Read ME

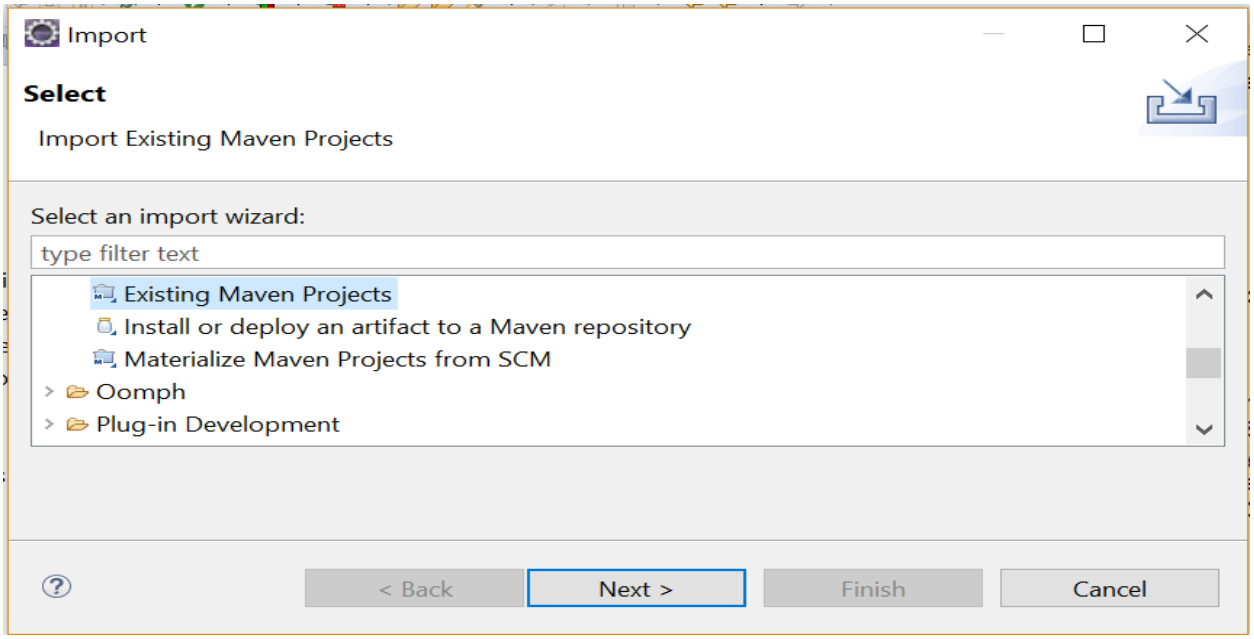
prerequisites

- Maven 3.0+
- IDE Eclipse.
- JDK 1.8+

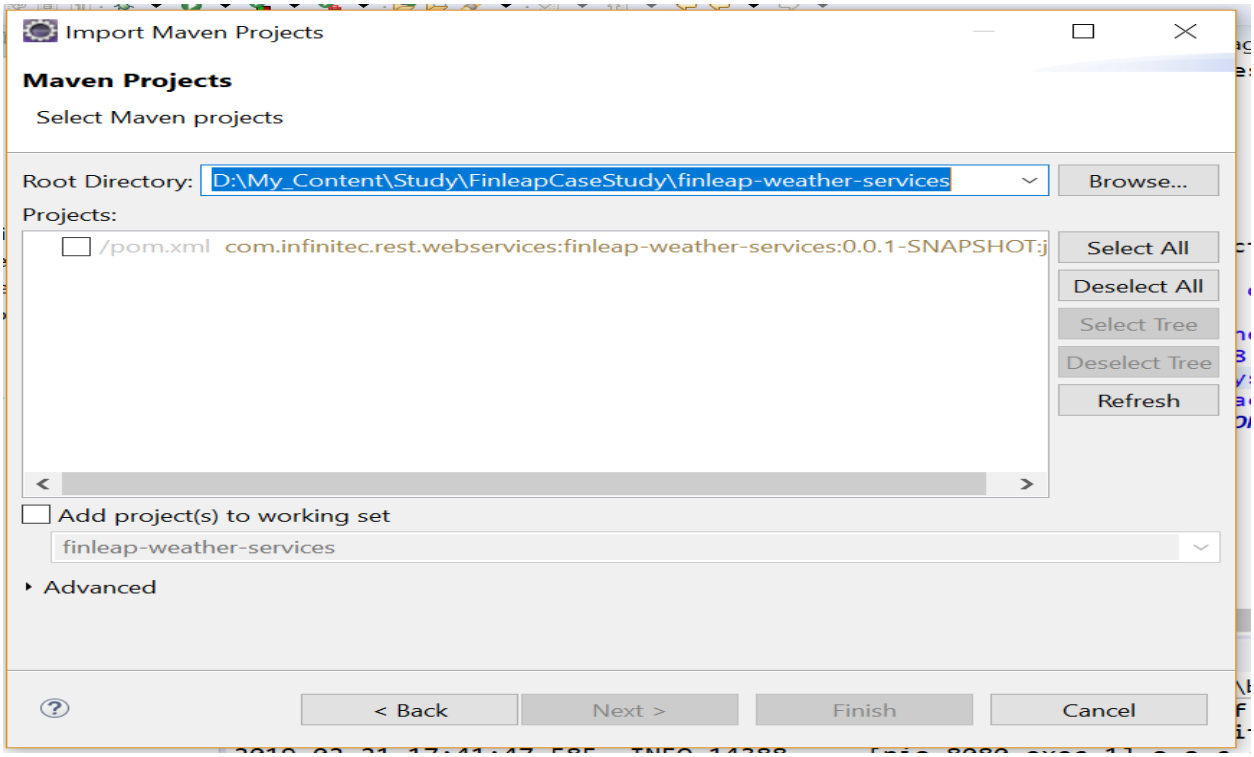
Setting up the project

1) Extract the zip finleap-weather-services.zip.

2) Import the project in eclipse by selecting existing maven projects from the wizard.

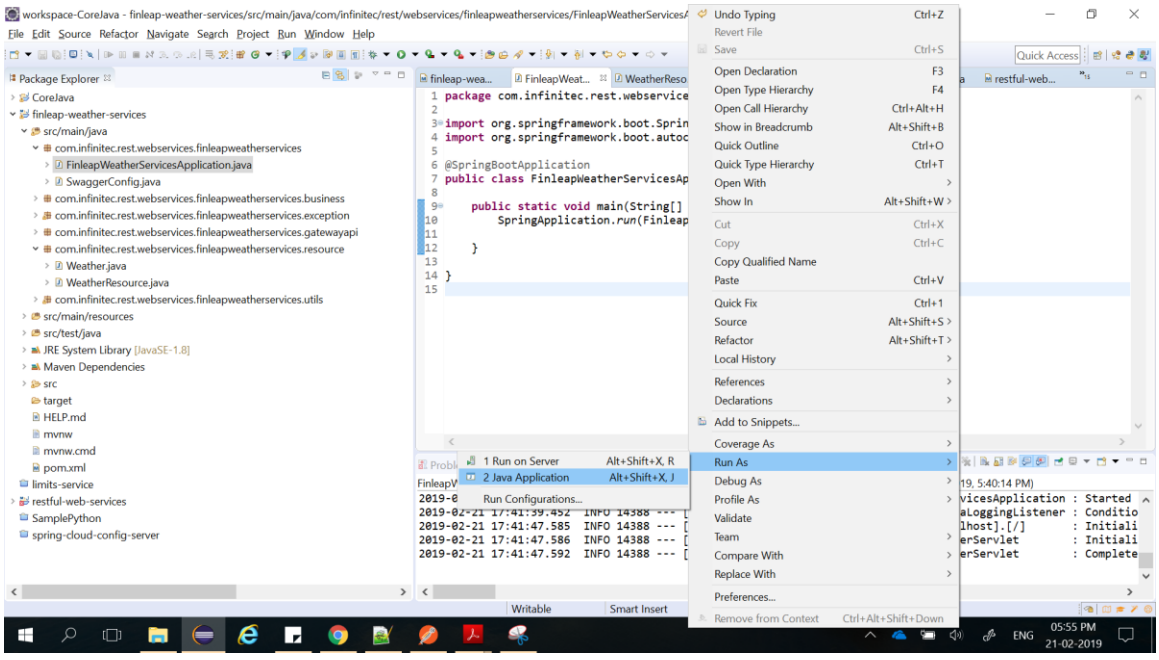


3) Browse for the project, select pom.xml file.



4) It takes some time to download the dependencies and build the project.

5) Running application:



- Run this class as shown in the figure.
- The tomcat server should have started running on port 8080.

Few insights of application startup:

- Dependencies such as

  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  </dependency>

- Build restful web services
- To develop web applications
- Easy for developing applications
  #application.properties
  logging.level.org.springframework=debug

  AUTO CONFIGURATION REPORT

  DISPATCHER SERVLET AUTO CONFIGURATION

  Springboot starter-web has a dependency of spring-web-mvc framework which configure dispatcher servlet.
  Dispatcher servlet is root of the all requests for the web application,it acts like front controller.
  Request first sent to dispatcher servlet which knows the mapping of entire application, inturn will navigate the request to the controller's method.

Error MVC Auto configuration

Error messages on web responses is achieved with the help of HttpMessage Converters & JacksonAutoConfiguration

Apart from above mentioned many other configurations are also loaded

Code level Explanation of project:

- **This package com.infinitec.rest.webservices.finleapweatherservices consists of two classes**
    1. FinleapWeatherServicesApplication
        - this class is required for starting up the applications
    2. SwaggerConfig
        - this class is required for the documentation of RestAPI's by using the following dependencies

```
<dependency>
    <groupId>io.springfox</
    <artifactId>springfox-
    <version>2.4.0</version>
    </dependency>
    <dependency>
    <groupId>io.springfox</
    <artifactId>springfox-swagger-
    <version>2.4.0</version>
    </dependency>
```

Exception handling
com.infinitec.rest.webservices.finleapweatherservices.exception in this package

- **CustomizedResponseEntityExceptionHandler** will centralize your error handling logic in spring by using the @ControllerAdvice annotation Reduce duplicate code and keep your code clean.
- **CityNotFoundException** custom exception class which has HttpStatus Code.
- **ExceptionResponse** class helps to format any exception.

UTILS

Com.infinitec.rest.webservices.finleapweatherservices.utils

This package consists of constants and utilities required for this use case such as RestServiceConstants and RestServicesUtils

RESTCONTROLLER

com.infinitec.rest.webservices.finleapweatherservices.resource

- Weather: this class is represented as JSON object which contains the Average temperature (in Celsius) of the next 3 days from today's date for Day Time (06:00 – 18:00) and Night time (18:00 – 06:00) Average of pressure for the next 3 days from today's date.

- Weather Resource: This is responsible for the rest service by using retriveWeather with @GetMapping ("/data/{cityName}").

- Here we are doing a **validation check** of cityname which should contain only characters (alphabets) in turn throwing **CityNotFoundException** if it doesn't match.

OpenWeatherMAP API which helps to retrieve the weather forecast for 5 days for 3 hours basics for a particular city using POSTMAN.

1. **APPID=2f7bd340037cb3f507e78b8dbeaa1d68** is generated by registering to the website.

2. http://api.openweathermap.org/data/2.5/forecast?q=london&**APPID=2f7bd340037cb3f507e78b8dbeaa1d68**

Response:


```
{
        "dt": 1550750400,
        "main": {
           "temp": 287.32,
           "temp_min": 285.967,
           "temp_max": 287.32,
           "pressure": 1031.69,
```

```
            "sea_level": 1039.39,
            "grnd_level": 1031.69,
            "humidity": 70,
            "temp_kf": 1.35
        },
        "weather": [
            {
                "id": 500,
                "main": "Rain",
                "description": "light rain",
                "icon": "10d"
            }
        ],
        "clouds": {
            "all": 0
        },
        "wind": {
            "speed": 3.66,
            "deg": 234
        },
        "rain": {
            "3h": 0.02
        },
        "sys": {
            "pod": "d"
        },
        "dt_txt": "2019-02-21 12:00:00"
    }
```


As mentioned above it contains the weather forecast for 5 days for a period of three hours for a city. This just the sample response which is mentioned above

Invoking the Rest API of OpenWeatherMap through SpringBoot

com.infinitec.rest.webservices.finleapweatherservices.gatewayapi

OpenWeatherMapGatewayImpl class will invoke the response by using these lines.

- RestTemplate restTemplate = new RestTemplate();

- OpenWeatherResponse responseObject = restTemplate.getForObject(RestServiceConstants.OPENWEATHERURL + cityNameOpenWeatherResponse.class);
  Response from rest service is mapped by using these two classes OpenWeatherResponse and WeatherTemplate.

Filtering & Calculating

com.infinitec.rest.webservices.finleapweatherservices.business

Step 1:
Filtered the response only for 3 days as it gives the data for 5 days by using streams and Predicate<WeatherTemplate> isBefore()

Step 2:
Predicate<WeatherTemplate> isDaytime () will filter the day time data dayTime = { "06:00:00", "09:00:00", "12:00:00", "15:00:00" };

BiFunction<List<WeatherTemplate>, Predicate<WeatherTemplate>, Double>

Which will return the average by taking two inputs the list and predicate gives back the average of day time temperature for a period of 3 days.

Step 3:
Predicate<WeatherTemplate> isDaytime ().negate () will filter the night time data.

BiFunction<List<WeatherTemplate>, Predicate<WeatherTemplate>, Double>

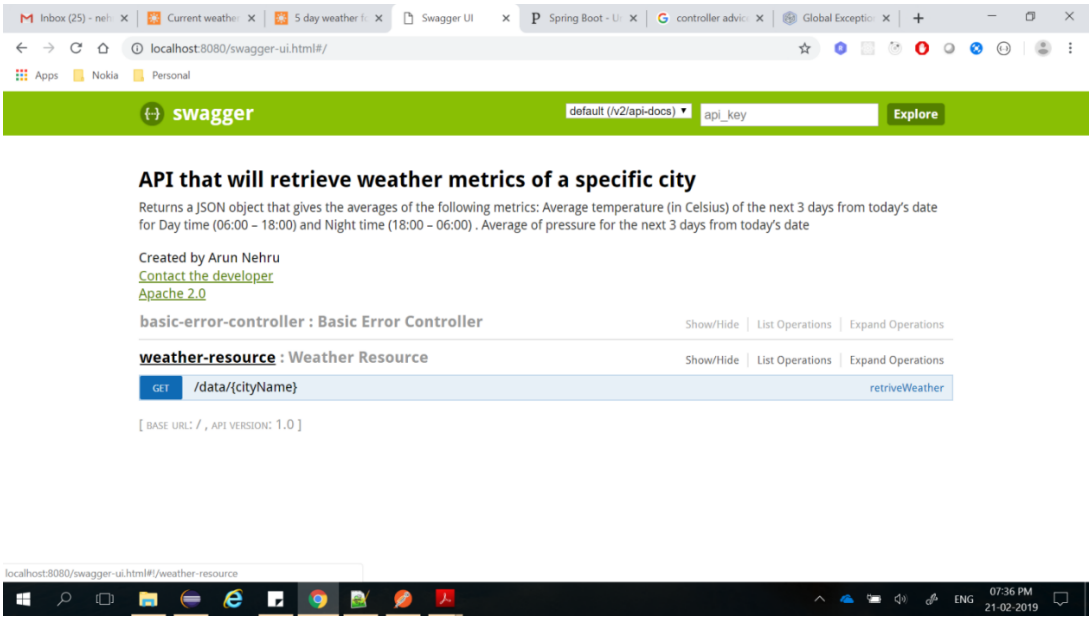
Which will return the average by taking two inputs the list and predicate with negate gives back the average of night time temperature for a period of 3 days.

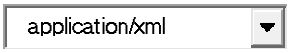
Step 4:
CalculateCelsius (double kelvin) by converting the kelvin to Celsius.

Step 5:
Initializing the object by assigning the calculated values for the response.

SwaggerUI:

- As mentioned above run the FinleapWeatherServicesApplication as java application.
- Access swagger ui by the link mentioned below
- http://localhost:8080/swagger-ui.html
- Then click on weather-resource as shown below.



- Click on /data/ {cityName} to see the request and response object.
- Response Content Type 

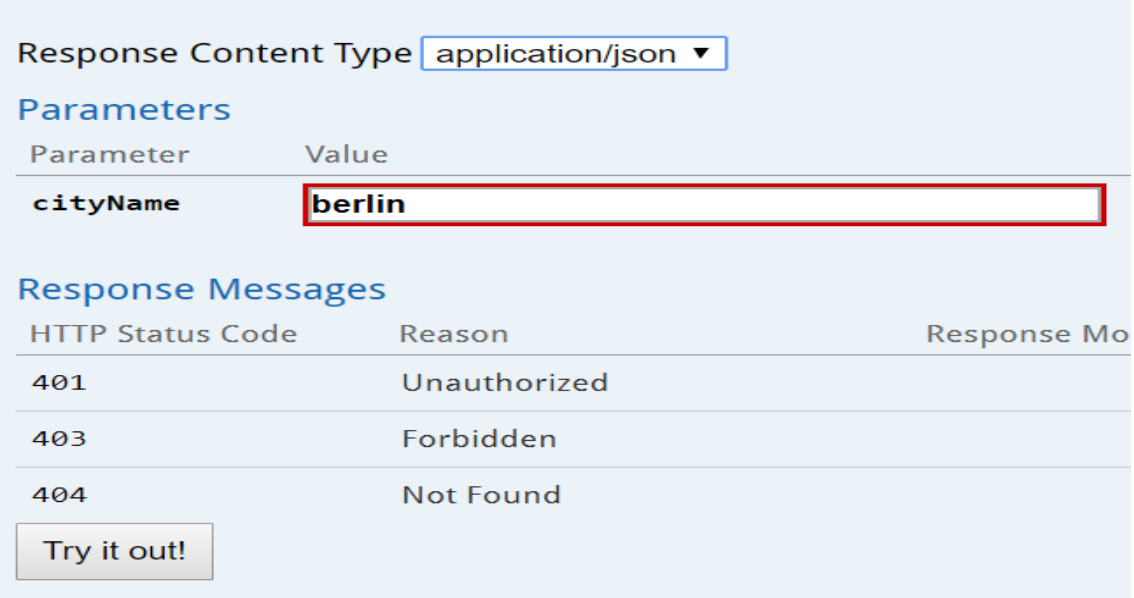
can be chosen as application/xml or application/json
- This conversion from json to xml or xml to json is handled by the dependency mentioned below.

```
<dependency>
<groupId>com.fasterxml.jackson.dataformat</groupId>
<artifactId>jackson-dataformat-xml</artifactId>
```

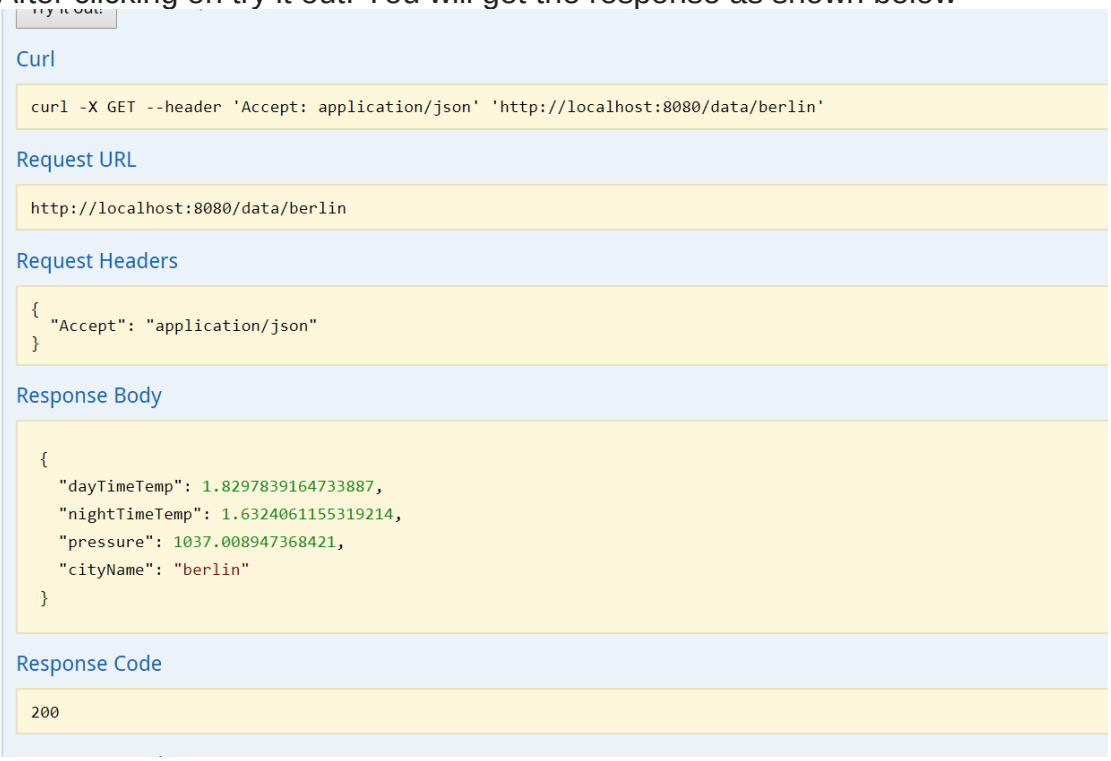
</dependency>

cityName

Enter the city name where you would like to see the average of temperature and pressure for 3 days.
After clicking on try it out! You will get the response as shown below



After clicking on try it out! You will get the response as shown below



Monitor Spring APIs using Spring Boot Actuator & HAL Browser:

- Dependencies required are mentioned below

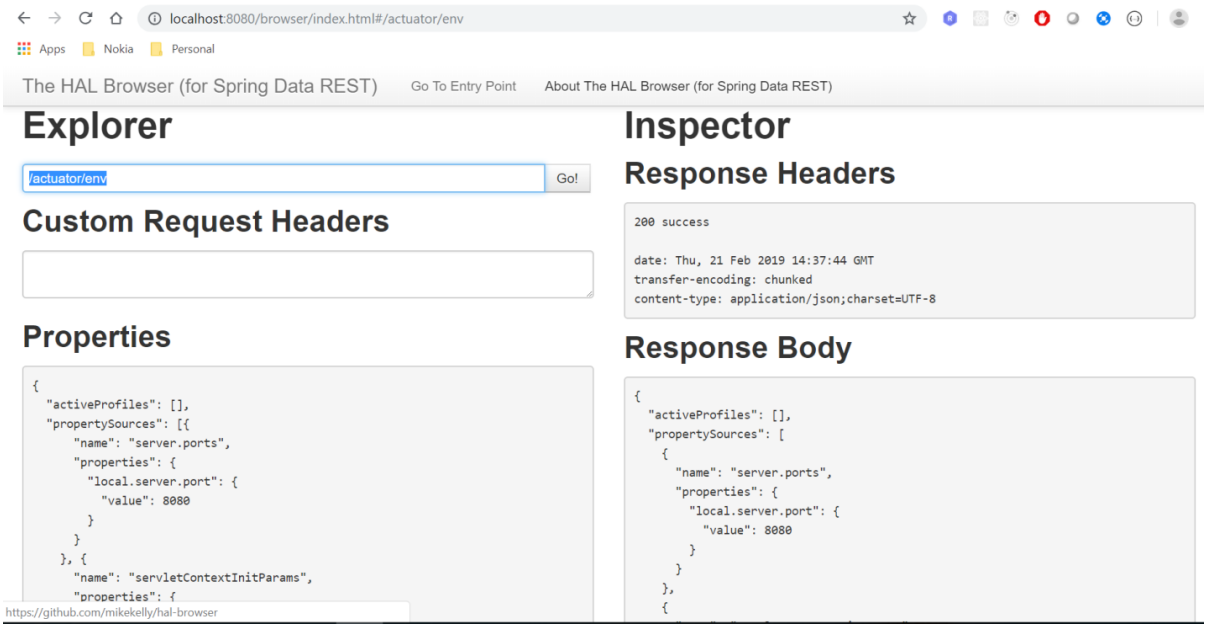
  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>
  <dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-rest-hal-browser</artifactId>
  </dependency>

- configure application.properties
  management.endpoints.web.exposure.include=*

  http://localhost:8080/browser/index.html

- This will bring up HAL browser to monitor the api in better way

- For example: /actuator/env which gives the insight of environment details.



Unit Test

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
</dependency>
```

The important dependency for unit testing is `spring-boot-starter-test`

*Spring Boot Test Starter is Starter for testing Spring Boot applications with libraries including **JUnit, Hamcrest and Mockito***

- `@RunWith(SpringRunner.class)` - Spring Runner is used to launch up a spring context in unit tests.
- `@SpringBootTest` - This annotation indicates that the context under test is a `@SpringBootApplication`. The complete SpringBootTutorialBasicsApplication is launched up during the unit test.
- `@MockBean OpenWeatherMapGateway` - @MockBean annotation creates a mock for openWeatherMapGateway. This mock is used in the Spring Context instead of the real Service.
- `@Autowired ForecastWeather` - Pick the ForecastWeather Service from the Spring Context and autowire it in.