Refactoring tests using Request Specification:

Refactoring means minimizing repitive codes thru Request Specification.

When do we use?

Consider,if we are writing many test cases,and we have same query params and headers in all the test cases,then we can write all those common scripts in init and add them to request spec.

Steps:

**public** **class** Requestspecification {

RequestSpecification spec;

@BeforeMethod

**public** **void** init() {

RestAssured.*baseURI* = "https://api.nytimes.com";

RestAssured.*basePath* = "/svc/books/v3";

RequestSpecBuilder builder = **new** RequestSpecBuilder();

builder.addQueryParam("api-key", "tdW5dbL1gMyF14kzX0dlLMeiekGphMdl");

builder.addQueryParam("format", "json");

spec = builder.build();

}

@Test

**public** **void** extractresponsebody() {

System.***out***.println("to print response body");

*given*().spec(spec)

// .queryParam("api-key", "tdW5dbL1gMyF14kzX0dlLMeiekGphMdl")

// .queryParam("format", "json")

.when()

.get("/lists/names").then().log().body();

//here,we have added all the queryparams in a RequestSpecBuilder and built it to create a RequestSpecification object in Beforemethod itself.

Now,we can access RequestSpecification object in all the test cases.In this way,we can avoid repitive codes in each test case.

Response Specification in Rest Assured:

With response specification,we can reduce code that we write to print headers and assertions that has to be checked for every test cases.

We can control all the repitive assertions in one place and access in all test cases with response specification.

@BeforeMethod

**public** **void** init() {

RestAssured.*baseURI* = "https://api.nytimes.com";

RestAssured.*basePath* = "/svc/books/v3";

ResponseSpecBuilder resbuilder = **new** ResponseSpecBuilder();

resbuilder.expectHeader("Content-Type", "application/json; charset=UTF-8");

resbuilder.expectHeader("Server", "Apache/2.2.15 (CentOS)");

resbuilder.expectStatusCode(200);

resbuilder.expectBody("num\_results", *equalTo*(55));

resbuilder.expectBody("results.list\_name", *hasItem*("Hardcover Fiction"));

spec = resbuilder.build();

}

@Test

**public** **void** extractresponsebody() {

System.***out***.println("to print response body");

*given*().queryParam("api-key", "tdW5dbL1gMyF14kzX0dlLMeiekGphMdl").queryParam("format", "json").when()

.get("/lists/names").then().spec(spec);

}

//here,add we can all the assertions with headers,direct parameter,array parameter and build it with responsespecification.