Merging e-Business Solution Framework with Java Components

Mayur R. Mehta
George W. Morgan
Department of Computer Information Systems, Southwest Texas State University
San Marcos, Tx 78666

Abstract

Since corporations first started conducting business on the Internet in 1993, it's moved quickly from being a curious spectacle to a matter of survival for most businesses. The number of organizations conducting business over the Internet over the past several years has far exceeded most market projections and expectations [21, p.1]. To achieve successful results on a consistent basis, companies need to rely on two critical success factors. A robust framework and development environment are more critical than ever for corporations to deploy successful e-Business applications. This is a especially true for companies that are interested in web-based applications that are robust, flexible, scalable, maintainable and platform-independent. These characteristics will gain importance as corporations start to migrate their e-Business applications from the traditional Web-based environment to wireless, mobile, hand-held and pervasive computing paradigm.

This paper will first describe an e-Business solution framework as presented by one of the leading providers of enterprise-level e-Business application development tools. Following the presentation of this framework, the paper will discuss how e-Business solutions based on this framework may be deployed using Java-based technology components.

Keywords: e-Business, e-Commerce, design framework, Java, J2EE

1. INTRODUCTION

The Web is changing every aspect of our lives, but no area is undergoing as rapid and significant a change as the way businesses operate. As businesses incorporate Internet technology into their core business processes they start to achieve real business value. Today, companies large and small are using the Web to communicate with their partners, to connect with their back-end data-systems, and to transact commerce. The next generation of business has arrived—it's called e-Business.

What is e-Business?

e-Business is defined as the transformation of key business processes through the use of Internet technologies [6]. In e-Business, companies use the Web technologies (such as HTTP/HTTPS, IIOP, Web clients and servers, and business objects) to communicate with their partners, to connect with their back-end data-systems, and to transact commerce in such a way that it leverages the strength and reliability of traditional information technology in the Internet environment [15, 16]. This new Web + IT paradigm merges the standards, simplicity and connectivity of the Internet with the core processes that are the foundation of business [6].

Successful e-Business systems have the following desirable properties [6, p. 2; 8, p. 12; 15]:

- Application simplicity and reusability,
- Leveraging current developer skills, data and information,
- Robust security with good performance,
- Applications and Systems Manageability,
- Deployment flexibility and scalability to migrate to a variety of computing paradigms including wireless, mobile, hand-held, and pervasive.

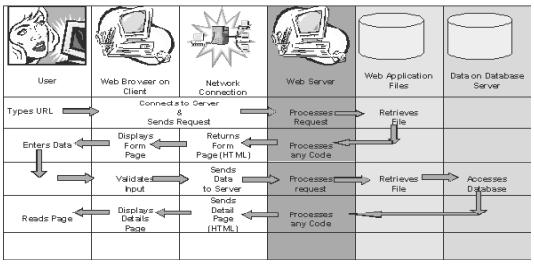
Therefore, successful e-Business applications are based on standards that span multiple platforms. They are server-centric. They extend existing applications. They are scalable, easy to develop and use, and they are built to be managed. To get applications with these characteristics, it is imperative that developers have a clear understanding of a robust solution framework. Furthermore, developers need to acquire an appreciation for the available development tooling that leads to robust, secured, reliable, and scalable e-Business applications within the guidelines of this framework.

The primary objective of this paper is to provide this introduction. This paper will first describe an e-Business solution framework as presented by one of the leading providers of enterprise-level e-Business application development tools [8, 15, 16]. Following the presentation of this framework, the paper will discuss how e-Business solutions based on this framework may be deployed using Java-based technology components.

2. e-Business Solution Framework

e-Business applications leverage Web clients (such as Web browsers running on PCs, PDA, and pervasive internet appliances), Web application servers, and standard Internet protocols. They also typically leverage existing applications and data from external non-Web services [8, p.1; 15, p. 3]. To fully appreciate the dynamics of e-Business application, one must first understand the underlying process of a simple Web-interaction that involves a business transaction. Figure 1 presents a simplistic scenario of such a Web-interaction.

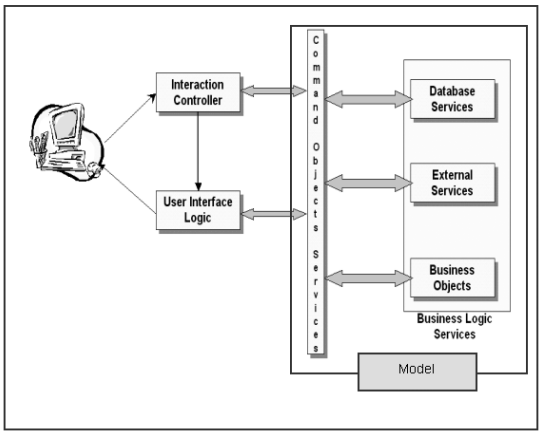
Figure 1. Process Underlying Web-Interaction



In this simplistic scenario, a user navigates to a secured web site to make a credit card payment. This process of navigating to the web site, by typing in the URL, is essentially making a request to the server to retrieve the web page file and deliver it as an HTML document to the browser, Upon receipt, the web client renders this HTML document as a web page to the user. The user next enters the required data and submits the request to the server for processing. The web client first validates user input and then forwards the request to the server. Upon receipt, the server processes the business transaction by accessing the user record in the database, updating the record for payment, and then generating a response to confirm the transaction. The confirmation response, in the form an HTML document, is delivered to the web client, which then renders it as a web page and presents it to the user. The server may call upon a variety of services to successfully complete this business transaction. Such services may include locating an appropriate database; validation logic to authenticate user; business logic to compute principle, interest, balance and late charges; and external services such as bill pay. Furthermore, data and business logic may have to be accessed from legacy, non-Web based systems.

While simplistic in nature, this scenario captures the elements that play a critical role in a robust e-Business solution framework. It also identifies that key processing requirements include interaction with user, executing and managing business logic, and application logic to control the interaction between user view and business logic. These key processing requirements can be easily mapped to the classical Model-View-Controller (MVC) paradigm, leading to a layered application architecture [8, p. 5; 12, p. 27-29]. Figure 2 illustrates the MVC paradigm, which provides a framework for separating functional processing involved in typical computing-oriented transactions into three major components - Model, View, and Controller [8, p. 4-6; 12, p. 26; 14, 15, 16].

Figure 2. MVC Paradigm



View (Presentation Layer in layered architecture) is the user interface logic part of the system. It is responsible for generating HTML pages that will be returned to the client. It includes objects defined to accept user inputs and to display formatted application output. The data is received by the UI logic in two ways. In some cases, the UI logic will invoke the necessary business logic to get the data directly from the database services. In other case, the data-retrieval may be delegated to the Interaction Controller.

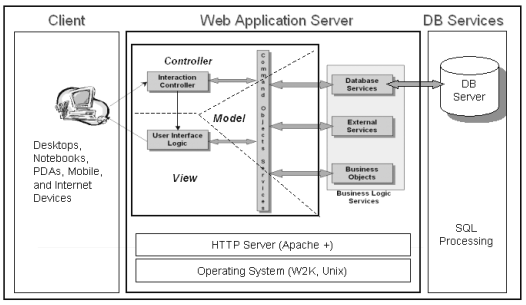
Model (Domain and Data Access layer in layered architecture) is the business logic portion of the system. It comprises of code that is ultimately responsible for satisfying client requests. As a result, business logic must address a wide range of potential requirements, which include quick access to application data in a secured manner, coordination of business workflow processes, and integration of new application components with existing applications. The Command Objects service layer encapsulates interactions between View/Interaction Controller and these core services.

Interaction Controller (Controller/Mediator layer in layered architecture) is responsible for mapping HTTP protocol specific input into the input required by the

protocol independent business logic (that might be used by several different types of applications), invoking appropriate business logic models, and then invoking appropriate view logic model to create the response page to be returned to the client. It, thus, handles client-side input and validation, mapping of the request and session parameters to the business logic components, and logic flow to correctly chain the business logic.

As can be seen, the MVC paradigm and layered architecture provides a mechanism to decouple business logic implementation from presentation logic. Such a separation offers several advantages including ability to develop, modify, and manage business logic independent of form and style of resulting presentation. More importantly, this layered architecture suggests a possible framework that would lead to e-Business solutions that are server-centric, scalable, and platform independent [12, p. 25]. Consequently, this framework would support any client device, including the traditional desktop, handheld, set-top and other pervasive devices. Additionally, the framework fosters component-based application development as well as reuse of enterprise business objects [12, p. 1]. This e-Business solution framework is illustrated in Figure 3, superimposed over the layered architecture discussed earlier. As indicated, the web application server provides a majority of the services required in this framework.

Figure 3. e-Business Solution Framework



The key development and deployment issue is obviously how to implement this framework. What technology is available to build applications to these specifications so that resulting solutions are platform independent, scalable, easily developed, and can be deployed to a variety of clients without much reprogramming?

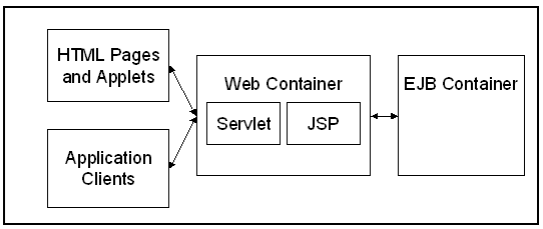
3. Java 2 Enterprise Edition (J2EE)

J2EE architecture has been strongly promoted by the proponents of this framework and e-business developer community as the leading and, perhaps, the only technology currently available that accomplishes the above [7, 12, 13, 15, 16]. The primary force behind this widespread adoption of J2EE is the fact that Java-based applications can be delivered over any network, operating system and hardware. J2EE technology simplifies enterprise applications by basing them on standardized, modular and reusable Enterprise JavaBeans (EJB) components, providing a complete set of services to those components, and handling many details of application behavior automatically [9]. A discussion of how the J2EE architecture maps to the e-solution framework follows.

The J2EE Architecture Applied to e-business Solution Framework

One way to address the merging of Java technologies with the e-Business framework presented earlier is to look at the Web based components. J2EE is a container-based architecture and is, therefore, a suitable technology to implement the MVC paradigm. Figure 4 illustrates this concept. Containers such as EJB, Web, JavaServer Pages (JSP), servlet, applet, and application client provide the component-specific services and will be the basis of our discussions. The core of a design pattern [12, p. 25] expressed as MVC that includes the three major technologies of servlets, JSPs, JavaBeans and EJBs will be included [9, 10].

Figure 4. Container Concept of J2EE Architecture



HTML Pages and Applets

Sharing Web pages between nodes on a network is most often accomplished using HyperText Transfer Protocol (HTTP) and its secured "cousin", HTTPS. It allows hypertext objects from remote hosts to be sent to other nodes for display. Many of these documents are written using the HyperText Markup Language (HTML) which is one of a file formats available for document handling on the World Wide Web (WWW). HTML tags are used by the developer so that references to other objects, images, sounds, video, fields, and other simple text formatting can be managed by the documents. As a result, Web documents make excellent input and output forms when developing Web applications.

One way to utilize this feature in Java is to insert code statements directly into the hypertext markup language of a Web document. Such code statements may be included either as embedded Java program classes or embedded JavaScript. JavaScript is a Web scripting language that is used to execute business logic in both browsers and Web servers. Like all scripting languages, it is used primarily to tie other components together or to accept user input. HTML/JavaScript and HTML/Java may be the preferred technology for many e-business applications when used in combination with server side JSPs and servlets.

An alternate approach is to construct Java mini-programs that can be downloaded and executed as part of a displayed Web page. This is an example of applet development and deployment. The resulting component that typically executes in a Web browser, can also execute in a variety of other applications or devices that support the applet-programming model. Several problems do exist for this approach however. One is that the required version of Java may not be supported by the active Web browser and two, some business logic may need to be downloaded from the Web server tying up internet resources when the applet is initialized. Many Web application developers stay away from applets for these reasons alone.

Application Clients
If the developer wants to use a GUI as the principle input or output form, then constructing form based applications as a first-tier client program may be the preferred approach. The application resides completely on the client and executes in its own Java Virtual Machine (JVM) providing full program control. Strategically placed references to Web based objects and direct access to databases make for a more powerful front-end connection. This approach will also allow database connectivity through JDBC for complete SQL processing without passing through the Web application server. Often times, developers prefer this approach for e-business applications that are deployed to company intranets, since this approach does offer considerable performance enhancement.

Web Container
The Web component of the J2EE architecture must be implemented using resources assigned to the Web container and therefore serviced by the Web server software. A Web server will host the affected Web site, provide HTTP support, and execute and manage server-side programs running under their own service engine performing their own particular functions (servlets and JSPs). The container provides an environment by which business logic (Model or Domain layer) can be separated form the view of an application's state (View or Presentation Layer) with user interaction providing the control. Whenever a request is received through a URL, the Web server will load the control (servlet) into the appropriate JVM, if necessary, and then execute it. Processing each user request through the various model and view components to a completed task will most likely end by generating a response back to the invoking Web browser in the form of another Web page. The Web container therefore houses components to implement an application's model, view, and controller logic.

Servlet
In the MVC paradigm, the controller must provide the logic necessary to process an HTTP request, reference the affected business logic program, and select the appropriate view response generator. In the Web server, the servlet becomes the controller. It is simply another Java program that extends the functionality of a Web server. The servlet container provides the network services over which a request may be validated, privileges verified, input mappings made to the business logic components (JavaBeans and EJBs), and response logic invoked to insure a completed client request-response cycle. In effect, servlets respond to user interface messages and build logic that is necessary to send dynamic contents back to web client.

JavaBean
The simple definition of a JavaBean expresses it as a reusable software component that can be combined in ways with other components to create applications. JavaBeans are portable, platform-independent components written in Java that enables developers to write reusable logic once and run it anywhere [11, p. 2; 22, p. 19]. They enable developers to encapsulate business logic into reusable software components, which may be used with any application that requires the encapsulated business logic. JavaBeans, along with EJBs, are therefore used to implement an application's business rules or the model layer of the MVC pattern [7, p. 11].

Java Server Pages (JSP)
Java Server Page (JSP) offers a template-based approach with custom elements, scripting languages, and server-side Java objects for integrating dynamic content into Web pages [11, p. 1; 12, p. 181]. Typically the template data is HTML or XML elements, and in many cases the client is a Web browser. Once invoked, tags are used in order to insert the properties of a JavaBean object and related script elements into a JSP file for transfer. The primary use of JSPs is to implement an application's view logic.

Enterprise JavaBean (EJB)
Enterprise JavaBean (EJB) is component architecture for the development, deployment and implementation of object-oriented, distributed, enterprise-level applications. Multiple EJB classes may reside in the EJB container, which is solely responsible for making each EJB class available to the requesting client. EJBs provide multiple client types (Web browsers, cell phones, PDAs, Java client applications, etc.) access to enterprise data and shared business logic [10, p. 10; 12, p. 264]. Applications written using the EJB architecture will implement a business task or business entity either as an entity bean or as a session bean.

Entity Bean
An Entity Bean is a persistent data object. The object will represent a view of information to be stored as well as the actual data that can be saved in a variety of persistent data stores; typically, a relational database [12, pp. 262-263]. In brief, entity beans represent permanent business data. They carry their own primary key for object identification. If the container in which an

entity bean is hosted crashes, the entity bean, its primary key, and any remote references survive the crash. Entity beans that manage their own persistence are called Bean-Managed Persistence (BMP) Entity Beans and those that delegate this function to their container are called Container-Managed Persistence (CMP) Entity Beans.

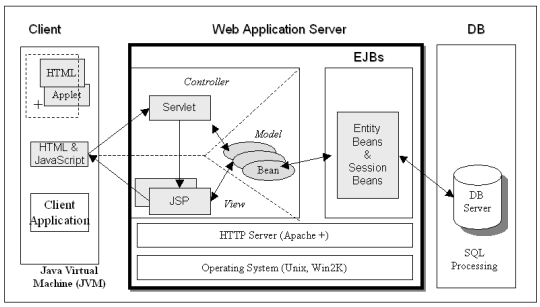
Session Bean

A session is an object used by a servlet to track a user's interaction with a Web application across multiple HTTP requests. A session bean is a form of EJB that is created by a client and usually exists only for the duration of a single client/server session. A session bean performs operations such as calculations or accessing a database for the client [20, p. 30]. It therefore represents some type of business process [12, pp. 262-263]. While a session bean may be transactional, it is not recoverable should a system crash occur. Session bean objects can be either stateless or they can maintain conversational state across methods and transactions. If they do maintain state, then the EJB container manages this state if the object must be removed from memory. However, the session beam object itself must manage its own persistent data.

4. E-Business Solution Framework using Java Technology (J2EE) Components

Having discussed an e-Business solution framework based on the MVC paradigm and the technology that is available to implement it, we are ready to see how his technology may be is applied to build an application that will comply with desirable characteristics of a successful e-Business solution. As before, we will walk through our simplistic scenario of a user accessing a secured web site to make a credit card payment to understand the implementation. Figure 5 illustrates the process involved.

Figure 5. e-Business Solution Framework using Java Technology (J2EE) Components



A user navigates to a secured web site to make a credit card payment by typing in the site's URL on the web client. The Web client may be any Internet device such as a PC, PDA, or even a pervasive appliance. This is essentially a request to the HTTP server to retrieve the web page file and deliver it as an HTML document to the browser.

The server invokes the appropriate Java servlet to determine the action to take based upon the request. For example, if the request was for a payment form and was made via a PDA, the servlet would invoke an appropriate JavaServer Page (JSP), which would then generate an HTML form document (perhaps with embedded java code for local validation) suitable for rendering on the PDA. The response could also materialize the payment form as a Java applet or as Java client application. Upon receipt, the web client renders the payment form to the user. Hence, the servlet acts as the interaction controller while the JSP handles the view logic.

The user next enters the required data into the displayed payment form and submits the request to the server for further processing. The web client first validates user input and then forwards the request to the server. This local validation is easily handled by the embedded logic written in JavaScript or by Java classes. Upon receipt, the web server again invokes the servlet engine to handle the request.

The Java servlet identifies the request as a credit card payment. It then builds the logic necessary to process this transaction by calling upon various JavaBeans and EJB objects, including:

- A command bean to compute principle, interest, and new account balance as well as returning information for generating a confirmation of the transaction;
- A command bean to invoke an EJB to connect to database and retrieve data; and
- A command bean to return information for a confirmation response.

Some of these are responsible for tracking session parameters while others are responsible for executing the business logic. In addition, Java servlet may call upon a variety of services to successfully complete this business transaction. Such services may include locating appropriate JavaBeans and EJBs via directory services offered by Java Naming and Directory Interface (JNDI); EJB SQL processing via Java Database Connectivity (JDBC); and external services such as bill pay via Java Messaging Services (JMS). Again, as is evident, Java servlet acts as a controller by controlling logic flow to correctly chain the business logic needed to process the transaction. On the other hand, JavaBeans and EJBs are used to encapsulate the business logic.

The Information for a confirmation response is passed to the appropriate JSP selected by the servlet via a command JavaBean. The JSP is then responsible for generating an HTML document, with embedded JavaScript or Java program class if further processing is

required on the client, and delivering it to the web client. The web client renders it as a web page and presents it to the user for further action.

While simplistic in nature, this scenario does illustrate the role of key J2EE technologies in implementing an e-Business solution framework that is based on the MVC paradigm. It further illustrates how the J2EE architecture provides a mechanism to distribute functional processing across the model, view, and controller layers of this paradigm.

5. Summary

Implementing a successful e-Business solution requires careful planning, both from strategic and technological perspectives. It requires that information technology professionals have clear understanding of the key design issues as well as a solution framework that guides the design and implementation of e-Business solution. This paper examined an e-Business solution framework, based on the Model-View-Controller (MVC) paradigm, as presented by one of the leading providers of enterprise-level e-Business application development tools. The role of J2EE architecture as applied to this e-Business solution framework was examined following an introduction to the framework. Finally, the paper discussed an approach to implementing the suggested framework using Java-based technology. It is suggested [8, p. 12] that e-Business applications designed and deployed using this J2EE implementation of the framework will, in all likelihood, result in applications that:

- are server-centric, use thin clients, and therefore, easy to manage;
- offer scalable architecture;
- offer robust security with good performance;
- foster reuse of enterprise business objects, knowledge and experience;
- leverage current developer skills, data and information; and above all,
- offer flexibility to deploy to a variety of computing paradigms including wireless, mobile, hand-held, and pervasive.

References

1. ___ "Check out these numbers," e.biz section, Business Week, March 22, 1999.
2. ___ "e-Commerce: Not your grandfather's five-and-ten.", Forrester Research Report, No. 617/520-5791, 1998
3. ___ "e-Commerce: transforms your web site into a profit center," IBM White Paper, http://www.software.ibm.com/ebusiness/e-commerce, May 1999, pp 1-2.
4. ___ "e-Life: How the Internet is Changing America," Special Report, Newsweek, September 20, 1999, pp. 26 - 64.
5. ___ "Gartner Group Reports That an e-Commerce Web Site Costs $1 Million to Build, " GartnerGroup Report, May 1999.
6. ___ "Ready for e-business: A CIO's Guide to e-business applications", IBM White Paper, http://www.software.ibm.com/ebusiness/cioguide, May 1999, pp. 1-9.
7. ___ "IBM Application Framework for e-business-Understanding Technology Choices," IBM White Paper, http://www3.ibm.com//software/ebusiness/buildapps/understand.html, May 2002, pp. 1-35.
8. ___ ""IBM Application Framework for e-business: Web Application Programming Model," IBM White Paper, http://www-106.ibm.com/developerworks/features/framework/framework.html, May 1999, pp. 1-13.
9. ___ "What is Java™ Technology? " Sun Microsystems, Inc. White Paper, http://java.sun.com/products/jsp/, 2000, pp. 1 – 5.
10. ___ " Glossary of Java™ Technology-Related Terms," Sun Microsystems, Inc. White Paper, Http://java.sun.com/docs/glossary.print.html, May 2002, pp.
11. ___ " JavaServer Pages™," Sun Microsystems, Inc. White Paper, Http://javasoft.com/products/jsp/, May 2002, pp. 1-2.
12. Brown, Kyle, et. al., Enterprise Java Programming with IBM WebSphere, Addison-Wesley, 2001.
13. Grehan, Rick., "A Scalar Force," JavaPro, Fall 2002, pp. 1 – 5.
14. Habibulah, Asif , and Jimmy Xu, "Take the Pain OUT of Distributed Java," JavaPro, July 2001, pp. 1-9.
15. Harkey, Dan, Ken Burgett, and Tim Stone, "From e-Technology to e-Commerce," IBM News on Web Application Servers, http://www-4.ibm.com/software/webserver/harkey/hmay99.html, May 1999, pp. 1-7.
16. Harkey, Dan, Ken Burgett, and Tim Stone, "From e-Technology to e-Commerce," IBM News on Web Application Servers, http://www-4.ibm.com/software/webserver/harkey/hjune992.html, June 1999, pp. 1-4.
17. Kalakota, R., Andrew B. Whinston (Contributor), and Tom Stone (Editor). Frontiers of Electronic Commerce, 1996.
18. Kalakota, R., Marcia Robinson, and Don Tapscott. e-Business: Roadmap for Success, Addison-Wesley Pub Co.,1999.
19. Korper, Steffano and Juanita Ellis. The e-Commerce Book: Building the e-Empire, Academic Pr; 1999.
20. McLaren, Bruce J. and Constance H. McLaren. e-Commerce: Business on the Internet, South-Western Pub; 1999.
21. Murphy, Kevin and Maureen Flemming (contributors), "A CEO's Internet Business

Strategy Checklist: The Leading Questions," GartnerGroup Report, Http://gartner5.gartnerweb.com/public/static/hotc/041999rr02.html, April 19, 1999.

22. Rosenberg, Jothi., "JavaX – An Approachable Examination of Java, JavaBeans, JavaScript, and All The Related Java Technologies, " Http://developer.netscape.com/docs/wpapers/javax/javax.html, 1997, pp. 1 – 36.