Practical_Guide_To_API

Here are some **commonly used HTTP status codes** and their meanings, along with examples of valid live APIs for testing:

Common HTTP Status Codes and Their Meanings

1. **200 OK**
    - **Meaning**: The request was successful, and the server returned the expected response.
    - **Example**: A successful GET request to retrieve data.
    - **Live API Example**:
        - `GET https://jsonplaceholder.typicode.com/posts/1`
        - This returns a specific post from a sample JSON API.
2. **201 Created**
    - **Meaning**: The request has been fulfilled, and a new resource was created.
    - **Example**: A POST request that creates a new resource on the server.
    - **Live API Example**:
        - `POST https://jsonplaceholder.typicode.com/posts`
        - Sending JSON data in the body to create a new post.
3. **204 No Content**
    - **Meaning**: The request was successful, but there is no content to return.
    - **Example**: A DELETE request where the resource was deleted successfully.
    - **Live API Example**:
        - `DELETE https://jsonplaceholder.typicode.com/posts/1`
        - This deletes a specific post and returns no content.
4. **400 Bad Request**
    - **Meaning**: The server could not understand the request due to invalid syntax.
    - **Example**: A malformed request that the server can’t process.
    - **Live API Example**:
        - `GET https://jsonplaceholder.typicode.com/posts/abc`
        - Trying to request a post with an invalid ID will return a 400 error.
5. **401 Unauthorised**
    - **Meaning**: Authentication is required, and the user is not authenticated.
    - **Example**: Accessing a protected resource without valid credentials.
    - **Live API Example**:
        - `GET https://reqres.in/api/secure-endpoint`
        - Accessing a secure endpoint without proper authentication.
6. **403 Forbidden**
    - **Meaning**: The request is valid, but the server refuses to authorize it.
    - **Example**: Trying to access a restricted resource.
    - **Live API Example**:

        - `GET https://jsonplaceholder.typicode.com/admin`
        - Accessing a restricted resource that you don't have permission to access.

7. **404 Not Found**
    - **Meaning**: The server cannot find the requested resource.
    - **Example**: A GET request to a non-existent endpoint or resource.
    - **Live API Example**:
        - `GET https://jsonplaceholder.typicode.com/posts/12345`
        - Requesting a post that does not exist.
8. **500 Internal Server Error**
    - **Meaning**: The server encountered an unexpected condition that prevented it from fulfilling the request.
    - **Example**: A server misconfiguration or issue causes the error.
    - **Live API Example**:
        - `GET https://httpstat.us/500`
        - This returns a 500 error to simulate a server issue.
9. **502 Bad Gateway**
    - **Meaning**: The server, while acting as a gateway or proxy, received an invalid response from the upstream server.
    - **Example**: A server in the middle cannot process the request properly.
    - **Live API Example**:
        - `GET https://httpstat.us/502`
        - This returns a 502 error to simulate a bad gateway issue.
10. **503 Service Unavailable**
    - **Meaning**: The server is temporarily unable to handle the request due to maintenance or overload.
    - **Example**: A server under maintenance or experiencing high load.
    - **Live API Example**:
        - `GET https://httpstat.us/503`
        - This returns a 503 error to simulate a service unavailability.

Live APIs for Testing:

1. **JSONPlaceholder API**:
    - A free online REST API for testing and prototyping.
    - **Base URL**: `https://jsonplaceholder.typicode.com/`
    - Example Endpoints:
        - `GET https://jsonplaceholder.typicode.com/posts`
        - `POST https://jsonplaceholder.typicode.com/posts`

2. **ReqRes**:
   - A hosted REST API that simulates user management scenarios.
   - **Base URL**: `https://reqres.in/`
   - Example Endpoints:
     - `GET https://reqres.in/api/users`
     - `POST https://reqres.in/api/users`
3. **HTTP Stat.us**:
   - A simple service to return different HTTP status codes for testing.
   - **Base URL**: `https://httpstat.us/`
   - Example Endpoints:
     - `GET https://httpstat.us/200`
     - `GET https://httpstat.us/404`
4. **The Dog API**:
   - A fun API that returns pictures and information about dogs.
   - **Base URL**: `https://thedogapi.com/`
   - Example Endpoints:
     - `GET https://api.thedogapi.com/v1/breeds`
     - `GET https://api.thedogapi.com/v1/images/search`

---

6 sľgp-by-sľgp pracľical guidg ľo 6PI ľgsľing, wiľh g…amplgs using Posľman and Pgsľ 6ssurgd.

---

1. Understanding API Basics

API stands for **Application Programming Interface**, a set of rules that define how software components should interact. APIs typically use HTTP requests to communicate, which can involve the following methods:

- **GET**: Retrieve data from a server.
- **POST**: Send data to a server to create a new resource.
- **PUT**: Update an existing resource.
- **DELETE**: Remove a resource from the server.

---

2. Setting Up Tools for API Testing

For this guide, we’ll use two common tools:

- **Postman**: A user-friendly GUI tool for testing APIs.
- **Rest Assured**: A Java-based API testing library for automation testing.

3. Using Postman for API Testing

Step 1: Install Postman

Download and install Postman from Postman's official website.

Step 2: Create a New Request

- Open Postman, and click **New** > **Request**.
- Select the request type (e.g., GET, POST) and enter the API URL. For example:
  - `GET https://jsonplaceholder.typicode.com/posts/1`

Step 3: Add Parameters/Body

- If needed, add query parameters or request body in **Params** or **Body** tabs.

Step 4: Send Request

- Click **Send** and examine the response (e.g., Status Code, JSON data).

Step 5: Writing Tests

- In the **Tests** tab, write simple assertions using JavaScript:

javascript

```
pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});
```

Step 6: Collection Runs and Automation

- Group your requests into **collections** and automate them using Postman's **Collection Runner** for repeated testing.

4. Using Rest Assured for API Automation

Rest Assured is a powerful tool for automating API tests in a Java environment.

Step 1: Set Up Rest Assured

Add Rest Assured as a dependency in your `pom.xml` (for Maven projects):

xml

```
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>4.4.0</version>
    <scope>test</scope>
</dependency>
```

Step 2: Writing a Basic Test

Here’s a simple example of testing a **GET** request using Rest Assured:

java

```
import io.restassured.RestAssured;
import io.restassured.response.Response;
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class ApiTest {
    public static void main(String[] args) {
        RestAssured.baseURI =
"https://jsonplaceholder.typicode.com";

        // Simple GET request
        given()
            .when()
            .get("/posts/1")
            .then()

            .statusCode(200)

            .body("userId", equalTo(1));

    }

}
```

Step 3: Running the Test

- Use a testing framework like **JUnit** or **TestNG** to run your test cases.
- Execute the test with assertions to validate the status code and response body.

---

5. Key API Testing Concepts

1. **Status Code Validation**: Verify the status code in each response. Common status codes include:
    - **200 OK**: Request was successful.
    - **201 Created**: Resource created successfully.
    - **400 Bad Request**: Invalid request syntax.
    - **401 Unauthorised**: Authentication required.
    - **500 Internal Server Error**: Server encountered an error.
2. **Response Body Validation**: Ensure the returned data matches expectations. Use assertions to check specific fields:

java

```
body("title", equalTo("foo"))
```

3. **Header Validation**: Validate HTTP headers for security and content:

java

```
header("Content-Type", "application/json; charset=utf-8")
```

4. **Authentication**: Some APIs require authentication. Add authorization tokens in the **Headers** section in Postman or pass tokens in Rest Assured:

java

```
given().auth().oauth2("YOUR_ACCESS_TOKEN")
```

5. **Negative Testing**: Send invalid data to test how the API handles errors.

---

6. Advanced Techniques

1. **Parameterized Tests**: Reuse test cases with different sets of input data using variables or environments.
2. **Data-Driven Testing**: Use CSV or JSON files to input multiple datasets in automated tests.
3. **Mocking APIs**: Use tools like **Postman Mock Servers** or **WireMock** to simulate APIs that may not be fully developed yet.

---

7. Reporting and Continuous Integration

- Use tools like **Newman** (Postman’s command-line runner) to integrate API tests into your **CI/CD pipelines**.
- For Rest Assured, combine with reporting frameworks like **Extent Reports** for detailed results.

---

8. API Testing Best Practices

1. **Clear Test Cases**: Ensure your test cases cover all scenarios, including happy paths and edge cases.
2. **Independent Tests**: API tests should not depend on each other to avoid false negatives.
3. **Monitor APIs**: After testing, set up API monitoring using tools like **Postman Monitors** to catch downtime or slow performance.

Conclusion

API testing is critical to ensuring that your backend services perform reliably. Whether you use Postman for manual testing or automate your API tests with Rest Assured, following this guide will help ensure that your APIs meet functionality, performance, and reliability expectations.