LAB- Spring Mvc - Content Negotiation

In this lab you will be understanding how to configure application such that it supports content negotiation

STEP -1

In this step, you will be working on **01mvc-content-negotiation-start** project

The code Provided to you is same as the code in **07mvc-tiles-solution**

1) we want to render the same content using different views dynamically.

If request comes for /browseCourses.htm we want to render the jsp using tilesView.

If request comes for /browserCourses.xls , we want the same Controller method to be executed which will return same view name. But Now we want to render the content using ExcelView

2) Open CoursesExcelView.java and observe that we are creating excel sheet here with the model data.

Observe that this class extends AbstractExcelView where the content-Type for this view is set as `"application/vnd.ms-excel";`

3) Open mvc-config.xml and observe that we have already declared CoursesExcelView as bean with bean id as "browseCourses.xls"

4) When the view name returned by the controller is " browseCourses.xls " , we want to render this CoursesExcelView.

So, configure BeanNameViewResolver so that it can map a view name to view object using bean name.

Now Open CourseController.java and modify the return value of the method as browseCourses.xls .

Deploy the application and give request to /home.htm.
Now Click on BrowseCourses button and observe that excel sheet will be downloaded.

So, Every time excel sheet only is rendered. Tiles View which renders JSP is not rendered.

Now, Modify browseCourses method of CourseController as shown below :

```
@RequestMapping(value = { "/browseCourses"}, method = RequestMethod.GET)
	public String browseCourses(HttpServletRequest request, Map<String, Object>
modelMap) {

			modelMap.put("courses",courseService.getAll(false));

			if(request.getRequestURI().endsWith(".xls")){
				return "browseCourses.xls";
			}else{
				return "browseCourses";
			}

	}
```

Deploy the application and give request to /home.htm.
Now Click on BrowseCourses button and observe that browseCourses tile view is rendered.

In BrowseCourses page there is a link to "Download as Excel " If you click on it, the request goes to "browseCourses.xls" so the controller method returns "browseCourses.xls" as view name. This time, BeanNameViewResolver will resolve the logical view name to ExcelView.

But we are not happy as we have to write the logic of returning a different view based on extension.

It will be good if there is a view resolver which can resolve the view based on either extension of URL or Accept header in request.

So, **Configure ContentNegotiationView Resolver** in mvc-config.xml

Now make the browseCourses() method in CoursesController to return "browseCourses" .(Just remove if else conditions)

Deploy the application and give request to /home.htm.
Now Click on BrowseCourses button and observe that browseCourses tile view is rendered.

In side browser Courses page, there is a download link. Click on it and observe that excel file is downloaded.

If you configure <mvc :annotation-driven> , a default ContentNegotiationManager is configured.

ContentNegotiationViewResolver internally uses this ContentNegotiationManager.

ContentNegotiation Manager is the one which maps extensions like .xls to mime types. The the ContentNegotiationViewResolver will use this mime type and will try to to find a view resolver which can map the logical view name and mime type to a view.

If you want to customize Content Negotiation Manager, you have to define it . But you should inject this ContentNetotiation Manager into ContentNegotiationViewResolver as well as <mvc:annotation-driven />

WAY2LEARN

Congratulations !! You know how to Configure application such that it supports content negotiation