REST Assured Request and Response Specs

In **Rest Assured**, **Request Specification** and **Response Specification** are used to eliminate duplication and standardize requests and responses in your API testing framework.

✅ RequestSpecification – Setting Up Request Details

A **`RequestSpecification`** helps you **configure common request settings** like base URI, headers, content type, authentication, and more.

🔹 Example: Create a RequestSpecification

```
java


import io.restassured.specification.RequestSpecification;
import static io.restassured.RestAssured.*;

RequestSpecification requestSpec = given()
    .baseUri("https://api.example.com")
    .basePath("/users")
    .header("Authorization", "Bearer your_token")
    .contentType("application/json");
```

You can now use **`requestSpec`** across multiple tests:

```
java


@Test
public void getUserTest() {
    given()
        .spec(requestSpec)
    .when()
        .get("/1")
    .then()
        .statusCode(200);
}
```

✅ ResponseSpecification – Setting Up Expected Response

A **`ResponseSpecification`** helps you **define expected status codes, response time, headers, etc.** consistently.

🔹 Example: Create a ResponseSpecification

```
java


import io.restassured.specification.ResponseSpecification;
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

ResponseSpecification responseSpec = expect()
    .statusCode(200)
```

```
    .contentType("application/json")
    .time(lessThan(2000L));  // response time < 2 sec
```

🔹 Use in a Test

```
java

@Test
public void validateUserResponse() {
    given()
        .spec(requestSpec)
    .when()
        .get("/1")
    .then()
        .spec(responseSpec)
        .body("name", equalTo("John Doe"));
}
```

✅ Reusability – Centralize in `@BeforeClass`

```
java

public class BaseTest {

    RequestSpecification requestSpec;
    ResponseSpecification responseSpec;

    @BeforeClass
    public void setup() {
        requestSpec = new RequestSpecBuilder()
                .setBaseUri("https://api.example.com")
                .setBasePath("/users")
                .addHeader("Authorization", "Bearer your_token")
                .setContentType(ContentType.JSON)
                .build();

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

✅ Summary Table

Specification Type	Purpose
`RequestSpecification`	Set common request config (base URI, headers)
`ResponseSpecification`	Define expected response (status, headers etc)

Would you like a **mini-project** showing multiple requests and tests using specifications?