SPRING DATA JPA

Spring JPA is an abstraction on top of hibernate or hibertis, such that you can easily switch from one ORM to ther under the hood.

Hibernate is a JPA specification implementation, it generates the queries

Hibernate internally uses jdbc to communicate with the database

@Table(

name = “tableName”,

uniqueConstraints = @UniqueConstraint(

name = “”

columnName = array of strings

)

@Column() – annotate class properties with this to give them column names and set other properties like nullable

@Id  
@SequenceGenerator(  
 name="student\_sequence",  
 sequenceName = "student\_sequence",  
 allocationSize = 1  
)  
@GeneratedValue(  
 generator = "student\_sequence",  
 strategy = GenerationType.*SEQUENCE*)  
private Long id;

Snippet above says the class property id will be a primary key and would be a sequence to be incremented by 1

@Table(shema = “database\_name”)

So you can only use @GeneratedValue(strategy = GenerationType.IDENTITY/AUTO) to define an autoincrementing column

AUTO lets hibernate use the databses preferred PK generation strategy which is mostly sequence for relational dbs

IDENTITY just uses the databases autoincrement function, this is bad for batch jdbc operations.

Table also creates a table in the database, but it is advised to use the sequence type if your db supports that.

@CreationTimestamp

@UpdateTimestamp

LOMBOK

@Getter

@Setter

Spring Data Commons – Spring Data LDAP, Spring Data MongoDB, Spring Data JPA, Spring Data JDBC.

JPARepository extends PagingAndSortingRepo which extends CrudRepository which extends RepositoryInterface

Public interface ProductRepository extends JPARepository <Product, integer > {

Entity type

Primary key type

}

Entity manager is the interface of Java Persistence Api – The repository methods internally call the entity manager methods which directly talks to the database

Junit Test Cases

@DataJpa test

@SpringBootTest

You will annotate the classes within your test classes with @Test

New BigDecimal(100)

@ToString – a Lombok annotation

Save()

findById()

saveAll() – inserting them one by one

findAll()

deleteById()

delete() – takes a whole entity object – so you have to first find the entity before using the delete method passing it the selected entity

deleteAll() – SELECTS all records from the table , and then run delete statements for all records.

deleteAll(List.of(iterable entities)) – SELECTS all records from the table within the list , and then run delete statements for all the records.

Count()

list.of(product1, product2) – Produces a list of products

Class Annotations for when you are building your CRUD Api

@RestController

@RequestMapping(method=RequestMethod.POST, value = “/subjects/{id}”)

Public List<Subject> updateSubject(@PathVariable Long id, @RequestBody Subject subject)

So essentially to run the application, you first run maven clean install – I guess this creates the compiled target files, then you now run the project as a java application, this will start up tomcat on 8080, as you will see in the console

LEARN JAVA MODEL VALIDATION

JAVA RETURN PARAMETERS

CUSTOMISE JAVA CONTROLLER RETURN OBJECT

\*\* Long.valueOf(String name) – to convert from long to string