Proof of Technology

The proof of concept for the web application is a Spring Boot project written in Java.

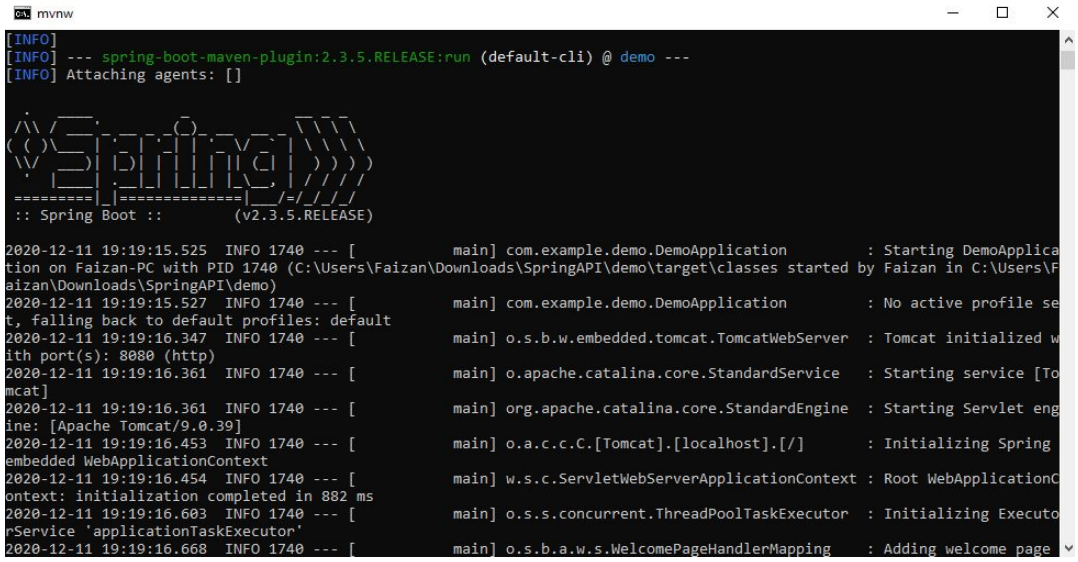
Goals

Team Delta set out to achieve the following goals in the Spring Boot project to prove that the Web Application was feasible:

1. Calling data from a database from the web application.
2. Displaying the retrieved data in JSON format.
3. Adding data to the database.

Running Instructions

1. Please ensure that you have installed Java and JDK on your system and the environment variables are set.
2. Open your terminal and `cd` into the 'SpringAPI' folder and then `cd` in to the 'demo' folder.
3. Run the command `mvnw spring-boot:run` if you are using Windows or `./mvnw spring-boot:run` on MacOS/Linux then wait for the web application to run. You should see something like this to indicate that the Spring Boot project is running:



4. Open your web browser and head to `localhost:8080`

List of accepted URL paths

1. GET /index
2. GET /greeting
3. GET /greeting/all
4. POST /greeting
    a. This path is used to add data to the test database, so this path accepts two parameters 'ColA' (String) and 'ColB' (integer)
5. GET /{ID}

Goal 1 & Goal 2: Calling Data from the Database and Displaying as JSON

As stated in the Technical Report, the database is externally hosted by FreeMySQLHosting. The database only has one table called 'random' and this table has three columns called: ColA (string), ColB (int) and ID (primary key).

←T→			ColA	ColB	id
Edit	Copy	Delete	Can you see this	2	1
Edit	Copy	Delete	Just anything	123	2
Edit	Copy	Delete	Python test script	445	3
Edit	Copy	Delete	Double checking	555	4
Edit	Copy	Delete	Showing JP	159	5

Database table 'random' shown in PhpMyAdmin

To call data from the database the application needs to connect to it first. This is achieved using JDBC.

```
public Random(){
    try{

        this.con = DriverManager.getConnection( url: "jdbc:mysql://sql2.freemysqlhosting.net:3306/sql2375559",  user: "sql2375559",  passwor

    }catch(Exception e){
        System.out.println(e);
    }
}
```

You can call all the data in the database by clicking the 'View Table' button or by typing in localhost:8080/greeting/all into the address bar which should return the below results.

localhost:8080/greeting/all

[{"id":1,"colA":"Can you see this","colB":2},{"id":2,"colA":"Just anything","colB":123},{"id":3,"colA":"Python test script","colB":445},{"id":4,"colA":"Double checking","colB":555},{"id":5,"colA":"Showing JP","colB":159}]

Below is how this data is retrieved and displayed:

```
public List getAll(){
    try {
        stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery( sql: "select * from random");

        List<RandomDB> rsreturn = new ArrayList<>();

        while (rs.next()){

            RandomDB rand = new RandomDB(rs.getString( columnIndex: 1), rs.getInt( columnIndex: 2), rs.getInt( columnIndex: 3));
            rsreturn.add(rand);

        }

        return rsreturn;
    }catch(Exception e){
        ResultSet rs = null;
        return null;
    }
}
```

The method getAll() in the 'Random' model executes a query on the 'random' table in the database that selects all of content in the table and stores the result of the executed query as a ResultSet. The ResultSet is iterated through to store each of its items as a custom class, RandomDB and these items are put into a list that gets returned to the main program.

```
@GetMapping("/greeting/all")
public List<RandomDB> greetingAll(){
    // Get data from database
    Random rnd = new Random();
    // Return data
    return rnd.getAll();
}
```

The URL /greeting/all maps to the greetingAll method in the GreetingController class that calls and returns the getAll() method in the 'Random' model. The GreetingController class is declared as Spring Boot's @RestController which effectively means that it generates whatever gets returned to it as JSON.

```java
public RandomDB getRow(int ID){
    try {
        this.stmt = con.createStatement();
        String query = String.format("select * from random where random.id = %2d", ID);
        ResultSet rs = stmt.executeQuery(query);
        rs.next();
        RandomDB rand = new RandomDB(rs.getString( columnIndex: 1), rs.getInt( columnIndex: 2), rs.getInt( columnIndex: 3));
        return rand;

    }catch(Exception e){
        System.out.println(e);
        return null;
    }
}
```

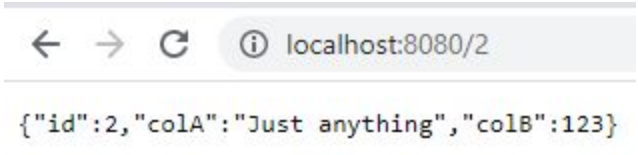
The getRow(int ID) is a method in the 'Random' model that works similarly to get the getAll() method but just retrieves a single row by ID instead of all rows.

```java
@GetMapping("/{ID}")
public RandomDB getRow(@PathVariable(value="ID") int ID){
    // Convert string ID to int
    try{

        Random rnd = new Random();
        return rnd.getRow(ID);

    }catch(Exception e){
        return null;
    }
}
```

The example below shows that http://localhost:8080/2 returns a row from the table (ID, String, int) in the database with ID 2:



Goal 3: Adding Data to the Database

Data can be added to the database through the web application using a Python script. The script, requestTester.py, sends a POST request to the locally hosted web application which is then handled by the greetingInsert method in the GreetingController class as shown below:

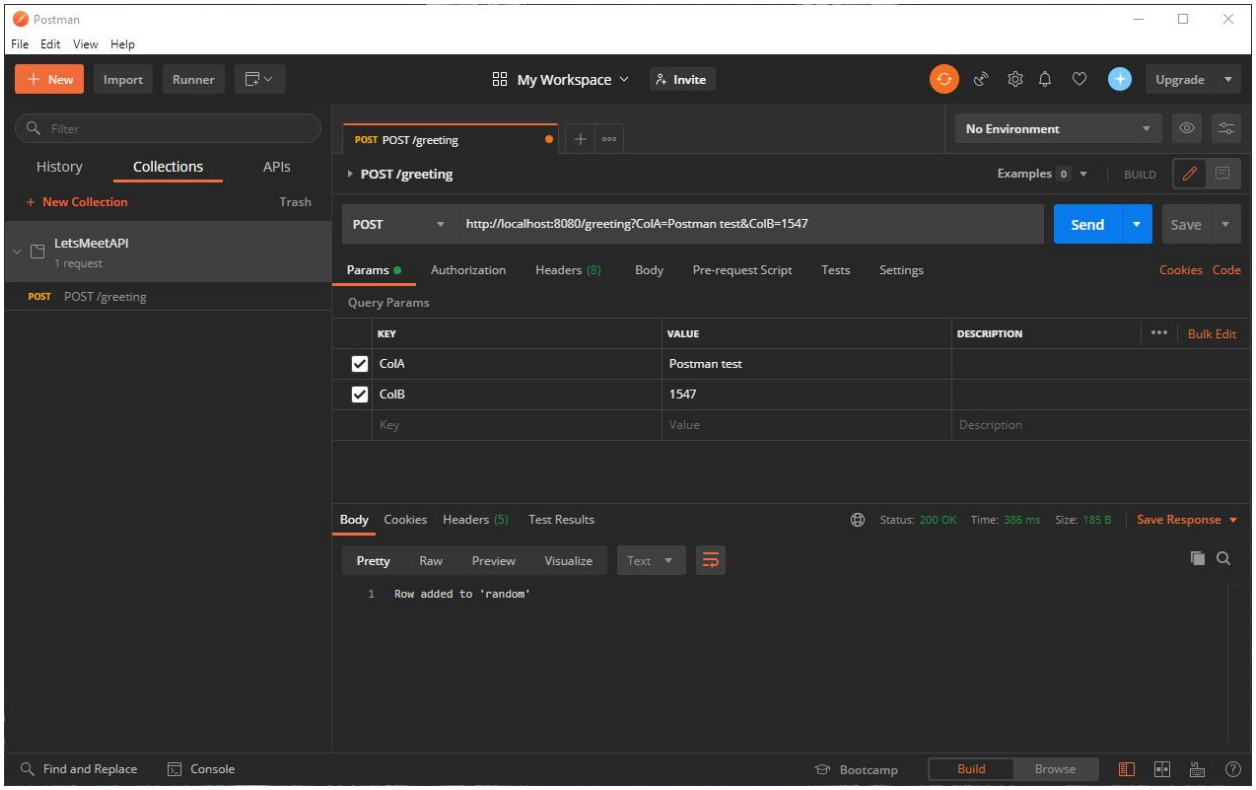
```
@PostMapping("/greeting")
public String greetingInsert(@RequestParam(value="ColA", defaultValue= "") String ColA, @RequestParam(value="ColB", defaultValue = "") int ColB){
    // Add row to DB
    Random rnd = new Random();
    return rnd.insert(ColA, ColB);
}
```

This method calls the insert(ColA, ColB) method from the 'Random' model:

```
public String insert(String ColA, int ColB){
    // Insert into the DB
    try{
        this.prestmt = this.con.prepareStatement(this.insert_sql);
        this.prestmt.setString( parameterIndex: 1, ColA);
        this.prestmt.setInt( parameterIndex: 2, ColB);

        int rows = this.prestmt.executeUpdate();
        if(rows > 0){
            return "Row added to 'random'";
        }else{
            throw new Exception("Nothing added to DB");
        }
    }catch(Exception e){
        System.out.println(e);
        e.printStackTrace();
        return "An error occured, nothing has been added to 'random' table";
    }
}
```

Below you can see how data can be added to the database through sending a POST request to the API from the Postman application. Alternatively, a POST request can be sent to the API from any similar application or library such as the 'requestTester.py' file provided.


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
New Collection
Trash
LetsMeetAPI
1 request
POST POST /greeting
POST POST /greeting
No Environment
POST /greeting
Examples 0
BUILD
POST
http://localhost:8080/greeting?ColA=Postman test&ColB=1547
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
ColA
Postman test
ColB
1547
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 386 ms
Size: 185 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1 Row added to 'random'
Find and Replace
Console
Bootcamp
Build
Browse