Different Types of Authentication in Rest Assured

BHAVIN THUMAR

Authentication vs. Authorization: Understanding the Basics

Authentication means verifying a user's or application's identity when accessing an API. It tries to answer the question, “Who are you?”.

Common authentication methods include Basic Auth, Digest Auth, OAUTH, API keys, Bearer tokens, and more. Proper authentication ensures that only legitimate users gain entry.

Authorization usually happens after authentication and is the process of granting or denying access based on a user’s or application’s permissions. It answers the question, “What are you allowed to do?”.

HTTP Status Codes 401 and 403

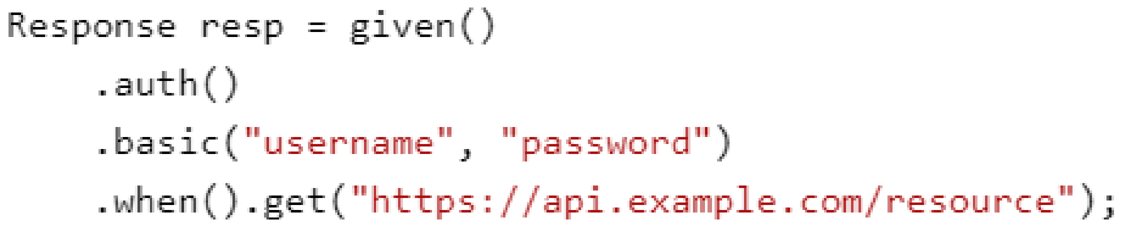
When dealing with authentication and authorization, you'll often encounter two HTTP status codes: **401 Unauthorised and 403 Forbidden.**

- **401 Unauthorised:** This code signifies that the client lacks valid authentication credentials. In other words, the client needs to provide valid credentials to proceed.
- **403 Forbidden:** The 403 code indicates that the client's authentication credentials are valid, but they don't have the necessary permissions to access the requested resource. It's a firm denial of access.

Different Authentication Methods in Rest Assured:

1. Basic Authentication:

Basic Authentication, a straightforward method employed in web apps and APIs, entails sending credentials (username and password) with every request to validate the requester’s identity. This approach, widely supported and easy to implement, is often used to secure resources. It’s preferred when simplicity and efficiency are paramount.



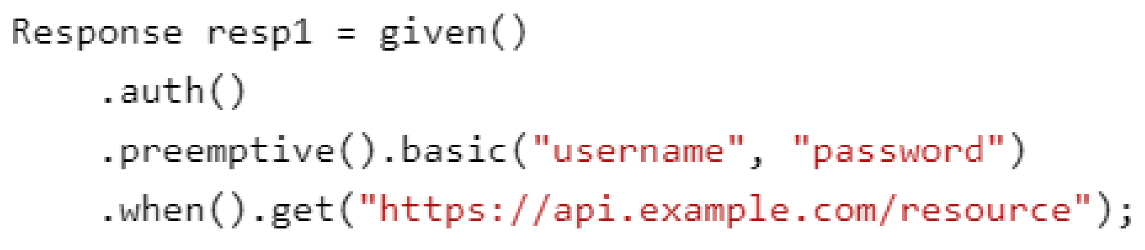
```
Response resp = given()
     .auth()
     .basic("username", "password")
     .when().get("https://api.example.com/resource");
```

Through .auth().basic(“username”, “password”), Rest Assured configures the request with your credentials.

2. Pre-emptive Authentication

Pre-emptive Authentication is an authentication strategy employed in HTTP clients to proactively send authentication credentials with the initial request, rather than waiting for the server to respond with a 401 Unauthorized status code.

In the context of Rest Assured and other HTTP client libraries, pre-emptive authentication means sending authentication credentials in the very first request, even before receiving any response from the server. This can be especially useful when dealing with APIs that require authentication for every request and do not challenge with a 401 status code.

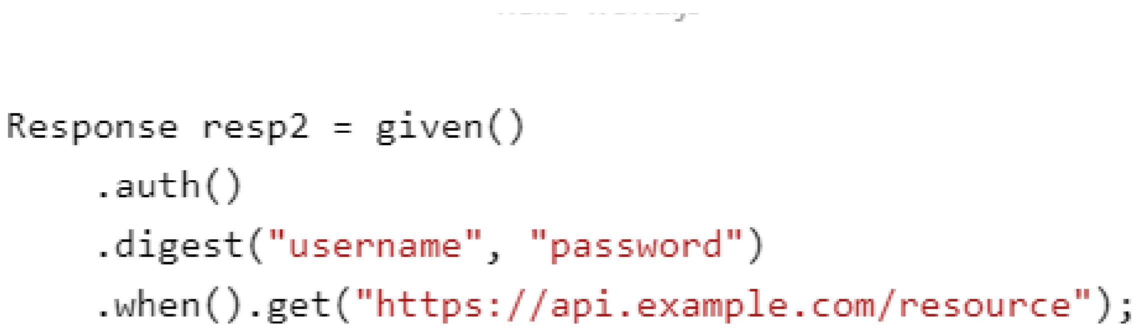


```
Response resp1 = given()
     .auth()
     .preemptive().basic("username", "password")
     .when().get("https://api.example.com/resource");
```

By using .preemptive() before .basic(), Rest Assured takes the initiative in including the credentials.

3. Digest Authentication

Digest Authentication is an authentication mechanism used in HTTP to enhance the security of Basic Authentication. It addresses some of the security vulnerabilities present in Basic Authentication, such as the transmission of credentials in plain text, by using a more secure approach. Digest Authentication challenges the client with a server-generated nonce (a unique token) and requires the client to respond with a hashed value of the nonce, username, password, and other request-specific information.



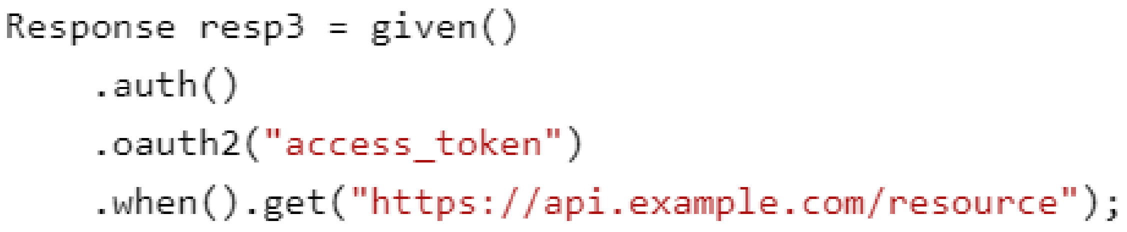
```
Response resp2 = given()
    .auth()
    .digest("username", "password")
    .when().get("https://api.example.com/resource");
```

Through .digest(“username”, “password”), Digest Authentication is configured.

4. OAuth2 Authentication:

OAuth2 (Open Authorization 2.0) is a widely used authorization framework that allows applications to obtain limited access to user accounts on behalf of a third-party application. It’s commonly used to enable secure and controlled access to APIs and resources without exposing the user’s credentials.

OAuth2 involves various roles, including the resource owner (user), client application (third-party app), authorization server (handles authentication and issues access tokens), and resource server (holds the protected resources). The process revolves around obtaining an access token, which serves as a temporary authorization token that allows the client application to access specific resources on behalf of the user.

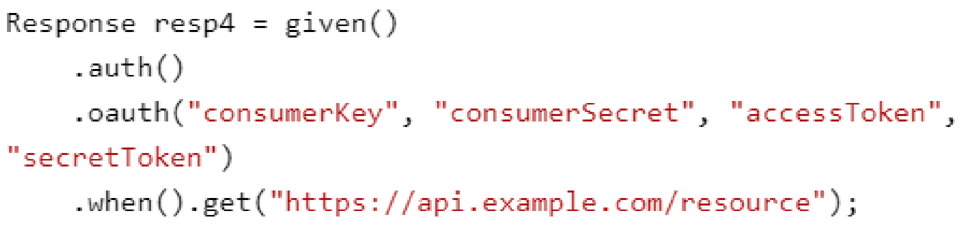


```
Response resp3 = given()
    .auth()
    .oauth2("access_token")
    .when().get("https://api.example.com/resource");
```

5. OAuth Authentication

OAuth1, often referred to simply as OAuth, is an earlier version of the OAuth protocol that focuses on granting third-party applications limited access to user resources on various online services. It’s designed to enable secure access to resources without the need for sharing the user’s actual credentials (username and password) with the third-party application.

OAuth1 authentication involves three main parties: the user (resource owner), the client application (consumer), and the resource server (service provider).



```
Response resp4 = given()
    .auth()
    .oauth("consumerKey", "consumerSecret", "accessToken",
"secretToken")
    .when().get("https://api.example.com/resource");
```

Through .oauth(), you provide the required credentials.