Integrated Architecture for Web Application Development Based on Spring Framework and Activiti Engine

Xiujin Shi，Kuikui Liu，Yue Li
School of Computer Science and Technology
Donghua University
Shanghai, China
sxj@dhu.edu.cn，liuk1989@yahoo.cn，frankyueli@dhu.edu.cn

***Abstract*—With the prevalence of the Java web application development, there are a lot of frameworks to make the development of Java web application easier, most of them are based on MVC (Model-View-Controller) design pattern, e.g., Spring framework, but they didn't pay enough attention on business logic. In this paper, an integrated architecture has been presented to integrate several frameworks in order to facilitate the process of building a web application. Moreover, this paper present a new annotation method for the configuration of the Spring framework instead of the XML files as we can see in most of the Java web project. This approach is implemented in the development of Taori e-commerce website project. The proposed architecture help the developer classify the project into several part, i.e., Activiti part and Spring part, This approach improves the maintainability and reusability of the application.**

Keywords— MVC, Spring, Acitviti, annotation, E-commerce

I. INTRODUCTION

Since the 1940s when the first electronic general-purpose computer ENIAC (Electronic Numerical Integrator and Computer) was created, people have been working hard on the information technology for over 60 years. For now, information can be spread through the Internet conveniently to every corner of the world. However, Web development is a very complex issue in these days. The desire of the Internet service providers and terminal users is becoming more and more complex and need high performance application [1]. To reach the demand of these Web users, software developer have made a lot of effort to figure out how to build high performance web application efficiently.

One of the most popular methods to build web application is Servlet, which is a Java-based server-side web technology [1].

Servlet can be generated automatically from JavaServer Pages (JSP) by the JavaServer Pages compiler. The difference between Servlet and JSP is that Servlet typically embeds HTML inside Java code, while JSPs embed Java code in HTML. While the direct usage of Servlet to generate HTML has become rare, the higher level MVC web framework in Java EE such as Spring MVC still explicitly uses the Servlet technology for the low level request/response handling via the DispatcherServlet class.

There are several mature and excellent framework or design pattern which are based on Servlet, e.g., the SSH combined framework which achieve MVC mode [2]. And based on those frameworks, there are some particular architecture proposed by researcher [3].In these frameworks, Spring framework has been used as a objects container frequently [4-5]. The IOC(Inversion of Control) of Spring decouple the dependence between classes, this is because the loosely coupled objects are easy to test and show higher level of reusability[6-7].But, these frameworks which based on MVC pattern didn't pay enough attention on the implementation business logic level, as a result , developer have to implement the business logic themselves in these framework, Currently, the workflow engine can be separately used by developer to help achieving business logic [8]. Furthermore, developer can even build a Platform-As-A-Service application business process in cloud-computing environment [9].

The rest of this paper is organized as follows. Section II starts with some related works in development pattern which aim to improve the efficiency of development of web application. In section III, we propose our web development architecture and describe its innovation in detail. As we mentioned before, this integrated architecture is used in the Taori project, which is a Java web application to manage the trade between client from china and Yahoo shop of Japan, a discussion of the benefits of using this architecture follows in Section IV. Finally, we summarize the conclusions and discuss future work in Section V.

II. RELATED WORK

There are a large amount of frameworks and technologies on Web application development; most of them are based on MVC pattern. On the other hand, for the business logic level, there are many workflow engines based on the workflow theory, e.g., JBPM and Activiti, they improve the efficiency of the development of the business logic level. Following are some related works in these fields.

We review the related work in four categories: MVC pattern, Spring MVC, BPM and Activiti engine, Java annotation.

A. *MVC*

MVC present the Model-View-Controller design pattern, which separates the representation of information from the user's interaction. The model consists of application data and business logic, while the controller mediates and converts input to commands for the model or view. By doing this, the model gather the application data and business logic together, then the user interface and the interaction which data is involved in can be modified without redeveloping the business logic. The central idea behind MVC is code reusability and separation of concerns.

B. *Spring web MVC*

Like Struts, Spring MVC is a request-based framework. The framework defines strategy interfaces for all of the responsibilities which must be handled by a model request-based framework. Each interface must be simple and clear. This feature helps developers write their own implementations easier.

The DispatcherServlet class is the front controller of the framework and is responsible for delegating control to the various interfaces during the execution phases of a HTTP request.

Central of the Spring Framework is its inversion of control container, which provides a consistent method of configuring and managing Java objects using reflection. The container is responsible for managing object lifecycles: creating objects, calling initialization methods, and configuring objects by wiring them together.

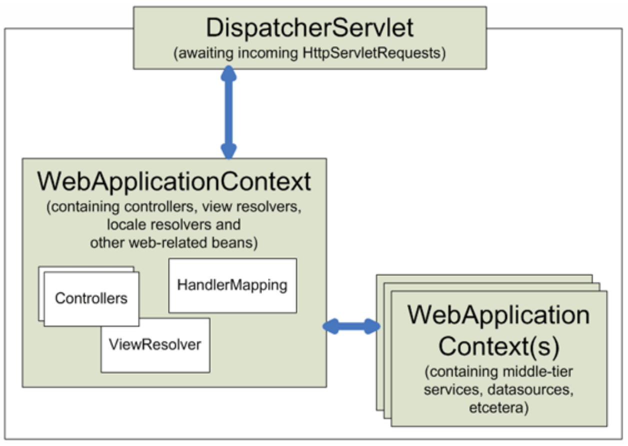
DispatcherServlet
(awaiting incoming HttpServletRequests)
WebApplicationContext
(containing controllers, view resolvers, locale resolvers and other web-related beans)
Controllers
HandlerMapping
ViewResolver
WebApplication Context(s)
(containing middle-tier services, datasources, etcetera)

Fig. 1. Context hierarchy in Spring web MVC

As we can see in Fig.1, the DispatcherServlet class initializes the Spring container called WebApplicationContext which contains other components. Objects created by the container are also called Beans. Typically, the container is configured by loading XML files containing Bean definitions which provide the information required to create the beans, but the usage of annotation to configure the container is more and more popular, it makes code clear and easy to maintain.

Objects can be obtained by means of Dependency lookup or Dependency injection. Dependency lookup is a pattern where a caller asks the container object for an object with a specific name or specific type. Dependency injection is a pattern where the container passes objects by name to other objects, via constructors, properties, or factory methods.

C. *BPM and Activiti Engine*

Business Process Management（BPM）, offers an approach to aligning an organization's business processes with the requirements and needs of clients. It promotes business effectiveness and efficiency while striving for innovation, flexibility, and integration with technology.

Activiti is a light-weight workflow and Business Process Management Platform targeted at developers and system administrators. Its core is a super-fast and rock-solid BPMN 2 process engine for Java. It's an open-source project and distributed under the Apache license. Activiti runs in any Java application, on a server, or on a cluster in the cloud. Activiti is actually a suite of following applications that work together:

- Modeler: a web-based graphical workflow authoring interface based on Signavio
- Designer: an Eclipse plug-in for developing workflows
- Engine: the core workflow processor
- Explorer: a web tool to deploy process definitions, start new process instances and carry-out work on workflows
- Cycle: a web app for collaboration between business users and software engineers

D. *Java Annotation*

An annotation in the Java language that added into Java from JDK1.5 is a form of syntactic metadata that can be added to Java source code. It can be used on classes, methods, variables, parameters and packages. Java annotations can be reflective and if developer wants to, it can be retained by the Java Virtual Machine for the applications achieving the class information at run-time.

Following are three annotations samples which are built into Java language:

@Override: Checks that the function is an override.

@Deprecated: Marks the function as obsolete.

@SuppressWarnings: Instructs the compiler to suppress the compile time warnings specified in the annotation parameters.

An annotation will cause a compile time error when certain circumstance, which is defined by the author of the annotation, happens.

Frameworks often use Annotation to apply behaviors to classes and methods that must otherwise be declared in an external source, such as XML configuration files or programmatically.

Annotation processors which can be defined by developers will process the annotations and do something else when Java source code is compiled.

III. PROPOSED ARCHITECTURE

There are many methods which can use Activiti to manage the business progress; but for the better integration with other framework, we use Spring framework to integrate the bean of Activiti engine and other beans we used in the web application. The SpringApplicationCnotext like a huge bean factory which is about to build and manage all the Java objects such as creating these objects, calling initialization methods, and configuring these objects by wiring them together, which owe to the feature of reflection of Java.

Fortunately, the Activiti leverage the parsing and dependency injection of Spring internally, therefor the integration of Activiti and Spring is very easy and efficient. Hence, we use the Spring framework to manage the Activiti engine class as a Java bean, just like a regular Spring bean. The start point of the integration is the class called "org.Activiti.Spring.ProcessEngineFactoryBean". This bean takes a process engine configuration and creates the process engine. To do that, we make declaration of "processEngineConfiguration" by adding several lines of code in Spring’s configuration file as we can see in Fig.2.

```
 <bean id="processEngineConfiguration"
class="org.Activiti.Spring.SpringProcessEngineConfiguratio
n"> </bean>
 <bean id="processEngine"
class="org.Activiti.Spring.ProcessEngineFactoryBean">
    <property name="processEngineConfiguration"
ref="processEngineConfiguration" />
 </bean>
```

Fig. 2. Configuration of Spring Acitviti Engine

In our architecture, we use Spring framework to contain the whole web application beans as a web context, and use Spring MVC to split model, view and controller. Developer can decide which function should be included in the business process management engine (as we know, in this case, it is Activiti engine) or implemented by developer themselves. The whole architecture is displayed in Fig.3.

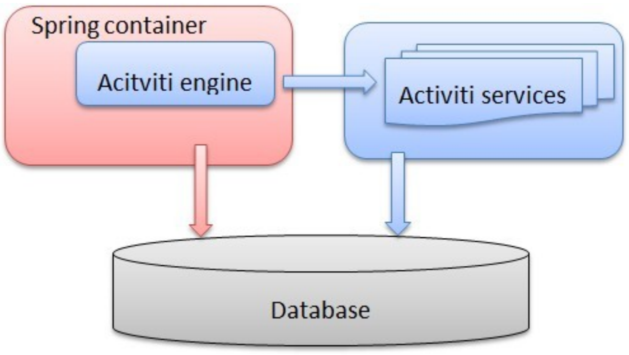

Fig. 3. Architecture of the integration of Activiti and Spring

As we can see in the Fig.3, both beans which used by Activiti engine and the other beans will be managed by Spring in the WebApplicationContext. When Activiti engine call those beans, it doesn't care about who initialized those beans, i.e., dependency injection, as we discuss in section 2.3, this method can really reduce the work of development. Developer can control not only the various dependencies and configuration values that are to be plugged into an object that is created from a particular bean definition, but also the scope of the objects created from a particular bean definition. This feature is really powerful and flexible, e.g., developer can add a @Scope("prototype") annotation, as a result, Spring will create a new bean instance every time a request for that specific bean.

Then, there was another problem during the development of the Taori project, which is that there are so many XML files required to maintain e.g., the configuration file of Activiti, web.xml, Spring configuration files and some additional XML files. Nevertheless, these XML files usually are very big and difficult to read. All these XML files will be maintained by several developers; if there is a little misspelling problem, it will cause the web application project fail.

For that reason, we propose an annotation solution. We define a bunch of annotations and an annotation processer which is named ConfAnnotationResolver, and we build a listener class which implement ServletContextListener interface to ensure that these annotations can be resolved before the Activiti and Spring initialization.

As a result of this annotation solution, there is only one XML file namely web.xml, in the web application project which is necessary for Java web application. At the same time, the configuration of the rest of the web project will be included in a class that implements the interface of ConfAnnotation, in which some necessary fields are defined for the Spring framework and Activiti engine.

The initialization process of our web application project is shown in Fig.4.

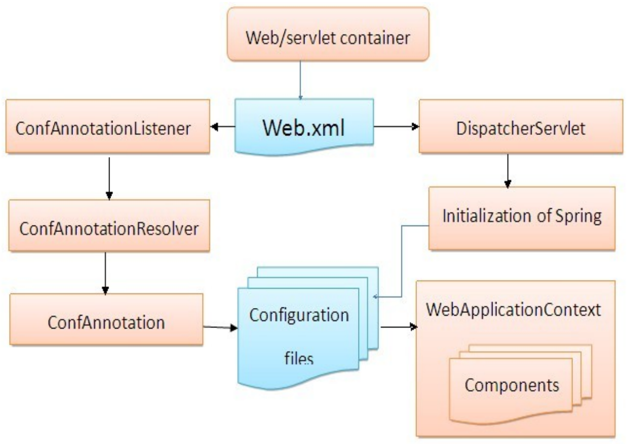

Fig. 4. Process of the initialization

When this web application starts running, the process of the initialization is shown as following:

Firstly, the web container reads the web.xml file, then the ConfAnnotationListener between <listener> and </listener>, will be instanced and loaded to the memory and be executed. It will instance the ConfAnnotationResolver class.

Then, the ConfAnnotationResolver will check the ConfAnnotation class, resolve the class by reflection and create XML files for the configuration of Spring framework or other framework used in the web application project.

In the end, the DispatcherServlet defined between <servlet> and </servlet> will be loaded and build the webApplicationContext, In the webApplicationContext, the Activiti engine will be initialized to build the whole Activiti context. Finally, the web application will start up.

Developers just need to maintain the configuration class which is so easy to do, and the ConfAnnotationListener will do the rest of the configuration process for us. For the business logic part, developers can use Activiti diagram editor to edit the business process, or use regular Java class to build the other logic which is not suitable for Activiti.

IV. Discussion

An integrated architecture for Web application development based on Spring framework and Activiti engine is proposed and discussed in this paper, which help Web developers to build highly maintainable and strong code. By using this approach in the Taori project, we finished the development work half month earlier than we expected at the first place. Moreover, we summarized the benefits that we gained from this approach as follows:

A. *Better Comprehensibility*

The inversion of controller container provided by Spring framework in this approach allow developer do not need to care about the instantiation management of the classes in a project. When we want to use an instance of a certain interface, we just need to add a "@Autowired" annotation, and the Spring framework will take it from there.

B. *Better Maintainability*

The configuration is strong and easy to modify. Developers don't need to check the spelling problem. Developers need only focus on the application itself. These problems will be checked by compiler, and exposed in compilation time when there is a compilation error.

C. *Better Reusability*

Thanks to the highly loosely coupling of the architecture, it has higher reusability than other frameworks. Any certain process we use the architecture to define can be reused in any other project, if only the project use BPMN 2.0 standard.

D. *Better Testability*

Since Activiti is an embeddable Java engine, writing unit tests for business processes is as simple as writing regular unit tests.

V. Conclusion and Future work

In this paper, we propose new development architecture for J2EE application. This architecture improves the efficiency of development. By using this architecture, software developer can build high-performance web application easily and efficiently. Then, we used this architecture in the Taori e-commerce purchase website project. With the effective of the integrated architecture for the development of the large scale application, we found that this architecture not only shorten our development cycle, but also improve the scalability of our project, for example it reduced the cost of future maintenance and upgrading.

In the future, we plan to continue our study on the optimization of this integrated architecture to make the development work easier. Moreover, we will use hibernate framework to do the ORM (Object/Relation Mapping) part to get higher performance. Furthermore, in the future, we also plan to use another framework to optimize the performance.

References

[1]. P.Gupta, and Prof.M.C.Grovil; "MVC design pattern for the multi framework distributed applications using XML, Spring and struts framework", International Journal on Computer Science and Engineering. Indian : engg journals publications ,Vol. 02, No. 04, 2010, pp. 1047-1051.

[2]. Y.Ren, T.Xing, Z.Xing, and J.Zheng; "Application research for integrated SSH combination framework to achieve MVC mode", International Conference on Computational and Information Sciences Chengdu, 2011, pp.499-502.

[3]. C.Zhao, M.Jiang, and Z.He; "The design of e-commerce system architecture based on struts2, spring and hibernate", International Conference on Computational and Information Sciences Hangzhou:IEEE, 2010, pp. 3251 – 3254.

[4]. F.Xue, F.Liang, S.Xu, and B.Wang; "Research on spring MVC framework based web and its application", Journal of Hefei University of Technology, Hefei , Vol.35 No.3, 2012, pp.337-340.

[5]. T.Bak, B.Sakowicz, and A.Napieralski; "Development of advanced J2EE solutions based on lightweight containers on the example of e-department application", International Conference Mixed Design of Integrated Circuits and System, Gdynia ,2006, pp.779-782.

[6]. J.Souer, and M.Mierloo; "A component based architecture for web content management:runtime deployable webmanager component bundels", International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science, Yorktown Heights, 2008, pp.366-369.

[7]. M.Mela, B.Sakowicz, and J.Chlapinski; "Advertising service based on spring framework", International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science , Lviv-Slavsko , 2008, pp.406-408.

[8]. Dr. Franz, J. Fritz, and H.Lau; "Workfow requirements for enterprise applications", Compcon Spring '94, Digest of Papers, San Francisco, 1994, pp.224-229.

[9]. Y.Zheng, J.Pang, J.Li and L.Cui; "Business process oriented platform-as-a-service framework for process instances intensive applications", International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai ,2012, pp.2320-2327.