OPEN ACCESS

International
Journal
Of Modern Engineering Research (IJMER)

Spring MVC: Companion of J2EE for Rapid Development of Web Applications

Vikas Kumar[1], Deepak Kasgar[2], Lokesh Kashyap[3]
[1]*M .Tech (C.S) (Assistant Professor, Bhagwant University, Ajmer, India)*
[2]*M.Tech (C.S) (Scholar, Bhagwant University, Ajmer, India)*
[3]*M.Tech (C.S) (Scholar, Bhagwant University, Ajmer, India)*

ABSTRACT: *With Java bean model the reusability concept is added, for java application. Industry developers identifies some problem in java bean model for application in real-time. Some of them are bean model because it does not support transaction support, security, distributing computing. To address the industry problem in March 1998 SUN has released Enterprise Java Bean (EJB technology). With EJB the developers identified that it is solving complex program, but the development of application are complex for developers. Rod Thomson and his team started experiment on ordinary java classes and provide enterprise services to industry application. In this paper, we briefly describe spring underlying architecture and present a case study using spring web MVC Framework.*

Keywords*- Spring Framework, IOC, J2EE, EJB, XML*

I. Introduction

In December 1996, Sun Microsystem introduced one component model for java called JavaBean Specification. With JavaBean model, reusability concept is added for java application. Industry developers identified some problem in java bean model for application in real time. To address this problem in March 1998 Sun released EJB. With EJB the real time services for application are added and many projects in industry used EJB technology for project development. With EJB the developers found it is solving complex program, but the development of application are complex for developers. So developers named EJB as **Heavy Weight technology.**

Spring Framework has got more popularity because of its principal:

1. Simplicity using POJO.
2. Testability using without 3rd party server or container.
3. Loose coupling without dependency injection.

Spring Framework is a complete and modular framework. It means we can develop either complete project only using spring framework or we can develop some selective operation of the project using spring framework. Spring framework modules are having some changes in spring 2.x and spring 3.x versions.

II. Spring Framework Has Multiple Features, Let Us Discuss These Features In Brief.

1. **Loose Coupling**: Loose coupling between objects is possible with following two things.
   - With interface orientation of dependency objects.
   - By applying dependency injection mechanism.

The purpose of interface orientation is that, the dependency classes will implement a common interface, so that the dependency classes contain some method name. The purpose of dependency injection is at runtime whatever the dependency object is required for a caller, then it will injected by without making any changes in caller class.

2. **Dependency Injection:** Dependency injection is a process of injecting the dependencies required for an object at runtime. **IOC (Inversion of Control)** is a design pattern and its implementation is dependency injection.

Types of Dependency Injection

- Setter Injection.
- Constructor Injection
- Interface Injection.

3. **AutoWiring:** In order to cut down the use of XML the framework has introduced autowiring facility. In case of autowiring we need to configure the dependencies of a bean automatically.

III. Major Spring Component

1. **Spring Core Container:** The central part of spring framework is spring container. It manages how the beans are created, configured and managed in a spring application. This module provides the fundamental/basics part of the spring framework called **Dependency Injection**.

2. **Spring AOP (Aspect Oriented Programming):** In this module spring framework provide separation of business logic services from the business operation. If the business services are implemented as part of business method then we have the following problems.
   I. Boiler plate code of services.
   II. Business class becomes complex
   III. Managing business services is complex.

   To reduce the entire above problem we got a new style of programming called as **AOP**.

3. **Data Access/Integration Module:** This module provides abstraction layer on existing JDBC ORM, JAXB (Java API for XML Binding), JMS and Transaction management. Spring JDBC is an abstraction layer of JDBC and it avoids boiler plate code of JDBC. This spring JDBC makes our database code as clean as simple.

4. **MVC/Remoting:** MVC is a design pattern which is commonly accepted approach for busisness web application web applications using java. Spring Framework has its own way for developing MVC based web application.

Spring has provided two ways of MVC
1. Based on Servlet.
2. Based on Portlet.

IV. Spring Architecture

The spring framework has its own MVC for building web applications. Spring framework can also be used with other Frameworks like iBatis, Hibernate, Struts etc. It can be configured with other technologies like JSP, Tiles, iText etc. Spring MVC separates the roles of the controller, model object, dispatcher Servlet and the handler object. Clear separation of objects and controllers makes them easier to customize.

Spring MVC Request Flow:

1. When a request is sent from a browse a front controller of spring MVC called **DispatcherServle**t traps the request.
2. DispatcherServlet takes the help of HandlerMapping class to find a suitable controller for handling the given request.
3. DispatcherServlet delegates the given request to the controller bean.
4. The method ( ) in which either business of intermediate logic defined for a request is executed.
5. The method of controller bean returns a ModelAndView class object.
6. The DispatcherServlet calls a ViewResolver bean to find the appropriate view for sending the response.
7. The DispatcherServlet forwards the request to a view.
8. Finally the response generated by view will be displayed on browser.

V. Spring Xml

Use of xml makes framework work faster, because these xml files are read by xml parsers and extracted value are stored in collections in form of <key, value> pair.
Advantages of XML

- Saves the development process.
- Any changes made in XML does not require recompile, reload of applications.

The web application starts its execution from Web.xml and navigates to the path which is defined in web.xml.

Sample of spring xml file:

```
<web-app version="1.0" xmlns="http//java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemalLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app-2_5.xsd">
<listener><listener-Class>org.springframework.web.context.ContextLoaderListener</listener-Class></listener>
<servlet>
<servlet-name>dispatcher</servlet-name><servletclass>
org.springframework.web.servlet.DispatcherServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>dispatcher</servlet-name>
<url-pattern>/send/*</url-pattern>
</servlet-mapping>
<welcome-file-list>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

ApplicationContext.xml :ApplicationContext containers are nothing but the classes which are implemented from ApplicationContext Interface. ApplicationContext containers are sub of BeanFactory containers because ApplicationContext interface is a subinterface of BeanFactory interface.

Comparatively ApplicationContext containers are greater than BeanFactory containers, because BeanFactory container can only provide dependency injection, but ApplicationContext containers provide dependency injection and also additional services like AOP, Messaging, Scheduling, eventhandling, internationalization.

Some implementation classes of ApplicationContext interface are:

1. ClassPathXmlApplicationContext class
2. FileSystemXmlApplicationContext class
3. XmlWebApplicationContext class

While building applications in a J2EE-environment ApplicationContext must be used.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-beans-2.5.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">
<bean id="superClass" class="packagename.SuperClass" />
<bean id ="subClass" class=" packagename.SubClass">
</bean>
<property name="superClass" ref="superClass"/>
</beans>
```

Dispatcher-servlet.xml

DispatcherServlet is a predefined servlet class which act as FrontController. A request flow of MVC application in spring will be taken care by DispatcherServlet. This servlet class we need to configure in web.xml file of web application. In an MVC application, a spring configuration will be read by the DispatcherServlet only.

```
<beans xmlns="http://www.springframework.org/schema/beans"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-beans-2.5.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">
<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="prefix">
<value>/WEB-INF/views/</value></property>
<property name="suffix"><value>.jsp</value></property>
</bean>
<bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
<property name="mappings">
<props><prop key ="/*">dispatchController</prop>
</props>
</property>
</bean>
<bean id="dispatchController" class=" packagename.DispatchController"></bean>
</beans>
```

VI. Architectural Benefit

J2EE applications tend to contain excessive amounts of "plumbing" code. Many J2EE applications use a distributed object model where this is inappropriate. The EJB component model is unduly complex. Many "J2EE design patterns" are not, in fact, design patterns, but workarounds for technology limitations. J2EE applications are hard to unit test. Certain J2EE technologies have simply failed.

Benefits:

1. The Spring is designed so that the most business objects in Spring apps have no dependency
2. It helps the developers to solve many problems using the EJB
3. The applications using Spring very easy to unit test
4. Spring can reduces the cost of programming to interfaces, rather than to the classes
5. The configuration management services can be used in any architectural layer in the runtime environment
6. The spring provides a consistent framework for the data access.

VII. Conclusion

Spring allows us to enjoy the key properties of J2EE, while minimizing the complexity encountered during application code. The core part of spring is in providing enterprise services to Plain Old Java Objects (POJOs). This is particularly valuable in a J2EE environment, but application code delivered as POJOs is naturally reusable in a variety of runtime environments.

REFERENCES

[1] Erxiang Chen Personnel department “Research and Design on Library Management System Based on Struts and Hibernate Framework”, IEEE Conference 2009.

[2] Hui Li, Jingjun Zhang, Lei Wang,” The Research and Application of Web-Based System with Aspect-Oriented Features”, IEEE International Conference Dated 2010.

[3] DipankarMajumdar, “Migration from Procedural Programming to Aspect Oriented Paradigm”, IEEE International Conference Dated 2009.

[4] HuiLi ,GuiJunXu , Mingji Zhou, Lingling Si, “Aspect-oriented Programming for MVC Framework” , IEEE Paper Dated 2010.

[5] Robert J. Walker, Elisa L.A. Baniassad and Gail C. Murphy, “An Initial Assessment of Aspect-oriented Programming”, ACM Paper Dated 2009.