



# SPRING DATA JPA

Daniel Evaristo Escalera Bonilla

## Introduction

Spring data builds on top of the JPA to simplify data persistence and access in Java applications by abstraction of certain processes such as the repository CRUD operations which are generated by creating an object that extends a crud or jpa repository which already has them.

## Implementation

To show the implementation of Spring Data JPA, the following entity class was created using JPA and lombok:

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name="timecard")
public class Timecard {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

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

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

    @Column(name="entryTime")
    private Date entryTime;

    @Column(name="exitTime")
    private Date exitTime;

    @Column(name="lunchTime")
    private Date lunchTime;
}
```

Next a Repository class was created, this class extends the JpaRepository interface, giving it the main CRUD operations:

```
package com.curso.v0.CrudJPA.dao;

import java.util.List;

public interface TimecardRepository extends JpaRepository<Timecard, Long>{

    List<Timecard> findByDepartment(String department);
}
```

A service interface and implementation were created to use the repository operations during execution:

```
1 import java.util.List;
2
3 public interface TimecardService {
4     List<Timecard> findAll();
5     List<Timecard> findByDepartment(String department);
6     Timecard findById(long theId);
7     Timecard save(Timecard theTimecard);
8     void deleteById(long theId);
9 }
```

```

@Service
public class TimecardServiceImpl implements TimecardService{

    private TimecardRepository timecardRepository;

    @Autowired
    public TimecardServiceImpl(TimecardRepository timecardRepository) {
        this.timecardRepository = timecardRepository;
    }

    @Override
    public List<Timecard> findAll() {
        return timecardRepository.findAll();
    }

    @Override
    public Timecard findById(long theId) {
        Optional<Timecard> optionalTimecard = timecardRepository.findById(theId);

        Timecard timecard = null;

        if (optionalTimecard.isPresent()) {
            timecard = optionalTimecard.get();
        }
        else {
            throw new RuntimeException("Did not find employee id - " + theId);
        }

        return timecard;
    }

    @Override
    public List<Timecard> findByDepartment(String department) {
        return timecardRepository.findByDepartment(department);
    }

    @Override
    public Timecard save(Timecard theTimecard) {
        return timecardRepository.save(theTimecard);
    }

    @Override
    public void deleteById(long theId) {
        timecardRepository.deleteById(theId);
    }
}

```

Finally, a rest controller was created to define the different routes of the CRUD API and the body and parameters necessary to apply them:

```

@RestController
@RequestMapping("/timecard-system")
public class TimecardRestController {

    private TimecardService timecardService;

    @Autowired
    public TimecardRestController(TimecardService timecardService) {
        this.timecardService = timecardService;
    }

    @GetMapping("timecards")
    public List<Timecard> getTimecards(){
        return timecardService.findAll();
    }

    @GetMapping("timecards/{id}")
    public Timecard TimecardById(@PathVariable long id){

        // Find the timecard by ID
        Timecard theTimecard = timecardService.findById(id);

        if (theTimecard == null) {
            throw new RuntimeException("Timecard id not found - " + id);
        }

        return theTimecard;
    }
}

```

```

@GetMapping("timecards/department/{department}")
public List<Timecard> getTimecard(@PathVariable String department) {
    return timecardService.findByDepartment(department);
}

@PostMapping("timecards")
public Timecard addTimecard(@RequestBody Timecard theTimecard) {

    theTimecard.setId(0);
    Timecard dbTimecard = timecardService.save(theTimecard);
    return dbTimecard;
}

@PutMapping("timecards")
public Timecard updateTimecard(@RequestBody Timecard theTimecard) {
    Timecard dbTimecard = timecardService.save(theTimecard);
    return dbTimecard;
}

@DeleteMapping("timecards/{id}")
public ResponseEntity<String> delete(@PathVariable Long id) {
    // Find the timecard by ID
    Timecard theTimecard = timecardService.findById(id);

    if (theTimecard == null) {
        throw new RuntimeException("Timecard id not found - " + id);
    }

    timecardService.deleteById(id);

    return ResponseEntity.ok("Timecard deleted successfully, id: " + id);
}

```

## Results

When the application is running we can use the specified URL's to perform operations in the database:

Read  
findAll

GET

http://localhost:8080/timecard-system/timecards

Send

ParamsAuthorizationHeaders (6)BodyScriptsSettings

Cookies

BodyCookiesHeaders (5)Test Results

200 OK13 ms528 B

PrettyRawPreviewVisualizeJSON

```
1 [
2   {
3     "id": 7,
4     "name": "Jane Doe",
5     "department": "IT",
6     "entryTime": "2024-09-05",
7     "exitTime": "2024-09-05",
8     "lunchTime": "2024-09-05"
9   },
10  {
11    "id": 8,
12    "name": "John Doe",
13    "department": "Sales",
14    "entryTime": "2024-09-04",
15    "exitTime": "2024-09-04",
16    "lunchTime": "2024-09-04"
17  },
18 ]
```

getByDepartment

HTTP Crud Tests / Read - getByDepartment

SaveShare

GET

http://localhost:8080/timecard-system/timecards/department/IT

Send

ParamsAuthorizationHeaders (6)BodyScriptsSettings

Cookies

Query Params

	Key	Value	Description	Bulk Edit
--	-----	-------	-------------	-----------

BodyCookiesHeaders (5)Test Results

200 OK1055 ms284 B

PrettyRawPreviewVisualizeJSON

```
1 [
2   {
3     "id": 7,
4     "name": "Jane Doe",
5     "department": "IT",
6     "entryTime": "2024-09-05",
7     "exitTime": "2024-09-05",
8     "lunchTime": "2024-09-05"
9   }
10 ]
```

getById

HTTP

Crud Tests / Read - getById

Save

Share

GET

http://localhost:8080/timecard-system/timecards/8

Send

Params

Authorization

Headers (6)

Body

Scripts

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK

17 ms

285 B

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 8,
3   "name": "John Doe",
4   "department": "Sales",
5   "entryTime": "2024-09-04",
6   "exitTime": "2024-09-04",
7   "lunchTime": "2024-09-04"
8 }
```

Create

HTTP

Crud Tests / Create

Save

Share

POST

http://localhost:8080/timecard-system/timecards

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

```
1 {
2   "name": "John Doe",
3   "department": "Sales",
4   "entryTime": "2024-09-04T08:30:00",
5   "exitTime": "2024-09-04T17:30:00",
6   "lunchTime": "2024-09-04T12:00:00"
7 }
```

Body

Cookies

Headers (5)

Test Results

200 OK

219 ms

286 B

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 10,
3   "name": "John Doe",
4   "department": "Sales",
5   "entryTime": "2024-09-04",
6   "exitTime": "2024-09-04",
7   "lunchTime": "2024-09-04"
8 }
```



# Update

HTTP

Crud Tests / Update

Save

Share

PUT

http://localhost:8080/timecard-system/timecards

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

1 {

2   "id": 10,

3   "name": "Jane Doe",

4   "department": "IT",

5   "entryTime": "2024-09-05T08:30:00",

6   "exitTime": "2024-09-05T17:30:00",

7   "lunchTime": "2024-09-05T12:00:00"

8 }

Body

Cookies

Headers (5)

Test Results

200 OK

67 ms

283 B

Pretty

Raw

Preview

Visualize

JSON

1 {

2   "id": 10,

3   "name": "Jane Doe",

4   "department": "IT",

5   "entryTime": "2024-09-05",

6   "exitTime": "2024-09-05",

7   "lunchTime": "2024-09-05"

8 }

# Delete

Crud 1

GET Read

GET Read

POST http: •

GET Read •

POST Crea

PUT Upde •

DEL Delete •

+

▼

No environment ▼

HTTP

Crud Tests / Delete

Save ▼

Share

DELETE ▼

http://localhost:8080/timecard-system/timecards/8

Send ▼

Params

Authorization

Headers (6)

Body

Scripts

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK • 64 ms • 200 B • ...

Pretty

Raw

Preview

Visualize

Text ▼

1 Timecard deleted successfully, id: 8

# Final DB records

Limit to 1000 rows ▼

1 • `SELECT * FROM timecard_system.timecard;`

<

>

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

Result Grid

Form Editor

	id	lunch_time	department	entry_time	exit_time	name
▶	7	2024-09-05	IT	2024-09-05	2024-09-05	Jane Doe
	9	2024-09-04	Sales	2024-09-04	2024-09-04	John Doe
	10	2024-09-05	IT	2024-09-05	2024-09-05	Jane Doe
*	NULL	NULL	NULL	NULL	NULL	NULL