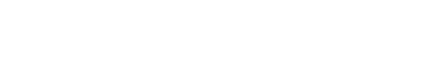
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | [**▼**](javascript:void(0)) | [**▼**](javascript:void(0)) | [**▼**](javascript:void(0)) |
|  |  | [Start H ere](https://www.baeldung.com/start-here) | [Courses](javascript:void(0)) | [G uides](javascript:void(0)) | [About](javascript:void(0)) |
|  |  |  |  |  |  |



REST APITesting w ith Cucum ber

Last m odi ed:O ctober 3, 2020



by baeldung REST [Testing](https://www.baeldung.com/category/testing/)



Ijustannounced the new ***Learn Spring*** course,focused on the fundam entals ofSpring 5

and Spring B oot2:

>>CHECK OUT THE COURSE



1.O verview

This tutorial gives an introduction to C ucum ber,a com m only used tool for user acceptance testing,and how to

use it in R EST A PItests.

In addition,to m ake the article self-contained and independent of any external R EST services,w e w ill use

* ireM ock,a stubbing and m ocking w eb service library.If you w ant to know m ore about this library,please refer to the introduction to W ireM ock.

2.G herkin – the Language of Cucum ber

C ucum ber is a testing fram ew ork that supports B ehavior D riven D evelopm ent (B D D ),allow ing users to de ne

application operations in plain text.It w orks based on the G herkin D om ain Speci c Language (D SL).This sim ple

but pow erful syntax of G herkin lets developers and testers w rite com plex tests w hile keeping it com prehensible

to even non-technical users.

2.1.Introduction to Gherkin

G herkin is a line-oriented language using line endings,indentations and keyw ords to de ne docum ents.Each

non-blank line usually starts w ith a G herkin keyw ord,follow ed by an arbitrary text,w hich is usually a description

of the keyw ord.

The w hole structure m ust be w ritten into a le w ith the ***fea ture*** extension to be recognized by C ucum ber.

H ere is a sim ple G herkin docum ent exam ple:



Feature: A short description of the desired functionality

Scenario: A business situation

Given a precondition

And another precondition

When an event happens

And another event happens too

Then a testable outcome is achieved

And something else is also completed

In the follow ing sections,w e'll describe a couple of the m ost im portant elem ents in a G herkin structure.

2.2.Feature

W e use a G herkin le to describe an application feature that needs to be tested.The le contains the ***Fea ture***

keyw ord at the very beginning,follow ed up by the feature nam e on the sam e line and an optional description

that m ay span m ultiple lines underneath.

C ucum ber parser skips all the text,except for the ***Fea ture*** keyw ord,and includes it for the purpose of

docum entation only.

2.3.Scenarios and Steps

* G herkin structure m ay consist of one or m ore scenarios,recognized by the ***S cena rio*** keyw ord.A scenario is basically a test allow ing users to validate a capability of the application.It should describe an initial context, events that m ay happen and expected outcom es created by those events.

These things are done using steps,identi ed by one of the



ve keyw ords:***G iven***,***W hen***,***Then***,***A nd*** ,and

***B ut***.

***G iven***:This step is to put the system into a w ell-de ned state before users start interacting w ith the

application.A ***G iven*** clause can by considered a precondition for the use case.



* ***hen***:A ***W hen*** step is used to describe an event that happens to the application.This can be an action takenby users,or an event triggered by another system .

***Then***:This step is to specify an expected outcom e of the test.The outcom e should be related to businessvalues of the feature under test.

***A nd*** and ***B ut***:These keyw ords can be used to replace the above step keyw ords w hen there are m ultiplesteps of the sam e type.



C ucum ber does not actually distinguish these keyw ords,how ever they are still there to m ake the feature m ore

readable and consistent w ith the B D D structure.

3.Cucum ber-JVM Im plem entation

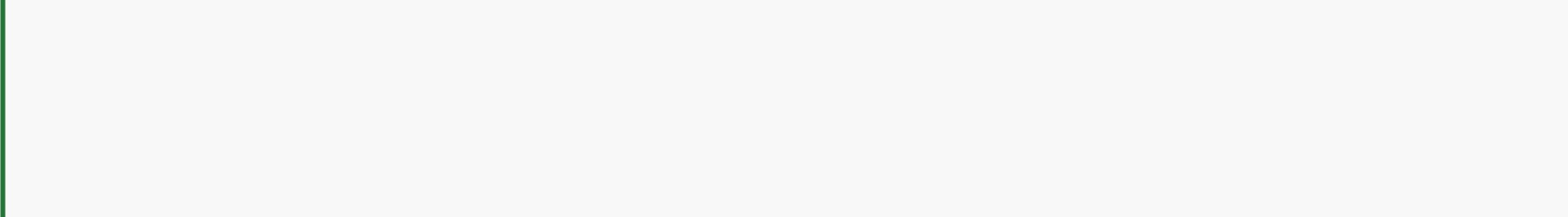
C ucum ber w as originally w ritten in R uby and has been ported into Java w ith C ucum ber-JV M im plem entation,

w hich is the subject of this section.

3.1.M aven Dependencies

In order to m ake use of C ucum ber-JV M in a M aven project,the follow ing dependency needs to be included in

the PO M :



<dependency>

<groupId>io.cucumber</groupId>

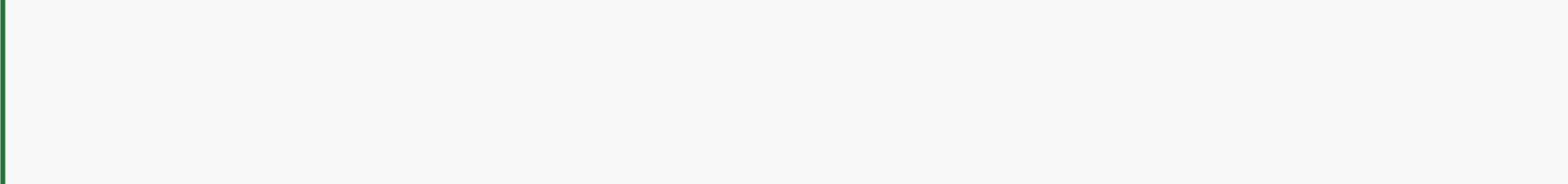
<artifactId>cucumber-java</artifactId>

<version>6.8.0</version>

<scope>test</scope>

</dependency>

To facilitate JU nit testing w ith C ucum ber,w e need to have one m ore dependency:



<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.8.0</version>

</dependency>

A lternatively,w e can use another artifact to take advantage of lam bda expressions in Java 8,w hich w on't be

covered in this tutorial.

3.2.Step De nitions

G herkin scenarios w ould be useless if they w ere not translated into actions and this is w here step de nitions

com e into play.B asically,a step de nition is an annotated Java m ethod w ith an attached pattern w hose job is to

convert G herkin steps in plain text to executable code.A fter parsing a feature docum ent,C ucum ber w ill search

for step de nitions that m atch prede ned G herkin steps to execute.

In order to m ake it clearer,let's take a look at the follow ing step:



Given I have registered a course in Baeldung

A nd a step de nition:



@Given("I have registered a course in Baeldung")

public void verifyAccount() {

// method implementation

}

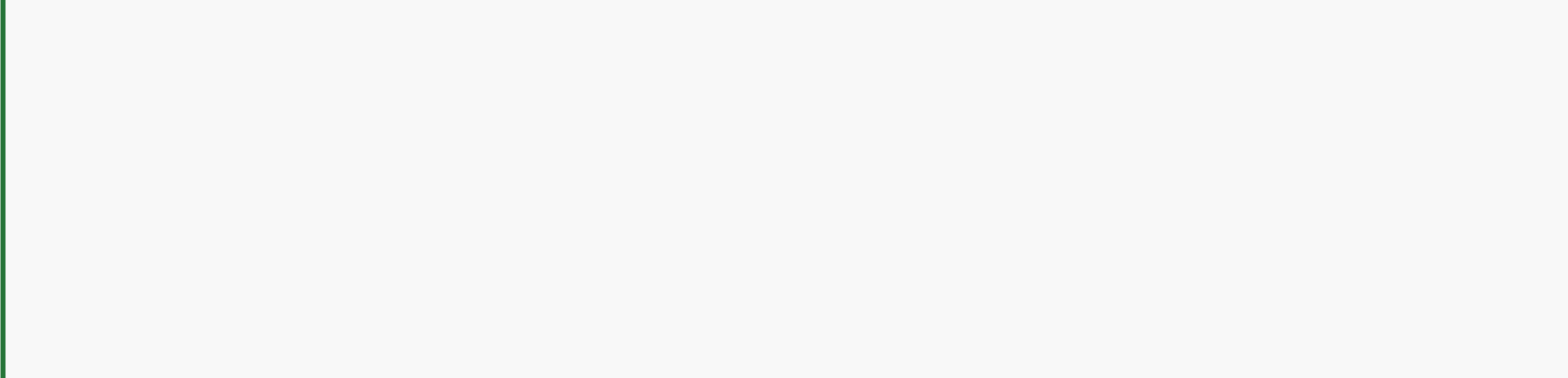
W hen C ucum ber reads the given step,it w ill be looking for step de nitions w hose annotating patterns m atch the

G herkin text.

4.Creating and Running Tests

4.1.W riting a Feature File

Let's start w ith declaring scenarios and steps in a le w ith the nam e ending in the ***.fea ture*** extension:



Feature: Testing a REST API

Users should be able to submit GET and POST requests to a web service,

represented by WireMock

Scenario: Data Upload to a web service

When users upload data on a project

Then the server should handle it and return a success status

Scenario: Data retrieval from a web service

When users want to get information on the 'Cucumber' project

Then the requested data is returned

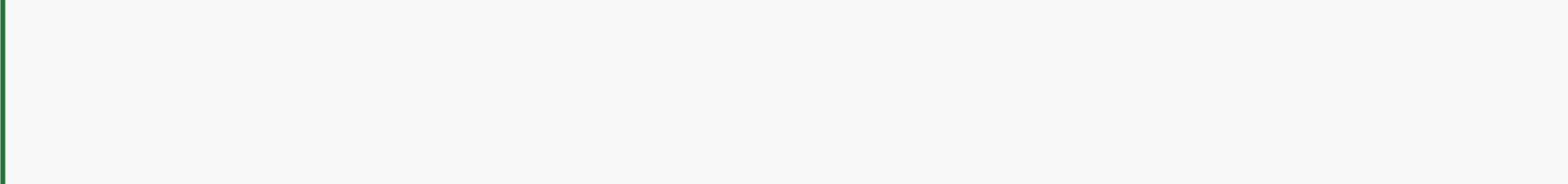
W e now save this le in a directory nam ed ***Fea ture***,on the condition that the directory w ill be loaded into the

classpath at runtim e,e.g.***src/m a in/resources***.

4.2.Con guring JUn it to W ork W ith Cucum ber

In order for JU nit to be aw are of C ucum ber and read feature les w hen running,the ***C ucum b er*** class m ust be

declared as the ***R unner***.W e also need to tell JU nit the place to search for feature les and step de nitions.



@RunWith(Cucumber.class)

@CucumberOptions(features = "classpath:Feature")

public class CucumberIntegrationTest {

}

A s you can see,the ***fea tures*** elem ent of ***C ucum b erO p tion*** locates the feature le created before.A nother

im portant elem ent,called ***g lue***,provides paths to step de nitions.H ow ever,if the test case and step de nitions are in the sam e package as in this tutorial,that elem ent m ay be dropped.

4.3.W riting Step De nitions

W hen C ucum ber parses steps,it w ill search for m ethods annotated w ith G herkin keyw ords to locate the

m atching step de nitions.

* step de nition’s expression can either be a R egular Expression or a C ucum ber Expression.In this tutorial,w e'll use C ucum ber Expressions.

The follow ing is a m ethod that fully m atches a G herkin step.The m ethod w ill be used to post data to a R EST w eb service:



@When("users upload data on a project")

public void usersUploadDataOnAProject() throws IOException {

}

A nd here is a m ethod m atching a G herkin step and takes an argum ent from the text,w hich w ill be used to get

inform ation from a R EST w eb service:



@When("users want to get information on the {string} project")

public void usersGetInformationOnAProject(String projectName) throws IOException {

}

A s you can see,the ***usersG etInform a tionO nA P roject*** m ethod takes a ***S tring*** argum ent,w hich is the project nam e.

This argum ent is declared by ***{string }*** in the annotation and over here it corresponds to ***C ucum b er*** in the step text.

A lternatively,w e could use a regular expression:



@When("^users want to get information on the '(.+)' project$")

public void usersGetInformationOnAProject(String projectName) throws IOException {

}

N ote,the ***‘^'***and ***‘$ '***w hich indicate the start and end of the regex accordingly.W hereas ***‘(.+)'***corresponds to the

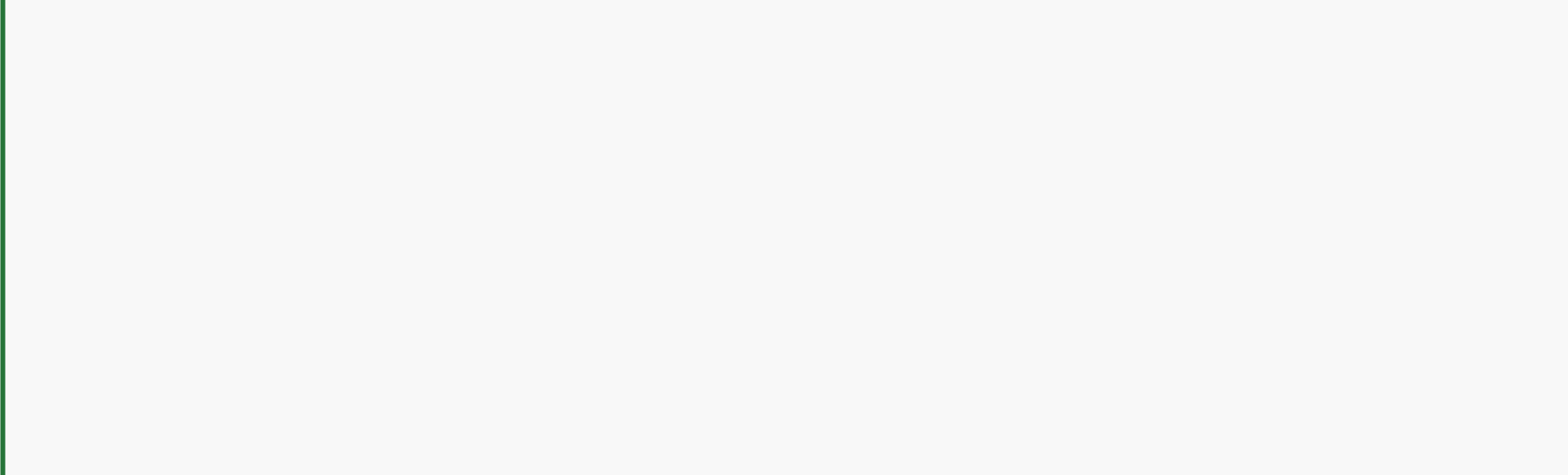
***S tring*** param eter.

W e'll provide the w orking code for both of the above m ethods in the next section.

4.4.Creating and Running Tests

First,w e w ill begin w ith a JSO N structure to illustrate the data uploaded to the server by a PO ST request,and

dow nloaded to the client using a G ET.This structure is saved in the ***jsonS tring*** eld,and show n below :



{

"testing-framework": "cucumber",

"supported-language":

[

"Ruby",

"Java",

"Javascript",

"PHP",

"Python",

"C++"

],

"website": "cucumber.io"

}

To dem onstrate a R EST A PI,w e use a W ireM ock server:



WireMockServer wireMockServer = new WireMockServer(options().dynamicPort());

In addition,w e'll use A pache H ttpC lient A PIto represent the client used to connect to the server:



CloseableHttpClient httpClient = HttpClients.createDefault();

N ow ,let's m ove on to w riting testing code w ithin step de nitions.W e w ill do this for the

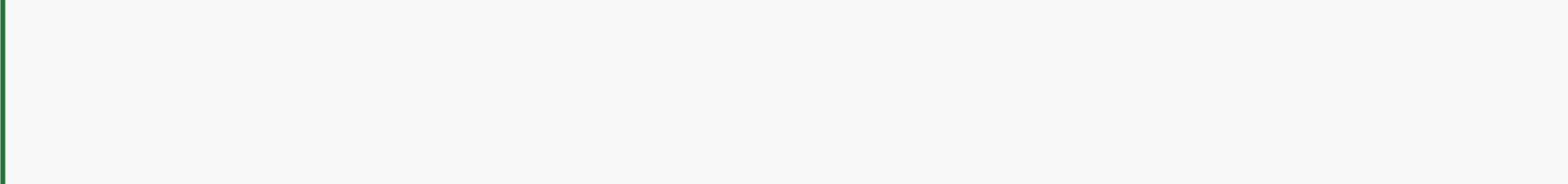
***usersU p loa d D a ta O nA P roject*** m ethod rst.

The server should be running before the client connects to it:



wireMockServer.start();

U sing the W ireM ock A PIto stub the R EST service:



configureFor("localhost", wireMockServer.port());

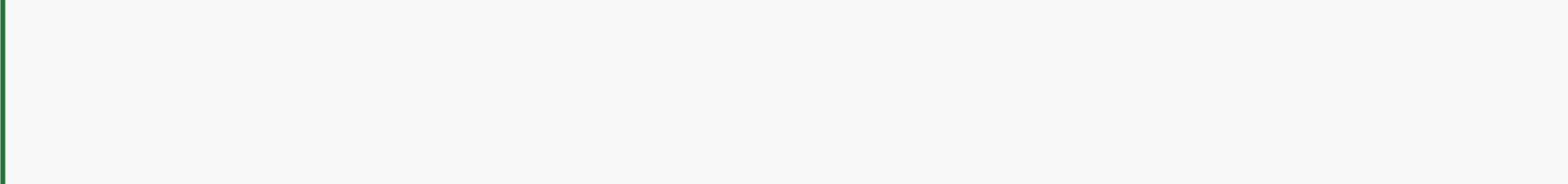
stubFor(post(urlEqualTo("/create"))

.withHeader("content-type", equalTo("application/json"))

.withRequestBody(containing("testing-framework"))

.willReturn(aResponse().withStatus(200)));

N ow ,send a PO ST request w ith the content taken from the ***jsonS tring*** eld declared above to the server:



HttpPost request = new HttpPost("http://localhost:" + wireMockServer.port() + "/create");

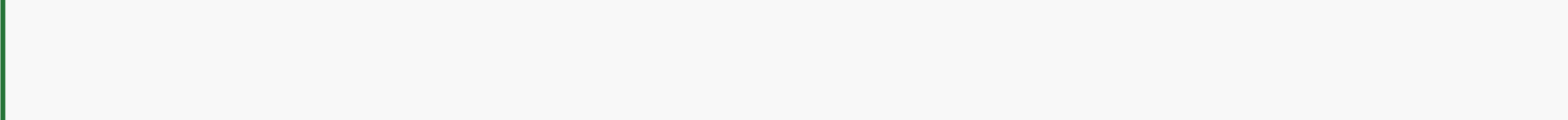
StringEntity entity = new StringEntity(jsonString);

request.addHeader("content-type", "application/json");

request.setEntity(entity);

HttpResponse response = httpClient.execute(request);

The follow ing code asserts that the PO ST request has been successfully received and handled:



assertEquals(200, response.getStatusLine().getStatusCode());

verify(postRequestedFor(urlEqualTo("/create"))

.withHeader("content-type", equalTo("application/json")));

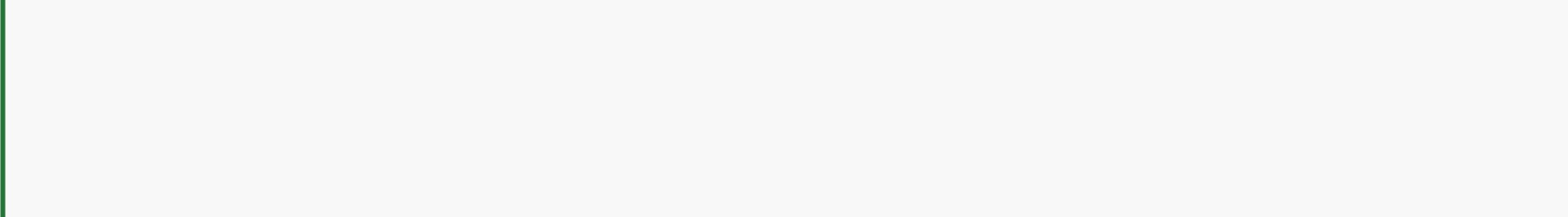
The server should stop after being used:



wireMockServer.stop();

The second m ethod w e w ill im plem ent herein is ***usersG etInform a tionO nA P roject(S tring p rojectN a m e)***.Sim ilar to the

rst test,w e need to start the server and then stub the R EST service:



wireMockServer.start();

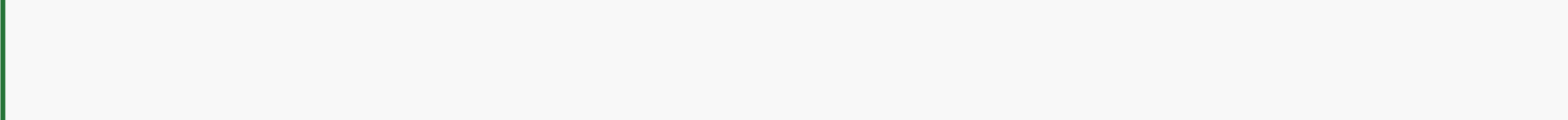
configureFor("localhost", wireMockServer.port());

stubFor(get(urlEqualTo("/projects/cucumber"))

.withHeader("accept", equalTo("application/json"))

.willReturn(aResponse().withBody(jsonString)));

Subm itting a G ET request and receiving a response:



HttpGet request = new HttpGet("http://localhost:" + wireMockServer.port() + "/projects/" + projectName.toLowerCase());

request.addHeader("accept", "application/json");

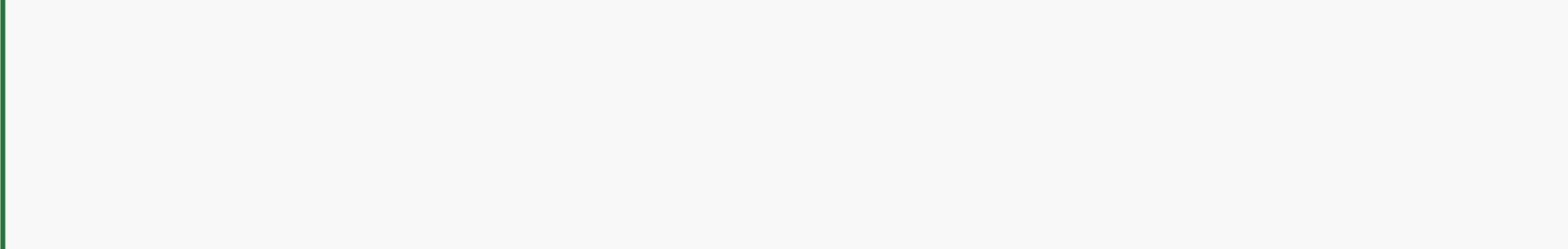
HttpResponse httpResponse = httpClient.execute(request);

W e w ill convert the ***http R esp onse*** variable to a ***S tring*** using a helper m ethod:



String responseString = convertResponseToString(httpResponse);

H ere is the im plem entation of that conversion helper m ethod:



private String convertResponseToString(HttpResponse response) throws IOException {

InputStream responseStream = response.getEntity().getContent();

Scanner scanner = new Scanner(responseStream, "UTF-8");

String responseString = scanner.useDelimiter("\\Z").next();

scanner.close();

return responseString;

}

The follow ing veri es the w hole process:



assertThat(responseString, containsString("\"testing-framework\": \"cucumber\""));

assertThat(responseString, containsString("\"website\": \"cucumber.io\""));

verify(getRequestedFor(urlEqualTo("/projects/cucumber"))

.withHeader("accept", equalTo("application/json")));

Finally,stop the server as described before.

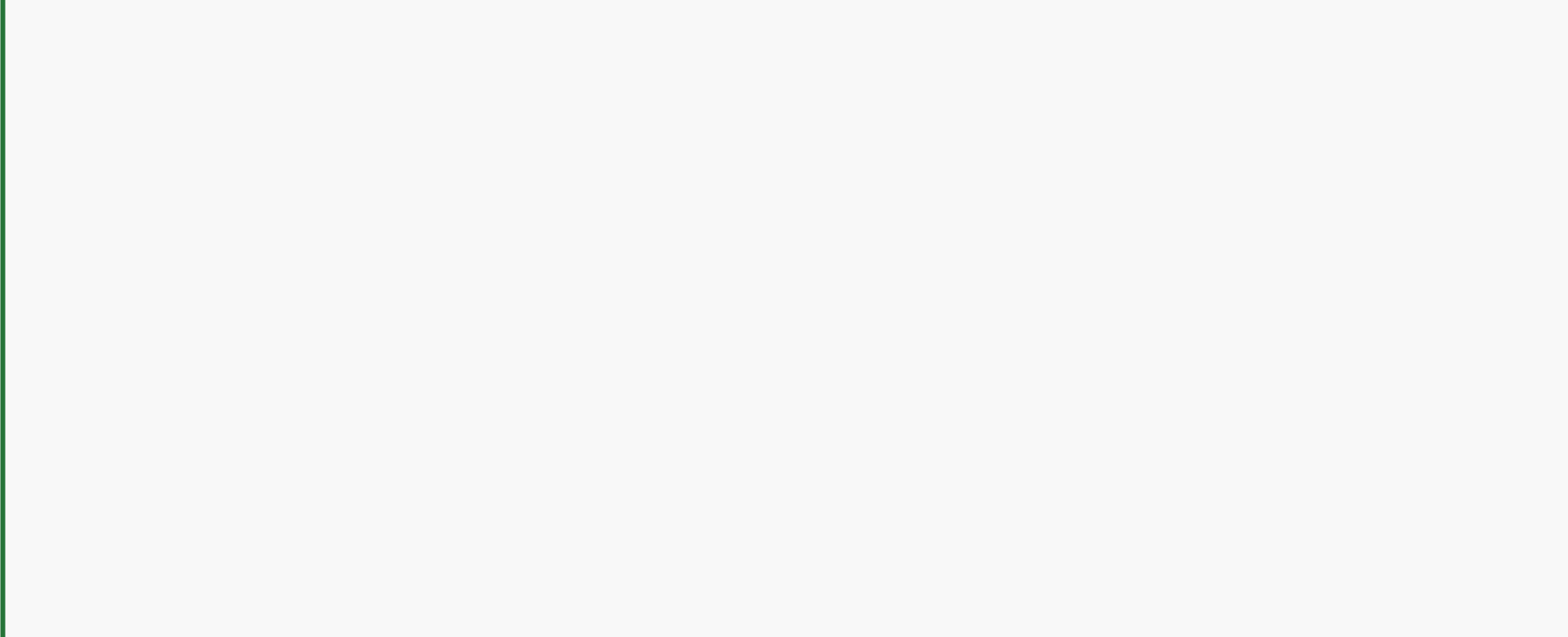
5.Running Features in Parallel

C ucum ber-JV M natively supports parallel test execution across m ultiple threads.W e'll use JU nit together w ith

M aven Failsafe plugin to execute the runners.A lternatively,w e could use M aven Sure re.

JU nit runs the feature les in parallel rather than scenarios,w hich m eans allthe scenarios in a feature le w illbe executed by the sam e thread.

Let's now add the plugin con guration:



<plugin>

<artifactId>maven-failsafe-plugin</artifactId>

<version>${maven-failsafe-plugin.version}</version>

<configuration>

<includes>

<include>CucumberIntegrationTest.java</include>

</includes>

<parallel>methods</parallel>

<threadCount>2</threadCount>

</configuration>

<executions>

<execution>

<goals>

<goal>integration-test</goal>

<goal>verify</goal>

</goals>

</execution>

</executions>

</plugin>

N ote that:



***p a ra llel:***can be ***cla sses, m ethod s***,or both – in our case,***cla sses*** w ill m ake each test class run in a separate

thread



***threa d C ount:***indicates how m any threads should be allocated for this execution

That's all w e need to do to run the C ucum ber features in parallel.

6.Conclusion

In this tutorial,w e covered the basics of C ucum ber and how this fram ew ork uses the G herkin dom ain-speci c

language for testing a R EST A PI.

A s usual,all code sam ples show n in this tutorial are available over on G itH ub.



Get started w ith Spring 5 and Spring Boot 2,through the ***Learn Spring*** course :

>>CHECK OUT THE COURSE



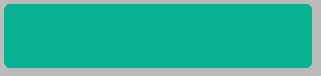
Build your M icroservice

Architecture w ith

Spring B oot and Spring C loud

|  |  |
| --- | --- |
| Enter your em ailaddress | D ow nload N ow |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3COMMENTS |  |  | | |  | Oldest | |  |
|  |  |  |  |  |  |  |  |



View Comments

C om m ents are closed on this article!



|  |  |  |
| --- | --- | --- |
| CATEG O RIES | SERIES | ABOUT |
| SPRIN G | JAVA “BACK TO BASICS”TU TO RIAL | ABOUT BAELDUNG |
| REST | JACKSO N JSO N TU TO RIAL | TH E CO U RSES |
| JAVA | H TTPCLIEN T 4 TU TO RIAL | JO BS |
| SECU RITY | REST W ITH SPRIN G TU TO RIAL | TH E FU LL ARCH IVE |
| PERSISTEN CE | SPRIN G PERSISTEN CE TU TO RIAL | ED ITO RS |
| JACKSO N | SECU RITY W ITH SPRIN G | O U R PARTN ERS |
| H TTP CLIEN T-SID E |  | AD VERTISE O N BAELD U N G |

TERM S O F SERVICE PRIVACY PO LICY CO M PAN Y IN FO CO N TACT