An introduction into Rest Assured & WireMock

BY RICK VAN OSCH & LOEK EHREN

Contents

- What is WireMock?
- What is Rest Assured?
- Why use them?
- Demos
- Exercises
- Discussion
- Questions

What is WireMock?

- Simulator for HTTP-based APIs
- Used within Java application, JUnit test, Servlet container or as standalone process.
- Open source
- http://wiremock.org

What is WireMock?

- In Java(Using JUnit):

```
@Rule
public WireMockRule wireMockRule = new WireMockRule();
```

- In Java(Without JUnit):

```
WireMockServer wireMockServer = new WireMockServer(options().port(8089));
wireMockServer.start();
```

- Standalone:

```
$ java -jar wiremock-standalone-2.8.0.jar
```

What is WireMock?

- Mocking responses in code...

```
stubFor(get(urlEqualTo("/some/thing"))
        .willReturn(aResponse()
            .withHeader("Content-Type", "text/plain")
            .withBody("Hello world!")));
```

- Fluent Interface
- Simple

What is WireMock?

- or JSON files(at runtime)
- POST to http://<host>:<port>/__admin/mappings
  or placed with a .json extension in src/test/resourc

```
{
    "request": {
        "method": "GET",
        "url": "/some/thing"
    },
    "response": {
        "status": 200,
        "body": "Hello world!",
        "headers": {
            "Content-Type": "text/plain"
        }
    }
}
```

What is WireMock?

```
stubFor(any(urlPathEqualTo("/everything"))
  .withHeader("Accept", containing("xml"))
  .withCookie("session", matching(".*12345.*"))
  .withQueryParam("search_term", equalTo("WireMock"))
  .withBasicAuth("jeff@example.com", "jeffteenjefftyjeff")
  .withRequestBody(equalToXml("<search-results />"))
  .withRequestBody(matchingXPath("//search-results"))
  .willReturn(aResponse()));
```

What is WireMock?

- Stateful behaviour to mocks
- More complex test cases

What is WireMock?

```
@Test
public void carScenario() {
    stubFor(get(urlEqualTo( testUrl: "/cars")).inScenario( s: "Car inventory")
            .whenScenarioStateIs(STARTED)
            .willReturn(aResponse()
                    .withBody("{ \"cars\": [ { \"id\": 0, \"name\": \"porsche\" } ] }")
                    .withHeader( key: "content-type",  ...values: "application/json")
                    .withStatus(200)));

    stubFor(post(urlEqualTo( testUrl: "/cars")).inScenario( s: "Car inventory")
            .whenScenarioStateIs(STARTED)
            .withRequestBody(containing( value: "Post car"))
            .willReturn(aResponse()
                    .withStatus(201)
                    .withHeader( key: "content-type",  ...values: "application/json"))
            .willSetStateTo("Car added"));

    stubFor(get(urlEqualTo( testUrl: "/cars")).inScenario( s: "Car inventory")
            .whenScenarioStateIs("Car added")
            .willReturn(aResponse()
                    .withBody("{ \"cars\": [ { \"id\": 0, \"name\": \"porsche\" }," +
                            "{ \"id\": 1, \"name\": \"ferrari\" } ] }")
                    .withStatus(200)
                    .withHeader( key: "content-type",  ...values: "application/json")));
```