LAB- Spring Mvc - Using Spring Boot

In this lab you will be understanding how to create a spring mvc application using Spring Boot

STEP -1

In this step, you will be working on **06mvc-usingboot-start** project

1) The code provided to you is same as the one in **05mvc-javaconfiguration-solution** project .

Deploy the Application and make sure it is working normally,

2) Open MvcConfig.java . Observe that this is extending MvcConfigurerAdapter and overriding some methods. Comment out everything inside the class. It should look empty like below :

```
@EnableWebMvc
@ComponentScan(basePackages={"com.way2learnonline.controllers"})
@Configuration
public class MvcConfig {

}
```

3) Since we want to use Spring Boot, add `@EnableAutoConfiguration` on top of the class.
You should not use @EnableWebMvc when you are enabling autoConfiguration .
Why ? Can you come up with reason ?

Press Ctrl+Shift+T and Open `WebMvcAutoConfiguration` and observe the code. It is annotated with `@ConditionalOnMissingBean(WebMvcConfigurationSupport.class)` .

So, This `WebMvcAutoConfiguration` will be used only when there is no Bean of type WebMvcConfigurationSupport is present.

Press Ctrl+Shift+T and Open EnableWebMvc . Observe that it is importing DelegatingWebMvcConfiguration . Open DelegatingWebMvcConfiguration and Observe that it is extending WebMvcConfigurationSupport .

So, If you are using @EnableWebMvc , there will be a bean of type WebMvcConfigurationSupport .

So, WebMvcAutoConfiguration which contains all the SpringBoot Configuration for MVC Will not be loaded.

Still Didn't understand ?(See the video once again or ask me if i am present)

Comment out @EnableWebMvc .

So, Remember that When you are using Spring Boot for MVC application, donot use @EnableWebMvc

4) Open RootConfig.java and observe it. comment out the DataSource Bean definition.

Open application.properties and observe how dataSource properties are configured .

Who is actually loading the datasource properties prefixed with "spring.datasource" from application.properties and creating DataSource ?

Press ctrl+Shift+t and Open DataSourceAutoConfiguration. Observe how DataSource is created.

4) We cannot avoid configuring SessionFactory Bean because Spring Boot has support auto configuration of JPA using Hibernate .

. Open `HibernateJpaAutoConfiguration and Observe how EntityManagerFactory is defined automatically.`

But doesnot have autoconfiguration for Hibernate

5) Since we are using Spring Boot, observe that MvcConfig.java is empty and RootConfig.java is containing only hibernate related beans which are not supported by spring boot.

Now You dont need `WebInitializer.java .` Delete it

When using Spring Boot for WebApplications to be deployed in a web container, you have to write a class extending `SpringBootServletInitializer` .

Fortunately, i have already given a class Way2LearnApplication.java which is already extending `SpringBootServletInitializer` and overriding configure() method to tell about the configuration files.

Just uncomment the code in that file.

Now Deploy the application and check if the views are rendered properly.

You will observe that nothing is displayed on the browser when request is given for home.htm

This is because we didn't configure ViewResolver Bean.

Open MvcConfig.java and check if we have configured ViewResolver. We didn't.

If you want InternalResourceViewResolver to be automatically registered, you just have to configure following in application.properties :

```
spring.view.prefix:/WEB-INF/view/
spring.view.suffix:.jsp
```

InternalResourceViewResolver will be automatically created using the above prefix and Suffix. Ctrl+Shift+T and open WebMvcAutoConfiguration and observe how InternalResourceViewResolver is defined using the prefix and suffix loaded from application.properties

6) Now Deploy the application and observe that Home page is displayed when request is given to /home.htm . but there is some css problem. Css pages are not loaded.

Observe that all the Css Files are present under /WEB-INF/resources/css folder.

Generally, how to map all static request coming to our web application?

Do you remember we have used following in xml configuration

```
<mvc:resources mapping="/resources/**"
 location="/WEB-INF/resources/"
        cache-period="600" />
```

In Java Configuration, we have used as below :

```
@Override
      public void addResourceHandlers(ResourceHandlerRegistry registry) {
            registry.addResourceHandler("/resources/**")
                  .addResourceLocations("/WEB-INF/resources/");
      }
```

When we are Enabling AutoConfiguration in Spring Boot, a resource handler is already added for /**

Press Ctrl+Shift+t and Open `WebMvcAutoConfiguration` . Observe `addResourceHandlers()` method . /** is mapped to below locations :

```
"classpath:/META-INF/resources/", "classpath:/resources/",
                  "classpath:/static/", "classpath:/public/"
```

So, you have to keep static resources under any of the above locations.

Now move the resources folder from WEB-INF to src/main/resources. so that all the resources will be under classpath:/resources/

Now deploy the application and observe that the Css files are not loaded yet.

Open Home.jsp and observe that we are linking CSS files as shown below :

```
<link rel="stylesheet" type="text/css" href="resources/css/common.css"  >
```

Change it as below as resource urls are mapped to /** instead of /resources/** by default.

```
<link rel="stylesheet" type="text/css" href="css/common.css"  >
```

Make similar changes to all link tags in home.jsp, browseCoursesmain.jsp and viewIndividualCourse.jsp .
Now Deploy the application and make sure that everything is working as expected.

Congratulations !! You know how to create mvc application using Spring Boot