© November 2015 | IJIRT | Volume 2 Issue 6 | ISSN: 2349-6002

SPRING MVC FRAMEWORK

Simran Bhatti
Computer Science, Maharishi Dayanand University

ABSTRACT:- **: There are a variety of frameworks used when we need to build web applications. Spring is also one of the web applications framework. It can be thought of as a lead framework because it provides support to various other frameworks such as Struts, Hibernate, EJB, JSF etc. The framework can be defined as a structure where we can find solution of the various technical problems. Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.**

Index Terms: Spring MVC, Inversion of control, Pojo classes, container, framework.

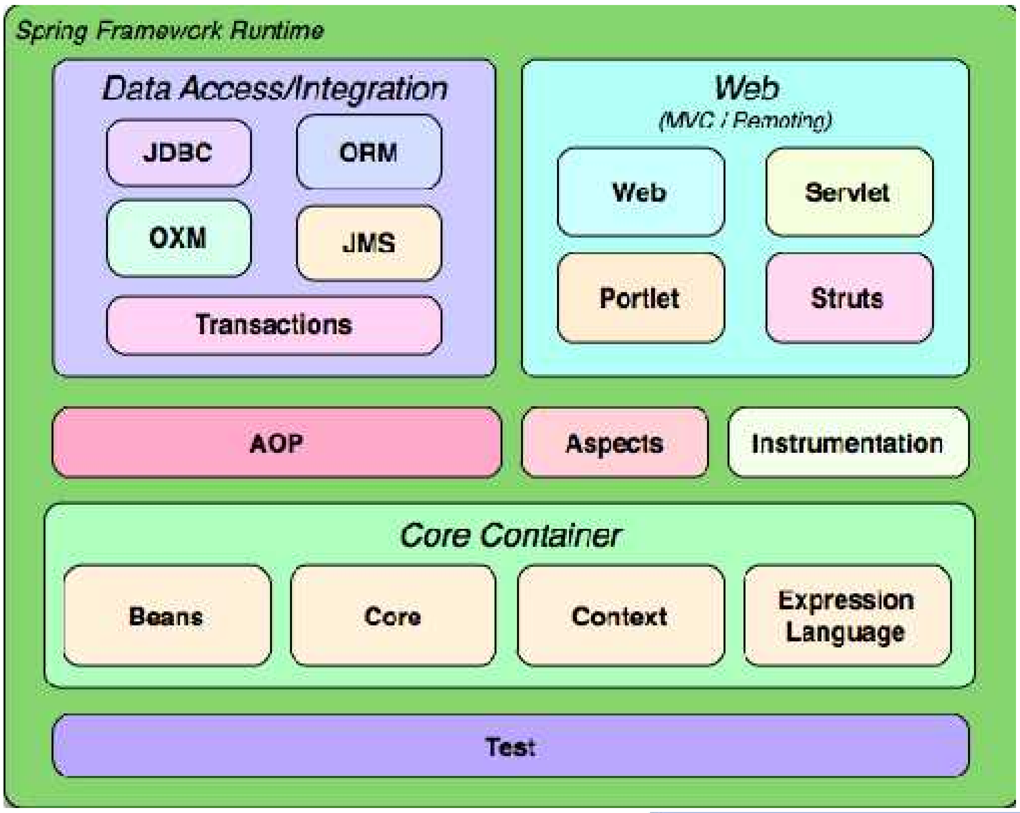
I. INTRODUCTION

Spring Framework is an application framework and inversion of control container for the Java platform. This framework was developed by Rod Johnson in 2003. It makes the easy development of Java EE applications. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. The framework does not impose any specific programming model but it has become popular in the Java community as an alternative to the Enterprise JavaBeans (EJB) model. The Spring Framework is also open source which proves to be an advantage. We can use it for creating a wide variety of web applications. Most of the organizations use this framework.

II. SPRING-Modules

The Spring Framework includes several modules that provide a wide variety of services. They are:

- **Spring Core Container**: This is the base module of Spring. It provides various spring containers.
- **Aspect-oriented programming:** This module enables implementing cross-cutting concerns.
- **Authentication and authorization:** These are the configurable security processes that support a range of standards, protocols, tools and practices.
- **Convention over configuration**: A rapid application development solution for Spring-based enterprise applications.
- **Data access**: Working with relational database management systems on the Java platform using JDBC and object-relational mapping tools and with NoSQL databases.
- **Inversion of control container**: Configuration of application components and lifecycle management of Java objects, done mainly through dependency injection.
- **Messaging:** configurative registration of message listener objects for transparent message-consumption from message queues.
- **Model–view–controller**: an HTTP- and servlet-based framework providing hooks for extension and customization for web applications.
- **Remote access framework**: configurative RPC-style marshalling of Java objects over networks supporting RMI, CORBA and HTTP-based protocols.
- **Transaction management**: unifies several transaction management APIs and coordinates transactions for Java objects.
- **Remote management**: configurative exposure and management of Java objects for local or remote configuration.
- **Testing:** support classes for writing unit tests and integration tests



Spring Framework Modules

III. Spring-IOC and Dependency Injection

These are the design patterns that are used to remove dependency from the programming code. They make the code easier to test and maintain.

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled.

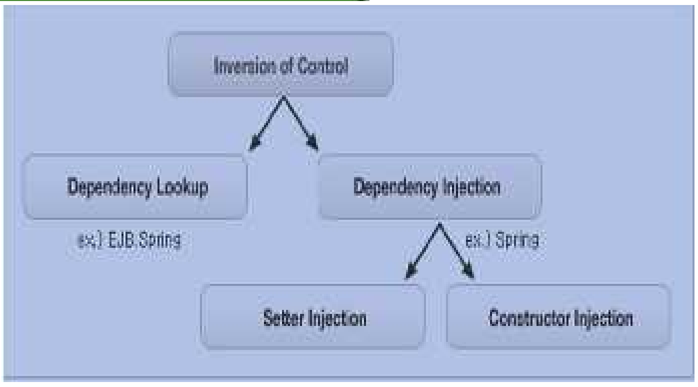
Advantages of Dependency Injection

- makes the code loosely coupled so easy to maintain
- makes the code easy to test

Inversion of Control

This provides a consistent means of configuring and managing Java objects using reflection. The IOC container is responsible for managing object lifecycles of specific objects , creating these objects, calling their initialization methods, and configuring these objects by wiring them together.

IOC makes the code loosely coupled. In such case, there is no need to modify the code if our logic is moved to new environment. In Spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.



Inversion of control

IV. MVC- Design & its Architecture

MVC basically stands for the Model View Controller**.** It is a software architectural pattern for implementing user interfaces. It divides a given software application into three different but interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

It was one of the influential insights in the early development of graphical user interfaces, and one of the first approaches to describe and implement software constructs in terms of their responsibilities**.**

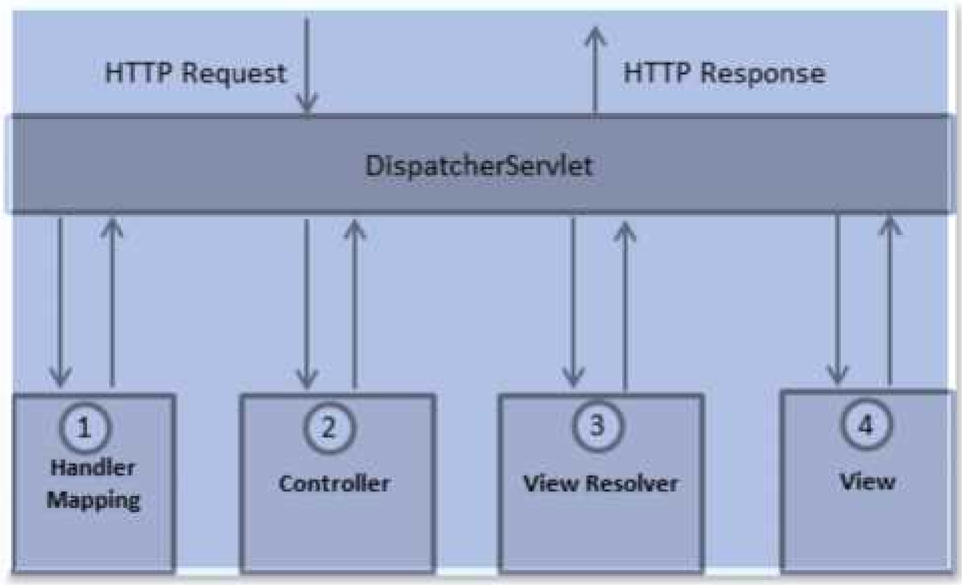
The three components of the MVC are: -

- The **Model,** which encapsulates the application data and in general they will consist of POJO classes.

- The **View** is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
- The **Controller** is responsible for processing user requests and building appropriate model and passes it to the view for rendering.

V. The Dispatcher Servlet

The Spring Web model-view-controller (MVC) framework is designed around a *Dispatcher Servlet* that handles all the HTTP requests and responses. The entire request processing workflow of the Spring Web MVC *Dispatcher Servlet* is depicted in the following diagram:



Following is the sequence of events corresponding to an incoming HTTP request to *Dispatcher Servlet*:

- After receiving an HTTP request, *Dispatcher Servlet* consults the *Handler Mapping* to call the appropriate *Controller*.
- The *Controller* takes the request and calls the appropriate service methods based on the used GET or POST method. The service method will set model data based on defined business logic and returns view name to the *Dispatcher Servlet*.
- The *Dispatcher Servlet* will take help from *View Resolver* to pickup the defined view for the request.
- Once view is finalized, The *Dispatcher Servlet* passes the model data to the view which is finally rendered on the browser.

VI. Advantages of Spring Framework

- **Predefined Templates**

Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.

- **Loose Coupling**

The Spring applications are loosely coupled because of dependency injection.

- **Easy to test**

The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

- **Lightweight**

Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

- **Fast Development**

The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

- **Powerful abstraction**

It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

- **Declarative support**

It provides declarative support for caching, validation, transactions and formatting.

VII. References

[1]https://en.wikipedia.org/wiki/Spring_Framework

[2]http://www.javatpoint.com/spring-3-mvc-tutorial
[3]docs.**spring**.io/**spring**-framework/docs/current/**spring**.../**mvc**.html