Advanced Computer Systems Engineering Laboratory ENCS511

Summer Course 2019

T.A: Eng. Sa’el Khlouf ***Spring Summary***

Spring MVC is a framework that is built on top of Java for building Web Services (APIs) in MVC structure, while SpringBoot is introduced to make building Spring MVC projects easier.

Testing your Spring MVC project requires a servlet that runs the local host installed at your PC along with a port that is specified to listen to the HTTP requests. That software in our experiments is the famous Tomcat.

Maven is a project management tool; in which can download external libraries and JAR files to use inside your project. We call these external libraries as a Dependency. The dependency is an external library that is found on the web. That’s why you need internet provided to be able to run Spring projects. the dependencies are stored in file called pom.xml

Spring STS software is a combination of everything you need, to start building Spring MVC projects ! (Maven+SpringBoot+Tomcat). It’s also Portable; meaning that there is no setup at all.

Review the Artifact ID, Group ID and Package ID that we have talked about in class when creating a new spring boot project. For example, Group ID uniquely identifies your project across all projects.

MVC Structure has 4 components:

- View: It’s the front-end; anything that can make an HTTP request to the Web Service API. View can be an HTML page (a web page), or Android, or iOS, or Desktop, etc. To make programming easy; Postman tool is very lightweight and used to make HTTP requests to your Web Services that you are developing! Postman is the replacement of View component !

- Controller: Maps the request sent to the Web Service; to call a specific method in the controller class.

- Service: Services are singleton classes, that can be used to do business logic (simple Java code, like adding an object to array, contacting the database, etc..). they can be used by calling them in the controller methods.

- Model: These are the entity classes that will represent the tables in the database. The variables inside a model class are equivalent to the columns’ names in the tables.

Remember that HTTP requests can be: GET, POST, etc…. (Search for them please and understand them also !).

<u>Review the relational databases (One-to-one, Many-to-one and Man-to-many), and how to construct the tables with the corresponding primary and foreign keys from the internet.</u>

There is a dependency that we have used which is called Hibernate; it does the Object Relational Mapping.

Object Relational Mapping is the process of mapping object oriented logic with relational databases logic.

Object relational mapping allows you to create the tables in the database automatically; by converting the model classes of the MVC structure to a relational database.

Spring Data JPA which is a dependency to support persisting our java object in database so we will have a serialized data. It is used with the object relational mapping.

Annotations:

@RestController: To specify that the class is a Controller.

for the methods inside the controllers' classes:

@RequestMapping(URL): If the method returns a list of objects; then this annotation will make it return JSONs Strings to be returned as a response after initiating a request to this method. The objects will be transformed to the equivalent JSON Strings using a library called JACKSON; this library is a part of Spring framework. This is by default a GET request. The URL is the one that when requested using a browser or Postman, will run the method below this annotation !

@RequestMapping(method=RequestMethod.POST, value="URL"): Here you can change the request type to POST, DELETE, PUT, etc..

For the parameters inside the methods inside the controllers' classes:

@RequestBody User user: This will convert the incoming parameter: JSON string when requesting a URL, and it will convert that JSON String to an object of type User, and the object will be passed to the method.

@PathVariable: This is for passing the incoming parameter from the request URL into a method.

for the service defined inside the controller:

@AutoWired: This is to get an instance of the singleton Service, so we can use the methods inside the service to do some logic. After we are having the instance; we can invoke the methods of the Service inside the Controller class.

@Service: To specify that the class is a Service.