Un poco de Java y +

Otra forma de hablar de nuestro día a día…

API, JAVA, MICROSERVICIOS, QUÉ ES, TESTING, UN POCO DE

¿Qué es WireMock?

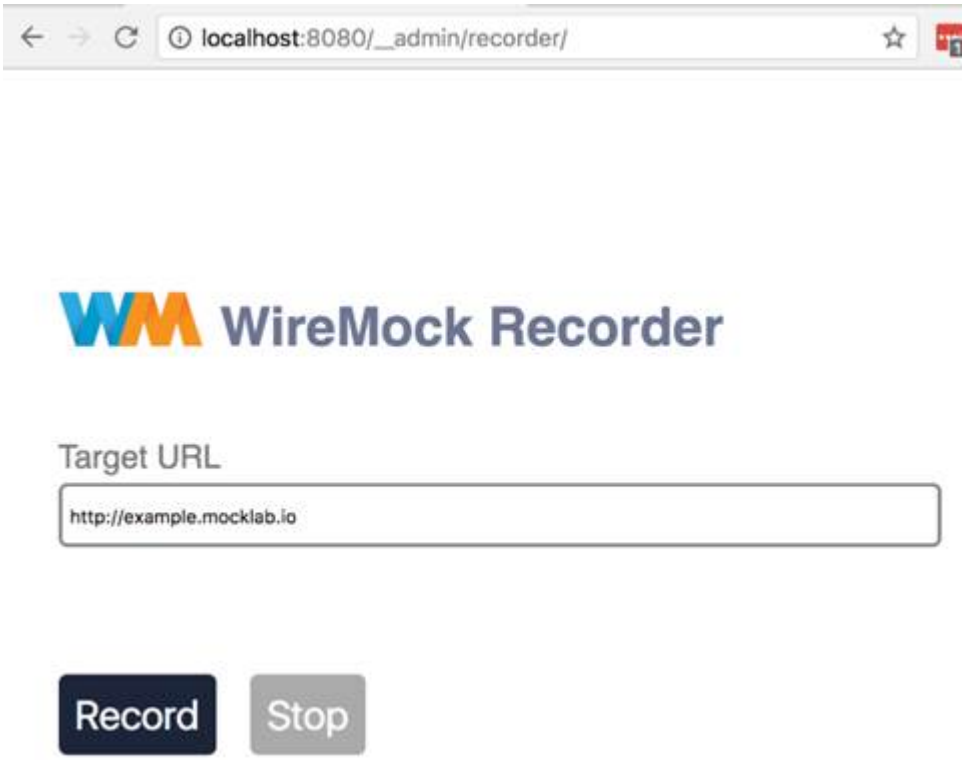
Fecha: 31 enero 2020 Autor/a: LuisMi Gracia 0 Comentarios
WireMock es un simulador de APIS REST HTTP, también podemos decir un **REST API Mock Server.**

WireMock puede usarse por ejemplo para avanzar en un desarrollo sobre una arquitectura de microservicios mientras un API no existe.

Soporte el test de los casos Edge y los errores que el API no debería generar.

Entre sus principales características tenemos:

- Stub de la respuesta HTTP macheable a nivel de patrones en la URL, header y body.
- Verificación de la petición
- Ejecución en Test Unitarios, como proceso standalone y como aplicación Web
- Configurable a través de API Java o con JSON.
- Grabación y reejecución de stubs: a partir de su UI podemos grabar el resultado de ejecución de un API para generar un stub (http://wiremock.org/docs/record-playback/)



- Inyección de errores
- Configurable a diferentes niveles: tiempos de delay, proxy condicional,…

Además está integrado con diferentes frameworks y tecnologías como Android, Junit o Spring Boot:

Ejecución con Junit: http://wiremock.org/docs/junit-rule/

```
@Rule
public WireMockRule wireMockRule = new WireMockRule(options().port(8888), false);
```

Ejecución con Spring Boot: https://cloud.spring.io/spring-cloud-contract/reference/html/project-features.html#features-wiremock

En este ejemplo se ve claramente cómo configurar un server WireMock y cómo hacer un stub que devuelva el body que queremos:

```java
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@AutoConfigureWireMock(port = 0)
public class WiremockForDocsTests {

    // A service that calls out over HTTP
    @Autowired
    private Service service;

    @Before
    public void setup() {
        this.service.setBase("http://localhost:"
                + this.environment.getProperty("wiremock.server.port"));
    }

    // Using the WireMock APIs in the normal way:
    @Test
    public void contextLoads() throws Exception {
        // Stubbing WireMock
        stubFor(get(urlEqualTo("/resource")).willReturn(aResponse()
                .withHeader("Content-Type", "text/plain").withBody("Hello World!")));
        // We're asserting if WireMock responded properly
        assertThat(this.service.go()).isEqualTo("Hello World!");
    }

}
```

Ejecución standalone: http://wiremock.org/docs/running-standalone/

```
$ java -jar wiremock-standalone-2.25.1.jar
```

Command line options

The following can optionally be specified on the command line:

`--port` : Set the HTTP port number e.g. `--port 9999` . Use `--port 0` to dynamically determine a port.

`--https-port` : If specified, enables HTTPS on the supplied port. Note: When you specify this parameter, WireMock will still, additionally, bind to an HTTP port (8080 by default). So when running multiple WireMock servers you will also need to specify the `--port` parameter in order to avoid conflicts.

Una herramienta muy útil!!!