

LOAD TESTING WITH GATLING



WHY LOAD TESTING?

- ▶ Process of putting demand on a system or device and measuring its response
- ▶ What's the performance when ten people are using a system, 1000? 10,000?
- ▶ Opportunity to improve your site speed or server performance
- ▶ Simply knowing your site's limits



GATLING: LOAD TESTING WITH SCALA

- ▶ Open-source load testing framework based on Scala
- ▶ Uses Scala DSL for scripting
- ▶ Allows for scripts to be written from scratch (Scala DSL)
- ▶ or generated through recording (via Selenium)



WRITE SCENARIOS USING THE SCALA DSL

```
val scn = scenario("ComputerStoreSimulation")

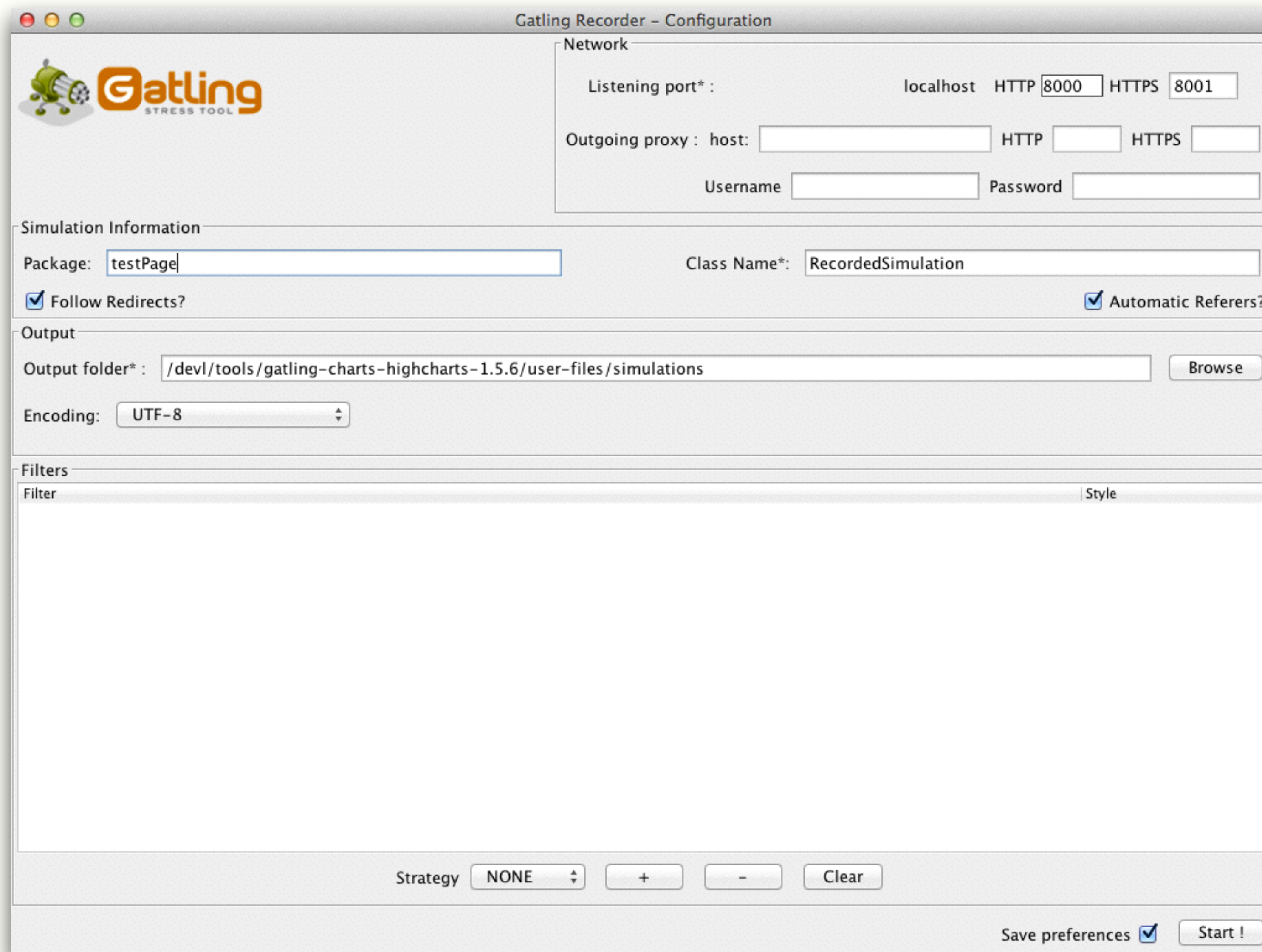
    .exec(http("index")
        .get("/"))

    .exec(http("new")
        .get("/computers/new"))

    .pause(3 seconds)

    .exec(http("create")
        .post("/computers")
        .formParam("name", "Gatling Stress Tool")
        .check(css("div.alert-message")))
```

... OR USE THE RECORDER



The screenshot shows the 'Gatling Recorder - Configuration' window. It features the Gatling logo (a green robot) and the text 'Gatling STRESS TOOL'. The configuration is organized into several sections: 'Network' with fields for listening ports (localhost HTTP 8000, HTTPS 8001) and outgoing proxy details; 'Simulation Information' with fields for Package (testPage) and Class Name (RecordedSimulation), and checkboxes for 'Follow Redirects?' and 'Automatic Referers?'; 'Output' with a text field for the output folder and a 'Browse' button; 'Filters' with a table header 'Filter | Style'; and a bottom section with a 'Strategy' dropdown (set to NONE), plus and minus buttons, and a 'Clear' button. At the very bottom, there are checkboxes for 'Save preferences' and a 'Start !' button.

Gatling
STRESS TOOL

Network

Listening port* : localhost HTTP 8000 HTTPS 8001

Outgoing proxy : host: HTTP HTTPS

Username Password

Simulation Information

Package: Class Name*:

☒ Follow Redirects? ☒ Automatic Referers?

Output

Output folder* :

Encoding:

Filters

| Filter | Style |
|--------|-------|
|--------|-------|

Strategy

Save preferences ☒

GATLING DSL BREAKDOWN

```
import scala.concurrent.duration._

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

class RecordedSimulation extends Simulation {

  val httpProtocol = http
    .baseUrl("http://gatling.io")
    .inferHtmlResources()

  val headers_0 = Map("Accept" -> "text/html,application/xhtml+xml,
    application/xml;q=0.9,image/webp,*/*;q=0.8")

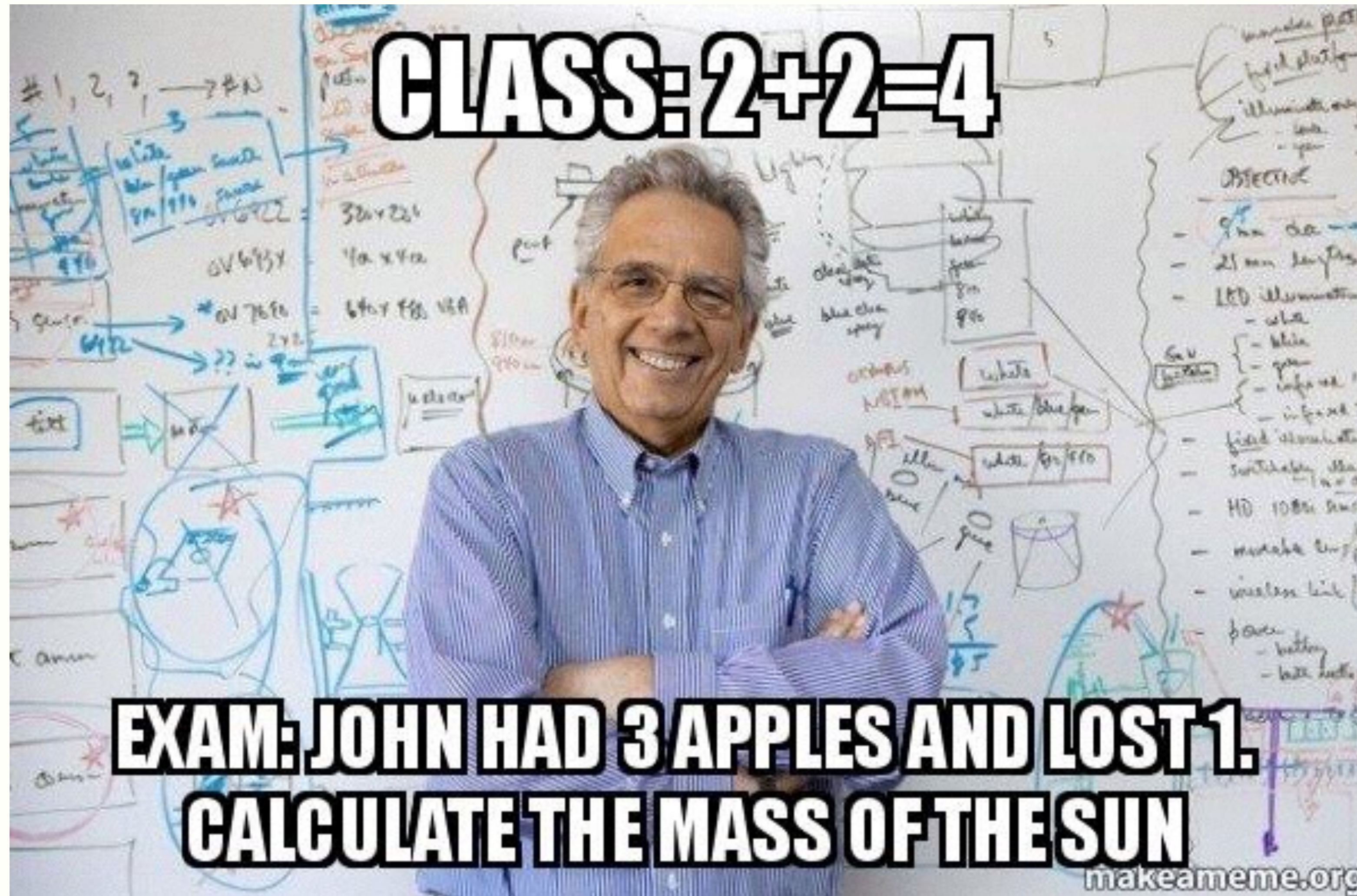
  val scn = scenario("GatlingIOSimulation")
    .exec(http("loadHomePage")
      .get("/")
      .headers(headers_0)
      .check(status.is(200))
      .check(bodyString.saveAs("homePageHtml")))
    .exec(session => {
      print(session("homePageHtml"))
      session
    })

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

- ▶ required Gatling class/library (Need at the top of each Simulation)
- ▶ All 'Simulations' classes extend *Simulation*
- ▶ Sets up http protocol for simulation.
- ▶ Includes Base URL you are testing against
- ▶ We tell the protocol to infer all the resources need to do the requests
- ▶ Sets up Scenario
- ▶ This example does the following:
 - ▶ perform a 'get' request against the root
 - ▶ performs the request with a given header
 - ▶ checks that the response of status '200'
 - ▶ Saves the body of the response to a variable
- ▶ This setup function executes the scenario with a given # of users

DEMO

YOUR TURN!



EXERCISES

All Exercises will use: <http://computer-database.herokuapp.com/>

1. Using the recorder, record a scenario where a user searches for a computer and selects a few. Run the scenario from the command line.
2. Without using the recorder, create a scenario that emulates a single user searching through the full list of users (next button)

EXERCISES CONTINUED

Use maven archetype for the rest of the examples:

http://gatling.io/docs/2.0.3/extensions/maven_archetype.html

<http://scala-ide.org/download/current.html>

1. Adding to the previous example, emulate adding a new computer to the list (make sure to add pauses to simulate a user accessing the site!).
 - Scenario: 10 users ramped over 60 seconds
2. Use a CSV feeder to provide a list of computers to be created; emulate several users adding computers to the list
3. Create a new class that runs the two scenarios.
 - The Search scenario: 30 users ramped over a 2 minute period, repeat 2 times
 - The Creation scenario: 3 users over 30 seconds

IF YOU'RE FEELING AMBITIOUS:

Take some time and check out the Clojure Koans:

<http://clojurekoans.com>

