4. Results and Evaluation

Contents

In previous chapters, three frameworks were chosen for the prototype implementation based on the framework capability against basic criteria and prioritized requirements. However, the information alone is not enough, so the prototype implementations were brought up for the proof of concept.

In this chapter, the results from prototype implementation are summarized against the requirements and basic criteria. The major benefits and drawbacks are also considered in framework analysis (i.e. Tapestry's backward compatibility problem eliminates Tapestry from being chosen). Finally, only one framework out of three frameworks, which were implemented as prototypes, will be state as the most suitable solution for this problem.

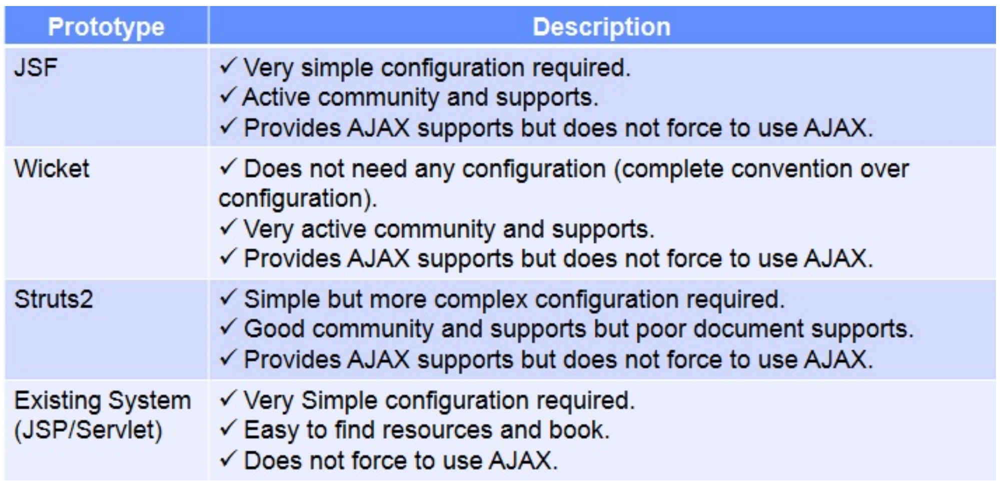
The analysis method is to compare each prototype including the existing system against the requirements in the form of measure factors, the characteristics and properties of the ideal solution. The fifth priority requirement will not considered in this section but will be mentioned in the next chapter, summary.

4.1. 1st Priority Requirement

Measure factors: low configuration complexity (convention over configuration), fine-grained system, active community and project development, AJAX support is a plus (must not force to use AJAX)

For the 1st requirement, Wicket prototype fulfilled this requirement completely. These following properties make Wicket becomes the most suitable framework for this requirement:

- There is no further configuration required.
- Half of the artifacts are low complexity HTML templates and other artifacts is easy to understand.



Prototype	Description
JSF	✓ Very simple configuration required. ✓ Active community and supports. ✓ Provides AJAX supports but does not force to use AJAX.
Wicket	✓ Does not need any configuration (complete convention over configuration). ✓ Very active community and supports. ✓ Provides AJAX supports but does not force to use AJAX.
Struts2	✓ Simple but more complex configuration required. ✓ Good community and supports but poor document supports. ✓ Provides AJAX supports but does not force to use AJAX.
Existing System (JSP/Servlet)	✓ Very Simple configuration required. ✓ Easy to find resources and book. ✓ Does not force to use AJAX.

Figure 4.1.: 1st Priority Requirement Result

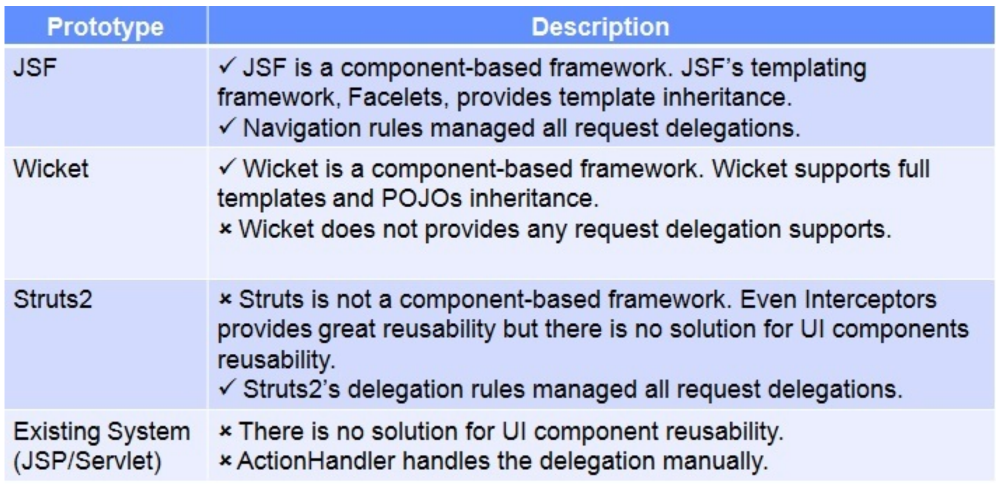
- It is one of the most active community and development team.
- It provides very good AJAX supported.

Other two frameworks and the existing system are working fine on this requirement. JSF requires only few lines of simple configuration for every Managed Beans (no longer required in JSF 2.0) and only few lines of Action class register for the existing system, while Struts2 required quite more effort on Action Beans mapping and Interceptors management. Moreover, even Struts2 has quite poor documentation and not friendly to search engine, but all frameworks have good supports from their community and provide AJAX supported without forcing developers to use AJAX. All of them fulfilled the 1st requirement lead by Wicket.

4.2. 2nd Priority Requirement

Measure factor: component-based architecture with inheritance-supported architecture or any solution that provides user interface component reusability provides user interface flow management supported or provides integration with other flow management technology.

For this requirement, by integrates with Facelets, JSF is the most suitable framework, which fulfills this requirement. JSF is a component-based framework, which might not originally provides template inheritance structure, but the official template technology build specifically for JSF, Facelets provides great user interface reusability with template inheritance. Facelets makes JSF comparable to Wicket, which is the natural-born component-based framework with component inheritance structure, while Struts2 is not a component-based framework and does not provides user interface inheritance structure or any strategy for user interface reusability at all. However, Wicket does not provide any flow



Prototype	Description
JSF	✓ JSF is a component-based framework. JSF's templating framework, Facelets, provides template inheritance. ✓ Navigation rules managed all request delegations.
Wicket	✓ Wicket is a component-based framework. Wicket supports full templates and POJOs inheritance. × Wicket does not provides any request delegation supports.
Struts2	× Struts is not a component-based framework. Even Interceptors provides great reusability but there is no solution for UI components reusability. ✓ Struts2's delegation rules managed all request delegations.
Existing System (JSP/Servlet)	× There is no solution for UI component reusability. × ActionHandler handles the delegation manually.

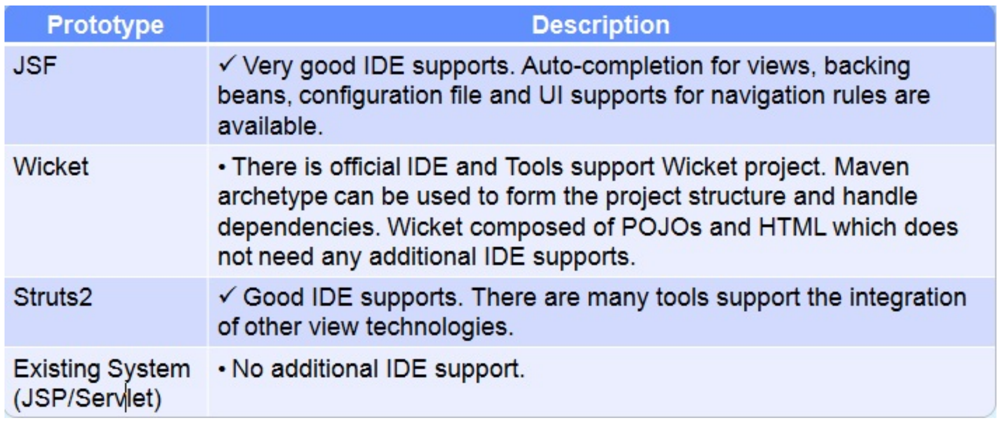
Figure 4.2.: 2nd Priority Requirement Result

management technology or even other flow management technology integration, while both Struts2 and JSF supports user interface flow management. JSF's user interface flow management, Navigation Rule is perfectly suits this requirement because of build-in Navigation Rule's diagram-like graphic user interface greatly simplify ease of use and configuration complexity, while Struts2 needs more effort on XML-based flow control.

The existing system fails both sub-requirements. There is no strategy for user interface reusability and an Action Handler class does the flow management manually.

4.3. 3rd Priority Requirement

Measure factors: good tools and IDE supports.



Prototype	Description
JSF	✓ Very good IDE supports. Auto-completion for views, backing beans, configuration file and UI supports for navigation rules are available.
Wicket	• There is official IDE and Tools support Wicket project. Maven archetype can be used to form the project structure and handle dependencies. Wicket composed of POJOs and HTML which does not need any additional IDE supports.
Struts2	✓ Good IDE supports. There are many tools support the integration of other view technologies.
Existing System (JSP/Servlet)	• No additional IDE support.

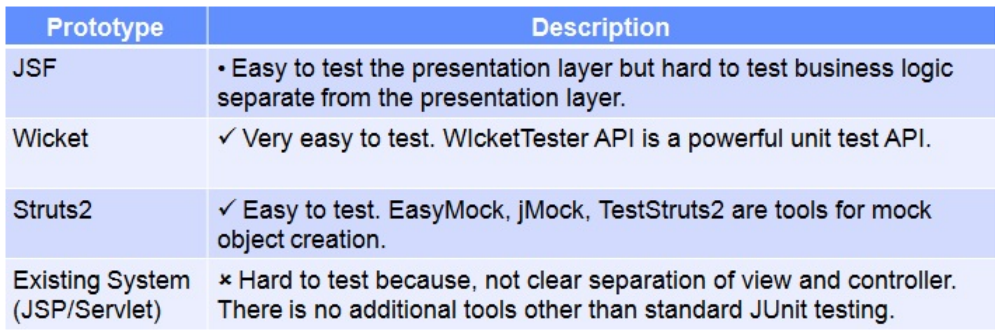
Figure 4.3.: 3rd Priority Requirement Result

JSF and Struts2 have very good tools and IDE supports, which provides auto-completion, beans manipulation, and graphical configuration supported. If based on famous Java IDE, Eclipse and its plug-ins, JSF supports are already included in every standard version of Eclipse. Whereas, Strus2 needs additional plug-in, MVC Web Project to handles the configuration file with node-like graphic user interface.

Wicket has no official IDE or tools supported, but because of Wicket's simple architecture, official tools and IDE are not necessary. Unlike JSF and Struts2, which plenty of custom tags, either user custom or 3rd party tags and their parameters are used in the view; only simple template Wicket tags are declared in HTML template backed with POJOs. Wicket and existing system does not need any additional tools and IDE supports which makes all frameworks fulfill this requirement equally.

4.4. 4th Priority Requirement

Measure factors: easy to do the testing.



Prototype	Description
JSF	• Easy to test the presentation layer but hard to test business logic separate from the presentation layer.
Wicket	✓ Very easy to test. WIcketTester API is a powerful unit test API.
Struts2	✓ Easy to test. EasyMock, jMock, TestStruts2 are tools for mock object creation.
Existing System (JSP/Servlet)	× Hard to test because, not clear separation of view and controller. There is no additional tools other than standard JUnit testing.

Figure 4.4.: 4th Priority Requirement Result

For this requirement, Wicket and its specific unit testing API is the most suitable solution. Wicket Tester is a unit testing API for Wicket applications without the need for the Servlet container, which is included in Wicket core library. Struts2 also provides many alternatives for unit testing outside the container such as EasyMock, jMock, and TestStruts2. JSF has problem in separation of view and business logic and the existing system does not do well in this requirement because of separation of view and business logic also.

There are also other factors that affect the decision-making such as JSF's performance issue, Struts2's Interceptor, Wicket's extra active community and light-weightiness. Even these factors might not related to the requirement directly, but these benefits and drawbacks effect the quality of the framework and should be considered.

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 requirement (component-based, inheritance structure supported framework). Even Struts2 is widely used and Interceptors provides great reusability, but compared to Wicket and JSF, Struts2 does not answer our question.

The efficiency of Wicket and JSF are proximate. Both of them fulfill most of the requirements. Wicket completely fulfill first and fourth priority requirements with zero configuration, very vibrant community, and powerful unit testing API, while JSF also fulfill first requirement with some artifact complexity and few simple configurations. However, JSF has some problem on testing each layer separately. Facelets and navigation rule fulfilled the second priority requirement of JSF with template inheritance and user interface flow management while Wicket was completely fulfilling the component-based inheritance structure without any extra technology, but Wicket does not provide user interface flow management at all. The third requirement JSF and Wicket are proximal. JSF has very good tools and IDE supported, while Wicket does not need any additional tools or IDE.

Since the result from comparing the requirements alone cannot indicates which one is the most suitable framework to replace the proposed architecture, other important factor which needs to be considered. The reason why Wicket was chosen is JSF has a very serious issue on very high memory consumption rate, while one of the Wicket's prominent points is its light-weightiness. Many source did the comparison between JSF and Wicket memory usage, and some source even proves that even JSF enhanced with Seam's JSF memory consumption optimization consumes more memory than stand alone Wicket [Tho11]. 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.