infogain

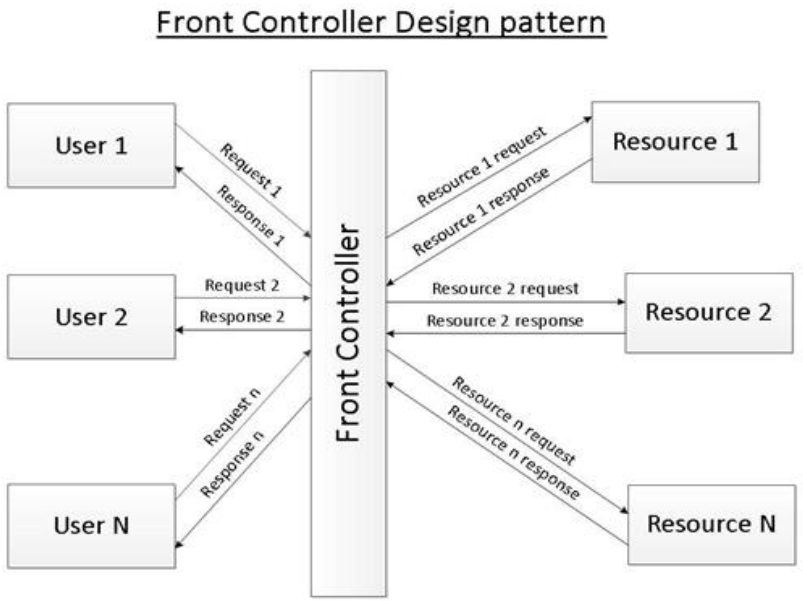
Spring MVC(Part-1)

Objectives

- **In this session, you will learn to:**
  - Explore Spring MVC
  - Handle Web requests
  - Explore Spring Frontend Controller
  - Explore helper Controller.
  - Explore view resolving
  - Explore ModelAndView Class

Front Controller Design Pattern

- **Front Controller Design Pattern**
  - Enforces a single point of entry for all the incoming requests. All the requests are handled by a single piece of code which can then further delegate the responsibility of processing the request to further application objects.

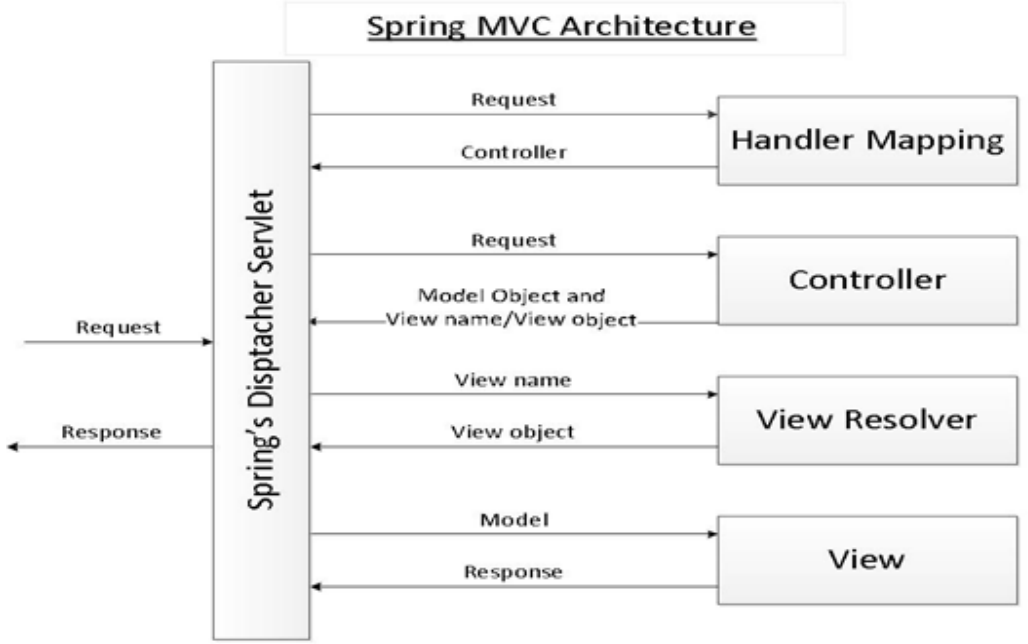


What is MVC?

- MVC
  - It is a software design pattern for developing web applications which isolates the application logic from the user interface layer and supports separation of concerns.
- Components of MVC
  - **Model** - The lowest level of the pattern which represents the application data.
  - **View** - Represents the application's user interface. It takes model as the input and renders it to the end user.
  - **Controller** - Software Code that controls the interactions between the Model and View. Responsible for handling the request and generating the model and selecting the appropriate view for the request.

Spring MVC

- Spring's MVC module is based on front controller design pattern followed by MVC design pattern. All the incoming requests are handled by the single servlet named **DispatcherServlet** which acts as the front controller in Spring's MVC module. The **DispatcherServlet** then refers to the HandlerMapping to find a controller object which can handle the request.
- The controller object perform the business logic to fulfil the user request.

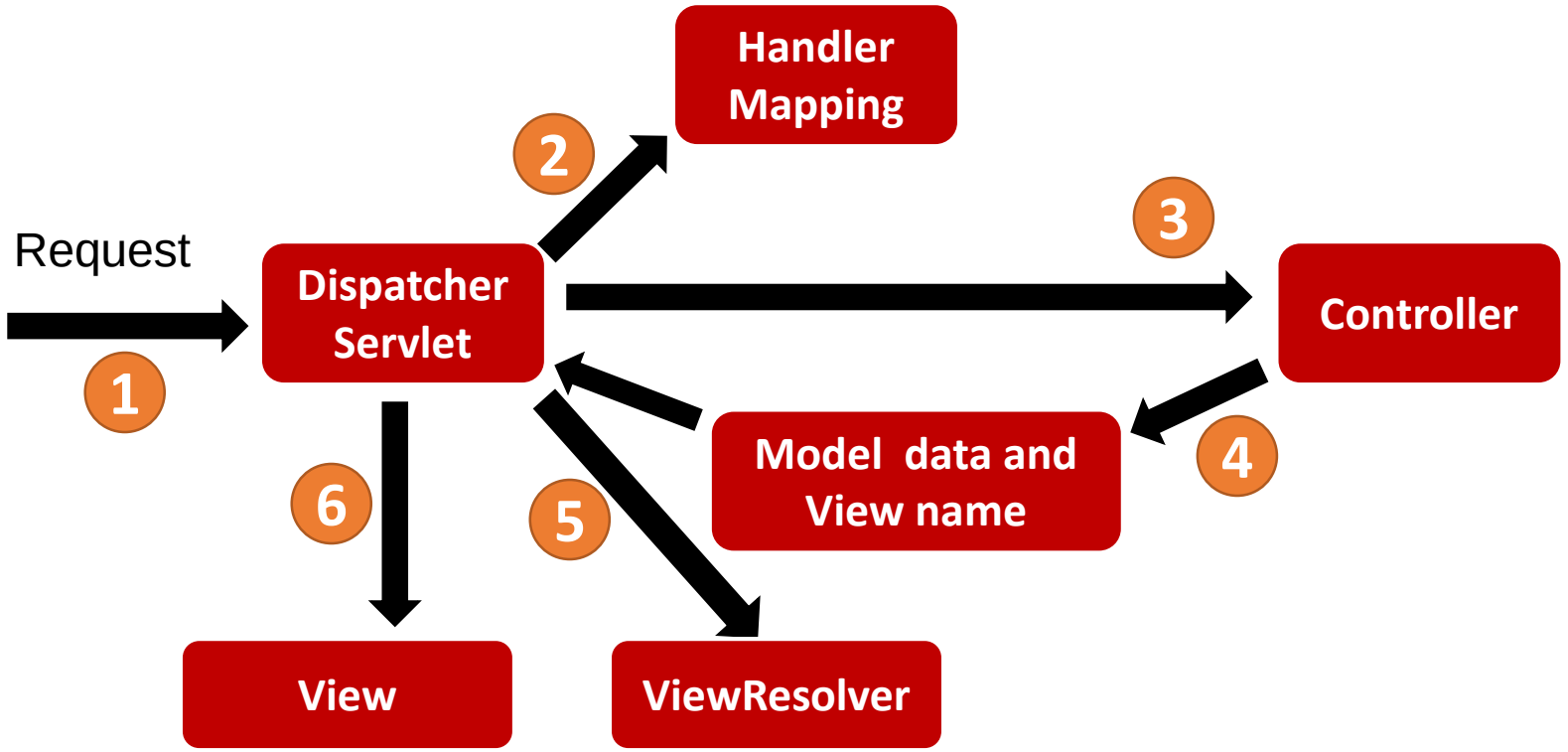


Life Cycle of a Web Request

- When ever you are sending a web request it carries information about the URL of the requested page.
- The **web.xml** file of a Web application contains information about how to handle a particular Web request.
- **Therefore, when a server receives a request, it:**
  - Reads the web.xml of the Web application.
  - Forwards the request to the dispatcher servlet assigned to handle that request.

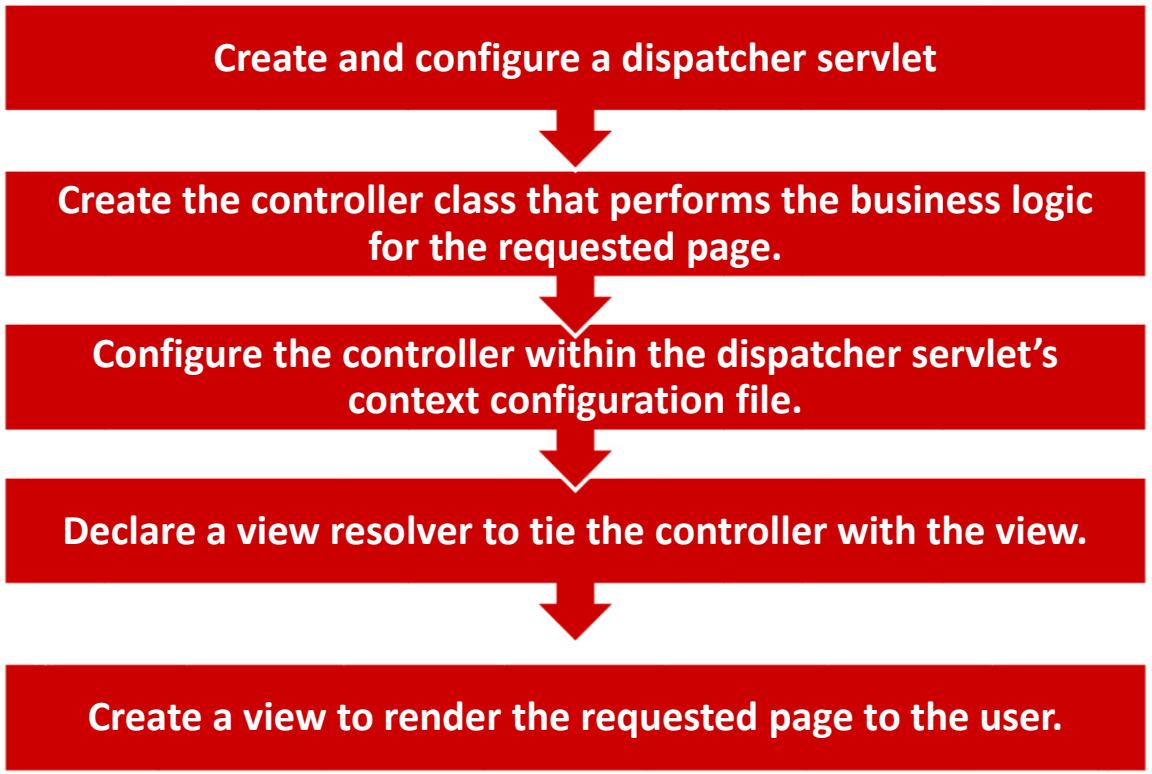
Life Cycle of a Web Request (contd..)

- The following figure shows how a request is processed by a dispatcher servlet.



Handling Web Requests

- To enable the server to handle the Web requests in a Spring-enabled MVC Web application, the following steps need to be performed:



Configuring Dispatcher Servlet

- When a user requests for a particular page, the request is received by a front controller at the server.
- The front controller is a servlet called dispatcher servlet and is represented by the **org.springframework.web.servlet.DispatcherServlet** class.
- The DispatcherServlet class is integrated with the Spring framework's ApplicationContext interface.
- This integration allows you to use major features, such as DI and AOP, of the Spring framework.

Configuring Dispatcher Servlet (contd..)

- The dispatcher servlet intercepts all user requests before passing them to a controller class.
- For a dispatcher servlet to intercept all user requests, declare and configure it in the web.xml configuration file with the help of the following elements:
  - <servlet>
  - <servlet-mapping>

Configuring Dispatcher Servlet (contd..)

- **Example:**

```
<servlet>
<servlet-name>dispatcher</servlet-name>
 <servlet-class> org.springframework.web.servlet.DispatcherServlet
</servlet-class>
<load-on-startup>1</load-on-startup>
 </servlet>
```

- Defines a name for the dispatcher servlet as dispatcher.
- Defines the order in which the various servlets defined in the web.xml file are initialized when the Web application is started.
- Defines the Spring API class, org.springframework.web.servlet.DispatcherServlet that represents the dispatcher servlet.

Configuring Dispatcher Servlet (contd..)

- **Consider the following code snippet:**

```
<servlet-mapping>
<servlet-name>dispatcher</servlet-name>
<url-pattern>*.htm</url-pattern>
</servlet-mapping>
```

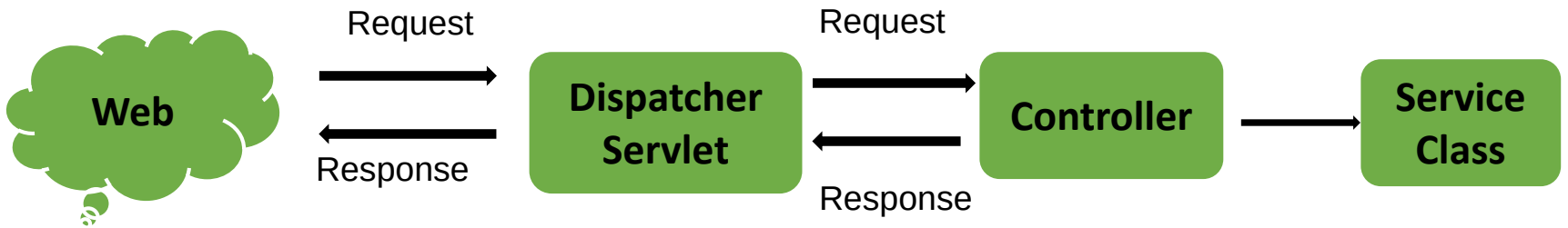
- Specifies the URLs that will be handled by the dispatcher servlet.
- Here all the requested URLs having an extension .htm will be handled by the dispatcher servlet.

Objectives

- In this session, you will learn to:
  - Handle Web requests

Creating a Controller(contd..)

- The dispatcher servlet delegates the request to another Spring MVC component known as controller.
- The controller is responsible for processing all the requests coming from the Web browser and generating the desired response.
- This response is the page requested by the user.
- The following figure shows the flow of a Web request.



Creating a Controller (contd..)

- The controller forwards the results back to the browser in the form of a ModelAndView object that is returned by the **org.springframework.web.servlet.ModelAndView** class.
- The ModelAndView class encapsulates the model and view data to be displayed to the user by the view.
- The response is a ModelAndView object that holds the view and the data to be rendered on that view.

Rendering Response to the Client (contd..)

- To render the response on the user's browser screen, a view, such as JSP, is used.
- The name of the JSP page is determined by the controller from the logical view name returned by the ModelAndView object.
- To enable the Spring MVC framework in locating the appropriate JSP page, you need to declare a view resolver bean in the dispatcher servlet's configuration file.

Rendering Response to the Client (contd..)

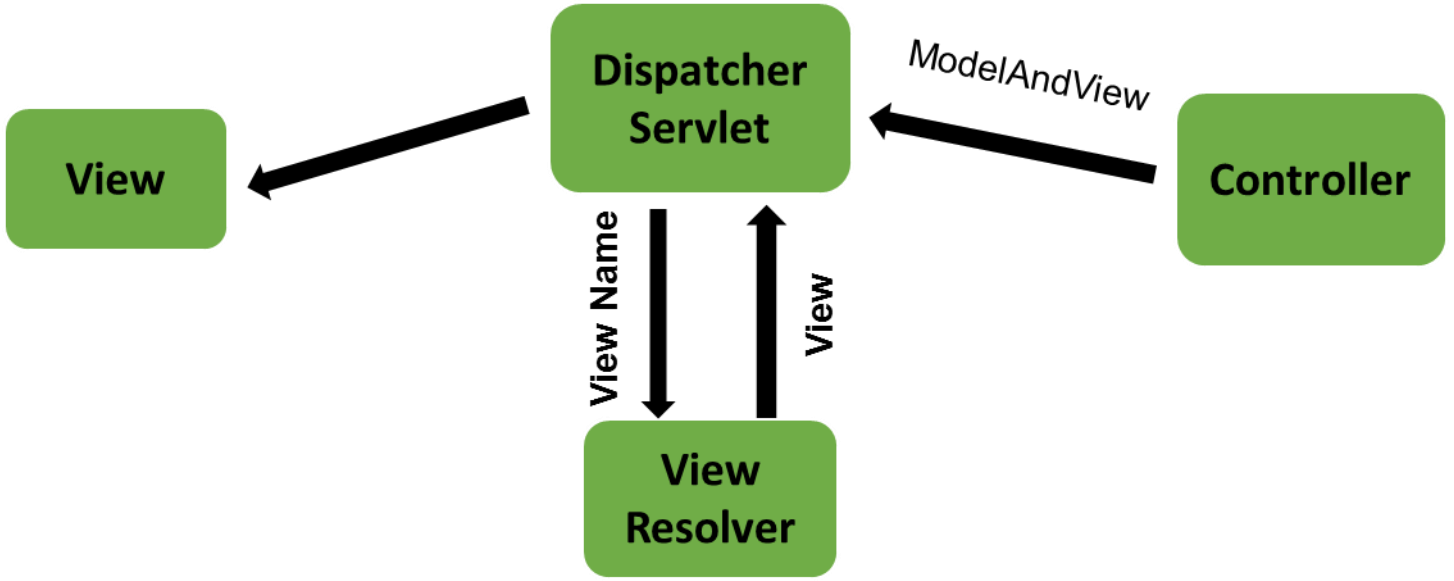
- The responsibility of a view resolver bean is to:
  - Take the logical view name returned by the ModelAndView object.
  - Map the logical view name to the view defined by the JSP page.

- Spring uses an interface, called View, that is responsible for handing over the user request to a specified view technology, such as JSP.

- To render the response to the client, the following steps need to be performed:
  - Declare a view resolver.
  - Create a JSP page.

Rendering Response to the Client (contd..)

- **Declaring a view resolver:**
  - Spring uses view resolvers to resolve the logical view names to the actual views defined by a JSP page.
  - The following figure shows the resolution of the logical view name to actual view name.



Rendering Response to the Client (contd..)

- To resolve logical view names, Spring makes use of the **org.springframework.web.servlet.ViewResolver** interface.
- You can either use the resolver implementations provided by Spring or write your own custom view resolvers for resolving view names.
- **Spring provides you with the following ViewResolver interfaces:**
  - InternalResourceViewResolver
  - ResourceBundleViewResolver
  - XmlViewResolver

Rendering Response to the Client (contd..)

- To resolve the views rendered using JSP, the InternalResourceViewResolver class is used.
- **Example:**

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

- Prefixes the view name with the value of the prefix property, and then suffixes it with the value of the suffix property to return the actual view name located at /WEB-INF/jsp/BookTickets.jsp.

Spring Stereotype Annotations

- **Stereotype Annotations**
  - @Component
  - @Repository
  - @Service
  - @Controller

Classes that are annotated with one of these annotations will automatically detected by the Spring container as part of the container's component scanning process and registered in the Spring application context.

for component scan you have to use **<context:component-scan>** inside the Spring configuration file.

Spring Stereotype Annotations(contd..)

infogain
www.infogain.com

Annotation	Meaning
@Component	Generic stereotype for any Spring-managed component
@Repository	Stereotype for persistence layer
@Service	Stereotype for service layer
@Controller	Stereotype for presentation layer (Spring-MVC)

Handling Form Data

- **Path Attribute**
  - It must correspond to a getter or setter of the model attribute.
  - When the page is loaded, the input fields are populated by Spring, which calls the getter of each field bound to an input field.
  - When the form is submitted, the setters are called to save the values of the form to the object

Handling Form Data

infogain
www.infogain.com

- **Command Object**
  - When an object of a model class is bound to a form, it is called form-backing object.
  - Think of Command Object as a POJO/JavaBean/etc.. that backs the form in your presentation layer.
  - At presentation layer the commandName attribute is the key which specifies name of the model class object that acts as a backing object for this form
  - Once the form is submitted, all the individual attributes are bound to this object.

Handling Form Data

- **@ModelAttribute**
  - Populates the fields of a class, which then populates an attribute of the model to be passed back to the view
  - The model, fields should be populated from all request parameters that have matching names.
  - This is known as data binding in Spring MVC, a very useful mechanism that saves you from having to parse each form field individually.

25

infogain

Thank You

infogain

www.infogain.com

Infogain Corporation, HQ
485 Alberto Way Los Gatos,
CA 95032 USA
Phone: 408-355-6000
Fax: 408-355-7000

Pune
7th Floor, Bhalerao Towers, CTS No.1669 - 1670, Behind Hotel Pride,
Shivaji Nagar, Pune - 411005
Phone : +91-20-66236700

Infogain Irvine
41 Corporate Park,
Suite 390 Irvine, CA 2606 USA
Phone: 949-223-5100
Fax: 949-223-5110

Infogain Austin
Stratum Executive Center Building D
11044 Research Boulevard Suite 200
Austin, Texas 78759

Noida
A-16, Sector 60, Noida Gautam Budh agar,
201301 (U.P.) India
Phone: +91-120-2445144
Fax: +91-120-2580406

Dubai
P O Box 500588 Office No.105,
Building No. 4, Dubai Outsource Zone,
Dubai, United Arab Emirates
Tel: +971-4-458-7336