ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 7, January 2015

Spring Framework: Spring Web Flow Navigation

Anant Shri Panthri
Web Application Developer

Abstract*— This paper explores spring framework integration with Spring Web Flow. Spring framework along with Spring Web Flow is used in most of the web applications by java developer and is one of the emerging trends, for developing web applications nowadays. This paper discusses spring web flow and Spring MVC separately. This paper also discusses a web application based on spring framework and uses Spring Web flow for navigation purposes.

***Index Terms*—Java Server Faces (JSF), Java Server Pages (JSP), Model View Controller (MVC), Plain Old Java Objects (POJO), spring framework, Spring Web Flow, Spring Web MVC.**

I. INTRODUCTION

Spring Framework is popular amongst most application developers. The main reason for its popularity is modularity. A developer can pick and choose their respective and related modules depending upon the business. The code becomes reusable as well as easy to test. Navigation between screens or java server pages can be done by Spring MVC or by Spring Web Flow. Spring Web Flow is getting popular because it uses reusability of flow thus like code, Flow can also be reused. Flow is a sequence of steps required for execution. For example consider a shopping cart application in which there can be a same flow for multiple contexts therefore for navigation spring web flow finds best use in handling these flows efficiently. We can also see the entire flow of the application by looking at a simple Extensible Markup Language (XML) file. Thus, for maintenance and debugging of the application spring web flow is useful.

II. SPRING FRAMEWORK

One of the key features of spring framework is that it is an open source application and it uses Inversion of Control (IOC) principle also known as dependency injection for passively giving dependencies to objects [1]. That means objects are created at runtime by resolving their dependencies. Spring also supports Aspect Oriented Programming in which application objects are only related with the business logic. They don't bother about the system services. Spring promotes loose coupling. Spring framework is modular that is one can add and use modules when needed.

A. Spring framework Architecture

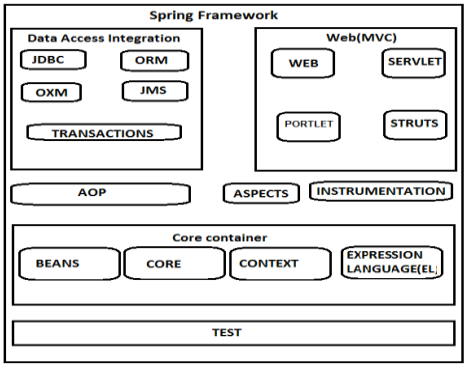

Fig 1. The Architecture of spring [4]

Data Access Integration layer comprises *of JDBC, ORM, OXM, JMS* and *Transaction modules [4]*.

- The JDBC module (Java Database Connectivity) helps to connect to the database and reusability of code is achieved.
- The ORM module (Object relational mapping) includes Java Persistence Application and Hibernate.
- The OXM module (Object Extensible mapping) includes mapping for XMlBeans.
- The JMS module uses features that can produce as well as consume messages.
- The transaction module supports programmatic and declarative transaction management for classes that implements special interfaces or for all Pain Old Java Objects (POJO).

The Web modules comprises of the *Web*, *Web-Struts*, *Web-Servlet* and *Web Portlet* modules.

- Web module- Initialization of the IOC container using the servlet listeners.
- Web-struts modules contain classes that can integrate Struts tier with spring.
- Web-Servlet contains Spring Model, View, and Controller implementation.
- Web Portlet provides MVC (Model View Controller) implementation to be used in Portlet environment.

The Spring Web Flow engine plugs into the Spring Web MVC platform that is Web-Servlet and Web Portlet modules and thus provides a declarative flow definition language.

The Core container comprises of *Core, Bean, Context* and *Expression Language (EL) modules*.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 7, January 2015

- Core module provides features like IOC and dependency Injection
- Bean module provides Bean Factory. It removes the need for programmatic singletons and helps in providing loose coupling.
- Context module is used to access any objects defined and configured.
- Expression Language module is a language that is used to communicate between the presentation layer (screens/java server pages) with the application

Other Miscellaneous modules

- *Aspect Oriented Programming (AOP)* module allows decoupling of code that performs the functionality by allowing the separation of cross cutting concerns (a program affecting other code) thus reduces code duplication and dependencies between systems.
- The Aspects module provides implementation with AspectJ.
- The Instrumentation module provides class instrumentation support and class loader implementation in some application servers.
- Test module supports testing of Spring Components with JUnit.

Using these modules an application can be developed suiting the business requirements.

III. SPRING MVC MODEL

Spring comes with a MVC (model, view, and controller) framework for building web application Spring MVC uses IOC container for the separation from controller logic from the business objects. All incoming HTTP (Hyper Text Transfer Protocol) requests from a web browser are handled by the Controllers. A controller controls the view and model by allowing them to exchange data between them. It segregates the model from the view, models only job is to worry about the data. The view is also not concerned with the business logic and simply renders the data case if the view changes the model is not affected and vice versa.

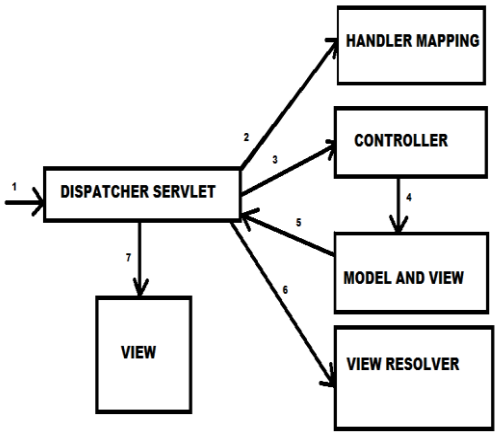

Fig 2.Spring MVC Execution Flow Diagram [1]

1. The request reaches the dispatcher servlet from web.xml.
2. The dispatcher servlet uses handler mapping to map the request with the controller class name.
3. The dispatcher servlet selects the controller.
4. The controller processes the request and executes some Business logic and returns model data and view name.
6. The dispatcher servlet with the help of view resolver selects The appropriate view java server pages, java server faces (jsp, jsf).

B. Spring MVC XML configurations

The dispatcherServlet is as follows:

```
<display-name>LoginSpringMVC</display-name>
<servlet>
<servlet-name>spring</servlet-name>
 <servlet-class>
  org.springframework.web.servlet.DispatcherServlet
 </servlet-class>
<init-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/Spring-servlet.xml</ >
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>spring</servlet-name>
<url-pattern>*.htm</url-pattern>
</servlet-mapping>
<welcome-file-list>
<welcome-file>Index.jsp</welcome-file>
</welcome-file-list>
```

Every URL (Uniform Resource Locater) in the address bar that have an extention*.htm will be catered by the dispatcher servlet. For example if the form action is login.htm then it will be handled by the dispatcher servlet. Spring-servlet.xml is as follows:

```
<context: component-scan base-package="com.controller" />
<bean
class="org.springframework.web.servlet.view.InternalResou
rceViewResolver">
      <property name="prefix" value="/WEB-INF/views/"/>
      <property name="suffix" value=".jsp" />
</bean>
```

The dispatcher servlet upon getting the request will look into the context file that is Spring-servlet.xml which contains the base package for the controllers. The view Resolver will do the following for example if the action of the form is success.htm. Then the view Resolver will prefix the file with WEB-INF/views and suffix it with .jsp that means the dispatcher servlet will search for WEB-INF/views/success.jsp

IV. SPRING WEB FLOW

A flow is a sequence of steps that can be executed in different contexts [2]. Like code flow can also be reused.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 7, January 2015

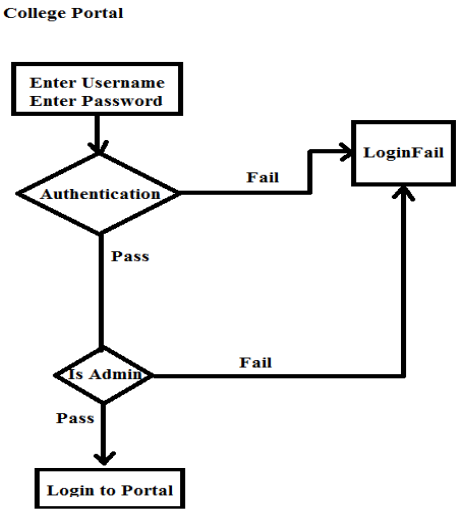

Fig 3. Flowchart of a college portal

For example let us take a College Portal. If the username or the password is incorrect then a login Failure page is displayed. If the registered user checks in as admin he is again authenticated. If the authentication is successful then the user gets logged in the portal as admin otherwise the login Failure page is displayed. The same business can be applied if there are various administrators for various departments and authentication flow remains the same .Thus if the authentication flow is reused we can save a lot of time in rework and debugging. We can achieve this by reusability of flow by plugging in the Spring Web flow engine into the Spring Web MVC platform.

V. SPRING WEB-FLOW CONFIGURATION FILES

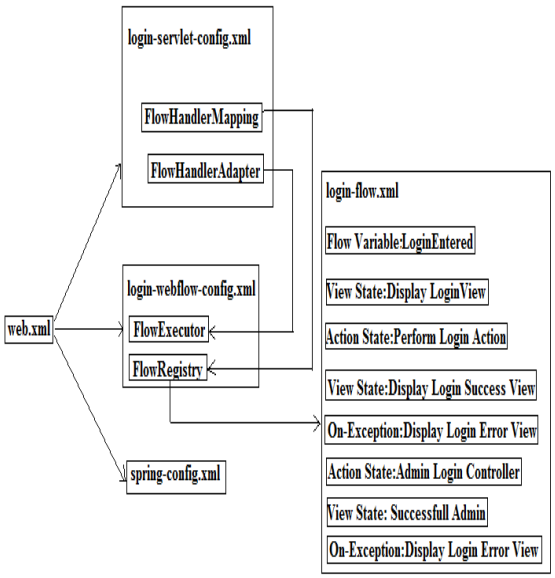

Fig 4. Xml files and inter-relationship between their contents

C. Spring Web Flow Dispatcher Servlet configuration

The dispatcher Servlet is as follows [5]:

```
<display-name>Spring Web Flow</display-name>
 <servlet>
  <servlet-name>loginPortal</servlet-name>
  <servlet-class>
org.springframework.web.servlet.DispatcherServlet
</servlet-class>
 <init-param>
 <param-name>contextConfigLocation</param-name>
<param-value>
   /WEB-INF/config/login-servlet-config.xml
  /WEB-INF/config/login-webflow-config.xml
 /WEB-INF/config/spring-config.xml</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
 </servlet>
  <servlet-mapping>
 <servlet-name>login</servlet-name>
  <url-pattern>/</url-pattern>
  </servlet-mapping>
```

The above three configuration files are as follows:

a) Login-Servlet-config.xml

We need to define the FlowHandlerMapping to tell DispatcherServlet (in web.xml) to send flow requests to Spring Web Flow

```
<bean
class="org.springframework.web.servlet.mvc.annotation.An
notationMethodHandlerAdapter" />
   <bean
class="org.springframework.webflow.mvc.servlet.FlowHand
lerMapping">
   <property name="flowRegistry" ref="loginFlowRegistry"
/>
   </bean>
```

Defined Flow Handler Adapter to handle Spring Web Flow request call. This is the Controller class in Spring Web Flow

```
<bean
class="org.springframework.webflow.mvc.servlet.FlowHa
ndlerAdapter">
    <property name="flowExecutor"
ref="loginFlowExecutor" />
 </bean>
```

b) Login-webflow-config.xml

Define the flow executor responsible for executing login web flow

```
<flow:flow-executor id="loginFlowExecutor"
flow-registry="loginFlowRegistry" />
```

Define the registry that holds references to all the flow related XML configuration

```
<flow:flow-registry id="loginFlowRegistry"
flow-builder-services="flowBuilderServices">
<flow:flow-location id="loginFlow"
path="/WEB-INF/flows/login-flow.xml" />
</flow:flow-registry>
<flow:flow-builder-services id="flowBuilderServices"
view-factory-creator="mvcViewFactoryCreator" />
```

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 7, January 2015

The location and name of the main flow.xml that is Loginflow is mentioned above.

```
<bean id="mvcViewFactoryCreator"
class="org.springframework.webflow.mvc.builder.MvcViewFactoryCreator" >
<property name="viewResolvers" ref="myViewResolver" />
</bean>
<bean id="myViewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="prefix" value="/WEB-INF/views/" />
<property name="suffix" value=".jsp" />
</bean>
```

Its working is same as that of Spring MVC. The file is suffixed with an extension of .jsp while the path where to find that file mapped after the request is mentioned under the property name of "prefix".

```
<bean id="handlerMapping"
class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping">
</bean>
```

a) Spring-config.xml

```
<bean id="loginService"
class="com.java.springwebflow.LoginService"/>
```

D. Login Flow

This user defined xml contains all the flow of the business. While looking at this xml one can understand the whole flow of the business.

```
<var name="firstlogin"
 class="com.java.springwebflow.FirstLogin"/>
```

FirstLogin is a java class containing getters and setters of username and password. A simple POJO class that is mapped with a view. All states of the flow are defined under <flow></flow>. The first state becomes the entry point of the flow. There are various states.

(i) View-state

This state defines the step which will render the view. Over here display the login Page.

```
<view-state id="displayLoginView" view="login"
 model="firstlogin">
 <transition on="loginEntered" to="performLoginAction"/>
 </view-state>
```

(ii) Transition on

The event id mentioned in the view pages is responsible for the transition from one state to another. The transition can be from one view to another view or from one view to as well.

(iii) Action-state

If we need to invoke a business service of the application or other actions. Then actions are used.

Evaluate expression is used to implement a business logic.

```
<action-state id="performLoginAction">
 <evaluate expression="loginService.login(firstlogin)"/>
<transition on =”success” to="displayLoginSuccessView"/>
<transition
on-exception="com.java.springwebflow.IncorrectLoginException"  to="displayLoginErrorView"/>
 </action-state>
```

The class mentioned in the expression LoginService is mentioned in Spring-config.xml so that it is available to loginFlow. Instead of having a "transition on" we can have "transition to" though it is not preferred because business logic will not be entertained and it is always a good programming practice to involve the controller in the logic. If the username and password is not verified from the database or an exception is thrown then also an error page is displayed.

```
<view-state id="displayLoginSuccessView"
view="login_success" model="firstlogin" />
<transition on="loginAdmin" to="adminLoginController"/>
</view-state>

<action-state id="adminLoginController">
 <evaluate
expression="LoginService.loginAdmin(firstlogin)"/>
<transition on="success" to="successfull_Admin"/>
<transition
on-exception="com.java.springwebflow.IncorrectLoginException" to="displayLoginErrorView"/>
</action-state>
```

According to figure3 if the user login is admin then he is allowed to do his tasks or else the same flow of unsuccessful login is initiated. Thus, the flow is reused and code is reused that is the same view (jsp).

```
<view-state id="displayLoginErrorView"
view="login_error"/>
<view-state id="successfull_Admin"
view="login_success_Admin"/>
</flow>
```

Based on the business logic the respective views can be displayed. Also, in case of exception a separate view can be rendered looking only at loginFlow.xml the navigation of the application can be deciphered.

A. Variable scope [3]

Web flow can store variables in the following scopes:

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 7, January 2015

1) Flow scope

The scopes of the variables last only till the flow. When a flow end the variables gets destroyed, any object stored in the flow needs to be Serializable.

2) View scope

The scope of the variables lasts till the view-states. Variables gets allocated when the view-state is allocated and destroyed when the view-states are destroyed, any object stored in the flow needs to be Serializable.

3) Request scope

Request scope gets allocated when the flow is called and destroyed when the flow returns any object stored in the flow needs to be Serializable.

4) Flash Scope

The variables are cleared after every view render and destroyed when the flow is terminated, any object stored in the flow needs to be Serializable.

5) Conversation Scope

The objects are stored in a HTTP Session and conservation scope gets allocated when a top-level flow starts and terminated when the top level flow ends, any object stored in the flow needs to be Serializable.

VI. SPRING HIBERNATE INTEGRATION ADVANTAGES

Spring with spring web flow (for navigation) can be used with hibernate for application development. The spring framework provides a Hibernate Template class, so steps like creating a Session, Build Session Factory, Configuration, and committing transaction etc. are not required and hence save a lot of time.

For example in sole hibernate:

```
//creating configuration
Configuration cfg=new Configuration();
cfg.configure("hibernate.cfg.xml");

//creating seession factory object
SessionFactory factory=cfg.buildSessionFactory();

//creating session object
Session session=factory.openSession();

//creating transaction object
Transaction t=session.beginTransaction();

Admin admin=new Admin (1,"John","Computers");
session.persist(admin);//persisting the object

t.commit();//transaction is commited
session.close();
```

Hibernation when integrated with spring:

```
Admin admin=new Admin(1,"John","Computers");
hibernateTemplate.save(admin);
```

Thus a lot of coding is saved and code can easily be debugged.

VII. CONCLUSION

Spring is a powerful lightweight framework that can be used with spring web flow for easy navigation. Lightweight framework means that spring have minimal impact to an application. The POJO's are unaware of the fact that they are being used by spring. Using Spring web flow for navigation purposes (by latching it into the portlet module of the spring framework) not only saves the time of the developers to search each classes just in order to know the flow of the application but also saves time in debugging the application for flow purposes. Looking at only the main flow xml file the developer can know about the flow, validation methods as well as views about to render after completion of the business logic. The spring framework can be easily integrated with any ORM (Object Relational Mapping) tool such as Hibernate with the help of XML mapping and also with iBATIS.

REFERENCES

[1] Spring in Action, "Craig WallsRyan Breidenbach".

[2] pring web flow references- http://docs.spring.io/autorepo/docs/webflow/2.3.x/reference/html/

[3] Spring Web Flow Reference Guide Version 2.4.0 Keith Donald, Erwin Vervaet, Jeremy Grelle, Scott Andrews, Rossen Stoyanchev, Phillip Webb.

[4] Spring framework architecture accessed from: http://www.tutorialspoint.com/spring/spring_architecture.htm

[5] Spring-web flow references: http://www.studytrails.com/frameworks/spring/spring-web-flow.jsp.

AUTHOR BIOGRAPHY

Anant shri panthri is a software engineer/application developer in Accenture has a one year experience in software development. Worked in multiple projects on technologies like struts, spring, hibernate, JavaScript mvc and html-5.