

Debugging

What are the objectives?

- Evaluate learners' ability to identify, analyze, and resolve issues in Java code.
- By the end of the exam, learners will have enhanced their ability to effectively debug Java applications, ensuring higher quality and more maintainable code.

MySQL Workbench Set up

- This will create the shedlock table in the banking database
- This table is necessary solely to avoid errors related to its existence or usage in the system.

USE banking;

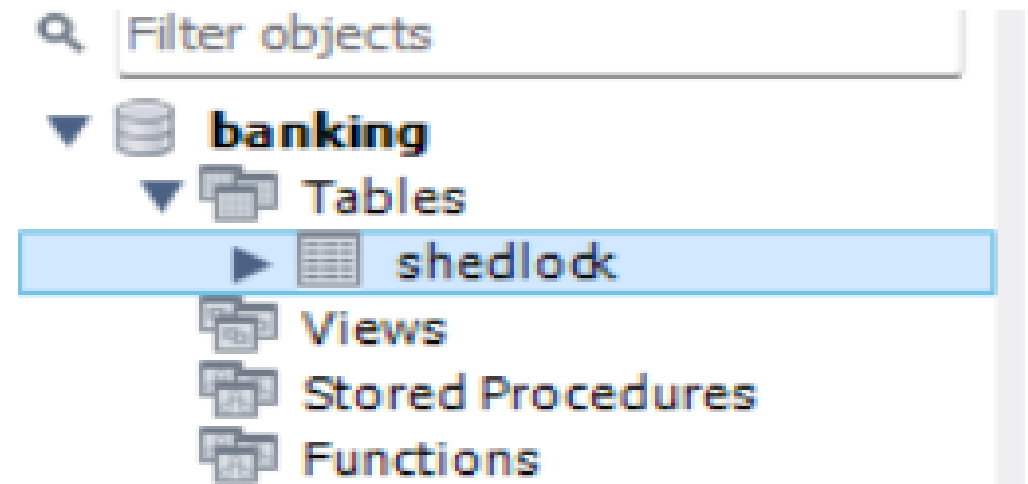
```
CREATE TABLE shedlock (  
  name VARCHAR(64) NOT NULL,  
  lock_until TIMESTAMP NOT NULL,  
  locked_at TIMESTAMP NOT NULL,  
  locked_by VARCHAR(255) NOT NULL,  
  PRIMARY KEY (name)  
);
```

MySQL Workbench Set up

- This will create the shedlock table in the banking database
- This table is necessary solely to avoid errors related to its existence or usage in the system.

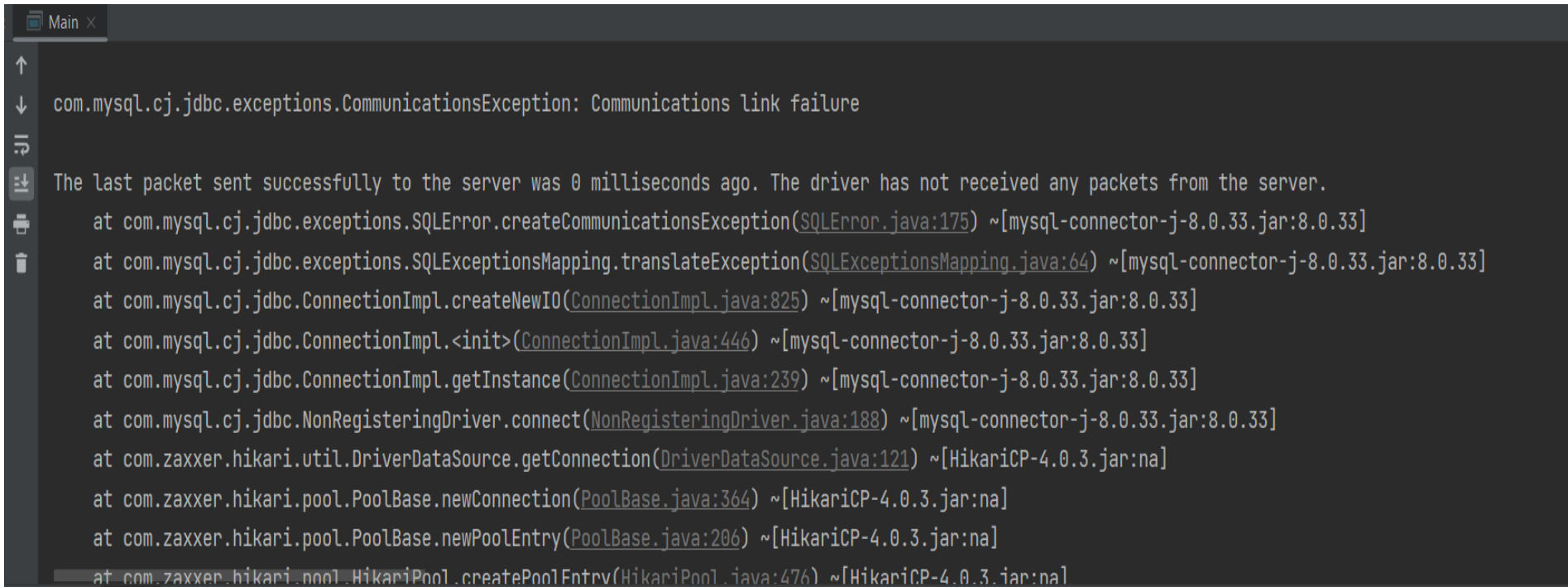
USE banking;

```
CREATE TABLE shedlock (  
  name VARCHAR(64) NOT NULL,  
  lock_until TIMESTAMP NOT NULL,  
  locked_at TIMESTAMP NOT NULL,  
  locked_by VARCHAR(255) NOT NULL,  
  PRIMARY KEY (name)  
);
```



1st Task

Start the application and check the console after it runs. Check the error message as shown below.



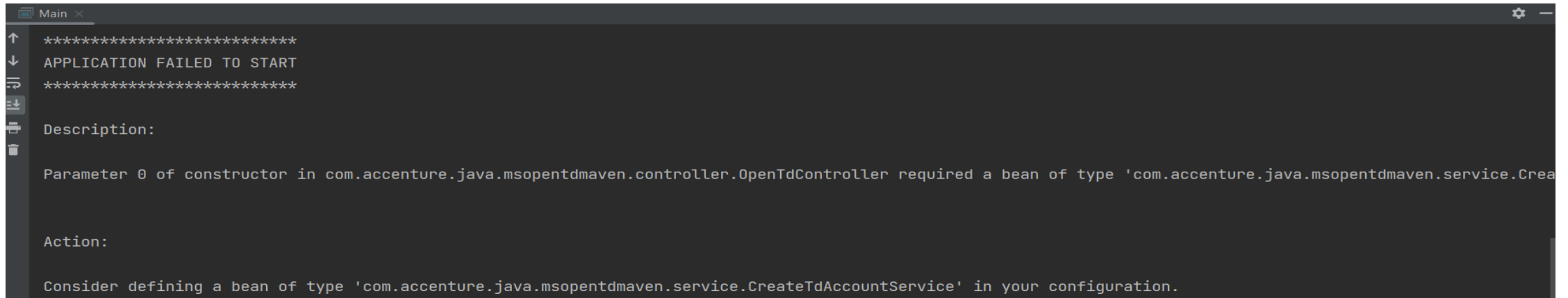
```
com.mysql.cj.jdbc.exceptions.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
at com.mysql.cj.jdbc.exceptions.SQLError.createCommunicationsException(SQLError.java:175) ~[mysql-connector-j-8.0.33.jar:8.0.33]
at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:64) ~[mysql-connector-j-8.0.33.jar:8.0.33]
at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:825) ~[mysql-connector-j-8.0.33.jar:8.0.33]
at com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:446) ~[mysql-connector-j-8.0.33.jar:8.0.33]
at com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:239) ~[mysql-connector-j-8.0.33.jar:8.0.33]
at com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:188) ~[mysql-connector-j-8.0.33.jar:8.0.33]
at com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.java:121) ~[HikariCP-4.0.3.jar:na]
at com.zaxxer.hikari.pool.PoolBase.newConnection(PoolBase.java:364) ~[HikariCP-4.0.3.jar:na]
at com.zaxxer.hikari.pool.PoolBase.newPoolEntry(PoolBase.java:206) ~[HikariCP-4.0.3.jar:na]
at com.zaxxer.hikari.pool.HikariPool.createPoolEntry(HikariPool.java:476) ~[HikariCP-4.0.3.jar:na]
```

2nd Task

If the issue in your first task has been resolved, the expected error will appear below.

Additionally, verify that the ``create_td_requests`` table has been created. You can check this using Workbench.



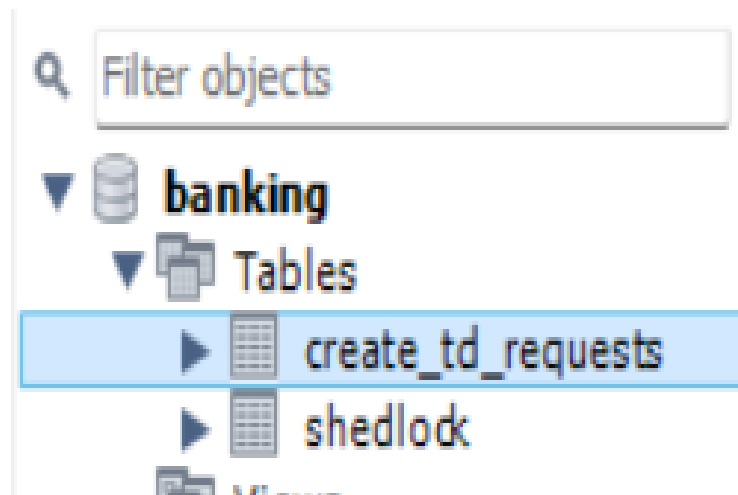
```
Main x
*****
APPLICATION FAILED TO START
*****

Description:

Parameter 0 of constructor in com.accenture.java.msopentdmaven.controller.OpenTdController required a bean of type 'com.accenture.java.msopentdmaven.service.Crea

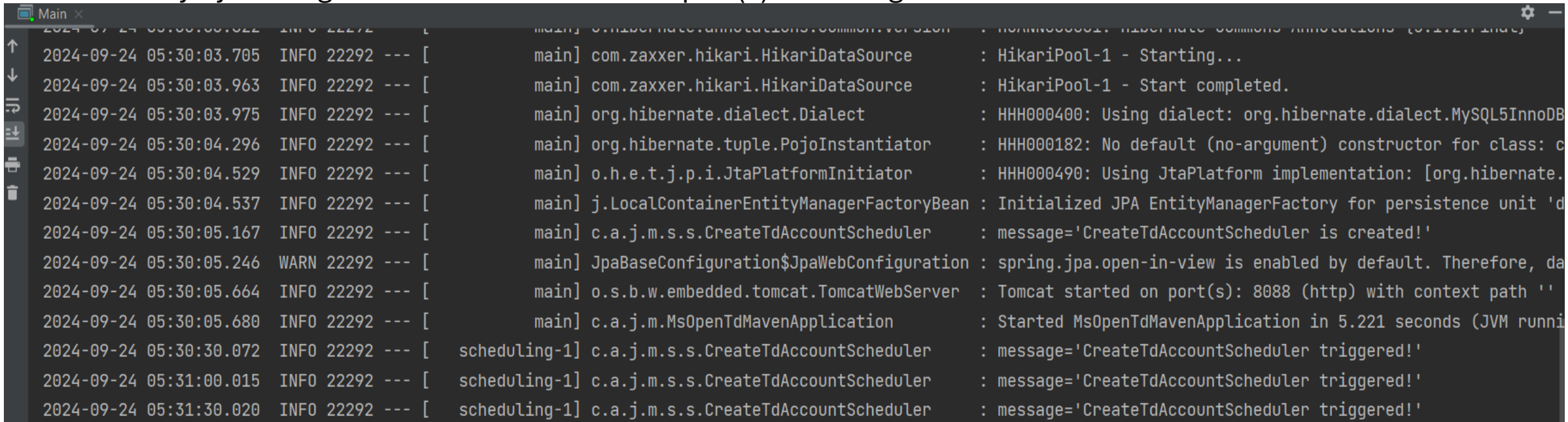
Action:

Consider defining a bean of type 'com.accenture.java.msopentdmaven.service.CreateTdAccountService' in your configuration.
```



3rd Task

If the issue in your second task has been resolved, check your console to confirm that Spring Boot is running successfully by looking for the 'Tomcat started on port(s): 8088' log.



```
2024-09-24 05:30:03.705 INFO 22292 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-09-24 05:30:03.963 INFO 22292 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-09-24 05:30:03.975 INFO 22292 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL5InnoDB
2024-09-24 05:30:04.296 INFO 22292 --- [main] org.hibernate.tuple.PojoInstantiator : HHH000182: No default (no-argument) constructor for class: c
2024-09-24 05:30:04.529 INFO 22292 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.
2024-09-24 05:30:04.537 INFO 22292 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'd
2024-09-24 05:30:05.167 INFO 22292 --- [main] c.a.j.m.s.s.CreateTdAccountScheduler : message='CreateTdAccountScheduler is created!'
2024-09-24 05:30:05.246 WARN 22292 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, da
2024-09-24 05:30:05.664 INFO 22292 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8088 (http) with context path ''
2024-09-24 05:30:05.680 INFO 22292 --- [main] c.a.j.m.MsOpenTdMavenApplication : Started MsOpenTdMavenApplication in 5.221 seconds (JVM runni
2024-09-24 05:30:30.072 INFO 22292 --- [scheduling-1] c.a.j.m.s.s.CreateTdAccountScheduler : message='CreateTdAccountScheduler triggered!'
2024-09-24 05:31:00.015 INFO 22292 --- [scheduling-1] c.a.j.m.s.s.CreateTdAccountScheduler : message='CreateTdAccountScheduler triggered!'
2024-09-24 05:31:30.020 INFO 22292 --- [scheduling-1] c.a.j.m.s.s.CreateTdAccountScheduler : message='CreateTdAccountScheduler triggered!'
```

3rd Task - continuation

Use Postman to execute this endpoint: <http://localhost:8088/ms-open-td/openAccount>

Use the JSON payload provided below to fill in the body of Postman.

```
{
  "productCode": "123456",
  "depositDetails": {
    "interestRate": "0.2",
    "depositAmount": "100000.00",
    "term": "1_YEAR",
    "effectiveDate": "21/02/2023",
    "expiryDate": "21/02/2024"
  },
  "maturityDetails": {
    "accountName": "John Sina",
    "accountNumber": "123456791"
  }
}
```

The screenshot shows the Postman interface for a POST request to the endpoint `http://localhost:8088/ms-open-td/openAccount`. The 'Body' tab is selected, and the 'JSON' format is chosen. The JSON payload is displayed in a code editor with line numbers 1 through 14. The payload is a JSON object with two main properties: 'productCode' and 'depositDetails' (which is an object itself), and 'maturityDetails' (which is also an object).

```
1 {
2   "productCode": "123456",
3   "depositDetails": {
4     "interestRate": "0.2",
5     "depositAmount": "100000.00",
6     "term": "1_YEAR",
7     "effectiveDate": "21/02/2023",
8     "expiryDate": "21/02/2024"
9   },
10  "maturityDetails": {
11    "accountName": "John Sina",
12    "accountNumber": "123456791"
13  }
14 }
```


3rd Task - continuation

Use Postman to execute this endpoint: `http://localhost:8088/ms-open-td/openAccount`

Populate the Header with the key-value pair below

Key: `correlation-id`

Value: `helloWorld`

The screenshot shows the Postman interface for a POST request. The URL bar displays `http://localhost:8088/ms-open-td/openAccount`. Below the URL bar, a tab bar contains 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Headers (9)' tab is selected and highlighted in yellow. Below the tab bar, a table lists the headers. The first header is 'correlation-id' with a checked checkbox and the value 'helloWorld'.

Key	Value
<input checked="" type="checkbox"/> correlation-id	helloWorld

3rd Task - continuation

You may encounter a 500 Internal Server Error in Postman, but you can resolve it by checking your console for more information.

Body Cookies Headers (4) Test Results

Status: 500 Internal Server Error

Pretty

Raw

Preview

Visualize

JSON



```
1 {
2   "timestamp": "2024-09-23T21:45:09.511+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "path": "/ms-open-td/openAccount"
6 }
```

```
Main x
2024-09-24 05:45:09.381 INFO 22292 --- [nio-8088-exec-2] o.s.w.s.v.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2024-09-24 05:45:09.381 INFO 22292 --- [nio-8088-exec-2] o.s.w.s.v.DispatcherServlet : Completed initialization in 0 ms
2024-09-24 05:45:09.504 ERROR 22292 --- [nio-8088-exec-2] o.a.c.c.C.[.[/].[dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet] in context
org.hibernate.InstantiationException Create breakpoint : No default constructor for entity: : com.accenture.java.msopentdmaven.repository.database.entity.CreateTdRec
at org.hibernate.tuple.PojoInstantiator.instantiate(PojoInstantiator.java:85) ~[hibernate-core-5.6.15.Final.jar:5.6.15.Final]
at org.hibernate.tuple.PojoInstantiator.instantiate(PojoInstantiator.java:105) ~[hibernate-core-5.6.15.Final.jar:5.6.15.Final]
```

4th Task

If the issue in your 3rd task has been resolved, you may continue to see a 500 Internal Server Error in Postman. The console will display “Connection refused: no further information,” indicating that your other services are not running. You can ignore this message.

Body Cookies Headers (4) Test Results

Status: 500 Internal Server Error

Pretty

Raw

Preview

Visualize

JSON



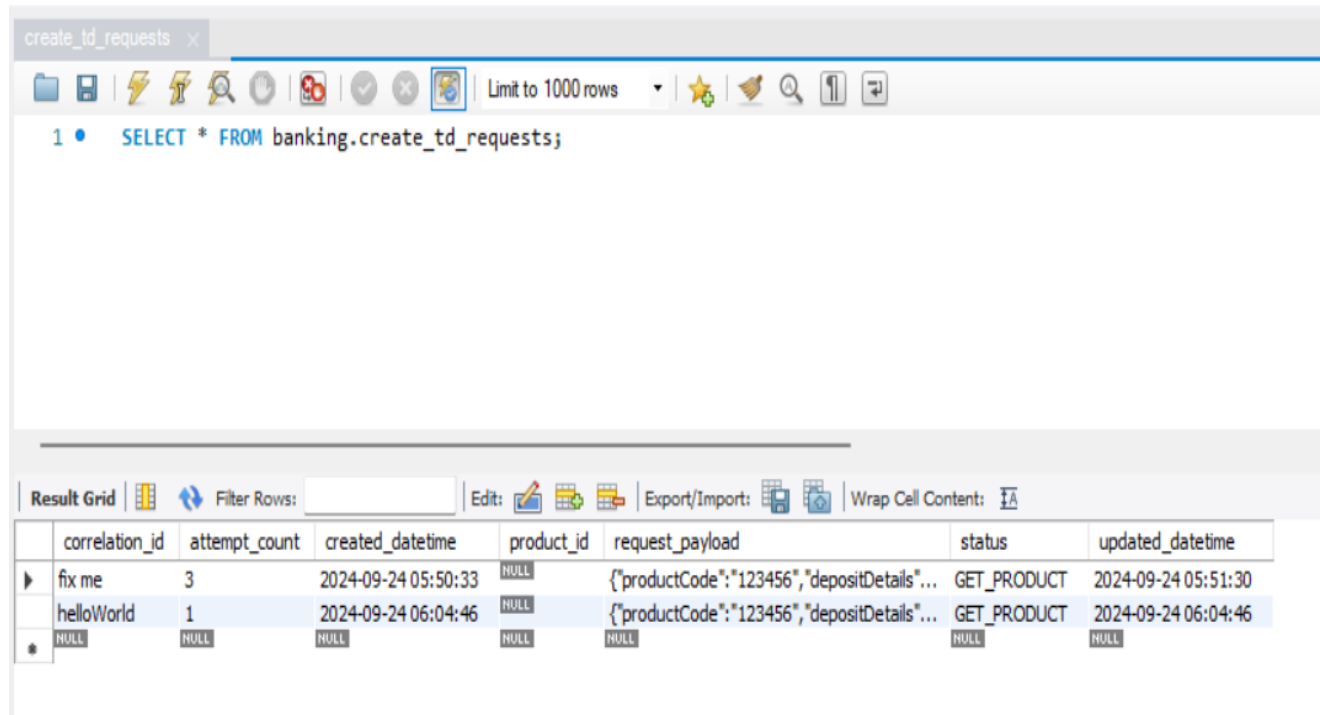
```
1 {
2   "timestamp": "2024-09-23T21:45:09.511+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "path": "/ms-open-td/openAccount"
6 }
```

```
Main x
2024-09-24 05:50:32.933 ERROR 38176 --- [nio-8088-exec-2] o.a.c.c.C.[...].dispatcherServlet : Servlet.se

java.net.ConnectException Create breakpoint : Connection refused: no further information
    at java.base/sun.nio.ch.Net.pollConnect(Native Method) ~[na:na]
    at java.base/sun.nio.ch.Net.pollConnectNow(Net.java:672) ~[na:na]
```


5th Task

If you think you resolved 4th task, run Postman again. Check if correlation_id is populated with the correct value. If it is, run postman again, and you will see the error message “Correlation id already exists.”



The screenshot shows a database client interface with a tab labeled 'create_td_requests'. The SQL query editor contains the following query:

```
1 • SELECT * FROM banking.create_td_requests;
```

Below the query editor, the 'Result Grid' displays the following data:

	correlation_id	attempt_count	created_datetime	product_id	request_payload	status	updated_datetime
▶	fix me	3	2024-09-24 05:50:33	NULL	{ "productCode": "123456", "depositDetails": ...	GET_PRODUCT	2024-09-24 05:51:30
	helloWorld	1	2024-09-24 06:04:46	NULL	{ "productCode": "123456", "depositDetails": ...	GET_PRODUCT	2024-09-24 06:04:46
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Status: 400 Bad Request

```
1 {
2   "errorCode": 201,
3   "errorMessage": "Correlation id already exists.",
4   "errorDetails": {}
5 }
```

5th Task

If you think you resolved 4th task, run Postman again. Check if correlation_id is populated with the correct value. If it is, run postman again, and you will see the error message “Correlation id already exists.”

The screenshot shows the Postman interface with a tab titled 'create_id_requests'. The SQL query entered is `SELECT * FROM banking.create_td_requests;`. Below the query, the 'Result Grid' is displayed with the following data:

	correlation_id	attempt_count	created_datetime	product_id	request_payload	status	updated_datetime
▶	fix me	3	2024-09-24 05:50:33	NULL	{"productCode":"123456","depositDetails"...	GET_PRODUCT	2024-09-24 05:51:30
	helloWorld	1	2024-09-24 06:04:46	NULL	{"productCode":"1		
*	NULL	NULL	NULL	NULL	NULL		

Below the table, the 'Body' tab is selected, showing the JSON response:

```
{
  "errorCode": 201,
  "errorMessage": "Correlation id already exists.",
  "errorDetails": {}
}
```

Status: 400 Bad Request

5th Task - continuation

The expected result is that the error code should be 400.

Note: Avoid using the hardcoded value 400.

Body Cookies Headers (4) Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "errorCode": 400,  
3   "errorMessage": "Correlation id already exists.",  
4   "errorDetails": {}  
5 }
```



Status: 400 Bad Request