Orientation in Objects

Tutorial: Flexible Lasttest mit Grinder

) Schulung)

) Beratung)

) Entwicklung)

) Artikel)

AUTOR

Andreas Spall
Orientation in Objects GmbH

Veröffentlicht am: 21.12.2009

TUTORIAL: FLEXIBLE LASTTEST MIT GRINDER

Das Tutorial zeigt, wie man mit The Grinder 3 Testskripte aufzeichnen und abspielen kann sowie die aufgezeichneten Skripte über das Einspielen von Parametern dynamisch gestalten kann.

Orientation in Objects GmbH

Weinheimer Str. 68
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.de info@oio.de

EINLEITUNG

In diesem Tutorial wollen wir ein Performance-Testskript mittels Grinder aufzeichnen, es dynamisieren und gegen eine Webanwendung abspielen. Für dieses Tutorial werden Sie Eclipse (Version 3.4 oder hoher) und einen Tomcat der Version 6.0.x benötigen.

WAS IST GRINDER

Neben einigen kommerziellen Performanz-Testtools wie beispielsweise der IBM Rational Performance Tester oder QALoad von Compuware existieren auch zwei namhafte Open Source basierte Produkte. Zum einen das Lasttesttool JMeter, welches bei Apache angesiedelt ist und das Lasttesttool The Grinder, welches aus dem Buch "Professional Java 2 Enterprise Edition with BEA WebLogic Server" von Paco Gómez und Peter Zadrozny stammt und von Philip Aston gepflegt wird.

Der aktuelle Systemkern von Grinder, welcher nach vielen Jahren der Entwicklung mit der Version 3 von Grinder (offizieller Name The Grinder 3) live gegangen ist, hat relativ wenig mit seinen vorangegangen Versionen gemeinsam. Vielmehr wurde durch die Verwendung von Jython als Scriptsprache direkter Zugriff auf jede Java basierte Technologie wie RMI, EJB, JMS uvm. zum Erstellen der Testskripte ermöglicht. Dadurch ist es nicht nur möglich die Web-Repräsentation einer Java EE - Anwendung in Form von HTTP-Anfragen einem Performance-Test zu unterziehen, sondern alle mittels Java adressierbaren Bestandteile der Anwendung. In diesem Zusammenhang wird von "white box stress testing" gesprochen, da auch innere Bestandteile einer Anwendung wie beispielsweise die Datenbank einem Test unterzogen werden kann.

TCP PROXY

Der TCP Proxy dient dem Aufzeichnen von Performance-Testskripten. Er wird als Java-Programm gestartet und kann alle an ihn gesendeten Anfragen im Arbeitsspeicher cachen. Hierzu muss man dessen IP und Port im Browser als Proxy eintragen. Jeder Aufruf an die Web-Anwendung wird dann von dem TCP Proxy protokolliert. Nach dem Beenden des TCP Proxy(s) werden die einzelnen Anfragen als Jython Testskript in eine Datei gespeichert oder auf der DOS-Konsole ausgegeben.

CONSOLE

Die Console stellt eine grafische Benutzerschnittstelle zur Verfügung und wird verwendet, um die Testinstanzen zu koordinieren und den Test zu überwachen. Zudem stellt sie Funktionalitäten zur Verfügung um aufgezeichnete bzw. manuell erstellte Testskripte zu editieren und an die Testinstanzen zu übertragen.

TESTINSTANZ

Eine Testinstanz bestehet aus einem Agent Prozess und einem oder mehreren Worker Prozessen.

Der Agent dient der Administration der einzelnen Worker und erhält seine Befehle und die zu startenden Testskripte von der Console. Hierzu verbindet sich der Agent nach dem Start mit der Console und wartet auf deren Befehle. Im Anschluss an den Startbefehl von der Console startet der Agent, entsprechend seiner Konfiguration, die Worker und übermittelt diesen die Anzahl der zu startenden Worker-Threads, das Testskript und einige andere Parameter (siehe:).

Die Worker, bzw. deren Worker-Threads, beginnen entsprechend ihrer Parameter mit dem Abfeuern der im Testskript definierten Anfragen an das "System under Test". Die Ergebnisse der Anfragen, wie z. B. Antwortzeit, Responsecode oder Responsegröße (in Bytes) werden in einer Log Datei gespeichert und an die Console übermittelt.

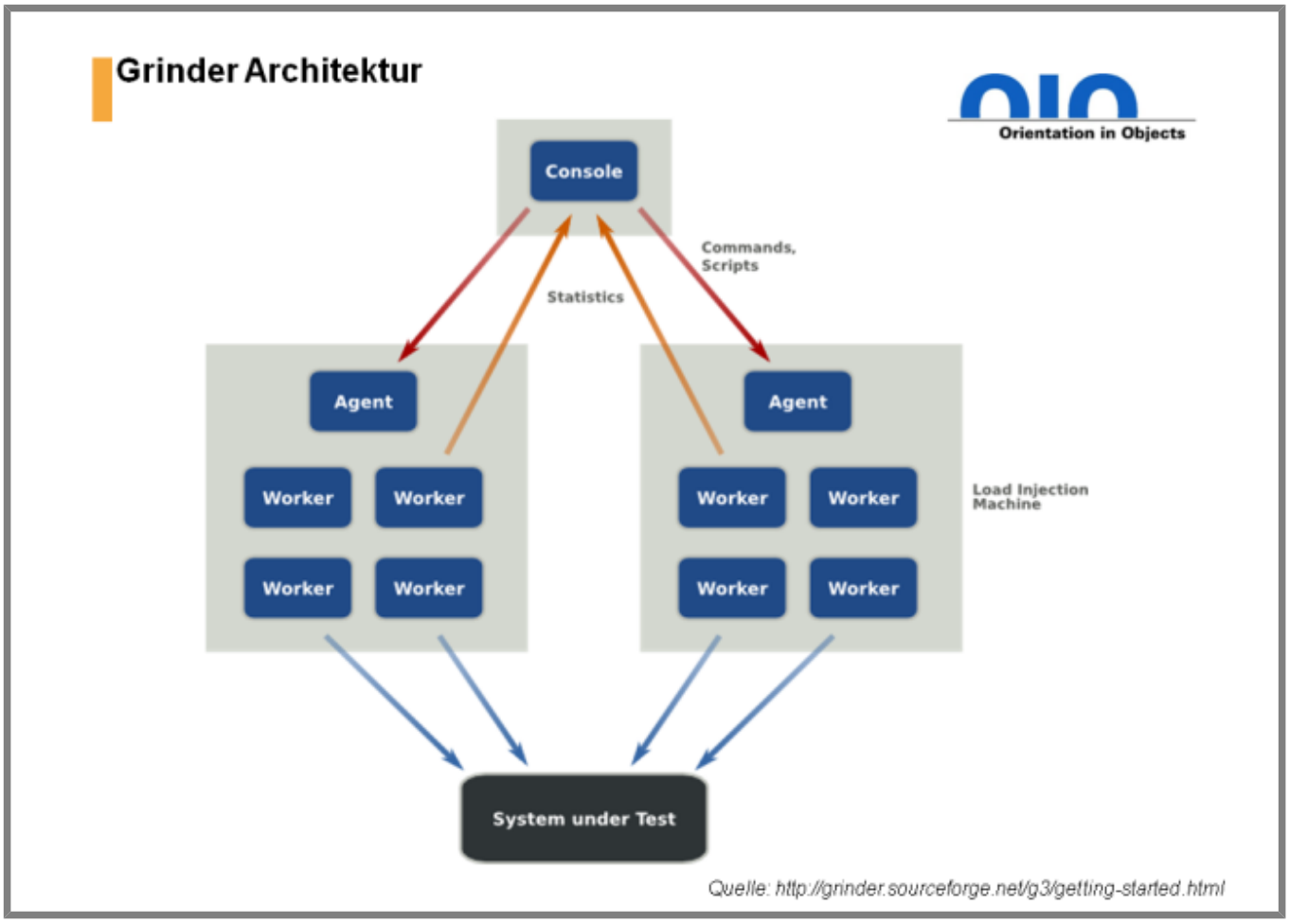

Abbildung 1: Grinder Architektur

DAS TUTORIAL TESTPROJEKT

Dieses Tutorial basiert auf einem bestehenden Projekt, welches am Ende dieser Ausführungen als Zip-Datei heruntergeladen werden kann.

Das in der Zip-Datei enthaltene Projekt kann über den Import-Wizard von Eclipse in den eigenen Workspace eingebunden werden. Im Anschluss an den Import sollte folgende Projektstruktur zur Verfügung stehen.

Das Projekt beinhaltet eine zum Zeitpunkt der Erstellung dieses Tutorials aktuelle Version von Grinder und ermöglicht es, die Funktionalitäten von Grinder direkt zu nutzen.

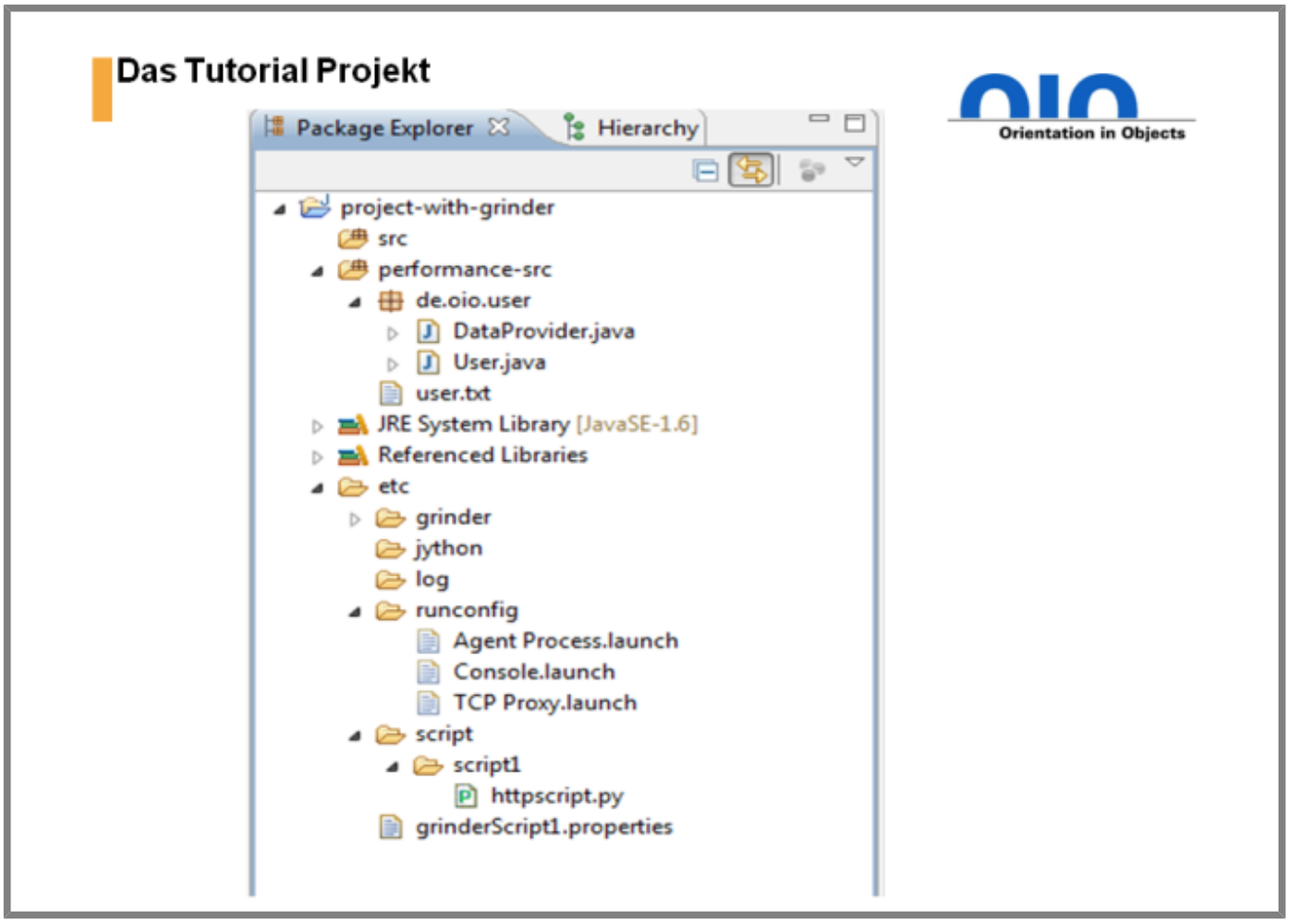

Abbildung 2: Das Tutorial Projekt

Betrachten wir, bevor wir uns mit der Aufzeichnung, Dynamisierung und dem Abspielen von Performance-Testskripten beschäftigen, die interessanten Bestandteile des Projektes.

Innerhalb des Quelltextordners performance-src befinden sich zwei Java-Quelltextdateien, welche später für die Dynamisierung des aufgezeichneten Testskripts benötigt werden. Hierzu beinhaltet user.txt die zu verwendeten Vor- und Nachnamen.

Der Ordner etc/grinder beinhaltet eine Grinder Distribution, deren Jar-Bibliotheken in den Classpath des Projektes aufgenommen wurden. Die obig beschriebenen Kernkomponenten Console, Agent Process und TCP Proxy wurden in startfähige Konfigurationen im Ordner etc/runconfig gespeichert. Die Konfiguration "Agent Process.launch" übergibt dem gestarteten Agent Process die Propertiesdatei "grinderScript1.properties, in welcher das abzuspielende Performance-Testskript "etc/script/script1/httpscript.py" festgelegt wurde.

AUFZEICHNEN

Beginnen wir mit dem Aufzeichnen eines Performance-Testskripts für die zu testende Webanwendung. Hierzu müssen Sie als Erstes den **TCP Proxy** starten. Die Launch Konfigurationsdatei (**etc/runconfig/TCP Proxy.launchn**), kann über das "Run As" Submenu des **Kontextmenüs gestartet** werden (vgl. Abbildung TCP Proxy starten Punkt 1).

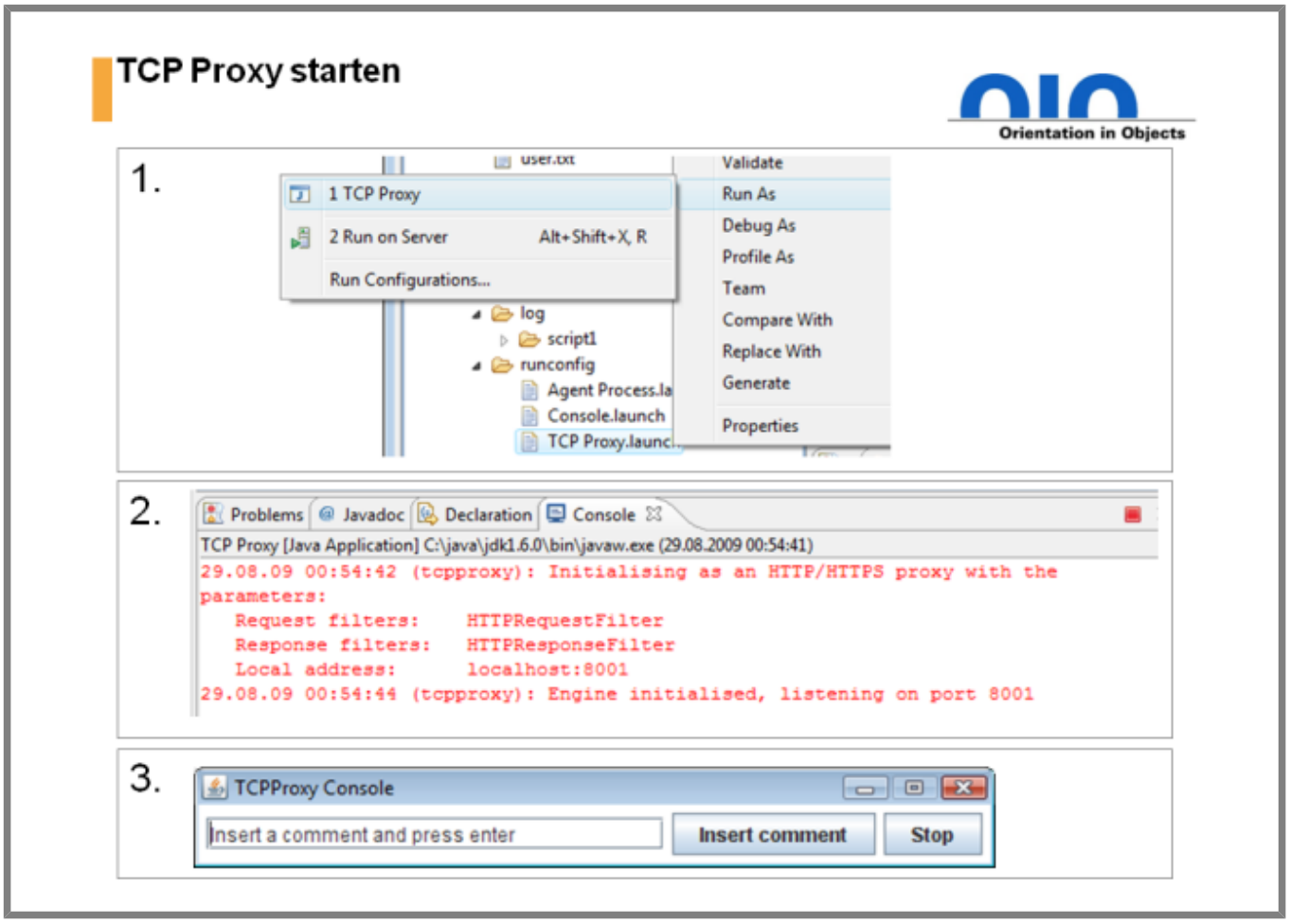

Abbildung 3: TCP Proxy starten

Der gestartete TCP Proxy gibt über die Java-Console der Entwicklungsumgebung aus, über welchen Port der Proxy angesprochen und auch im Browser Ihrer Wahl konfiguriert werden soll.

Entsprechend den Ausgaben von Punkt 2 in der Abbildung "TCP Proxy starten" müssen Sie im **Browser** folgenden **Proxy konfigurieren**:

- Adresse: localhost
- Port: 8001

Beim Starten gibt der TCP Proxy nicht nur auf der Java-Console einige Ausgaben aus, vielmehr wird auch eine grafische Benutzerschnittstelle gestartet. Hierüber ist es möglich, vor interessanten Benutzerinteraktionen über den Browser Kommentare über diese Aktion abzugeben und die Skripterstellung zu beenden.

Im Anschluss an die erfolgreiche Konfiguration des Browsers können wir das Testskript erzeugen lassen. **Starten** Sie zunächst den **Tomcat**. Nachdem der Tomcat erfolgreich gestartet wurde, müssen Sie im **Browser** die Url http://localhost:8080 **eingeben** und die folgenden Interaktionen durchführen (siehe hierzu auch Abbildung "Das Request Parameter Beispiel"):

- **Drücken** Sie im Miscellaneous Bereich auf **Servlet Examples**.
- **Drücken** Sie die bei Request Parameters auf **Execute**.
- **Geben** Sie im Feld "First Name" Andreas und im Feld "Last Name" Spall **ein** und **warten sie einige Sekunden**, dann **Enter drücken**.

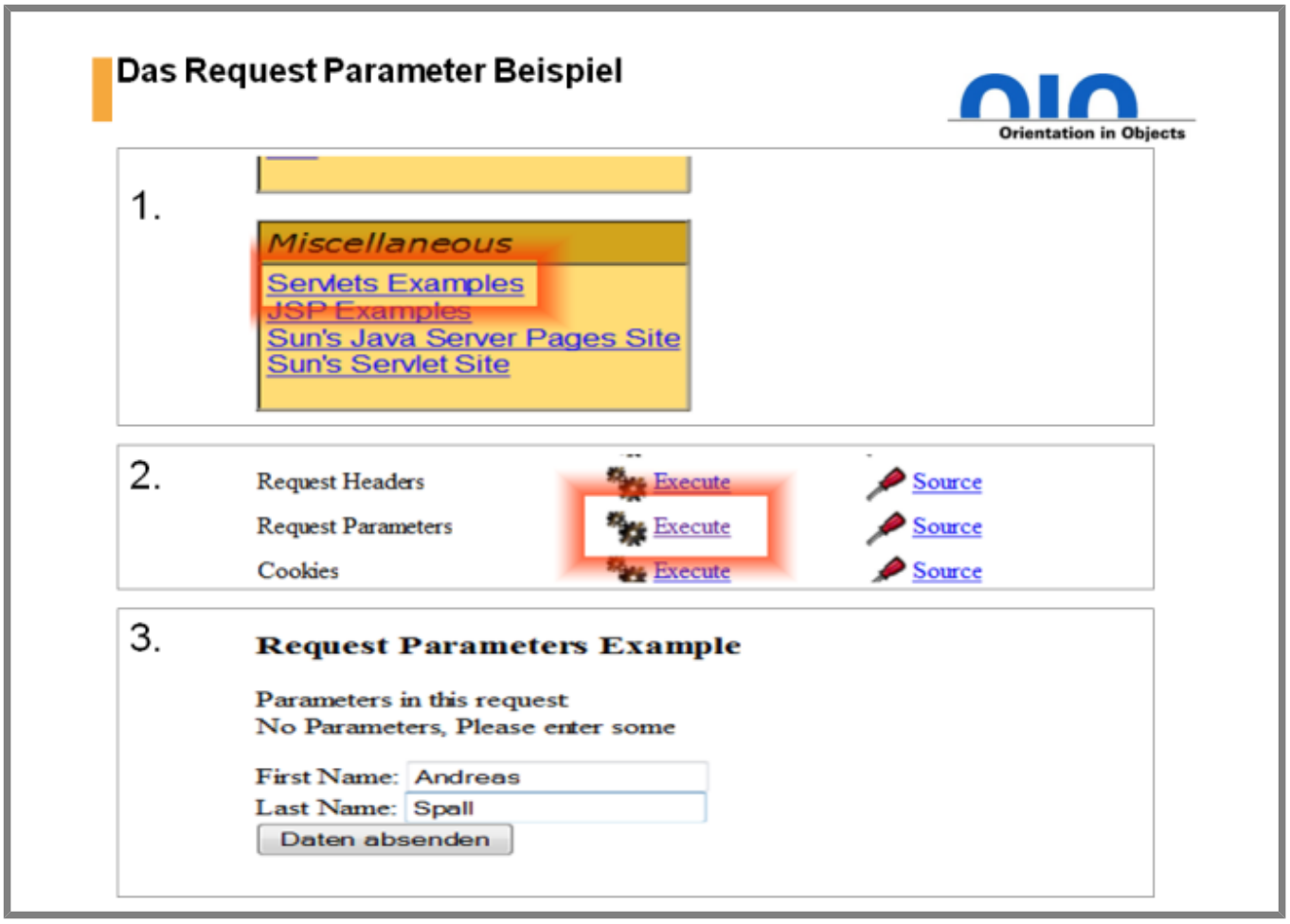

Abbildung 4: Das Request Parameter Beispiel

Durch das **drücken** von **Stop** auf der grafische Benutzerschnittstelle des TCP Proxies wird das aufgezeichnete Skript über die Java-Console ausgegeben.

Das auf der Java-Console ausgegebene Skript muss vor einer Bearbeitung permanent im Dateisystem gespeichert werden. Hierzu müssen Sie den zu speichernden Inhalt (die in der **Java-Console als schwarz ausgegeben Zeilen**) in die **Zwischenablage kopieren**.

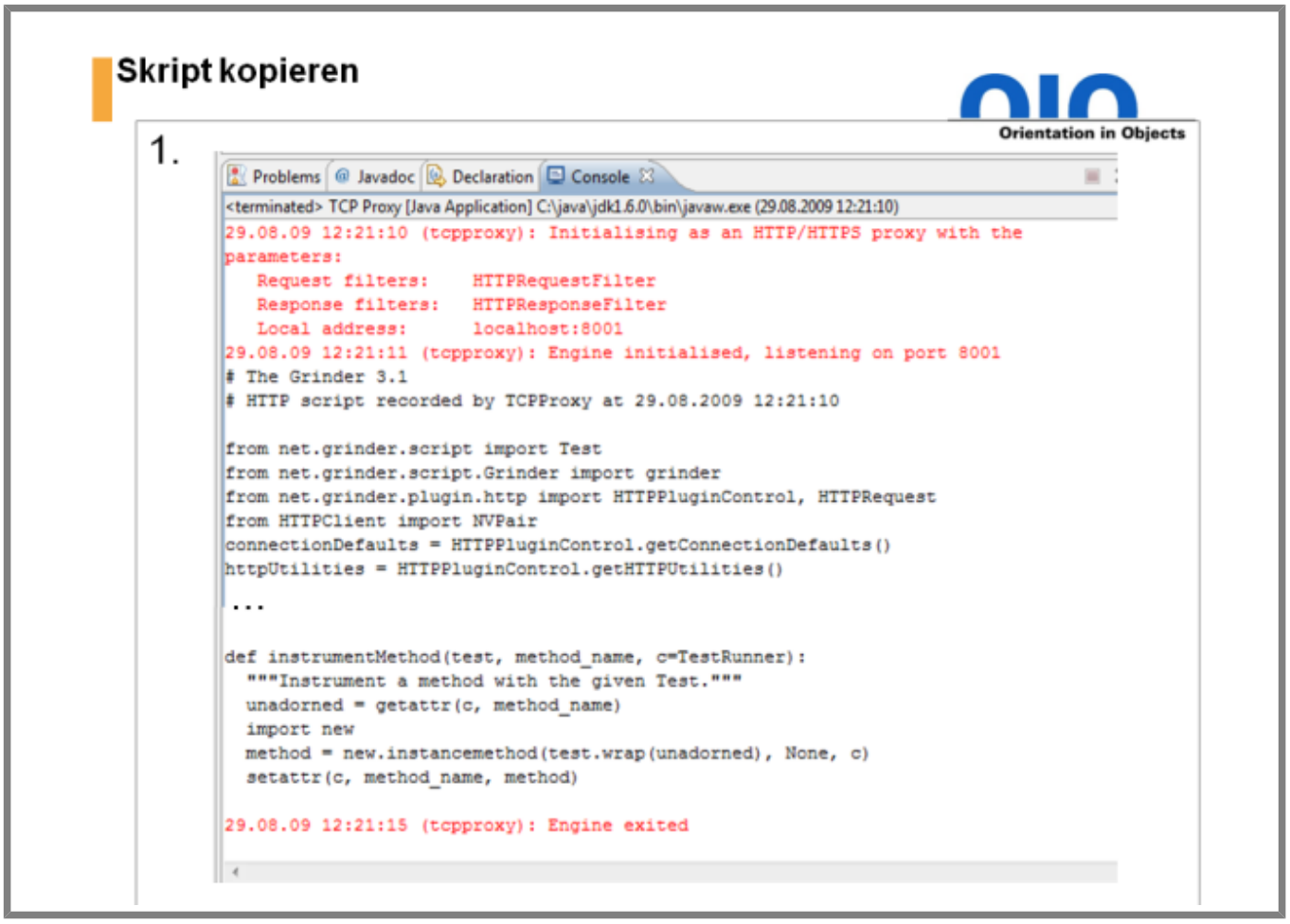

Abbildung 5: Skript kopieren

Die Zieldatei hat den Namen **httpscript.py** und befindet sich in dem Ordner **etc/script/script1**. Der Inhalt dieser Datei muss durch den **Inhalt der Zwischenablage ersetzt** und das hierdurch entstandene Performance-Testskript **gespeichert** werden.

DYNAMISIEREN DER TESTSKRIPTE

Bevor wir mit einer Dynamisierung beginnen, berachten wir, mit welchen Daten wir diese durchführen möchten.

Der Quelltextordner performance-src biete hierfür zwei interessante Klassen. Die Klasse DataProvider, welche als Singleton implementiert wurde, beinhaltet eine Liste von User Objekten. Er gibt über die Methode next() ein User-Objekt zurück. Dieses User Objekt beinhaltet zwei String Werte, welche von dem DataProvider aus der Datei performance-src/user.txt gelesen wurden. Bei diesen zwei Werten handelt es sich um die Dynamisierung für unseren Vor- und Nachnamen.

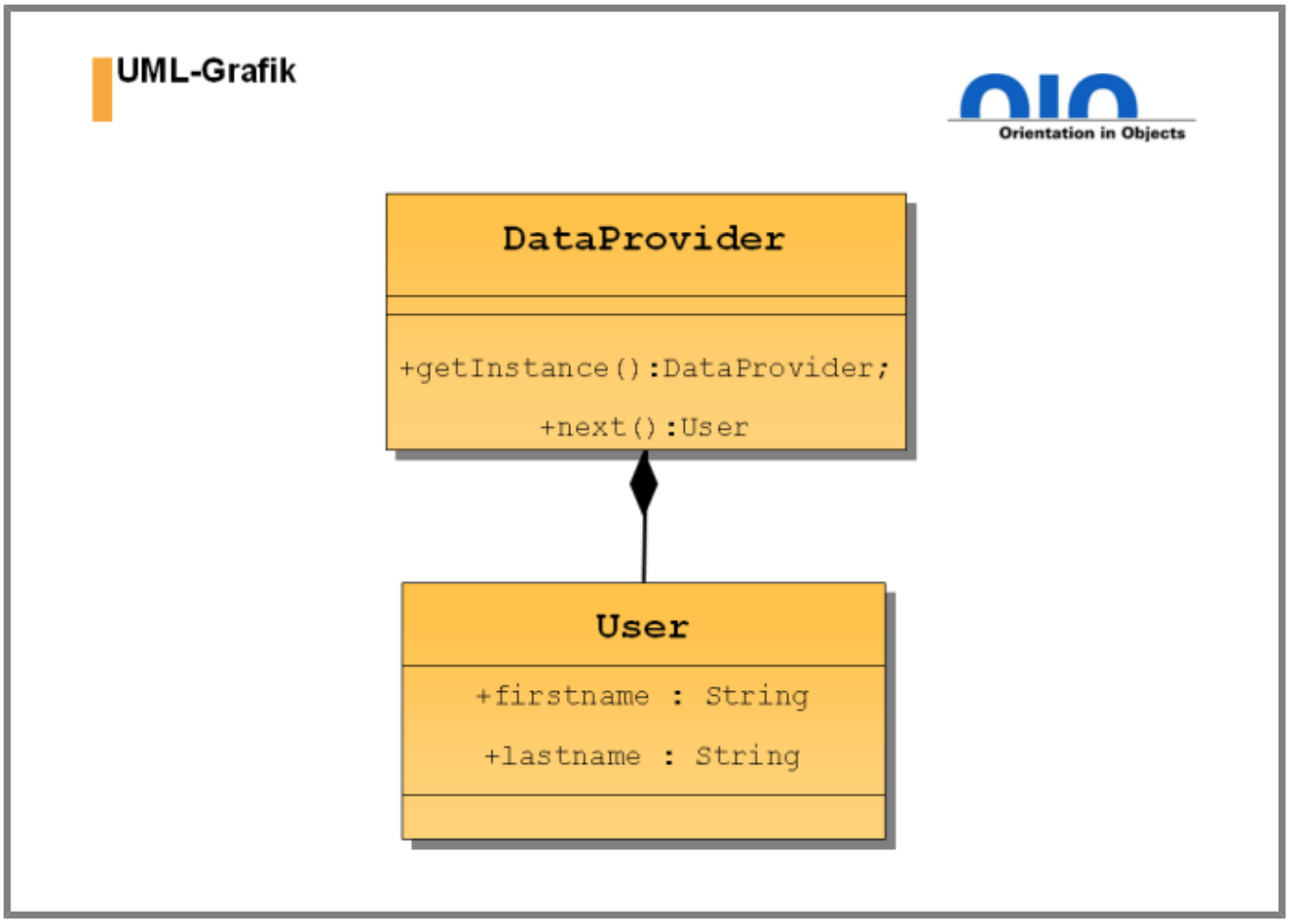

Abbildung 6: UML-Grafik

Beginnen wir mit der Dynamisierung:

Öffnen Sie, falls noch nicht geschehen, das aufgezeichnete Jython Skript **etc/script/script1/httpscript.py**

Fügen Sie auf der Höhe der anderen Import Zeilen die **Folgende ein**:

```
from de.oio.user import DataProvider
```

Nachdem wir unserem Jython Skript die Klasse DataProvider bekannt gemacht haben, können wir diese verwenden, um eine Instanz des Users zu erhalten.

Hierzu müssen Sie die **folgende Zeile suchen**:

```
def __call__(self):
```

Fügen Sie die **folgende Zeile** als nachfolgende Zeile **ein.**

```
user = DataProvider.getInstance().next()
```

Durch das Hinzufügen der obigen Zeile halten wir mit der Referenz auf ein User-Objekt, welches die dynamischen Werte beinhaltet.

Suchen Sie **nach 'Andreas'**. Die folgende Zeilen müssen gefunden werden

```
def page3(self):
   """POST RequestParamExample (request 301)."""
      result = request301.POST('/examples/servlets/servlet/RequestParamExample',
   ( NVPair('firstname', 'Andreas'),
      NVPair('lastname', 'Spall'), ),
```

Ersetzen Sie 'Andreas' durch den folgenden Ausdruck

```
user.getFirstname()
```

Ersetzen Sie 'Spall' durch den folgenden Ausdruck

```
user.getLastname()
```

Hierdurch werden nicht mehr die Begriffe Andreas und Spall beim Aufruf übergeben, sondern vielmehr die String-Variable, welche sich hinter dem Aufruf getFirstname() bzw. getLastname() befinden. Damit dieser Aufruf an der Referenz "user" erfolgreich durchgeführt werden kann, muss diese, wie die "self" Referenz, in dem Quelltextabschnitt definiert werden.

Hierzu muss die Zeile **gesucht**:

```
def page3(self):
```

und durch diese **ersetzt werden**:

```
def page3(self, user):
```

Damit der Parameter user der Methode page1 auch übergeben wird, muss man den Aufruf von **page3() durch page3(user) ersetzten**.

Die httpscript.py-Datei sollte am Ende dem folgenden Listing ähneln.

```
def page3(self, user):
   """POST RequestParamExample (request 301)."""
      result = request301.POST('/examples/servlets/servlet/RequestParamExample',
   ( NVPair('firstname', user.getFirstname()),
      NVPair('lastname', user.getLastname()), ),
...
def __call__(self):
   user = DataProvider.getInstance().next()
   self.page1()
   grinder.sleep(436)
   self.page2()
   grinder.sleep(7570)
   self.page3(user)
```

ABSPIELEN DES LASTTESTS

Bevor wir einen Performance-Test gegen die Web Anwendung starten können, müssen Sie die **Console starten**. Die hierzu gehörige Launch Konfigurationsdatei (**etc**/**runconfig**/**Console**.**launch**) , kann über das "Run As" Submenu des **Kontextmenüs gestartet** werden.

Als Nächstes müssen Sie den **Agent Prozess** starten. Die hierzu gehörige Launch Konfigurationsdatei (**etc**/**runconfig**/**Agent Process**.**launch**) , kann über das "Run As" Submenu des **Kontextmenüs gestartet** werden.

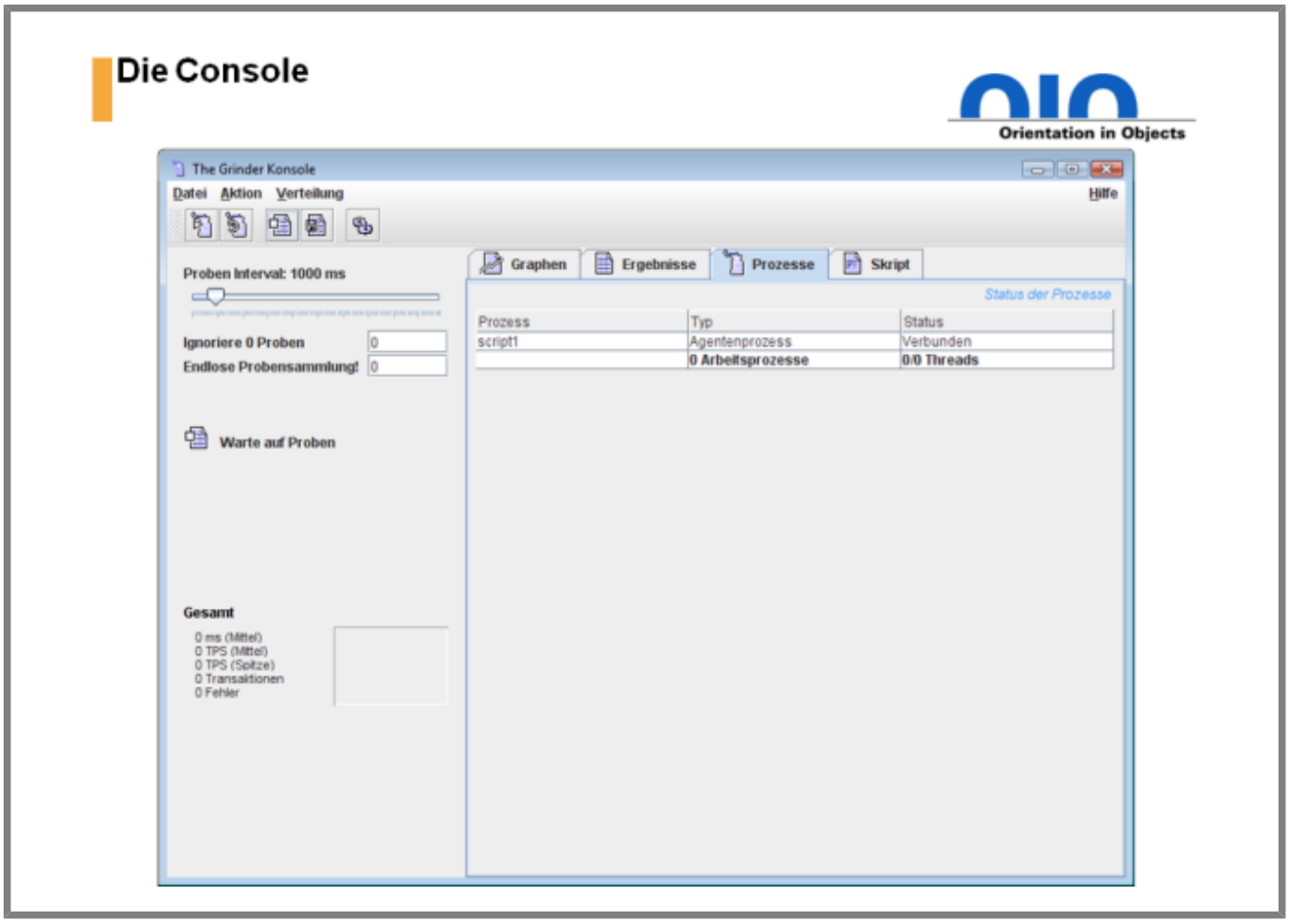

Abbildung 7: Die Console

Wie man der obigen Darstellung entnehmen kann, hat sich der Agent Prozess mit der Console verbunden und wartet auf den Startbefehl.

Zum Starten des Performance-Tests müssen Sie im Menü, über den **Menüpunkt Aktion, Start Prozesse auswählen**.

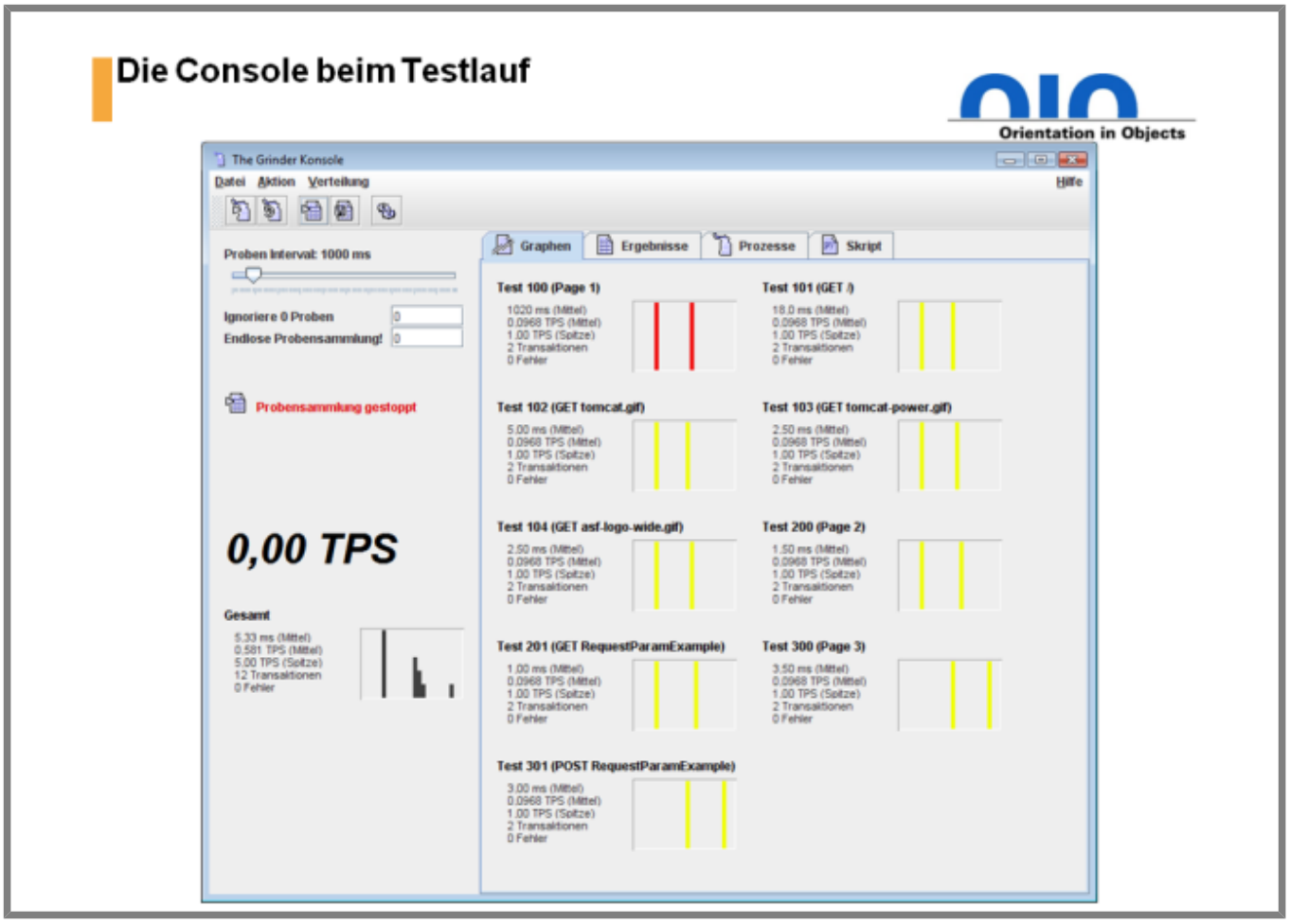

Abbildung 8: Die Console beim Testlauf

Wie man der obigen Darstellung entnehmen kann, wurde der Performanz-Test entsprechend der Konfiguration (etc/grinderScript1.properties) zweimal zur Ausführung gebracht.

FAZIT

Wie dieses Tutorial dargestellt hat, ist es möglich mit Grinder dynamische Testskripte zur Ausführung zu bringen.

Über Anpassungen in der Property-Datei, grinderScript1.properties, ist es möglich die Testdurchführung zu beeinflussen. So wäre es denkbar, den Test von einem Thread und Prozess auf X Threads a Y Prozesse zu skalieren.

Sollten Ihnen die Ausgaben der Console bei Ihrer Analyse nicht ausreichen, so besteht die Möglichkeit, basierend auf den Client-Log Dateien, nachträgliche Analysen durchzuführen.

ANHANG: DAS GRINDER PROJEKT

Download project-with-grinder.zip

REFERENZEN

- The Grinder, a Java Load Testing Framework
  grinder.sourceforge.net (http://grinder.sourceforge.net/)
- Grinder Analyzer
  Grinder Analyzer (http://sourceforge.net/project/showfiles.php?group_id=160220)