

# Payment Application III

## Problem Statement:

← Classroom

Hibernate and Relationships  
Payment Application 3

Problem Submissions Solutions Doubts

Payment Application 3

Easy • Score 9080 • java • Advanced java • Spring Hibernate

Problem statement

In your current **Payment application**, using One-to-One mapping, you need to introduce a new Payment Details feature allowing users to view additional information about their past transactions.

▼ Tasks:-

1. Create the PaymentDetails entity class with following attributes:

- int id
- String creditAccount
- String debitAccount
- Integer amount
- String currency
- Payment payment (One To One mapping)

2. In the payment entity add the below attribute:

- PaymentDetails paymentDetails (One To One)

3. Since a single **payment detail** is linked with only a single **payment** account, create a one-to-one mapping between the Payment and PaymentDetails classes.

4. Add the below API in the **PaymentController** class:

- GET "/payment/currency/{currency}" (@PathVariable String currency): It fetches the list of all payments by the given currency.

5. In the PaymentService class add the implementation for the below method:

- getAllPaymentsByCurrency(String currency): This method returns the list of payments by the given currency.
- It only accepts the following currency in any format i.e. LowerCase/Upper/Case.
- "INR","Rupee","Dollar","Yen","Pound","USD".
- It throws an InvalidInputException if a currency different from above is received.

6. In the **PaymentDetailsController** class complete the methods to handle HTTP requests with the required annotation for the following APIs:

Send feedback

1 Step 1  
Download starter kit

2 Step 2  
Complete project on local IDE

3 Step 3  
Export code as .zip file

4 Step 4  
Upload .zip file max 50mb

Drag and drop .zip here or [browse](#)

Submit

Problem Submissions Solutions Doubts

6. In the **PaymentDetailsController** class complete the methods to handle HTTP requests with the required annotation for the following APIs:

- GET "/details/id/{id}": It fetches a payment detail object by id.
- POST "/details/save" (Body - PaymentDetails object): It saves a PaymentDetails object.
- DELETE "/details/id/{id}": It deletes a PaymentDetails object by id.
- PUT "/details/update" (Body - PaymentDetails object): It updates a PaymentDetails object.
- GET "/details/allPaymentDetails": It fetches the list of all PaymentDetails.
- GET "/details/currency/{currency}": It fetches a list of all PaymentDetails objects with the given currency.

7. Complete the **PaymentDetailsService** class as mentioned below:

a. Autowire PaymentDetailsDAL

b. Complete the following methods:

- getPaymentDetailsById(int id): This method fetches PaymentDetails for a specific id.
- spoti getAllPaymentDetails(): This method fetches a list of PaymentDetails from the database.
- savePaymentDetails(PaymentDetails newPaymentDetails): This method saves the PaymentDetails entity into the database.
- delete(int id): This method deletes a paymentDetails object from the id.
- update(PaymentDetails paymentDetails): This method updates paymentDetails object in the database.
- getByCurrency(String currency): This method fetches the list of paymentDetails for a specific currency.

8. Complete the **PaymentDetailsDALImpl** class as mentioned below:

a. Autowire EntityManager.

b. Override the following methods:

- getById(int id): This method fetches PaymentDetails for a specific id from the database.
- getAllPaymentDetails(): This method fetches the list of PaymentDetails from the database.
- save(PaymentDetails paymentDetails): This method saves the PaymentDetails entity into the database.
- delete(int id): This method deletes the PaymentDetails entity for a specific id.

## Output:

The screenshot shows a REST client interface with a POST request to `localhost:8080/payment/save`. The request body is a JSON object. The response is a 200 OK status with a response time of 1826 ms and a body size of 123 B. The response body is displayed in the 'Body' tab, showing a single line of text.

```
POST localhost:8080/payment/save
```

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "paymentType": "Credit Payment",
3   "description": "Sample description for credit payment",
4   "paymentDetails": {
5     "creditAccount": "Credit Account 1 ",
6     "debitAccount": "Debit Account 1",
7     "amount": 1000,
8     "currency": "INR"
9   }
10 }
```

Body Cookies Headers (4) Test Results 200 OK 1826 ms 123 B Save as Example

Pretty Raw Preview Visualize Text 1

The screenshot shows a REST client interface with a GET request to `localhost:8080/details/id/252`. The response is a 200 OK status with a response time of 71 ms and a body size of 274 B. The response body is displayed in the 'Body' tab, showing a JSON object with details for the payment.

```
GET localhost:8080/details/id/252
```

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results 200 OK 71 ms 274 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 252,
3   "creditAccount": "Credit Account 1 ",
4   "debitAccount": "Debit Account 1",
5   "amount": 1000,
6   "currency": "INR"
7 }
```