in deployment descriptor. The container mediates between the servlet components and web browser for each HTTP [14] requests and applies the services as configured in the deployment descriptor.

III. WEB-APPLICATION DEVELOPMENT PROCESS ON JAVA EE PLATFORM

In this section, we discuss about web-application development process in Java EE platform [7] and the support available for web-application development from the Java EE platform.

A. *Web-Application Development Process*

A typical development process on Java EE platform involves following tasks: designing, coding, creating deployment descriptor, packaging, assembling, and deployment. These tasks are also applicable to web-application development also. The Java EE platform specification [8] specifies the roles and their responsibilities in Java enterprise application development. We present here these roles and their

responsibilities in various phases of web-application development process. The Table I shows the major roles played by involved people during development, their role based responsibilities and their involvement in specific development phases.

TABLE I. ROLE BASED DEVELOPMENT ON JAVA EE PLATFORM: ROLE, RESPONSIBILITIES, AND INVOLVEMENT OF PEOPLE IN DEVELOPMENT PROCESS

Role	Responsibilities	Involvement in Development Phases
Application component provider	Creating EJB components, web components	Development
Application assembler	Assemble different components into a deployable application. Configuring deployment descriptor	Development, Integration
Deployer	Deployment of the assembled application in a specific operational environment.	Deployment
System administrator	Maintenance and monitoring of application. Decisions on Load balancing, redundancy, Fail over, etc.	Maintenance

B. *Support in Development Process*

The development becomes efficient if tools/technology support is available during coding, testing, integration, deployment and maintenance phases. Here, we discuss how Java EE platform provides support during all these phases to make development process efficient.

1) Support in Coding: The development environment should provide facilities of avoiding typos, require minimal effort for adding new functionalities, and provide support for standards based technology and tools. Very sophisticated development tools are available for software development on Java EE platform. The Netbeans IDE [15] and Eclipse IDE [16] are among them. These IDEs are equipped with sophisticated editors. Few of such features are auto code completion, re-factoring, code insertion, syntax highlighting to avoid typo errors, fixing package imports, getter/setter handling, and code insertion for calling of EJBs. These features help a lot while developing servlet, JSP, JavaBean and EJB components.

2) Support in Testing: Good debugging and unit testing support are basic requirement in testing process. The Java platform [9] comes with debugger jdb and the IDEs such as Netbeans and Eclipse provide good debugging support. The JUnit [17] provides regression testing framework for unit testing. It can accelerate programming and increase the quality of code. It provides API for easily creating Java test cases, comprehensive assertion facilities, test runners for running tests, aggregation facility and reporting. All these features help the developers from embedding several println() calls in actual code and manually preparing test results in structured fashion. Adding and then removing individual println() for testing purpose is very frustrating task; and it can not be replicated into similar classes easily.

3) Support in Integration and Deployment: The integration and deployment of web-application should be as easy as possible to minimize deployment time and downtime. And the deployment process should be independent of different hosting application server providers. Since Java technology [9] is standard based, the web-application developed using it will run on any Java EE compliance [8] application server without any modification in code. The web-application is free from vendor lock-in. The Java EE platform specifies vendor neutral configuration in standard deployment descriptor and vendor specific configuration in vendor specific deployment descriptor files. The vendor specific configuration includes abstract security roles mapping with target security system, data source references, and other resource configuration.

4) Support in Maintenance: The Java EE platform specification clearly specifies the roles and responsibilities of different involved people. As discussed above. All these roles apply to web as well as enterprise module/application development. Three main roles: developer, assembler, and deployer simplify the whole development task. All three basic components: servlet, JSP and JavaBean/EJB components can be implemented and maintained independently by servlet developer, JSP developer, and Java/EJB developer, respectively.

IV. VALUABLE FEATURES FOR WEB-APPLICATION ON JAVA EE PLATFORM

As Java technology [9] is object oriented and platform independent, many features such as scalability, portability, reusability, security, high performance, flexibility are inherent in Java classes or components. Both Servlet [10] and JavaBean components are java classes, so above stated features are inherent in them. This also applies to EJB [12] component also, which is collection of java classes and deployment descriptor. JSP [11] scripting language is used to create JSP page. Although a JSP page looks like an HTML [13] kind of page, at execution time it is a translated java class which gets executed. In short, all three components: servlet, JavaBean/EJB, and JSP that are used in implementing MVC [2] design pattern [3] on Java EE platform [7] are scalable, portable, reusable, secure, high performance, flexible. The following available features on the Java EE platform add value to web-application. And in certain business/customer requirements, they are indispensable.

A. *Security*

The communication security (confidentiality and integrity) is provided through SSL [18] support. For SSL support, the SSL connector should be configured on the Java EE container and the server certificate signed by Certifying Authority (CA) [18] should be installed on the Java EE container. A Java EE based web-application is configured for confidentiality and integrity declaratively. The Figure 3 shows three configurations

related to security aspect of web-application. The part shown under <!—SECURE COMMUNICATION --> is related to confidentiality and integrity of data that gets transmitted between Java EE container and web-browser.

```
<!-- SECURITY ROLE -->
<security-role>
    <role-name>manager</role-name>
</security-role>

<!-- SECURITY CONSTRAINT -->
<security-constraint>
<!--ACCESS CONTROLLED RESOURCE AND OPERATION -
->
    <web-resource-collection>
        <web-resource-name>Inbox</web-resource-name>
        <url-pattern>/user/inbox/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
    </web-resource-collection>
<!—AUTHORIZED USER -->
    <auth-constraint>
        <role-name>user</role-name>
    </auth-constraint>
<!—SECURE COMMUNICATION -->
    <user-data-constraint>
        <transport-guarantee>
            CONFIDENTIAL
        </transport-guarantee>
    </user-data-constraint>
</security-constraint>

<!-- LOGIN CONFIGURATION-->
    <login-config>
        <auth-method>FORM</auth-method>
    </login-config>
```

Figure 3. Declarative Security Configurations for the Java EE platform based web-applications

The part under <!-- LOGIN CONFIGURATION--> in Figure 3 declares that user authentication must be performed before granting access to resources. And it declares that authentication method is of type FORM.

The authorization part includes which operations on which resources are allowed by which users. The part inside <web-resource-collection> indicates which resource is access controlled indicated by <url-pattern> and which operations are allowed indicated by <http-method>. And which users are allowed to perform selected operations on selected resources is configured under <!—AUTHORIZED USER --> part.

B. *Transaction Processing*

Java EE platform supports container-managed transaction and bean managed transaction for session bean and message driven bean (MDB) [12]. In Container-managed transaction, the enterprise bean code does not explicitly mark the transaction's boundaries using begin transaction and commit transaction. Instead, the transaction is configured in deployment descriptor. The container begins a transaction immediately before a business method in enterprise starts. It commits the transaction just before the business method in enterprise bean exits. If an exception gets generated during business method execution, the container will automatically roll back the transaction. The support for only either a single transaction or no transaction at all is available from container managed transaction. For fine-grained control over transaction, the bean-managed transaction can be used. Here the code in the session or message driven bean explicitly marks the boundaries of the transaction using Java Transaction API (JTA).

C. *Support for session managemen*

Most of the web-applications on the Internet handle the session using cookie mechanism. So, the users, who are accessing such web-applications from browsers that do not support cookie mechanism, cannot participate in session and will not be able to access any personal resource. The encode URL mechanism on Java EE platform automatically determines whether the client's browser supports the cookie or not, and then decides how the information about session identification should be stored on client machine. If the client's browser does not support the cookie or the cookie feature has been disabled by the user, the session id value is encoded in URL part of each hyper-link on the page that client is going to use.

D. *Customized error-pages*

If a user is trying to access a resource, which is not available, the server shows the error message 404 (SC_NOT_FOUND) resource not found exception. Instead, the error page containing beautiful description in non-technical English language is shown, the user will really understand the cause of problem and sometimes feels that you care for the users. A good web-application should not show the error messages generated by application server or web-server directly to users. As an example, a web-application can be configured for 404 resource not found exception as shown in Figure 4. Using this configuration, the error pages for user-defined error messages can also be specified. Instead of error-code, the exception-type can also be specified. The resourceNotFound.html file would contain error message in a language understandable to user.

```
<error-page>
        <error-code>404</error-code>
        <location>/resourceNotFound.html</location>
</error-page>
```

Figure 4. Declarative Error-page Configuration using error-code and location

E. *Internationalization/Multi-language Support*

If information is provided to users in a language that they understand and use, it would be easier for users to understand and use the application. The Unicode support for strings is inherent in the Java language [9]. The language also supports locale specific formatting of number, currency, date, time, etc. All such supports help a lot in making the Java EE platform [7] based web-application an internationalized one and they also increase the usability of web-application across the world.

V. ACHIEVING QUALITY FOR THE WEB-APPLICATION AND ITS COMPONENTS ON JAVA EE PLATFORM

The quality of individual components also affect in achieving overall quality of whole application. We, first, present here important quality attributes of components and how these quality attributes are satisfied for all three components: Model, View, and Controller. The related analysis and results are presented in Table II. Second, the design of any web-application includes various design tasks [6]. These design tasks and available support for them on Java EE platform [7] are presented in Table III. Third, the overall support for quality attributes [1] to web-application as a whole determines the overall quality. These quality attributes, criteria involved in achieving these quality attributes and available support for them on Java EE platform are presented in Table IV.

TABLE II. QUALITY ATTRIBUTES OF COMPONENTS AND SUPPORT FOR THEM ON JAVA EE PLATFORM

Attributes	**Support for quality attributes to MVC components**		
	Model	***View***	***Controller***
High Performance	JIT compiler for fast compilation of byte code, Separate thread per user request, Local EJB components to improve performance.		
Extensibility	All components are Object Oriented.		
Scalability	Java EE platform is providing distributed load balancing mechanism. It is a vendor specific feature mentioned in specification.		
Security	Declarative and Programmatic security for EJB component. Java Bean components are accessible through view/controller.	Using URL masking.	Declarative and Programmatic security for Servlet component.
Robustness	Components run under control of JVM, dynamic class loader and byte code verifier.		
Flexibility	Using JNDI based declarative database access configuration. Change in model object using scoped attributes is easily possible.	Using master page, file/page inclusion, request dispatching, request forwarding	Using request-dispatching and forward methods, filter chain.
Modularity	EJB jar modules	Tag library, war modules	war modules, filters
Reusability	Remote EJB object support	Custom tag library, page/file inclusion, links	Using request dispatching and forward methods. Filter chain.

TABLE III. WEB-APPLICATION DESIGN TASKS AND SUPPORT FOR THEM ON JAVA EE PLATFORM

Attributes [6]	**Java EE Platform support for design task**
Interface Design	HTML controls, Java Server Faces (JSF) framework controls.
Aesthetic Design	Cascading Style Sheet (CSS) support.
Content Design	Container managed relations for entities and entity beans and provides EJB-QL.
Navigation Design	In JSF framework, we can configure navigation graphically as well as manually in faces-config.xml.
Architecture Design	MVC design pattern.
Component Design	Life cycle of all components is handled by Java EE container and provides basic skeleton for all components.

TABLE IV. WEB-APPLICATION QUALITY ATTRIBUTES AND SUPPORT FOR THEM ON JAVA EE PLATFORM

Attributes [1]	**Criteria**	**Support on Java EE platform**
Usability	- Global site understandability - On-line feedback and help features - Interface and aesthetic features - Special features	- Internationalization support, locale specific resource bundles - Custom error pages - Components: inbuilt as well as Custom user interface component development in JSF. CSS support - Transaction processing, security
Functionality	- Searching and retrieving capability - Navigation and browsing features - Application domain-related features	- create, read, update, delete, find, business functions support on EJB entity beans and persistence entities - View: declarative navigation in JSF framework Model: Container Managed Persistence relations in EJB entity beans and persistence entities. - EJB session bean: business logic, business rules. EJB entity beans and persistent entities: business data EJB Message Driven Bean: Asynchronous business event handling.
Reliability	- Correct link processing - Error recovery - User input validation and recovery	- Declarative navigation is accurate. - Exception handling support. - JSF framework supports data conversion and validation.
Efficiency	- Response time performance - Page generation speed - Graphics generation speed	- Separate thread per user request. - JSP is complied once. It is fast. - No explicit support is available.
Maintainability	- Ease of correction - Adaptability - Extensibility	- Role based and concurrent maintenance task. - Java EE is standard based, no vender lock-in. - Components are object oriented

VI. CONCLUSION

We analyzed the Java EE platform as per quality attributes requirements of web-applications and found that quality attributes: usability, functionality, reliability, efficiency and maintainability can all be satisfied with it. From the discussion it is evident that Java EE platform simplifies the designing, development, deployment, integration and testing process of web-application without compromising high quality. It also empowers the development of web-application with scalability, portability, interoperability, reusability, flexibility and security.

Many frameworks for web-application development are available on Java EE platform, but no one is found providing support for high quality distributed web-application. In future, we intend to work on developing distributed high quality web-application framework on Java EE platform.

REFERENCES

[1] Olsina, L. et al., "Specifying quality characteristics and attributes for web sites," in Proc.1st ICSE Workshop on Web Engineering, ACM, Los Angeles, May 1999.

[2] G. E. Krasner and S. T. Pope, "A Cookbook for Using the Model-View-Controller User-Interface Paradigm in Smalltalk-80," Journal of Object-Oriented Programming, August/September 1988.

[3] E. Gamma, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.

[4] A. Leff and J. T. Rayfield "Web-application development using the model/view/controller design pattern," in Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing, 2001, pp. 118–127.

[5] F. A. Masoud, Dana. H. Halabi and Deema. H. Halabi, "ASP.NET and JSP frameworks in model view controller implementation," in Proceedings of Information and Communication Technologies, 2006. ICTTA '06, pp. 3593–3598.

[6] R. S. Pressman, Software Engineering: A Practitioner's Approach 6e, McGraw Hill, 2005.

[7] Sun MicroSystem. Java Enterprise Edition. http:// java.sun.com/javaee/

[8] Java Community Process. http://jcp.org/

[9] Sun MicroSystem. Java Technology. http://java.sun.com.

[10] Sun MicroSystem. Java Servlet Technology Implementation and Specification. http://java.sun.com/products/servlet/download.html#specs

[11] Sun MicroSystem. Java Server Pages. https://java.sun.com/products/jsp/

[12] Sun MicroSystem. Enterprise JavaBeans Specifications. http://java.sun.com/products/ejbs/docs.html.

[13] W3C Specifications. http://www.w3.org

[14] RFC: http protocol, http://www.w3.org/Protocols/rfc2616/rfc2616.html

[15] Netbeans IDE, http://www.netbeans.org.

[16] Eclipse IDE, http://www.eclipse.org.

[17] JUnit Framework, http://www.junit.org.

[18] RFC : SSL, www.ietf.org/rfc/rfc2246.txt