Lab 4.1

Transactions

In your application create a new class com.student.exception.MyException extends Exception. This is a checked exception and Spring will not automatically rollback a transaction if this exception is thrown in the transaction scope.

In the class StudentController, have the method tied to a post (@PostMapping) , throw a MyException if the student fees are greater than 200.00

```
@PostMapping
public ResponseEntity<String> add(@RequestBody Student student) throws MyException{
    studentRepository.save(student);
    if(student.getFees() > 200.00) {
        throw new MyException("Blow Up");
    }
    return ResponseEntity.accepted().header("location", "/student/"+ student.getId()).build();
}
```

Now annotate the method as being @Transactional and to rollback for MyException.class

Lab 4.1

Transactions

In the tester class StudentControllerRestTemplateTest, comment out the existing test and add a new negative test that you can find in the lab setup for this lab in the file test.txt

Relaunch your Spring Boot application

Run the test, did we get anything inserted? Or did we get a BAD_REQUEST? Although the methods from CrudRepository are marked as Transactional, it means they will use a transaction if called within a transaction otherwise I will create one for the “save” method. In this case we called it in an existing transaction that then got rolled back, hence no commit of the data

Lab 4.1

Transactions

- **However, add the following method to the StudentRepository**

```
@Transactional(propagation = Propagation.REQUIRES_NEW)
Student save(Student entity);
```

- **Relaunch your Spring Boot application, re-test. We still get the Exception and BAD_REQUEST back from the service to the client, but the Student insertion ran in its own transaction this time and was subsequently committed even though the original transaction was rolled back**