TDD - Back-end

TDD - Back-end

- Practice: Software Development Practice

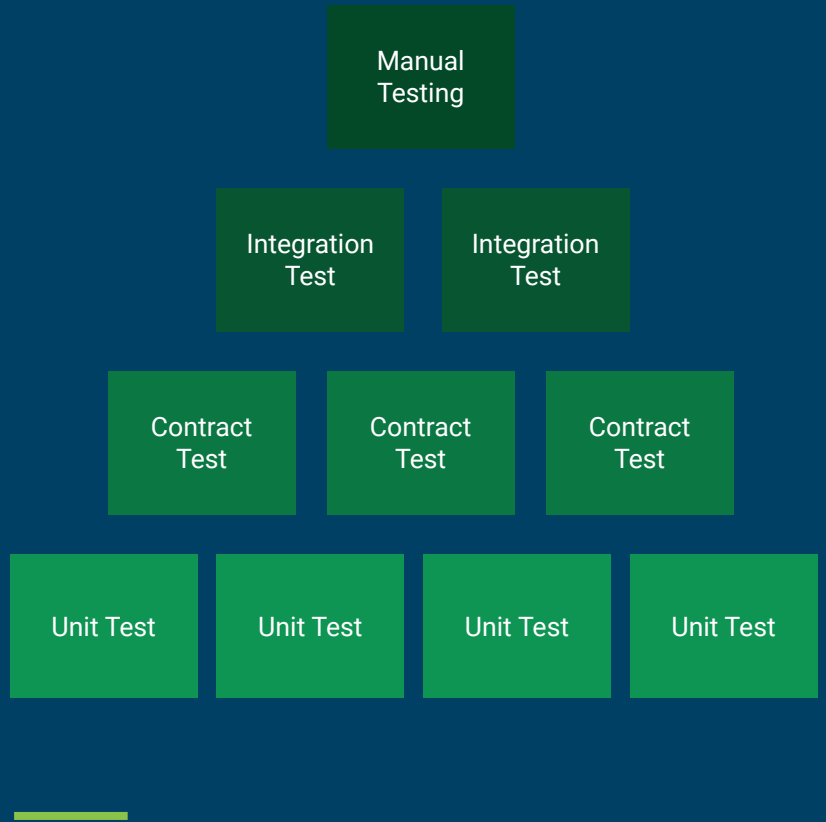
Pattern:

- Red
- Green
- Refactor

TDD - Agile Test Pyramid

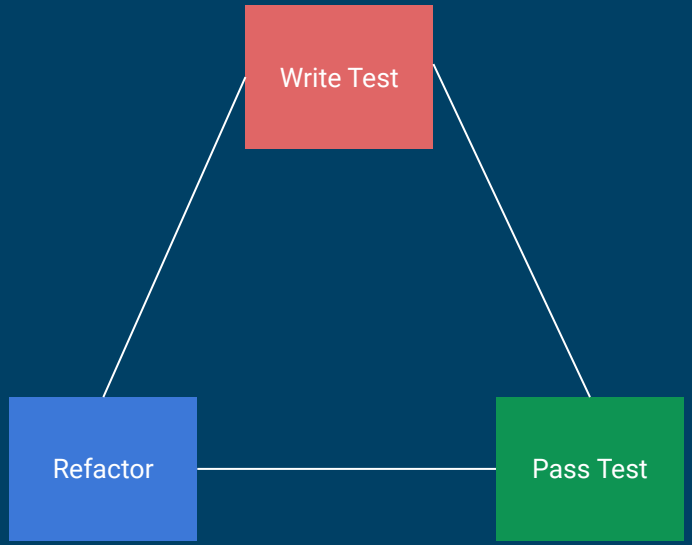
"Ideal World"

Who did Unit Testing in their University Project (COS301)?



TDD - Process

Write Test
Pass Test
Refactor



TDD - RED

Write Test

Write Test

```java
public class EnglishConvertOneToNameServiceTest {

    private EnglishConvertOneToNameService englishConvertOneToNameService
            = new EnglishConvertOneToNameService();

    @Test
    public void shouldReturnValidStringAsNumber() {
        String one = "one";

        String result = englishConvertOneToNameService.convertToString(1);
        Assertions.assertEquals(one, result);
    }

}
```

```java
@Service
public class EnglishConvertOneToNameService {

}
```

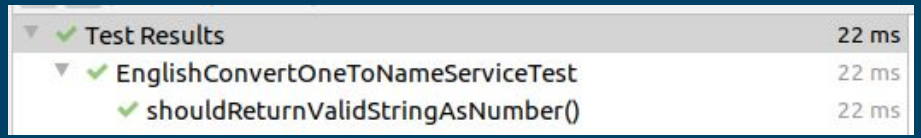
TDD - GREEN

Pass Test

```java
public class EnglishConvertOneToNameServiceTest {

    private EnglishConvertOneToNameService englishConvertOneToNameService
            = new EnglishConvertOneToNameService();

    @Test
    public void shouldReturnValidStringAsNumber() {
        String one = "one";

        String result = englishConvertOneToNameService.convertToString( i: 1);
        Assertions.assertEquals(one, result);
    }

}
```

Pass Test

```java
@Service
public class EnglishConvertOneToNameService {

    public String convertToString(int i) {
        if (i == 1) {
            return "one";
        }
        return "";
    }

}
```



TDD - BLUE

Refactor

```
@Service
public class EnglishConvertOneToNameService {
```

Refactor

```
@Service
public class EnglishConvertNumberToNameService {
```

TDD - RED

Write Test

```
@Service
public class EnglishConvertNumberToNameService {

    public String convertToString(int i) {
        return "one";
    }

}
```

```
org.opentest4j.AssertionFailedError:
Expected :two
Actual   :one
<Click to see difference>
```

Write Test

```
public class EnglishConvertNumberToNameServiceTest {

    private EnglishConvertNumberToNameService englishNumberNameService
            = new EnglishConvertNumberToNameService();

    private static final Map<Integer, String> otherNumbers = new HashMap<>();

    @BeforeAll
    static void init() {
        otherNumbers.put(1, "one");
        otherNumbers.put(2, "two");
        otherNumbers.put(5, "five");
        otherNumbers.put(9, "nine");
        otherNumbers.put(18, "eighteen");
        otherNumbers.put(19, "nineteen");
    }

    @Test
    public void shouldReturnValidStringAsNumber() {
        for (int key : otherNumbers.keySet()) {
            String result = englishNumberNameService.convertToString(key);
            Assertions.assertEquals(otherNumbers.get(key), result);
        }
    }
}
```

TDD - GREEN

Pass Test

Pass Test

```java
public class EnglishConvertNumberToNameServiceTest {

    private EnglishConvertNumberToNameService englishNumberNameS
            = new EnglishConvertNumberToNameService();

    private static final Map<Integer, String> otherNumbers = new

    @BeforeAll
    static void init() {
        otherNumbers.put(1, "one");
        otherNumbers.put(2, "two");
        otherNumbers.put(5, "five");
        otherNumbers.put(9, "nine");
        otherNumbers.put(18, "eighteen");
        otherNumbers.put(19, "nineteen");
    }

    @Test
    public void shouldReturnValidStringAsNumber() {
        for (int key : otherNumbers.keySet()) {
            String result = englishNumberNameService.convertToSt
            Assertions.assertEquals(otherNumbers.get(key), resul
        }
    }
}
```

```java
public class NumNames {

    public static final String[] NUM_NAMES = {
            "",
            "one",
            "two",
            "three",
            "four",
            "five",
            "six",
            "seven",
            "eight",
            "nine",
            "ten",
            "eleven",
            "twelve",
            "thirteen",
            "fourteen",
            "fifteen",
            "sixteen",
            "seventeen",
            "eighteen",
            "nineteen",
            "twenty",
    };
}
```

```java
@Service
public class EnglishConvertNumberToNameService {

    public String convertToString(int i) {
        return NumNames.NUM_NAMES[i % 20];
    }

}
```

Test Results 15 ms
EnglishConvertNumberToNameServiceTest 15 ms
shouldReturnValidStringAsNumber() 15 ms

TDD - Takeaways for Spring boot

- Spring Boot

- @Data - Lombok annotations
- @Mapper - Mapper - you don’t have to write everything

TDD - Takeaways for testing

- Postman
- Unit Test Annotations
- Mockito
- H2 database for integration testing
- Value of Testing

- @Test: Code that are executed as an individual test, can contain multiple assertions
- @BeforeAll: Code that executes before all other tests are executed
- @ExtendWith(MockitoExtension.class): Allows @Mock for services - then you can mock values

TDD - Demo

What we have:

- Spring Boot Initializr App (https://start.spring.io/)
- PostgreSQL Database Created

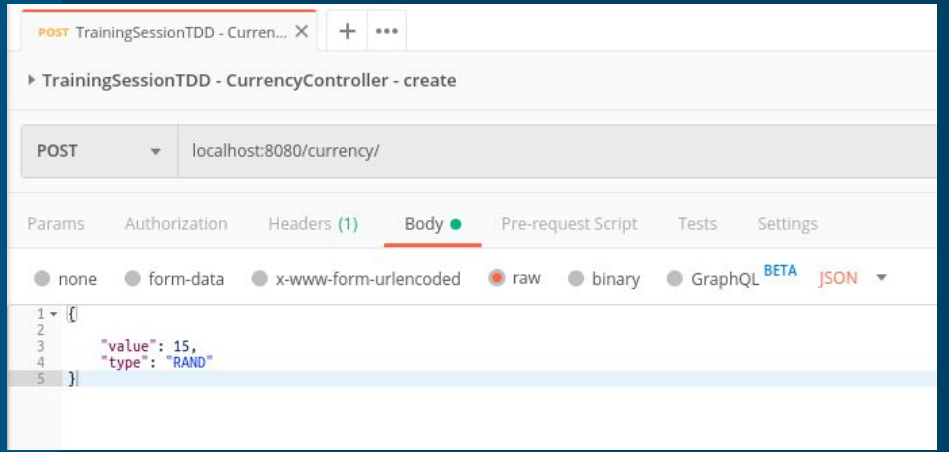
```
postgres=# create user tdd with password 'tdd';
CREATE ROLE
postgres=# CREATE DATABASE tdd OWNER tdd;
CREATE DATABASE
postgres=#
```

- Liquibase Scripts Creating Database (schema.sql)
- All necessary maven dependencies (pom.xml)

- Spring Initializer:
  - spring-boot-starter-data-jpa
  - Spring-boot-starter-web
  - spring-boot-starter-test
  - Lombok
- Maven (pom.xml):
  - Postgresql
  - Spring-boot-devtools
  - Mapstruct
  - junit-jupiter-engine
  - junit-jupiter-params
  - mockito-core
  - junit-jupiter-api
  - h2

TDD - What else?

Manual Testing



TDD - What else?

Manual Testing

```
mycurl() {
    response=$(
      curl \
      --write-out %{http_code} \
      --output /dev/null \
      --silent \
      --location --request POST 'http://192.168.99.101:31359/shipments/find' \
      --header 'Content-Type: application/json' \
      --header 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCbB0ZRaIZvRyUQ' \
      --data-raw '
      {
        "pagination": {
          "size": 5,
          "page": 0
        }
      }'
    )
    echo "$1 = $response"
    if [ "$response" != "200" ]
    then
        # invalid, so we can echo the response and get the value from it
        echo "$1 = $response"
        echo "invalid"
    fi
}
export -f mycurl

i=0

numTimes=100

echo "Running curl for $numTimes times in parralel"
seq $numTimes | parallel -j0 mycurl
```

TDD - What else?

Some strange manual testing I have done myself...

Manual Testing

file:///home/avoid/Dev/Instasense/Santova/Source/mappings/CosmosDB/Tests/shipment-udf-betweenDateRange-test.html

```
        writeContent(`shouldFalseMoreThanEndForActual == ${!betweenDateRange('2019-06-05', '2019-06-13', null, '2019
        writeContent(`shouldFalseMoreThanEndForActualAndOptForActualOverEstimated == ${!betweenDateRange('2019-06-05

        writeContent('--------------divider--------------');

        //FALSE (base cases)
        writeContent(`shouldFalseNothingInput == ${!betweenDateRange()}`);
        writeContent(`shouldFalseNoEstimatedOrActualDates == ${!betweenDateRange('2019-06-05', '2019-06-13')}`);
        writeContent(`shouldFalseNoStartOrEndDate == ${!betweenDateRange(null, null, '2019-06-04', '2019-06-04')}`);


        function writeContent(content) {
            let listOfTests = document.getElementById('listOfTests');
            let newElementContent = document.createElement(`li`);
            newElementContent.textContent = content;
            listOfTests.appendChild(newElementContent);
        }
    }
</script>
</head>
<body>
    <h1>Open with browser file:///__LocationWhereFileSits__/file.html</h1>
    <h1>Unit Tests for 'shipment-udf-betweenDateRange' - betweenDateRange(start, end, estimated, actual) function</h1>
    <h2>Note: Relative path used - open in file explorer! Not through IDE - which opens server</h2>
    <h3>All unit tests below should output True, negation is used for the false values</h3>

    <p>Individual unit tests:</p>
```

TDD - Where is everything?

"Ideal World"

Manual Testing

Integration Test

Contract Test

Unit Test

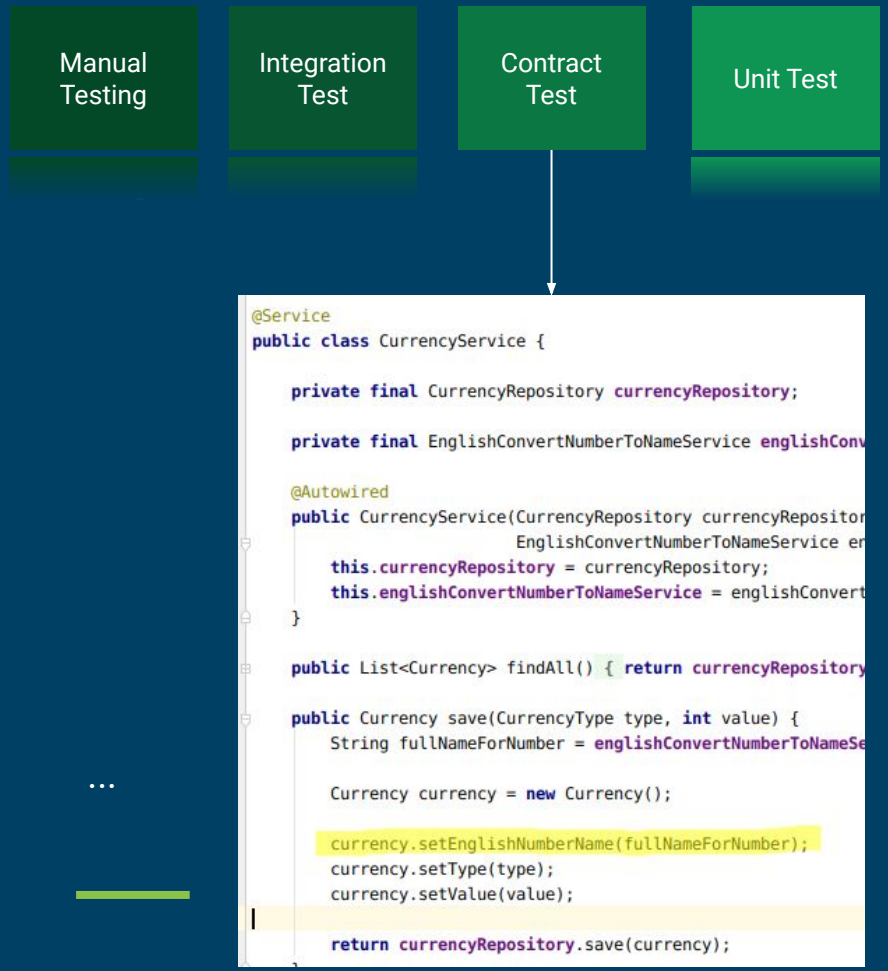
```
public class EnglishConvertNumberToNameServiceTest {

    private EnglishConvertNumberToNameService englishNumberNameService
            = new EnglishConvertNumberToNameService();

    private static final Map<Integer, String> otherNumbers = new HashMap<>();

    @BeforeAll
    static void init() {
        otherNumbers.put(1, "one");
        otherNumbers.put(2, "two");
        otherNumbers.put(5, "five");
        otherNumbers.put(9, "nine");
        otherNumbers.put(18, "eighteen");
        otherNumbers.put(19, "nineteen");
    }

    @Test
    public void shouldReturnValidStringAsNumber() {
        for (int key : otherNumbers.keySet()) {
            String result = englishNumberNameService.convertToString(key);
            Assertions.assertEquals(otherNumbers.get(key), result);
        }
    }
}
```

... ...

TDD - Where is everything?

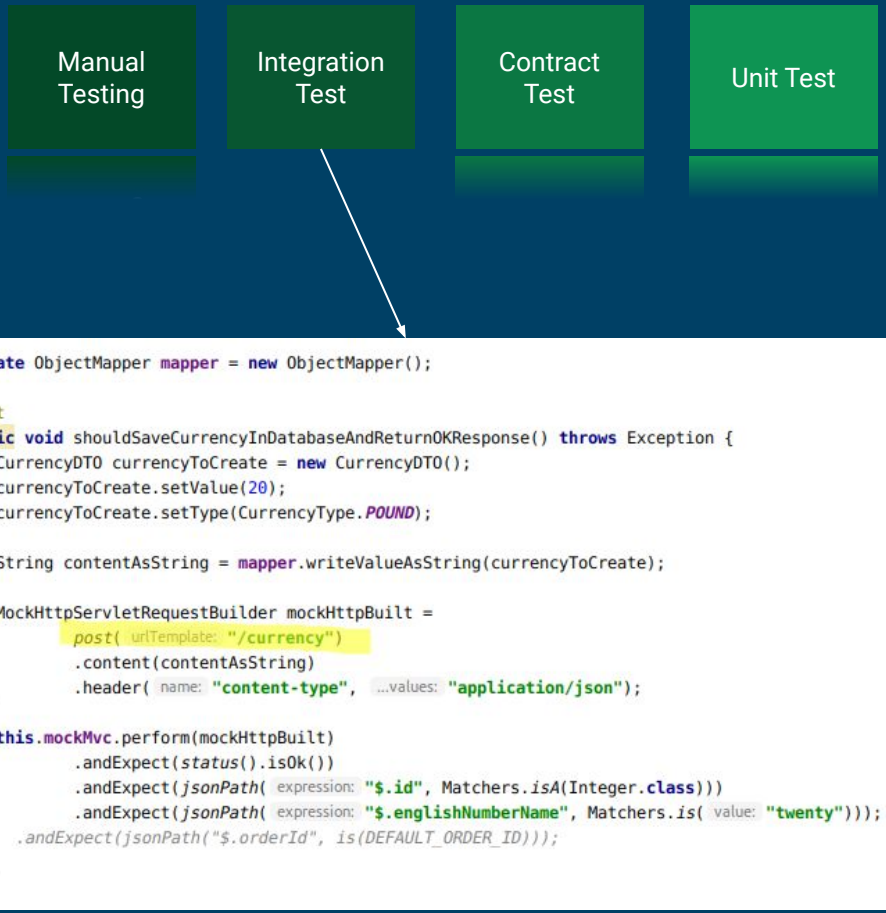
"Ideal World"

...



TDD - Where is everything?

"Ideal World"



TDD - Where is everything?

"Ideal World"

Manual Testing | Integration Test | Contract Test | Unit Test

```
mycurl() {
    response=$(
      curl \
      --write-out %{http_code} \
      --output /dev/null \
      --silent \
      --location --request POST 'http://192.168.99.101:31359/shipments/find' \
      --header 'Content-Type: application/json' \
      --header 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6I
kYTc2NWMtYzBmNS00OWE3LTg5ZjgtNzBlNDUwZjg1ZmQzL3YyLjAvIiwiZXhwIjoxNTc5ODYzNTY3LCJuYm
4NjcyLTJlMGEtNGUwOS1iOWFhLTU4M2Q3ZjhiODkzNCIsInN1YiI6IjQyZWE4NjcyLTJlMGEtNGUwOS1iOW
FVVJTT05NTkM7UkhPRk9PX1pBIiwiZXh0ZW5zaW9uX3JvbGVzIjoiVVNFUiIsImVtYWlscyI6WyJodmtvZX
1VzdLb3JUVEJhIiwiYXpwIjoiYzI4NTE4NDEtNThmYS00MjdiLTk1ZjYtYjE2MTk3MDUyMDM5IiwidmVyIj
ujxy9GdnKsSmpvQ8E_9apoE6NNwvlrO9YgsW13NSJ1Uy5S8NDmZ7oIPbUfC6uwPfZjtJM3c07MiM_rRr6N6
Llz6KHwso3EoccZeMe5GLYcahInrZWqcLmOE0bB0ZRaIZvRyUQ' \
      --data-raw '

      {
        "pagination": {
          "size": 5,
          "page": 0
        },

         "organizationCodes": ["DEV_TEST"],

          "transportMode" : null,

          "query": "",

          "startDepartureDate": null,
          "endDepartureDate": null,
          "startArrivalDate": null,
          "endArrivalDate": null

      }'
    )
    echo "$1 = $response"
    if [ "$response" != "200" ]
    then
        # invalid, so we can echo the response and get the value from it
        echo "$1 = $response"
        echo "invalid"
    fi
}
export -f mycurl

i=0

numTimes=100

echo "Running curl for $numTimes times in parralel"
seq $numTimes | parallel -j0 mycurl
```

... | ... | ... | ...

TDD - Benefits

- More code is tested
- More modular - Single Responsibility Role
- Motivational = Pass own tests
- Narrow mental bandwidth
- Leads to SOLID Design Principles:
    - S - Single-responsibility principle
    - O - Open-closed principle
    - L - Liskov substitution principle
    - I - Interface segregation principle
    - D - Dependency Inversion Principle

TDD

Any questions?

Further Reading

- https://stackabuse.com/test-driven-development-for-spring-boot-apis/
- https://medium.com/@tdeniffel/advantages-of-test-first-over-test-after-by-geepaw-hill-c66b4d31d280
- https://www.baeldung.com/spring-boot-testing