luv2code

Set Up SQL Scripts in properties file

SQL for Sample Data

- Currently the SQL for sample data is hard-coded in our tests
- We would like to move the SQL to our properties file

@BeforeEach and @AfterEach

StudentAndGradeServiceTest.java

```
import org.springframework.jdbc.core.JdbcTemplate;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
...

@TestPropertySource("/application.properties")
@SpringBootTest
public class StudentAndGradeServiceTest {

   @Autowired
   private JdbcTemplate jdbc;

   @BeforeEach
   public void setupDatabase() {
       jdbc.execute("insert into student(firstname, lastname, email_address) " +
               "values ('Eric', 'Roby', 'eric.roby@luv2code_school.com')");
       ...
   }


   @AfterEach
   public void setupAfterTransaction() {
       jdbc.execute("DELETE FROM student");
       jdbc.execute("ALTER TABLE student ALTER COLUMN ID RESTART WITH 1");
       ...
   }

}
```

This is what we currently have

Insert sample data

Delete sample data

SQL is hard coded

Development Process

1. Add SQL to application.properties
2. Inject SQL into test using @Value
3. Refactor @BeforeEach and @AfterEach

Step 1: Add SQL to application.properties

application.properties

```
...

sql.script.create.student=insert into student(firstname,lastname,email_address) \
  values ('Eric', 'Roby', 'eric.roby@luv2code_school.com')

sql.script.delete.student=DELETE FROM student; ALTER TABLE student ALTER COLUMN ID RESTART WITH 1

...
```

Step 2: Inject SQL into test using @Value

StudentAndGradeServiceTest.java

```
...

@TestPropertySource("/application.properties")
@SpringBootTest
public class StudentAndGradeServiceTest {

   @Autowired
   private JdbcTemplate jdbc;

   @Value("${sql.script.create.student}")
   private String sqlAddStudent;

   @Value("${sql.script.delete.student}")
   private String sqlDeleteStudent;

   ...
}
```

application.properties

```
...

sql.script.create.student=insert into student(firstname,lastname,email_address) \
  values ('Eric', 'Roby', 'eric.roby@luv2code_school.com')

sql.script.delete.student=DELETE FROM student; ALTER TABLE student ALTER COLUMN ID RESTART WITH 1

...
```

Step 3: Refactor @BeforeEach and @AfterEach

StudentAndGradeServiceTest.java

```
...

@TestPropertySource("/application.properties")
@SpringBootTest
public class StudentAndGradeServiceTest {

    @Autowired
    private JdbcTemplate jdbc;

    @Value("${sql.script.create.student}")
    private String sqlAddStudent;

    @Value("${sql.script.delete.student}")
    private String sqlDeleteStudent;

    @BeforeEach
    public void setupDatabase() {
        jdbc.execute(sqlAddStudent);
    }


    @AfterEach
    public void setupAfterTransaction() {
        jdbc.execute(sqlDeleteStudent);
    }

}
```

Refactored code

Refactored code

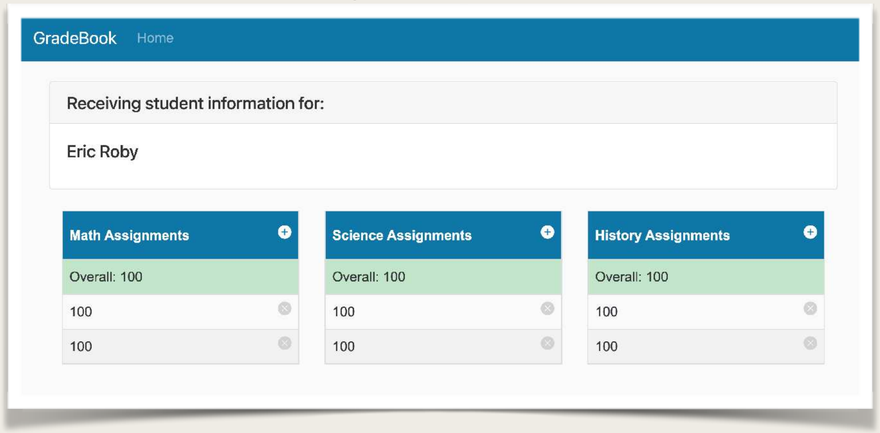
MVC Tests for Student Information

MVC Tests for Student Information

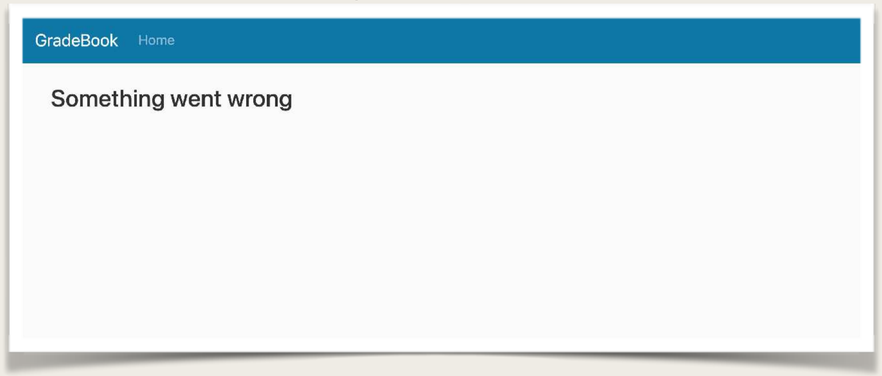
- If we have valid student id, return view name: `studentInformation`
- If we have invalid student id, return view name: `error`

Thymeleaf templates

File: src/main/resources/templates/studentInformation.html



File: src/main/resources/templates/error.html

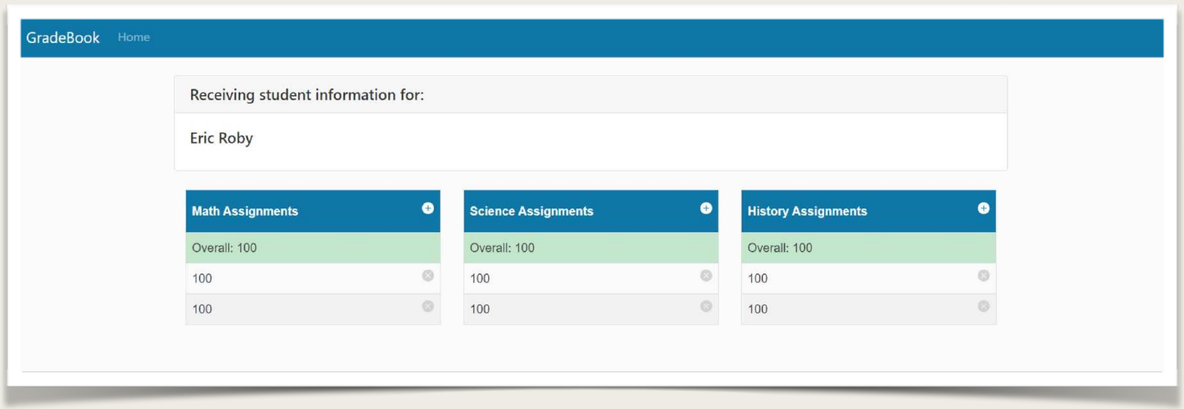


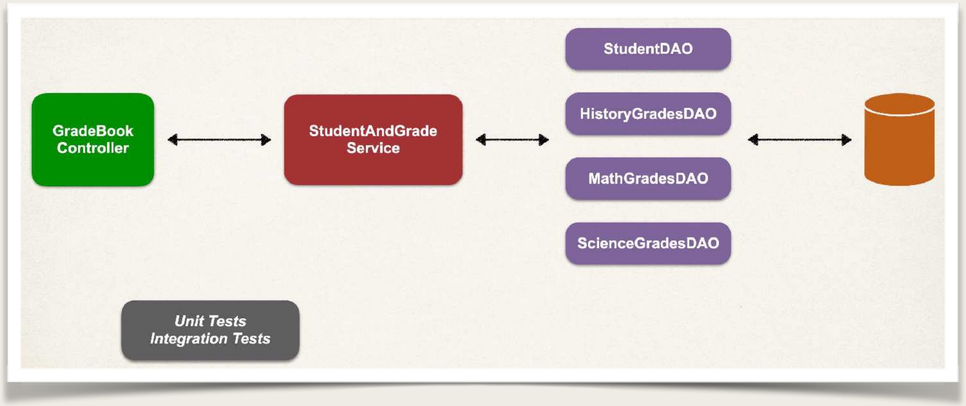
luv2code

Create Grades with MVC Controller

To Do: Create Grades

- Currently the app does not support creating new grades via MVC controller
- Apply TDD to add support for this functionality

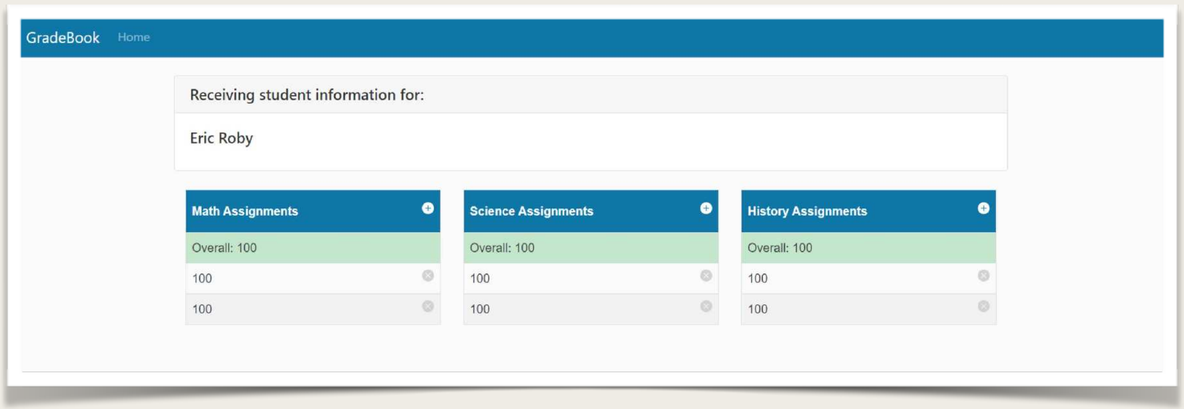


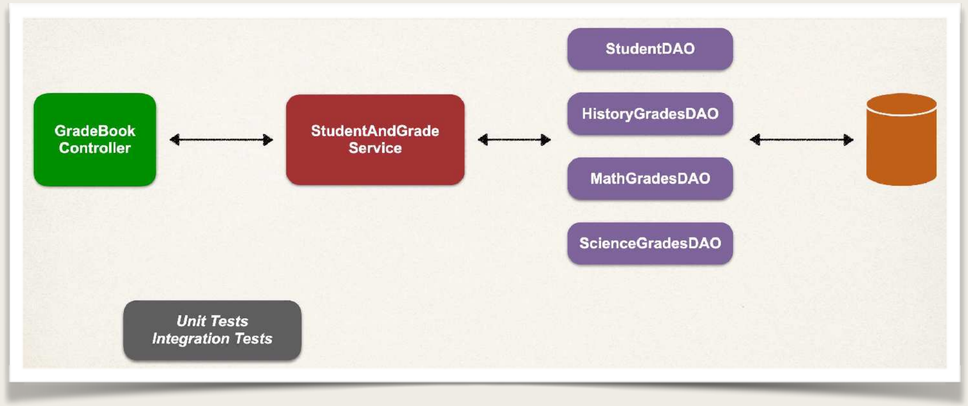


Delete Grades with MVC Controller

To Do: Delete Grades

- Currently the app does not support deleting grades via MVC controller
- Apply TDD to add support for this functionality





luv2code

Update Thymeleaf Template
Student Information

To Do

- Currently the UI for Student Information has hard-coded data ✗
- Update the Thymeleaf template for Student Information to use dynamic data ✓

Thymeleaf template

File: src/main/resources/templates/studentInformation.html