



Blick hinter die **Kulissen**

Java-Webtechnologien gibt es in Hülle und Fülle und nicht wenige erheben den Anspruch, die beste Lösung für die Entwicklung einer Webanwendung zu bieten. Das Java Magazin hatte letztes Jahr die Community aufgefordert, ihre Lieblingstechnologie bei der W-JAX Challenge vorzustellen – in Form eines Tippspiels für die WM 2010. Jetzt wollen wir die Ergebnisse vorstellen.

von Claudia Fröhling

Nach unserem Aufruf im vergangenen Herbst hofften wir auf eine rege Teilnahme bei der Challenge, wurden am Ende sogar von der Fülle an Einreichungen positiv überrascht. Acht Projekte wurden zum Stichtag eingereicht und boten ein buntes Potpourri an Webtechnologien, die zum Einsatz gekommen waren: Von Groovy über Scala/Lift bis hin zu Apache Wicket und Google App Engine war einiges vertreten.

Einzige Vorgabe war, sich an die von der Challenge-Jury erarbeitete Spezifikation zu halten. Diese gab Minimalanforderungen für Frontend und Backend vor, unter anderem wurden eine Datenbank und ein Application Server vorgegeben und die Anwendung sollte auf JRE 1.6.0_16 lauffähig sein.

Die Jury, bestehend aus Arno Haase, Markus Stäuble, Thilo Frotscher und Stefan Zörner, evaluierte die eingereichten Pro-

jekte und definierte vier Finalisten, die dem W-JAX-Publikum zur finalen Siegerkürung vorgestellt wurden. Alle Challenge-Teilnehmer durften im November nach München zur W-JAX reisen und die Entscheidung des Publikums über den Sieger miterleben.

In dieser und der kommenden Ausgabe wird ein Großteil der Teams ihre Projekte vorstellen und einen Blick unter die Motorhaube ihrer Anwendung werfen, den Anfang machen die drei Projekte mit den meisten Stimmen: Der Sieger der Challenge, das CATS-Projekt von Canoo, erhielt 221 Stimmen und macht daher den Anfang. Ihm folgen das zweitplatzierte Tipp-Top.net (86 Stimmen) und KIX (76 Stimmen). In der nächsten Ausgabe stellen sich dann Kick@26° (69 Stimmen), Zoccer (63 Stimmen) und Packung! (nicht unter den Finalisten) vor.



And the Winner is: **CATS**

Die W-JAX Challenge war für uns ein besonderer Anlass – nicht nur weil uns das Publikum dankenswerterweise zum Sieger gewählt hat, sondern vor allem weil wir einmal im Vergleich mit den besten unser Motto „Efficiency & Beauty“ mit Grails und Canoo ULC auf die Probe stellen konnten.

von Dierk König

Die Challenge stieß bei Canoo in Basel erst auf zurückhaltendes Echo. Sollten wir neben dem Tagesgeschäft auch noch „zum Spaß“ programmieren? So haben wir zunächst abgewartet, ob sich nicht ein anderes Grails-Team anmeldet. Als Sebastian Meyen dann aber twitterte „Wie? Die Grails Community kneift?“ (oder so ähnlich), war klar, dass wir etwas tun mussten. Die Einreichungsfrist war aber schon fast abgelaufen und so haben wir die Applikation nahezu vollständig am Wochenende vor dem Abgabetermin fertiggestellt – im Rahmen des Canoo Code Camps. Diese Veranstaltung nutzen wir, um zweimal im Jahr aus dem Tagesgeschäft auszubrechen. Wir reisen Mittwochabends ins Berner Oberland und programmieren bis Samstagmittag gemeinsam neue Ideen aus. Der Zeitaufwand war also 2,5 Tage, wobei man die Nächte auch mitrechnen muss! Lediglich die Ausgestaltung des Spiels haben wir vorher in einigen abendlichen Runden grob festgelegt.

Die sechs Teilnehmer organisierten sich in drei Paaren, denn im Code Camp ist Pairing Pflicht. Das Team bildeten vier Canooeys, ein ehemaliger Kollege und ein Student aus dem Begabtenkolleg des Karlsruher Institute of Technology (KIT), das Canoo fördert. Die Wahl der IDE stand jedem Pair offen. Wir haben eine gemeinsame Arbeitsinsel gebildet und uns auf Zuruf synchronisiert. Zusätzlich haben wir mit JIRA die Anforderungen an das Spiel verwaltet. Integriert wurde per SVN mit Teamcity für die Continuous Integration. Weil wir auf der

Grails-Plattform aufsetzen, war das automatische Build-System gesetzt, denn das kommt ja mit Grails bereits mit.

Wir haben also Grails als Plattform und programmierten Applikationscode vollständig in Groovy. Die erste Stärke dieser Wahl zeigt sich beim Team: Obwohl nur ein Teammitglied nennenswerte Erfahrung in Groovy hat, konnten alle sofort produktiv werden. Mit guten Java-Kenntnissen ist der Einstieg in Groovy sehr leicht. Der zweite große Vorteil resul-

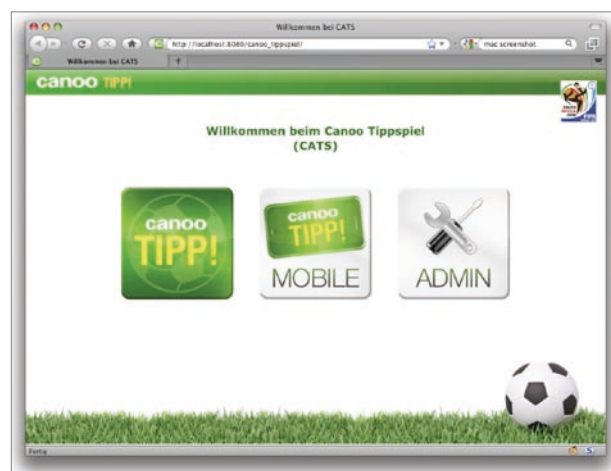


Abb. 1: Einstiegsseite

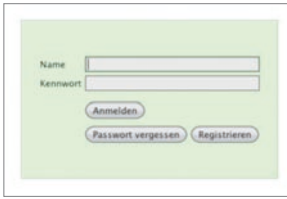


Abb. 2: Anmeldedialog

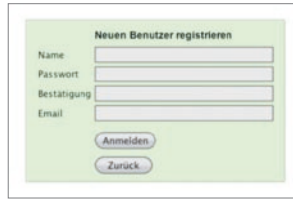


Abb. 3: Dialogrückseite

```

ULCBuilder builder = new ULCBuilder()
def save = builder.action(name: "Speichern", closure: save )

ULCBoxPane content = builder.boxPane(columns:1,verticalGap:30) {
  boxPane(id:"form", columns: 2, verticalGap:5, horizontalGap:5, background:GREEN)
  label "Name";          username = label("")
  label "Passwort";      password = passwordField()
  label "Bestätigung";   confirm = passwordField()
  label "Email";         email = textField()
  filler();              filler()
  filler();              button action: save, id: "go"
}
label icon: icon, constraints: BOX_CENTER_BOTTOM
}
    
```

Abb. 4: Code für den Dialog

tiert aus dem Ansatz von Groovy, die Reichhaltigkeit der Java-Plattform auszunutzen. Wir verwenden die Grails-Plug-ins für Spring Security, Mail, Canoo WebTest und Canoo ULC. Letzteres zusammen mit Erweiterungen für OpenGL: JOGL und SwOGL. Das heißt wir bekommen Authentisierung und rollenbasierte Autorisierung geschenkt, können Mails für vergessene Passwörter verschicken, haben funktionale Tests für unseren HTML-Kanal und können interaktive und animierte Komponenten bauen. Wir setzen auf eine klassische Webapplikationsarchitektur, die sämtlichen applikationsspezifischen Code – auch die Präsentationslogik – auf der Serverseite hält. Das schlanke Domänenmodell und das daraus automatisch abgeleitete Persistenzschema bestücken drei Benutzerkanäle, die von je einem Pair implementiert werden:

- einen Canoo-UltraLightClient-Kanal (www.canoo.com/ulc) für Power User
- einen iPhone-Kanal für mobiles Tippen per Ajax



Abb. 5: 3-D Coverflow Widget

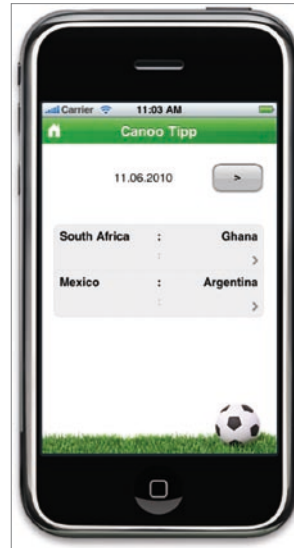


Abb. 6: iPhone-Kanal



- einen klassischen HTML-Kanal für die Administratoren

Die Einstiegsseite zeigt den Zugang zu den drei Kanälen (Abb. 1).

Alle Kanäle folgen einem klassischen MVC-Ansatz. Views und Controller werden für den HTML-Kanal dynamisch zur Laufzeit erstellt, d. h. sämtliche Standard-Views und Aktionen zum Anzeigen, Anlegen, Editieren und Löschen werden effektiv zu einem Einzeiler wie in:

```

import com.canoo.cats.User
class UserController {
  static scaffold = User
}
    
```

Businesslogik ist dem Vorgehen des Domain-driven Designs folgend mehrheitlich in Services ausgelagert. Die Wahl von Grails als Framework erlaubt es uns, bequem und schnell die Businesslogik zu erstellen und durch den Verzicht auf materialisierte Controller und Views eine hohe Änderungsfreundlichkeit und einen schnellen Mikroentwicklungszyklus umzusetzen.

Für den Power-User-Kanal kommt Canoo UltraLightClient (ULC) zum Einsatz, um dem Benutzer eine flüssige, reichhaltige Oberfläche zu geben, die sich verhält und aussieht wie eine Desktopapplikation. So werden sortierbare Listen mit In-place Editing, Flip-Animationen und sogar Elemente wie CoverFlow möglich. Dafür schreiben wir ausschließlich serverseitigen Applikationscode in sauberer Komponentenbauweise. Die Client-serverkommunikation übernimmt ULC. Das erste Beispiel dafür ist der einfache und übersichtliche Anmeldedialog (Abb. 2).

Die Überraschung kommt, wenn man sich registrieren will – das geschieht auf der Rückseite des Dialogs nach einer animierten, horizontalen Drehung des Dialogs. Schließlich sind Anmelden und Registrieren zwei Seiten einer Medaille. Leider kann man den tollen Effekt in einem Artikel nicht zeigen.

Man kann auch leicht wieder zurück-„flippen“. Den meisten Anwendern macht das so viel Spaß, dass sie das erst ein paar Mal ausprobieren, bevor sie die Applikation verwenden.

Der Applikationscode für solche Dialoge ist sehr übersichtlich und folgt dem in Groovy üblichen Builder Pattern. Hier ein Auszug für den Dialog, in dem ein Benutzer seine Daten verwalten kann:

Auf der Seite zum Tippen erlaubt ein 3-D CoverFlow Widget schnell durch die verschiedenen Gruppen (und Hauptrunden) zu blättern. Die jeweils zugehörigen Begegnungen sind darunter aufgeführt, und Tipps können durch direktes Schreiben in die Tabelle abgegeben werden. Das Ganze funktioniert mit direkter Anwahl einer Gruppe per Maus, Cursortasten, Mausrad oder Mac-Gesten (Abb. 5).

Die 3-D-Effekte bei CATS erreichen wir nicht über einfaches „Pixelschieben“ auf Clientseite, sondern wir projizieren echte, funktionale Swing-Komponenten auf eine beliebige OpenGL-Oberfläche. Den CoverFlow haben wir erst im Rahmen des CATS-Projekts gebaut.

Neben HTML und ULC hat CATS noch einen iPhone-Kanal für mobiles Tippen. Damit kann man das iPhone für eine kleine Zerstreung bei Wartezeiten nutzen. Der iPhone-Kanal setzt auf Ajax, wie es von Grails standardmäßig unterstützt wird. Dabei nutzen wir die Stärken von Grails Content Negotiation, dynamischem Rendering von Templates und JSON-Codierung aus. Es entsteht der Eindruck einer nativen Applikation, obwohl es „nur“ eine HTML-Seite im Safari-Browser ist (Abb. 6)

Der iPhone-Kanal benötigt weniger als 160 Lines of Code, die Gesamtapplikation knapp unter 2,5 kLOC. Diese Zahlen sind aber wie immer mit Vorsicht zu genießen. Was bei anderen Technologien als Konfiguration nicht zu den LOC gezählt wird, ist bei Grails ausführbarer Code. Andererseits sind HTML-Templates, Layouts, CSS und Ähnliches nicht enthalten.

Alle Kanäle unterliegen der Kontrolle der Spring Security, die über ein Grails-Plug-in angezogen wird. Grails stellt außerdem den Zugang zu den per Hibernate persistierten Daten zur Verfügung. In CATS teilen sich alle Kanäle die gleichen Hibernate-Objekte. Das erleichtert auch das Testen, denn man braucht es für alle Kanäle nur einmal zu tun. Unsere Unit-Tests

validieren sowohl persistentes Datenmodell wie auch die darauf aufbauenden Services dank Grails Mocks sogar ganz ohne Datenbank. Funktionale Tests mit Canoo WebTest überprüfen insbesondere die Übertragung der Rankings für die nachfolgenden Runden. 95 % aller Requests laufen in WebTest in weniger als einer Sekunde durch. Die Software entspricht dem JEE-Standard und ist clusterfähig, d. h. auch viele Spieler mit vielen Tipps sollten kein Problem sein.

CATS ist einfach konfigurierbar und kann auch für zukünftige WMs mit wenig Aufwand wiederverwendet werden. So werden unter anderem die Icons für die Vorrundengruppen automatisch aus den Flaggen der jeweiligen Teilnehmerländer generiert. Wir haben bereits Pläne für weitere Ausbaustufen und werden das Tippspiel rechtzeitig zur WM für den produktiven Einsatz rüsten. In der Rückschau meinen wir das Versprechen von „Efficiency & Beauty“ gut eingelöst zu haben. In so kurzer Zeit diese Menge an Funktionalität zu liefern, hatten wir selbst nicht erwartet – und das Erscheinungsbild scheint auch gefallen zu haben.

Viel Spaß beim Tippen!

Das CATS-Team

Dierk ist Fellow bei der Canoo Engineering AG, Basel. Er betreibt das Open-Source-Projekt Canoo WebTest und ist Committer in den Projekten Groovy, Grails und GPar. Er publiziert und spricht auf internationalen Konferenzen zu Themen moderner Softwareentwicklung. Er ist Autor des Buchs „Groovy in Action“. Twitter: @mittie

Links & Literatur

- [1] Beispielanwendung: http://cats.canoo.com/canoo_tippspiel
- [2] Quellcode: <http://people.canoo.com/mittie/download/cats-open.zip>

1/3 Wibu