**Spring Web MVC and Restful Services**

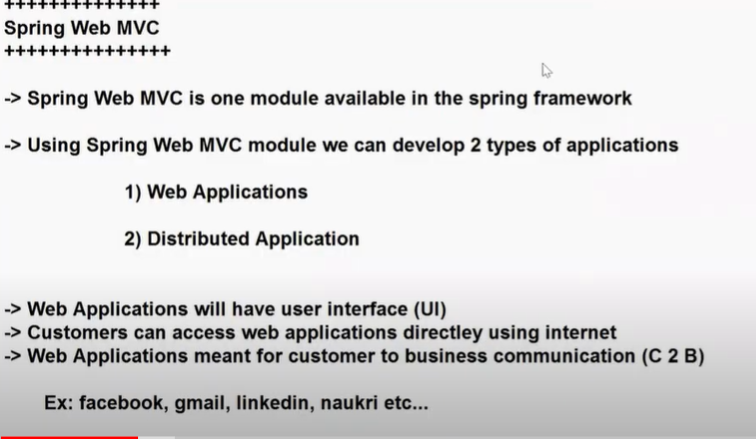
**SBMS-Part2-04:**

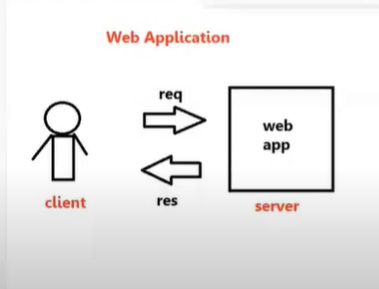
**SBMS-Part2-05 : 3 different ways to send data from controller to UI**

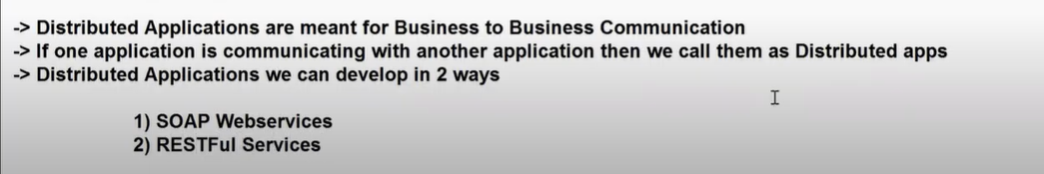
**SBMS-Part2-06: How to develop a form by using Spring Web MVC**

**SBMS-Part2-01**















Example: MakeMyTrip application is having facility to book flight tickets, train tickets, hotel tickets etc..

We have another application called IRCTC.It is the owner of Indian Railways system.How many trains available, how many tickets available,timings of trains…all the details are available with IRCTC .

A user can book the ticket directly from IRCTC.This process is C2B communication.

At the same time, IRCTC having some business integrations also.

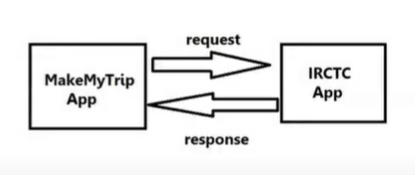
MakeMyTrip,Yatra,GoBibo,etcc…are available. You can book flight tickets using these apps also.

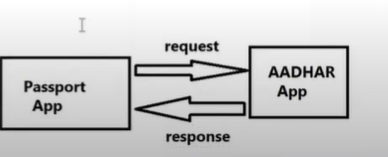
When user books a ticket by using MakeMyTrip, it will further communicate with IRCTC.

IRCTC has already a business logic implemented to book tickets.MakeMyTrip does not implement a new logic to book tickets.Instead it uses logic of IRCTC.

Whenever we want to use the logic of one application in another application, we go for Distributed applications.

If we want to reuse one project functionality in another project, we are going for Distributed applications.

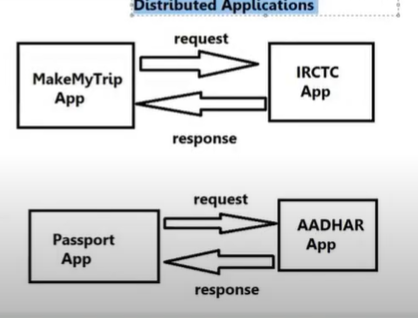


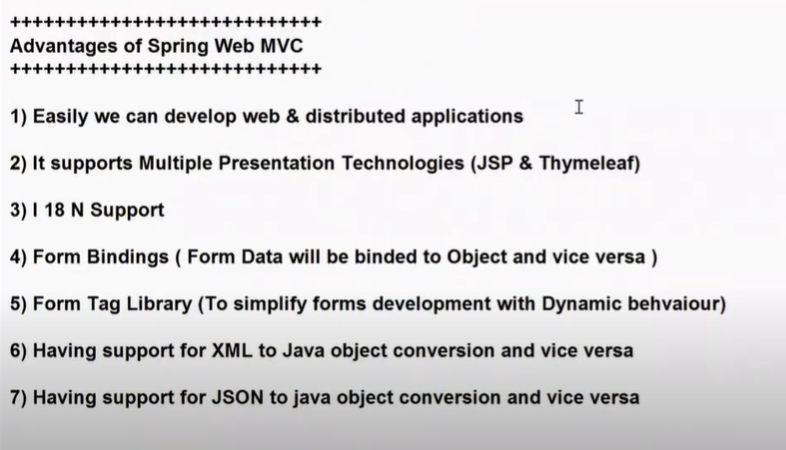


When user applies for a Passport, it asks user to enter Aadhar card number.

Aadhar app has already business logic implemented to check if it’s a valid Aadhar number.

Passport Application is doing one business and Aadhar application is doing one business.As a part of business, passport app wants to communicate with Aadhar app.This is called B2B(Business to Business) communication.





2)With Servlets, we can use only JSP to build Presentation logic.

3)I 18 N--> Internationalization--> Users across Globe can use the application. We can develop our application in multiple languages.

4)Form Bindings: For every application, Forms will be available.Whenever you submit a form, Form data will be binded to object and Object data can be binded to the form.

We don’t need to capture data by using request.getParameter() manually.



**Dispatcher Servlet: It is a predefined servlet class in web MVC. It is going to act as a Front Controller/Framework Servlet.It is responsible to perform pre-processing and post-processing of every request.**

**Pre-processing: Capturing Form data and Binding Form data to object**

**Post-processing: Sending the data to the UI and displaying data in UI**

**Handler Mapper: It is also a predefined class in Spring web MVC.It is used to identify which request should be processed by which controller class.That means it is going to identify Request Handler.**

**Controller: It is a class which will have logic to handle Request and Response.Controller is also called as Request Handler.**

**Controller is a class which contains logic to handle request and response.**

**We will create Controller classes using @Controller annotation**

**Model and View:Model represents data to display in the view and View represents logical file name.So which data should be displayed in which view page is decide by Model and View.**

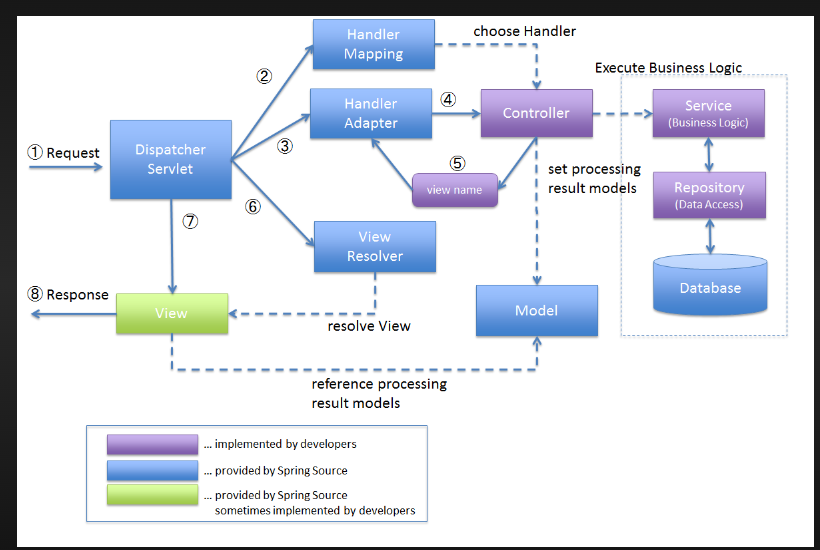
**Model represents data in Key-Value pairs format.**

**View represents presentation file name.**

**To display data in view file we will use ModelandView object**

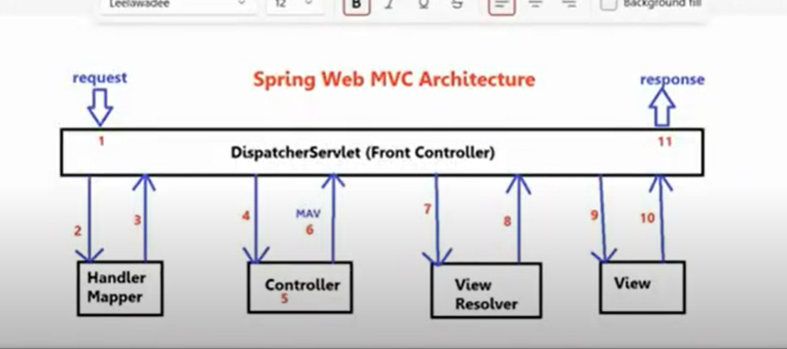
**View Resolver:It is used to identify where the view files are available in the project.**

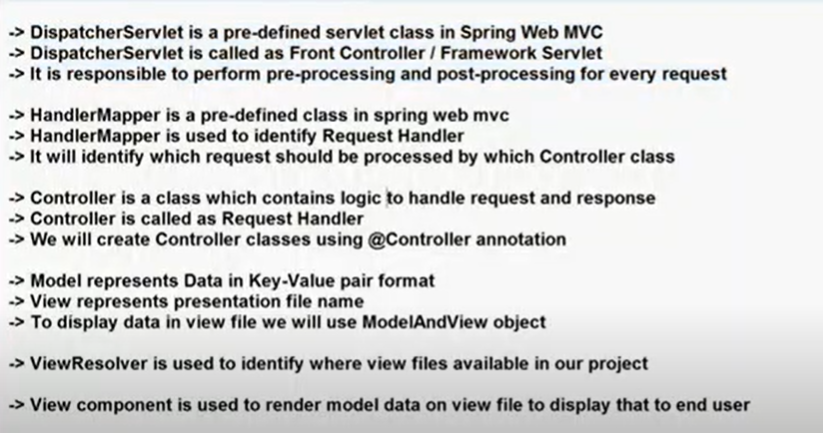
**View: View is used to render model data on view file to display that to end user.**

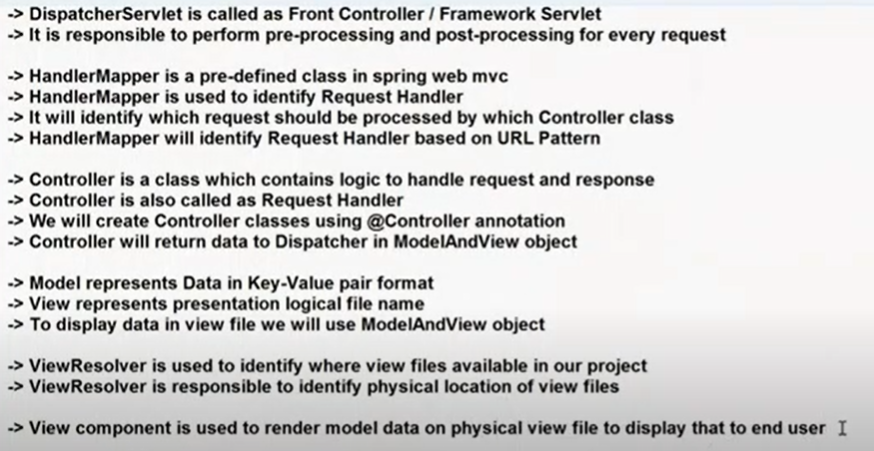


**SBMS-Part2-02**

* Dispatcher servlet will call Handler Mapper.
* Based on URL pattern, Handler Mapper will identify which request should be handled by which controller.If there is no controller with that URL pattern, then it will give 404 error
* Dispatcher servlet will call Controller.Controller will talk to service layer and then service layer will talk to DAO layer.Finally after the method execution, Controller will send data to Dispatcher servlet.
* Controller will send Data to Dispatcher servlet in the form of Model and View object.
* Spring Web MVC Supports multiple presentation technologies.It can be JSP, it can be Thymeleaf.Controller does not know which presentation technology is used in application.Controller just gives Model and View object to Dispatcher servlet.Controller does not know where the view file is present.It just gives view name.
* Dispatcher servlet will give view name to View resolver.View resolver will identify where the view file is present in our project.Physical location of View file will be identified by View resolver.
* Once it is done, Dispatcher servlet will talk to View component.Then view component is going to render that data.Then the data will be given to Dispatcher and Dispatcher will send it as a response to the user.

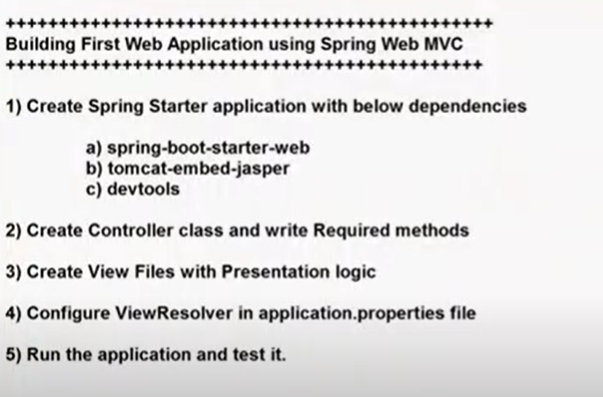








We are going to develop our first web application using Spring Boot.When we use Spring Boot, Spring boot will internally uses Web MVC to develop web applications.



1. spring-boot-starter-web dependency:

It will provide facility to develop a web application,restful services using MVC architecture and it will provide tomcat embedded server.You don’t need to download and setup tomcat server manually.It will be taken care by web-starter.

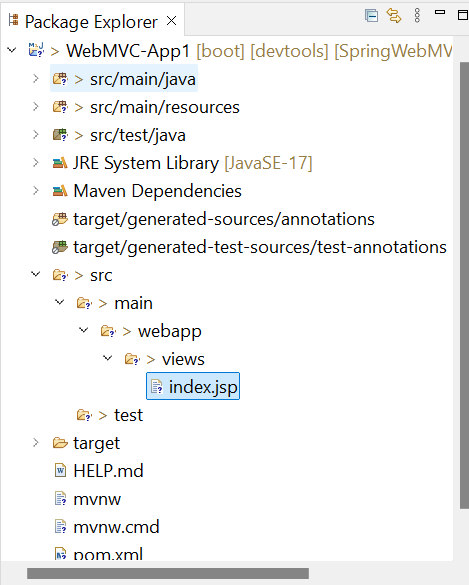
We don’t need to setup the server, install the server, deploy the application to server.The deployment part will be taken care by Spring boot only.

1. Tomcat-embed-jasper:

It will provide support to work with JSP files in Spring Web MVC.By default, Spring web MVC will not understand JSP.With this jasper dependency, Web MVC will understand JSP files and render JSP files.

1. Devtools dependency is used to restart the server when changes happened in the code.Instead of manually restarting the server after making code changes, with this dependency, server will be restarted automatically.
2. Java class will be represented as a Spring controller using @Controller annotation.In one project we can create multiple controllers also.
3. Controller class methods should be binded to HTTP protocol methods to handle HTTP requests.

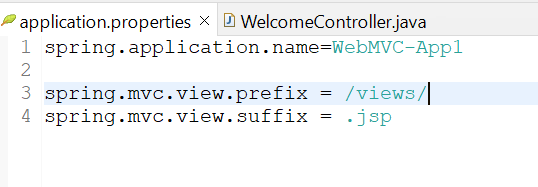
At the time of creating Spring Application, if we select file type as WAR instead of JAR, that means it is a web app.



Configure the ViewResolver in application.properties file with prefix and suffix

Prefix represents where is the view file, it represents physical location of the view.

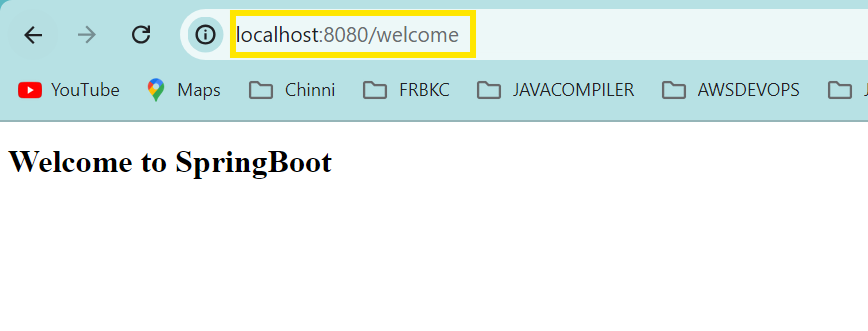
Suffix represents what is the View technology.



View resolver will use prefix + view-name + suffix formula to identify view

/views/ + index + .jsp === /views/index.jsp

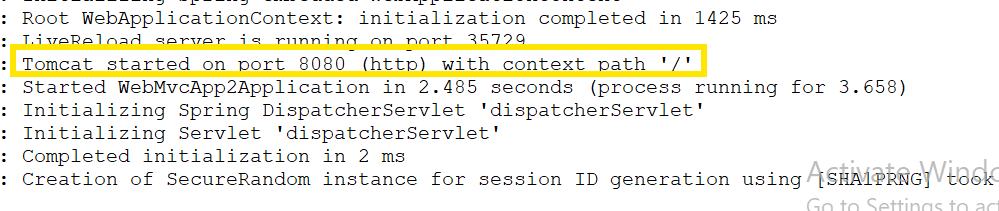
**SBMS-Part2-03**



If you observe,project name is not available in the URL.

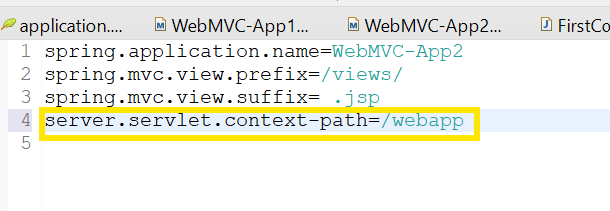
Localhost is server

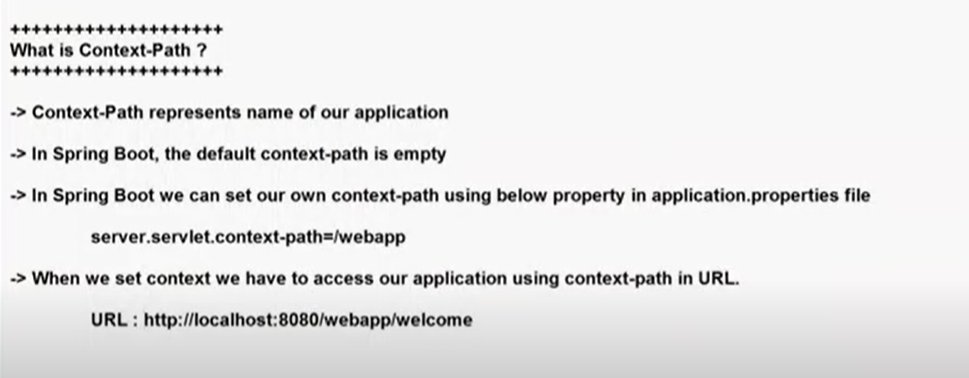
8080 is port number



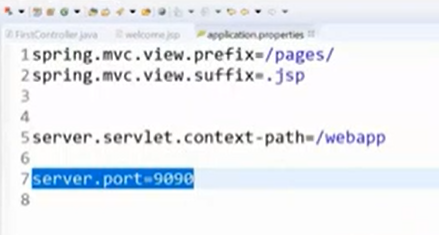
By default context-path will be empty for SpringBoot apps

If you don’t want to keep it empty and want to give name for your project, then configure context-path in application.properties file

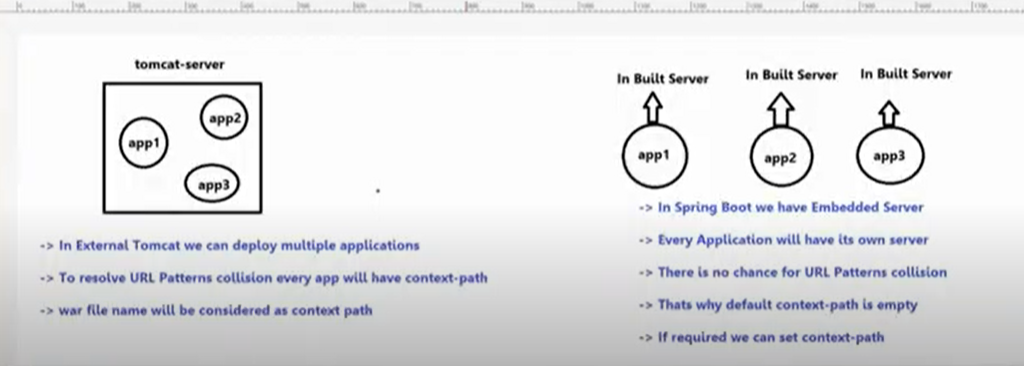




We can change port number by using server.port = 9090 in application.properties file







**SBMS-Part2-04**

Doubts and errors resolving session:

Shortcut to format code: **Ctrl + Shift + F**

Shortcut to comment code: **Ctrl+Shift+/**

Whether you run as java application or spring boot application, there is no difference.Whenever you run Spring Boot app, execution starts from main method.

If we receive 1000 requests to an application, even then only one object will be created for Dispatcher servlet.For each request, a thread will be created and the thread is going to handle that request.

· **One instance** of DispatcherServlet is created in the Spring MVC application (due to its singleton nature).

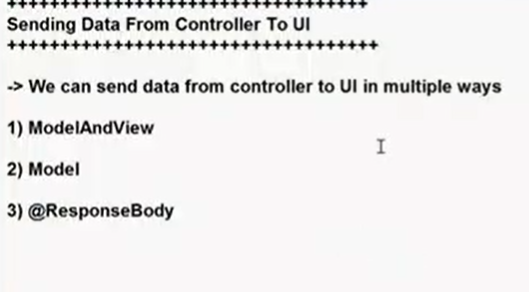
· The **1000 requests** are handled concurrently, typically by using a thread per request, managed by the servlet container.

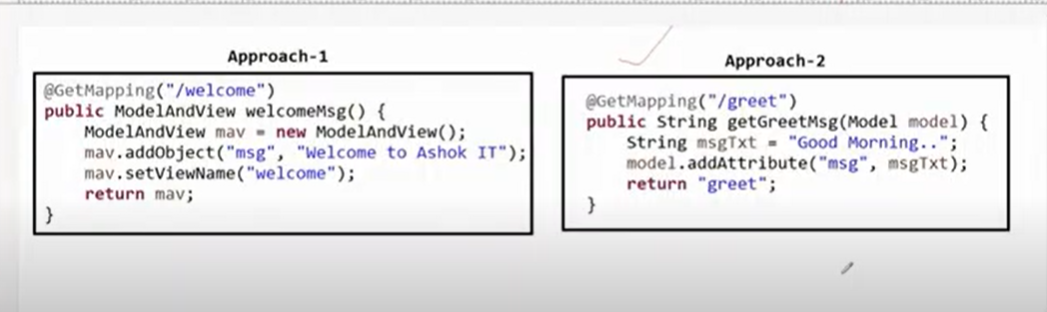
· The DispatcherServlet delegates each request to appropriate controllers, which are also typically singletons and stateless.

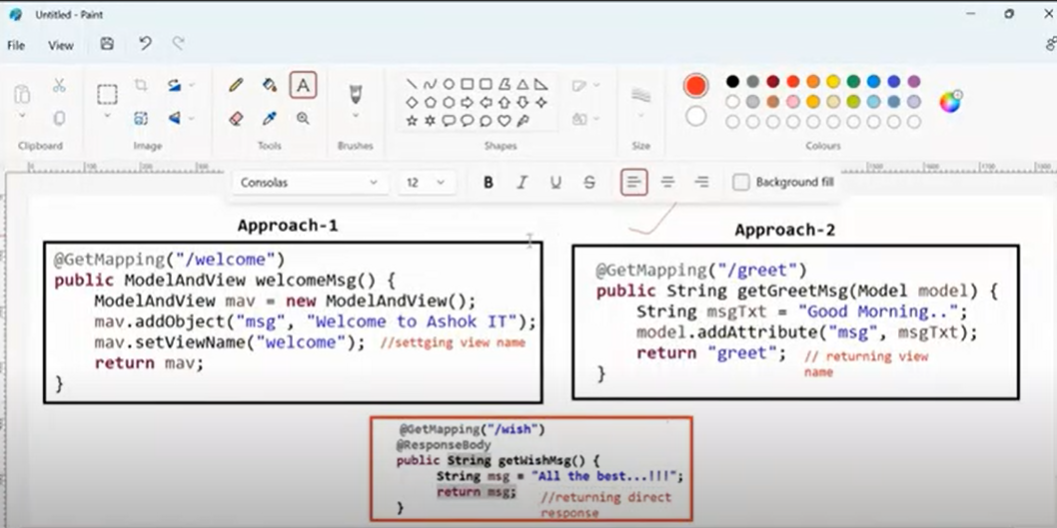
· **Request and Response objects** are created for each individual request, ensuring thread safety and isolation between requests.

**SBMS-Part2-05**

How to send the data from controller to UI in the form of objects?





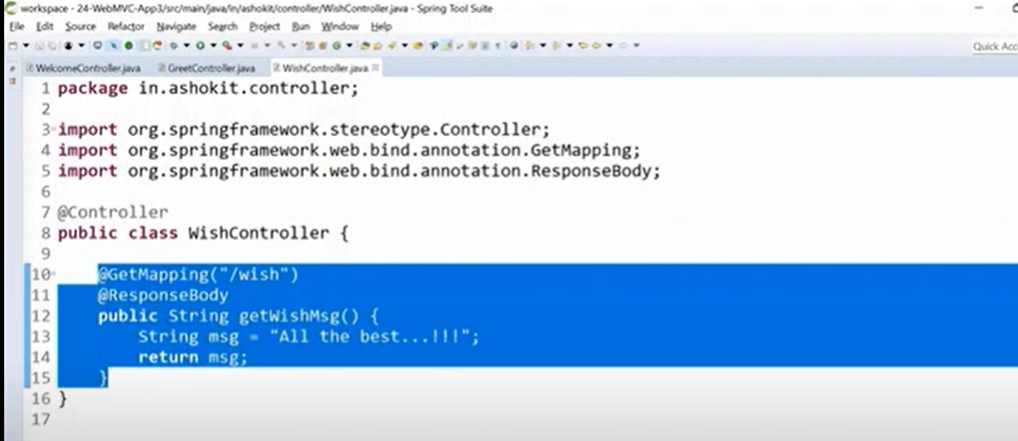


If you want to develop your application with a presentation logic, then you should return the view from controller.

But if you don’t want to develop a presentation logic inside your application and your application will execute only like a RestController, whenever the request comes , send the response directly, then you can go for **approach 3.**

**In approach3 we will not have View resolver.**

**@RestController = @Controller +@ResponseBody**



The @ResponseBody annotation in Spring Boot is used to indicate that the return value of a method should be written directly to the HTTP response body, rather than being interpreted as a view name. This is particularly useful in RESTful web services where the response is typically in a format like JSON or XML rather than an HTML page.

### Key Points about @ResponseBody:

**Direct Response Writing**:

* 1. When you annotate a method with @ResponseBody, Spring converts the returned object into a format that can be directly written to the HTTP response body (e.g., JSON, XML).
  2. For example, if the method returns a Map or a custom object, it will be converted to JSON (by default) and sent back to the client.

**Used in REST Controllers**:

* 1. In RESTful web services, @ResponseBody is commonly used to send JSON or XML data in the response.
  2. When combined with @RequestMapping or other request mapping annotations, it allows you to create REST endpoints easily.

**Implicit in** @RestController:

* 1. If your class is annotated with @RestController, which is a combination of @Controller and @ResponseBody, you do not need to use @ResponseBody on every method. The @RestController annotation automatically applies @ResponseBody to all methods in the class.

**Serialization and Content Negotiation**

* 1. The @ResponseBody annotation triggers the HttpMessageConverter mechanism, which converts the returned object to the appropriate format based on the Content-Type header of the request.
  2. By default, Spring uses Jackson (if on the classpath) for converting objects to JSON and JAXB for XML.

**Whenever you go for RestControllers or Angular as UI or React as UI, our project does not contain web pages.Our project will only contain Logic**

**When to return View name from our projects? When to return direct response from our projects?**

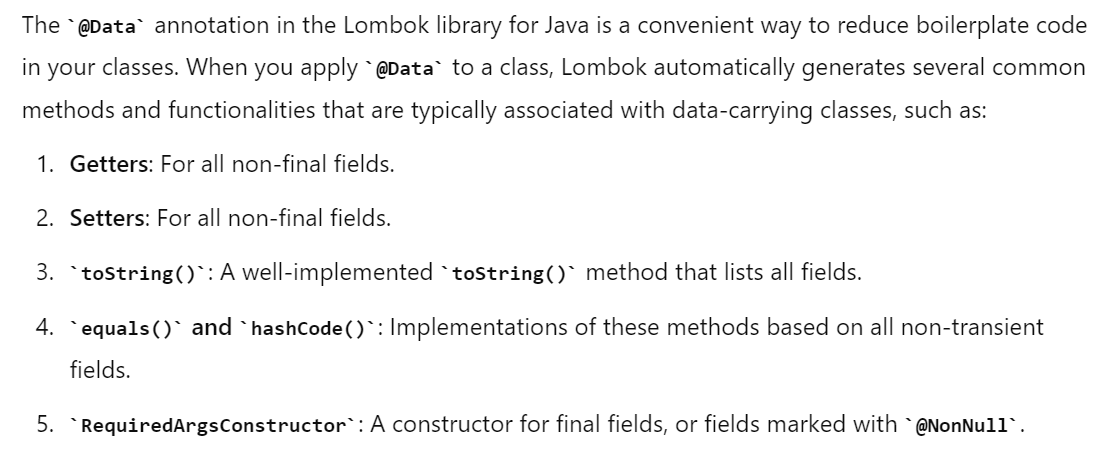
If you want to develop your application along with Presentation logic,then we will return the view pages.

When you want to develop a B2B application or when you want to develop a REST API, then we are going to use @ResponseBody.We will directly return the response.

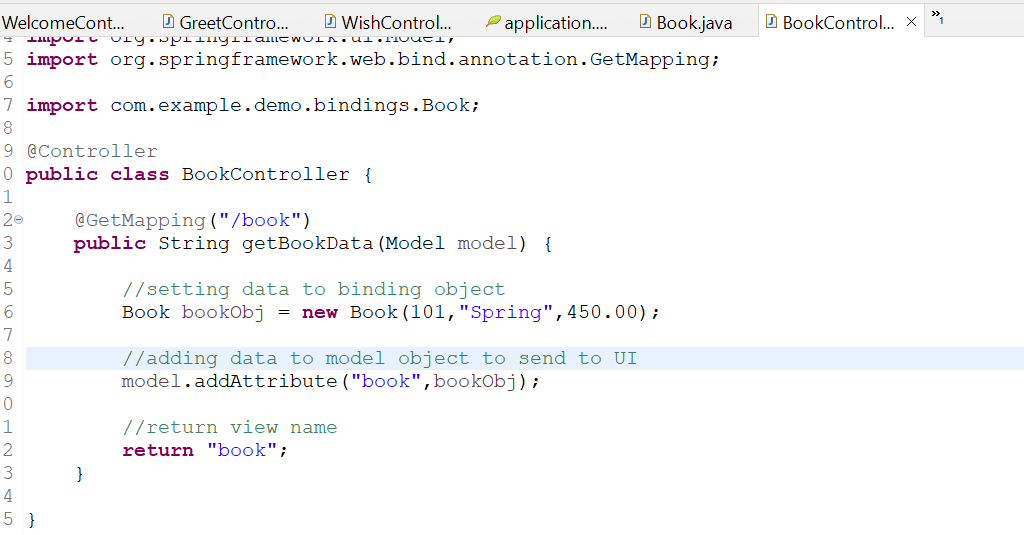
In Industries, now a days, they are using SpringBoot to develop Microservices(RESTAPIs) and they have Angular or React in Frontend.

If you give @ResponseBody annotation at method level, then it is applicable only to that method.But if you give that annotation at class level, then it is applicable to all methods.

When to use @ResponseBody annotation in the web application?



**Sending object data from BookController to UI**



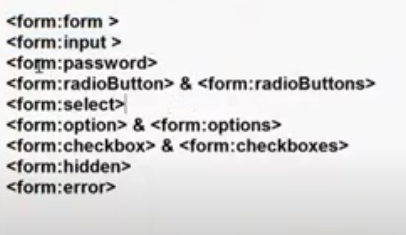


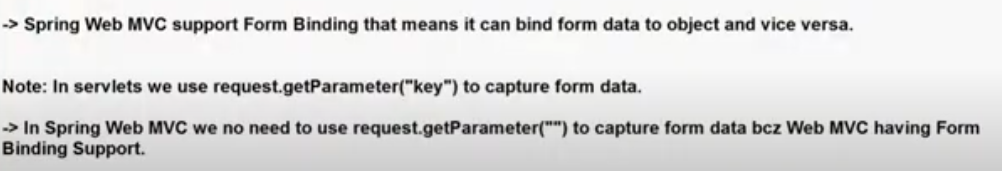
Error to be resolved

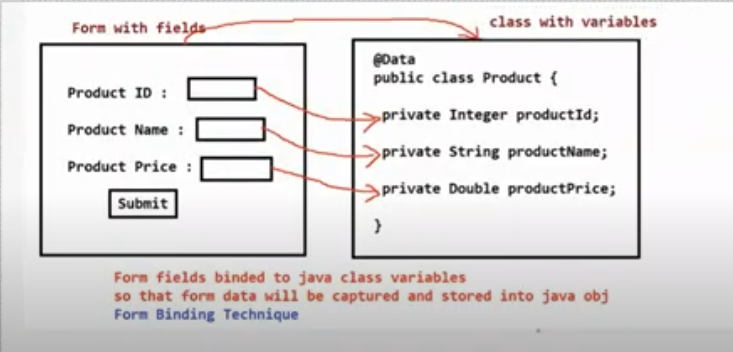
**SBMS-Part2-06**

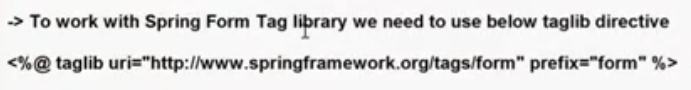
**Forms development using Spring Web MVC**





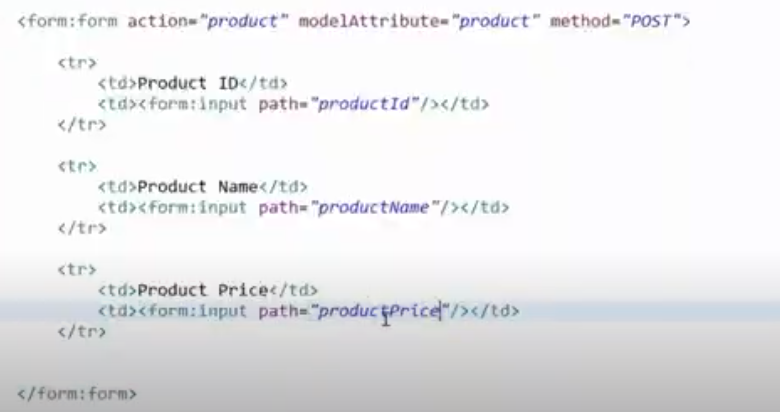




---outdated content--not so important



We use path attribute to maintain binding between form elements and Java class attributes.



When tomcat-embed-jasper dependency is not added in pom.xml file, JSP is going to download instead of rendering