SPRING MVC

What is Spring MVC?

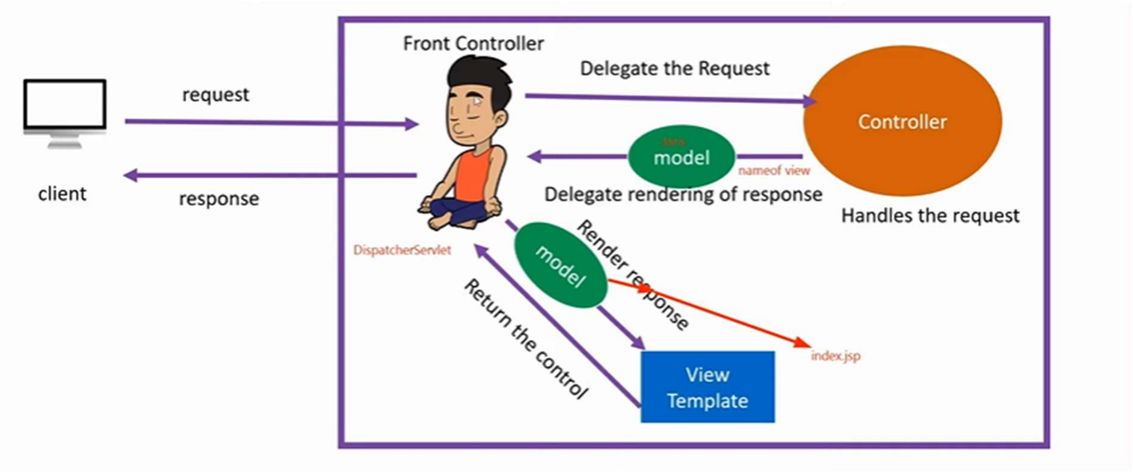
What is MVC Design Pattern?

What is Web application?

Why Spring MVC?

What is the problem without MVC Design Pattern?

Working with Spring MVC?



jj

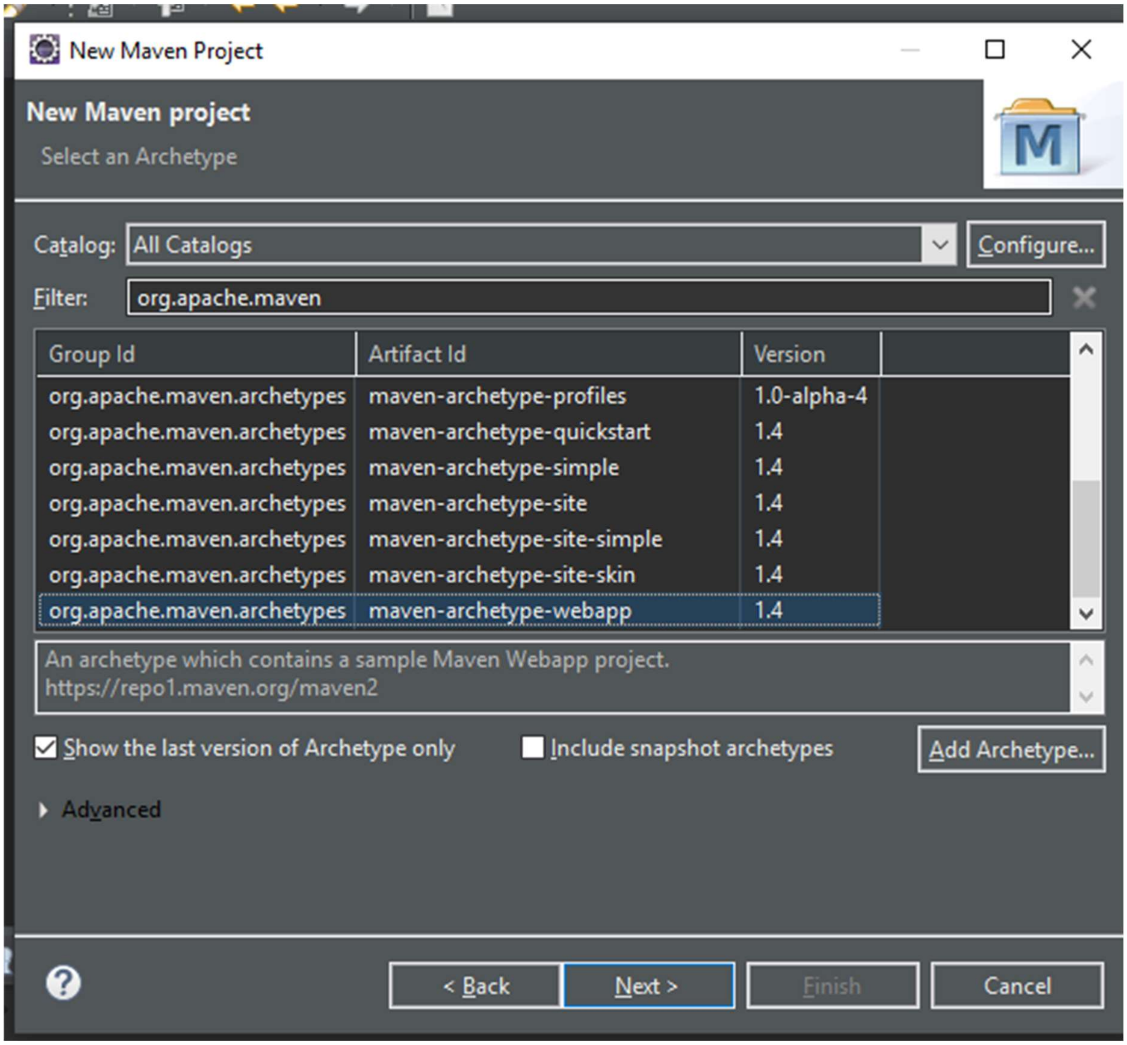
.

@Prepared by ***Shubham Machindra Pardeshi***

SPRING MVC

How to Create MVC Project?

Create new Maven project -> in filter type org.apache.maven - > select archtype – webapp 1.4



New Maven Project

New Maven project

Select an Archetype

Catalog: All Catalogs

Filter: org.apache.maven

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.4
org.apache.maven.archetypes	maven-archetype-simple	1.4
org.apache.maven.archetypes	maven-archetype-site	1.4
org.apache.maven.archetypes	maven-archetype-site-simple	1.4
org.apache.maven.archetypes	maven-archetype-site-skin	1.4
org.apache.maven.archetypes	maven-archetype-webapp	1.4

An archetype which contains a sample Maven Webapp project.
https://repo1.maven.org/maven2

☑ Show the last version of Archetype only

☐ Include snapshot archetypes

Click next-> group id can be your package name - > artefact ID can be your project name

Click on finish.

1. Go to src/main/webapps/WEB-INF
2. Delete the web.xml file
3. Right Click on Project Name
4. Show in – System Explorer
5. Go inside the project -> go inside .setting -> Right click on xml file - > update version as 3.1 from 2.3.
6. Back to eclipse
7. Right click on project -> JAVA EE TOOLS -> UPDATE EAR LIBRARIES
8. AGAIN Right click on project -> JAVA EE TOOLS -> Generate deployment descriptor
9. Now, You will not be seeing src/main/java folders for the reason

@Prepared by ***Shubham Machindra Pardeshi***

10. Right click on project -> build path -> configure build path -> order and export -> select
    a. src/main.java
    b. Src/test/java
    Now click on Apply and close
11. Now go to src down to deployed resources
    a. Src/main/webapp/
        i. You will see index.jsp with error
        ii. To remove that error need to add Servlet JAR
12. Servlet JAR Dependecy

```
<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

<dependency>

   <groupId>javax.servlet</groupId>

   <artifactId>javax.servlet-api</artifactId>

   <version>3.1.0</version>

   <scope>provided</scope>

</dependency>
```

13. Add one more dependency
    a. Spring Web MVC

Spring Web MVC Dependecy

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

   <groupId>org.springframework</groupId>

   <artifactId>spring-webmvc</artifactId>

   <version>5.2.9.RELEASE</version>

</dependency>
```

Note:

If we add Spring Web MVC dependency, it will all other dependency which are needed to create spring core project

a. Spring core
   5.2.9.Relieas
b. Spring context
   5.2.9

14. Web.xml configuration

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

15. We can given any name in <servlet-name> tag but we need to create the file for front controller with the name passed in <servlet-name> along with -servlet.xml

For example if you have passed <Servlet-name> Spring </Servlet-name>

Then we need to create file with name Spring-servlet.xml other wise it will not work

Hibernate EntityManager Relocation » 5.4.10.Final

```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>5.4.10.Final</version>
</dependency>
```

SPRING MVC

Hibernate Core Relocation » 5.4.10.Final

```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
   <groupId>org.hibernate</groupId>
   <artifactId>hibernate-core</artifactId>
   <version>5.4.10.Final</version>
</dependency>
```

MySql Dependency

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>5.1.38</version>
</dependency>
```

Lombok

goto maven repositories

search project lombok

```
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
   <groupId>org.projectlombok</groupId>
   <artifactId>lombok</artifactId>
   <version>1.18.12</version>
   <scope>provided</scope>
</dependency>
```

Note : install the JAR file if not installed already

add jar in POM

----

persistence.xml configuration

create src folder inside project with name

SPRING MVC

src/main/Resource*

under that

create folder with name (META-INF) - > create persistence.xml

and paste below configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
        xmlns="http://xmlns.jcp.org/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
        <persistence-unit name="demo"
                transaction-type="RESOURCE_LOCAL">
                <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
                <properties>
                        <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
                        <property name="javax.persistence.jdbc.url"
                                value="jdbc:mysql://localhost:3306/db-name?createDatabaseIfNotExist=true" />
                        <property name="javax.persistence.jdbc.user" value="root" />
                        <property name="javax.persistence.jdbc.password" value="root" />
                        <property name="hibernate.hbm2ddl.auto" value="update" />
                        <property name="hibernate.show_sql" value="true" />
                        <property name="hibernate.format_sql" value="true" />
                        <property name="hibernate.dialect"
                                value="org.hibernate.dialect.MySQL5Dialect" />
                </properties>
        </persistence-unit>
</persistence>
```

SPRING MVC

Dispatcher-servlet.xml configuration

```
<?xml version="1.0" encoding="UTF-8"?>
                    <!-- NEED TO REMOVE SCHEMA "p". IT IS NOT PRESENT / 404 FOR THIS URL -->

<beans xmlns="http://www.springframework.org/schema/beans"

          xmlns:mvc="http://www.springframework.org/schema/mvc"

          xmlns:context="http://www.springframework.org/schema/context"

          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

          xmlns:p="http://www.springframework.org/schema/p"

          xsi:schemaLocation="http://www.springframework.org/schema/mvc

          http://www.springframework.org/schema/mvc/spring-mvc.xsd

          http://www.springframework.org/schema/beans

          http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

          http://www.springframework.org/schema/context

          http://www.springframework.org/schema/context/spring-context-3.0.xsd">



</beans>
```

Create one base package - > under that create one more package with name .configuration

Base package Com.mvc

Sub package com.mvc.config

Under that create class with any name and annotate that class with annotation @Configuration

If we use annotation @configuration then that will act as XML file

Inside the file create method to return the object for class

InternalResourceViewResolver

```
package com.mvc.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration

public class ConfigurationClass {

        @Bean

        public InternalResourceViewResolver name() {

                InternalResourceViewResolver resolver = new InternalResourceViewResolver();

                resolver.setPrefix("/WEB-INF/views/");

                resolver.setSuffix(".jsp");

                return resolver;

        }

}
```

As we following the MVC pattern, before returning the page to the user we have view resolver who resolves the page and returns the page. So above bean will be used to perform the same task

It will add the prefix and suffix to the page that being returned

For example

Hello page is being returned

Then above bean will add **/WEB-INF/views/hello.jsp as we have provided the suffix and prefix so it becomes complete path of the file that being returned**

@Configuration

- **Class annotated with @Configuration represents annotation based configuration of the xml based configuration when we want IOC to create object for the bean defined in the xml file that we are writing bean in the java class and annotating class with @Configuration so that IOC will look for bean in the respective class and create object for us**
- **Class annotated with @Configuration will act like IOC for us.**

@Bean

- **This annotation is used with the method which returns the object for particular class.**
- **When we annotate any method with @Bean then IOC will be able to provide Object for the particular class implicitly.**

Now, under the base package create another package for controller layer

Com.mvc.controller

```
package com.mvc.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

@Controller

public class ControllerClass {

        @RequestMapping("/Home")

        public String getHomepage() {

                return "Homepage";

        }

}
```

@Controller

- It is stereotype annotation.
- If we make use of @Controller annotation then it represents that particular class as controller class. If the class is annotated with @controller then dispatcher scans for such classes for mapped method and detects @RequestMapping, @GetMapping, @PostMapping annotations

@RequestMapping(“/requestedpagename”)

This annotation is used to map the url request with the method with annotated with @RequestMapping which responsible to handle the request for the requested url

Example given below.

Sending data from controller to view(jsp page)

We have two ways

a. **Model <Interface>**
   i. **Model is an interface which is used to send data from controller to a JSP page.**

   Methods present
   addAttribute(Object object)

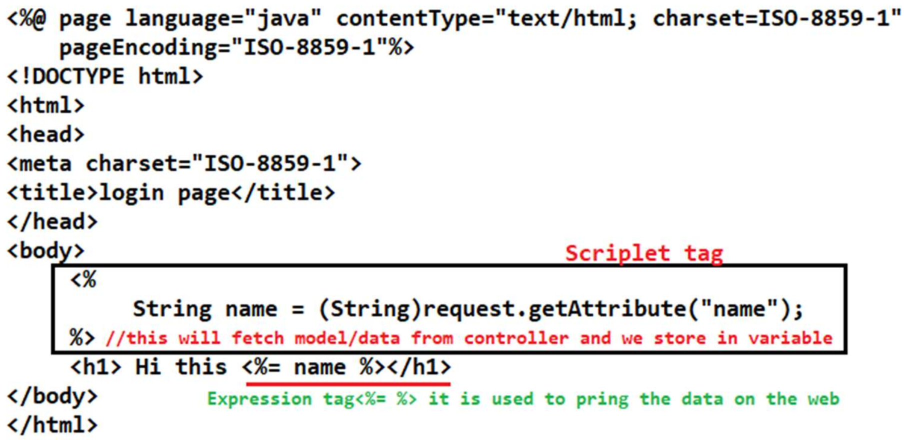
addAttribute(“key”, “value”)
it has many more method above two are widely used methods

Create Controller method which will transfer the data to View “login” which is nothing but JSP page

```
@Controller

public class ControllerClass {

        @RequestMapping("/login")

        public String getLogin(Model model) {

                model.addAttribute("name", "shubham");

                return "login";

        }

}
```

Create JSP page login page with name login.jsp as we are returning login in above method. Under views folder create page.

Login.jsp



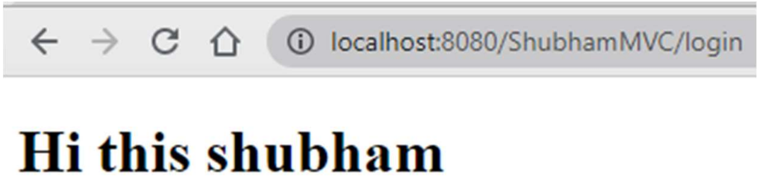
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>login page</title>
</head>
<body>
    <%
        String name = (String)request.getAttribute("name");
    %> //this will fetch model/data from controller and we store in variable
    <h1> Hi this <%= name %></h1>
</body>
</html>
```

Scriplet tag

Expression tag<%= %> it is used to pring the data on the web

To get this on the web browser we need pass the url along with requestmapping path

@RequestMapping("/login") localhost:8080/projectname/login



Hi this shubham

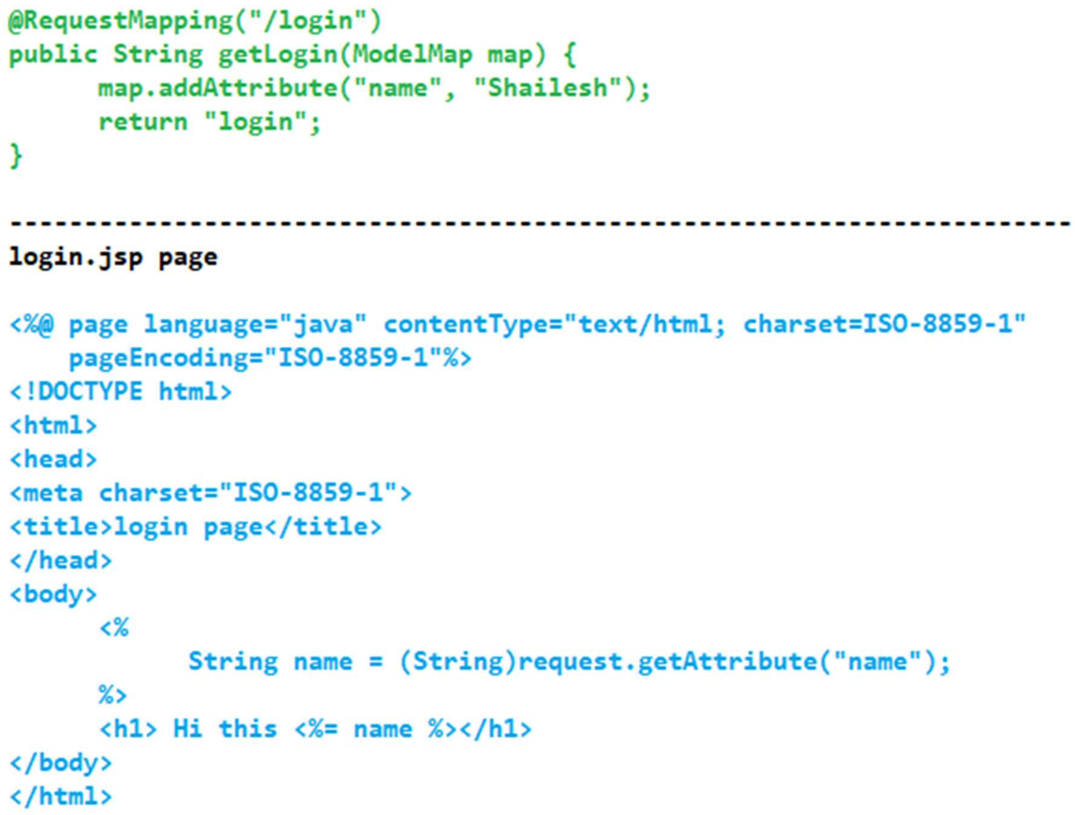
b. ModelMap <Class>

i. ModelMap is a Class which is used to send data from controller to a JSP page.

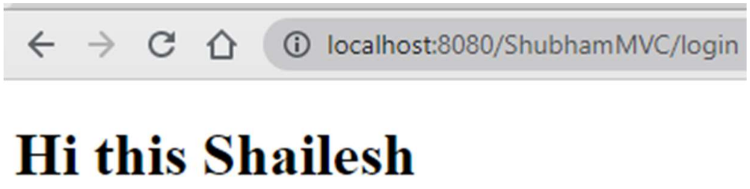
Methods present
addAttribute(Object object)
addAttribute("key", "value")
it has many more method above two are widely used methods

ii. ModelMap is quite same as Model interface.

iii. You can use ModelMap reference to send the data in the above example.



```
@RequestMapping("/login")
public String getLogin(ModelMap map) {
      map.addAttribute("name", "Shailesh");
      return "login";
}

----------------------------------------------------------------------
login.jsp page

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>login page</title>
</head>
<body>
      <%
            String name = (String)request.getAttribute("name");
      %>
      <h1> Hi this <%= name %></h1>
</body>
</html>
```



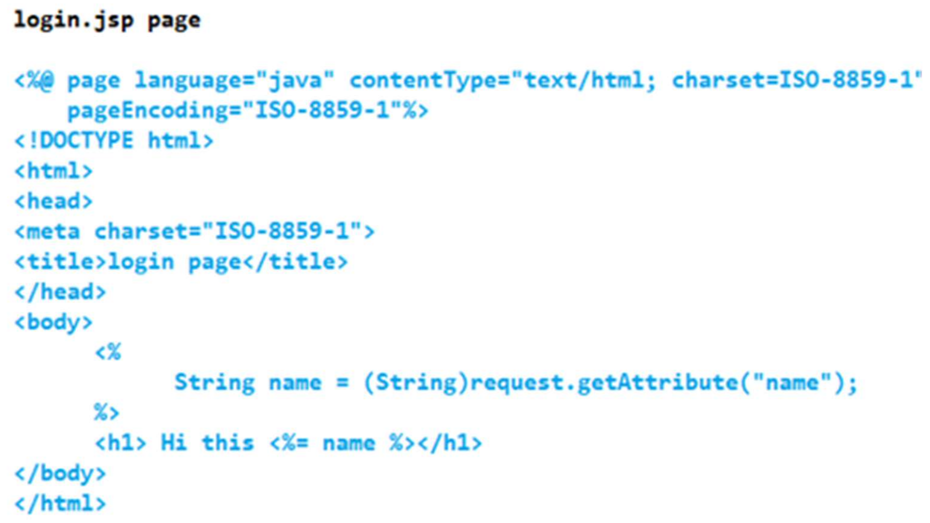
Hi this Shailesh

c. ModelAndView

   i. It has method addObject(String key, Object value)

```
@Controller

public class ControllerClass {

        @RequestMapping("/login")

        public ModelAndView getLogin() {

                //Create object for ModelAndView Object

                ModelAndView mav = new ModelAndView();

                mav.addObject("name", "ShubhamPardesie");

                mav.setViewName("login");

                return mav;

        }

}
```

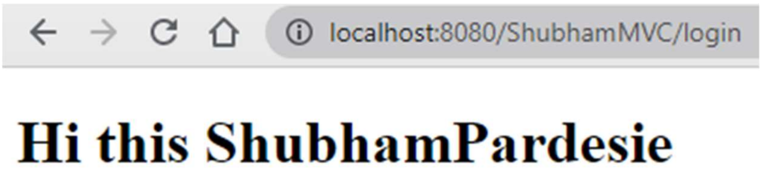
Login.jsp



login.jsp page

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>login page</title>
</head>
<body>
      <%
            String name = (String)request.getAttribute("name");
      %>
      <h1> Hi this <%= name %></h1>
</body>
</html>
```

Output



localhost:8080/ShubhamMVC/login

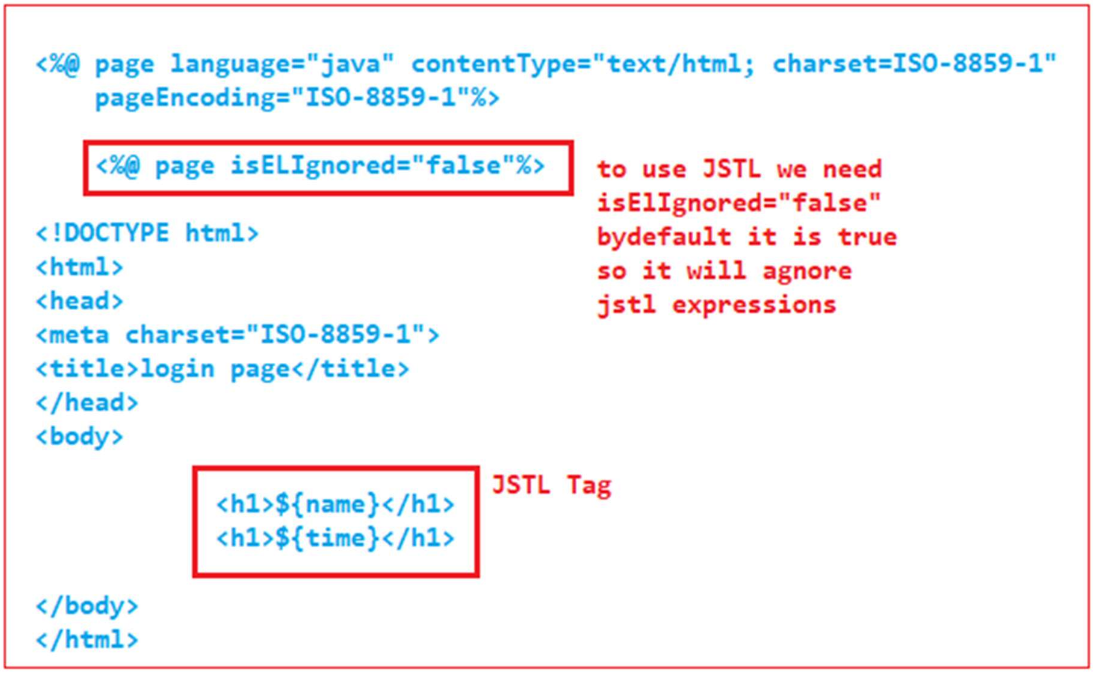
Hi this ShubhamPardesie

When make use of ModelAndView to transfer data to jsp page we set the data along with the view (jsp page) where data needs to be displayed

When we used Model and ModelMap **we were returnining View name.**

But here are returning object for ModelAndView

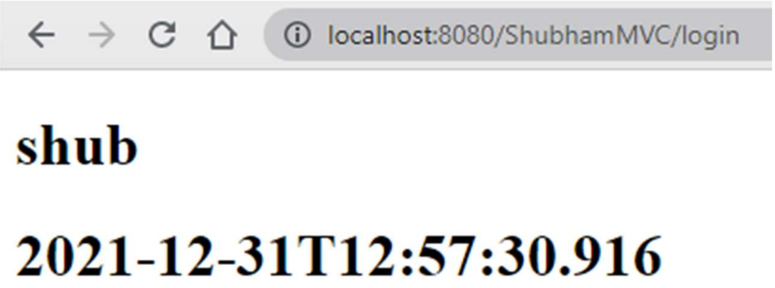
1. JSP Expression Language to print values



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

    <%@ page isELIgnored="false"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>login page</title>
</head>
<body>

            <h1>${name}</h1>
            <h1>${time}</h1>

</body>
</html>
```

to use JSTL we need isElIgnored="false" bydefault it is true so it will agnore jstl expressions

JSTL Tag



```
shub

2021-12-31T12:57:30.916
```

SPRING MVC

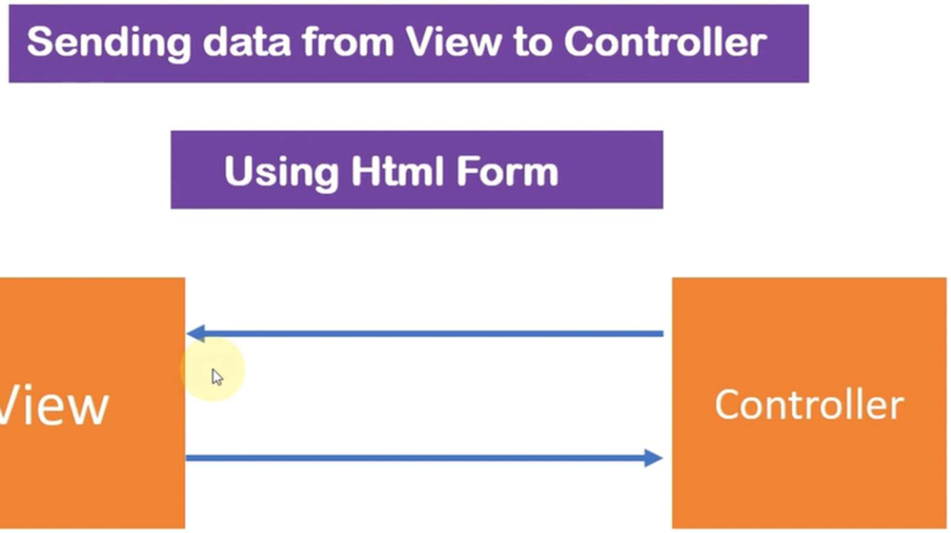
JSTL Dependency.

```
<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>jstl</artifactId>
   <version>1.2</version>
</dependency>
```

Add taglib in jsp page to use JSTL

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
```

❖ Sending Data from View to Controller



Sending data from View to Controller

Using Html Form

1. Using HTML form using submit button

How to get data from JSP page to Controller

2. @RequestParam
3. @ModelAttribute

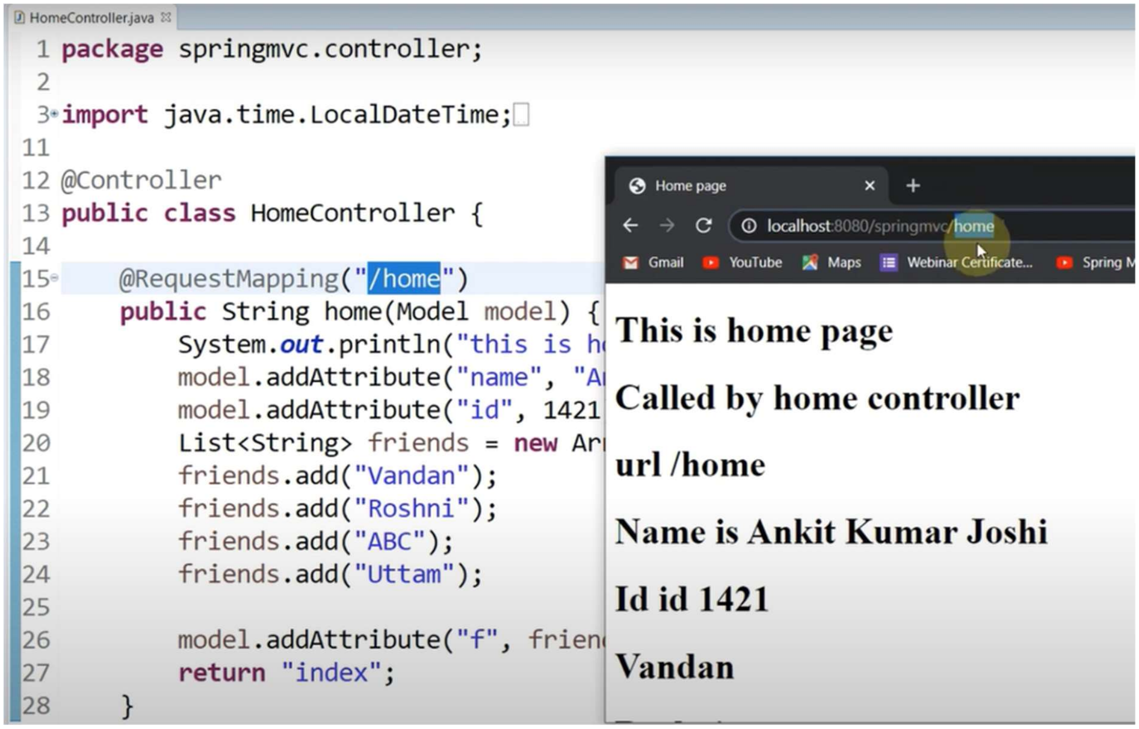
4. HttpServletRequest request

@RequestMapping("/url")

```
@RequestMapping(path = "/login", method = RequestMethod.POST)
        public void name(String name) {
                System.out.println(name);
        }
```

This Annotation can be used to Class as well as Method

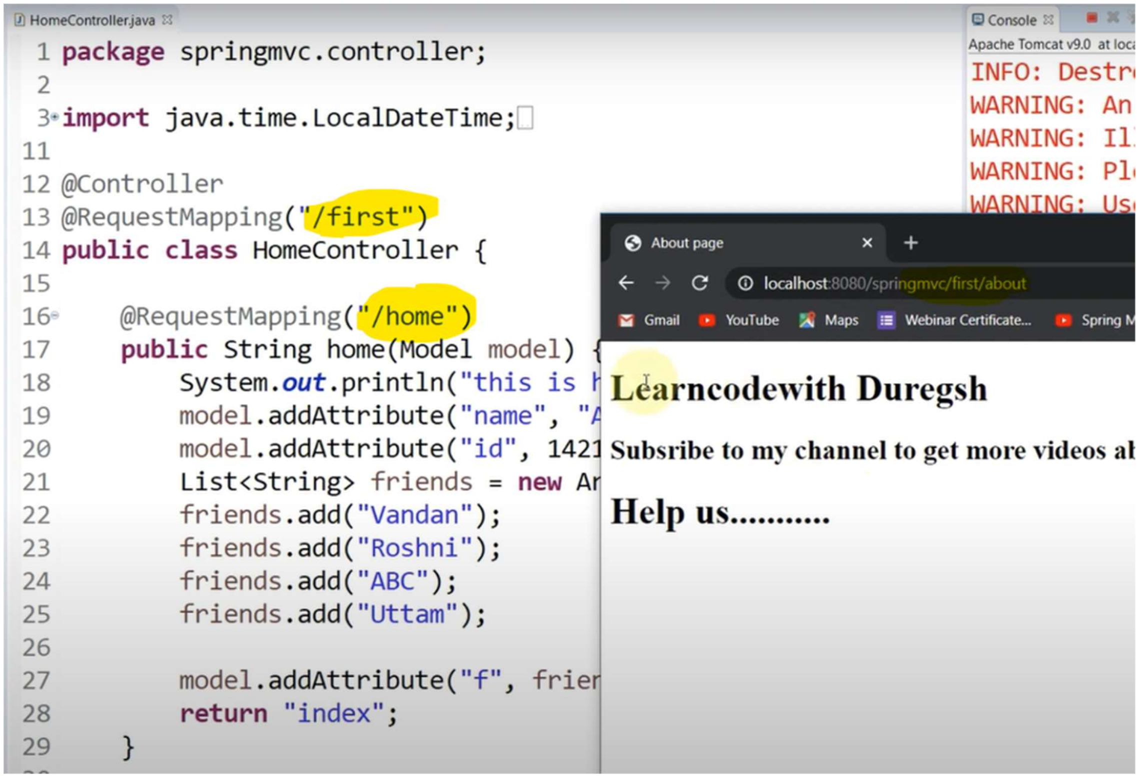
It is used to map the request URL such as /home on to and particular handler method



```
package springmvc.controller;

import java.time.LocalDateTime;

@Controller
public class HomeController {

    @RequestMapping("/home")
    public String home(Model model) {
        System.out.println("this is h
        model.addAttribute("name", "A
        model.addAttribute("id", 1421
        List<String> friends = new Ar
        friends.add("Vandan");
        friends.add("Roshni");
        friends.add("ABC");
        friends.add("Uttam");

        model.addAttribute("f", frien
        return "index";
    }
```

```
This is home page

Called by home controller

url /home

Name is Ankit Kumar Joshi

Id id 1421

Vandan
```

How to use @RequestMapping("/") on class?



```
package springmvc.controller;

import java.time.LocalDateTime;

@Controller
@RequestMapping("/first")
public class HomeController {

    @RequestMapping("/home")
    public String home(Model model) {
        System.out.println("this is h
        model.addAttribute("name", "A
        model.addAttribute("id", 1421
        List<String> friends = new Ar
        friends.add("Vandan");
        friends.add("Roshni");
        friends.add("ABC");
        friends.add("Uttam");

        model.addAttribute("f", frier
        return "index";
    }
```

```
INFO: Destr
WARNING: An
WARNING: Il
WARNING: Pl
WARNING: Us
```

localhost:8080/springmvc/first/about

Learncodewith Duregsh

Subsribe to my channel to get more videos al

Help us..........

❖ Creating HTML form to Take input from user

1. Using HTML Form

To fetch the data from View to controller we can use

First Way

HttpServletRequest request object

This request object will hold the complete HTML form data

That we can get with the help of **request.getAttribute (“fieldname”)**

```
Public String formData(HttpServletRequest request){
request.getParameter(“fieldname”);
return “viewname”;}
```

```
@RequestMapping(path = "/login", method = RequestMethod.POST)
	public void name(@RequestParam("name") String name,
					@RequestParam("password") String
password) {
		System.out.println(name + " " +password);
	}
}
```

@RequestParam(“htmlfieldname”) datatype variable

Is used as replacement for HttpServletRequest request object

If we use above annotation we need not write extra lines of code to fetch the data from view

request.getParameter(“field’)

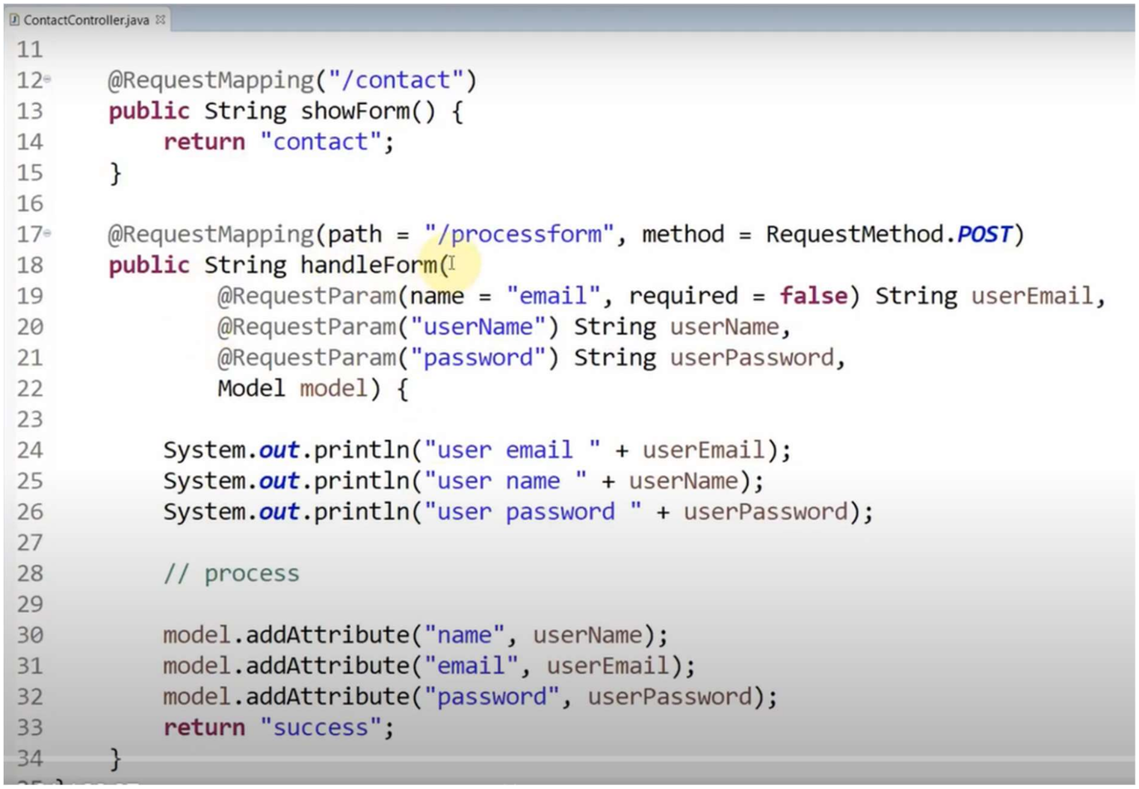
@RequestParam(“htmlfieldname”) datatype variable

This annotation will get us the data from view and will starte in the datatype variable.

Data stored in the variable we make use of it anytime.

Second Way

Working with @RequestParam



ContactController.java

```java
    @RequestMapping("/contact")
    public String showForm() {
        return "contact";
    }

    @RequestMapping(path = "/processform", method = RequestMethod.POST)
    public String handleForm(
            @RequestParam(name = "email", required = false) String userEmail,
            @RequestParam("userName") String userName,
            @RequestParam("password") String userPassword,
            Model model) {

        System.out.println("user email " + userEmail);
        System.out.println("user name " + userName);
        System.out.println("user password " + userPassword);

        // process

        model.addAttribute("name", userName);
        model.addAttribute("email", userEmail);
        model.addAttribute("password", userPassword);
        return "success";
    }
```

Third Way

Fourth Way

(@ModelAttribute Classname **objectreference**)

Create class Student class

```
package com.mvc.bean;

import lombok.Data;

@Data

public class Student {

        private String name;

        private String password;

}
```

JSP Page :

```
<form action="login" method="post">

        <input type="text" name="name">

        <input type="text" name="password">

        <input type="submit" value="Submit">

</form>
```

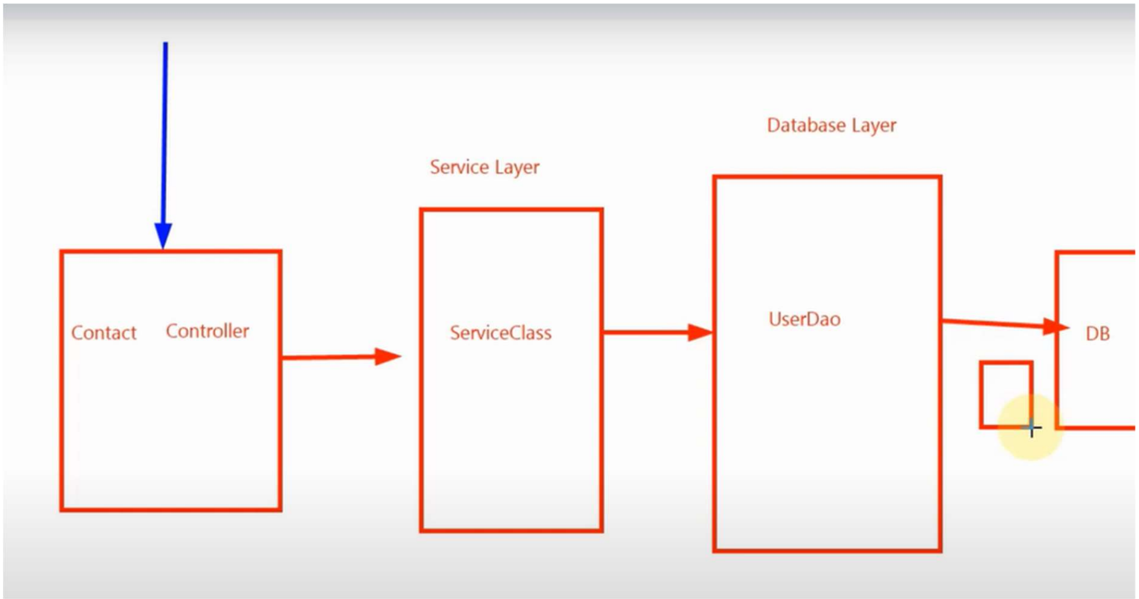
Data members in Student class and Fieldname in View should be matching else ModelAttribute will not work

Fourth Way (@ModelAttribute Student student

```
@RequestMapping(path = "/login", method = RequestMethod.POST)

        public String name(@ModelAttribute Student student) {
```

```
                    System.out.println(student.getName() + " " +
student.getPassword());

return "login";

}
```

Spring ORM



We use interfaces to achieve loose coupling

SPRING MVC

@configuration

@Controller

@Service

@Repository

@ID

@Entity

@Column

@Value

@GeneratedValue(Strategy=GenerationType.Auto)

@Autowired

@Transactional

How to redirect in Spring MVC

1) HttpServletResponse

We can use HttpServletResponse response

Response.sendRedirect(“path”)

But it is not recommended to use it because there would no meaning of using Spring MVC if we are using old Servlet way

2) Redirect prefix

```
@Controller

public class ControllerClass {

        @RequestMapping("/login")

        public String getLogin() {

                        return "redirect:/homepage";

        }

        @RequestMapping(path = "/homepage")
        public String name() {

                return "Homepage";

        }

}
```

3) RedirectView

```
@Controller

public class ControllerClass {

        @RequestMapping("/login")

        public RedirectView getLogin () {

                RedirectView view =new RedirectView ();

                view.setUrl("homepage");

                return view;

        }

        @RequestMapping (path = "/homepage")

        public String name () {

                return "Homepage";

        }

}
```

SPRING MVC

@SessionAttrinute(name=”” required=false) non-premitiveDatatype variablename)

```
@PostMapping("/welcome")
public String getWelcome(@ModelAttribute AdminLogin login,ModelMap map, HttpServletRequest request) {
        HttpSession session = request.getSession();
        if(service.validate(login)) {
            map.addAttribute("msg", "logged in Successfully");
            map.addAttribute("login", login);
            session.setAttribute("loggedin", login);
            return "welcome";
        }else {
            map.addAttribute("errmsg", "Something went wrong");
            return "Home";
        }
    }
```

```
    @PostMapping("/delete")
    public String deleteEmp(ModelMap map, @ModelAttribute EmpDB db, @SessionAttribute(name = "loggedin", required = true) AdminLogin login) {
        if(login!=null) {
            if(service.deleteEmp(db)) {
                map.addAttribute("msg", "record has been deleted successfully");


            }else {
                map.addAttribute("msg", "Could not find employee");
            }
            return "delete";
        }else {
            map.addAttribute("msg", "Please Login First");
            return "Home";
        }
    }
```

```
@GetMapping("/logout")

public String getLogout(ModelMap map, HttpSession session) {

                                    session.invalidate();

                                    map.addAttribute("msg", "Logged out successfully");

                                    return "Home";

            }
```