Spring Boot by Usha Shah

1. Create a Spring Boot project
   1. Create a maven project
      1. Eclipse -> new Project -> Maven Project -> create a simple project
      2. Group Id – like a package name – uniquely identify project
      3. Artifact id – like project name –
      4. Version – version of component
      5. Pom.xml – where u can declare dependencies
      6. Till now we have created a simple Maven project, now will make it spring boot project
      7. Add parent tag inside project tag in pom.xml with following value

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.2.RELEASE</version>

</parent>

This parent is not adding any dependency jars. But it specifies the version number for the dependency jars that you will add.

*Note:* ***Bill of Materials***

*the possible list of jars and their versions that works well together is called bill of materials, that is been approved by spring boot.*

* + 1. Above step declares that our project is child of this parent project.
    2. So this parent project is created by spring-boot, and it contains all default maven configurations. So now our project will contain the same configuration.
    3. Spring boot also helps us when we do not list out all the jars that is needed in a web project. We simply need to add parent dependency created by spring-boot.

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

* + 1. Save the pom.xml and maven will create Maven dependencies folder and you can see all the dependencies there
    2. To find out errors in project – markers window – maven problems
    3. To specify a particular version of java

Add

<properties>

<java.version>1.8<j/ava.version>

</properties>

* + 1. To download the updated dependencies -> righ click on project – maven – update project
    2. Create java class with main method, that will be our main entry point, where will bootstrap spring boot application.
    3. Annotate this class with @SpringBootApplication, this will tell the JDK that this is springbootapplication and this class is starting point.
    4. Inside main SpringApplication.run(class having main method.class and args that are arg of main method)
    5. Here , run is static method of SpringApplication class.
    6. This line will do following jobs
       1. Sets up default config
       2. Starts spring application context
       3. Performs class path scan (for annotations)
       4. Starts tomcat server – whether u hv it or not, it comes with spring boot, that’s y spring boot runs as stand alone application – means no servlet container, spring boot is all u need to start the application.

**Embedded Tomcat Server:**

no need to download tomcat server, install or deploy it.

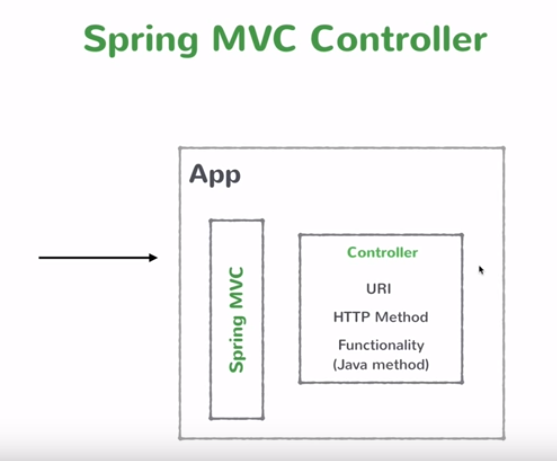
You can have tomcat related configuration in application configuration.

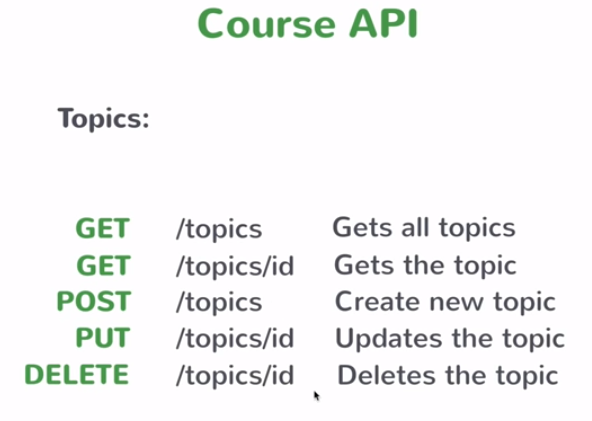
Stand alone application

Microservices architecture – for a bunch of micro services, you do not need to have additional steps to deploy each.

* + 1. This is our very basic spring boot application.
    2. Run as java application
    3. Add a controller – java class having certain annotations, with info like what url to map with it and what it should do
    4. Create java class – HelloController with annotation @RestController
    5. Create a method sayHi with return “Hi” with annotation @requestMapping(“/hello”) by default it will take get method.
    6. You can also return list of a bean class, for example student bean.
    7. Then it automatically returns the list as json array. With key as bean class property and value as the value set in that object.

Spring MVC Controller





Business Service:

A class with @Service annotation

In spring it is single ton.

When the application starts up, Spring creates instance of the service and keeps that in its memory.

Spring creates the object of all class which are declares with @Service annotation. So now whenever you want to use the object of that class simply write @Autowired above the variable.

Ways of creating a Springboot project :

1. Maven project
2. Spring Initializr
3. Spring boot CLI
4. Spring Tool Suit(STS) IDE

Spring Initialzr :

start.spring.io

Application.properties file :

You can set any spring properties in this file. Ex, change port number for spring boot app.

Which properties you can set here – you can find out by “common application properties in spring” on google.

**Spring Data JPA :**

JPA – Java Persistence API

* Allow you to create Object-Relational Mapping(ORM) => Class <-> Table to database.

Apache Derby – allow you to have an embedded (internal) database, that starts when your application starts. Mostly used as a temporary database while development, not used in production.

Dependency for JPA

<artifactId>spring-boot-starter-data-jpa</artifactId>

Dependency for Apache Derby

<groupId>org.apache.derby</groupId>

<artifactId>derby</artifactId>

<scope>runtime</scope>

Now, work with JPA

1. You need to map a class with database table. So in POJO class you need to mention that it is an @Entity and also primary key as @Id. So JPA will create a table mapped with given POJO class.
2. You need to tell JPA, which class to treat as data Service. Now JPA provides functionalities for standard CRUD operation. So you can create as an interface which simply inherits CrudRepository interface of JPA. That contains logic for all CRUD operation.
3. CrudRepository is an generic interface, U need to pass two types, one is POJO class type and second is primary key type.
4. So by this you will have all common CRUD operations available, U only need to create those methods that are special requirement.
5. Now in service class, you can create instance of your repository, with @Autowired.
6. For ex you want list of all Expense from database, u simply can write in service class that ourRepository.findAll() -> that will create connection as well fire query, convert all rows in Expense instance and return it.
7. It returns Iterable so iterator over it and add it in arraylist like ourRepository.findAll().forEach(listname::add)
8. Save method can add a new entry in databse.
9. findOne method – for get data of particular id
10. update – save
11. delete – delete a particular id data
12. to add foreign key you can use annotation like @manyToOne