Load Testing using Gatling and JUnit in Scala

Overview

Gatling is tool for load/stress testing that can be used to create stress tests in websites.

In this tutorial, we will create a Gatling simulation containing some basic HTTP interaction and we evaluate the obtained results in terms of performance and failure requests, using Gatling's assertions.

Gatling v2.2 release added support for JUnit reports, which can be processed by Xray.

Description

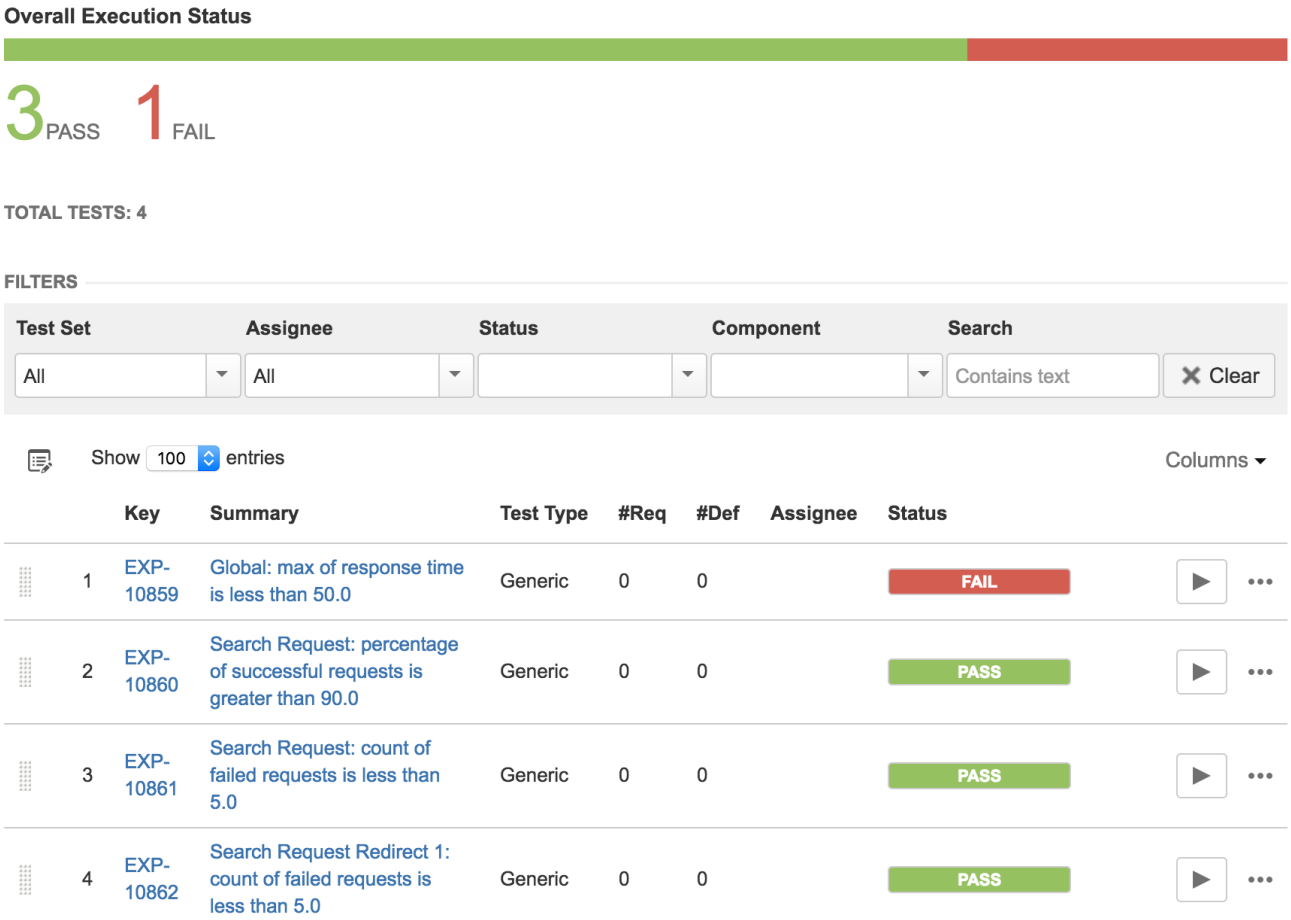
Below is the simulation class containing 3 assertions. This code was directly run inside Gatling's folder after extracting the v2.2.3 release Gatling bundle.

user-files/simulations/http/GoogleSimulation1.scala

```
package xpandaddons.xray.gatling.simulations

import io.gatling.core.Predef._
import io.gatling.http.Predef._
import io.gatling.http.request.builder.HttpRequestBuilder.toActionBuilder

class GoogleSimulation1 extends Simulation {
    val theHttpProtocolBuilder = http
        .baseURL("http://www.google.com")

    val theScenarioBuilder = scenario("GoogleTest")
        .exec(
            http("Search Request")
                .get("/search?q=x")

                /*
                 * Check the response of this request. It should be a HTTP status 200.
                 * Since the expected result is 200, the request will be verified as being OK
                 * and the simulation will thus succeed.
                 */

                .check(
                    status.is(200)
                )
        )

    setUp(
        theScenarioBuilder.inject(atOnceUsers(10))
    )

        /*
         * This asserts that, for all the requests in all the scenarios in the simulation
         * the maximum response time should be less than 50 ms.
         * If this is not the case when the simulation runs, the simulation will considered to have failed.
         */

        .assertions(
            global.responseTime.max.lt(50),
            forAll.failedRequests.count.lt(5),
            details("Search Request").successfulRequests.percent.gt(90)
        )
        .protocols(theHttpProtocolBuilder)
}
```

After successfully running the Test Case and generating the JUnit XML report, it can be imported to Xray (by using the either the REST API or the **Import Execution Results** action within the Test Execution).

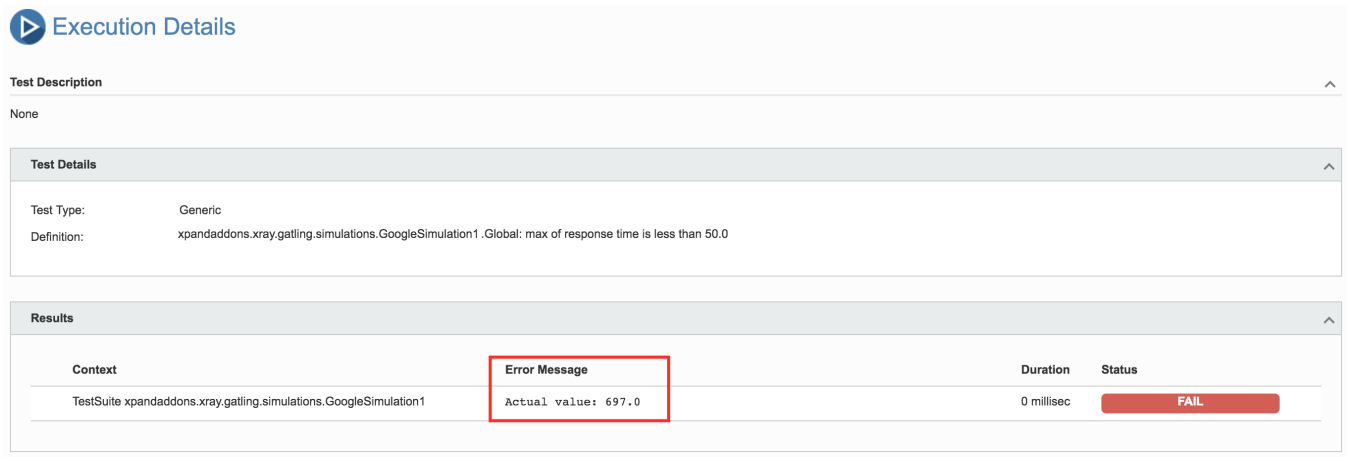
```
bin/gatling.sh --simulation xpandaddons.xray.gatling.simulations.GoogleSimulation1
```



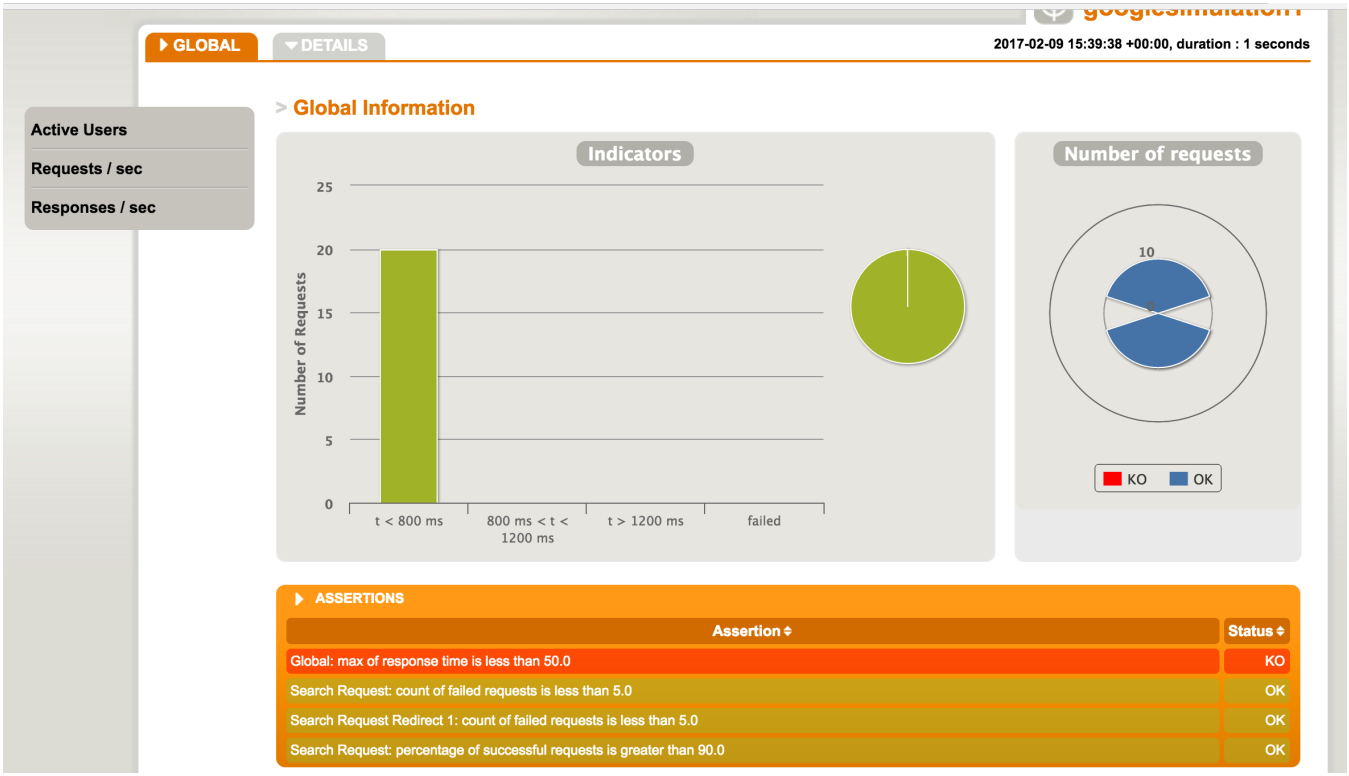
Every Gatling's assertion is mapped to a Generic Test in Jira, as is the initial check for the HTTP status code. The **Generic Test Definition** field contains the full name of the class.

The Execution Details of the Generic Test contains information about the Test Suite, which in this case corresponds to the full name of the simulation class.

When an assertion fails, then the actual value is displayed in the execution details.



As a side note, Gatling also generates an HTML report that provides many interesting and useful information in the context of load testing, including some interactive charts.



References

- http://gatling.io/docs/2.2.3/general/assertions.html?highlight=junit
- https://www.ivankrizsan.se/2016/05/06/introduction-to-load-testing-with-gatling-part-4/