CORE JAVA

WEB SERVICES

KavinSchool

1

API TESTING

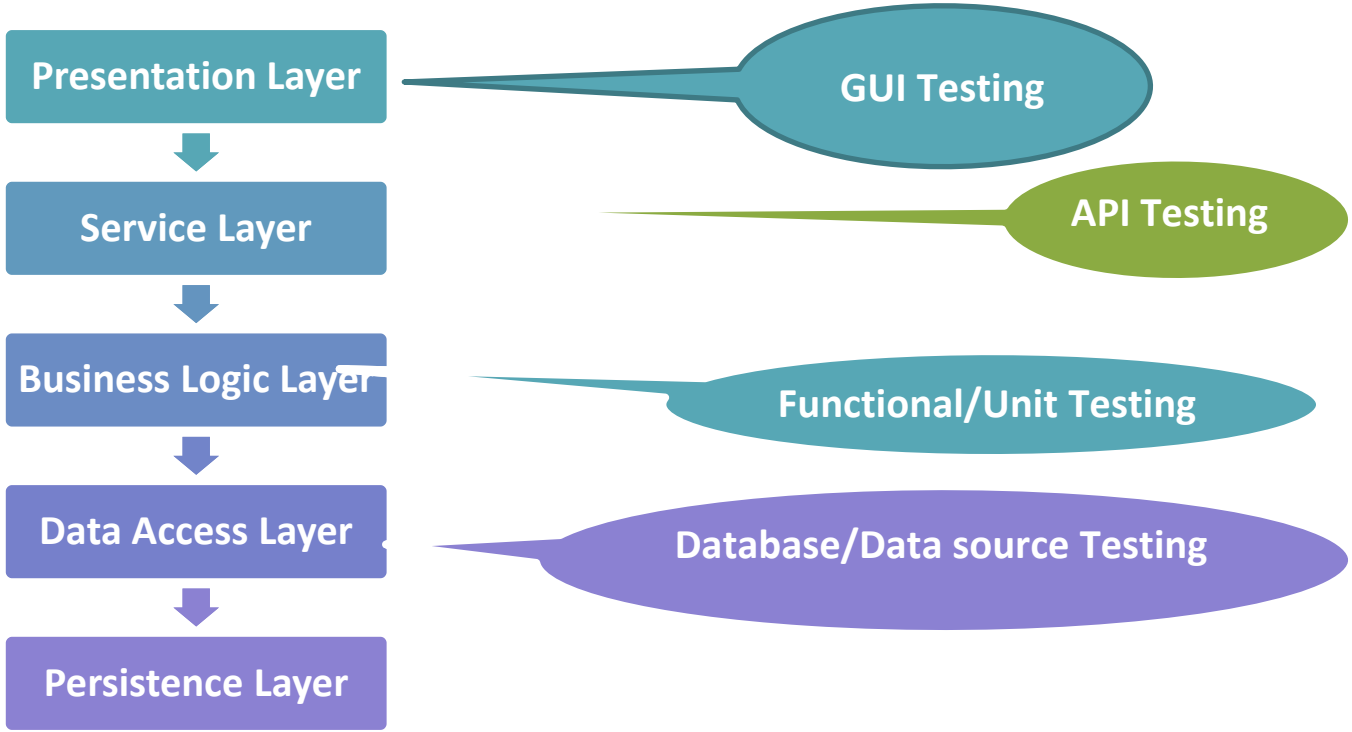
Learning Objectives

- What is API testing?
- N-tier architecture
- More info on API testing
- Workflow of API
- Most used API testing tools
- Types of API testing
- Advantages of API testing
- Challenges of API testing
- What to test?
- Unit vs. Api testing

WHAT IS API TESTING?

- **API** testing is a type of software testing that involves testing Application Programming Interfaces (APIs) directly without involving GUI.
- API testing concentrates on using software to make API calls in order to receive an output before observing and logging the system's response. Verify the API returns a correct response or output under varying conditions.
- The output of typical API testing is:
  - A Pass or Fail status
  - Data or information
  - A call to another API

N-TIER ARCHITECTURE



MORE INFO ON API TESTING

- API testing is performed on APIs that the development team produces as well as APIs that the team consumes within their application (including third-party APIs)
- API testing is concentrates on the business logic layer which is exposed through a service layer (using Web Services) of the software architecture
- API testing utilizes programming to send calls to the API and get the result
- The objective of API testing is to confirm right execution and blunder treatment of the part preceding its coordination into an application

WORKFLOW OF API

Client
- Request
- Response

API
- Request
- Response

Server
- Process

User

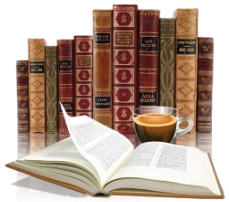
Request

Response

Librarian

Request

Response



Books

MOST USED API TESTING TOOLS

- Postman
- ReadyAPI
- REST-Assured
- Swagger.io
- JMeter
- Karate DSL
- Apigee
- Katalon Platform
- HyperTest
- Airborne

TYPES OF API TESTING

- API doesn’t return any response data
- API Response Data is not structured
- API connection and getting response based on the provided input and verifying the expected output
- Verification of the API whether it triggers some other event or request another API
- Verification of the API whether it is updating any data structure
- API Response time from different region
- API performance for huge load of request
- Security and Multi-threading issues

ADVANTAGES OF API TESTING

- API testing is language-independent
- Using API, testing the business functionality is easy
- Reduces the testing cost
- Reduces the risks
- GUI testing is brittle and time-consuming compared to API testing

CHALLENGES OF API TESTING

- Finding Parameter Combination, Parameter Selection, and Call Sequencing
- Understanding and testing API functionality is challenging comparing to GUI application
- Validating and Verifying the output in different system is difficult for testers
- Parameters selection and categorization required to be known to the testers
- Exception handling function needs to be tested
- Coding knowledge is necessary for testers

WHAT TO TEST?

- Unit testing – Individual units of API functionality are tested
- Functional testing – Verifying scenario-based API functionality using the existing unit tests as a building block for E2E tests
- Load testing – Stress and performance testing of the existing functional test cases
- Runtime error detection – Checking the race conditions, exceptions, and resource leaks
- Security testing - Penetration testing, fuzz testing, authentication, encryption, and access control

WHAT TO TEST?

- Discovery testing - Verifying that a specific resource exposed by the API can be created, selected, and deleted as documented by the API
- Usability testing - Verifying whether the API is functional and user-friendly and integrates well with another platform as well
- Automated testing – Using automation is the best way to test the APIs
- Documentation Testing - Documentation should be a part of the final deliverable, verified for its correctness

UNIT VS. API TESTING

Unit Testing	API Testing
Owned by Development team	Owned by QA team
White Box Testing	Block Box Testing
Individual Units tested	Full functionality of the exposed API is tested
Source Code Access Needed	API name, parameter, call Sequence information needed
Less documented as internal team mostly involved	Highly documented as API integrates with multiple and external systems, APIs need to be understood by external teams and vendors

CORE JAVA

REST-ASSURED

REST-assured

KavinSchool

2

REST-ASSURED FRAMEWORK WITH JAVA

REST-assured

Learning Objectives

- What are restful web services?
- Key concepts of rest
- What is REST assured?
- Setting up REST assured
- Writing and executing **get** request example
- Writing and executing **post-request** example
- Writing and executing **put** request example
- Writing and executing **delete** request

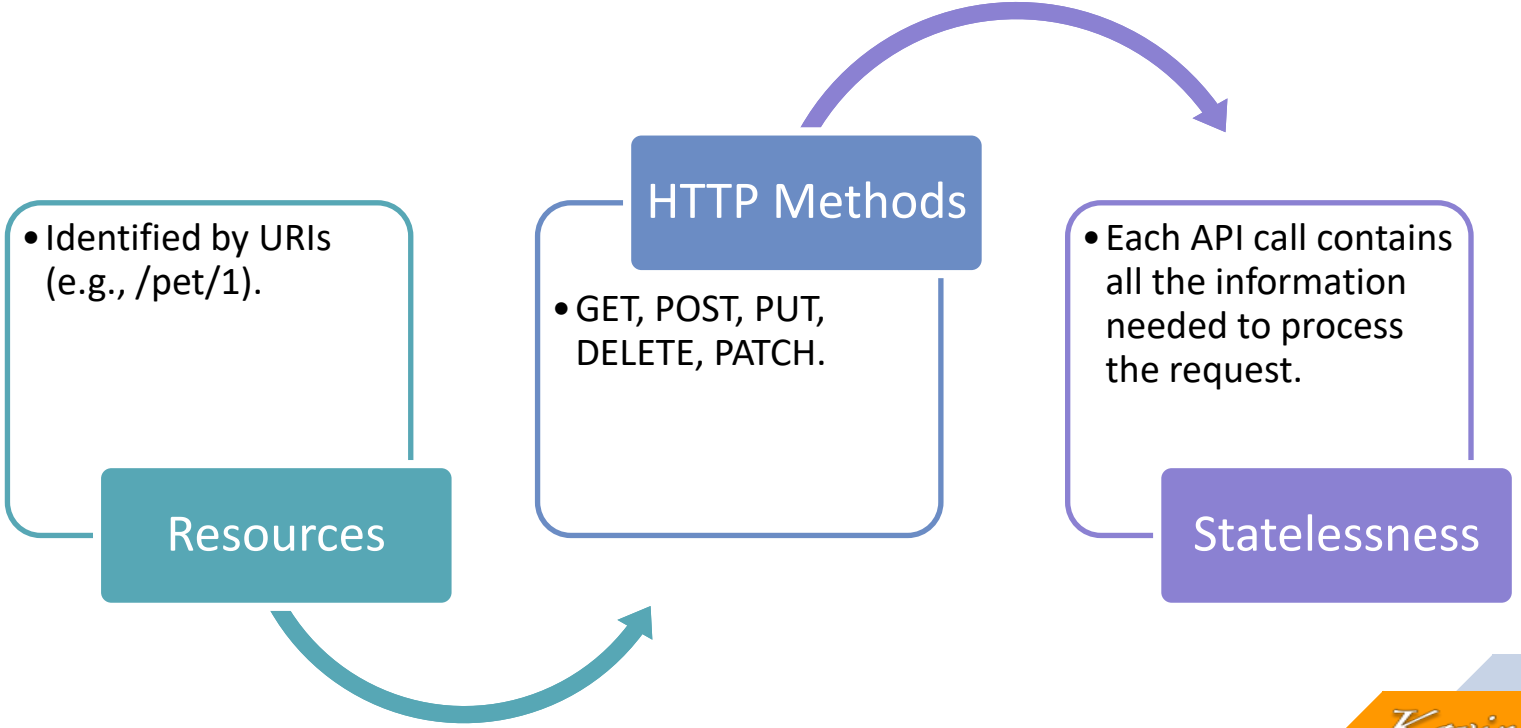
WHAT ARE RESTFUL WEB SERVICES?

REST (Representational State Transfer) is an architectural style for designing networked applications.

RESTful APIs allow communication between systems using HTTP methods.

Key principles: statelessness, client-server architecture, and cacheability.

KEY CONCEPTS OF REST



WHAT IS REST-ASSURED?

- Rest Assured is a Java library used for testing RESTful APIs.
- It simplifies writing HTTP requests and validating responses.
- Built on top of popular libraries like Apache HTTPClient and Hamcrest.

SETTING UP RESTASSURED

- Required dependencies: Add REST Assured and TestNG (or JUnit) to the pom.xml (for Maven users).

```
<dependency>
	<groupId>org.testng</groupId>
	<artifactId>testng</artifactId>
	<version>${testng.version}</version>
</dependency>

<dependency>
	<groupId>io.rest-assured</groupId>
	<artifactId>rest-assured</artifactId>
	<version>${rest-assured.version}</version>
	<scope>test</scope>
</dependency>
```

- Basic Maven Dependency

WRITING AND EXECUTING GET REQUEST EXAMPLE

```
Response Code: 200
Response Body:
{"id":1,"category":{"id":0,"name":"John"},"name"
:"","photoUrls":[],"tags":[],"status":"8000"}
```

```java
import io.restassured.RestAssured;
import io.restassured.response.Response;

public class GetPet {
  public static void main(String[] args) {
    Response response = RestAssured.get("https://petstore.swagger.io/v2/pet/1");
    System.out.println("Response Code: " + response.getStatusCode());
    System.out.println("Response Body: " + response.getBody().asString());
  }
}
```

BASIC AUTH EXAMPLE USING GET REQUEST

```
public class BasicAuthExample {
    public static void main(String[] args) {
        Response response = RestAssured
                .given()
                .auth()
                .preemptive()
                .basic("username", "password")
                .when()
                .get("https://petstore.swagger.io/v2/pet/10");

        System.out.println("Response Code: " + response.getStatusCode());
    }
}
```

```
Response Code: 200
```

WRITING AND EXECUTING POST REQUEST EXAMPLE

```java
public class CreatePet {
    public static void main(String[] args) {
        String requestBody = """
                {
                  "id": 1000,
                  "category": {
                    "id": 0,
                    "name": "string"
                  },
                  "name": "kangs",
                  "status": "available"
                }""";
        Response response = RestAssured
                .given()
                .contentType(ContentType.JSON)
                .body(requestBody)
                .when()
                .post("https://petstore.swagger.io/v2/pet");
        System.out.println("Response Code: " + response.getStatusCode());
        System.out.println("Response Body: " + response.getBody().asString());
    }
}
```

```
Response Code: 200
Response Body:
{"id":1000,"category":{"id":0,"name":"string"},"name":"kangs",
"photoUrls":[],"tags":[],"status":"available"}
```

WRITING AND EXECUTING PUT REQUEST EXAMPLE

```
public class UpdatePet {
    public static void main(String[] args) {
        String updateRequestBody = """
                { "id": 10, "name": "doggie updated", "status": "sold" }""";

        Response response = RestAssured
                .given()
                .contentType(ContentType.JSON)
                .body(updateRequestBody)
                .when()
                .put("https://petstore.swagger.io/v2/pet");

        System.out.println("Response Code: " + response.getStatusCode());
        System.out.println("Response Body: " + response.getBody().asString());
    }
}
```

```
Response Code: 200
Response Body: {"id":10,"name":"doggie
updated","photoUrls":[],"tags":[],"status":"sold"}
```

WRITING AND EXECUTING DELETE REQUEST

```
package com.kavinschool.petstore.tests.rest.assured;

import io.restassured.RestAssured;
import io.restassured.response.Response;

public class DeletePet {
   public static void main(String[] args) {
      Response response = RestAssured.delete("https://petstore.swagger.io/v2/pet/10");

      System.out.println("Response Code: " + response.getStatusCode());
   }
}
```

```
Response Code: 200
```

Valid URI

```
package com.kavinschool.petstore.tests.rest.assured;

import io.restassured.RestAssured;
import io.restassured.response.Response;

public class DeletePet {
   public static void main(String[] args) {
      Response response = RestAssured.delete("https://petstore.swagger.io/v2/pet/1005");

      System.out.println("Response Code: " + response.getStatusCode());
   }
}
```

```
Response Code: 404
```

Invalid URI

3

ADVANCED RESTASSURED

Learning Objectives

- Api key auth example using get request
- Validate schema example for get request
- Upload image example using a post request
- Chaining example using post & get request
- Handling query param ex. Using get request
- Handling cookies ex. Using get request

API_KEY AUTH EXAMPLE USING GET REQUEST

```java
public class ApiKeyAuthExample {
    public static void main(String[] args) {
        // Set the API key
        String apiKey = "special-key";

        // Send a GET request with the API key in the header
        Response response = RestAssured
            .given()
            .header("api_key", apiKey) // Pass the API key in the header
            .when()
            .get("https://petstore.swagger.io/v2/pet/1000");

        // Print the response details
        System.out.println("Response Code: " + response.getStatusCode());
        System.out.println("Response Body: " + response.getBody().asString());
    }
}
```

```
Response Code: 200
Response Body:
{"id":1000,"category":{"id":0,"name":"string"},"
name":"kangs","photoUrls":[],"tags":[],"status":
"available"}
```

VALIDATE SCHEMA EXAMPLE FOR GET REQUEST

```java
public class ValidateSchemaExample {

    public static final String PET_SCHEMA_FILE = "pet-schema.json";

    public static void main(String[] args) {
        RestAssured.baseURI = "https://petstore3.swagger.io/api/v3";

        Response response = given()
                .when()
                .get("/pet/1");

        // Validate against the schema
        response
                .then()
                .statusCode(200)
                .body(matchesJsonSchemaInClasspath(PET_SCHEMA_FILE));
    }
}
```

pet-schema.json

```json
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "id": {
      "type": "integer"
    },
    "name": {
      "type": "string"
    },
    "status": {
      "type": "string"
    }
  },
  "required": ["id", "name", "status"]
}
```

UPLOAD IMAGE EXAMPLE USING POST REQUEST

```java
public class UploadImageExample {
    public static void main(String[] args) {
        File imageFile = new File("src/test/resources/dog.png");

        Response response = RestAssured
                .given()
                .contentType(ContentType.MULTIPART)
                .multiPart("file", imageFile)
                .when()
                .post("https://petstore.swagger.io/v2/pet/1/uploadImage");

        System.out.println("Response Code: " + response.getStatusCode());
        System.out.println("Response Body: " + response.getBody().asString());
    }
}
```

```
{
  "code": 200,
  "type": "unknown",
  "message": "additionalMetadata: null\nFile uploaded to
./dog.png, 56081 bytes"
}
```

CHAINING EXAMPLE USING POST & GET REQUEST

```
public static void main(String[] args) {
    String requestBody = """
            { "id": 100, "name": "doggie", "status": "available" }""";

    // Create a Pet
    RestAssured
            .given()
            .contentType(ContentType.JSON)
            .body(requestBody)
            .when()
            .post("https://petstore.swagger.io/v2/pet");

    // Retrieve the Pet
    RestAssured
            .given()
            .pathParam("petId", 100)
            .when()
            .get("https://petstore.swagger.io/v2/pet/{petId}")
            .then()
            .statusCode(200)
            .body("name", equalTo("doggie"));
}
```

HANDLING QUERY PARAM EX. USING GET REQUEST

```java
public class FindPetsByStatus {
    public static void main(String[] args) {
        Response response = RestAssured
                .given()
                .queryParam("status", "available")
                .when()
                .get("https://petstore.swagger.io/v2/pet/findByStatus");

        System.out.println("Response Code: " + response.getStatusCode());
        System.out.println("Available Pets: " + response.getBody().asString());
    }
}
```

```
Response Code: 200
Available Pets: [{
    "id":
9223372036854776000,
    "name": "newname",
    "photoUrls": [],
    "tags": [
      {
        "id": 0,
        "name": "cute"
      }
    ],
    "status": "available"
  },
  {
    "id": 24,
    "category": {
      "id": 0,
      "name": "string"
    },
    "name": "doggie",
    "photoUrls": [
      "string"
    ],
    "tags": [
      {
        "id": 0,
        "name": "string"
      }
    ],
    "status": "available"
}]
```

HANDLING COOKIES EX. USING GET REQUEST

```
Response Code: 200
```

```java
public class CookieExample {
    public static void main(String[] args) {
        // Setting a cookie
        Response response = RestAssured
                .given()
                .cookie("session_id", "123456")
                .when()
                .get("https://petstore.swagger.io/v2/pet/1");

        System.out.println("Response Code: " + response.getStatusCode());
    }
}
```


THANK
YOU..

THANKS!

Your feedback is welcome
support@kavinschool.com