Advanced Java

Lesson 10—Spring MVC

Learning Objectives

- Explain Spring MVC architecture and components
- List the steps to write Spring MVC program in Eclipse

Advanced Java

Topic 1—Spring MVC Architecture and Components

Why Spring MVC ?

Problems in Enterprise Application development with Servlet and JSP Technology

1. Servlet and code are not reusable
2. Web designing tools can’t be used in case of servlet
3. Parallel development is not possible

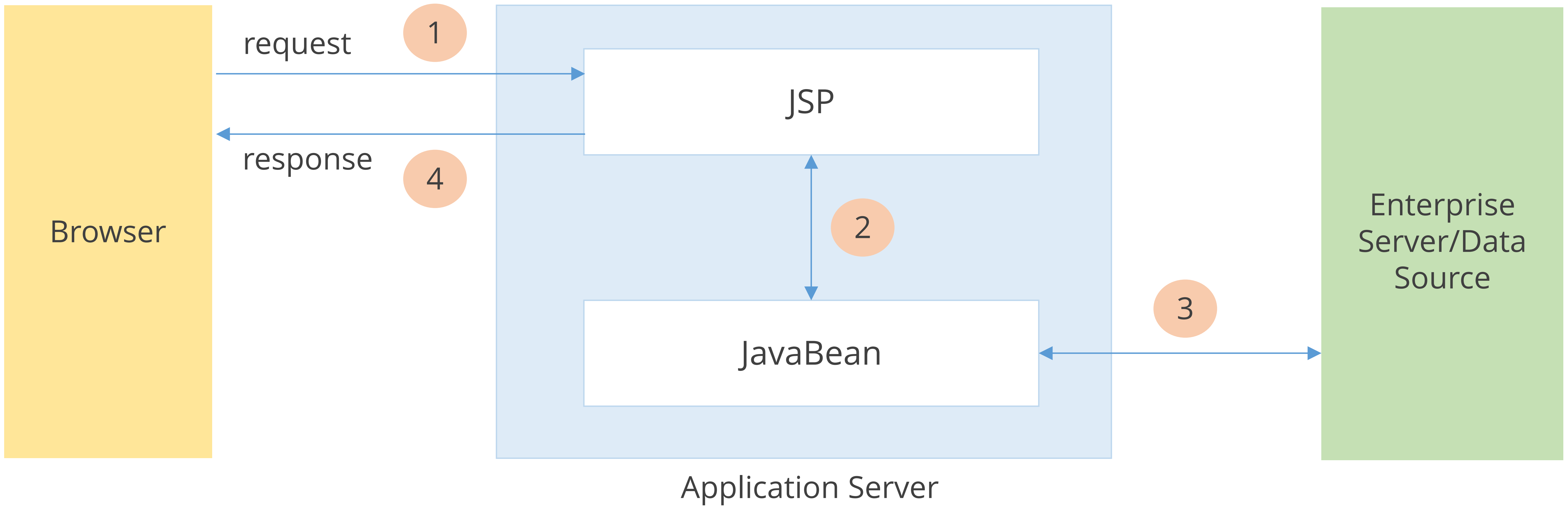
Solution

To solve the above problems, Sun microsystem introduced two design patterns for developing enterprise applications:

1. Model 1 (Page Centric Model)
2. Model 2 (MVC)

Model 1: Page Centric Model

ARCHITECTURE



Model 1: Page Centric Model

WORKFLOW

1. Request is sent by the browser and received by JSP.
2. Java bean object is created and business methods are called. If required, it communicates with the database to get the required data
3. JSP displays the processed data to the end user

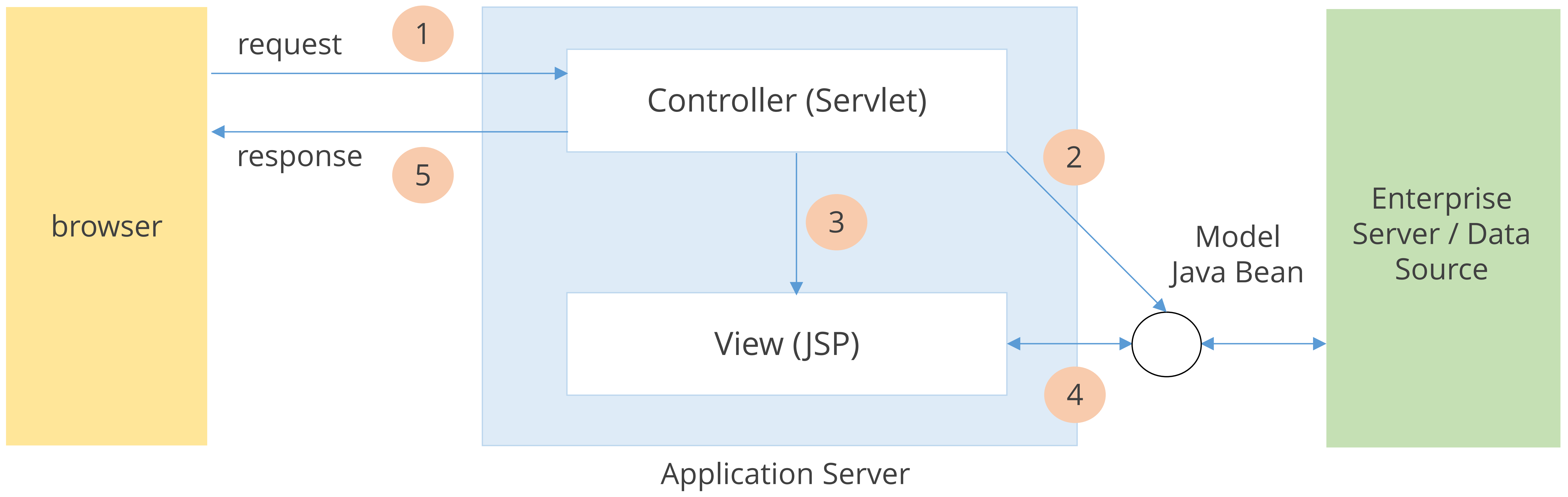
Model 1: Page Centric Model

LIMITATIONS

- There is no clear separation of responsibilities
- JSP acts as both controller and view
- Business logic is separated using Java Bean

Model 2: MVC Model

ARCHITECTURE



Model 2: MVC Model

WORKFLOW

1. Controller or Servlet receives request from browser and captures the input
2. Controller invokes the business method of the model or Java bean
3. Model connects with the database and gets business data
4. Model sends response to controller (Keeps the process data in heap memory request, session, and ServletContext)
5. Controller switches the control to appropriate view of the application

Model 2: MVC Model

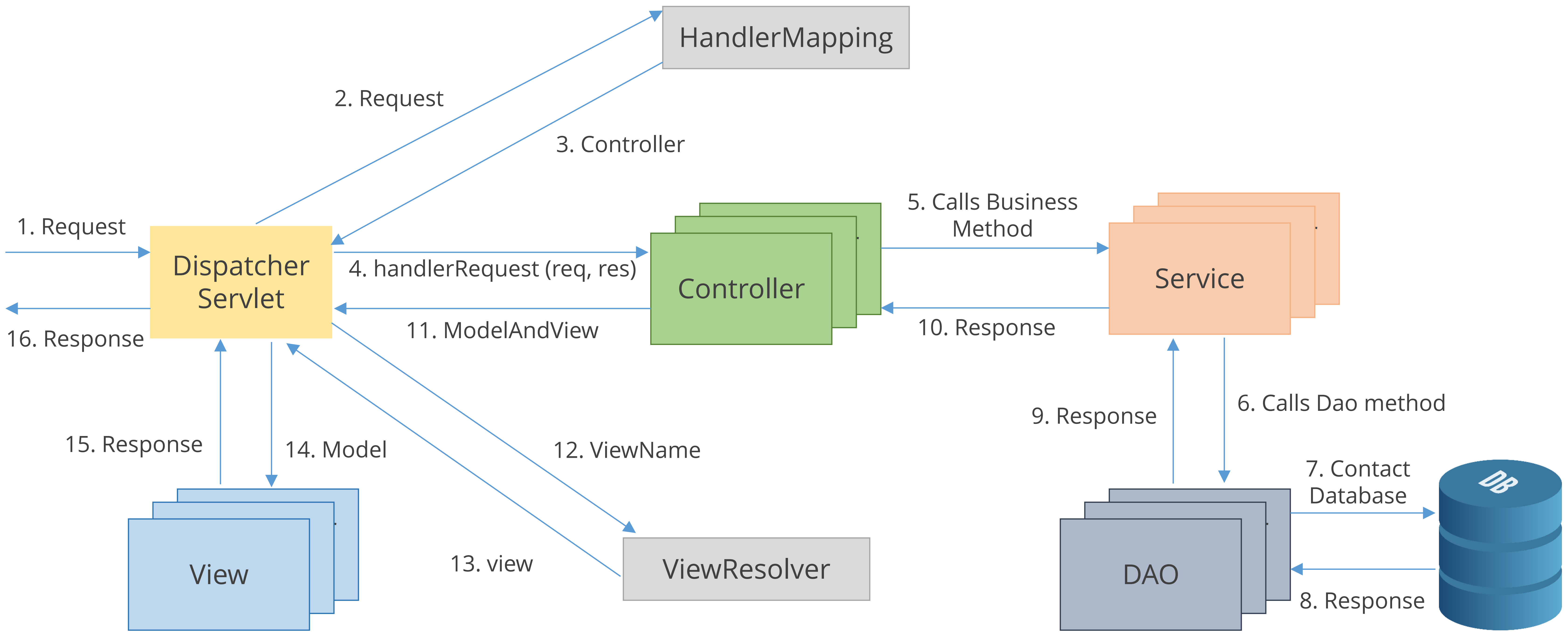
LIMITATIONS

- Clear separation of responsibilities
- Code reusability
- Single point entry for the application
- Support for multiple view technologies

Advantages of Spring MVC

- Spring MVC is used to develop the web application that uses MVC design pattern
- Spring MVC is meant to make web application development faster, cost-effective, and flexible

Spring MVC Architecture



Spring MVC Workflow

1. Client requests a resource in the Web Application
2. The Spring Front Controller (DispatcherServlet) requests the HandlerMapping to identify the particular controller for the given URL
3. HandlerMapping identifies the controller and sends to the DispatcherServlet
4. DispatcherServlet calls handleRequest (req, res) method and passes these two objects to that controller
5. Controller calls the business method
6. Service class calls the DAO method for business data
7. DAO interacts with the database to get the data
8. Database shares the result
9. DAO returns this result data to service
10. DAO data is processed according to business requirement and returns to Controller
11. Controller returns ModelAndView object back to front Controller
12. Dispatcher resolves actual View by consulting the View Resolver object
13. ViewResolver renders View to the Dispatcher
14. DispatcherServlet consults the View to the DispatcherServlet
15. View executes and returns HTML to the DispatcherServlet
16. DispatcherServlet sends output to the Browser

Components of Spring MVC

1. DispatcherServlet (org.springframework.web.servlet)
2. HandlerMapping (org.springframework.web.servlet)
3. Controller
4. ModeAndView (org.springframework.web.servlet)
5. ViewResolver

Components of Spring MVC

DispatcherServlet

DispatcherServlet
HandlerMapping
Controller
ModelAndView
ViewResolver

1. It is given by org.springframework.web.DispatcherServlet
2. It follows FrontController Design Pattern
3. Whatever URL comes from the Client, Servlet intercepts the Client Request before passing the Request object to the Controller
4. In web configuration file, write <servlet-mapping> in such a way that Dispatcher Servlet is invoked for ClientRequest

```
<servet>
<servlet-name> front-controller</servlet-name>
<servlet-class>
org.springframework.web.servlet.DispatcherServlet
</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name> front-controller </servlet-name>
<url-pattern>*.extensionname</url-pattern>
</servlet-mapping>
```

Definition given in the web.xml to invoke Spring's Dispatcher

Components of Spring MVC

HandlerMapping

- DispatcherServlet
- HandlerMapping
- Controller
- ModelAndView
- ViewResolver

- It is an Interface implemented by objects to define mapping between requests and handler objects.
- When a request is made to Spring’s dispatcher servlet, it hands over the request to handler mapping.
- Handler mapping inspects the request and identifies the appropriate handler execution chain and delivers it to dispatcher servlet.

Components of Spring MVC

HandlerMapping: Example

- DispatcherServlet
- HandlerMapping
- Controller
- ModelAndView
- ViewResolver

Hander mapping provided by Spring’s MVC module can be implemented in many ways. Lets take a example:

BeanNameUrlHandlerMapping: It is the default handler mapping class~~,~~ that maps the URL request to the names of the beans.

```
<bean…>
<bean class=“org.springframework.web.servlet.handler.
BeanNameUrlHandlerMapping”/>
<bean name=“/welcome.htm” class=“WelcomeController”/>
<bean name=“/streetName.htm” class=“StreetNameController”/>
<bean name=“/process*.htm” class=“ProcessController”/>
```

If URL pattern:

- /welcome.htm is requested; DispatcherServlet will forward the request to the “WelcomeController.”
- /streetName.htm is requested; DispatcherServlet will forward the request to the “StreetNameController.”
- /processCreditCard.htm or /process{anything}.htm is requested; DispatcherServlet will forward the request to the “ProcessController.”

Components of Spring MVC

Controller

DispatcherServlet

HandlerMapping

Controller

ModelAndView

ViewResolver

- Controllers are components that are called by the Dispatcher Servlet for any kind of Business logic.
- All controllers implements Controller interface.

Types of Controllers:

1. AbstractController
2. MultiActionController
3. AbstractwizardFormController

There are components called ViewResolver. Their job is to provide mapping between the Logical View Name and the actual Physical Location of the View Resource.

Components of Spring MVC

ModelAndView

- DispatcherServlet
- HandlerMapping
- Controller
- ModelAndView
- ViewResolver

- It is represented by the class org.springframework.web.servlet.ModelAndView and is returned by the Controller object back to the DispatcherServlet
- This class is just a CONTAINER CLASS FOR HOLDING THE mODEL AND THE vIEW information
- The Model object represents some piece of information that can be used by the View to display the information. These give abstraction in the Spring framework
- Any kind of View Technology (org.springframework.web.servlet.View) can be plugged into the Framework
  Excel, Jasper Reports, Pdf, Xslt, freeMarker, Html, Tiles are supported frameworks.

Example:

```
ModelAndView mv=new ModelAndView("successView", "greetingMsg", "greetingMessage);
```

Components of Spring MVC

ModelAndView

- DispatcherServlet
- HandlerMapping
- Controller
- ModelAndView
- ViewResolver

It is an Interface implemented by objects to resolve views using name.Spring's MVC module.

It encapsulates the model object and the view object in a single entity, which is represented by the object of class ModelAndView.

To resolve the view object, DispatcherServlet ViewResolver is used.

All view resolvers implement the interface org.springframework.web.servlet.ViewResolver.

Components of Spring MVC

ModelAndView: Types

- DispatcherServlet
- HandlerMapping
- Controller
- ModelAndView
- ViewResolver

InternalResourceViewResolver

It resolves the logical name of the view to an internal resource by prefixing the logical view name with the resource path and suffixing it with the extension.

BeanNameViewResolver

It resolves the logical name of the view to the bean name, which will render the output to the user. The bean should be defined in the Spring app context file.

XMLFileViewResolver

This view resolver is the same as BeanNameViewResolver. The only difference is that instead of looking for the beans in Spring's application context file, it looks for beans defined in a separate XML file (/WEB-INF/views.xml by default).

Advanced Java

Topic 2—Spring MVC Program in Eclipse

Writing Spring MVC Program in Eclipse

1. Create dynamic web project in eclipse
2. Create 'index.jsp' (View)
3. Create HelloWorldController.java (Controller)
4. Create mvc-dispatcher-servlet.xml file. (Spring Configuration)
5. Create web.xml (Deployment Descriptor)
6. Run the project

Writing Spring MVC Program in Eclipse

Create dynamic web project in eclipse

Create 'index.jsp' (View)

Create HelloWorldController .java (Controller)

Create mvc-dispatcher-servlet.xml file. (Spring Configuration)

Create web.xml (Deployment Descriptor)

Run the project

Create 'index.jsp' (View)

In this case, “view” is a JSP page. We can display the value “hello Spring MVC” that is stored in the model “msg” via expression language (EL) ${msg}

```
HelloWorldPage.jsp

<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<body>
      <h1>Spring MVC Hello World Example</h1>

      <h2>${msg}</h2>
</body>
</html>
```

We have already discussed the steps to create a web project in eclipse in the previous lessons.

Writing Spring MVC Program in Eclipse

Create HelloWorldController.java (Controller)

- Create dynamic web project in eclipse
- Create 'index.jsp' (View)
- Create HelloWorldController .java (Controller)
- Create mvc-dispatcher-servlet.xml file. (Spring Configuration)
- Create web.xml (Deployment Descriptor)
- Run the project

Spring has Controllers. Here, AbstractController overrides the handleRequestInternal() method

```
public class HelloWorldController extends AbstractController{
      @Override
      protected ModelAndView handleRequestInternal(HttpServletRequest
request,HttpServletResponse response) throws Exception {
            ModelAndView model = new ModelAndView("HelloWorldPage");
            model.addObject("msg", "Welcome");
            return model;
      }
}
```

ModelAndView(“HelloWorldPage”) identifies which view should return to the user. In this example, HelloWorldPage.jsp will be returned.

model.addObject (“msg”, “Welcome”) adds a “welcome” string into a model named “msg.” You can later use EL ${msg} to display the “hello world” string.

Writing Spring MVC Program in Eclipse

Create mvc-dispatcher-servlet.xml file. (Spring Configuration)

- Create dynamic web project in eclipse
- Create 'index.jsp' (View)
- Create HelloWorldController .java (Controller)
- Create mvc-dispatcher-servlet.xml file. (Spring Configuration)
- Create web.xml (Deployment Descriptor)
- Run the project

Declare the Spring Controller and viewResolver (mvc-dispatcher-servlet.xml).

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
    <bean name="/welcome.htm"
          class="HelloWorldController" />
    <bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver" >
        <property name="prefix">
            <value>/WEB-INF/pages/</value>
        </property>
        <property name="suffix">
            <value>.jsp</value>
        </property>
    </bean>
</beans>
```

Writing Spring MVC Program in Eclipse

Create mvc-dispatcher-servlet.xml file. (Spring Configuration)

- Create dynamic web project in eclipse
- Create 'index.jsp' (View)
- Create HelloWorldController .java (Controller)
- Create mvc-dispatcher-servlet.xml file. (Spring Configuration)
- Create web.xml (Deployment Descriptor)
- Run the project

1. Controller declares a bean name /welcome.htm and maps it to HelloWorldController. If a URL with /welcome.htm pattern is requested, the HelloWorldController controller will handle the request.

2. viewResolver defines how Spring will look for the view template. In this case, the controller HelloWorldController will return a view named HelloWorldPage, and the viewResolver will find the file with the following mechanism: prefix + view name + suffix, which is /WEB-INF/pages/HelloWorldPage.jsp.

Writing Spring MVC Program in Eclipse

Create web.xml (Deployment Descriptor)

- Create dynamic web project in eclipse
- Create 'index.jsp' (View)
- Create HelloWorldController .java (Controller)
- Create mvc-dispatcher-servlet.xml file. (Spring Configuration)
- Create web.xml (Deployment Descriptor)
- Run the project

web.xml declares a DispatcherServlet servlet to act as the front-controller to handle all the web requests that end with htm extension.

```
web.xml
<web-app >
  <display-name>Spring Web MVC Application</display-name>
  <servlet>
      <servlet-name>mvc-dispatcher</servlet-name>
        <servlet-class>
             org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
      <servlet-name>mvc-dispatcher</servlet-name>
        <url-pattern>*.htm</url-pattern>
  </servlet-mapping>
</web-app>
```

Writing Spring MVC Program in Eclipse

Create dynamic web project in eclipse

Create 'index.jsp' (View)

Create HelloWorldController.java (Controller)

Create mvc-dispatcher-servlet.xml file. (Spring Configuration)

Create web.xml (Deployment Descriptor)

Run the project

Run the project

To run application, the following URL is used:

http://localhost:8080/projectname/welcome.htm

Key Takeaways

- Spring MVC is used to develop the web application that uses MVC design pattern. Spring MVC is used to make web application development faster, cost-effective, and flexible
- DispatcherServlet (org.springframework.web.servlet), HandlerMapping (org.springframework.web.servlet), Controller, ModeAndView (org.springframework.web.servlet), and ViewResolver are the components of spring MVC.

Quiz

Thank You