**Spring MVC 2nd Lesson.**

* If you would like to see first lesson from live classes. It will be named mvc-app in the same directory.
* In this lesson first start by creating a dynamic web project called mvc-app-demo. Here use module version 5. Generate web.xml file. Next convert this to a maven project.
* Now from previous project which **was mvc-app**. Copy the folder of views and paste it into this current projects **WEB-INF folder**.
* Next copy your spring web mvc xml file from old project and paste it into your current projects WEB-INF folder.
* Then copy index.jsp from older project and paste it into your current projects webapp folder.
* Next copy your spring mvc controller package from old project and paste it into your src/main/java folder which will contain your Front Controller that is called HomeController from old project.
* And now in in **pom.xml file** lets import from dependencies. We are going to import two dependencies that are apart of Spring Framework. That is Spring-web and Spring-webmvc
* Pom.xml file looks like this:

A screenshot of a computer

Description automatically generated

Now in your old project. Just copy the content from web.xml and paste it into your current web.xml file. And remove the

<display-name>Index Page</display-name>

<welcome-file-list>

<welcome-file>/WEB-INF/index.jsp</welcome-file>

</welcome-file-list>

* You did this last time to map your index.jsp and it worked. Now you do not need it so paste old information and just remove that.
* By default your web.xml file looks like this:

A screenshot of a computer

Description automatically generated

* Now paste in your old information from last project and it will look like this:
* Load on startup element means loading the servlet at the time of deployment if value is positive.

A screenshot of a computer

Description automatically generated

* And as of now HomeController looks like this:

A screenshot of a computer program

Description automatically generated

* My spring web mvc xml looks like this:

A screenshot of a computer

Description automatically generated

* Index.jsp looks like this:

A screen shot of a computer

Description automatically generated

* Welcome.jsp looks like this:

A screenshot of a computer

Description automatically generated

* Home.jsp looks like this:

A screenshot of a computer program

Description automatically generated

* As of now if you run this application, in the console every function in your homecontroller will initiate a statement letting you know its being executed once you open the page url. Once you run, index.jsp will be first. You can enter home after the / in the URL and it will take you to home controller where your homePage function will be fired in the console because you triggered that response.
* You can click the link in the home page and it will take you to the welcome page where the welcome servlet will be triggered as well due to the response.
* Once you run application it will look like this:

A screenshot of a computer

Description automatically generated

* Enter home after the forward slash and it will look like this:

A screenshot of a computer

Description automatically generated

* Then click the link and it will look like this:

A screenshot of a computer

Description automatically generated

* Now we are currently up to date with everything we did in the last project now. We changed projects because we need to do dynamic web project and then convert it to maven. Rather then doing maven and trying to implement properties from dynamic web. This is the proper way and will avoid some errors and difficulties in the project.
* Our instructor was facing errors in last project but since we did it right there are no errors in our previous project either.
* Now when we are using spring before version 6, you would use value instead of path in HomeController and would map by using a / and the servlet mapping in web.xml could be a / you wouldn’t need to call the path directly. Now after version 6, you need to call the proper name of the servlet like we did in HomeController using path name and in web.xml just like we did.
* **Recap**: In the web.xml we have a DispatcherServlet which is acting as our Front Controller. With this Front Controller servlet. We have specified few information. First one is the contextConfigLocation. That our configuration details related to Spring Web MVC presented in /WEB-INF location under my-spring-web-mvc.xml file.
* So we are trying to provide the location details of the configuration file to the FrontController. And below in web.xml file we have defined two servlet mappings for home and welcome front which is part of Front Controller servlet. We did that because **the Dispatcher/Front Controller servlet** is the first one. It is the entry point of your incoming requests.
* So whenever the request reaches your web application. Front Controller is intercepting that request, and looks for the **URL pattern/path** to find out the right controller for your request handling.
* In this case when the request comes for /home. It tries to identify that where this /home path is given. This /home is given as a request mapping inside HomeController class. So the FrontController servlet identifies HomeController as the right controller to handle the request. And the homePage method is going to be the actual method that will handle your request.
* In **this homePage method** we are just returning home as a string. The moment the FrontController servlet receives home from HomeController. Then it tries to look for the InternalResourceViewResolver which is located in the spring web mvc xml file an which it tries to identify the jsp page for home. Then it returns the result as home.jsp which we are able to see in the browser.
* After that **in the home.jsp page** we have a link. Where we are just saying welcome in the href, nothing else. So whenever we click the link, welcome will be appended in the path “/welcome”. So what happens with this request, it is intercepted again by Front Controller servlet, and it looks for the URL pattern which is /welcome. This is in the web.xml file servlet mapping.
* Since it is correctly matching in the servlet mapping, it checks to see where this path is present. It is able to find out that this path /welcome is available in the HomeController, this time with welcomePage function. So it tries to forward your request /welcome to this particular welcome page. And that’s why you will see serving the request at welcome in the console.
* After that **in HomeController**, you have created a new date, changed the format of the date, set the new date in a attribute that is of model. The name of the attribute is welcomeDateTimeStamp. So apart of returning a welcome string, you are also returning some data by setting that data in the model.
* So now what will happen, InternalResourceViewResolver can help Front Controller to identify the welcome.jsp page. After it identifies the welcome.jsp page it tries to resolve line #12 which is the ${welcomeDateTimeStamp} after resolving this it tries to return the complete result and you are able to see the date time stamp on the welcome.jsp page when running it on the server.