	Exp.No: 10 **Date: 06-10-2023**

Aim: CREATE & INSERT DATA IN THE TABLE (SEVER SIDE PROGRAMMING)

Write a java server program to add or insert data into created table.

A common requirement when you are developing and testing your application is to use create and drop your database schema at start up. Let's learn how to do that in a simple Spring Boot application. Configuration steps for automatic Table generation

Here is a sample application.properties configuration that will let Hibernate create the Database tables out of your Entity beans:

spring.datasource.url=jdbc:mysql://localhost:3306/springdemo_db?useSSL=false

spring.datasource.username=root

spring.datasource.password=root

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

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

spring.jpa.generate-ddl=true

spring.jpa.hibernate.ddl-auto = create

Here are the possible values for hibernate.hbm2ddl.auto :

validate: validate the schema, makes no changes to the database.

update: update the schema.

create: creates the schema, destroying previous data

create-drop: drop the schema at the end of the session

Automatic schema creation does not work?

If you are unable to see the database schema generation, check the following points:

Are you including the configuration file in the right place? In a Maven project, you should place your application.properties (or equivalent) in src/main/resources folder.

Your Entity classes should be in the same or in a sub-package relative to where you have you class with @EnableAutoConfiguration. If that's not trye, your Spring Boot application simply cannot detect them and hence will not create your Database tables.

As workaround, you can use @EntityScan annotation in the main class, specifying the Entity you want to save or package too. See the following example:

@SpringBootApplication

@EnableConfigurationProperties

@EntityScan(basePackages = {"com.mlritm.model"}) // force scan JPA entities

```
public class Application {

private static ConfigurableApplicationContext applicationContext;

public static void main(String[] args) {

Application.applicationContext = SpringApplication.run(Application.class, args);

}

}
```

Delegate schema creation to the Database

For some databases, it is possible to delegate the creation of the schema you are using to the

Database itself, via the JDBC Driver.

For example, if you are using MySQL Database, you could use the createDatabaseIfNotExist

property as follows:

spring.datasource.url=jdbc:mysql://localhost:3306/sampledb?createDatabaseIfNotExist=true&useU

nicode=true&characterEncoding=utf-8&autoReconnect=true

Output: