1st step in demo

Don’t Initialize Redis Cache only provide MySQL connection

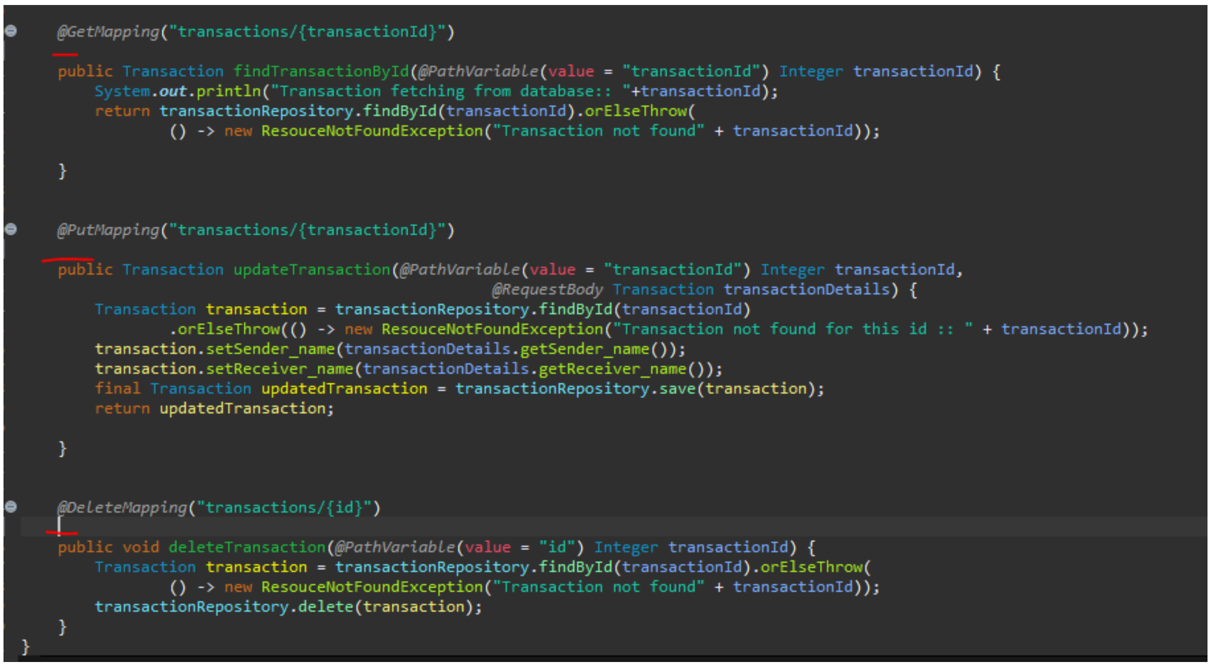
In demo.controller the class “Transactioncontroller”

```
@Cacheable(value = "transactions",key = "#transactionId")
```

```
@CachePut(value = "transactions",key = "#transactionId")
```

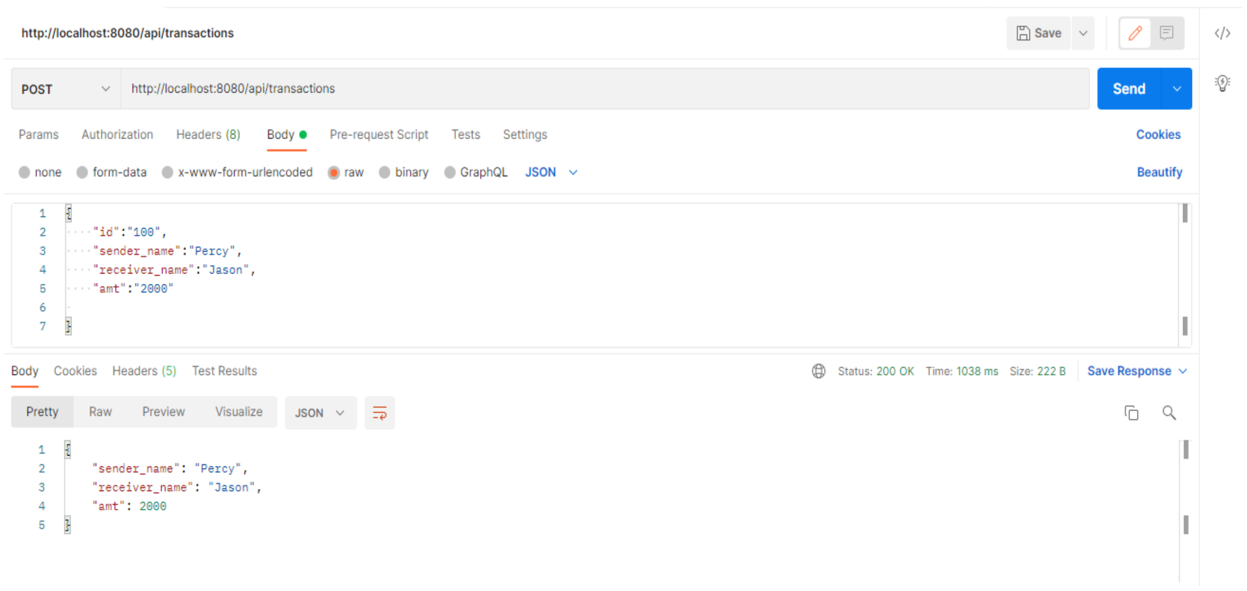
```
@CacheEvict(value = "transactions", allEntries = true)
```

Do not provide these codes



```
    @GetMapping("transactions/{transactionId}")

    public Transaction findTransactionById(@PathVariable(value = "transactionId") Integer transactionId) {
        System.out.println("Transaction fetching from database:: "+transactionId);
        return transactionRepository.findById(transactionId).orElseThrow(
                () -> new ResouceNotFoundException("Transaction not found" + transactionId));

    }


    @PutMapping("transactions/{transactionId}")

    public Transaction updateTransaction(@PathVariable(value = "transactionId") Integer transactionId,
                                         @RequestBody Transaction transactionDetails) {
        Transaction transaction = transactionRepository.findById(transactionId)
                .orElseThrow(() -> new ResouceNotFoundException("Transaction not found for this id :: " + transactionId));
        transaction.setSender_name(transactionDetails.getSender_name());
        transaction.setReceiver_name(transactionDetails.getReceiver_name());
        final Transaction updatedTransaction = transactionRepository.save(transaction);
        return updatedTransaction;

    }


    @DeleteMapping("transactions/{id}")

    public void deleteTransaction(@PathVariable(value = "id") Integer transactionId) {
        Transaction transaction = transactionRepository.findById(transactionId).orElseThrow(
                () -> new ResouceNotFoundException("Transaction not found" + transactionId));
        transactionRepository.delete(transaction);
    }
}
```

Directly run RedisCacheApplication.java


http://localhost:8080/api/transactions

POST http://localhost:8080/api/transactions

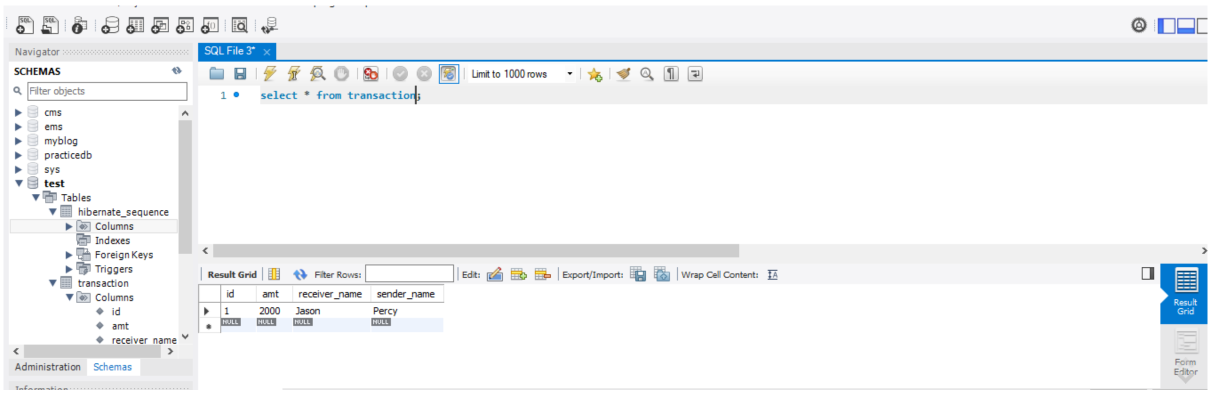
Body: raw, JSON

```
{
    "id":"100",
    "sender_name":"Percy",
    "receiver_name":"Jason",
    "amt":"2000"

}
```

Status: 200 OK Time: 1038 ms Size: 222 B

```
{
    "sender_name": "Percy",
    "receiver_name": "Jason",
    "amt": 2000
}
```

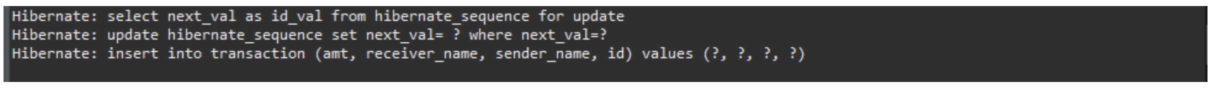
The following will be executed in MySQL database



```
select * from transaction;
```

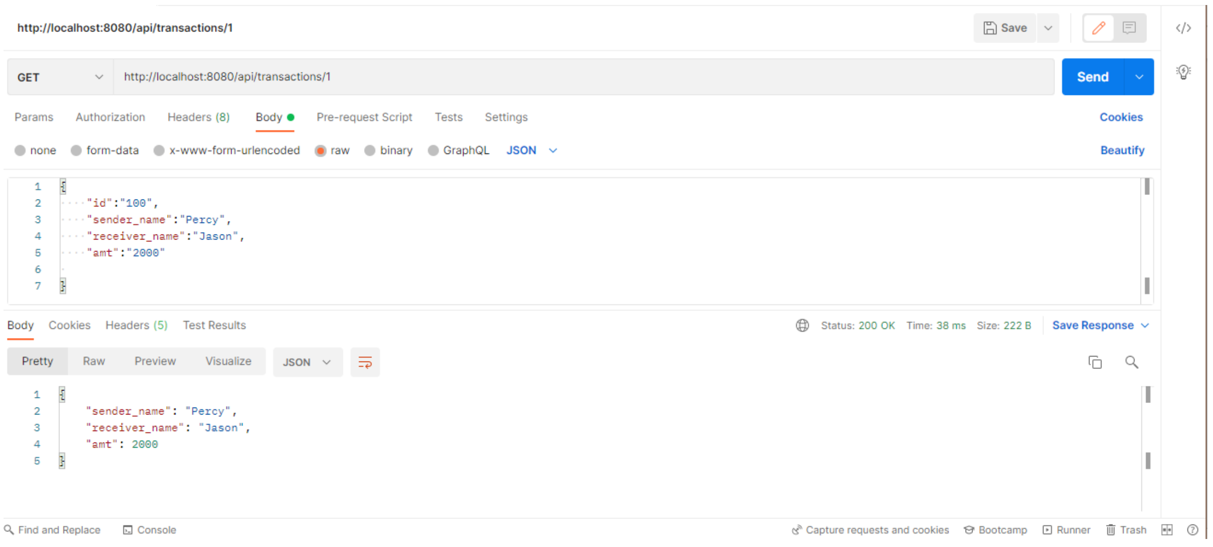
id	amt	receiver_name	sender_name
1	2000	Jason	Percy
NULL	NULL	NULL	NULL

From IDE console



```
Hibernate: select next_val as id_val from hibernate_sequence for update
Hibernate: update hibernate_sequence set next_val= ? where next_val=?
Hibernate: insert into transaction (amt, receiver_name, sender_name, id) values (?, ?, ?, ?)
```

Now we will try GET method:


http://localhost:8080/api/transactions/1

GET http://localhost:8080/api/transactions/1

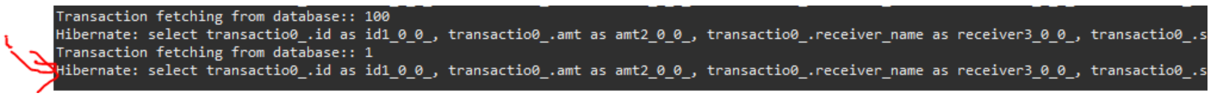
Body: raw, JSON

```
{
    "id":"100",
    "sender_name":"Percy",
    "receiver_name":"Jason",
    "amt":"2000"

}
```

Status: 200 OK Time: 38 ms Size: 222 B

```
{
    "sender_name": "Percy",
    "receiver_name": "Jason",
    "amt": 2000
}
```

You will get this output in console:



```
Transaction fetching from database:: 100
Hibernate: select transactio0_.id as id1_0_0_, transactio0_.amt as amt2_0_0_, transactio0_.receiver_name as receiver3_0_0_, transactio0_.s
Transaction fetching from database:: 1
Hibernate: select transactio0_.id as id1_0_0_, transactio0_.amt as amt2_0_0_, transactio0_.receiver_name as receiver3_0_0_, transactio0_.s
```

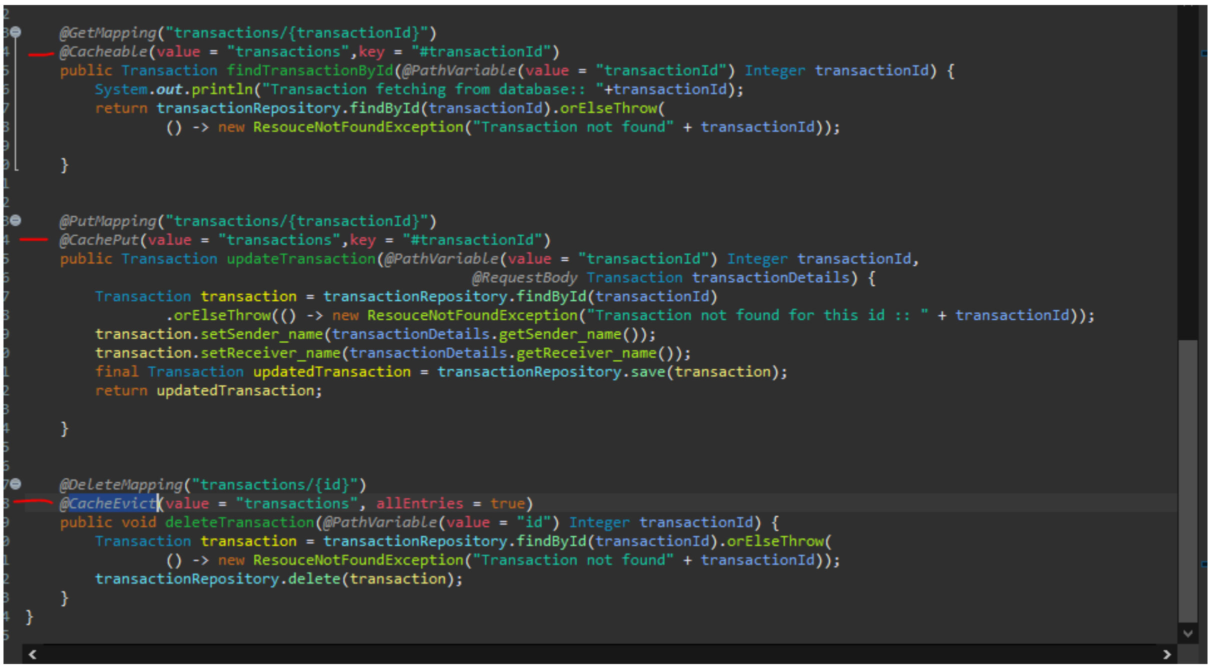
Let’s Shorten the time when we try GET method with redis cache

@Cacheable(value = "transactions",key = "#transactionId")

@CachePut(value = "transactions",key = "#transactionId")

@CacheEvict(value = "transactions", allEntries = true)

Put this 3 codes in TransactionController

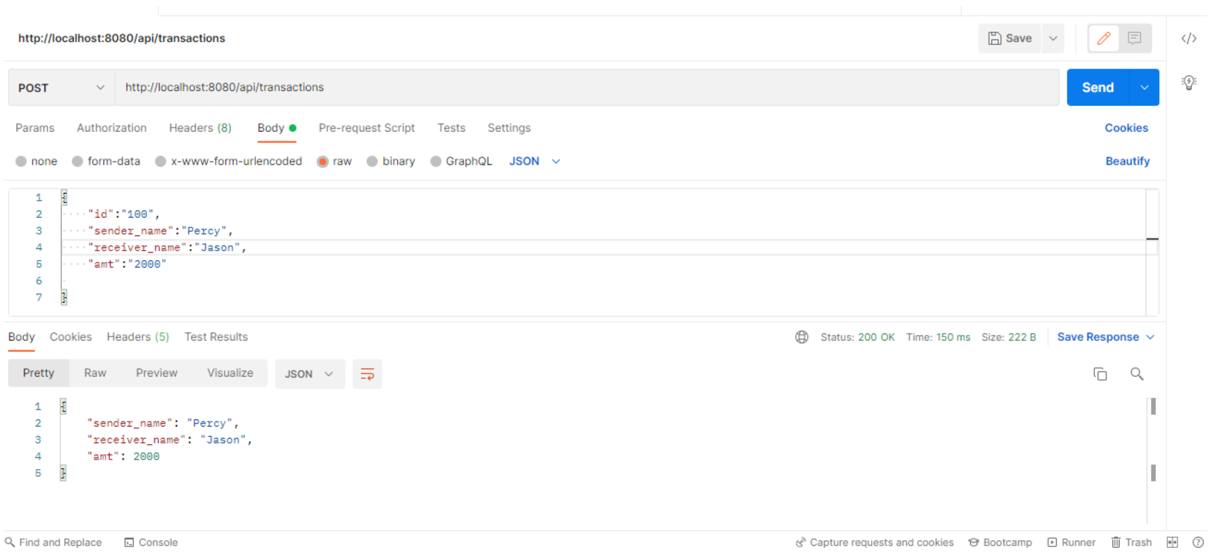


```java
    @GetMapping("transactions/{transactionId}")
    @Cacheable(value = "transactions",key = "#transactionId")
    public Transaction findTransactionById(@PathVariable(value = "transactionId") Integer transactionId) {
        System.out.println("Transaction fetching from database:: "+transactionId);
        return transactionRepository.findById(transactionId).orElseThrow(
                () -> new ResouceNotFoundException("Transaction not found" + transactionId));

    }

    @PutMapping("transactions/{transactionId}")
    @CachePut(value = "transactions",key = "#transactionId")
    public Transaction updateTransaction(@PathVariable(value = "transactionId") Integer transactionId,
                                                       @RequestBody Transaction transactionDetails) {
        Transaction transaction = transactionRepository.findById(transactionId)
                .orElseThrow(() -> new ResouceNotFoundException("Transaction not found for this id :: " + transactionId));
        transaction.setSender_name(transactionDetails.getSender_name());
        transaction.setReceiver_name(transactionDetails.getReceiver_name());
        final Transaction updatedTransaction = transactionRepository.save(transaction);
        return updatedTransaction;

    }

    @DeleteMapping("transactions/{id}")
    @CacheEvict(value = "transactions", allEntries = true)
    public void deleteTransaction(@PathVariable(value = "id") Integer transactionId) {
        Transaction transaction = transactionRepository.findById(transactionId).orElseThrow(
                () -> new ResouceNotFoundException("Transaction not found" + transactionId));
        transactionRepository.delete(transaction);
    }
}
```

And `@EnableCaching`

RedisCacheApplication.java

And run



http://localhost:8080/api/transactions

POST http://localhost:8080/api/transactions

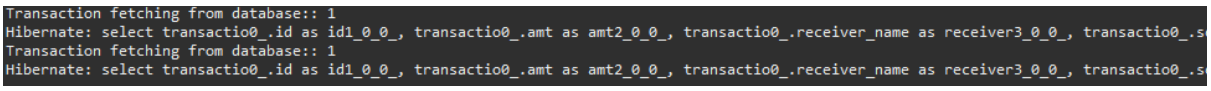
Body: raw JSON

```
{
    "id":"100",
    "sender_name":"Percy",
    "receiver_name":"Jason",
    "amt":"2000"

}
```

Status: 200 OK Time: 150 ms Size: 222 B

```
{
    "sender_name": "Percy",
    "receiver_name": "Jason",
    "amt": 2000
}
```

IDE console

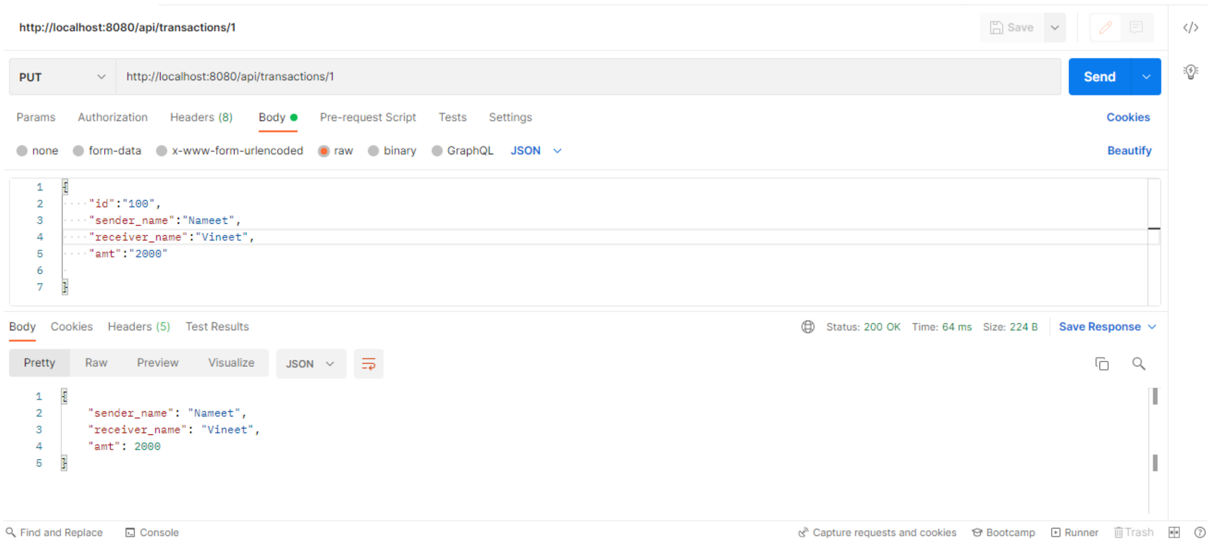


```
Transaction fetching from database:: 1
Hibernate: select transactio0_.id as id1_0_0_, transactio0_.amt as amt2_0_0_, transactio0_.receiver_name as receiver3_0_0_, transactio0_.s
Transaction fetching from database:: 1
Hibernate: select transactio0_.id as id1_0_0_, transactio0_.amt as amt2_0_0_, transactio0_.receiver_name as receiver3_0_0_, transactio0_.s
```

No matter how many times you "GET" Method

Transaction fetching won't come again and again

Update by "PUT"



http://localhost:8080/api/transactions/1

PUT http://localhost:8080/api/transactions/1

Body: raw JSON

```
{
    "id":"100",
    "sender_name":"Nameet",
    "receiver_name":"Vineet",
    "amt":"2000"

}
```

Status: 200 OK Time: 64 ms Size: 224 B

```
{
    "sender_name": "Nameet",
    "receiver_name": "Vineet",
    "amt": 2000
}
```

Lets put in swagger

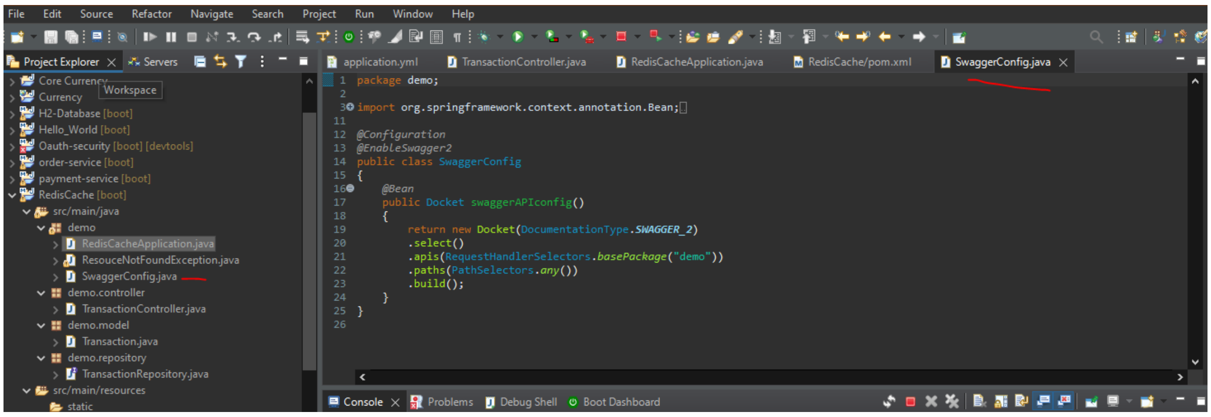
Adding Swagger

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->
<dependency>
<groupId>io.springfox</groupId>
```

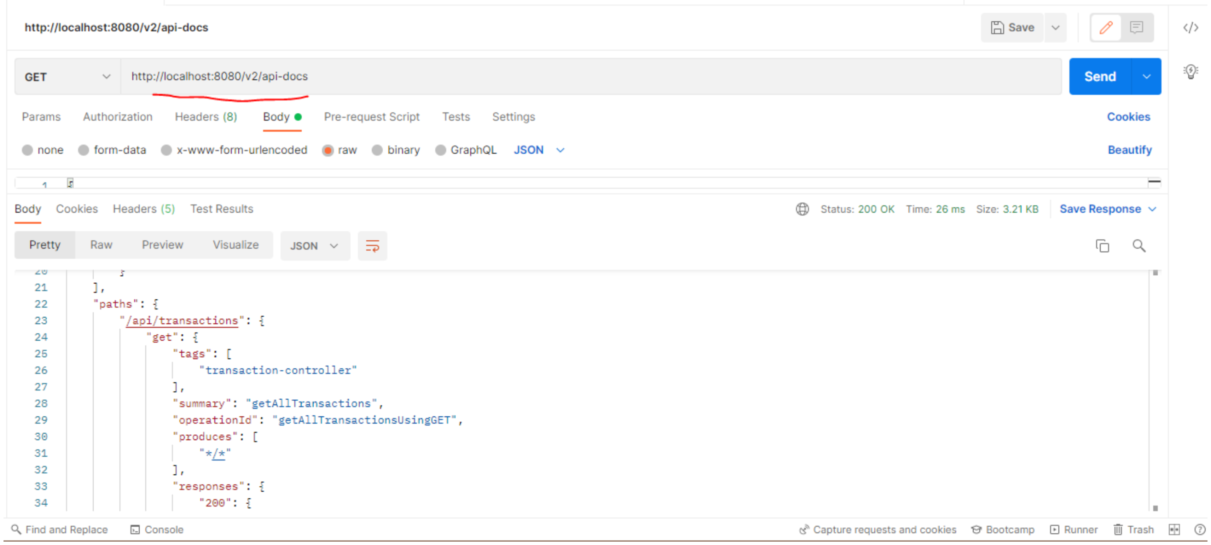
```
<artifactId>springfox-swagger2</artifactId>
<version>2.9.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger-ui</artifactId>
<version>2.9.2</version>
</dependency>
```

Add these dependency

Make a Swagger Config class



```java
package demo;

import org.springframework.context.annotation.Bean;

@Configuration
@EnableSwagger2
public class SwaggerConfig
{
    @Bean
    public Docket swaggerAPIconfig()
    {
        return new Docket(DocumentationType.SWAGGER_2)
        .select()
        .apis(RequestHandlerSelectors.basePackage("demo"))
        .paths(PathSelectors.any())
        .build();
    }
}
```

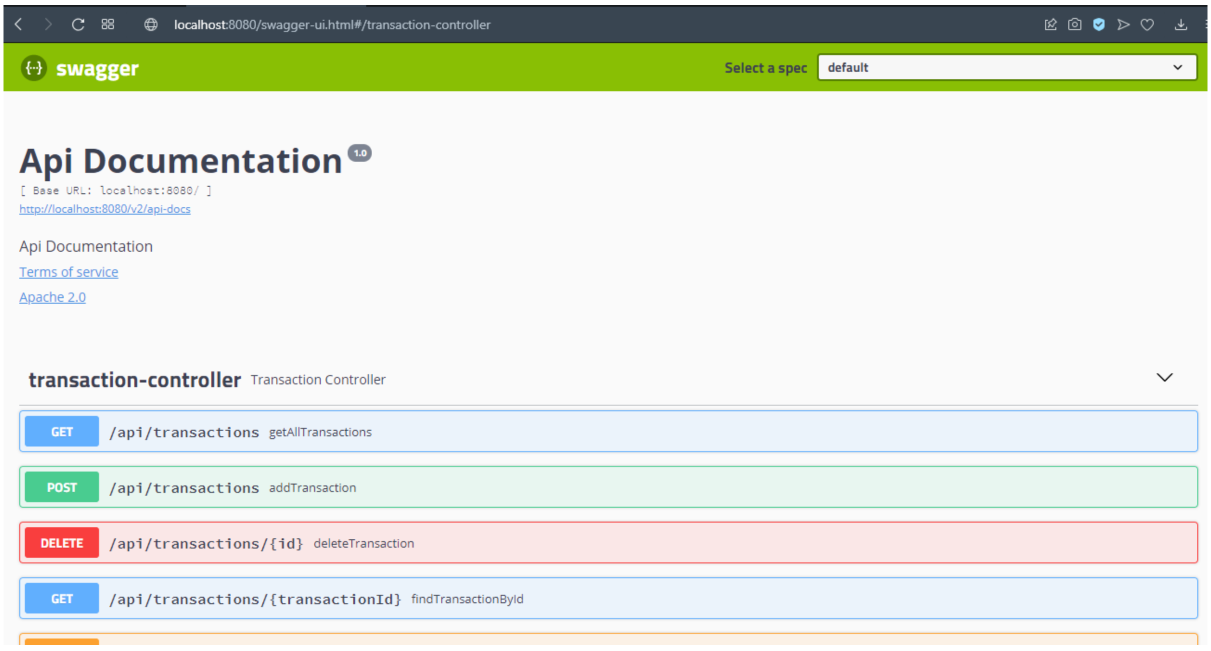


GET http://localhost:8080/v2/api-docs

Status: 200 OK Time: 26 ms Size: 3.21 KB

```
    ],
    "paths": {
        "/api/transactions": {
            "get": {
                "tags": [
                    "transaction-controller"
                ],
                "summary": "getAllTransactions",
                "operationId": "getAllTransactionsUsingGET",
                "produces": [
                    "*/*"
                ],
                "responses": {
                    "200": {
```

Open UI


localhost:8080/swagger-ui.html#/transaction-controller

swagger

Select a spec: default

Api Documentation 1.0

[Base URL: localhost:8080/]

http://localhost:8080/v2/api-docs

Api Documentation

Terms of service

Apache 2.0

transaction-controller Transaction Controller

GET /api/transactions getAllTransactions

POST /api/transactions addTransaction

DELETE /api/transactions/{id} deleteTransaction

GET /api/transactions/{transactionId} findTransactionById