**OverView:**

**WireMock** is a library for stubbing and mocking web services. It constructs a HTTP server that we could connect to as we would to an actual web service.

When a [WireMock](http://wiremock.org/) server is in action, we can set up expectations, call the service, and then verify its behaviors.

WireMock is simulator for HTTP-based Apis.Considered as a service virtualization tool or a mock server.

**Maven Dependencies**

<**dependency**>

<**groupId**>com.github.tomakehurst</**groupId**>

<**artifactId**>wiremock</**artifactId**>

<**version**>1.58</**version**>

<**scope**>test</**scope**>

</**dependency**>

**Programmatically Managed Server**

Manually we can configure a WireMock server. i.e. without the support of JUnit auto-configuration.

## ****JUnit Managed Server****

## A WireMock server can be integrated into JUnit test cases by using the @Rule annotation. This allows JUnit to manage the lifecycle, starting the server prior to each test method and stopping it after the method returns.

@Rule

**public** WireMockRule wireMockRule = **new** WireMockRule(**int** port);

If no arguments are supplied, server port will take the default value, *8080*. Server host, defaulting to *localhost*,

After setting up a WireMockRule instance, the next step is to configure a stub.

**We can don stub for:**

**URL Matching**

In this subsection, we will provide a REST stub for a service endpoint using regular expression:

stubFor(get(urlPathMatching("/google/.\*"))

.willReturn(aResponse()

.withStatus(200)

.withHeader("Content-Type", "application/json")

.withBody("\"testing-library\": \"WireMock\"")));

### **Request Header Matching**

### **Request Body Matching**

## Let's move on to creating an HTTP client, executing a request and receive a response

in last we can write verify/assert statement.