UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Spring - Spring MVC Framework

innovative • entrepreneurial • global

Spring Framework

- Spring framework makes the easy development of JavaEE application.
- It is helpful for beginners and experienced persons.
- Spring is a *lightweight* framework.
- It can be thought of as a *framework of frameworks* because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF, etc.

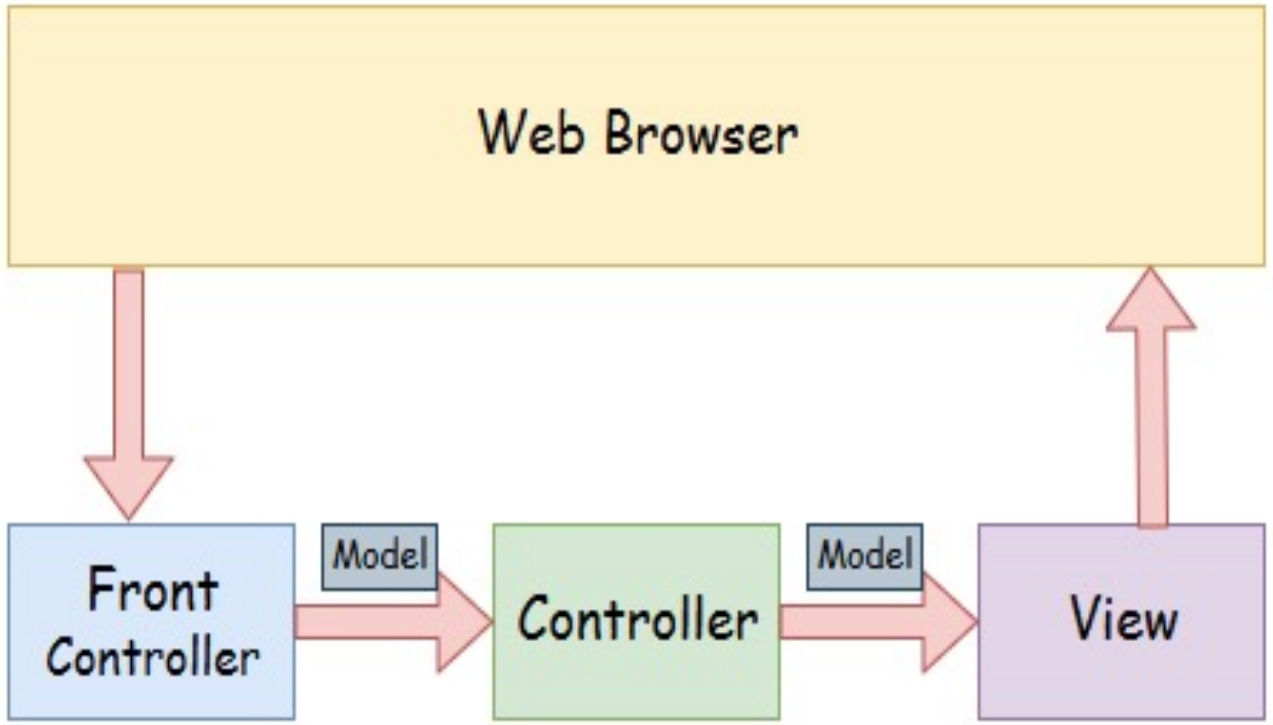
Advantages of Spring Framework

1) **Predefined templates**
   - Provides templates for JDBC, Hibernate, JPA etc. technologies. There is no need to write too much code. It hides the basic steps of these technologies.
2) **Loose coupling**
   - The Spring applications are loosely coupled because of dependency injection.
3) **Easy to test**
   - The Dependency Injection makes easier to test the application.
4) **Lightweight**
   - Lightweight because of its POJO implementation. It doesn't force the programmer to inherit any class or implement any interface.
5) **Fast development**
   - The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

What is Spring MVC?

- Spring MVC is an open source MVC 2 framework for developing web applications.
- It follows the Model-View-Controller design pattern.
- It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.
- A Spring MVC provides an elegant solution to use MVC in Spring framework by the help of **DispatcherServlet**.
- **DispatcherServlet** is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.

Spring Web Model-View-Controller



Spring Web Model-View-Controller

1) Model - A model contains the data of the application. A data can be a single object or a collection of objects.

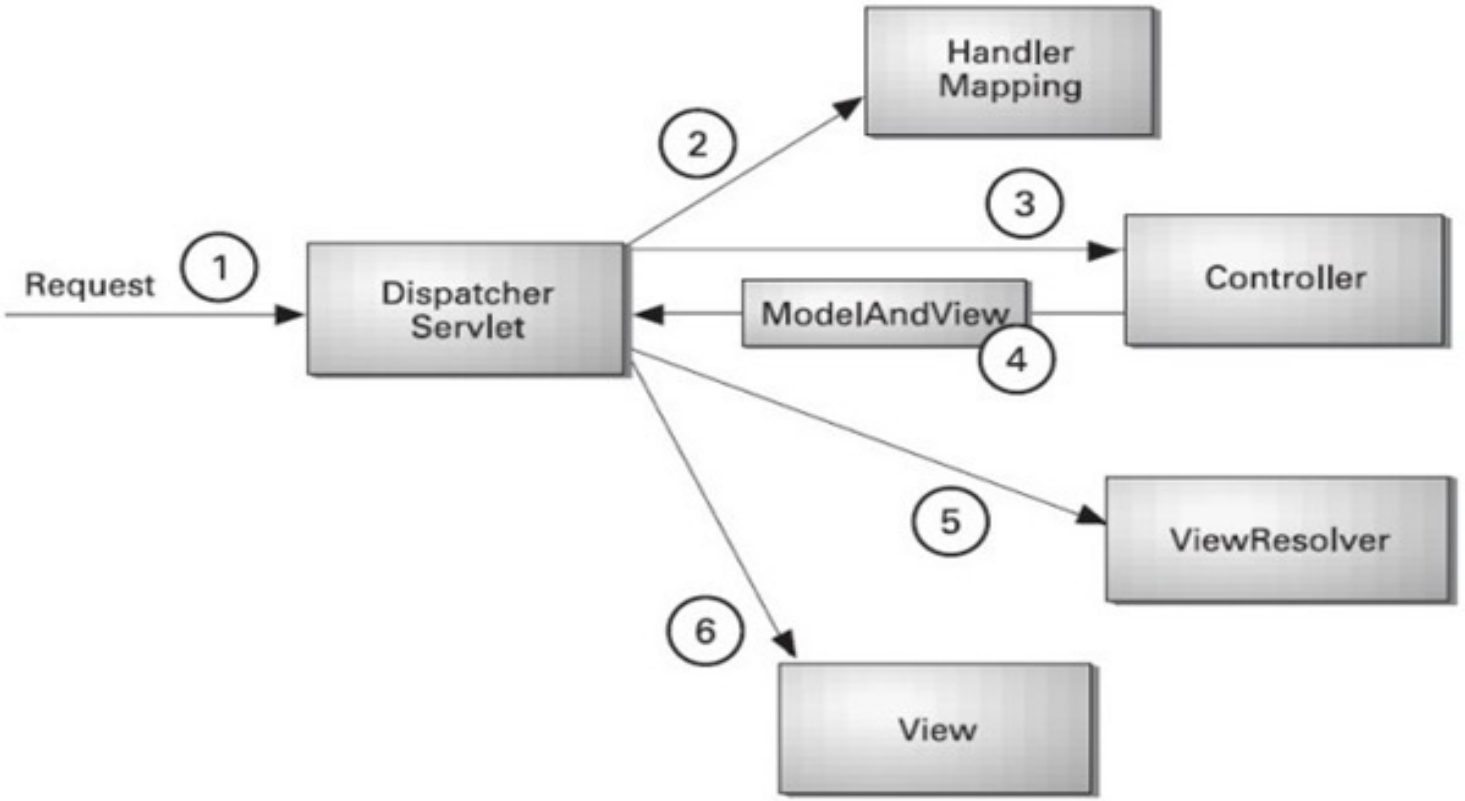
2) Controller - A controller contains the business logic of an application. @Controller annotation is used to mark the class as the controller.

3) View - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.

4) Front Controller - In Spring Web MVC, the DispatcherServlet class works as the front controller.

It is responsible to manage the flow of the Spring MVC application.

The flow of Spring MVC



The flow of Spring MVC

- All the incoming request is intercepted by the DispatcherServlet that works as the front controller.
- The DispatcherServlet gets an entry of handler mapping from the XML file and forwards the request to the controller.
- The controller returns an object of ModelAndView.
- The DispatcherServlet checks the entry of view resolver in the XML file and invokes the specified view component.

DispatcherServlet

- The central components of Spring MVC
- It is where the request first hits the framework
- It is responsible for routing the request through all the other components

```
<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"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
    http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">
```

HandlerMapping

- When a request comes to Spring's dispatcher servlet, it hands over the request to the handler mapping.
- Handler mapping then inspects the request and identifies the appropriate handler execution chain delivers it to dispatcher servlet.
- Use @RequestMapping to map method to URL

```
@Controller
@RequestMapping("/hello.htm")
public class HelloController {

   @RequestMapping(method = RequestMethod.GET)
   public String hello(ModelMap modelMap) {
      System.out.println("on method");
      modelMap.put("printme", "Hello Spring !!");
      return "index";
   }
}
```

Controller

The controller is responsible for processing user request and building appropriate model and passes to front controller (DispatcherServlet)

- Model
- Logical view name

Simply add @Controller annotation to a class

```
@Controller
@RequestMapping("/hello.htm")
public class HelloController {

   @RequestMapping(method = RequestMethod.GET)
   public String hello(ModelMap modelMap) {
      System.out.println("on method");
      modelMap.put("printme", "Hello Spring !!");
      return "index";
   }
}
```

ViewResolver

All controllers in the Spring Web MVC framework return a ModelAndView instance. Views in Spring are:

- addressed by a view name and
- are resolved by a view resolver

```
<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver"
      p:prefix="/WEB-INF/jsp/"
      p:suffix=".jsp" />

      <bean name="indexController"
      class="org.springframework.web.servlet.mvc.ParameterizableViewController"
      p:viewName="index" />

</beans>
```

View name will now be resolve to:
/WEB-INF/jsp/logical-view-name-returned-by-controller.jsp

/WEB-INF/jsp/index.jsp

View

- Spring MVC integrates with many view technologies:
  - JSP
  - Velocity
  - Freemarker
  - JasperReports
- Values sent to controller with POST or GET as usual
- Values made available to the view by the controller

web.xml

- Web applications define servlets in web.xml
- Maps URL patterns to servlets

Example of web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:s
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext.xml</param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>*.htm</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>redirect.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Any URL ending with .htm pattern is routed to the DispatcherServlet, the DispatcherServlet loads dispatcher-servlet.xml file and routes the user to the correct controller.