



# JPA

JAVA ACADEMY-XIDERAL MTY

Daniel Evaristo Escalera Bonilla

## Introduction

Java Persistence API is a set of concepts that allows the persistence of data outside of the Java space, normally with a relational database, as it is just a specification, it needs other tool to implement it, such as Hibernate.

## Implementation

To tests the concepts of JPA, I created a CRUD application that connects to a database to create, read, update and delete objects, the objects in this projects are simulating timecards, which contain information from employees and their entrance, lunch and exit time, the object was defined with the help of lombok to create its getters, setters and constructors.

```
@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;

}
```

Here many JPA annotations were used, including Entity and Table to define the object as an entity and assign it to a table, then ID and its generated value to create a primary key for the entity and perform an automatic update of its value when inserted and finally, the Column annotation to indicate more properties of the object as columns in the database.

To specify all the operations needed to operate the entity, a Data Access Object was created, in which the methods that are going to be applied to the object are defined:

```

package com.curso.v0.CrudJPA.dao;

import java.util.List;

public interface TimecardDAO {

    List<Timecard> findAll();

    List<Timecard> findByDepartment(String department);

    Timecard findById(long theId);

    Timecard save(Timecard theTimecard);

    void deleteById(long theId);
}

```

The methods include three Read operations in `findAll`, `findById` and `findByDepartment`, a write and update operation in `save` and a Delete operation in `deleteById`.

A DAO implementation class was created to define the logic of these methods and perform the actions through an Entity Manager which is the element that interacts with the database:

```

@Repository
public class TimecardDAOImpl implements TimecardDAO {

    private EntityManager entityManager;

    @Autowired
    public TimecardDAOImpl(EntityManager entityManager) {
        super();
        this.entityManager = entityManager;
    }

    @Override
    public List<Timecard> findAll() {

        TypedQuery<Timecard> theQuery = entityManager.createQuery("from Timecard", Timecard.class);

        List<Timecard> timecards = theQuery.getResultList();

        return timecards;
    }

    @Override
    public Timecard findById(long theId) {
        Timecard theTimecard = entityManager.find(Timecard.class, theId);

        return theTimecard;
    }
}

```

```

@Override
public List<Timecard> findByDepartment(String department) {
    TypedQuery<Timecard> theQuery = entityManager.createQuery("SELECT t FROM Timecard t WHERE t.department = :departmentName", Timecard.class);
    theQuery.setParameter("departmentName", department);
    List<Timecard> timecards = theQuery.getResultList();
    return timecards;
}

@Override
public Timecard save(Timecard theTimecard) {
    Timecard dbTimecard = entityManager.merge(theTimecard);
    return dbTimecard;
}

@Override
public void deleteById(long theId) {
    Timecard theTimecard = entityManager.find(Timecard.class, theId);
    entityManager.remove(theTimecard);
}

```

To activate the dao methods a Service component was defined:

```

package com.curso.v0.CrudJPA.service;

import java.util.List;

public interface TimecardService {

    List<Timecard> findAll();

    List<Timecard> findByDepartment(String name);

    Timecard findById(long theId);

    Timecard save(Timecard theTimecard);

    void deleteById(long theId);
}

```

It contains a method for each operation defined in the DAO, and calls it operation with the parameters used during the execution:

```
private TimecardDAO timecardDAO;

@Autowired
public TimecardServiceImpl(TimecardDAO timecardDAO) {
    this.timecardDAO = timecardDAO;
}

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

@Override
public Timecard findById(long theId) {
    return timecardDAO.findById(theId);
}

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

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

@Transactional
@Override
public void deleteById(long theId) {
    timecardDAO.deleteById(theId);
}
```

Finally a Rest controller defines the routes used to access the methods defined in the service, and the parameters and values that must be introduced during the request:

```

3 @RestController
4 @RequestMapping("/timecard-system")
5 public class TimecardRestController {
6
7     private TimecardService timecardService;
8
9     @Autowired
0     public TimecardRestController(TimecardService timecardService) {
1         this.timecardService = timecardService;
2     }
3
4     @GetMapping("timecards")
5     public List<Timecard> getTimecards(){
6         return timecardService.findAll();
7     }
8
9     @GetMapping("timecards/{id}")
0     public Timecard TimecardById(@PathVariable long id){
1
2         // Find the timecard by ID
3         Timecard theTimecard = timecardService.findById(id);
4
5         if (theTimecard == null) {
6             throw new RuntimeException("Timecard id not found - " + id);
7         }
8
9         return theTimecard;
0     }
1

```

```

@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

HTTPCrud 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 ]
```



getByld

HTTP

Crud Tests / Read - getByld

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 1GET ReadGET ReadPOST http:GET ReadPOST CreaPUT UpdeDEL Dele+

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 OK64 ms200 B

PrettyRawPreviewVisualizeText

1Timecard deleted successfully, id: 8

# Final DB records

Limit to 1000 rows

1 • SELECT \* FROM timecard\_system.timecard;

Result Grid

	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

Result GridForm Editor