Xideral

Java Academy

Week 4-Day 2

REST API with Spring Data JPA

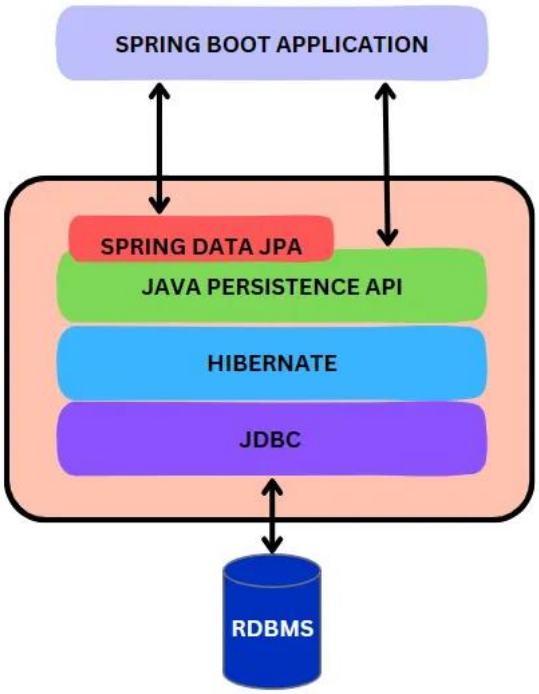
Presented by:

Edgar Itzak Sánchez Rogers

Monterrey Nuevo León México at 28 august 2024

Introduction:

Spring Data JPA, part of the larger Spring Data family, makes it easy to easily implement Java Persistence API repositories. It makes it easier to build Spring-powered applications that use data access technologies. Spring Data JPA aims to significantly improve the implementation of data access layers by reducing the effort to the amount that's actually needed.



JpaRepository:

JpaRepository is a generic interface that allows interact with a database in a abstract and efficient way. By implementing JpaRepository, you can take advantage of a number of features:

- **CRUD operations**: Methods like save(), findById(), findAll(), deleteById(), etc.
- **Custom Queries**: You can define specific queries using the method names method.
- **Pagination and sorting**: Methods to obtain paginated and ordered results.

Java example:

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/nursery
spring.datasource.username=student
spring.datasource.password=student
```

Entity tree

```
package com.edgaritzak.nurserySystemDataJPA.entity;

import jakarta.persistence.Column;


@Data
@NoArgsConstructor @AllArgsConstructor
@Getter @Setter
@Entity
@Table(name="tbl_trees")
public class Tree {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="name")
    private String name;

    @Column(name="description")
    private String description;

    @Column(name="price")
    private float price;

}

```

Tree repository extends JpaRepository

```
package com.edgaritzak.nurserySystemDataJPA.dao;

import java.util.List;

public interface TreeRepository extends JpaRepository<Tree,Integer>{

    // CUSTOM METHOD SELECT LIKE();
     @Query("Select t from Tree t where t.name like :name")
     List<Tree> findNameLike(@Param("name")String name);
}
```

Service interface

```java
package com.edgaritzak.nurserySystemDataJPA.service;

import java.util.List;

public interface TreeService {

    //SELECT
    List<Tree> findNameLike(String text);
    List<Tree> findAll();
    Tree findById(int id);
    //INSERT
    Tree save(Tree tree);
    //UPDATE
    Tree update(Tree tree);
    //DELETE
    Tree deleteById(int id);

}
```

Service Implementation

```java
package com.edgaritzak.nurserySystemDataJPA.service;

import java.util.List;


@Service
public class TreeServiceImpl implements TreeService {

    private TreeRepository treeDAO;

    @Autowired
    public TreeServiceImpl(TreeRepository treeDAO) {
        this.treeDAO = treeDAO;
    }

    //SELECT ALL
        @Override
        public List<Tree> findAll() {
            List<Tree> myList = treeDAO.findAll(); //SPRING DATA JPA METHOD
            return myList;
        }

    //SELECT LIKE
    @Override
    public List<Tree> findNameLike(String text) {
        return treeDAO.findNameLike(text); //SPRING DATA JPA METHOD (CUSTOP QUERY)
    }

```

```
36      //SELECT BY ID
37      @Override
38      public Tree findById(int id) {
39          Tree tree;
40          Optional<Tree> container = (treeDAO.findById(id)); //SPRING DATA JPA METHOD
41
42          if(container.isPresent()) {
43              tree = container.get();
44          } else {
45              throw new RuntimeException("Did not find tree id - " + id);
46          }
47          return tree;
48      }
49
50      //INSERT
51          @Override
52          public Tree save(Tree tree) {
53              treeDAO.save(tree); //SPRING DATA JPA METHOD
54              return tree;
55          }
56
57      //UPDATE
58      @Override
59      public Tree update(Tree tree) {
60          Tree treeidToCompare = treeDAO.findById(tree.getId()) .orElseThrow(() -> new
61          return treeDAO.save(treeidToCompare); //SPRING DATA JPA METHOD
62      }
64      //DELETE
65      @Override
66      public Tree deleteById(int id) {
67          Optional<Tree> tempTree = treeDAO.findById(id); //SPRING DATA JPA METHOD
68          if(tempTree.isPresent()) {
69              treeDAO.deleteById(id); //SPRING DATA JPA METHOD
70              return tempTree.get();
71          } else {
72              throw new IllegalStateException("Error - ID not found");
73          }
74      }
```

REST Controller

```java
package com.edgaritzak.nurserySystemDataJPA.controller;

import java.util.List;

@RestController
@RequestMapping("/api")
public class TreeRestController {

    TreeServiceImpl treeService;

    @Autowired
    public TreeRestController(TreeServiceImpl treeService) {
        super();
        this.treeService = treeService;
    }


    //SELECT
    @GetMapping("/treelist")
    public List<Tree> selectAll(){
        List<Tree> list = treeService.findAll();
        return list;
    }
    //SELECT
    @GetMapping("/treelist/{treename}")
    public List<Tree> selectLike(@PathVariable("treename") String text){
        List<Tree> list = treeService.findNameLike(text);
        return list;
    }
     //SELECT
     @GetMapping("/tree/{id}")
     public Tree selectById(@PathVariable("id") int id) {
         return treeService.findById(id);
     }

     //INSERT
     @PostMapping("/addtree")
     public Tree insert(@RequestBody Tree tree) {
         treeService.save(tree);
         return tree;
     }

     //UPDATE
     @PutMapping("/update")
     public Tree update(@RequestBody Tree tree) {
         Tree updatedTree = treeService.update(tree);
         return updatedTree;
     }

     //DELETE
     @DeleteMapping("/delete/{id}")
     public Tree delete(@PathVariable("id") int id) {
         return treeService.deleteById(id);
     }
}
```

Main class:

```java
package com.edgaritzak.nurserySystemDataJPA;
import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class Main {

    public static void main(String[] args) {
        SpringApplication.run(Main.class, args);
    }
}
```

Explanation:

To implement the query methods simply add

'extends JpaRepsitory<ENTITY,PRIMARY_KEY>' to the interface

```java
public interface TreeRepository extends JpaRepository<Tree,Integer>{
```

JpaRepository methods are now accessible when creating TreeRepository instance, methods such as:

- save()
- SaveAll ()
- FindById ()
- FindAll ()
- FindAllById ()
- deleteById ()

The service interface and the service implementation are now created. The service implementation uses the JPARepsoitory query methods and these services are used by the REST Controller.

The results of using the REST service are shown below.

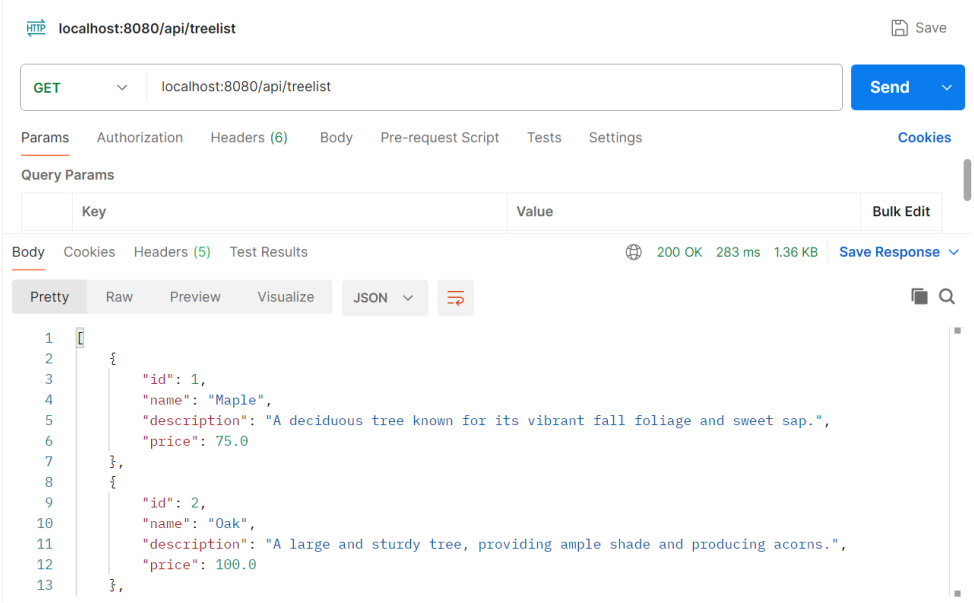
Results:

Table:

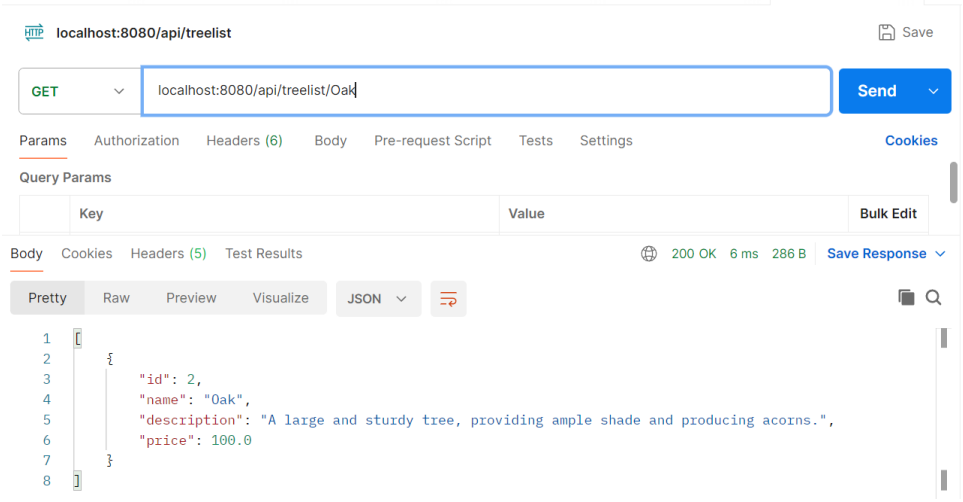
1	Maple	A deciduous tree known for its vibrant fall foliage and sweet sap.	75.00
2	Oak	A large and sturdy tree, providing ample shade and producing acorns.	100.00
3	Pine	An evergreen tree with needle-like leaves and a conical shape.	50.00
4	Cherry Blossom	A beautiful tree known for its stunning pink spring flowers.	120.00
5	Willow	A tree with long, flowing branches and a graceful appearance.	80.00
6	Birch	A tree with distinctive white bark and delicate, small leaves.	65.00
7	Cypress	An evergreen tree often found in wetlands, known for its unique, feathery foliage.	90.00
8	Magnolia	A tree with large, fragrant flowers and shiny, dark green leaves.	110.00
9	Elm	A sturdy tree with a broad canopy and serrated leaves, ideal for urban areas.	85.00
10	Redwood	A majestic and towering evergreen tree, one of the tallest in the world.	150.00

Testing REST service

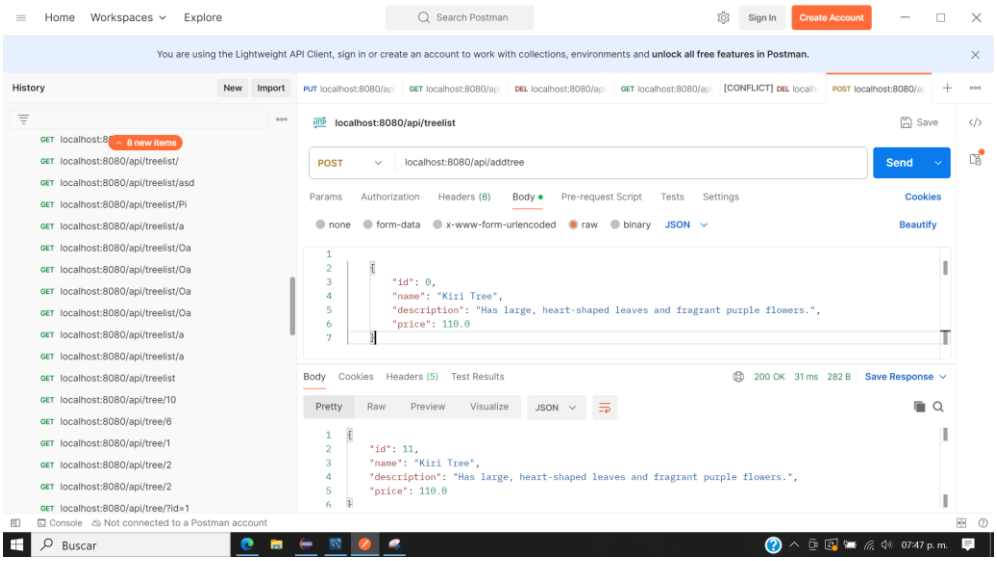
Seach list



Search by name

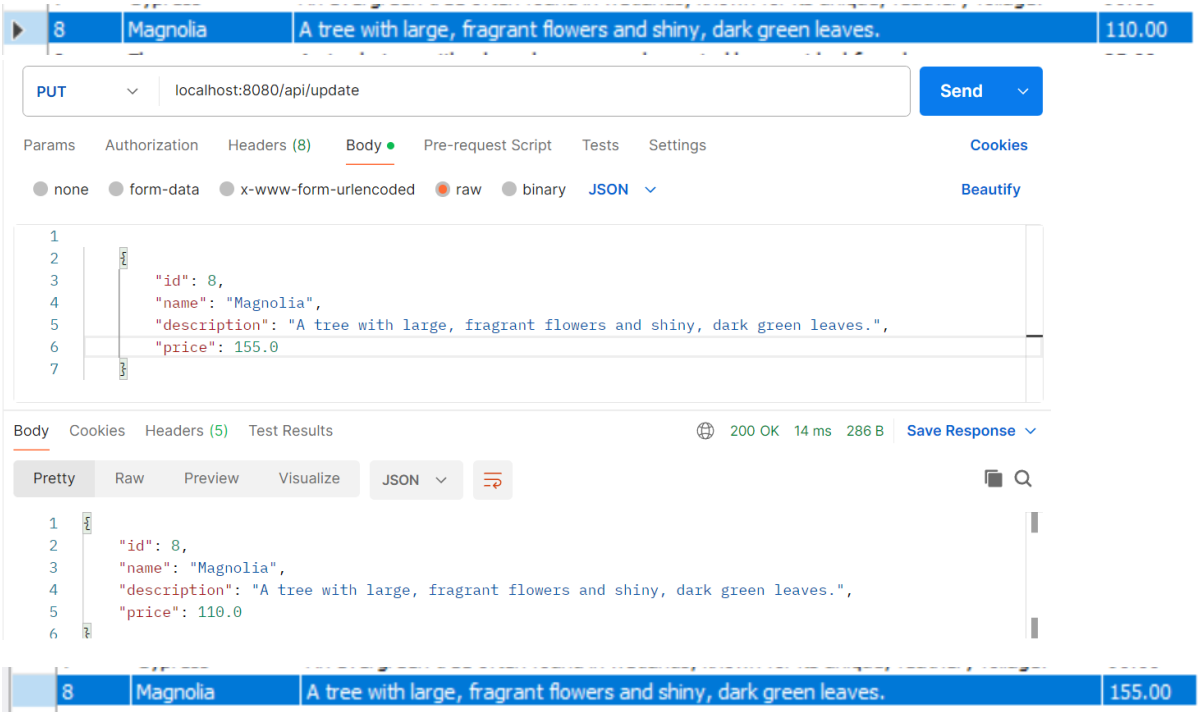


Insert:

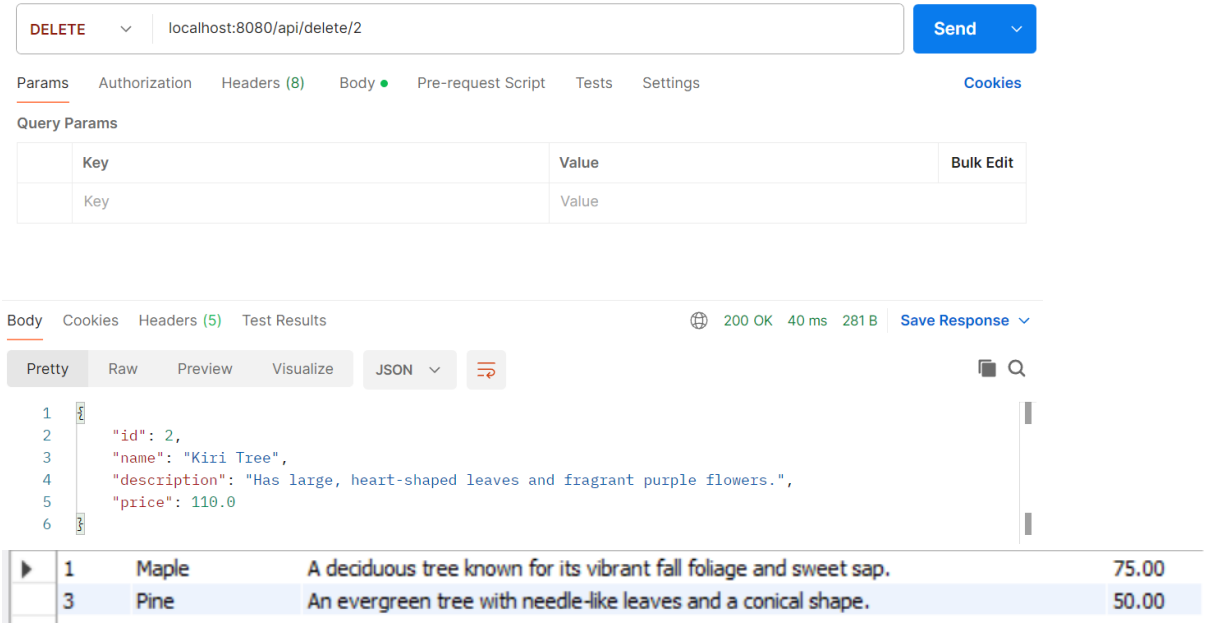


id	name	description	price
1	Maple	A deciduous tree known for its vibrant fall foliage and sweet sap.	75.00
2	Kiri Tree	Has large, heart-shaped leaves and fragrant purple flowers.	110.00
3	Pine	An evergreen tree with needle-like leaves and a conical shape.	50.00
4	Cherry Blossom	A beautiful tree known for its stunning pink spring flowers.	120.00
5	Willow	A tree with long, flowing branches and a graceful appearance.	80.00
6	Birch	A tree with distinctive white bark and delicate, small leaves.	65.00
7	Cypress	An evergreen tree often found in wetlands, known for its unique, feathery foliage.	90.00
8	Magnolia	A tree with large, fragrant flowers and shiny, dark green leaves.	110.00
9	Elm	A sturdy tree with a broad canopy and serrated leaves, ideal for urban areas.	85.00
10	Redwood	A majestic and towering evergreen tree, one of the tallest in the world.	150.00
11	Kiri Tree	Has large, heart-shaped leaves and fragrant purple flowers.	110.00

Update:



Delete:



Conclusion:

Spring Data JPA simplifies application development by providing an abstraction layer on top of JPA (Java Persistence API). Spring Data JPA eliminates the need to write much of the repetitive code associated with data management, thanks to its pre-defined repository interfaces and automatic query generation. It includes integrated functionalities for pagination and sorting of results, enabling efficient management of large volumes of data.

References:

[1] Spring Data JPA

[2] What is Spring Data JPA? - GeeksforGeeks

[3] Getting Started | Accessing Data with JPA (spring.io)