Analysis of the Frame work Standard and the Respective Technologies

Vijayakumar C,
Software Engineer,
Zagro Singapore Pte Ltd,
e-mail: vijayakumar.c@zagro.com

***Abstract*— We propose a novel approach to specifying, documenting, and reasoning about object oriented frameworks. The novelty of our approach is in combining standard executable statements of a programming language (we choose Java as an example) with possibly nondeterministic specification constructs. A specification of the intended behavior given in this language can serve as a precise documentation for users of the framework and its extension developers. To illustrate the applicability of our method to specification of object oriented frameworks, we demonstrate how one can specify the Java Collections Framework which is a part of the standard Java frameworks (Struts, Spring, JSF)**

Keywords- JAVA/J2EE Frameworks - Struts, Spring, JSF

I. INTRODUCTION

Java 2 Enterprise Edition has excelled at standardizing many important middleware concepts. For example, J2EE provides a standard interface for distributed transaction management, directory services, and messaging. In addition, Java 2 Standard Edition (J2SE), which underpins J2EE, provides a largely successful standard for Java interaction with relational databases. However, the platform has failed to deliver a satisfactory application programming model. Many in the open source community, especially smaller vendors, have chosen the alternative of developing frameworks designed to simplify the experience of building J2EE applications. Popular frameworks such as Struts Framework, Spring Framework, and JSF Framework play an important role in many of today's J2EE development projects.

A) Apache Struts

Apache Struts is an open-source web application framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a model-view-controller (MVC) architecture. It was originally created by Craig McClanahan and donated to the Apache Foundation in May, 2000. Formerly located under the Apache Jakarta Project and known as Jakarta Struts, it became a top-level Apache project in 2005.

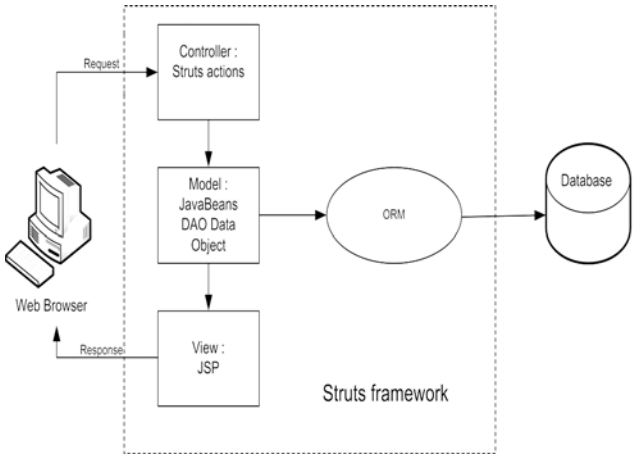

Fig-1: Apache Struts framework

B) Spring

The Spring Framework is an open source application framework for the Java platform.[1]

The first version was written by Rod Johnson, who released the framework with the publication of his book Expert One-on-One J2EE Design and Development in October 2002. The framework was first released under the Apache 2.0 license in June 2003. The first milestone release, 1.0, was released in March 2004, with further milestone releases in September 2004 and March 2005. The Spring 1.2.6 framework won a Jolt productivity award and a JAX Innovation Award in 2006.[2][3] Spring 2.0 was released in October 2006, and Spring 2.5 in November 2007. In December 2009 version 3.0 GA was released. The current version is 3.0.6.[4]

The core features of the Spring Framework can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the Spring Framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBean (EJB) model.

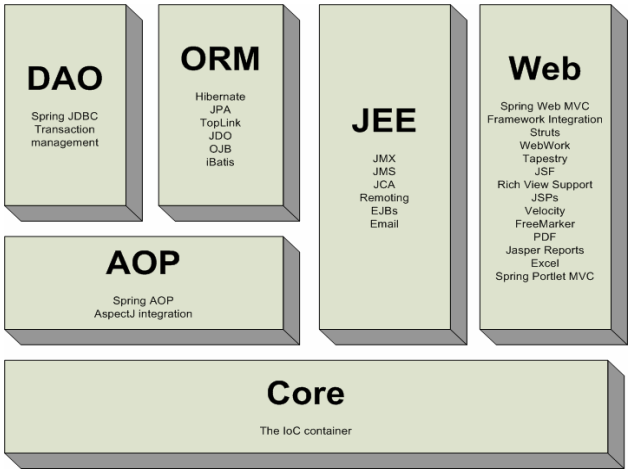

Fig-2: Enterprise JavaBean (EJB) model.

C) JSF (Java Server Faces)

JavaServer Faces (JSF) is a Java-based Web application framework intended to simplify development integration of web-based user interfaces.

JSF is a request-driven MVC web framework based on component-driven UI design model, using XML files called view templates or Facelets views. Requests are processed by the FacesServlet, which loads the appropriate view template, builds a component tree, processes events, and renders the response (typically HTML) to the client. The state of UI components (and some other objects) is saved at the end of each request (called stateSaving (note: transient true)), and restored upon next creation of that view. Several types of state-saving are available, including Client-side and Server-side state saving. Out of the box, JSF 1.x uses JavaServer Pages (JSP) for its display technology, but can also accommodate other technologies (such as XUL and Facelets). JSF 2 uses Facelets by default for this purpose. Facelets is a more efficient, simple, and yet more powerful view description language (VDL).

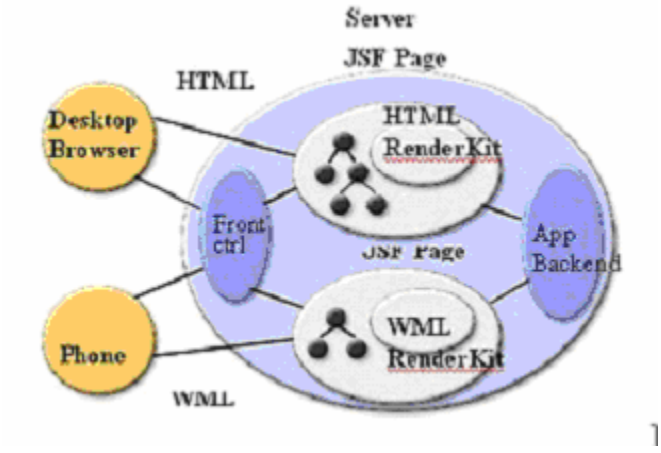

Fig-3: JSF Architecture

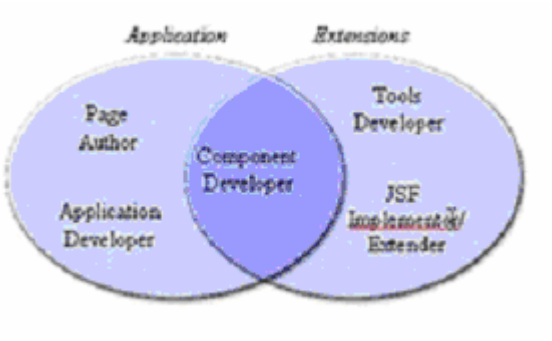

Fig-4: Developer Roles

II. COMPARISON OF FRAME WORKS (FW) BASED ON EVALUATION CRITERIA

Table 1 below presents the key elements of a comparison between the Frame works standards.

III. ADVANTAGES AND DIS-ADVANTAGES OF FRAME WORKS

A) Struts

Advantages:

- Realize MVC pattern, structure, clarity, so that developers focus on business logic implementation. Model View Controller (MVC2:- change in the model doesn't cause a change in the view)
- There is a wealth of tag can be used, Struts Tag Library's (Taglib), such as the flexibility to spend, you can greatly improve development efficiency. In addition, the current domestic JSP developers, apart from the use of commonly used built-in JSP tag, very few develop its own tag and perhaps Struts are a good starting point.
- Page navigation. Page navigation will be a future direction of development, in fact, to do so, allowing the system to the context more clearly. Through a configuration file, you can grasp the entire system between the various parts of the contact, which the latter has an immense amount of good maintenance. Especially when another group of developers to take over the item, the embodiment of this advantage may become more pronounced.
- Exception handling mechanisms provided.
- Database link pool management
- Supporting I18N
- Tiles: Common look and feel
- Validation Framework

Disadvantages:

- go to display layer, need to configure forward, each time to the show floor, I believe most of them are directly to the jsp, which involves the shift, need to configure forward, if there are ten display layer jsp, need to configure the ten second struts, and does not include some directory, file changes, modifications need to re-forward, note that each modified configuration, the request to re-deploy the entire project, while Tomcat such a server, it is also necessary to restart the server, if the business changes frequently complex systems, such as simple unimaginable. Now is the case, dozens of hundreds of individuals at the same time online using our system, everyone can imagine, I have little trouble.
- Struts in Action is necessary to thread-safe manner, it is merely an example of permit to deal with all requests. Action so used all the resources necessary for a unified synchronization, this would cause a thread safety problem.
- Test inconvenient. Struts each Action are coupled together with the Web layer so that it depends on the test Web containers, unit testing is also difficult to achieve. But there is Junit extensions of the instrument can Struts TestCase implementation of its unit tests.
- Types of conversion. Struts put the FormBean all data as String type; it can use the Commons-Beanutils instrument for the type of transformation. But its transformation is at Class level, but also the type of transformation cannot be configured. Type conversion error when the information back to users is also very difficult.
- The dependence on the Servlet too. Struts must deal with when Action needs to rely on the ServletRequest and ServletResponse, all it could not shake off their Servlet container.
- The front of the expression language. Struts integrates JSTL, so it is mainly the use of JSTL expression language to access data. But the JSTL expression language in the Collection and property aspects of the index appears to deal with very weak.
- Action to implement the control difficult. Struts create an Action; if it want to control the implementation of the order would be very difficult. Even you want to go to Writing Servlet you achieve the functional requirements.
- Action prior to and after treatment. Struts deal when Action is based on the class of hierarchies; it is very difficult in the action before and after treatment instructions.
- Not enough to support the case. In the struts, the actual form is a Form corresponding to a Class Action (or DispatchAction), In other words: In the Struts actually are a form can only correspond to a case, struts this case means that for the application event, application event and component event is compared to a coarse-grained case.
- No rich UI Components.
- All the logic is in the action classes and so tightly coupled
- Struts screens can't be rendered in different manner on a computer screen and pda.

B) Spring

Advantages:

It is an open source project, and currently very active; It is based on IoC (Inversion of Control, inversion of control) and the AOP framework of multi-layer system framework j2ee, but it does not force you to be at every level must be used in Spring because it is a good modular, allowing you to choose according to their own needs to use it to a module; it is a very elegant implementation of the MVC, the different data access technology to provide a unified interface, using IoC can easily make the bean implementation of assembly, provided a concise and, accordingly AOP implementation Transaction Management, and so on merits:

- Spring can effectively organize your middle tier objects, regardless of whether you choose to use the EJB. If you only Struts or other use of the API for the J2EE special framework, Spring to address the remaining questions.
- Spring can eliminate the common lot of projects on the excessive use of Singleton. In my experience, this is a big problem; it reduces the system testability and object-oriented level.
- Through a different application and projects consistent approach to deal with configuration files, Spring can eliminate a wide variety of custom property file format required. Once on a category to find what are the magic-like system properties or property they did not understand why this had to read Javadoc or even source code? With Spring, you only need to look at the type of JavaBean property. Inversion of Control of the use (discussed below) to help achieve this simplification.
- Put on the interface through programming rather than on the type of programming is almost reduced to the cost of NOT,
- Spring can promote the development of good programming habits.
- Spring is designed to allow applications to use it to create as small as possible dependent on his APIs. Spring applications at most business object does not depend on Spring.
- The use of Spring applications built unit testing easier.
- Spring enable the use of EJB implementation to become a choice rather than an inevitable choice for the application architecture. You can choose to use local EJBs or POJOs to achieve business interface, but will not affect the calling code.
- Spring assist you to solve many problems without the use of EJB. Spring can provide a replacement EJB objects, they apply to many web applications. For example, Spring can use AOP to provide declarative transaction management rather than through the EJB

container, if you only need to deal with a single database, or even do not need a JTA implementation.

- Spring data access to provide a coherent framework, whether it is using JDBC or O / R mapping products (such as Hibernate).
- Spring can really make you the most simple and feasible solution to solve your problem. And this is there is there is great value.
- Dependency injection: Directly we can use the DAO's in the JSP pages
- Very rich Spring tags
- View Agnostic: The JSP's can be replaced by Velocity, PHP and other things.
- It has different types of controllers like SimpleFormController, CommandController and all and each does a
- Specialized task.
- Avoid Forced Concrete Inheritance:
- Spring MVC does not force your model to inherit any custom classes related to Spring. It does all its work through interfaces. This helps in a way that once can inherit from a standard Java class. It also helps in testing, as it would rather difficult to test or reuse an object which is extending another custom one.

Disadvantages:

- No workflow like arrangement.
- The use of a small number.
- Java Server Pages (jsp) in to write a lot of code.
- The controller is too flexible
- Lack of a common controller.
- Spring is huge, fat and bulky - 160+ MB size of files, have to ship every time with my application archive.
- Documentation is cluttered and growing everyday
- Loads of XMLs needed- XML is good, if it is small and less number of files.
- Awful naming - RequestContext , ProviderManager – authentication class
- No clear difference in Code and Config - Simply java programmers spent equal amount of ‘coding XML’ as coding Java

C) JSF

Advantages:

- Big vendors (Oracle, IBM, JBoss, etc) backing JSF implementation like EJB. Can expect good level of support and quality components from these vendors.
- By design and concept it allows creating reusable components. That will help to improve productivity and consistency.
- Many quality and ready to use components are available from Apache, Richfaces, Infragistics, Oracle, etc.
- The concept of action and action listener for button invocation is good.
- Has very good support for EL expression that improves the user interface code readability.
- The concept the validator and converter is excellent. Unlike struts JSF keeps the validation logic very close to the component declaration.
- JavaScript codes are embedded as part of the component; this keep less confusion for developers and more re-usability on JavaScript code.
- Rich UI Components: - There would be classes for buttons and other things and we get a lot of rich UI functionality
- All the navigation logic is in the faces-config.xml file and so loose coupling.
- There is a renderer class so the look and feel of the jsp page would be different in the pda and computer screen
- JSF will give a identifier to each view and so the next time the request is made the view is already in the cache and so fast access.
- Very rich validations can be written and no need to create separate xml files for them likes struts. All things are done in the faces-config.

Disadvantages:

- There is no benchmarking report or promise from Sun Microsystems about the performance of JSF framework. By seeing their concept I believe it is not suitable for high performance application.
- The specification doesn’t consider bookmarking facility.
- Hardly a very few examples available for developing dynamic pages including new component and removing a component from a page based on business rule.
- Every button or link clicked results in a form post. That’s just wrong - why can’t I have true links like the web is supposed to? Form submission for page navigation make complex coding for simple requirement like Cancel button. Read here to know the work around.
- Data table component requires same data from bean on restore view phase. If the data retrieved from database, this will have impact on performance. Click here to know more about this issue.

- There is no tight coupling between managed bean and phase listener. This is a major drawback of JSF which makes JSF phase listener feature unusable.
- Default error message is not good. Need to customize the default error message.
- Not Scalable. It uses session object to store the component state across the request. In server farm environment it is too costly to replicate the session data.
- Dependent on using JSPs like Struts
- There is no Tiles framework like Struts for look and feel.

IV. SUMMAR OF FRAME WORKS

A) What's your project's 'sweet spot'?

Struts:

- Although Struts 1 has essentially been replaced by Struts 2 (also in this list), and is not recommended for projects commencing today, it is still in use and extremely popular with enterprises so we've included it here.
- Struts 1 provided an excellent fit for projects that involved form submission. When Struts was first released developers migrated to it from home grown servlet frameworks. Struts gave the industry a defacto standard for developing browser-based applications.
- Two notable sites using Struts are http://www.virgin-atlantic.com and https://www.21st.com/.
- Struts 2, WebWork renamed, are an excellent web application framework of the "action controller" family. Due to similarity of paradigm, those tens-of-thousands of developers out there that are familiar with Struts 1 can learn Struts 2/WebWork in a matter of a day, realizing huge productivity increases. At the time my team made that transition about 3 years ago, we estimated about a 40% productivity increase in work on the web application. This is due to the (what was at the time WebWork was built) "next generation" features that WebWork introduced upon the "action controller framework" paradigm. Validation, auto-form handling, type conversion, interceptors, etc. all add up to having a real usable tool set, without having to entirely change the way you "think".
- If a team is coming from Struts 1 and looking for a quick worthwhile change, WebWork/Struts 2 is for you. It vastly improves upon Struts 1, and has features that are quite competitive against the newer frameworks.

Spring MVC

- Projects that use – or intend to use – Spring
- Projects that need to expose business logic as HTTP addressable URLs
- Projects that must provides multiple view rendering techniques

JSF

- Separation of MVC layers reduces developer effort.
- Framework addresses every complex aspect of Web development.
- Developers can create new components or use existing ones.
- Many sources for additional components.

B). What sorts of projects does this package not work as well for?

Struts

- Projects that will be deployed to a 1.5 JVM
- A lot of boilerplate code needs to be written for simple tasks
- A Struts application will always have more classes than an equivalent application written in a different framework

Spring MVC

- Projects with a complex business model

JSF

- No native AJAX support
- No support for RESTful pages
- Standard component set too limited for complex enterprise applications

c) What is the future of this project?

Struts

- Bug fixes
- Struts have no support for AJAX.
- Developers also think that they will be required to use JSPs.
- Struts 1.x have a very limited future. Whilst officially new functionality is being added to the 1.x releases if backward compatibility can be maintained, in reality only very minor functionality improvements are being added. Third-party tools (e.g. XDoclet, IDE support) are mature and unlikely to offer any significant changes from the functionality offered today.

Spring MVC

- Easier configuration and better validation
- Spring MVC and Spring WebFlow are not competing projects.
- Spring MVC continues to benefit from the improvements made to the core Spring framework.

Future improvements include easier configuration and better validation.

JSF

- Community is writing the new JSR 314
- Artifacts from Shale will probably make it into JSF 2.0
- The JSF 2.0 community is writing the new JSR 314. Interested parties can find it here: http://jcp.org/en/jsr/detail?id=314. This effort solves a number of the problems addressed in other questions.
- Some artifacts from Shale will probably make it into JSF 2.0. For instance, there are the Shale-Tiger extensions that allow the usage of annotations to register a managed bean.

Conclusion

Java framework is a set of related classes and other supporting elements that make Java web application development easier by supplying pre-built parts [Ford, 2004]. Frameworks become popular because they ease the complexity and enable web developers to write at a high level of abstraction without compromising the application content. However, to take full advantage of frameworks` benefit, necessary studies must be done to find out the optimum framework applied to the application. The main purpose of this thesis was to help web developers or technique managers gain deep insight of four popular Java web framework: Struts1.X, WebWork2.2X, Tapestry 4 and JSF1.2 through the comparison conducted in this these and try to conclude the best suited web application types of these frameworks. In order to achieve the research result several steps are taken:

First, the four chosen framework's infrastructure were investigated separately, the content includes framework introduction, framework key components and the lifecycle of the framework. The aim of investigating infrastructure of different frameworks was to reveal the general view of each framework and lays the understanding foundation for the web features comparison. After the research on the four chosen frameworks I concluded different framework's typical characteristics and some advantage and disadvantage at the end of chapter three.

Second, I selected six basic but essential web features, which are Navigation rules, validation mechanism, Internationalization, Type conversion, IoC support and Post and Redirect, as the comparison yardstick for different frameworks. In the chapter five, I first briefly introduced the concept of the six web features and then described detailedly the six feature implementations of each framework. Meanwhile in order to provide practical support, I also combined the theoretical discussion with presenting a "Project Track" case study web application.

Third, I carried out a conclusion on framework researches. The advantage and disadvantages of each framework feature implementation were summarized and based on the research results of framework infrastructure investigation and feature comparison, the suitable web application types for four chosen frameworks were also deduced at the end.

Choosing a suitable framework that best match the application from numerous peer software products is a time-consuming and complicated process since there is no one

web framework that will be best for all projects. Though the research result of this thesis, web developers or managers have the great opportunities to rapidly master the essential of the four popular frameworks and make the accurate framework choice according to the recommendation application types showed in this thesis. They may also make the judgment by themselves based on their project requirements and presented framework web feature implementation analysis.

Because of the constant web feature improvement and introduction of new features of frameworks, the future work of this thesis may focus more on feature analysis amendment, and framework recommended application types in this thesis should also be adjusted to make the research result consistent with framework status.

REFERENCES

[1] Shan, Tony (2006). "Taxonomy of Java Web Application Frameworks". Proceedings of 2006 IEEE International Conference on e-Business Engineering (ICEBE 2006). http://portal.acm.org/citation.cfm?id=1190953. Retrieved 2010-10-10.

[2] http://shale.apache.org/

[3] http://www.andygibson.net/blog/index.php/2008/08/28/is-spring-between-the-devil-and-the-ejb Spring VS EJB3

[4] "Pitchfork FAQ". http://www.springsource.com/web/guest/pitchfork/pitchfork-faq. Retrieved 2006-06-06.

[5] http://houseofhaug.wordpress.com/2005/08/12/hibernate-hates-spring Hibernate VS Spring

[6] Johnson, Expert One-on-One J2EE Design and Development, Ch. 12. et al.

[7] http://www.theserverside.com/tt/articles/article.tss?l=SpringFramework

[8] http://blog.oio.de/2010/11/24/jsr-000314-javaservertm-faces-2-1/

[9] http://it-republik.de/jaxenter/news/Was-ist-neu-in-JSF-2.1-057653.html (German)

[10] Ryan Lubke (5 December, 2007). "Project Mojarra - the JSF RI gets a code name". http://blogs.sun.com/rlubke/entry/project_mojarra_the_jsf_ri.

[11] Bergsten, Hans. "Improving JSF by dumping JSP". O'Reilly. http://onjava.com/pub/a/onjava/2004/06/09/jsf.html. Retrieved 18 August 2011.

Project	**Language**	**Ajax**	**MVC FW**	**MVC Push/Pull**	**i18n&i10n**	**ORM**	**Testing FWs**	**Security FWs**	**Form Validation FWs**
Struts	Java	Yes	Yes	Push & Pull	Yes	Yes	Unit Test	Yes	Yes
Spring	Java	Yes	Yes	Push	Yes	Hibernate, iBatis, etc	Yes, mock objects& unit test	Spring Security (Formerly Acegi)	Commons validator
Java Server Faces	Java	Yes	Yes	Pull	Yes	Yes, with extensions	JUNIT	Yes	Native validators and integration with Bean Validation

Table-1: Comparison of Frames works