1. What is chaining in Rest Assured?

   Chaining in Rest Assured refers to the technique of linking multiple method calls together in a single statement to improve readability and conciseness. Each method returns an instance of a class that allows further method calls.

```
given()
   .header("Content-Type", "application/json")
   .body("{ \"name\": \"John\" }")
.when()
   .post("/users")
.then()
   .statusCode(201)
   .body("name", equalTo("John"));
```

   In the above example, given(), when(), and then() methods are chained together. The given() method sets up the request, when() sends it, and then() validates the response.

2. What is the role of assertions in API automation testing?

   Assertions are used to verify that the response received from an API matches the expected outcome. They ensure that the API behaves correctly and returns the correct data.

```
given()
.when()
   .get("/users/1")
.then()
   .statusCode(200)
   .body("name", equalTo("John"));
```

   Here, the statusCode(200) assertion checks if the response status code is 200, and body("name", equalTo("John")) verifies that the name in the response body is "John".

3. What is the difference between authentication and authorization?
   - Authentication: The process of verifying the identity of a user or system (e.g., using a username and password).
   - Authorization: The process of checking the permissions or access levels of an authenticated user.

   Example:

   - Authentication: Logging in with a username and password.
   - Authorization: Checking if the logged-in user has permission to access a specific resource.

4. What is OAuth1 and OAuth2, why is it used?

- **OAuth1**: An older, more complex protocol for authentication that involves cryptographic signing of requests. It is used for secure API authentication without sending user credentials.
- **OAuth2**: A simpler and more modern framework for authentication that provides tokens to allow applications to access user information without exposing user credentials. It supports different types of tokens and flows suitable for various applications.

5. If the server is not available for the request then what status code do you get?

   If the server is not available, you typically receive a 503 Service Unavailable status code. This indicates that the server is temporarily unable to handle the request.

6. What is the difference between String, StringBuilder, and StringBuffer? Which should be used when?
   - **String**: Immutable sequence of characters. Once created, the value cannot be changed.
   - **StringBuilder**: Mutable sequence of characters, not thread-safe. Suitable for single-threaded environments where frequent modifications are needed.
   - **StringBuffer**: Mutable sequence of characters, thread-safe. Suitable for multi-threaded environments where frequent modifications are needed.

   Use StringBuilder for better performance in single-threaded environments and StringBuffer in multi-threaded environments to ensure thread safety.

7. If there are a lot of transactions in the database, then which one of the above should be used?

   Use StringBuffer to ensure thread safety during concurrent transactions in a multi-threaded environment. It prevents data inconsistency by synchronizing methods.

8. How to create user-defined custom exceptions?

   Custom exceptions are created by extending the Exception class or any of its subclasses.

```
public class CustomException extends Exception {
   public CustomException(String message) {
      super(message);
   }
}
```

   You can throw and catch this exception like any other exception.

```
if (conditionFails) {
   throw new CustomException("Custom error message");
}
```

9. If the design pattern is singleton and you are using a parallel run, will it execute in parallel or not?

   Singleton pattern ensures that only one instance of a class is created. However, if you are using a singleton in a multi-threaded environment, you need to ensure thread safety using synchronization.

```
public class Singleton {
   private static Singleton instance;

   private Singleton() {}

   public static synchronized Singleton getInstance() {
      if (instance == null) {
         instance = new Singleton();
      }
      return instance;
   }
}
```

   Proper synchronization is required for safe parallel execution.

10. What is serialization in Rest Assured?

Serialization is the process of converting a Java object into a JSON or XML representation for sending it in the API request.

```
User user = new User("John", 30);

given()
   .contentType("application/json")
   .body(user)
.when()
   .post("/users")
.then()
   .statusCode(201);
```

11. Different ways to create a payload?

- As a JSON string
- Using a HashMap
- Using a POJO (Plain Old Java Object)

```
// JSON String
String jsonString = "{ \"name\": \"John\", \"age\": 30 }";

// HashMap
Map<String, Object> jsonMap = new HashMap<>();
jsonMap.put("name", "John");
```

```
jsonMap.put("age", 30);

// POJO
class User {
   private String name;
   private int age;

   // Constructors, getters, and setters
}
User user = new User("John", 30);
```

12. What are query parameters?

Query parameters are key-value pairs appended to the URL to pass additional data to the server.

```
given()
   .queryParam("search", "John")
.when()
   .get("/users");
```

In the URL /users?search=John, search is the query parameter.

13. Difference between PUT and PATCH?

- PUT: Replaces the entire resource with the new data provided.
- PATCH: Partially updates the resource with the new data provided.

```
// PUT request
given()
   .contentType("application/json")
   .body(new User("John", 31))
.when()
   .put("/users/1")
.then()
   .statusCode(200);

// PATCH request
given()
   .contentType("application/json")
   .body("{ \"age\": 31 }")
.when()
   .patch("/users/1")
.then()
   .statusCode(200);
```

14. How do you handle authentication and authorization in API automation testing?

```
// Basic Authentication
given()
   .auth()
```

```
    .basic("username", "password")
.when()
    .get("/protected");

// OAuth2
given()
    .auth()
    .oauth2("accessToken")
.when()
    .get("/protected");
```

Authentication and authorization are handled by setting the appropriate headers or tokens in the request.

15. What is JSON? How do you parse and validate JSON responses in API automation testing?

JSON (JavaScript Object Notation) is a lightweight data interchange format. In Rest Assured, you can parse and validate JSON responses using the `body()` method and JSONPath expressions.

```
given()
.when()
    .get("/users/1")
.then()
    .body("name", equalTo("John"))
    .body("age", equalTo(30));
```

16. How do you handle dynamic data in API responses during automation?

Extract dynamic data from the response and use it in subsequent requests.

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

String dynamicValue = response.path("dynamicField");

// Use the dynamic value in another request
given()
    .queryParam("value", dynamicValue)
.when()
    .get("/otherEndpoint");
```

17. What is endpoint testing, and how do you perform it?

Endpoint testing involves testing individual API endpoints to ensure they work as expected. This includes checking the status code, response body, headers, and other response details.

```
given()
.when()
```

```
    .get("/endpoint")
.then()
    .statusCode(200)
    .body("key", equalTo("value"));
```

18. How do you handle error responses and status codes in API automation testing?

Validate error responses and status codes to ensure the API handles errors correctly.

```
given()
.when()
    .get("/nonexistent")
.then()
    .statusCode(404)
    .body("error", equalTo("Not Found"));
```

19. What are some common challenges faced in API automation testing, and how do you overcome them?

- Dynamic data: Use response extraction and parameterization.
- Authentication: Implement proper authentication mechanisms.
- Environment differences: Use environment-specific configurations.
- Data consistency: Use database setup and teardown methods.

20. How do you perform a GET request using Rest Assured?

```
given()
.when()
    .get("/users/1")
.then()
    .statusCode(200)
    .body("name", equalTo("John"));
```

21. What is the significance of status codes in API responses?

Status codes indicate the result of the API request. For example:

- 200 OK: Request succeeded.
- 404 Not Found: Resource not found.
- 500 Internal Server Error: Server encountered an error.

22. How do you pass query parameters in a Rest Assured request?

```
given()
    .queryParam("key", "value")
.when()
    .get("/endpoint");
```

23. Explain how to validate JSON responses using Rest Assured.

```
given()
.when()
   .get("/users/1")
.then()
   .body("name", equalTo("John"))
   .body("age", equalTo(30));
```

24. How do you handle authentication in Rest Assured?

```
// Basic Authentication
given()
   .auth()
   .basic("username", "password")
.when()
   .get("/protected");

// OAuth2
given()
   .auth()
   .oauth2("accessToken")
.when()
   .get("/protected");
```

25. What is deserialization in Rest Assured?

Deserialization is converting a JSON or XML response back into a Java object.

```
User user = given()
.when()
   .get("/users/1")
.as(User.class);
```

26. How do you perform POST and PUT requests using Rest Assured?

```
// POST request
given()
   .contentType("application/json")
   .body(new User("John", 30))
.when()
   .post("/users")
.then()
   .statusCode(201);

// PUT request
given()
   .contentType("application/json")
   .body(new User("John", 31))
.when()
```

```
    .put("/users/1")
.then()
    .statusCode(200);
```

27. What is ResponseSpecBuilder in Rest Assured?

ResponseSpecBuilder is used to create reusable response specifications, which can be applied to multiple requests to enforce consistent validation rules.

```
ResponseSpecification responseSpec = new ResponseSpecBuilder()
    .expectStatusCode(200)
    .expectContentType(ContentType.JSON)
    .build();

given()
.when()
    .get("/users/1")
.then()
    .spec(responseSpec);
```

28. How can you extract data from a response using Rest Assured?

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

String name = response.path("name");
```

29. How do you handle dynamic values in API responses?

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

String dynamicValue = response.path("dynamicField");

// Use the dynamic value in another request
given()
    .queryParam("value", dynamicValue)
.when()
    .get("/otherEndpoint");
```

30. What are the benefits of using Rest Assured for API testing?

- Easy to use and readable syntax.
- Integration with popular testing frameworks like JUnit and TestNG.
- Supports various types of authentication.
- Provides detailed validation and assertion mechanisms.
- Enables seamless chaining of requests and responses.