Spring MVC - II

MVC In Spring & Request Life Cycle

Harish B Rao

Talent Transformation | Wipro Technologies

MVC in Spring

Spring's MVC web module is based on front controller and MVC design patterns.

All the incoming requests are handled by a single servlet, *DispatcherServlet*, which acts as the front controller. The *DispatcherServlet* then refers to the HandlerMapping to find a Controller object, which can handle the request.

Individual Controllers can be used to handle requests from different clients.

Controllers are Classes of Java. Controllers are managed exactly like any other bean in the Spring ApplicationContext.

DispatcherServlet

The Spring Web MVC framework is designed around the ***DispatcherServlet*** that handles all the HTTP requests and responses.

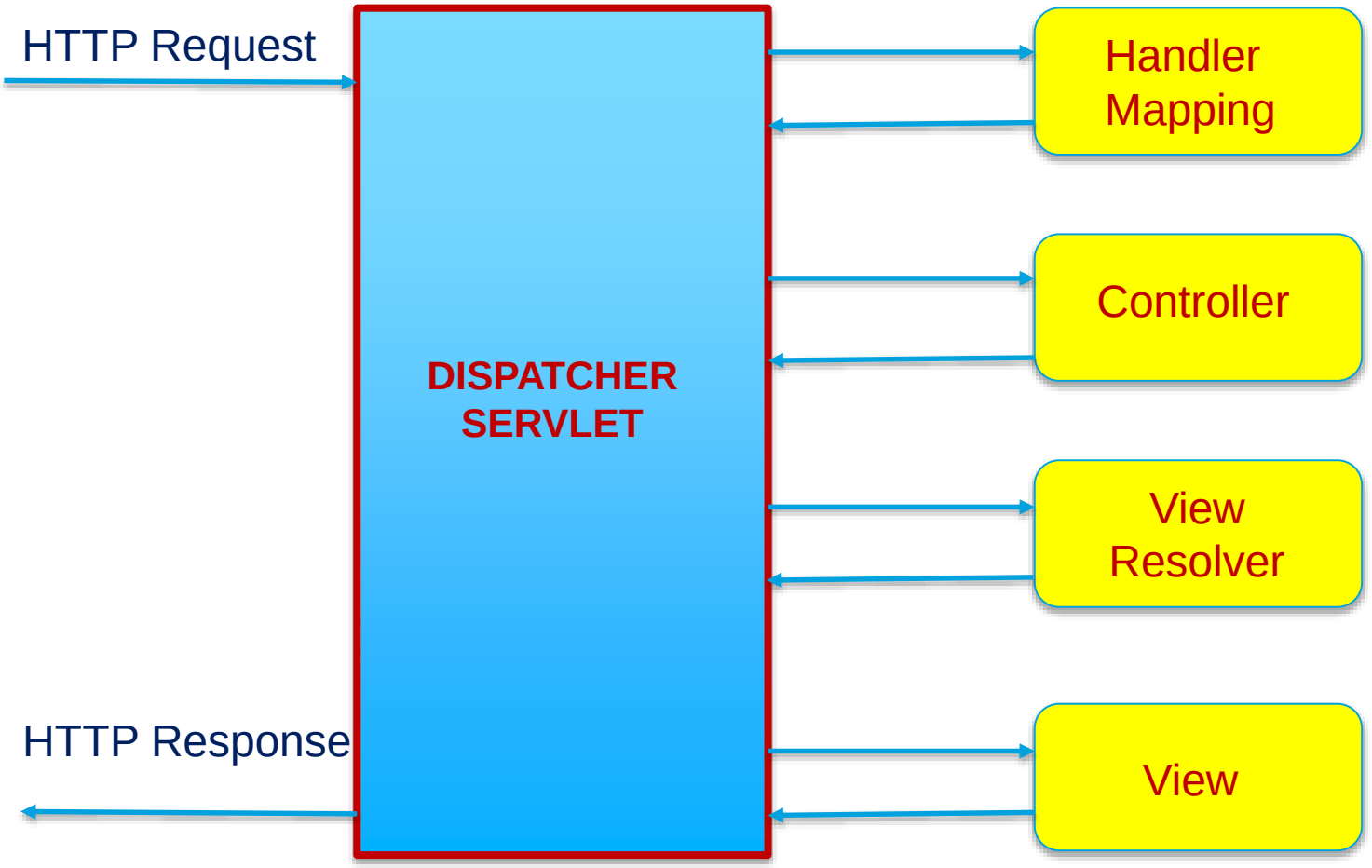
DispatcherServlet is found in the package *org.springframework.web.servlet.*

Since all the requests are routed through DispatcherServlet, all Spring MVC web applications will have the following entries in the web.xml file.

```
<servlet>
      <servlet-name>…</servlet-name>
       <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>…</servlet-name>
      <url-pattern>/</url-pattern>
</servlet-mapping>
```

MVC in Spring

The DispatcherServlet



Core Components of Spring MVC

- **DispatcherServlet**
  - Spring's Front Controller implementation

- **Controller**
  - User created component for handling requests
  - Encapsulates navigation logic
  - Delegates to the service objects for business logic
  - The default handler is a very simple **Controller** interface with method:

    ModelAndView **handleRequest**(request, response).

- **View**
  - Responsible for rendering output

**** No need to break your head over terms like navigation logic, service object, business logic, default handler etc. As you proceed further, you will automatically understand.*

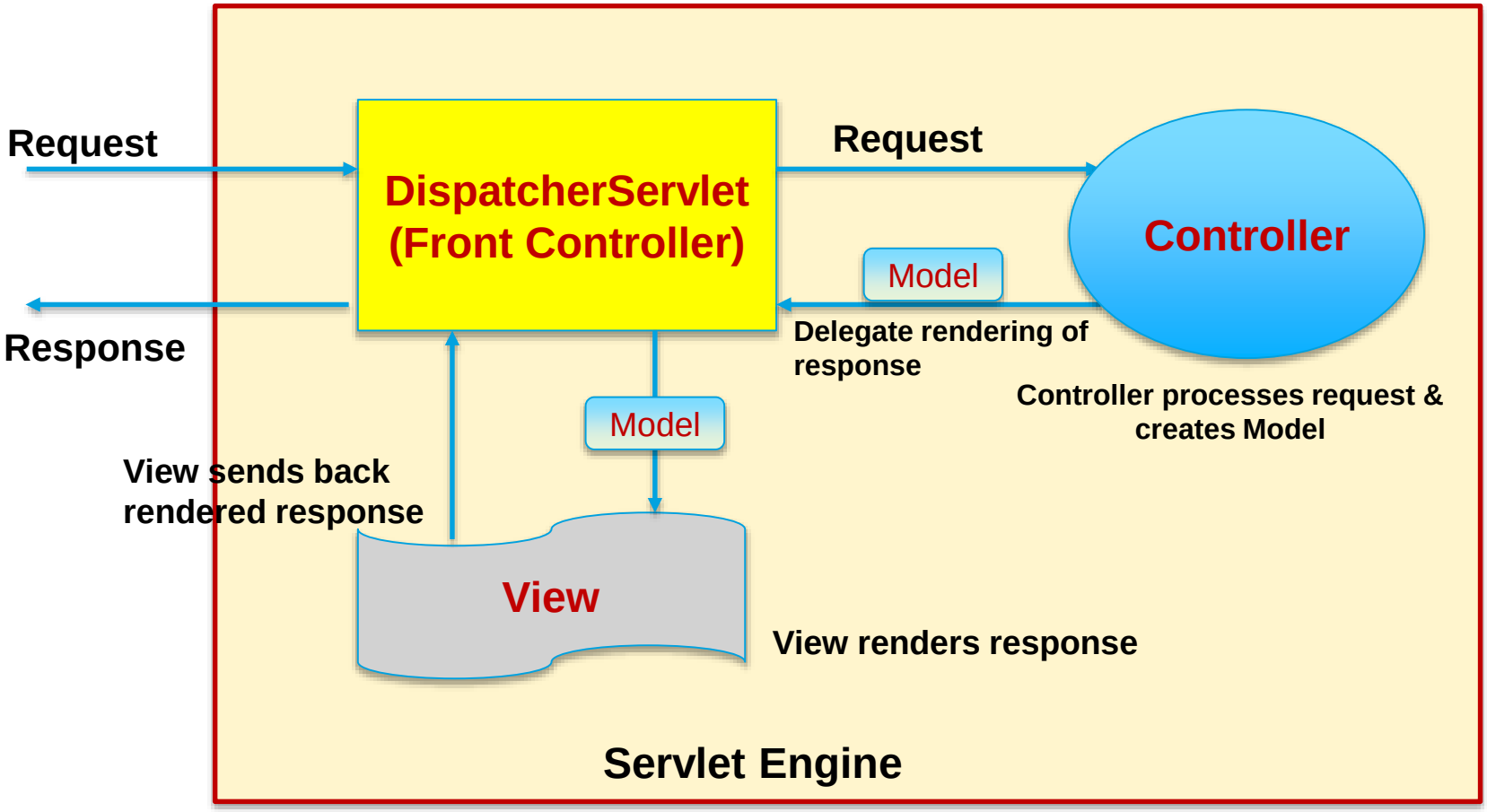
Core Components of Spring MVC (Contd.).

- **ModelAndView**
  - Created by the Controller
  - Stores the Model data
  - Associates a View to the request (usually a logical view name)
  - The ModelAndView class is simply a container by which the model may be transported to and exposed by the view.

- **ViewResolver**
  - Used to map logical View names to actual View implementations

- **HandlerMapping**
  - Strategy interface used by DispatcherServlet
  - for mapping incoming requests to individual Controllers

Work Flow in Spring MVC (High Level)



Lifecycle of a request in Spring MVC

A request leaves the browser asking for a URL and optionally with request parameters.

The request is first examined by **DispatcherServlet.**

DispatcherServlet consults **handler-mappings** defined in a configuration file.

It selects an appropriate **controller** and delegates to it to handle the request.

The **controller** applies appropriate logic to process the request which results in some information (i.e.model).

Lifecycle of a request in Spring MVC (Contd.).

This information is associated with the logical name of a result rendering entity (i.e. **view**)

The entire processed data is returned as a **ModelAndView** object along with the request back to **DispatcherServlet**.

DispatcherServlet then consults the logical view name with a view resolving object to determine the actual view implementation to use.

DispatcherServlet delivers the **model** and **request** to the view implementation which renders an output and sends it back to the browser.

Understanding Life Cycle of a Request

Have you understood the flow described in the previous slide?

Not sure?

Let us now try to understand the flow in our own way..

Let us begin with the client sending a request to the server. Where does this request land?

Yes.. You are right. Requests coming from different clients always land with the Front Controller. In our case, the front controller is the DispatcherServlet.

What happens next?

It is the responsibility of the DispatcherServlet to direct a particular request to an appropriate Controller. How does the DispatcherServlet know, which Controller should process this request?

Continued on next page…

Understanding Life Cycle of a Request (Contd.).

The DispatcherServlet reads the HandlerMapping details provided in the configuration file.

What is HandlerMapping and how does the HandlerMapping code look like ?

HandlerMapping mechanism is used to map URL requests to the name of the beans(In this case, the Controller).

The handler mapping code within the configuration file may look something like this :

```
<beans ...>
  <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"/>
  <bean name="/insert.jsp"
      class="com.wipro.controller.InsertController" />
  <bean name="/display.jsp"
      class="com.wipro.controller.DisplayController" />
</beans>
```

In above example, If URI pattern "/insert.jsp" is requested, the DispatcherServlet will forward the request to the “InsertController“. Similarly, if the URI pattern is “/display.jsp”, it will forward the request to “DisplayController”.

Continued on next page…

Understanding Life Cycle of a Request (Contd.).

Thus, depending upon the configuration details available, the DispatcherServlet will forward the request to appropriate Controller.

The Controller will contain appropriate handler methods, which will have the capability to process user requests.

Once the handler method processes the request, it will return a ModelAndView Object, which will contain the processed data(Model object) along with information related to View.

The information related to view is actually a String, which will be used by a ViewResolver object to decide, which Jsp page(View) is to be invoked by the DispatcherServlet.

What is ViewResolver ? (You will have the answer in the next page)

Understanding Life Cycle of a Request (Contd.).

Most of the MVC Frameworks provide facilities, using which you can work with Views.

Spring provides ViewResolver, which helps us in rendering models in the browser, without being tied to a specific View technology.

ViewResolver maps view names(Strings) to actual views. The DispatcherServlet will work with ViewResolver to know which View(in our case JSP) page will render the output to the client. (If you are wondering how ViewResolver maps view names to actual views, wait till you reach the section, which describes ViewResolver).

Once the DispatcherServlet gets details of the View from ViewResolver, it will forward the request to the View(jsp page).

View uses the Model Object(which contains the processed data) to render the output to the user.

Thank You