Lasttests mit Selenium

Übersicht

1. Kurze Vorstellung
2. Motivation
3. Verschiedene Lasten
4. Ausgangslage
5. Klassische Tools
6. Warum Selenium?
7. Demo
8. Hinweise und Fazit

Vorstellung Referent

Christian Kumpe
Expert Developer

- Informatikstudium am KIT (Universität Karlsruhe)
- Freelancer im Bereich Web und Java
- Seit Mai 2011 bei der diva-e Netpioneer GmbH in Karlsruhe
- Seit 2002 in der Java-Welt unterwegs

diva-e Netpioneer Steckbrief

- Seit 1996 berät die diva-e Netpioneer GmbH Unternehmen in den Bereichen IT und Internetstrategie

- Langjährige Erfahrung im agilen Projektmanagement und arbeiten in allen Projekten mit Scrum
- *Anfang 2016*: Zusammenschluss mit fünf anderen Unternehmen zu diva-e,
  um Kunden ein ganzheitliches Portfolio entlang der digitalen Wertschöpfungskette bieten zu können

Motivation

Zu viele Besucher brechen den Checkout ab.

Das Live-System verhält sich heute aber zäh!

„…“

Funktioniert es nur langsam, oder funktioniert es gar nicht?

Wie viele gleichzeitige Besucher verkraftet der Shop?

Aber es wurden doch keine Geschwindigkeitsvorgaben gemacht!

Arten von System-Lasten

Grundlast:
Wird meist nicht durch Tests abgesichert

Erwartete **Lastspitzen**:
Hierfür sollte das System ausgelegt sein.
Kann durch **Lasttests** abgesichert werden.

Was passiert mit dem System bei **Überlast**? Kann durch Lasttests überprüft werden.

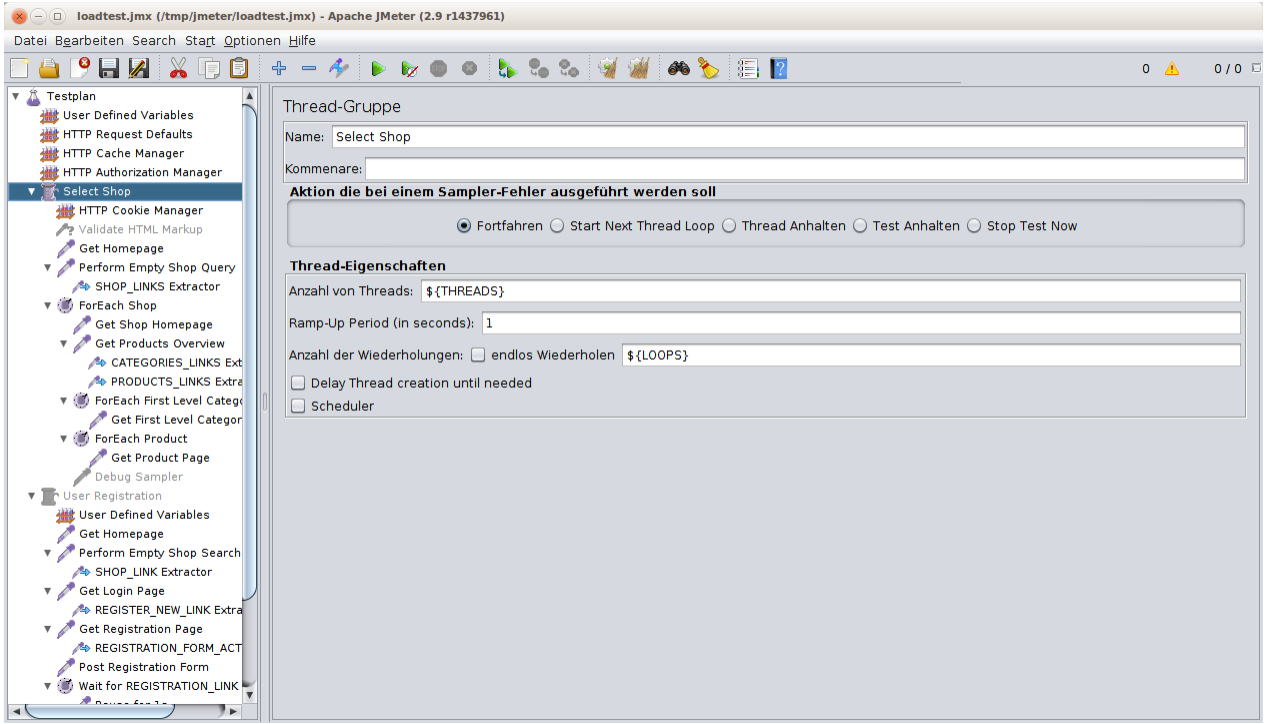
Kurzer Blick in den Webshop...

Ausgangslage im Projekt

- Webshop mit „klassischen“ HTML-Bordmitteln
- Wenig JavaScript und viele Seiten-Reloads
- Erste Lasttests waren mit JMeter umgesetzt
- Webtests wurden mit Selenium umgesetzt

Standard-Tools für Lasttests

JMeter Beispiel



Selenium für Webtests

Webdriver API

steuert

Firefox

Chrome

Internet Explorer

...

HtmlUnit

PhantomJS

Kleines Selenium Beispiel


```
WebDriver driver = new FirefoxDriver();
// goto shop
driver.get("http://localhost:8080");

// add a red flower to the basket
driver.findElement(By.name("red")).click();
assertThat(driver.findElement(By.id("basket")).getText(),
       is("[red flower]"));

// checkout current basket
driver.findElement(By.name("checkout")).click();
assertThat(driver.findElement(By.id("message")).getText(),
       is("[red flower] will be delivered tomorrow."));
```

Umstieg auf AngularJS und REST

- Deutliche Reduzierung der Seiten-Reloads für bessere UX
- Zunehmend mehr AJAX-lastige Prozesse im gesamten Shop
- Anpassungen an den Webtests abgeschlossen
- Die Anpassungen der JMeter-Tests wurden zunehmend aufwendiger ☹

Vielleicht mit Selenium?

- Know-how im Team vorhanden
- Probleme mit AngularJS wurden bereits für die Webtests gelöst
- Bei Änderungen am Shop müssen die Lasttests nicht extra gepflegt werden
- Erzeugte Last nahe am echten Userverhalten

Abgrenzung

- In der aktuellen Ausbaustufe waren für den Shop maximal zweistellige gleichzeitige Userzahlen zu erwarten
- Die mit Selenium-Clients erzeugbare Last wird sich mit vertretbarem Aufwand nur bedingt steigern lassen
- In größeren Ausbaustufen werden alternative Tools sicher wieder interessant

Die Grundidee hinter dem Ganzen

JUnit

Vorhandene Webtests mehrfach wiederholt und parallel ausführen

Ein mehrfach wiederholter paralleler Test...

Paralleles JUnit

- Eigene JUnit-Runner-Implementierung auf Basis des SpringJUnit4ClassRunner
- Konfiguration der Tests mit Spring
- Skalierung der Webtests durch ein Selenium Grid
- Einfache Statistik-Komponente zum Erfassen von Zeitstempeln auch im JavaScript

Beispiel zum Test-Setup


```
loadtest.parallel.threads = 10
loadtest.parallel.repetitions = 3

loadtest.window.width = 1024
loadtest.window.height = 768

loadtest.browser = firefox
loadtest.url = http://localhost:8080

# thread specific configuration
loadtest.browser.thread.0 = firefox
loadtest.window.width.thread.0 = 600
loadtest.window.height.thread.0 = 450
```

Und jetzt das Ganze in Aktion!

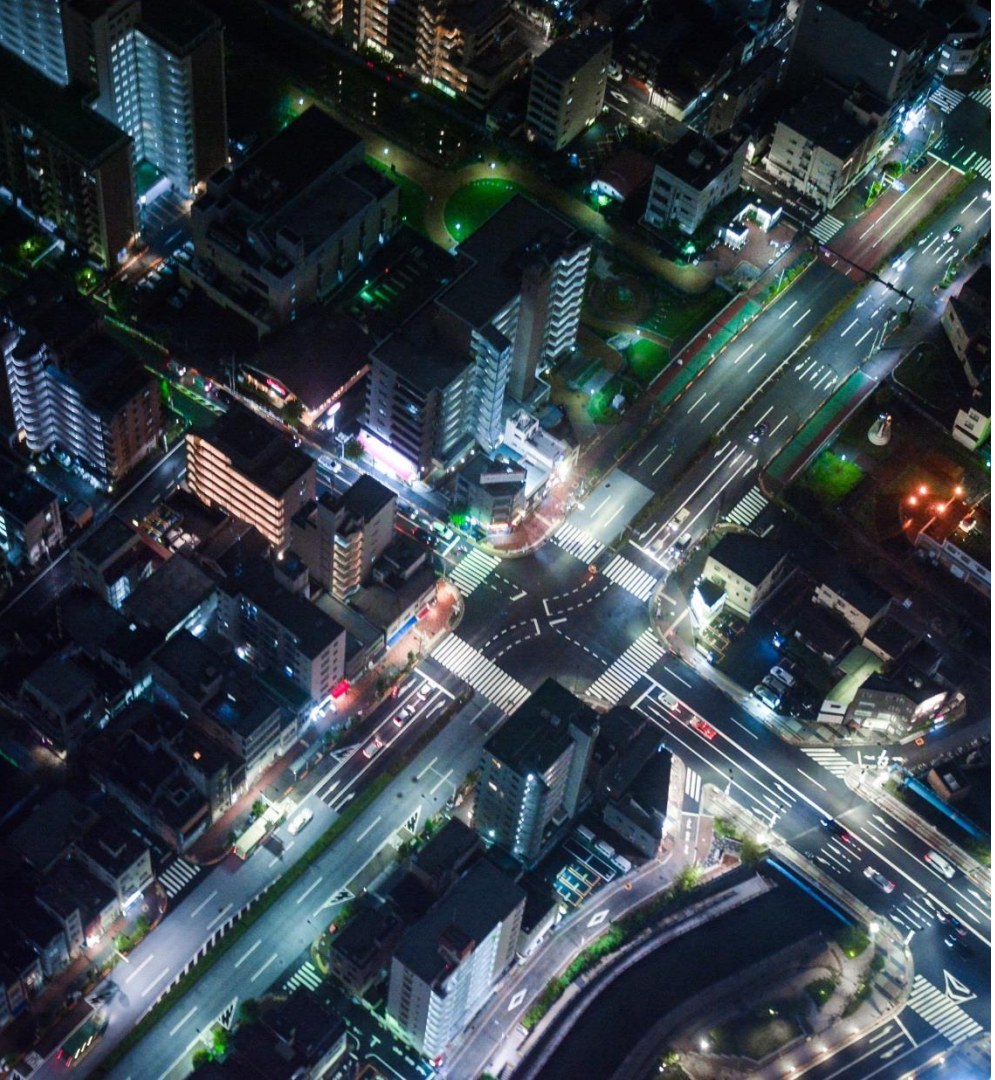
Was noch wichtig ist...

- Vorgaben und Ziele für die Lasttests vom Kunden einfordern!
- Verlauf der Lasttest und deren Ergebnisse sauber dokumentieren!
- Lasttests regelmäßig ausführen!
- Gut überlegen was man eigentlich testet!

Fazit

- Bis zur aktuellen Ausbaustufe des Shop können wir nach wie vor die benötigte Last mit unserem Setup erzeugen
- Die Wartung der Lasttests erzeugt kaum zusätzlichen Aufwand
- Über eine zusätzliche Statistik-Komponenten können wir über die Lasttests auch Teile er UX messen

- In unserem Projekt haben wir die Entscheidung nicht bereut!



Vielen Dank für Ihre Aufmerksamkeit.

Bis zum nächsten Mal

Adresse
diva-e Digital Value Enterprise GmbH
Office Karlsruhe

Ihr Kontakt
Christian Kumpe
Expert Developer

T +49 721 92060 710
christian.kumpe@diva-e.com
www.diva-e.com