BridgeLabz

Employability Delivered

Java Web API Introduction and Testing

Agenda

BridgeLabz
Employability Delivered

- Firstly we will start with Installation of **Node.js** Server which will also install **Node Package Manager (npm)**. This is need to create Web Server
- Using npm we will download **JSON Server**. This helps in mocking REST API which is needed for API Testing.
- We would test JSON Server API from the Terminal using **CURL**
- Also Test the JSON Server API using **Postman**
- Use Hamcrest on top Junit, this allows Condition matching using Matcher Class
- Finally we will use **REST Assured** to do API Testing with JSON Server

NodeJs – CLI (Command Line Interface) Installation

- **Step 1:** Check Homebrew Package Manager is installed using
  - ***brew –version*** - in the Terminal of Linux/Mac or in the Command Line of Windows
- **Step 2:** If not, Installation of Homebrew Package Manager
  - **For Windows –** Download Windows Subsystem for Linus using link https://docs.microsoft.com/en-us/windows/wsl/about
  - **For Linux & Windows -** https://docs.brew.sh/Homebrew-on-Linux
  - **For Mac -** https://docs.brew.sh/Installation
- **Step 3:** ***brew –version*** in command line to check for successful installation
- **Step 4:** ***brew –update*** – This updates Homebrew with a list of the latest version of Node.
- **Step 5:** ***brew install node*** **–** This will install node and npm
- **Step 6:** ***node -v*** **&** ***npm -v*** **–** This will check the installation of node and npm
- Refer https://treehouse.github.io/installation-guides/mac/node-mac.html for installation notes for Mac, Linux and Windows with Homebrew

What is Node.js and npm

- Node.js® is an environment which you can uses for compiling and running JavaScript code in command line and more importantly to create web-servers and networked applications.
- NPM is a “package manager” that makes installing Node “packages” fast and easy.

What is JSON Server

- JSON Server is a Node Module that you can use to create demo REST json webservice in less than a minute. All you need is a JSON file for sample data.
- Installing JSON Server.

```
npm install -g json-server
```

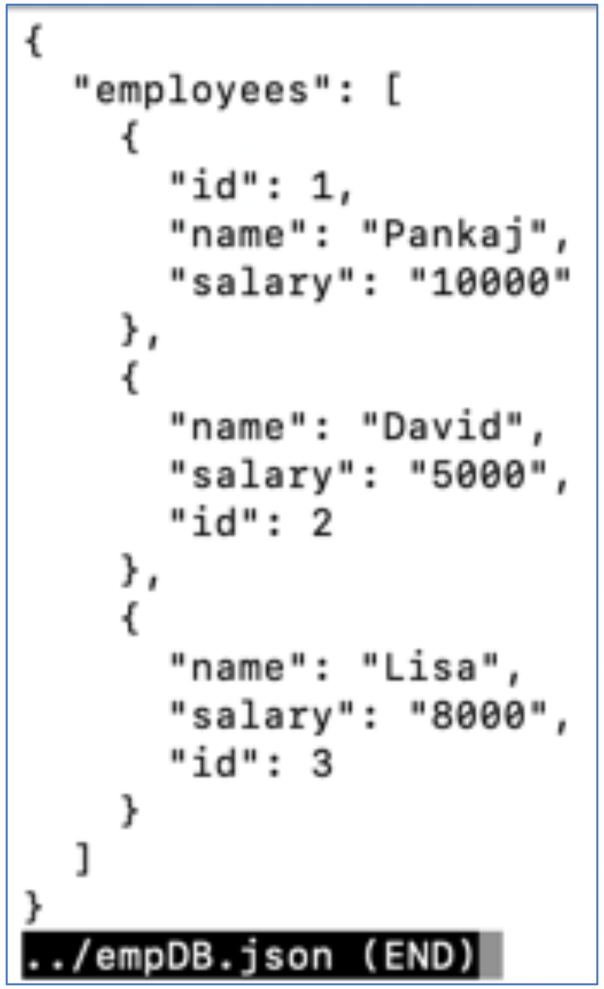
- Check JSON Server version

```
json-server -v
```

Run JSON Server

- **Step 1:** Create empDB.json file.
- **Step 2:** Run JSON Server. This will create REST API for empDB

  json-server --watch empDB.json
- REST API are essentially for i.e. CURD operation - Create, Read, Update and Delete
- Next steps we would call the json server to Get all Employees, to Add Employee and finally to Update and Delete Employees



```
{
  "employees": [
    {
      "id": 1,
      "name": "Pankaj",
      "salary": "10000"
    },
    {
      "name": "David",
      "salary": "5000",
      "id": 2
    },
    {
      "name": "Lisa",
      "salary": "8000",
      "id": 3
    }
  ]
}
../empDB.json (END)
```

REST API using JSON Server from Terminal

BridgeLabz
Employability Delivered

- **Step 3:** Get all Employees using curl command

HTTP Method Call

Content Type: json

Getting all Employees

```
curl -X GET -H "Content-Type: application/json"  "http://localhost:3000/employees"
```

HTTP Header

Calling JSON Server running in the local machine in the port number 30000

- **Step 4:** Get Employee Details with id 1

```
curl -X GET -H "Content-Type: application/json"  "http://localhost:3000/employees/1"
```

REST API using JSON Server from Terminal

BridgeLabz
Employability Delivered

- **Step 5:** Adding New Employee Lisa. Here you will notice HTTP Method POST is used as well as using –d option JSON is passed in the Body

HTTP Method - Post

Employee JSON in the Body

```
curl -X POST -H "Content-Type: application/json" -d '{"name": "Lisa","salary": "2000"}'
"http://localhost:3000/employees"
```

Calling JSON Server running in the local machine in the port number 30000

- **Step 6:** Update Employee Salary for Lisa to 8000 for id 3

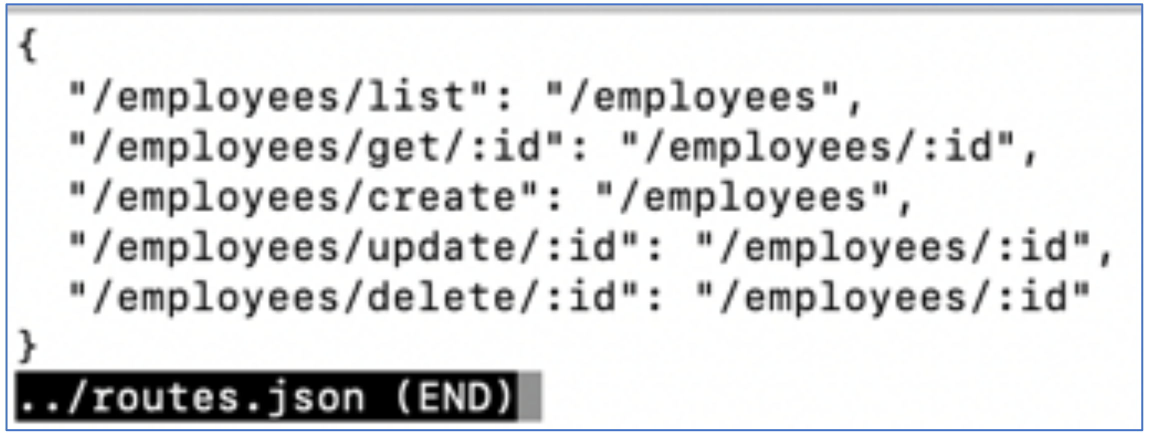
```
curl -XPUT -H "Content-Type: application/json" -d '{"name": "Lisa", "salary": "8000"}'
"http://localhost:3000/employees/3"
```

Custom Routes for REST API

- **Step 7:** Delete Employee with Employee Id 2

curl -X DELETE -H "Content-Type: application/json" "http://localhost:3000/employees/2"

- **Step 8:** Create Custom Routes for CURD REST APIs



```
{
  "/employees/list": "/employees",
  "/employees/get/:id": "/employees/:id",
  "/employees/create": "/employees",
  "/employees/update/:id": "/employees/:id",
  "/employees/delete/:id": "/employees/:id"
}
../routes.json (END)
```

- **Step 8:** Starting JSON Server with new Routes and Port

json-server --port 4000 --routes routes.json --watch empDB.json

Custom REST API

BridgeLabz
Employability Delivered

```
curl -X GET -H "Content-Type: application/json" "http://localhost:4000/employees/list"
```

```
curl -X GET -H "Content-Type: application/json" "http://localhost:4000/employees/get/1"
```

```
curl -X POST -H "Content-Type: application/json" -d '{"name": "Lisa","salary": "2000"}'
"http://localhost 4000/employees/create"
```

```
curl -XPUT -H "Content-Type: application/json" -d '{"name": "Lisa", "salary": "8000"}'
"http://localhost:4000/emloyees/update/4"
```

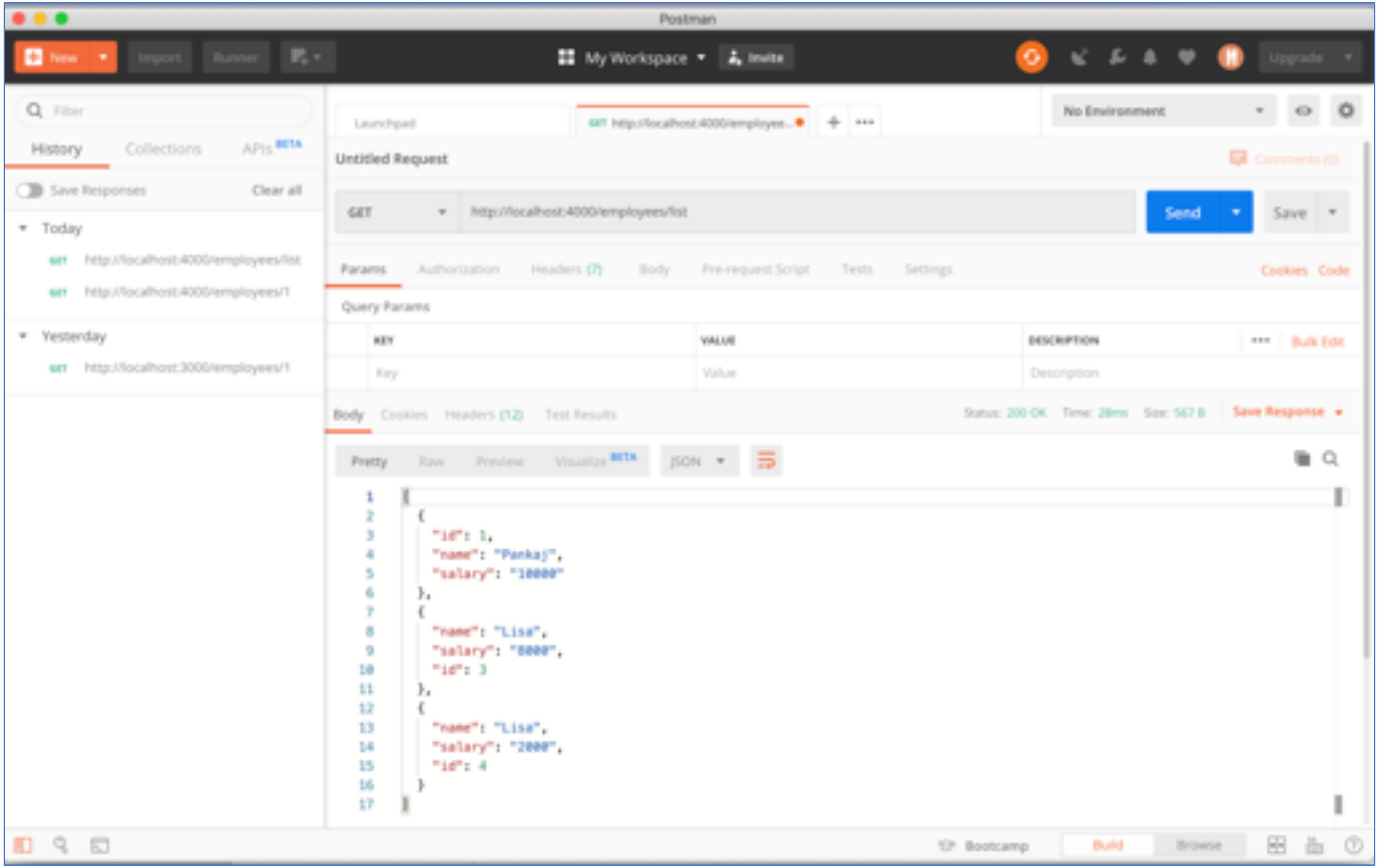
```
curl -XDELETE -H "Content-Type: application/json" "http://localhost:4000/employees/delete/4"
```

```
curl -GET -H "Content-Type: application/json" "http://localhost:4000/employees/list"
```

JSON server provides some other useful options such as sorting, searching and pagination. For more refer link for more details.

Why Postman

- Using Postman you can Send Requests and View Responses.
- Specify Complex Requests – HTTP Methods, URL, Query Params and HTML Body.
- View and Inspect Responses – Status Code, Response Time and Response Size.
- Can download by Clicking the Link.



History:

GET http://localhost:4000/employees/list

GET http://localhost:4000/employees/1

GET http://localhost:3000/employees/1

GET http://localhost:4000/employees/list

Status: 200 OK Time: 28ms Size: 567 B

```
[
  {
    "id": 1,
    "name": "Pankaj",
    "salary": "10000"
  },
  {
    "name": "Lisa",
    "salary": "8000",
    "id": 3
  },
  {
    "name": "Lisa",
    "salary": "2000",
    "id": 4
  }
]
```

BridgeLabz
Employability Delivered

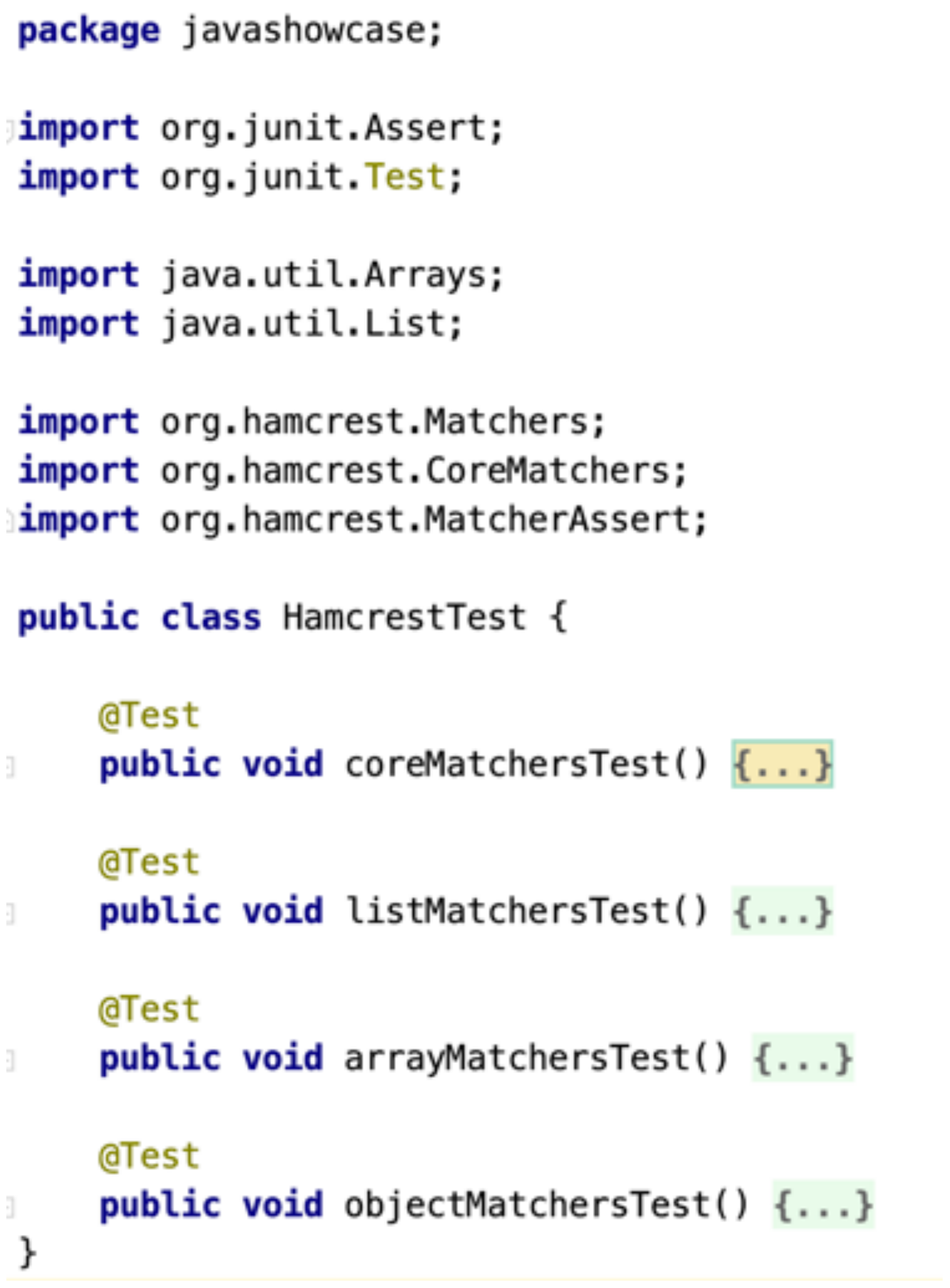
What is Hamcrest

- Hamcrest is a framework for software tests. Hamcrest allows checking for conditions via existing matchers classes.
- Use Hamcrest matchers along with Junit. For this use the assertThat statement followed by one or several matchers.
- Setting Dependency in build.gradle

```
dependencies {
  // Setting dependency to Hamcrest
  testImplementation 'org.hamcrest:hamcrest-library:1.3'
}
```

- For more [Link]

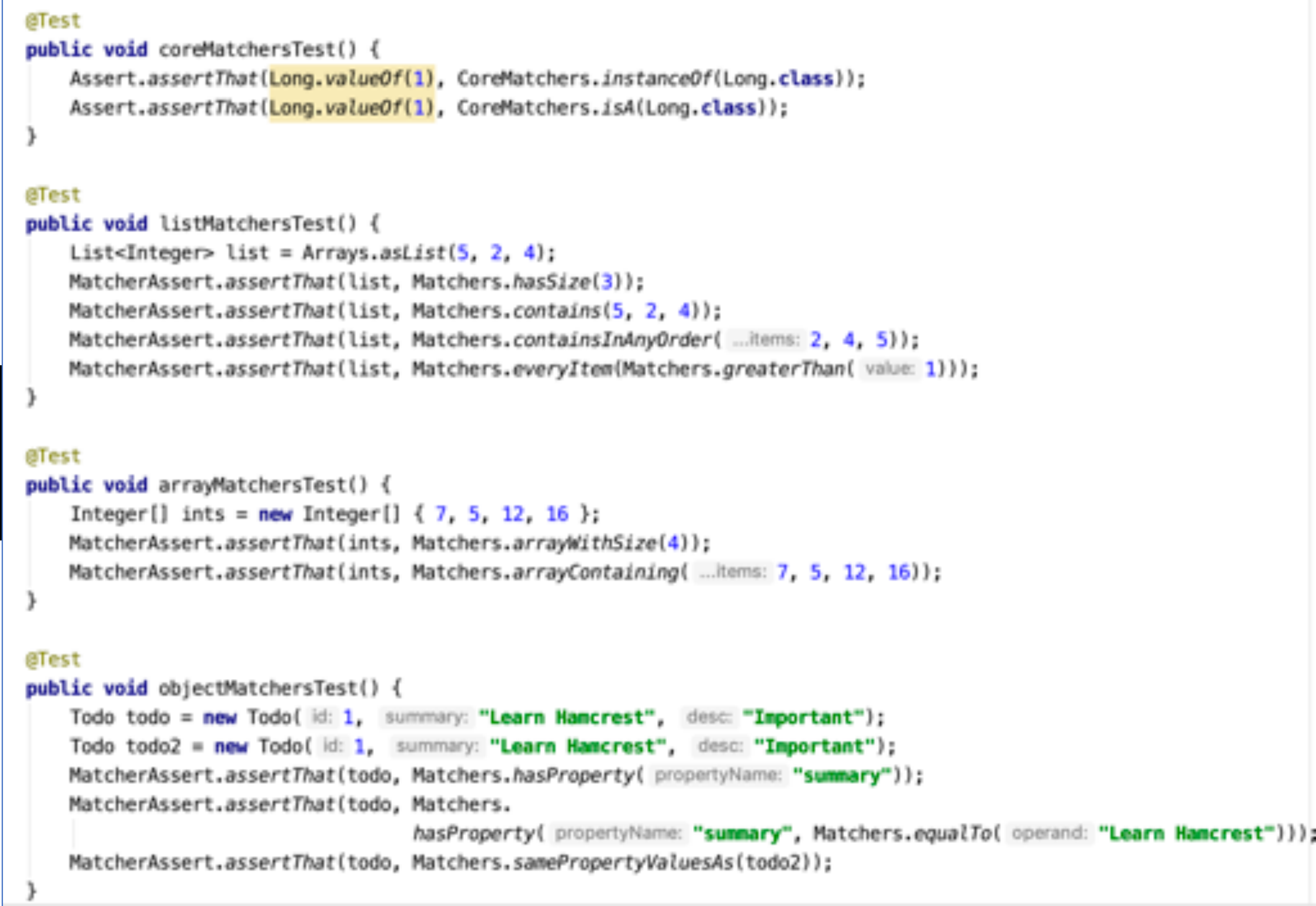
Create HamcrestTest Java File



```java
package javashowcase;

import org.junit.Assert;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

import org.hamcrest.Matchers;
import org.hamcrest.CoreMatchers;
import org.hamcrest.MatcherAssert;

public class HamcrestTest {

    @Test
    public void coreMatchersTest() {...}

    @Test
    public void listMatchersTest() {...}

    @Test
    public void arrayMatchersTest() {...}

    @Test
    public void objectMatchersTest() {...}
}
```

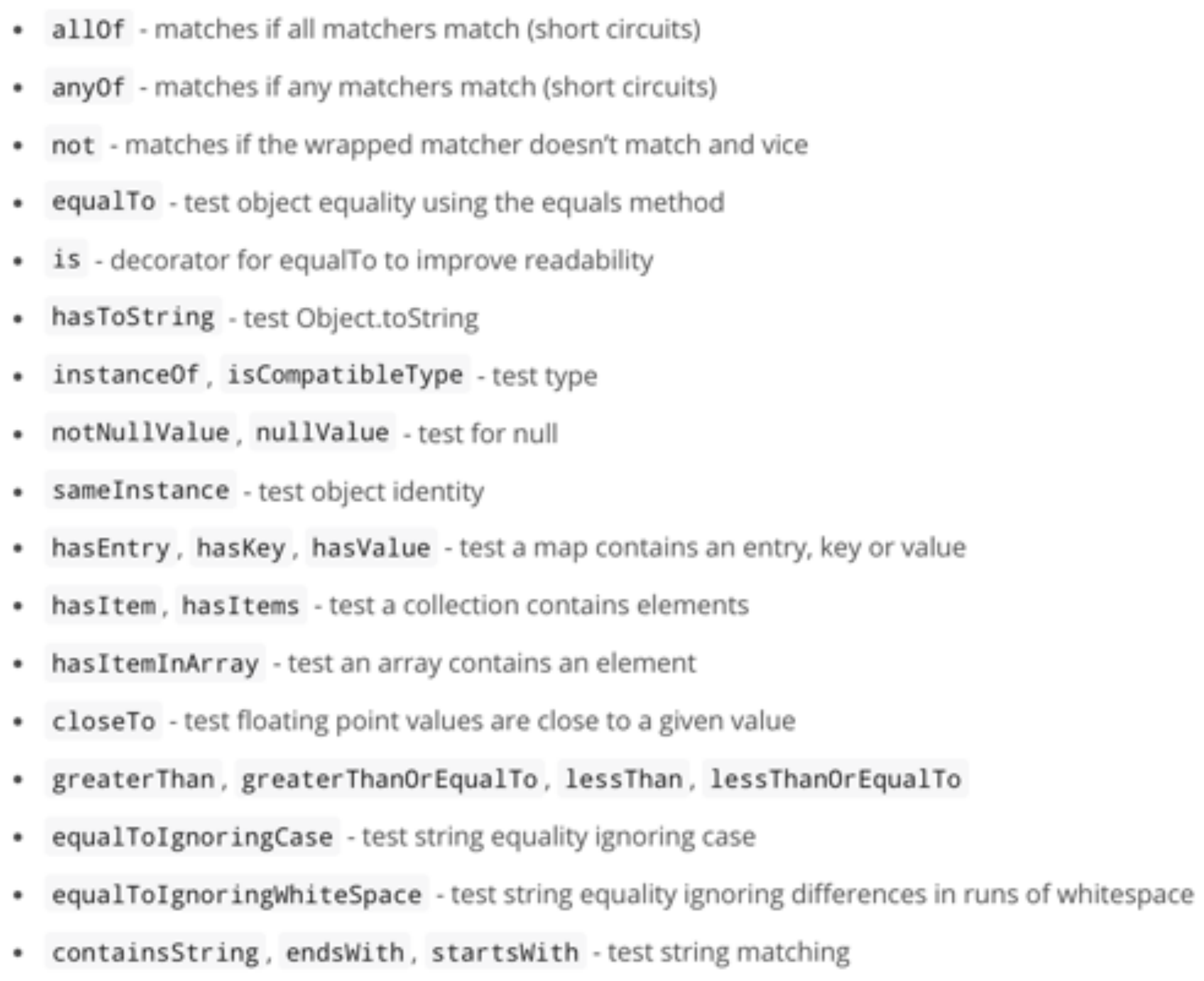
BridgeLabz
Employability Delivered

Hamcrest Test API



```
@Test
public void coreMatchersTest() {
    Assert.assertThat(Long.valueOf(1), CoreMatchers.instanceOf(Long.class));
    Assert.assertThat(Long.valueOf(1), CoreMatchers.isA(Long.class));
}

@Test
public void listMatchersTest() {
    List<Integer> list = Arrays.asList(5, 2, 4);
    MatcherAssert.assertThat(list, Matchers.hasSize(3));
    MatcherAssert.assertThat(list, Matchers.contains(5, 2, 4));
    MatcherAssert.assertThat(list, Matchers.containsInAnyOrder( ...items: 2, 4, 5));
    MatcherAssert.assertThat(list, Matchers.everyItem(Matchers.greaterThan( value: 1)));
}

@Test
public void arrayMatchersTest() {
    Integer[] ints = new Integer[] { 7, 5, 12, 16 };
    MatcherAssert.assertThat(ints, Matchers.arrayWithSize(4));
    MatcherAssert.assertThat(ints, Matchers.arrayContaining( ...items: 7, 5, 12, 16));
}

@Test
public void objectMatchersTest() {
    Todo todo = new Todo( id: 1,  summary: "Learn Hamcrest",  desc: "Important");
    Todo todo2 = new Todo( id: 1,  summary: "Learn Hamcrest",  desc: "Important");
    MatcherAssert.assertThat(todo, Matchers.hasProperty( propertyName: "summary"));
    MatcherAssert.assertThat(todo, Matchers.
                                   hasProperty( propertyName: "summary", Matchers.equalTo( operand: "Learn Hamcrest")));
    MatcherAssert.assertThat(todo, Matchers.samePropertyValuesAs(todo2));
}
```

Hamcrest Matchers API



- `allOf` - matches if all matchers match (short circuits)
- `anyOf` - matches if any matchers match (short circuits)
- `not` - matches if the wrapped matcher doesn't match and vice
- `equalTo` - test object equality using the equals method
- `is` - decorator for equalTo to improve readability
- `hasToString` - test Object.toString
- `instanceOf`, `isCompatibleType` - test type
- `notNullValue`, `nullValue` - test for null
- `sameInstance` - test object identity
- `hasEntry`, `hasKey`, `hasValue` - test a map contains an entry, key or value
- `hasItem`, `hasItems` - test a collection contains elements
- `hasItemInArray` - test an array contains an element
- `closeTo` - test floating point values are close to a given value
- `greaterThan`, `greaterThanOrEqualTo`, `lessThan`, `lessThanOrEqualTo`
- `equalToIgnoringCase` - test string equality ignoring case
- `equalToIgnoringWhiteSpace` - test string equality ignoring differences in runs of whitespace
- `containsString`, `endsWith`, `startsWith` - test string matching

To See all – [Matchers API Reference]

What is REST Assured

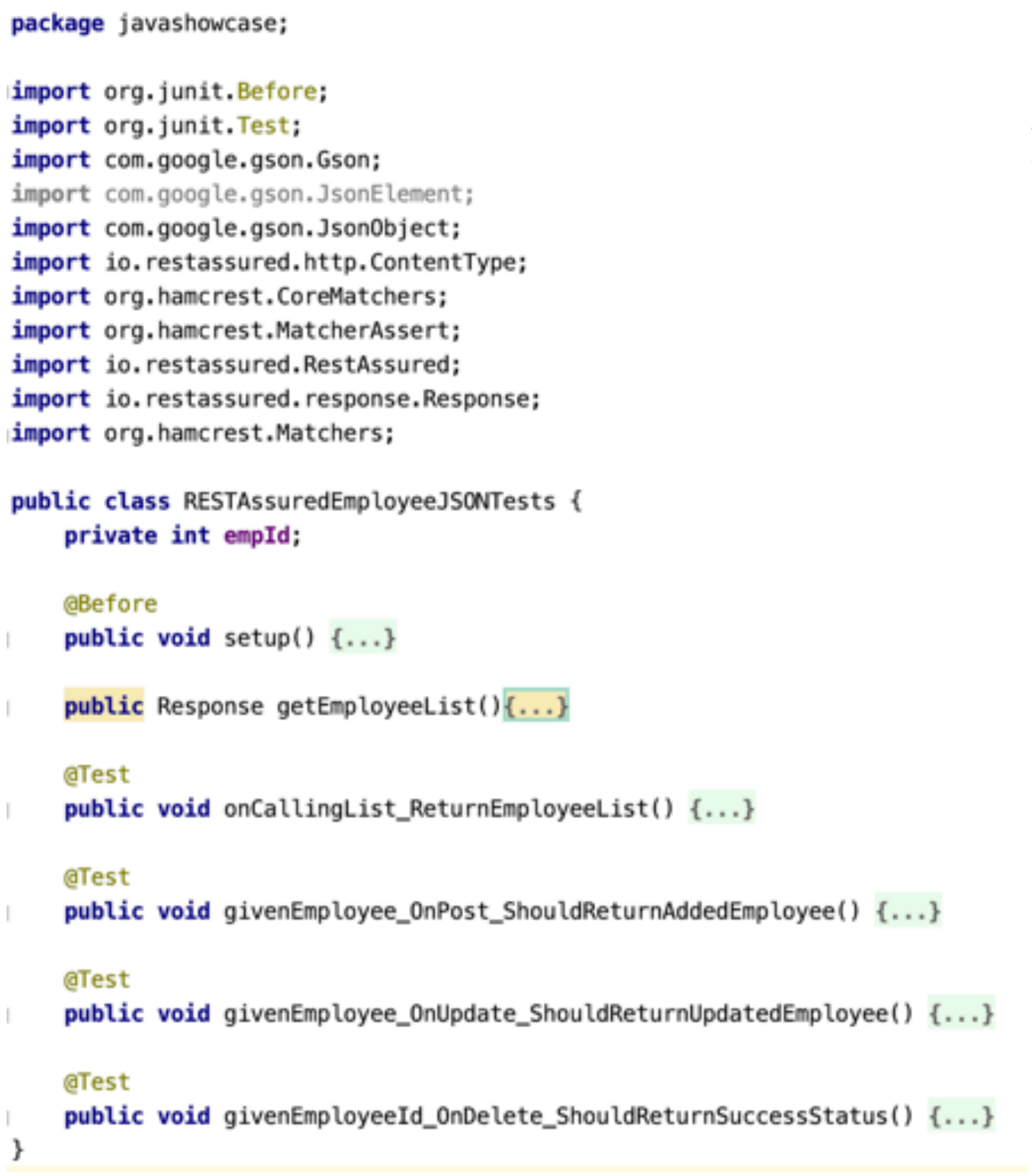
- REST Assured is a Java Domain Specific Language API for simplifying testing of RESTful web services.
- REST Assured API can be used to invoke REST web services and match response content to test them.
- Build.gradle dependency

```
testImplementation 'io.rest-assured:rest-assured:4.1.2'
```

- For more Link

BridgeLabz
Employability Delivered

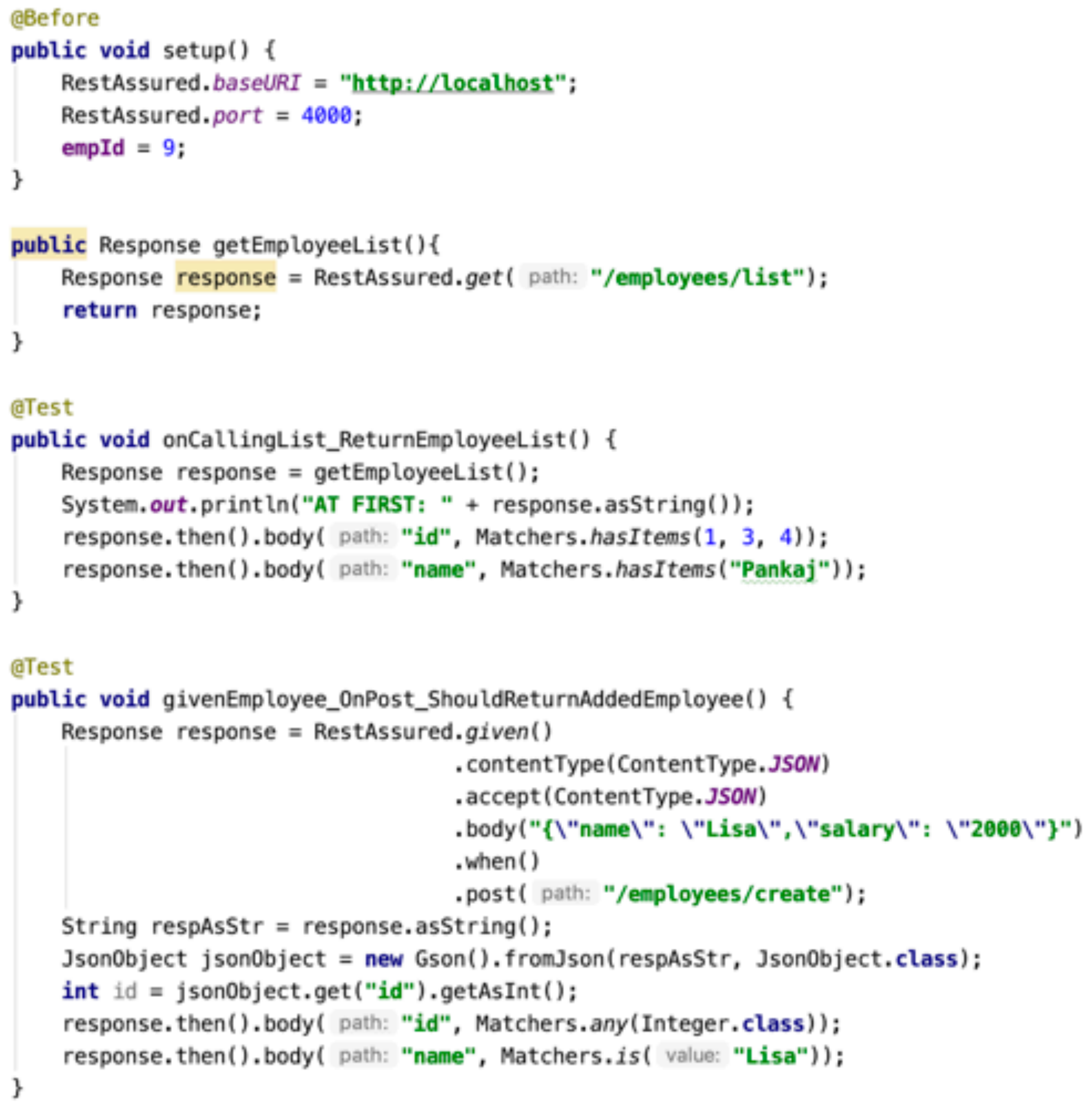
REST Assured Employee JSON Tests



```java
package javashowcase;

import org.junit.Before;
import org.junit.Test;
import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import io.restassured.http.ContentType;
import org.hamcrest.CoreMatchers;
import org.hamcrest.MatcherAssert;
import io.restassured.RestAssured;
import io.restassured.response.Response;
import org.hamcrest.Matchers;

public class RESTAssuredEmployeeJSONTests {
    private int empId;

    @Before
    public void setup() {...}

    public Response getEmployeeList(){...}

    @Test
    public void onCallingList_ReturnEmployeeList() {...}

    @Test
    public void givenEmployee_OnPost_ShouldReturnAddedEmployee() {...}

    @Test
    public void givenEmployee_OnUpdate_ShouldReturnUpdatedEmployee() {...}

    @Test
    public void givenEmployeeId_OnDelete_ShouldReturnSuccessStatus() {...}
}
```

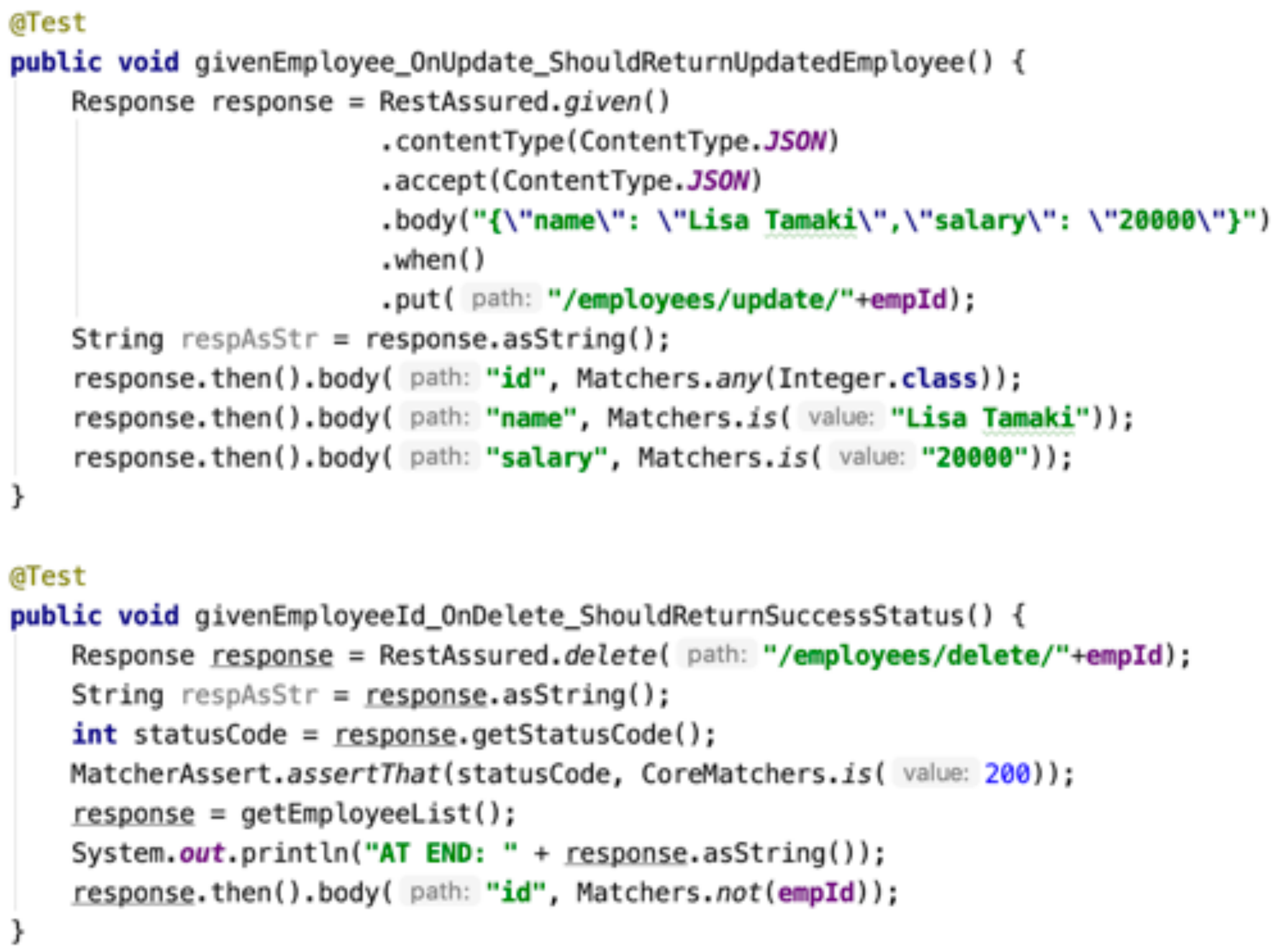
BridgeLabz
Employability Delivered

Test Methods – Get and Post Calls



```
@Before
public void setup() {
    RestAssured.baseURI = "http://localhost";
    RestAssured.port = 4000;
    empId = 9;
}

public Response getEmployeeList(){
    Response response = RestAssured.get( path: "/employees/list");
    return response;
}

@Test
public void onCallingList_ReturnEmployeeList() {
    Response response = getEmployeeList();
    System.out.println("AT FIRST: " + response.asString());
    response.then().body( path: "id", Matchers.hasItems(1, 3, 4));
    response.then().body( path: "name", Matchers.hasItems("Pankaj"));
}

@Test
public void givenEmployee_OnPost_ShouldReturnAddedEmployee() {
    Response response = RestAssured.given()
                                   .contentType(ContentType.JSON)
                                   .accept(ContentType.JSON)
                                   .body("{\"name\": \"Lisa\",\"salary\": \"2000\"}")
                                   .when()
                                   .post( path: "/employees/create");
    String respAsStr = response.asString();
    JsonObject jsonObject = new Gson().fromJson(respAsStr, JsonObject.class);
    int id = jsonObject.get("id").getAsInt();
    response.then().body( path: "id", Matchers.any(Integer.class));
    response.then().body( path: "name", Matchers.is( value: "Lisa"));
}
```

Test Methods – Update and Delete Calls



```
@Test
public void givenEmployee_OnUpdate_ShouldReturnUpdatedEmployee() {
    Response response = RestAssured.given()
                            .contentType(ContentType.JSON)
                            .accept(ContentType.JSON)
                            .body("{\"name\": \"Lisa Tamaki\",\"salary\": \"20000\"}")
                            .when()
                            .put( path: "/employees/update/"+empId);
    String respAsStr = response.asString();
    response.then().body( path: "id", Matchers.any(Integer.class));
    response.then().body( path: "name", Matchers.is( value: "Lisa Tamaki"));
    response.then().body( path: "salary", Matchers.is( value: "20000"));
}

@Test
public void givenEmployeeId_OnDelete_ShouldReturnSuccessStatus() {
    Response response = RestAssured.delete( path: "/employees/delete/"+empId);
    String respAsStr = response.asString();
    int statusCode = response.getStatusCode();
    MatcherAssert.assertThat(statusCode, CoreMatchers.is( value: 200));
    response = getEmployeeList();
    System.out.println("AT END: " + response.asString());
    response.then().body( path: "id", Matchers.not(empId));
}
```

BridgeLabz

Employability Delivered

Thank You