Exercise 1:: Deploy a Spring Application(hello world program) on the localhost.

Install Spring Tools for Eclipse

1. Visit spring.io/tools and download Spring Tools for Eclipse.[4.21.0-windows x86-64]

2. click on the downloaded file and Extract the downloaded JAR file.

3. Open Spring Tool Suite (STS) from the extracted folder. (In " springtoolsuite…RELEASE…" – contents .zip - double click on springtoolsuite4.exe)

Create a Spring Boot Project

1. Open Spring Tool Suite.[File-New-Spring Starter Project]

2. Select "Spring Starter Project."

3. Set the following project details:

   - Project Name: demoproject

   - Type: Maven

   - Packaging: JAR

   - Java Version: 17

   - Language: Java

4. Click "Next."

5. Search for "Spring Web" and select it. Click "Finish."

Create a Democontroller

1. Expand the project: `demoproject` -> `src` -> `main` -> `java`.

2. Right-click on `demoproject` -> `New` -> `Class`.

3. Name the class as "Demoontroller" and click "Finish."

4. In `Democontroller.java`, add the following code:

Program

[package com.example.demo;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class demoprojectcontroller {

@RequestMapping("/")

public String hello()

{return "hello javapoint";

} }

(In the java program import the dependencies@RestController and @RequestMappoing)

Right Click on demoproject – Run as - spring boot app

If bottom split shows error related to port number then

Change Port Number

1. Open `src/main/resources/application.properties`.

2. Add the following line to change the port number to 8081:

server.port=8081

Run Again and check the output at localhost:8081/

Exercise 2: Deploy a Spring Application(Hello world program) on AWS-Beanstalk

1. Open springtool|suite4. Right click on the demoproject created in exercise 1-> properties->Maven-> remove pom.xml (let it be blank) -> apply ->(message do you want to update project configuration-yes and close)

2. Right click on the demoproject->run as-> Mavin Build-> No goal-> apply ->Run

3. In case of “Build Failure”, Go to “pom.xml”(left side). Change the following

```
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven....
```

1) **https to http**
2) Modify <build>
   <plugins>
   <plugin>

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
to
<build>
<defaultGoal>install</defaultGoal>
<plugins>
<plugin>
```

4. Right click on the demoproject->run as-> Mavin Build-> No goal-> apply ->Run (repeat of step 2)

5. look into the path of jar file---- in the bottom split. Copy the jar file on the desktop.

6. **Deploy on Elastic Beanstalk:**

    a. Go to the AWS Elastic Beanstalk.

    Console](https://console.aws.amazon.com/elasticbeanstalk/).

    b. Select on application from the sidebar and create application. Enter Name and Description.

    c. Select the application and goto actions to create environment.

    d. Enter Platform as "Java", Next.

    e. Service Access: Create and use new service role. View permission details

    f. Goto IAM Console. Leftside , click on Roles->create Role Choose AWS Service and usecase as "EC2" .

    g. Next Permissions-> Check AWS ElasticBeanstalkworker tier, ElasticBeanstalkweb tier, ElasticBeanstalk Multicontainer Docker Next

    h. Enter Rolename as "Beanstalk Create Role"

    i. Create role.

    j. In the Beanstalk console, EC2 Instance Profile -> refresh -> new role created will appear – choose it- Next

    k. Choose , enable Public IP, instance subnets,Next

    l. Managed platform updates-Deactivate-Next-Next-Submit

    m. Sucessfully created. Goto health and domain. Click on domain.[to see that some default beanstalk page opens)

    n. Select the Beanstalk Environment.Click on upload and deploy (Right side)

    o. Choose your JAR file

    p. Click "Deploy."

    q. Monitor the deployment progress in the Elastic Beanstalk console.

r. Once the deployment is complete, click on domain link and check the output in new screen.

Exercise 3: Post data from remote application to AWS-RDS using POSTMAN

1. Open Spring Tool Suite.[File-New-Spring Starter Project]

2. Select "Spring Starter Project."

3. Set the following project details:

   - Project Name: Ticket

   - Type: Maven

   - Packaging: JAR

   - Java Version: 17

   - Language: Java

4. Click "Next."

5. Search for "Spring Web" and select it.

   Search for "Spring boot dev tools" and select it

   Search for "Spring data jpa" and select it

   Search for "My Sql Driver" and select it Click "Finish."

6. Add .java files

   Right Click on Ticket(project) -> `New` -> `Class`.

   Name the class as "TicketDAO" and click "Finish."

   Right Click on Ticket(project) -> `New` -> `Class`.

   Name the class as "TicketModel" and click "Finish."

   Right Click on Ticket(project) -> `New` -> `Class`.

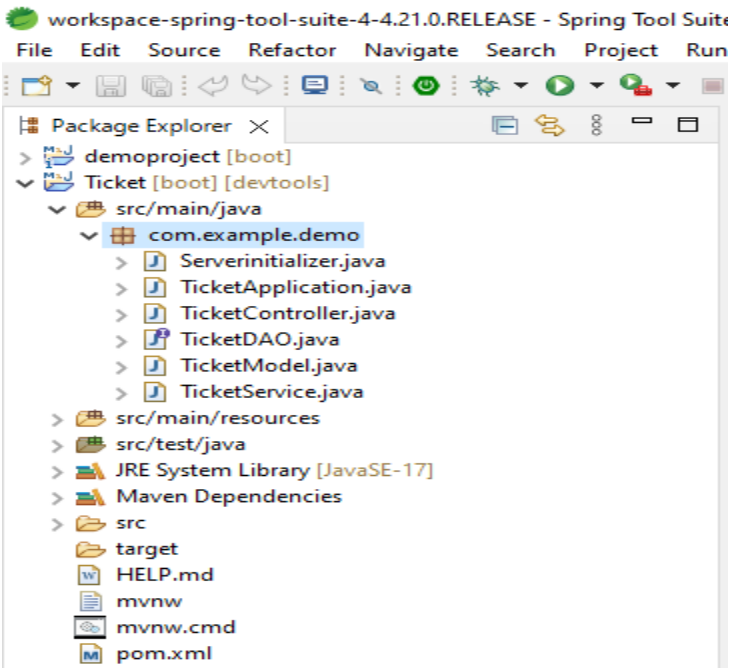
   Name the class as "TicketController" and click "Finish."

   Right Click on Ticket(project) -> `New` -> `Class`.

   Name the class as "TicketService" and click "Finish."

   Right Click on Ticket(project) -> `New` -> `Class`.

   Name the class as "Serviceinitializer" and click "Finish."



7. Left panel, under src/main/resources- expand. Click on application properties.

```
[ spring.jpa.show-sql = True
  spring.jpa.properties.hibernate.format_sql=True
  spring.jpa.hibernate.ddl-auto=create
  spring.datasource.url=jdbc:mysql://endpoint of database:3306/databasename
  spring.datasource.username=root
  spring.datasource.password=<password>  ]
```

8. Click on TicketModel Class-> Write the code

```
[package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class TicketModel {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private Integer id;
	private String Name;
	private String Email; ]
On the next line,
Right click -> source-> generate constructor using fields-> check the attributes->
generate
Right click -> source-> generate getter and setter-> check the attributes-> generate
Right click -> source-> generate constructor from superclass-> check the attributes->
generate
Right click -> source-> generate tostring()-> check the attributes->  generate
```

9. Add code to
TicketDAO.java, TicketController.java, TicketService.java, Serviceinitializer.java, application.java

Click on TicketDAO Class-> Write the code

```
[package com.example.demo;

import org.springframework.data.repository.CrudRepository;

public interface TicketDAO extends CrudRepository<TicketModel,Integer> {

}
```

Click on TicketController Class-> Write the code

```
package com.example.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class TicketController {
@Autowired
private TicketService ticketservice;
@PostMapping(value = "/create")
public TicketModel createticket(@RequestBody TicketModel ticketobj) {
return ticketservice.createticket(ticketobj);
    }
    }    ]
```

Click on TicketService Class-> Write the code

```
[package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class TicketService {

    @Autowired
    private TicketDAO ticketdao;
    public TicketModel createticket(TicketModel ticketobj) {
    return ticketdao.save(ticketobj);

    }

} ]
```

Click on TicketApplication Class-> verify the code

[package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class TicketApplication {
    public static void main(String[] args) {
            SpringApplication.run(TicketApplication.class, args);
    } }]

Click on Serviceinitializer Class-> verify the code

```
[    package com.example.demo;

public class Serverinitializer {

}    ]
```

10. save all the java programs (cntrl +s)

11. goto aws.com, Start the Database.

12. Copy the end point and username and password to src/main/resources expand – application properties

[`spring.datasource.url=jdbc:mysql://endpoint of database:9095/databasename  ]`

13. Right click on project name -> run as -> spring boot app

14. Download and install Postman

15. Left Side: click on “Environments”, From Top click on “+”

16. Click on “POST”, click on “Body”, click on “raw”, click on “json”

17. Type the code { “name”; “MB”,”email”:”anything”}

18. Next to POST, enter localhost:9095/create

19. Can check for output/errors in bottom split.(id is auto generated)

20. open DATABASE(AWS-RDS) through workbench. The row is saved in your ticket table.