Erweiterung eines Webdienstes um einen mobilen Webzugang

In diesem Projekt[1] wird ein webbasierter Dienst um einen Zugang für Mobiltelefone erweitert. Die mobile Webapplikation stellt dem Benutzer eine Teilmenge der Funktionalität des Basisdienstes über ein entsprechend optimiertes Benutzerinterface zur Verfügung. Im Artikel werden wichtige Entscheidungen in der Systemarchitektur erklärt, damit die mobile Webapplikation jederzeit flexibel an die Bedürfnisse der Benutzer und an die Vorgaben des Basisdienstes angepasst werden kann. Wir wollen aber auch zeigen, dass sich die Entwicklung einer mobilen Webapplikation nicht gross von der Entwicklung einer normalen Webapplikation unterscheidet.

Jürg Luthiger | juerg.luthiger@fhnw.ch

Der mobile Markt ist ein Zukunftsmarkt; entsprechend wichtig ist es die Entwicklungen in diesem Bereich nicht zu verpassen. Ein Dienstleister verspricht sich mit einem mobilen Zugang eine Attraktivitätssteigerung seiner Plattform sowohl für die Benutzer, für die Online Werber wie auch für die Partner. Ausserdem können neue Benutzergruppen oder Geschäftsmodelle erschlossen werden.

Webbasierte Dienste erlauben einen raschen Einstieg in den mobilen Markt. In der Regel kann man einfache Dienste mit einem modernen, mobilen Webbrowser problemlos nutzen. Werden bei der Entwicklung solcher Seiten die Eigenschaften mobiler Browser aber zu wenig berücksichtigt, können unschöne Effekte bei der Darstellung der Webseiten auftreten.

In diesem Artikel wollen wir ein paar Entscheidungen aufgreifen, die wir bei einer konkreten Umsetzung eines mobilen Webzugangs zu einem bestehenden Dienst getroffen haben. Die Überlegungen betreffen die Integration in das Backend System. Es ist wichtig festzuhalten, dass diese Überlegungen im Rahmen einer Prototypen-Entwicklung entstanden sind. Mit diesem Prototyp haben wir die Benutzerbedürfnisse anhand einer konkreten mobilen Anwendung aufgenommen und getestet. Die Erkenntnisse aus diesem Projekt sind zwischenzeitlich in den mobilen Zugang für das Doodle System eingeflossen [DooMob].

Kontext

Doodle AG bietet einen Terminplaner als Gratisdienstleistung für alle Internetbenutzer an [Doodle]. Die Einstiegshürde ist besonders tief, denn Doodle erfordert keine Registrierung, keine Softwareinstallation und ist sehr einfach zu bedienen. Deshalb ist dieser Dienst sehr beliebt und in der Schweiz der klare Führer bei den Terminfindungswerkzeugen und eine der meistbesuchten Websites der Schweiz.

Der Doodle-Dienst unterstützt auf eine effiziente Art die Terminfindung in verteilten Teams. Der Initiator erstellt über ein Webformular eine Terminumfrage, indem er den Zweck und verschiedene Terminvorschläge angibt. Aus der Anfrage generiert Doodle eine Einladung in Form einer eindeutigen Webadresse. Diese URL kann der Initiator nun via E-Mail an das restliche Team senden. Die Teammitglieder wählen anschliessend ihre Präferenzen auf der entsprechenden Webseite und Doodle erstellt eine Statistik der gewählten Termine. So kann das Team sehr schnell einen passenden Termin finden.

Mobile Doodle-Benutzer meldeten jedoch, dass die Webseiten in den mobilen Webbrowsern teilweise falsch dargestellt werden und dass besonders die Termine, die standardmässig in einer grossen Tabelle aufgelistet sind, zu übermässigem Scrollen führen.

Doodle AG hat sich deshalb entschlossen diese Kundenwünsche aufzunehmen und der mobilen Benutzergruppe ein entsprechend optimiertes Webinterface anzubieten.

Problemstellung

Bei der Entwicklung dieses mobilen Dienstes sollen folgende Fragenstellungen adressiert werden:

- Welche Funktionalität soll dem Benutzer zur Verfügung stehen? Obwohl die Benutzung der Doodle-Plattform über den Webclient einfach und selbsterklärend ist, scheint es nicht sinnvoll zu sein, auch für den mobilen Webclient den vollen Funktionsumfang anzubieten. Auf einem mobilen Gerät sind zum Beispiel die Eingabemöglichkeiten beschränkter, vor allem die Eingabe von Text kann mühsam werden.
- Welche Interaktionen sind für den Nutzer mit der mobilen Anwendung sinnvoll? Die Interaktionsmöglichkeiten der Doodle-Plattform sind

1 Mit freundlicher finanzieller Unterstützung der Hasler Stiftung.

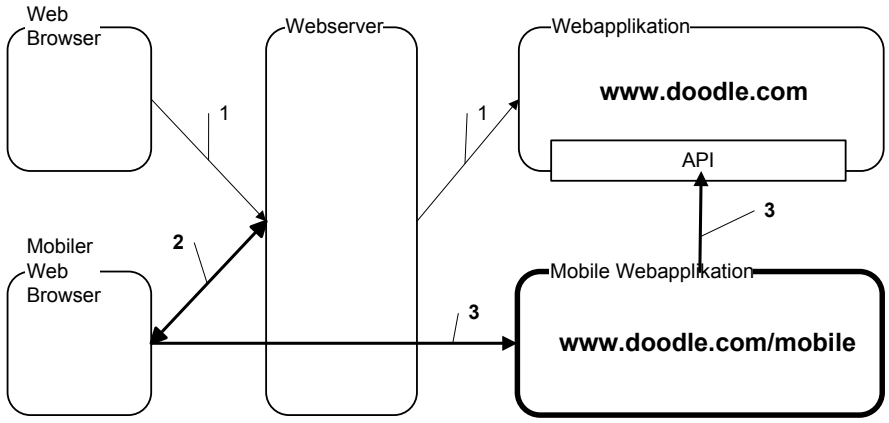

Abbildung 1: Konzept

auf die Möglichkeiten eines Desktop Systems mit Tastatur und Maus ausgelegt. Auf einem mobilen Gerät stehen diese Inputmöglichkeiten jedoch nicht zur Verfügung. Die mobile Plattform muss dies berücksichtigen.

- WiekanndasbestehendeDoodleBackendsinnvoll erweitertwerden,umweiterenClientTypeneinen sicheren, zuverlässigen Zugang zu ermöglichen? Die Systemarchitektur der Doodle-Plattform ist eng mit der Weblösung gekoppelt. Diese enge Koppelung zwischen Backend und Web-Client muss aufgelöst und durch eine lose Koppelung ersetzt werden. Erst dann wird es möglich sein, weitere Client-Typen wie die mobile Plattform in das zentrale Doodle-System einzubinden.

Funktionalität

Mobile Benutzer brauchen die Software-Dienste auch ausser Haus. Ihre Bedürfnisse an den Dienst sind in diesen Situationen in der Regel aber nicht gleich, wie wenn sie zu Hause vor dem PC sitzen und die Site von dort aus besuchen. Oft wollen sie sich aktuelle Informationen beschaffen oder kleinere, klar terminierte Arbeiten erledigen. Es ist deshalb wichtig, den mobilen Nutzer als ein neues Kundensegment zu betrachten und die mobile Website fokussiert auf diese Bedürfnisse auszurichten. Daher kann es durchaus Sinn machen, gewisse Funktionalität des Basisdienstes nicht in der mobilen Version anzubieten. Die Benutzeranalyse zum Beispiel ergab, dass die Teilnahme an einer bereits existierende Terminumfrage im Vordergrund steht, um auch ausser Haus die eigenen Präferenzen bei einer Einladung einreichen zu können.

Da die Funktionalität des mobilen Dienstes in der Regel vom Basisdienst abweicht, ist es wichtig, dass der Basisdienst komplett unabhängig vom mobilen Client bleibt. Darum wird die mobile Website in diesem Projekt als eine eigenständige Webapplikation entwickelt, die parallel zum Basisdienst betrieben werden kann (siehe Abb. 1). Gleichzeitig entsteht auf dem Basisdienst eine Schnittstelle [DooAPI], die von beliebigen Clients programmatisch genutzt werden kann und auch von der mobilen Webapplikation angesprochen wird. Diese starke Separierung ergibt eine grosse Unabhängigkeit und erlaubt das problemlose Testen verschiedener Alternativen auf der mobilen Seite. Dies ist ein grosser Vorteil vor allem in der Phase des Prototypenbaus.

Interaktionsmöglichkeiten

Die modernen, mobilen Webbrowser beherrschen die aktuellen Webstandards. Deshalb ist ein Einsatz veralteter WAP-Technologien [WAP] überflüssig. Stattdessen lassen sich die heute gängigen Webtechnologien einsetzen, wie XHTML und CSS und zur dynamischen Aufbereitung Java Server Faces (JSF). JavaScript kann ebenfalls genutzt werden.

Der Zugang auf eine mobile Website muss möglichst einfach gestaltet werden, da das Eintippen einer URL auf einem mobilen Gerät mühsam ist. Das Platzieren des entsprechenden Links auf der Startseite des Basisdiensts stellt eine erste Variante dar. Man kann jedoch auch die Header-Informationen einer Browseranfrage nutzen, um eine Umleitung auf eine eigenständige mobile Site automatisch zu aktivieren und den Benutzer komplett von einer manuellen Eingabe zu befreien. Da jeder Browsertyp eine eigene Kennung hat – sie ist unter dem Schlüsselwort „user-agent“ im Header abgelegt – ist auf dem Server eine Identifikation des Browsers problemlos möglich. In dieser zweiten Variante wird die Webapplikation als auch die mobile Webapplikation über die gleiche URL angesprochen, zum Beispiel mit www.doodle.com (siehe Abb. 1). Mit den Header-Informationen aus dem *Request* wird nun entschieden, ob die Anfrage von einem normalen oder von einem mobilen Browser stammt. Im ersten Fall wird der Webserver die Anfrage an den Basisdienst weiterleiten (siehe 1 in Abb. 1) und im zweiten Fall wird eine geeignete Komponente im Webserver ein *Redirect* auf die mobile Webapplikation initiieren (2). Die mobile Applikation kann dann über das Doodle API mit dem Basisdienst kommunizieren (3).

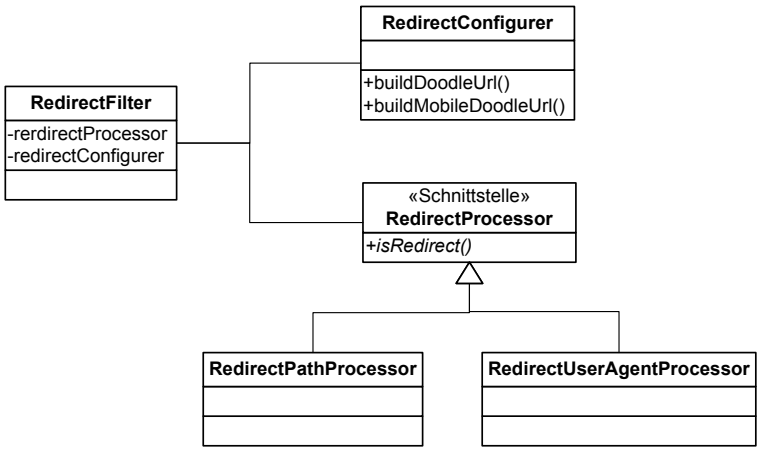

Abbildung 2: Klassendiagramm Filterkonzept

Die Erkennung des Browser-Typs auf dem Server wird in Java am einfachsten mit einer Filterkomponente umgesetzt. In Abbildung 2 ist das entsprechende Klassendiagramm aufgezeichnet, welches aus der Anforderung entstanden ist, Varianten für das Umleiten (*Redirect*) austesten zu können. Der Filter hat dabei je eine Referenz auf die Klasse *RedirectConfigurer* und das Interface *RedirectProcessor*, zu welcher momentan zwei konkrete Implementierungen existieren. Die Klasse *RedirectPathProcessor* dient lediglich zu Testzwecken; sie untersucht den Pfad der Anfrage. Die Klasse *RedirectUserAgentProcessor* hingegen verarbeitet die Header-Informationen, im Speziellen den Eintrag in der Zeile „user-agent", um eine Umleitung feststellen zu können. Dank dieser Entkoppelung der Funktionalität reduziert sich die Logik in der Filterklasse *RedirectFilter* zu den Schritten in Listing 1.

In Zeile 7 wird der aktuell konfigurierte *processor* aufgerufen, um festzustellen ob eine Umleitung ausgelöst werden muss. Falls ja, erstellt der *configurer* die entsprechende URL, welche nun über die *response* an den Browser zurückgeschickt wird. Falls nein, kann der *request* in der Filterkette weitergegeben werden (Zeile 11). In den Zeilen 2 bis 5 erkennt man den Einsatz des Spring Frameworks [Spring]. Die Konfiguration der Klassen *RedirectProcessor* und *RedirectConfigurer* können dank dieses Frameworks externalisiert werden und erlauben eine weitere Abstraktion der Klassen untereinander, was für das flexible Testen der verschiedenen Komponenten sehr nützlich ist.

Anbindung an das Doodle Backend

Wie in Abbildung 1 dargestellt, wird das Basissystem über eine Schnittstelle (API) in die mobile Webapplikation integriert. Es ist ganz wichtig, dass die Funktionalität nicht in den Doodle Clients dupliziert werden muss und deshalb hat die Gestaltung der Schnittstelle eine grosse Bedeutung. Das API soll alle Anwendungsfälle unterstützen, die über die Client-Applikation abgewickelt werden müssen. In einer Projektphase, wo eine solche Schnittstelle ständigen Anpassungen aus praktischen Gründen unterworfen ist, macht es Sinn, ein Beschreibungskonzept zu wählen, welches sehr flexibel ist und dadurch schnelle Änderungen erlaubt.

Wir haben die Doodle-Schnittstelle in der REST-Technologie gestaltet und umgesetzt. So kann gleichzeitig an der Client-Applikation entwickelt und serverseitig die Schnittstelle den Bedürfnissen des Clients angepasst werden. Die Beschreibung der Schnittstelle erfolgt mit einem

```
1   ...
2   WebApplicationContext wac =
3       WebApplicationContextUtils.getRequiredWebApplicationContext(
            getServletContext());
4   RedirectProcessor processor =
        (RedirectProcessor)wac.getBean("redirectProcessor");
5   RedirectConfigurer configurer =
        (RedirectConfigurer)wac.getBean("redirectConfigurer");
6
7   if (processor.isRedirect(request)) {
8       String redirectString =
            configurer.buildMobileDoodleUrl(pollId, processor.getDetails());
9       response.sendRedirect(response.encodeRedirectURL(redirectString));
10  } else {
11      chain.doFilter(request, response);
12  }
13  ...
```

Listing 1: Filterlogik

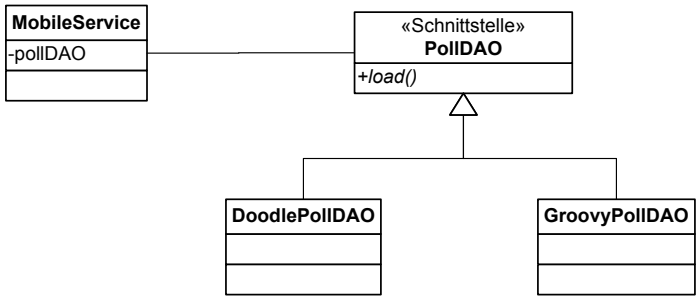

Abbildung 3: Einsatz des DAO Patterns zur Abstraktion der Doodle Systemintegration

XML-Schema, welches genutzt werden kann, um mit einem geeigneten Framework entsprechende Java-Klassen zum Lesen und Schreiben der Kommunikationsdaten automatisch zu generieren. Sie erlauben das Umwandeln von Java-Objekten in eine XML-Darstellung und umgekehrt. Bei Schemaänderungen werden diese Klassen neu generiert, was vollständig automatisiert werden kann. Weitere Anpassungen im Klassenmodell des Clients sind nur bei grösseren Änderungen in der Schnittstelle erforderlich. Jedoch sind mit der XML-Variante Anpassungen zur Laufzeit nicht möglich.

Während der Entwicklungsphase können die Änderungen aber doch gross sein. Daten müssen zum Beispiel besser strukturiert und in verschiedene Klassen aufgeteilt werden. In dieser Phase empfiehlt sich der Einsatz einer Skriptsprache, da diese Sprachen in der Regel eine kompaktere Implementierung und Anpassungen auch zur Laufzeit erlauben.

Im Java-Umfeld ist Groovy [Groovy] ein möglicher Kandidat. Mit Groovy ist eine sehr kompakte Implementierung der Anbindung möglich. Listing 2 stellt das Groovy-Skript dar, um über eine gegebene URL das XML-Dokument auf der Serverseite auszulesen (Zeile 4), den Inhalt zu parsen und das entsprechende Domänenmodell aufzubauen (Zeilen 6 bis 31). Die entsprechende Implementierung der Variante mit dem XML-Schema ist um ein Mehrfaches grösser und umfasst über 200 Zeilen Java Code.

Um eine Abstraktion zwischen der mobilen Webapplikation *MobileService* und den Server-

```
1   ...
1 class GroovyPollDAO implements PollDAO {
2       Poll load(String id) {
3           def poll = new Poll()
4           def pollSrc = new XmlParser().parse(«$id»)
5
6           poll.description = pollSrc.description.text()
7           poll.title = pollSrc.title.text()
8           poll.initiator = pollSrc.initiator.text()
9           poll.type = (pollSrc.type.text == ‹DATE›) ? poll.type = Poll.Type.DATE
10                                          : poll.type = Poll.Type.TEXT
11          poll.timeZone = pollSrc.timeZone.text()
12          poll.levels = pollSrc.levels.text()
13
14          pollSrc.participants.participant.each {participant ->
15              def vote = new Vote(participant.name.text(), new LinkedList())
16              participant.preferences.option.each { option ->
17                  vote.votes << option.text()
18              }
19              poll.votes << vote
20          }
21
22          pollSrc.options.option.each { opt ->
23              def option
24              if (poll.type == Poll.Type.DATE) {
25                  option = new DateOption(opt.text(), Locale.GERMAN)
26              } else if (poll.type == Poll.Type.TEXT) {
27                  option = new TextOption(opt.text())
28              }
29              poll.options << option
30          }
31          return poll
32      }
33  }
```

Listing 2: Groovy Skript für die flexible Anbindung des Doodle Backends

Zugriffsklassen wie das Groovy Skript zu erhalten, empfiehlt sich der Einsatz des Data Access Object (DAO) Design-Patterns (Abbildung 3). Das DAO übernimmt die Kommunikation mit dem Server und die Umwandlung der Daten in das korrekte Schnittstellenformat. Durch die strikte Trennung zwischen Nutzung und Besorgung der Daten können verschiedene Implementierungen zur Serveranbindung einfach getestet werden, da sie ohne Auswirkungen auf den Client austauschbar sind.

Zusammenfassung

Die Entwicklung einer mobilen Webapplikation unterscheidet sich nicht wesentlich von der Entwicklung einer normalen Webanwendung; es lassen sich die gleichen Technologien einsetzen. Das Spring Framework [Spring] bildet bei unserem Prototyp das Fundament. Mit diesem Framework kann man sehr einfach verschiedene Implementierungen testen. Bei der View-Technologie kam Java Server Faces (JSF) zum Einsatz, da Doodle JSF auch für ihre normale Webapplikation nutzt.

Es hat sich in unsere Analyse gezeigt, dass die modernen mobilen Webbrowser sehr mächtig sind und viele Webstandards (XHTML, CSS, JavaScript) unterstützen. Ebenfalls sind die mobilen Telefone mit den entsprechenden Webbrowsern leistungsfähig genug, um eine normale Webseite in nützlicher Zeit aufzubereiten und darzustellen. Dennoch gibt es zwischen den Browsertypen immer wieder kleinere Unterschiede, die nur durch entsprechende Tests eruiert werden können. Wir haben während der Analysephase alle Interaktionselemente, die zum Einsatz kommen sollen, mit den verschiedenen Browsern auf Darstellung und Benutzung untersucht. Grosse Unterschiede zeigen sich vor allem zwischen den Browsern auf normalen mobilen Telefonen und Smartphones. So erstaunt zum Beispiel sehr, wie unterschiedlich eine Auswahlliste mit Mehrfachauswahl dargestellt werden kann.

Bei der Portierung einer Webapplikation auf mobile Geräte muss das mobile User Interface genau analysiert werden. In vielen Fällen wird man die Benutzeroberfläche neu gestalten müssen, um den kleineren Bildschirmen, der fehlenden Tastatur und Maus Rechnung tragen zu können. Es ist zum Beispiel wichtig, die Texteingaben auf ein Minimum einzuschränken, da dies auf einem Handy viel schwerfälliger ist als bei einem normalen Computer mir Tastatur. Für einen Teil der normalen Webseiten kann man mit einer Anpassung des CSS Stylesheets die Portierung an die mobilen Geräte lösen. Muss der Informationsgehalt hingegen verdichtet werden, wird die Entwicklung einer eigenen Webseite unumgänglich.

Die Browserkennung über das Header-Kennwort „user-agent“ wird von den verschiedenen Browser sehr gut und detailliert unterstützt. Das Kennwort kann auf der Serverseite sinnvoll genutzt werden, um zum Beispiel Serverdienste oder CSS Stylesheets dem Browser entsprechend programmatisch auszuwählen.

Wir haben unseren Prototyp als eigenständige Webapplikation realisiert, um aus Gründen der unterschiedlichen Tests eine möglichst lose Kopplung an den Doodle-Dienst zu etablieren. Dabei haben wir gesehen, dass eine flexible Serveranbindung bei einem instabilen Server-API am besten mit einer Skriptsprache zu realisieren ist. Sobald sich das API aber stabilisiert hat, kann ein Wechsel zu einer anderen, engeren Serveranbindung aus Performanzgründen Sinn machen.

Doodle hat in der Zwischenzeit auf ihrem produktiven System einen mobilen Zugang aufgeschaltet [DooMob]. Dabei haben sie verschiedene Resultate aus dem vorliegenden Projekt aufgenommen. Besonders die Erkenntnisse bei der Gestaltung der Benutzeroberfläche sind sehr wichtig gewesen. Für die Implementierung der mobilen Webapplikation haben sie selber eine engere Anbindung an den Server gewählt und somit die aus einer Anbindung über das Doodle-API resultierenden Performanceverluste verhindert.

Referenzen

[Doodle] Website Doodle AG, http://doodle.com/

[DooAPI] RESTful Doodle, http://www.doodle.com/xsd1/RESTfulDoodle.pdf

[DooMob] Mobile Website von Doodle AG, http://doodle.com/mobile/main.html

[Groovy] Groovy Website, http://groovy.codehaus.org/

[Spring] Website Spring Framework, http://www.springsource.org/

[WAP] Wireless Application Protocol, Wikipedia, http://de.wikipedia.org/wiki/Wireless_Application_Protocol

[Zob01] Zobel, J. Mobile Business und M-Commerce. Hanser, 2001