Evaluating Presentation Layer Development Frameworks for EJB Applications in J2EE Architecture

Ohm Samkoses, Dipl. -Inform. Matthias Vianden, Prof. Dr. rer. nat. Horst Lichter

Abstract—For medium to large organizations, information systems play an important role for information storage and retrieval. They are used to support business processes such as decision-making. In information systems, huge amounts of data need to be manipulated and visualized. One way to handle this complexity is to use Enterprise JavaBeans (EJB) [1] in a J2EE architecture. Since EJB has not been designed to work in the Presentation Layer, suitable Presentation Layer Development frameworks are introduced to enhanced the presentation layer of the information systems. The MeDIC (Metric Definition Integration Calculation) system and XAM (eXam Assignment and Management) system [2] are typical representatives of such information system.

Nowadays, many frameworks, such as Java Server Faces (JSF), Wicket, and Tapestry, exist to cover the Presentation Layer. They provide a variety of features and architecture enhancements. The goal of this research paper is to evaluate frameworks for the Presentation Layer of information systems, which shared the same architecture with the proposed architecture of the MeDIC and XAM system.

Keywords—J2EE Architecture; Enterprise JavaBeans; Presentation Layer Development Frameworks; Information System;

I. INTRODUCTION

In the proposed architecture, the Data Layer and Business Logic Layer are managed by EJB. The Presentation Layer communicates to the Business Logic Layer via Application Facade. The Presentation Layer is currently implemented using JSP/Servlet technology. However, the current design still has several defects, for instance: code redundancy, huge amount of classes, and low reusability. In consequence, J2EE presentation development frameworks minimize those defects and improve the efficiency of the system, which shared the similarities with the proposed architecture.

The goal of this research paper is to evaluate frameworks for the Presentation Layer based on the proposed architecture, including the current solution (JSP/Servlet). The steps to achieve the goal of this research paper include requirements gathering and analysis, multiple rounds of frameworks selection with different criteria, prototype implementations, and results evaluation. The chosen frameworks for prototype development should fulfill most of the requirement and the prototypes implementation should reflect the solution for each requirement clearly, and must be able to work with the existing system’s environment: the back-end is managed by EJB 3.0, and IBM Websphere Application Server 7.0.

II. TASKS / STEPS

A. First frameworks selection

First step was to narrow-down the scope of the focused framework based mainly on framework popularity. Other factors were general web framework criteria [3], such as, learning curve, testability, configuration complexity, amount of artifact produced, architecture and pattern, tools and IDE, and supports from community. This first selection aimed for 5-10 frameworks as an output.

B. Frameworks analysis

The outputs from the first framework selection were analyzed at this step. The selected frameworks were compared to each other under five general web framework criteria: configuration complexity, learning curve, testability, community and support, tools and IDE.

C. Requirements gathering and analysis

All attendances were developers, who experienced using the system based on the proposed architecture. Also requirements prioritization was used to identify the important factor of each requirement.

D. Second frameworks selection

In the previous step, 3–4 frameworks were aimed to be selected as an output. The main factor of the selection is the requirements, the general web framework criteria were also affected, but less because the requirements represent the characteristics of the solution itself. In contradiction, even a framework fulfills most of the requirements, it should not fail on the general web framework criteria, which represent the characteristic of high quality, efficient, productive web framework.

E. Prototypes implementation

The selected frameworks were implemented for a proof of concept. For each prototype, there were five aspects need to be
mentioned: working environment and tools, framework's basic concepts, complete architecture after the integration of framework, project structure and artifacts produced, and steps to migrate from the existing system.

F. Results evaluation

All implemented frameworks were analyzed against the requirements. The result from the comparison against general web framework criteria in step 2 also affects the evaluation at this point.

These important steps were demonstrated and explained in detail in the following chapters of this report.

III. BACKGROUND

The architecture of the existing system is based on the J2EE architecture. Follow the J2EE approach, the system is divided into 3 layers: Presentation layer, Business Logic layer, and Data layer.

The presentation layer composed with plain JSPs and Servlets. The request from the browser will be centralized to the Action Handler Servlet and delegates to the corresponding Action class. The Action class communicates with the Application Facade in the Business logic layer.

The Business Layer and Data layer are managed by EJB framework. The Application Facade delegates the request sent by Actions in Presentation Layer to the corresponding Controller. The controllers are the interfaces to simplify the business logic in Management. The Entity persisted, and the results from the database render in JSP page in the presentation layer.

On the client layer, the existing system uses the Command pattern to encapsulate the request parameters. All requests pass through the Action Handler then the Action Handler delegates the request to the corresponding Action class. Both Action Handler and Actions are all ordinary Servlets built for request delegation and communication to the business layer. One example of the problem caused by this pattern is huge amount of Action classes since, for each user operation, one action class needs to be created. In consequence, at least ten action classes must be created per Entity (create, retrieve, update, delete, and assign actions), this will increase dramatically, if there are more object types in the system.

IV. FIRST FRAMEWORK SELECTION

Based on the statistic from Google Trends [4] and Zero Turnaround's Java EE productivity report 2011 [5], these following seven most widely-used frameworks are selected for further research and analysis:

- Spring MVC (SpringSource.org)
- Java Server Faces (J2EE and JSR Standard)
- Wicket (Apache Software Foundation)
- Seam (JBoss)
- Struts2 (Apache Software Foundation)
- Tapestry (Apache Software Foundation)
- Stripes (Stripes)

Even though, web frameworks give you many benefits, but there are several criteria that should not be neglected. For example, testability, learning curve, community and support, tools and IDE, architecture and patterns, configuration complexity, and amount of artifact produced.

The following diagram shows the comparison result of the frameworks selected from the 1st framework selection, in the context of some part of the criteria explained in previous section.

V. REQUIREMENT GATHERING AND ANALYSIS

There were 5 attendances. All of the attendees are developers working in the MeDIC or the XAM project. All attendances were free to express their own opinions about the original prototype, expected characteristics of the new architecture, and prioritize all requirements.

The purpose of this requirements meeting is to get technical feedbacks and suggestions from the users of the original prototype and use those information as criteria for the 2nd frameworks selection and comparison. The selected frameworks will be analyzed in detail, implemented as prototype of the existing system, and evaluated in the following chapters. These following contents are the requirements:

1st priority (1)

According to the feedback from user, convention over configuration framework is preferred. One factor that affect the complexity is large files in the system especially, configuration files such as xml files. There are very few tools/IDE, which support xml auto-completion and verification.

Asynchronous Javascript And XML (AJAX) is a web development method used on the client-side to create
interactive web application. With AJAX, web application can send and retrieve the data from web server asynchronously (in the background) without any sign of page refresh. For the existing system, any frameworks have AJAX support are advantageous, but the system must still working perfectly, even without AJAX.

Community and support also important. The chosen framework's community should be active and provides support.

2nd priority (2)

The framework should be Component-based and support inheritance architecture.

Reusability is always first factor to indicates the quality of the software system. For the backend and business logic, one way to obtains the reusability is through inheritance structure. Most of the web application view created by markup language, which does not support inheritance structure, or any mechanism that provides the user interface reusability. One way to accomplished the 2nd priority requirement is component-based architecture framework.

In component-based aspect, each page considered as a component, and each page can be constructed with multiple components, which means, each component can be reused to construct pages, which shared some similarity as many time as needed without code duplication.

Also, the framework should be able to manage the request delegation automatically. This request delegation is responsible for moving a particular request through the right class. In the original prototype, the ActionHandler Servlet is responsible for the manual request delegation. The framework, which support better flow control is needed.

3rd priority (3)

The framework with good IDE support such as auto-completion, UI builder, multiple project support is needed.

4th priority (4)

The framework should provide high testability.

VI. SECOND FRAMEWORKS SELECTION

<table>
<thead>
<tr>
<th>Frameworks/Requirements</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring MVC</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>JSF</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Wicket</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Seam</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Struts2</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Tapestry</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Stripes</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>JSP/Servelt</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

Fig. 3 Result from the output of second framework selection

For the artifact complexity (1), almost of the frameworks passed, except Spring MVC and Stripes. Spring MVC has a serious problem with very high configuration complexity (pure XML, configuration over convention), and Stripes has a small community and not actively developed.

For the component-based, inheritance structure architecture, and the request delegation supported (2), four frameworks have component-based architecture, which are JSF, Seam, Wicket, and Tapestry. Also, delegation and navigation supported can be achieved by Business Process Management framework integration such as jBPM. However, only Spring MVC and Seam supports perfect integration with jBPM, and only JSF and Struts2 have navigation rule. The navigation rule controls the flow of page navigation. Seam might be the most direct solution of this requirement since, Seam provides both navigation rule and jBPM to control page flow and business process flow.

Spring MVC and Stripes have serious problem with poor IDE support (3). Even Wicket has no official IDE support, but since Wicket composed of HTML templates and pure Java, any Java IDEs and HTML editor provides full support. Other frameworks passed this requirement especially, JSF and Seam.

After the requirements analysis, three frameworks were chosen to implement the prototypes. Those three frameworks are Wicket, JSF, and Struts2.

Originally, Seam was chosen instead of JSF because, it fulfills all requirements. Seam is a full-stack framework which integrates several frameworks together. For presentation layer, Seam provides various choices including JSF and Wicket. There are several advantages of Seam using JSF over stand alone JSF including performance improvement. Due to integration conflict between Seam environment and existing environment, the prototype implementation using Seam failed.

Most of the frameworks are not qualified. Spring MVC fails most of the requirements especially, almost first 3 priority requirements. Tapestry was precluded because of these serious problem about no backward compatibility supported. Stripes fails 1st and 3rd priority requirements because the project is not actively developed and has no official IDE supported. Since all other frameworks are precluded, Struts2 is the best choice amongst the frameworks left.

VII. PROTOTYPE IMPLEMENTATION

A. JBoss Seam

Seam provides alternative presentation layer development frameworks fully integrated with EJB covered backend. The EJB provided by Seam is the proposed architecture from the beginning, and two out of three presentation layer integration provided by Seam are the chosen frameworks (JSF, Wicket). The benefits of Seam integration other than the enhancements mentioned in chapter two is the ease of prototype development. Seam tied Wicket and JSF seamlessly to the backend and does not required any Action beans. Also, it is simple to develop the next prototype after the first one by changing only view technology without changing anything at the backend.

Unfortunately, Seam prototype implementation failed. The cause of the failure is suspected to be the incompatibility of Seam and IBM Websphere 7.0 Application Server (WAS). Even the JBoss community’s resource already stated the Seam installation on IBM Websphere 7.0 Application Server [6], but in practical use, it is not working in the existing environment.

In consequence, the prototype implementation plan changed back to the original plan, which is developed by the remaining prototype independently without Seam.
Java Server Faces (JSF)

JSF has a very complex architecture, which consists of several design patterns working behind the scene [7]. Developers only need to know five components: FacesServlet, UI Components, JSP, Managed Beans, and Navigation Rules.

The FacesServlet is the front controller of the system. All the requests from the client have been translated into mapped action and delegated to the corresponding page and the response from the system is sent back to the requested client. Every JSF page contains JSF components both basic components (JSTL tags, Facelets), which represent only standard HTML, or third party libraries (PrimeFaces, ADF Faces, Trinidad, IceFaces, RichFaces), which provide advance widgets like trees, grids, tabs, or personal custom components. Each component's value might bound to the value in Managed Bean, and some components have event listeners, which call the methods in the Managed Bean, when triggered. Those methods in the Managed Bean handle the communication with the backend. The Navigation rules manage all page flowsof the system.

Apache Wicket

Wicket's architecture is very simple. It is mainly composed of HTML templates bind tightly to POJOs with the same name. These paired HTML templates and POJOs represent web pages. Minimum number of artifact to build a page is two: one HTML page, and a POJO. In consequence, huge amount of artifacts are produced. However, Wicket's component-based architecture greatly reduces the complexity of artifacts. The developer may use basic Wicket components or even, create custom components using the same method as building a page. As a result, more than half of the artifact especially, HTML files have few simple lines of components' interface declaration, and POJOs are well-structured, highly reusable, and easy to understand with inheritance structure.

The flow is quite simple. The Request Cycle acts like a front controller, delegates the request to the corresponding page, then the user interface components in HTML template will be initialized by POJO bound to that HTML.

Apache Struts2

The goal of Struts is to separate the model from view and controller. Struts2 introduced Action bean as the controller to facilitate the writing of templates for the view or presentation layer (typically in JSP, but XML/XSLT, Tiles, FreeMarker and Velocity are also supported). The web application programmer is responsible for writing the model code, and for creating a central configuration file struts-config.xml that binds together with model, view and controller.

Interceptor is the core concept of Struts2, which provides great code redundancy reduction. Many Action beans share common concern. Several Action beans need similar input validation. Some needs a file upload to be pre-processed. Another Action beans might need double form submission protection. Interceptors can execute code before and after the Action bean is invoked. Features like double-submit guards, type conversion, object population, validation, file upload, page preparation, and more, are all implemented with the help of Interceptors. Each and every Interceptor is pluggable, so developers can decide exactly which features an Action bean needs to support. Also, if a lot of Action beans are plugged with the same pattern of interceptors, Interceptor stack can be defined in the struts.xml to provide even more convenient. For example, every Action beans which related to very important task need to apply double-submit guards, user session validation, and encryption interceptor. Instead of spending three lines to apply all required interceptor on every Action bean, registering an interceptor stack contained three required interceptors, and spent only a line for applying the stack to the Action beans is more productive.

RESULTS AND EVALUATION

From the current result, the most suitable framework to replace the proposed architecture is Apache Wicket. Struts2 is eliminated from the list because, it does not completely fulfill the first priority requirement (configuration complexity, community and supports) and does not fail the main second
The priority requirements were gathered, prioritized, and analyzed into the proposed architecture. The main reason for this was the light-weightiness of Wicket. Many sources did the comparison between JSF and Wicket memory usage, and some sources even proved that even JSF enhanced with Seam's JSF memory consumption optimization consumes more memory than stand-alone Wicket [8]. Combining with very low learning curve and simple architecture with complete separation of presentation layer and business logic layer, Wicket is the best choice amongst all frameworks to replace the proposed architecture.

IX. SUMMARY

This research paper indicates that Wicket is the most suitable J2EE-based presentation layer development framework for replacing the presentation layer of the proposed architecture. On the first step, seven frameworks were chosen based on popularity and general web framework criteria in order to thoroughly filter out the low quality web frameworks from this research paper scope. Those frameworks are Spring MVC, Java Server Faces (JSF), JBoss Seam, Apache Wicket, Apache Struts2, Apache Tapestry, and Stripes. The main criteria of this research paper is the requirements gathered from the developers who had experience in using any system based on the existing architecture (proposed architecture). The requirements were gathered, prioritized, and analyzed into four main prioritized requirements and became the main criteria for the second round selection. Against requirements and general web framework criteria, seven frameworks were analyzed and compared. The result was four frameworks were chosen for the prototypes implementation. Those frameworks are Seam, JSF, Wicket, and Struts2. The purpose of the prototypes implementation is to prove the concepts and frameworks comparison in the same environment. All prototypes' structures were based on the structure of the existing system, Customer and Contract Management System. The result is Seam prototype implementation which was a failure and the rest were successes. Finally, all results indicate that Wicket is the most suitable framework to replace the proposed architecture.

REFERENCES