Demonstrates testing of common HTTP methods of a REST Web Service using Rest-Assured and Cucumber.

Below are the Maven dependencies to execute a testing solution using Rest-assured, Cucumber, Gson, and JUnit.

|  |  |
| --- | --- |
| Description | Maven Dependency |
| Rest-assured | <dependency>  <groupId>io.rest-assured</groupId>  <artifactId>rest-assured</artifactId>  <version>3.0.1</version>  <scope>test</scope>  </dependency> |
| Cucumber | <dependency>  <groupId>info.cukes</groupId>  <artifactId>cucumber-java</artifactId>  <version>1.2.4</version>  </dependency>  <dependency>  <groupId>info.cukes</groupId>  <artifactId>cucumber-picocontainer</artifactId>  <version>1.2.4</version>  </dependency>  <dependency>  <groupId>info.cukes</groupId>  <artifactId>cucumber-junit</artifactId>  <version>1.2.4</version>  </dependency> |
| Gson | <dependency>  <groupId>com.google.code.gson</groupId>  <artifactId>gson</artifactId>  <version>2.7</version>  </dependency> |
| JUnit | <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  </dependency> |

Cucumber uses feature files to specify use cases that describe a specific function of the software being tested. The feature files use the Gherkin language to define the test cases. The Gherkin syntax is designed to be non-technical and human readable in effort to promote business driven development practices across an entire development team. A feature file is separated into three parts: Features, Scenarios, and Steps. The Features describe specific function of the software being tested. Each feature is made of a collection of scenarios and each Scenario is defined by a sequence of Steps. Using java annotations, the Cucumber Steps associate with JUnit tests.

Below is the defined feature file for this application: restAssured.feature.

Feature: To test the JSONPlaceholder REST API using RestAssured.

@getRequest

Scenario: Test the Get REST method.

Given Test the Get REST method.

| Field | Value |

| url | http://jsonplaceholder.typicode.com/ |

| resource | users/ |

| id | 1 |

@postRequest

Scenario Outline: Test the Post REST method.

Given Test the Post REST method by <value>

| Field | Value |

| url | http://jsonplaceholder.typicode.com/ |

| resource | posts/ |

| userID | 1 |

| id | 101 |

| title | new title |

| body | new body |

| content type | application/json |

| json schema | posts-schema.json |

# Post REST Request by Object Mapping or JSON String

Examples:

| value |

| Object Mapping |

| JSON String |

@putRequest

Scenario: Test the Put REST method.

Given Test the Put REST method.

| Field | Value |

| url | http://jsonplaceholder.typicode.com/ |

| resource | posts/ |

| userID | 1 |

| id | 1 |

| title | new title |

| body | new body |

| content type | application/json |

| json schema | posts-schema.json |

@deleteRequest

Scenario: Test the Delete REST method.

Given Test the Delete REST method.

| Field | Value |

| url | http://jsonplaceholder.typicode.com/ |

| resource | posts/ |

| id | 1 |

| content type | application/json |

Below are highlights of some functionality as demonstrated in the [RestAssured](https://github.com/gdombchik/RestAssured) GitHub repository. The Cucumber feature and JSON schema files are located in the src/test/resources source folder. The JUnit files can be located in the com.cucumber package.

|  |  |  |
| --- | --- | --- |
| Functionality | Description | Java Files |
| CucumberOptions | CucumberOptions specifies JUnit configuration items such as location of Cucumber feature files, Cucumber reporting, and Cucumber test suites. | RunnerTest.java |
| Test Rest Web Service | Test the HTTP Get, Post, Put, and Delete Methods of a Rest Web Service. The Post Method has been tested using two different approaches: By Object Mapping and By JSON String. The REST Web Service is provided by JSONPlaceholder (http://jsonplaceholder.typicode.com/) | *Cucumber Feature File:*  restAssured.feature  *Test File:*  RestAssuredTest.java  *Data Transfer Object (DTO):*  PostsDto.java |

In reviewing how to run the Java tests using a continuous integration (ci) tool such as Jenkins, an option is to use Maven to run the Selenium tests using a headless environment using tools such as PhantomJS.

The steps to install Jenkins, Java, Maven, and Git on AWS E2 Ubuntu server: <https://gist.github.com/jsuwo/9038610>.

The Jenkins project setup for a Java project in GitHub. This example Jenkins has been installed on an AWS E2 Ubuntu server.

|  |  |
| --- | --- |
| Description | Command |
| *AWS E2 Ubuntu server* | |
| Switch to the Jenkins user. | $ sudo su – jenkins |
| Create public private key pair. | $ cd .ssh  $ ssh-keygen -t dsa |
| Cat the public key and copy the results. | $ cat RestAssured.pub |
| *GitHub* | |
| Add and paste deploy key in GitHub project. | <https://github.com/gdombchik/RestAssured/settings/keys> |
| *Manage Jenkins – Configure System* | |
| Jenkins Location | Scroll to the “Jenkins Location” section.  Jenkins URL:  <http://ec2-52-25-64-69.us-west-2.compute.amazonaws.com/>  System Admin e-mail address:  [greg@gregorydombchik.com](mailto:greg@gregorydombchik.com) |
| E-mail Notification | Scroll to the “E-mail Notification” section.  SMTP server:  mail.gregorydombchik.com |
| *Jenkins Server – Global Tool Configuration* | |
| Maven Installation | Scroll to the “Maven” section.  Select “Maven installation…” button.  Name:  Apache Maven 3.0.5  MAVEN\_HOME:  /usr/share/maven/ |
| JDK Installation | Scroll to the “JDK” section.  Select “JDK installation…” button.  Name:  Oracle JDK 1.7  JAVA\_HOME:  /usr/lib/jvm/java-7-oracle/ |
| *Jenkins Server – New Project* | |
| Add A Jenkins Project for a Java project. | Select New Item.  Enter an item name.  Select Freestyle project.  Select Ok. |
| Specify the URL of the remote GitHub repository. | Scroll to the “Source Code Management” section.  Select the “Git” option.  Repository URL:  https://github.com/gdombchik/RestAssured.git |
| Update the Poll SCM. | Scroll to the “Build Triggers” section.  Select the “Poll SCM” option.  Enter the following in the “Schedule” text box:  H \*/3 \* \* \* |
| Update the Build. | Scroll to the “Build” section.  Select from the “Add build step” and select the “Invoke top-level Maven targets” option.  Maven Version:  (Default)  Goals:  clean install  POM:  pom.xml |
| Update E-mail Notification. | Scroll to the Post-build Actions.  Select from the “Add post-build action” and select the “E-mail Notification” option.  Enter email address in the “Recipients” text box:  [greg@gregorydombchik.com](mailto:greg@gregorydombchik.com) |
| *GitHub* | |
| Add Jenkins GitHub Plugin you can automatically trigger build jobs when  pushes are made to GitHub. | <https://github.com/gdombchik/RestAssured/settings/installations>  Select from the “Add service” and select the “Jenkin’s (Git plugin)” option. (NOT GITHUB PLUGIN. This service does not appear to work).  Enter the following in the “Jenkins url” text box:  <http://ec2-52-25-64-69.us-west-2.compute.amazonaws.com/> |

