Kickstart Your Gatling Performance Testing

Siegfried Goeschl

Version 1.0.0, 2018-11-04

Introduction

Siegfried Goeschl

- Senior Software Engineer
- Writing server-side code
- Java Meetup Vienna co-organizer
- Apache Software Foundation member
- Currently working at Erste Bank Austria

Introducing Gatling

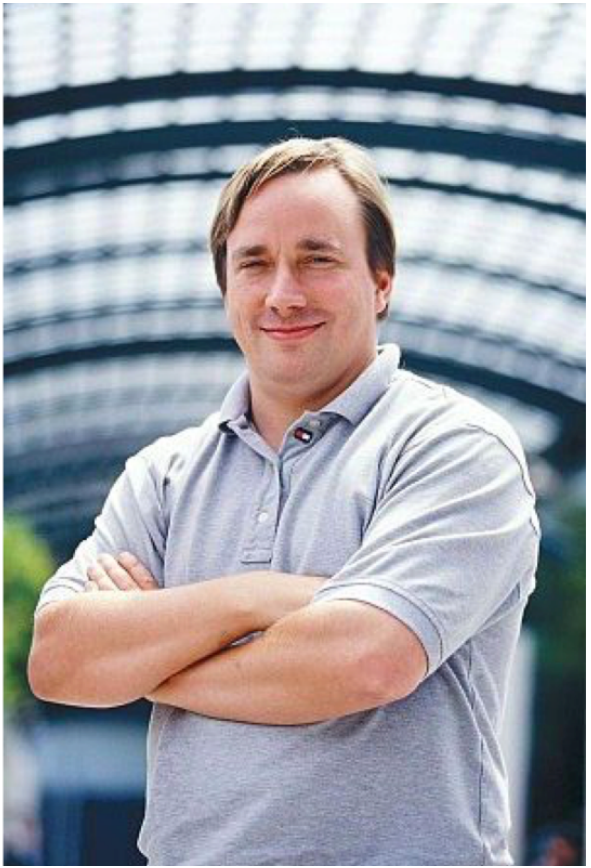
- Performance testing framework
- Tests are written in in Scala
- Developer-centric test tool
- Development started in 2010
- Gatling 3.0.1 released now

> - Since V3 there are two license models - free & commercial.
> - As you know some guys have a strong opinion about OSS

What Linus Says

Software is like sex, it’s better when it's free.

Linus Torvalds



- Having said that a commercial license could generate more revenue keeping the Open Source version alive.

Money Makes The World Go Round

Gatling vs. FrontLine

- Gatling Open Source is under ASL 2.0
- Gatling FrontLine is the enterprise edition
  - Annual license or "pay as you go"
  - Web-based,
  - More bells & whistle
  - Real-time reporting

Under The Hood

- Supports HTTP 1.1/2.0 & JMS protocol
- Response validation
  - Regular expressions
  - XPath & JSONPath
  - CSS selectors

Under The Hood

- Provides Domain Specific Language (DSL)
- Uses asynchronous non-blocking HTTP client
- Integrates with Maven, SBT & Gradle
- Test data feeders CSV, JSON, JDBC, Redis
- Management-friendly HTML reports

> - No more 1:1 mapping between virtual users and worker threads.

When To Use Gatling?

- Want to write test code in your IDE?
- Need some integration & performance tests?
- Want to run those test on your CI server?
- Do you care about reviews and version control?

Getting Started

Getting Started

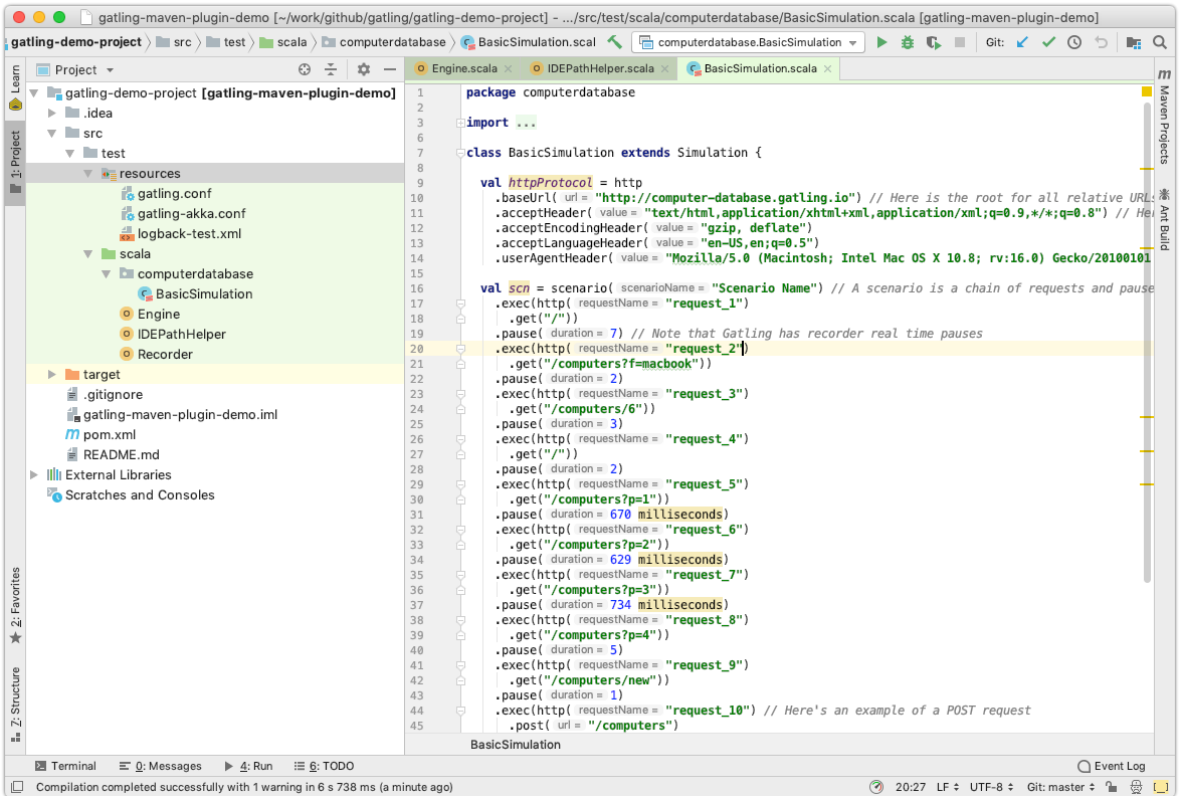
- JDK 1.8
- Apache Maven 3.5.x
- IntelliJ Community Edition
- IntelliJ Scala Plugin

Getting Started

- https://github.com/gatling/gatling-maven-plugin-demo
- Import the Maven project into your IDE
- Write and debug Scala code there
- Execute Gatling tests on the command line
- Simple CI integration using Maven

- The official Gatling distributable is not suited for development.
- The Gatling Maven archetype project does not use Maven Gatling plugin.
  - You can also use SBT & Gradle if you know your way around.

IntelliJ & Gatling



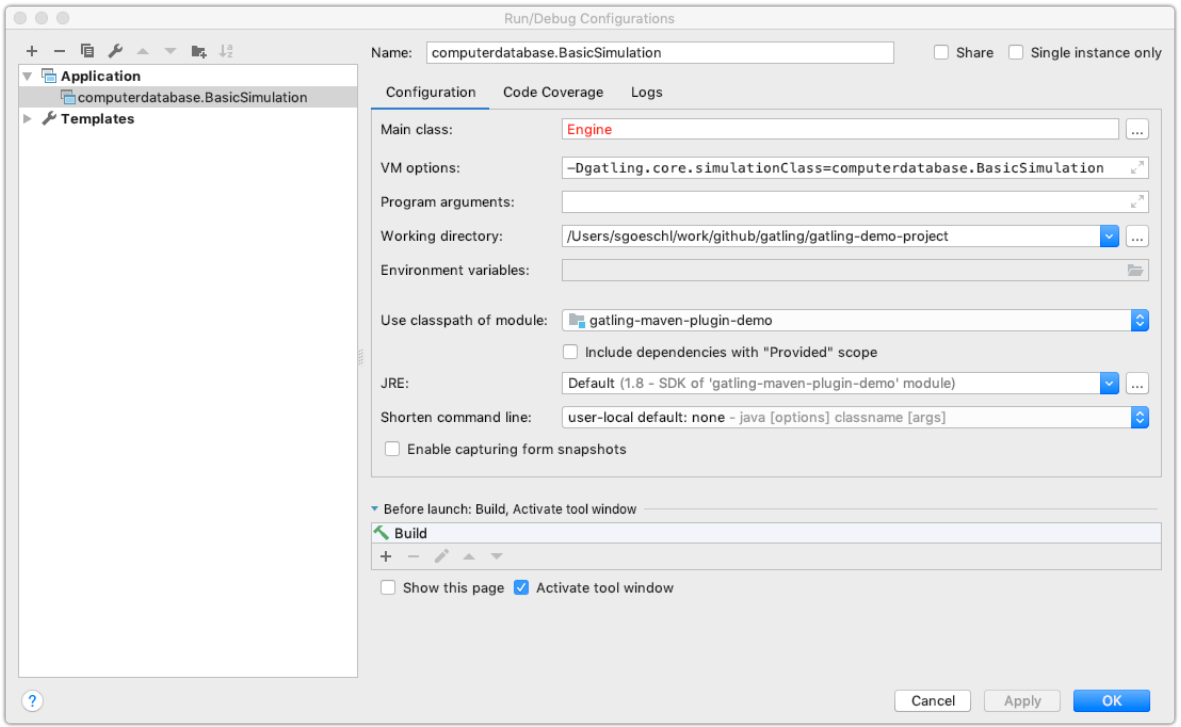
- That is what you see when you import the `gatling-maven-plugin-demo` into IntelliJ.

Gatling Run Configuration

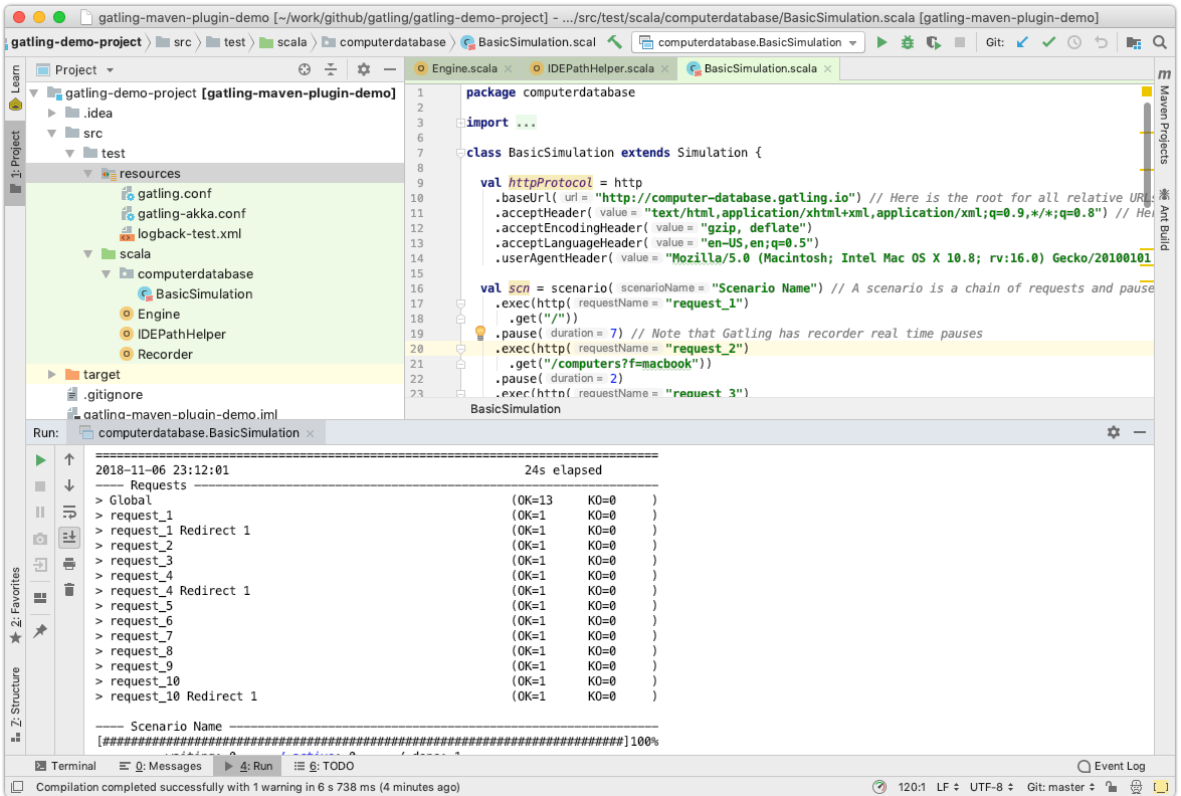
Parameter	Value
Main Class	Engine
VM Options	-Dgatling.core.simulationClass=XXX

- You need to tell IntelliJ which Gatling tests to execute ....

Gatling Run Configuration



Execute Gatling in IntelliJ



- The output of the pre-packaged demo project (computer database)

First Gatling Report

- The report is generated in the `target/gatling` folder

Execute Gatling Wit Maven

```
mvn -Dgatling.simulationClass=computerdatabase.BasicSimulation gatling:test
```

- Start Gatling from the Maven command line.
- Please note that different system properties are used!!!
- Perfect way to integrate with Jenkins

Hello World

Gatling Hello World

```
package postman

import io.gatling.core.Predef._
import io.gatling.http.Predef._

class HelloWorldSimulation extends Simulation {

  val httpProtocol = http.baseUrl("https://postman-echo.com")

  val scn = scenario("Hello World")
    .exec(http("GET").get("/get?msg=Hello%20World"))

  setUp(scn.inject(atOnceUsers(1)).protocols(httpProtocol))
}
```

Gatling For Rookies

- Script setup
- Common HTTP configuration
- Scenario & load simulation setup
- Load simulation text report
- Creating Gatling scripts

Script Setup

```
package postman

import io.gatling.core.Predef._
import io.gatling.http.Predef._

class PostmanSimulation extends Simulation {
```

- Gatling tests are deriving from `Simulation`

Common HTTP Configuration

```
val httpProtocol = http
    .baseUrl("https://postman-echo.com")
    .acceptHeader("text/html,application/xhtml+xml,;q=0.9,*/*;q=0.8")
    .acceptEncodingHeader("gzip, deflate")
    .acceptLanguageHeader("en-US,en;q=0.5")
    .userAgentHeader("Gatling/3.0.0")
```

Scenario Setup

```
val scn = scenario("Postman")
    .exec(http("GET")
      .get("/get?msg=Hello%20World")
      .check(bodyBytes.transform(_.length > 200).is(true))
    )
    .exec(http("POST")
      .post("/post")
      .formParam("""foo""", """bar""")
      .check(status in (200, 201))
      .check(bodyBytes.exists)
    )
```

Load Simulation Setup

At Once User

```
setUp(
  scn.inject(
    atOnceUsers(10) ①
  ).protocols(httpConf)
)
```

① Injects a given number of users at once

Rampup Users

```
setUp(
  scn.inject(
    rampUsers(10) over(5 seconds) ①
  ).protocols(httpConf)
)
```

① Start 10 user within 5 seconds ⇒ 10 users

Constant Users

```
setUp(
  scn.inject(
    constantUsersPerSec(20) during(15 seconds) ①
  ).protocols(httpConf)
)
```

① Start 20 users / second for 15 seconds ⇒ 300 users

Heaviside Users

```
setUp(
  scn.inject(
    heavisideUsers(1000) over(20 seconds) ①
  ).protocols(httpConf)
)
```

① Create 1.000 users in 20 seconds using Heaviside step function

Response Time Assertions

```
setUp(scn)
  .assertions(global.responseTime.max.lt(100)) ①
```

① Max response time of all requests is less than 100 ms

Simulation Text Report

```
================================================================
2018-11-16 20:43:51                          2s elapsed
---- Requests ------------------------------------------------
> Global                                   (OK=2      KO=0     )
> GET                                      (OK=1      KO=0     )
> POST                                     (OK=1      KO=0     )

---- Postman -------------------------------------------------
[############################################################]100%
       waiting: 0      / active: 0      / done: 1
================================================================
```

Simulation Text Report

```
---- Global Information --------------------------------------
> request count                          2 (OK=2      KO=0     )
> min response time                    118 (OK=118    KO=-     )
> max response time                    604 (OK=604    KO=-     )
> mean response time                   361 (OK=361    KO=-     )
> std deviation                        243 (OK=243    KO=-     )
> response time 50th percentile        361 (OK=361    KO=-     )
> response time 75th percentile        483 (OK=483    KO=-     )
> response time 95th percentile        580 (OK=580    KO=-     )
> response time 99th percentile        599 (OK=599    KO=-     )
> mean requests/sec                      2 (OK=2      KO=-     )
---- Response Time Distribution ------------------------------
> t < 800 ms                             2 (100%)
> 800 ms < t < 1200 ms                   0 (  0%)
> t > 1200 ms                            0 (  0%)
> failed                                 0 (  0%)
================================================================
```

Creating Gatling Scripts

- Gatling Web Proxy Recorder
- Start from the scratch
  - More initial work
  - Clean test code
- Import HTTP Archive Format

> - Since I’m testing REST APIs I’m crafting my Gatling scripts from the documentation.

Beyond Hello World

Things Not Being Told In Tutorials

Please Note That The Following Problems Are Not Specific To Gatling!

Hard-coded Server Address

```
val httpConf = http
    .baseURL("http://computer-database.gatling.io") ①
    .acceptHeader("text/html,application/xhtml+xml,application/xml")
    .doNotTrackHeader("1")
    .acceptLanguageHeader("en-US,en;q=0.5")
    .acceptEncodingHeader("gzip, deflate")
    .userAgentHeader("Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:16.0)")
```

① Need to support staging environments?

- You might start writing your test hitting you local box or DEV environment.
- later you want to switch to FAT, UAT & PROD.

Hard-coded CSV Files

```
val feeder = csv("users.csv").random ①
```

① Different users / passwords for staging environments?

Hard-coded User Injection

```
setUp(
  users.inject(rampUsers(10) over (10 seconds)), ①
  admins.inject(rampUsers(2) over (10 seconds))
).protocols(httpConf)
```

① Different load for staging environments?

- Your load testing environment might be a lot smaller than PROD.

Configuration Overload

```
Http(getURL("identity", "oauth/token"))
    .postForm(Seq(
        "scope" -> identityScope,          ①
        "grant_type" -> identityGranType,
        "client_id" -> identityClientId,
        "client_secret" -> identityClientSecret,
        "resource" -> identityResource
      ))
```

① Tons of configurable properties?

- How to pass the configuration properties which might be dependent on your staging environment?

How To Pass Settings

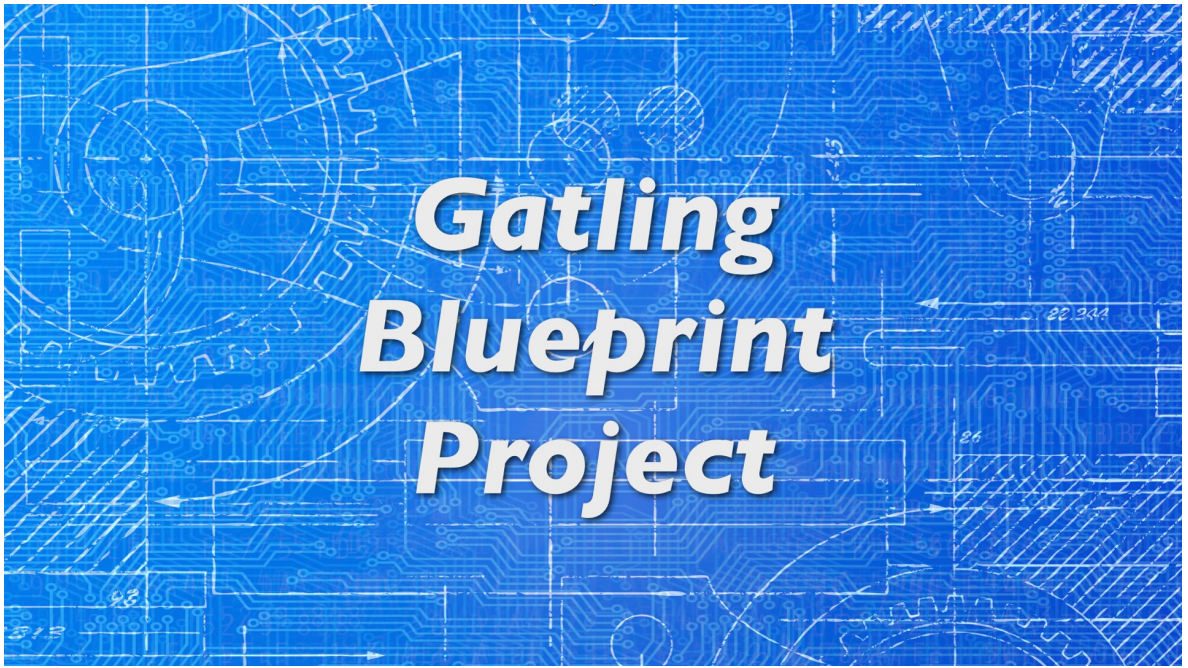

THERE'S MORE THAN ONE WAY
TO SKIN A CAT
memegenerator.net

How To Pass All The Settings

- System properties
- Maven profiles
- Custom Scala class

> - Unhappy with those approaches
> - I came up with Gatling Blueprint Project

Gatling Blueprint Project



Gatling Blueprint Project

- https://github.com/sgoeschl/gatling-blueprint-project
- Staging & multi-tenant support
- Hierarchical configuration & file resolver
- Pretty-printing & filtering of JSON responses
- Stand-alone Gatling distribution
- Implementing some best practices

> - Gatling Blueprint Project - a recipe of how to do things with Gatling
> - Use it like a cooking recipe - try it and change it to your personal taste

Simulation Coordinates

Simulation Coordinates

Tenant	The tenant to test (AT, CZ, SK)
Application	Application to simulate (web, mobile)
Site	Staging site to be tested (dev, prod)
Scope	Scope of test (smoke, performance)

Why Did I Write The Gatling Blueprint Project?

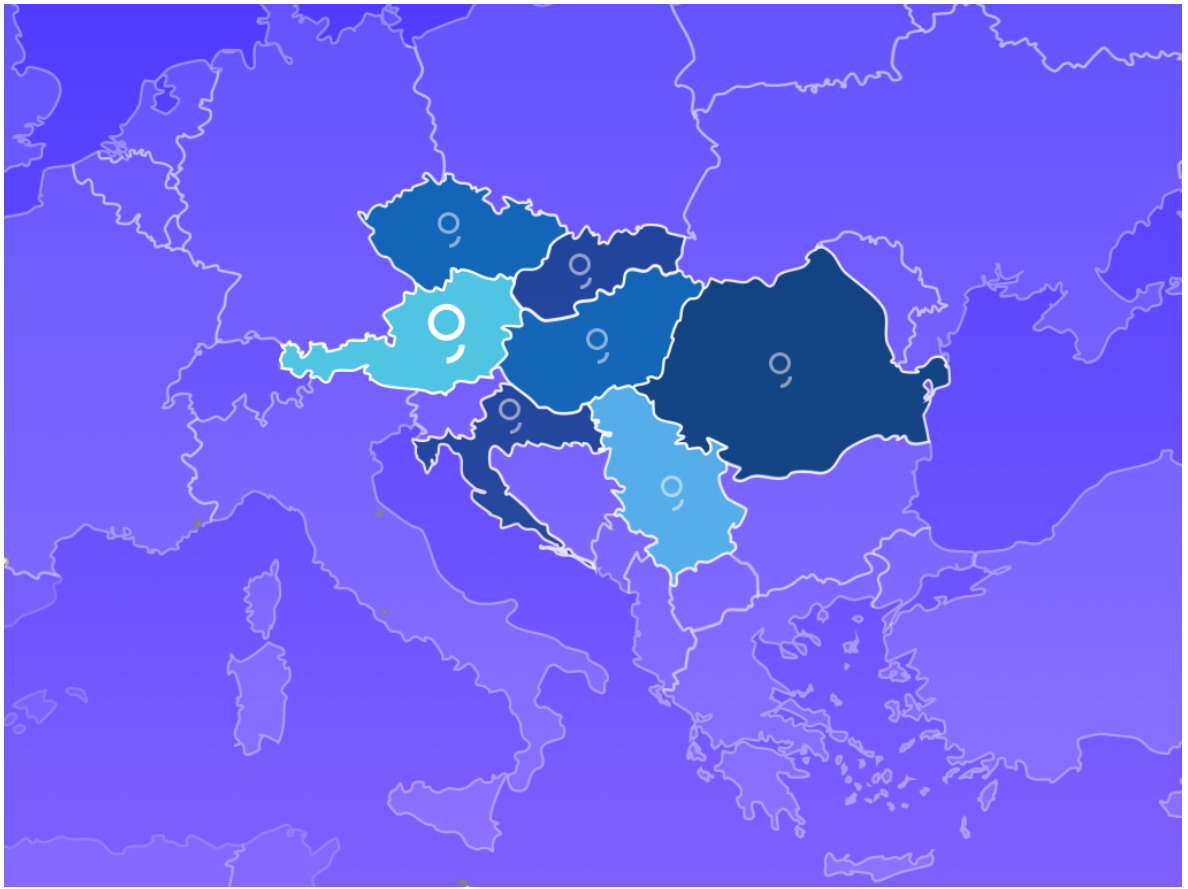
Why Did I Write The Gatling Blueprint Project?

- I had a problem - it was called George

Introducing George

- George is Erste Bank Austria’s Online Banking
- It became a group-wide solution for Online Banking

Introducing George



George International Team

- Erste Bank Austria,
- Česká spořitelna,
- Slovenská sporiteľňa
- Banca Comercială Română

- I was part of George International Team for 2 years
- George is Erste Bank Austria’s Online Banking
- It became a group-wide solution for Online Banking

George & Gatling

- Many moving parts & staging sites
- Gatling for automated integration tests
- Internal performance testing
- Continuous performance testing?

> - Other teams use JMeter & Neoload
> - Continuous performance testing is a cultural problem not a technical

When Are We Using Gatling?

- Integration tests across tenants & sites
- Developer driven performance testing
- Elastic Search server migration & tuning
- Desaster recovery tests
- Detecting changes between releases

Real Test Code

Real Test Code

```
package george.at.performance

import gatling.beeone.HttpProtocolFactory
import gatling.blueprint.ConfigurableSimulation
import george.at.ATTenantTestBuilder
import io.gatling.core.Predef._
import io.gatling.core.structure.ScenarioBuilder

class Test extends ConfigurableSimulation {

  private val userCsv = csv(resolveFile("user.csv"))
  private val httpConfServer = HttpProtocolFactory.create("george")

  val users: ScenarioBuilder = scenario(scenarionName)
    .tryMax(simulationTryMax) {
      feed(userCsv.circular.random)
        .exec(
          ATTenantTestBuilder.create("performance")
        )
    }

  setUp(users.inject(rampUsers(simulationUsers) over simulationUsersRampup))
    .maxDuration(simulationDuration)
    .protocols(httpConfServer)
    .assertions(global.failedRequests.count.is(0))
}
```

- This is real code being used for George API performance testing
- CSV file being used is resolved dynamically
- HTTP configuration hidden behind a factory
- Test steps are also create by a factory method
- Load scenario configurable using external properties

Is Gatling For You?!

Is Gatling For You?!

- Gatling's DSL is elegant & powerful
  - Programming power at your finger tips
- Scala & DSL learning curve
  - Requires solid development skills
- Works on Windows, Linux & OS X

Is Gatling For You?!

- Developer-friendly tool
- Code only, IDE support & refactoring
- Integrates nicely into your build process
- Do you need to onboard your test team?

Is Gatling For You?!

Questions?!

Gatling Resources 01

- https://gatling.io
- https://gatling.io/docs/3.0/extensions/maven_plugin
- https://github.com/sgoeschl/gatling-blueprint-project
- https://github.com/sgoeschl/presentations

Gatling Resources 02

- https://automationrhapsody.com/performance-testing-with-gatling
- https://theperformanceengineer.com/tag/gatling
- https://www.blazemeter.com/blog/how-to-set-up-a-gatling-tests-implementation-environment
- https://www.blazemeter.com/blog/how-to-set-up-and-run-your-gatling-tests-with-eclipse

Gatling Resources 03

- https://groups.google.com/forum/#!forum/gatling