Install Java latest version

Download Maven

Setup Environment variable:

JAVA\_HOME -> C:\Program Files\Java\jdk-18.0.1.1

MAVEN\_HOME -> C:\Users\cgandhi\Projects\aptos\desktop\apache-maven-3.8.5

Path ->

%JAVA\_HOME%\bin

%MAVEN\_HOME%\bin

Close and start cmd again, run -> mvn --version

Install IntelliJ Idea Community version

Set environment variable

Install MySQL community server and MySQL Workbench

Go to <https://start.spring.io/> and generate spring boot project

Run project from cmd

Go to porm.xml folder location

Run -> mvn spring-boot:run

To add dependency in porm.xml

Go to <https://start.spring.io/>

Click on add dependencies button, search for dependency to add and click on explore

Copy dependency from porm.xml file

[spring boot hotswap with Intellij IDE](https://stackoverflow.com/questions/23155244/spring-boot-hotswap-with-intellij-ide)

Adding devtools to your project

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

</dependency>

Enabling automatic build

Open the Settings --> Build-Execution-Deployment --> Compiler and enable

Build Project Automatically

Update the value of compiler.automake.allow.when.app.running

Press ctrl+shift+A and search for the registry. In the registry, enable

compiler.automake.allow.when.app.running

Open the settings -> Advanced Settings and enable

Allow auto-make to start even if developed application is currently running

Enable MySQL database

Add below dependencies in application.properties

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

Add below properties in application.properties

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/dcbapp

spring.datasource.username=root

spring.datasource.password=admin

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.show-sql:true

Create dcbapp schema in mysql database

Add controller endpoints to Add, Get, Update, Delete

We will create entity, service, and repository

Entity will be annoted with @Entity

Service will be annoted with @Service

Repository will be annoted with @Repository and extends JpaRepository class

Add below in porm.xml

<dependency>

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

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

</dependency>

JpaRepository has methods to do basic save, get methods

We can add new methods in JpaRepository like fetchByDepartmentName, findByDepartmentNameIgnore and even write sql query, need to follow jpql spring boot documentation)

We can add validation to entity like @NotBlank,….

Add below in porm.xml

<dependency>

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

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

</dependency>

We can remove properties getter and setter, Consturctors, tostring methods from entity using Lombok plugin

Add below in porm.xml

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

Install Plugins

File -> Settings -> Plugins -> Search Lombok and install

Now annote entity with @Data, @NoArgsConstructor, @AllArgsConstructor, @builder

Spring uses slf4j for logging

In class

private final Logger LOGGER =

LoggerFactory.getLogger(DepartmentController.class);

In Method

LOGGER.info("Inside fetchDepartmentList of DepartmentController");

Controller handle exception

Create class and extends ResponseEntityExceptionHandler

Annoted class with @ControllerAdvice, @ResponseStatus

Use application.yml instead application.properties

Create a new application.yml file in resource folder

Remove all the properties from application.properties file

Add properties in application.yml file

Use application.yml / application.properties based on the environment

* We can create separate application.yml / application.properties file for each env
* We can add multiple section in application.yml file

server:

port: '8082'

spring:

profiles:

active: dev

---

spring:

profiles: dev

datasource:

driver-class-name: com.mysql.jdbc.Driver

username: root

url: jdbc:mysql://localhost:3306/dcbapp

password: Nothing@123

jpa:

hibernate:

ddl-auto: update

show-sql: 'true'

welcome:

message: Welcome to Daily Code Buffer!!

---

spring:

profiles: qa

datasource:

driver-class-name: com.mysql.jdbc.Driver

username: root

url: jdbc:mysql://localhost:3306/dcbapp-qa

password: Nothing@123

jpa:

hibernate:

ddl-auto: update

show-sql: 'true'

welcome:

message: Welcome to Daily Code Buffer!!

Create .jar file

Open cmd and got to directory where pom.xml

mvn clean install

run jar file

from cmd

java –jar Spring-boot-tutorial-1.0.0.jar –spring.profile.active=dev

Monitor Application

<dependency>

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

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

<scope>test</scope>

</dependency>

Browse - <http://localhost:8082/actuator>

Enable all the actuator endpoints, add below in application.yml file

management:

endpoints:

web:

exposure:

Include: "\*"

Create actuator endpoint

Create a class and add below attributes

@Component

@Endpoint(id="features")

Create a method and add below attribute

@ReadOperation

public Map<String, Feature> features(){

return featureMap;

}

Run an application and actuator endpoint will be browse

"<http://localhost:8082/actuator/features>"

Exclude some actuator endpoints, add exclude in application.yml file

management:

endpoints:

web:

exposure:

include: "\*"

exclude: "env,beans"