189: Web App Introduction:

Spring has multiple projects

Most of the mobile applications also have web applications.

Server components which will process and in fact it will accept the request. It will process the request, it will send data back to the client.

For running servlets, we need to have Special container called Servlet container. These servlets cannot be run on an JVM as we are getting request from the internet and sending back the data .

Tomcat is server in which we can run our servlets. Even if we are using Spring boot Web or Spring MVC behind the scenes it would be using Servlets.

190:

Creating a Servlet Project:

If you are building a web application and want to run it on an server. We need to create a package of it as war(web Archive) and keep it in the Tomcat Server.

If it is an console based application we can use .jar file

So we need an Tomcat server in our machine to run that project.

In the WebApp Folder we will placing our project that we wanted to run.

For Start and Shutdown of Project:

Go to bin folder and it will files

stratup.sh ---> For starting the server

shutdown.sh ---> For Shutdowing the Server.

We can also have Embedded Tomcat in our Projects.

Servlet is not part of JDk.so we need to add extra dependency. (jakarta.servlet-api)

For Embedded Tomcat as well we need to dependency(tomcat-embed-core).

Once we add the dependencies and save the Project the corresponding jar files will be downloaded into the project.

191: Running Tomcat

Servlet has features like accepting the input from the user and responding to the user.

We can make a class as Servlet by extending it from the HttpServlet

If we want to send request we need to make request to browser.

```
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.*;

public class ServletExample extends HttpServlet {

     //This method gets called whenever client makes request
   public void service(HttpServletRequest req, HttpServletResponse res)
   {
      System.out.println("Servlet is running");
   }
}
```

Tomcat by default goes with 8080 port.

By default Tomcat server will not be running we need to manually start it.

```
import org.apache.catalina.Context;
import org.apache.catalina.LifecycleException;
import org.apache.catalina.startup.Tomcat;

import java.io.*;
import java.util.*;

public class Application {
   public static void main(String[] args) throws LifecycleException, InterruptedException {
      System.out.println("Hello World");
      Tomcat tomcat=new Tomcat();
      //tomcat.setPort(8888);
      //System.out.println("Hello World");
      Context context=tomcat.addContext("",null);
      Tomcat.addServlet(context,"ServletExample",new ServletExample());
      context.addServletMappingDecoded("/hello","ServletExample");

      tomcat.start();
      //Making the server wait
     tomcat.getServer().await();
      tomcat.stop();
      System.out.println("Hi");
   }
}
```

If we dont have an Embedded Tomcat then we can provide the url and mapping related to it web.xml or else in the Annotation based way like below

```
@WebServlet("/hello")
public class ServletExample extends HttpServlet {

   public void service(HttpServletRequest req, HttpServletResponse res)
   {
      System.out.println("Servlet is running");
   }
}
```

199:Responding to Client

The below Code will print HelloWorld in the browser.

```
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.*;

@WebServlet("/hello")
public class ServletExample extends HttpServlet {

   public void service(HttpServletRequest req, HttpServletResponse res) throws
IOException {
        //This Code will print the HelloWorld in the browser.
        res.getWriter().println("Hello World");
        System.out.println("Servlet is running");
    }
}
```

Servlet Methods:

doGet()

```
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.*;

@WebServlet("/hello")
public class ServletExample extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException
{
        //This Code will print the HelloWorld in the browser.
        res.setContentType("text/html");
       PrintWriter out = res.getWriter();
        out.println("Hello World");
    }
}
```

doPost()

doPut()

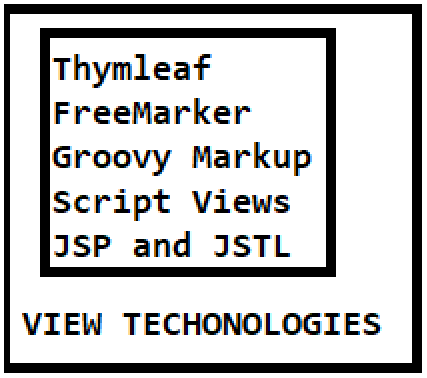
doDelete()

200: Introduction to MVC

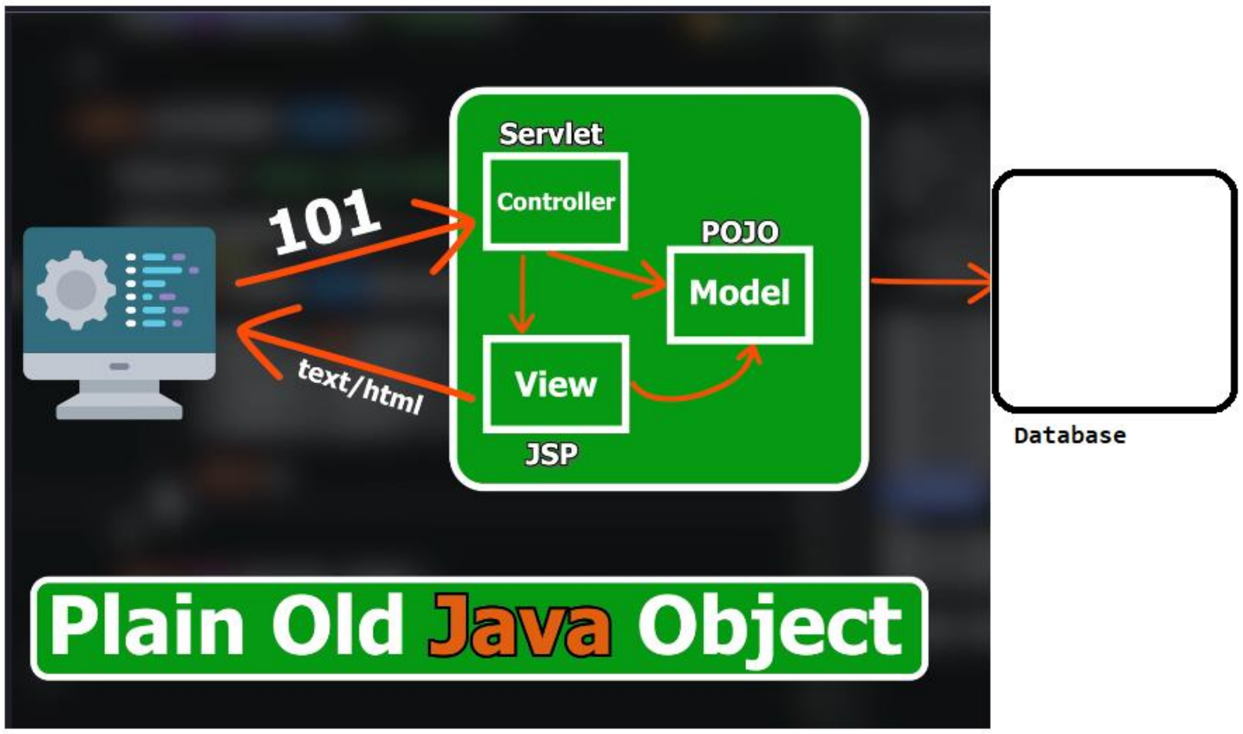
MVC-Model View Controller

JSP-Java Server Pages (For Viewing the data)

POJO -Plain Old Java Object.



View Technologies



Tomcat is Servlet Container

Even though we have JSP Pages in our code they will be converted to servlets.

201.Creating a Spring Boot WebApp Project.

It order to run an Application on Server we need to make that as package then we can run it on server. as of now we are working on web application, so we are going with war (Web Archive).

Project which we download with spring boot web will have Embedded tomcat in it.

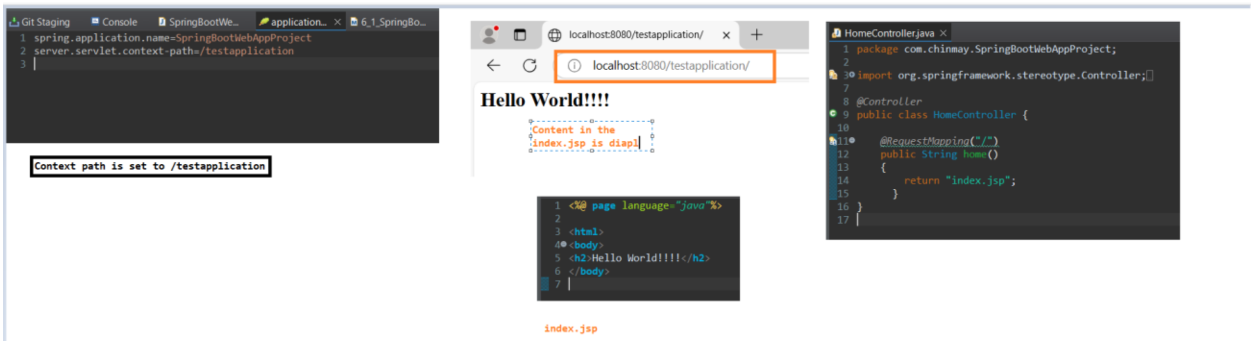
202: Creating a JSP Page

Spring will look for webapp folder .

jsp will called by the controller.

203: Creating a Controller

204:Request Mapping



```
spring.application.name=SpringBootWebAppProject
server.servlet.context-path=/testapplication
```

Context path is set to /testapplication

localhost:8080/testapplication/

Hello World!!!!

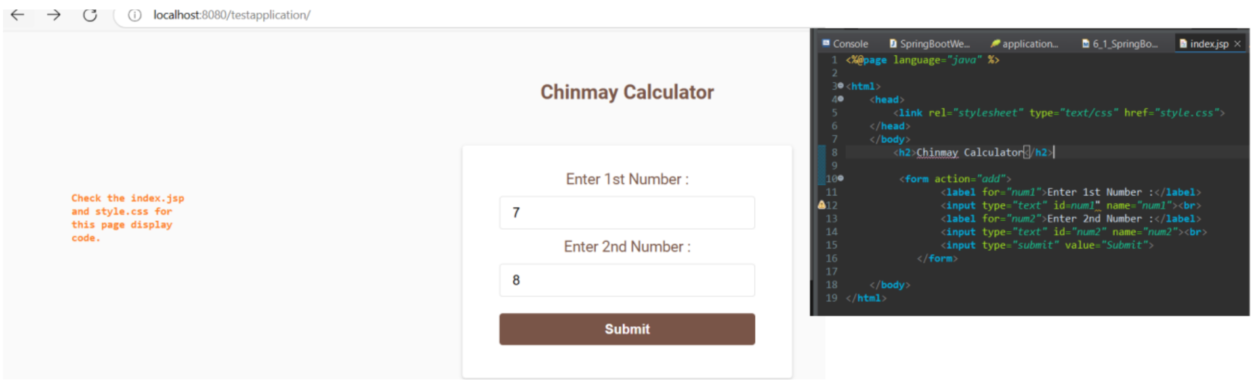
Content in the index.jsp is diapl

```
<%@ page language="java"%>

<html>
<body>
<h2>Hello World!!!!</h2>
</body>
```

index.jsp

```java
package com.chinmay.SpringBootWebAppProject;

import org.springframework.stereotype.Controller;

@Controller
public class HomeController {

    @RequestMapping("/")
    public String home()
    {
        return "index.jsp";
      }
}
```

205:Sending Data to the Controller



localhost:8080/testapplication/

Chinmay Calculator

Enter 1st Number :

7

Enter 2nd Number :

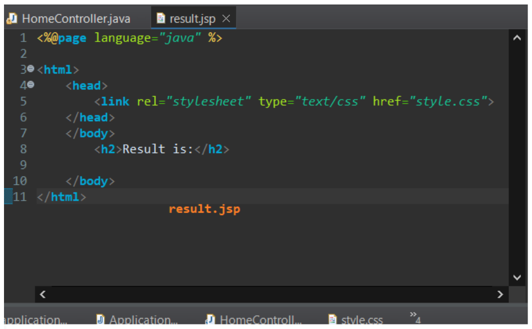
8

Submit

Check the index.jsp and style.css for this page display code.

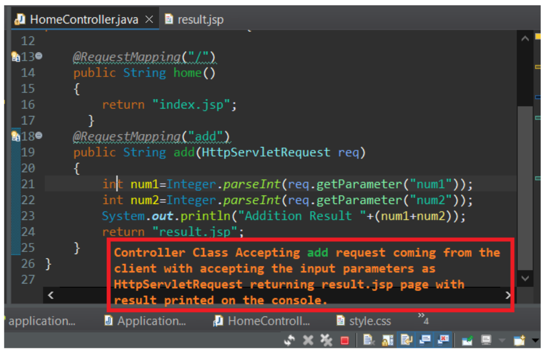
```
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    </body>
        <h2>Chinmay Calculator</h2>

         <form action="add">
                <label for="num1">Enter 1st Number :</label>
                <input type="text" id=num1" name="num1"><br>
                <label for="num2">Enter 2nd Number :</label>
                <input type="text" id="num2" name="num2"><br>
                <input type="submit" value="Submit">
            </form>

    </body>
</html>
```

206: Accepting the Data the Servlet Way



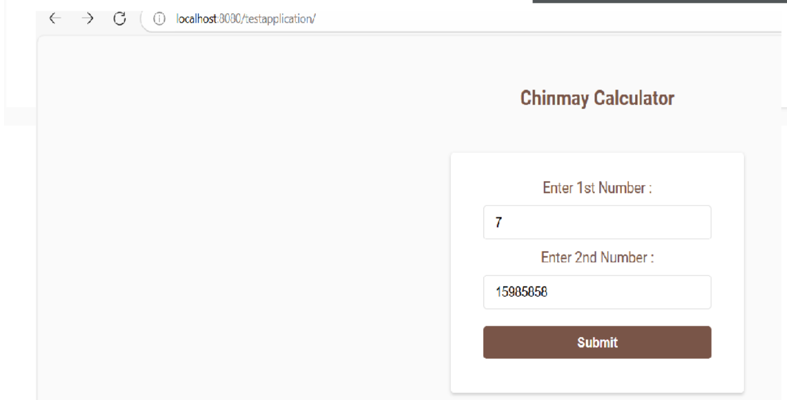
```
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    </body>
        <h2>Result is:</h2>

    </body>
</html>
```

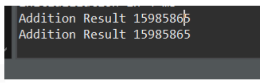
result.jsp



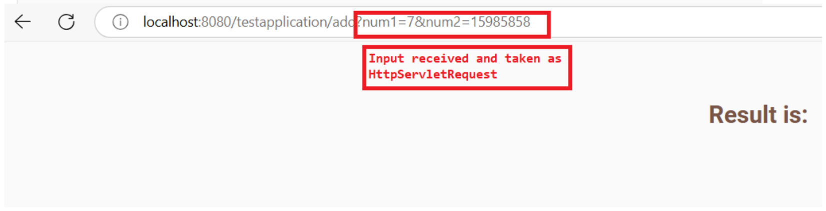
```
    @RequestMapping("/")
    public String home()
    {
        return "index.jsp";
      }
    @RequestMapping("add")
    public String add(HttpServletRequest req)
    {
        int num1=Integer.parseInt(req.getParameter("num1"));
        int num2=Integer.parseInt(req.getParameter("num2"));
        System.out.println("Addition Result "+(num1+num2));
        return "result.jsp";
    }
}
```

Controller Class Accepting add request coming from the client with accepting the input parameters as HttpServletRequest returning result.jsp page with result printed on the console.





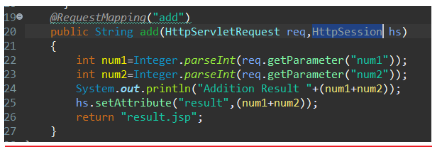
```
Addition Result 15985865
Addition Result 15985865
```



Input received and taken as HttpServletRequest

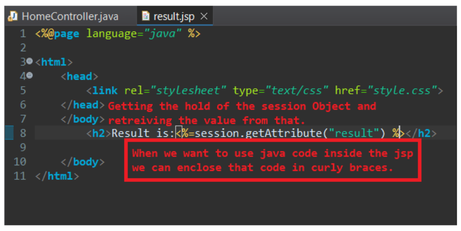
Dispatcher servlet is responsible for Mapping the Requests.

207: Display data on Result page



```
    @RequestMapping("add")
    public String add(HttpServletRequest req,HttpSession hs)
    {
        int num1=Integer.parseInt(req.getParameter("num1"));
        int num2=Integer.parseInt(req.getParameter("num2"));
        System.out.println("Addition Result "+(num1+num2));
        hs.setAttribute("result",(num1+num2));
        return "result.jsp";
    }
```

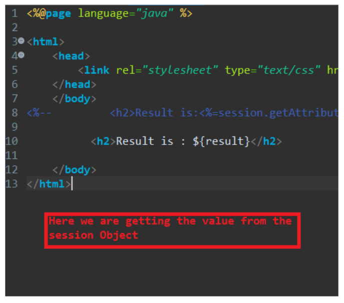
we can put the data in a HttpSession object.
So basically when you have multiple pages and if a user is accessing multiple pages, you want to maintain this data between the pages.
we can get help of session, and to get the hold on a session we need to use HttpSession -- is an interface and Spring will inject Reference for it.



```
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    </body>
        <h2>Result is:<%=session.getAttribute("result") %></h2>

    </body>
</html>
```

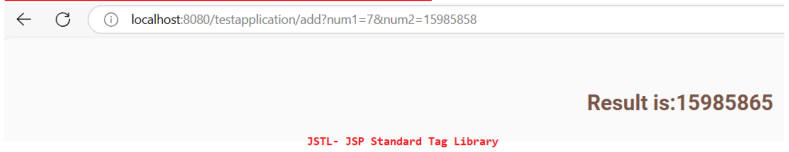
Getting the hold of the session Object and retreiving the value from that.

When we want to use java code inside the jsp we can enclose that code in curly braces.



```
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" hr
    </head>
    </body>
<%--        <h2>Result is:<%=session.getAttribut

          <h2>Result is : ${result}</h2>

    </body>
</html>
```

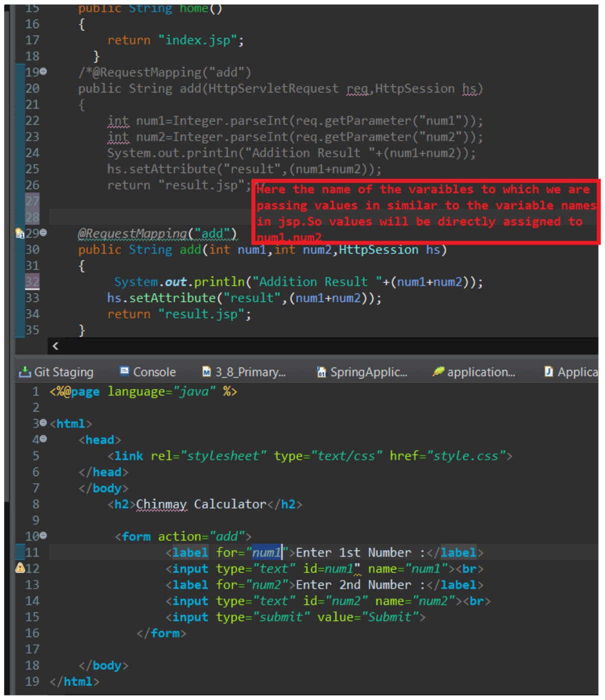
Here we are getting the value from the session Object



JSTL- JSP Standard Tag Library

We can also get the value from the Session Object in the below way as well.
<h2>Result is : ${result}</h2>

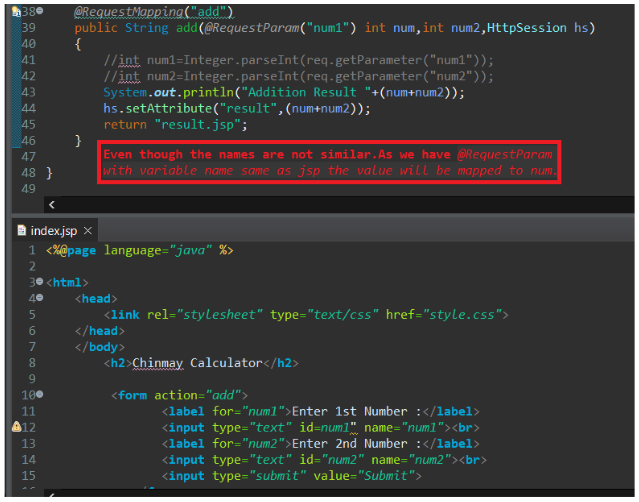
208 .RequestParam



```java
    public String home()
    {
        return "index.jsp";
      }
    /*@RequestMapping("add")
    public String add(HttpServletRequest req,HttpSession hs)
    {
        int num1=Integer.parseInt(req.getParameter("num1"));
        int num2=Integer.parseInt(req.getParameter("num2"));
        System.out.println("Addition Result "+(num1+num2));
        hs.setAttribute("result",(num1+num2));
        return "result.jsp";*/


    @RequestMapping("add")
    public String add(int num1,int num2,HttpSession hs)
    {
         System.out.println("Addition Result "+(num1+num2));
        hs.setAttribute("result",(num1+num2));
        return "result.jsp";
    }
```

Here the name of the varaibles to which we are passing values in similar to the variable names in jsp.So values will be directly assigned to num1,num2

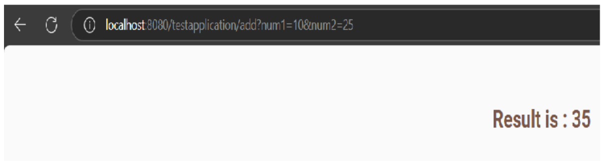
```jsp
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    </body>
        <h2>Chinmay Calculator</h2>

         <form action="add">
                <label for="num1">Enter 1st Number :</label>
                <input type="text" id=num1" name="num1"><br>
                <label for="num2">Enter 2nd Number :</label>
                <input type="text" id="num2" name="num2"><br>
                <input type="submit" value="Submit">
            </form>

    </body>
</html>
```



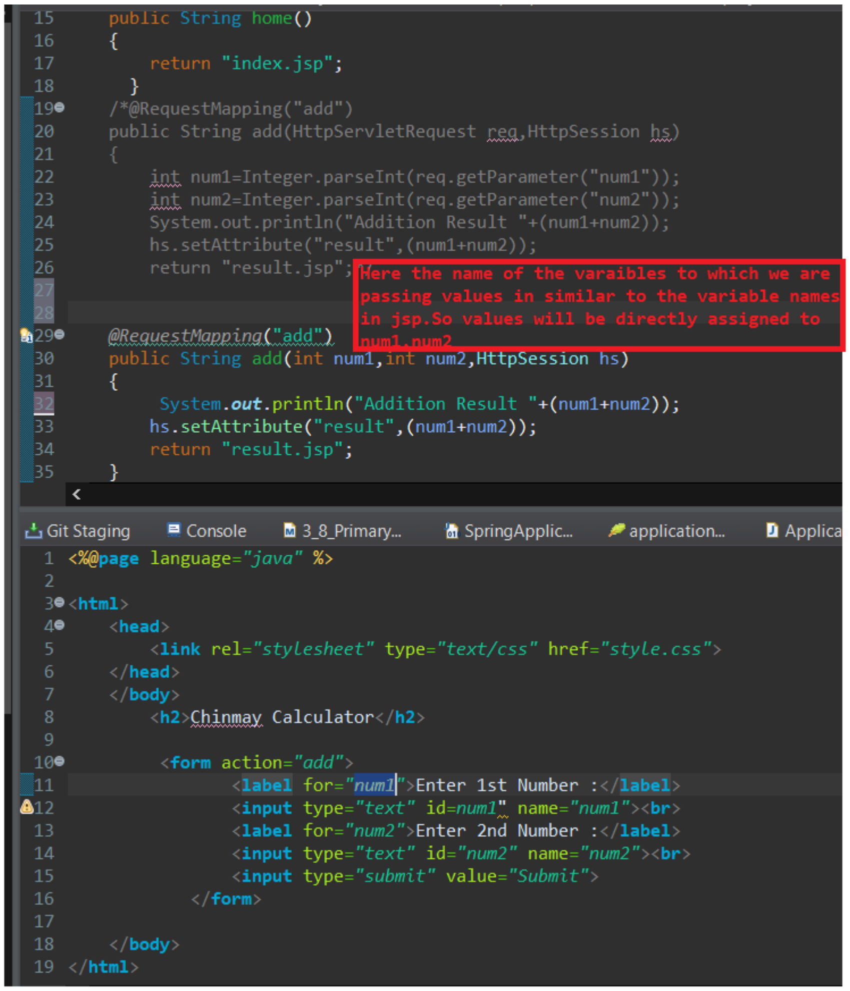
```java
    @RequestMapping("add")
    public String add(@RequestParam("num1") int num,int num2,HttpSession hs)
    {
        //int num1=Integer.parseInt(req.getParameter("num1"));
        //int num2=Integer.parseInt(req.getParameter("num2"));
        System.out.println("Addition Result "+(num+num2));
        hs.setAttribute("result",(num+num2));
        return "result.jsp";
    }

}
```

Even though the names are not similar.As we have @RequestParam with variable name same as jsp the value will be mapped to num.

```jsp
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    </body>
        <h2>Chinmay Calculator</h2>

         <form action="add">
                <label for="num1">Enter 1st Number :</label>
                <input type="text" id=num1" name="num1"><br>
                <label for="num2">Enter 2nd Number :</label>
                <input type="text" id="num2" name="num2"><br>
                <input type="submit" value="Submit">
```



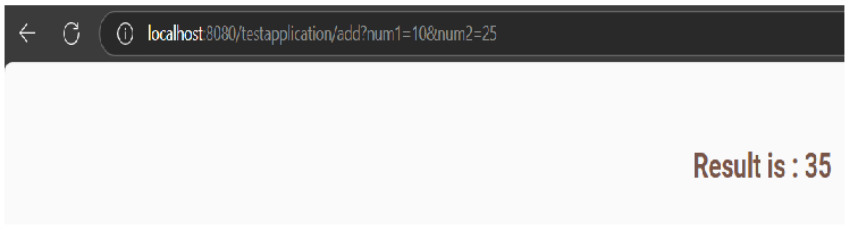
Result is : 35



```
    public String home()
    {
        return "index.jsp";
      }
    /*@RequestMapping("add")
    public String add(HttpServletRequest req,HttpSession hs)
    {
        int num1=Integer.parseInt(req.getParameter("num1"));
        int num2=Integer.parseInt(req.getParameter("num2"));
        System.out.println("Addition Result "+(num1+num2));
        hs.setAttribute("result",(num1+num2));
        return "result.jsp";


    @RequestMapping("add")
    public String add(int num1,int num2,HttpSession hs)
    {
         System.out.println("Addition Result "+(num1+num2));
        hs.setAttribute("result",(num1+num2));
        return "result.jsp";
    }
```

Here the name of the varaibles to which we are passing values in similar to the variable names in jsp.So values will be directly assigned to num1,num2

```
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    </body>
        <h2>Chinmay Calculator</h2>

         <form action="add">
                <label for="num1">Enter 1st Number :</label>
                <input type="text" id=num1" name="num1"><br>
                <label for="num2">Enter 2nd Number :</label>
                <input type="text" id="num2" name="num2"><br>
                <input type="submit" value="Submit">
            </form>

    </body>
</html>
```

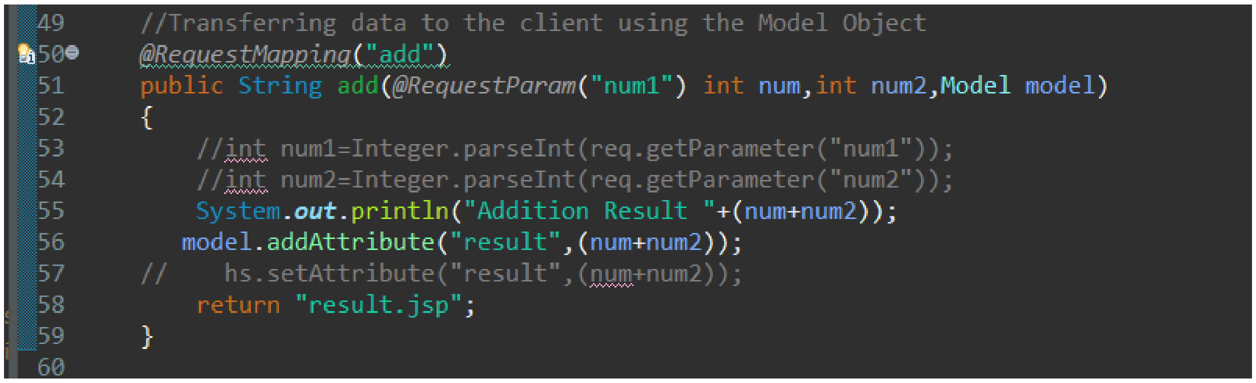


localhost:8080/testapplication/add?num1=10&num2=25

Result is : 35

209 : Model Object

To transfer data b/w controller and jsp we can use Model Object



```java
    //Transferring data to the client using the Model Object
    @RequestMapping("add")
    public String add(@RequestParam("num1") int num,int num2,Model model)
    {
        //int num1=Integer.parseInt(req.getParameter("num1"));
        //int num2=Integer.parseInt(req.getParameter("num2"));
        System.out.println("Addition Result "+(num+num2));
       model.addAttribute("result",(num+num2));
    //    hs.setAttribute("result",(num+num2));
        return "result.jsp";
    }
```

We can add the data to Model Object and jsp will retrieve the data.

210: Setting Prefix and suffix:

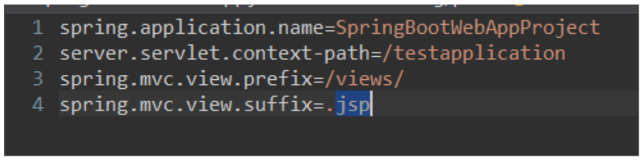
when our jsp files were in other folder

we are returning the file names without extension then in it that case in -order to make our application work we need to provide.

Spring Framework has a view Resolver.To view Resolver we can provide the properties where it can find the files and extension of them by providing them in the application.properties file

Commit : Moved the jsp/css to views folder and provided the properties to the

Commit 1312994

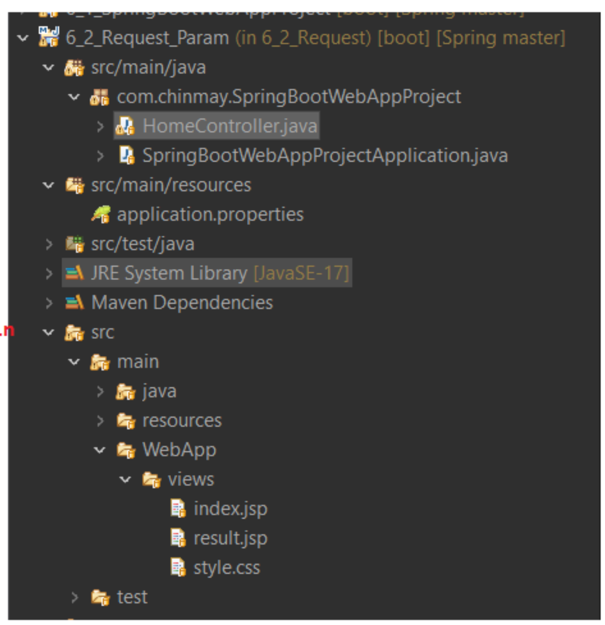


```
spring.application.name=SpringBootWebAppProject
server.servlet.context-path=/testapplication
spring.mvc.view.prefix=/views/
spring.mvc.view.suffix=.jsp
```

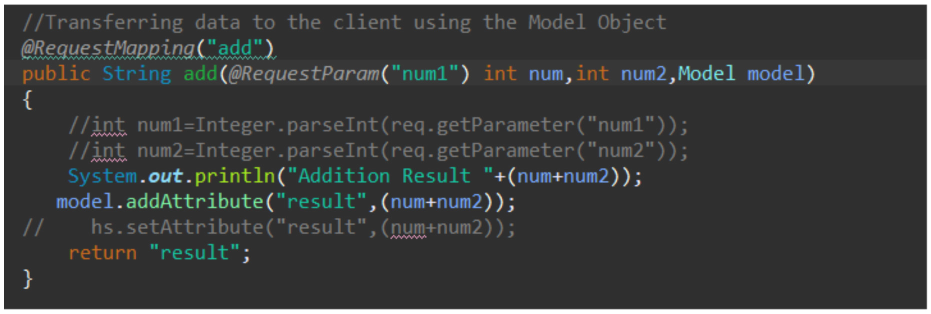
Moved the jsp/css to views folder and provided the properties to the
view and provide the view details as prefix in application.properties
file

Removed the file extension and provided it as property in application.properties file as suffix.

View Resolver takes the provided properties and resolve the view issues.



```
6_2_Request_Param (in 6_2_Request) [boot] [Spring master]
  src/main/java
    com.chinmay.SpringBootWebAppProject
      HomeController.java
      SpringBootWebAppProjectApplication.java
  src/main/resources
    application.properties
  src/test/java
  JRE System Library [JavaSE-17]
  Maven Dependencies
  src
    main
      java
      resources
      WebApp
        views
          index.jsp
          result.jsp
          style.css
    test
```

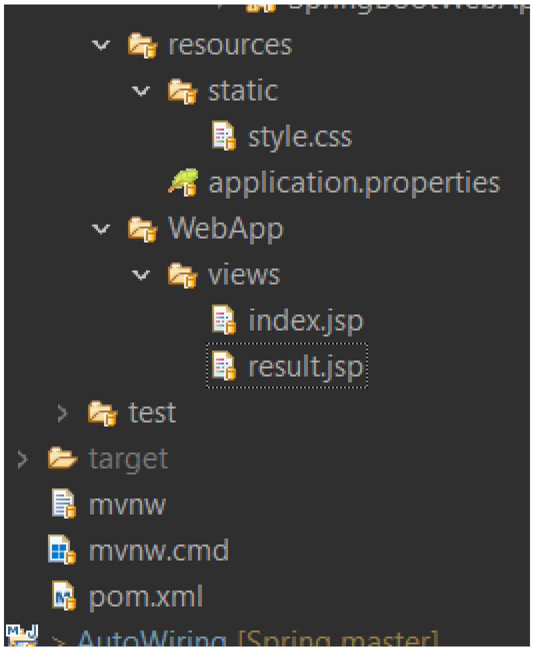


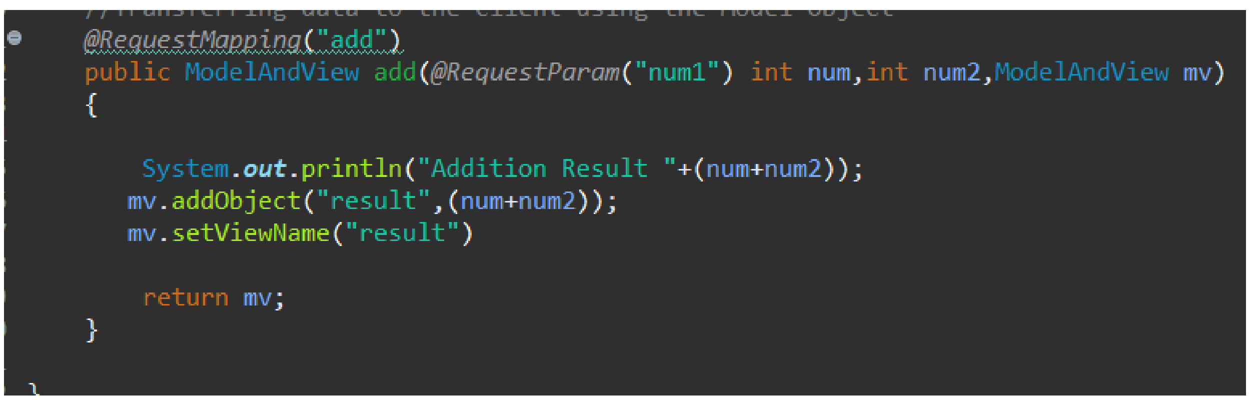
```
//Transferring data to the client using the Model Object
@RequestMapping("add")
public String add(@RequestParam("num1") int num,int num2,Model model)
{
    //int num1=Integer.parseInt(req.getParameter("num1"));
    //int num2=Integer.parseInt(req.getParameter("num2"));
    System.out.println("Addition Result "+(num+num2));
   model.addAttribute("result",(num+num2));
//    hs.setAttribute("result",(num+num2));
    return "result";
}
```

211 .Model and View

Moved the style.css to static folder.

It can be place in static folder or webapp folder.


```
resources
  static
    style.css
  application.properties
WebApp
  views
    index.jsp
    result.jsp
test
target
mvnw
mvnw.cmd
pom.xml
```


```
    @RequestMapping("add")
    public ModelAndView add(@RequestParam("num1") int num,int num2,ModelAndView mv)
    {

        System.out.println("Addition Result "+(num+num2));
       mv.addObject("result",(num+num2));
       mv.setViewName("result")

        return mv;
    }
```

```
Instead of returning Model and View Seperatley we can return both of them at a time
using ModelAndView .

We can Model(data) as Object to ModelAndView
       view as setViewName to ModelAndView

            Project-- 6_3_Model_View
```

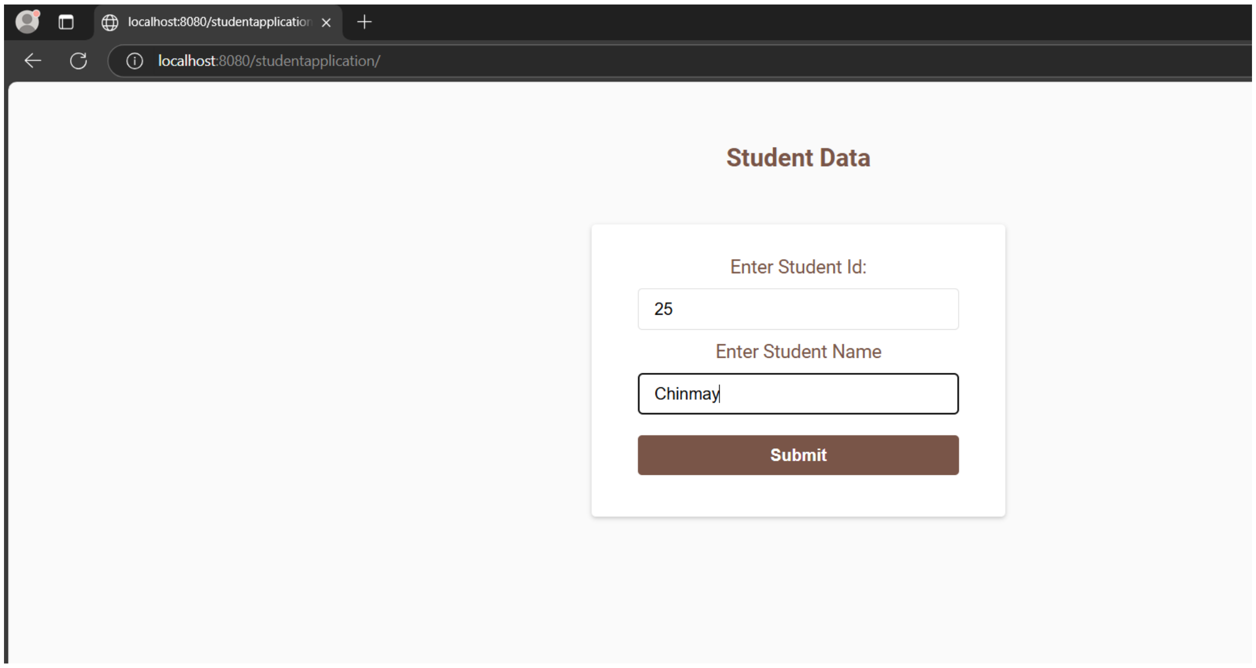
212: Need for Model Attribute

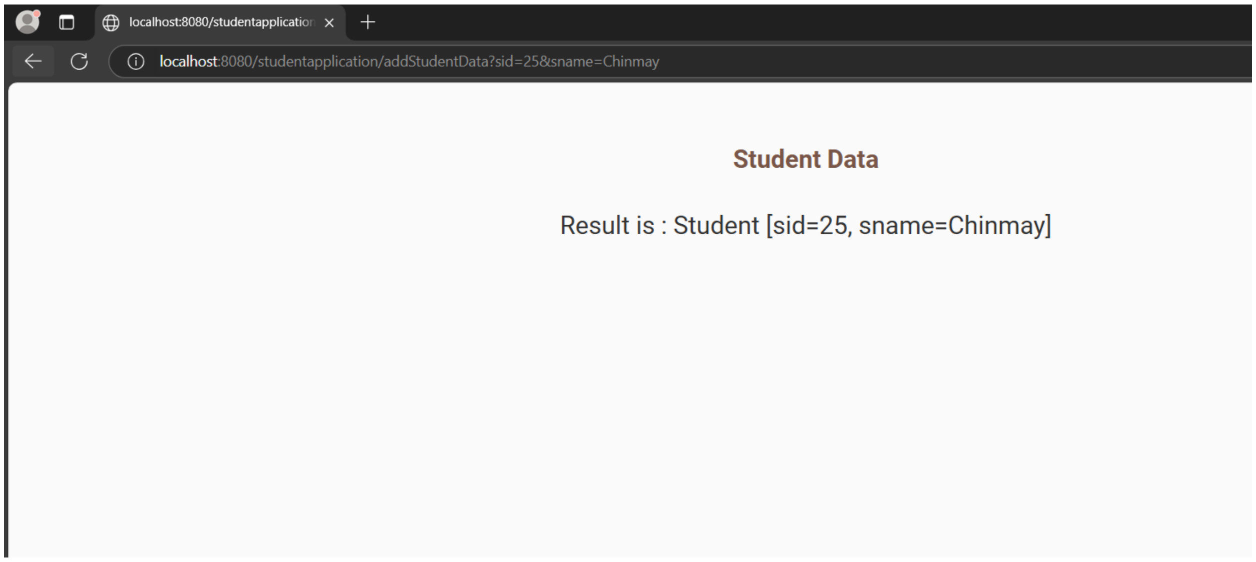
Consider a scenario where we are trying to take the Student data(Object) and storing in the database.

We are trying to store this Object/Entity in the database .

If we are taking at least 10 inputs from the user in that case we need to 10 RequestParam in the Controller in-order to accept the input avoid that we can go with Model Attribute.

In our Scenario we have taken sid and sname as input and same is displayed on the result page.

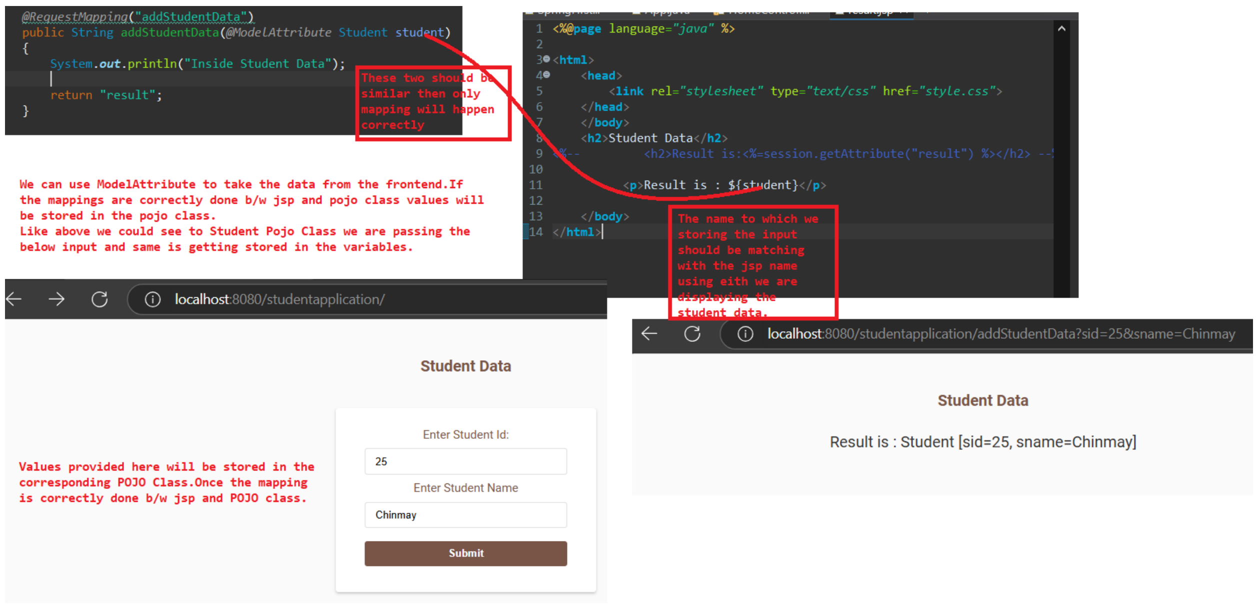

localhost:8080/studentapplication
localhost:8080/studentapplication/
Student Data
Enter Student Id:
25
Enter Student Name
Chinmay
Submit


localhost:8080/studentapplication
localhost:8080/studentapplication/addStudentData?sid=25&sname=Chinmay
Student Data

Result is : Student [sid=25, sname=Chinmay]

https://github.com/ChinmaySai/Spring/commit/1ef3a5e9a04d230ad9d70c18e82b0b116148da18

213 : Using Model Attribute



```java
@RequestMapping("addStudentData")
public String addStudentData(@ModelAttribute Student student)
{
    System.out.println("Inside Student Data");
    |
    return "result";
}
```

These two should be similar then only mapping will happen correctly

We can use ModelAttribute to take the data from the frontend.If the mappings are correctly done b/w jsp and pojo class values will be stored in the pojo class.
Like above we could see to Student Pojo Class we are passing the below input and same is getting stored in the variables.

```jsp
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    </body>
    <h2>Student Data</h2>
<%--        <h2>Result is:<%=session.getAttribute("result") %></h2> --%>

          <p>Result is : ${student}</p>

    </body>
</html>
```

The name to which we storing the input should be matching with the jsp name using eith we are displaying the student data.

localhost:8080/studentapplication/

Student Data

Values provided here will be stored in the corresponding POJO Class.Once the mapping is correctly done b/w jsp and POJO class.

Enter Student Id: 25

Enter Student Name: Chinmay

localhost:8080/studentapplication/addStudentData?sid=25&sname=Chinmay

Student Data

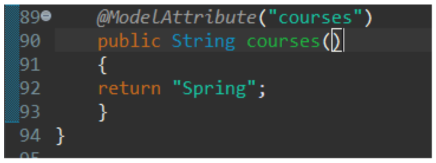
```
Result is : Student [sid=25, sname=Chinmay]
```

https://github.com/ChinmaySai/Spring/commit/8d26c1c0eca32a3d3aea28736876da6c88cf5d37

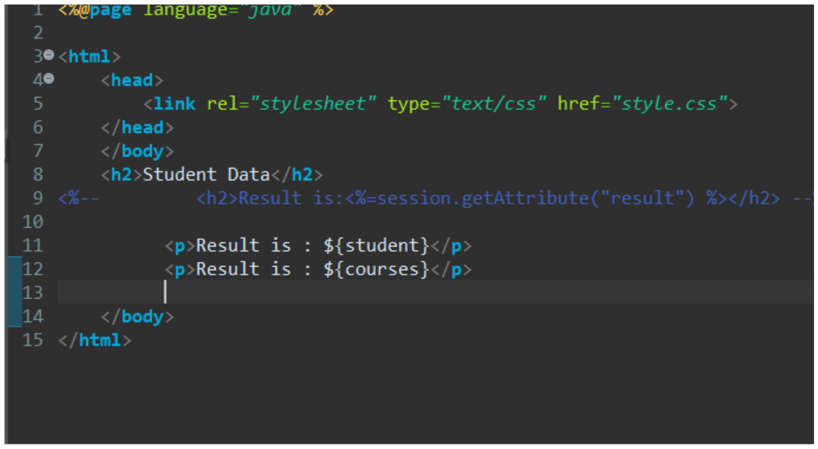
ModelAttribute is Optional

https://github.com/ChinmaySai/Spring/commit/8d26c1c0eca32a3d3aea28736876da6c88cf5d37

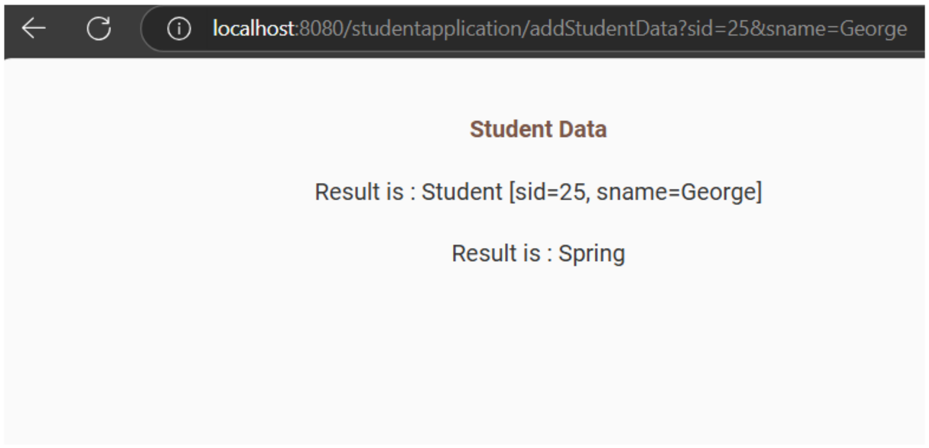
We can pass the values to jsp using model attribute



```
    @ModelAttribute("courses")
    public String courses()
    {
    return "Spring";
    }
}
```



```
<%@page language="java" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    </body>
    <h2>Student Data</h2>
<%--          <h2>Result is:<%=session.getAttribute("result") %></h2> --

          <p>Result is : ${student}</p>
          <p>Result is : ${courses}</p>

    </body>
</html>
```



localhost:8080/studentapplication/addStudentData?sid=25&sname=George

Student Data

Result is : Student [sid=25, sname=George]

Result is : Spring