Eine Einführung in das Google Web Toolkit — oder — Closed-Source zu Open-Source Moderne Altlastenentsorgung?

David Schwarz und Bernd Müller

Fachhochschule Braunschweig/Wolfenbüttel
Fachbereich Informatik
Am Exer 2, 38302 Wolfenbüttel
{david.schwarz,bernd.mueller}@fh-wolfenbuettel.de

Abstract: In den letzten Jahren wurden einige kommerzielle Software-Produkte zu Open-Source-Systemen deklariert. Dies kann durch verschiedenste Gründe motiviert sein, z.B. durch mangelnde Qualität der Systeme oder einen kommerziellen Misserfolg und damit ein stark nachlassendes unternehmerisches Interesse an den Systemen. Wir sind der Meinung, dass dies in vielen Fällen nicht so ist, sondern dass entsprechende Systeme der näheren Vergangenheit Maßstäbe in ihrem jeweiligen Gebiet gesetzt haben. Wir stellen das Google Web Toolkit als das jüngste Mitglied dieser erkennbaren closed-source zu open-source Bewegung vor und motivieren, dass dieses System durchaus die Messlatte im Bereich der Frameworks für interaktive Web-Anwendungen gesetzt hat bzw. im Augenblick dabei ist, diese zu setzen.

1 Einführung

Die open-source Software-Entwicklung ist mittlerweile eine akzeptierte Option der IT-Welt. Die entstehende, frei erhältliche Software wird auch im kommerziellen Bereich vielerorts eingesetzt. Die umgekehrte Richtung vom kommerziellen in den nichtkommerziellen Bereich ist weniger ausgeprägt, weniger häufig und scheinbar auch nicht attraktiv. Dem Vorgang, eine closed-source entwickelte Software open-source zu machen, hängt der fade Beigeschmack des Erfolglosen an. Warum sollte eine Firma, die erhebliche Aufwände in die Entwicklung einer Software investiert hat, diese open-source machen und somit auf Einnahmen aus Lizenzkosten verzichten? Kann dies nicht nur darin begründet sein, dass aus der kommerziellen Vermarktung kein Gewinn zu erzielen ist? Muss die Software nicht zwangsweise qualitativ schlecht und wenig zukunftsträchtig sein?

Dass dies nicht so ist, belegen viele Beispiele der jüngeren Vergangenheit. Sun Microsystems Inc. machte sein StarOffice zu open-source Software, es entstand OpenOffice. Die IBM Corp. legte auf diese Art und Weise den Grundstein für Eclipse, Netscape Communications Corp. den Grundstein für Mozilla, Firefox und Thunderbird. Alle genannten

open-source Systeme sind mittlerweile sowohl im privaten als auch im kommerziellen Bereich sehr weit verbreitet und stehen in einem ausgezeichneten Ruf bzgl. Ihrer Qualität und Funktionalität.

In diesem Beitrag wollen wir die aktuellste Entwicklung aus dem Bereich closed-source zu open-source vorstellen, das Google Web Toolkit (GWT) [gwta]. GWT ist ein von der Google Inc. entwickeltes Java-Framework für AJAX-basierte Web-Applikationen. Die zentrale Idee ist die server-seitige Entwicklung von Anwendungen in Java, deren Oberfläche ebenfalls in Java definiert, dann aber nach JavaScript übersetzt wird und in Kombination mit HTML und CSS von jedem Browser dargestellt werden kann. Google Inc. machte im Dezember 2006 GWT mit Version 1.3 zu einer open-source Anwendung, die unter der Apache-Lizenz 2.0 steht. GWT ist die Basis vieler Google-eigener Anwendungen, wird aber auch von anderen Unternehmen und Privatpersonen eingesetzt.

Wir gehen in Abschnitt 2 auf grundlegende Konzepte des GWT ein. Wir beschreiben die Architektur des Systems und entwickeln ein kleines Beispiel. In Abschnitt 3 führen wir einige Systeme auf, die GWT verwenden. Der Beitrag endet in Abschnitt 4 mit einer kurzen Zusammenfassung.

2 Grundlegende Konzepte und Architektur des GWT

2.1 Ziele

Das Web der 90-er Jahre und der ersten Jahre des neuen Jahrtausends war durch statische HTML-Seiten und CGI-Anwendungen geprägt. Java-Applets und ActiveX-Komponenten brachten ein wenig mehr Dynamik in Web-Anwendungen, diese blieben aber im Vergleich zu Desktop-Anwendungen in ihrer intuitiven Bedienbarkeit, Interaktivität und Antwortzeitverhalten zurück. Durch die Einführung des `XMLHttpRequests` durch Microsoft wurde ein API definiert, das es JavaScript oder anderen Skriptsprachen ermöglicht, Anfragen an den Server zu stellen und XML-Strukturen als Antworten zu erhalten, die beliebig verwendbar sind und in der Regel zu einer Aktualisierung der HTML-Seite verwendet werden. Das Konzept des `XMLHttpRequests` ist Grundlage des AJAX-Hypes (Asynchronous JavaScript and XML) und weiter des sogenannten Web 2.0.

Das Ziel von Google bei der Entwicklung von GWT war die möglichst einfache Entwicklung hochinteraktiver client-server-basierter Web-Anwendungen, also z.B. Möglichkeiten für Drag-and-Drop und Verbergen des Request-Response-Zyklus und seiner sichtbaren Auswirkungen. Neben diesen für den Benutzer sichtbaren Ziele sollte eine server-seitige Entwicklung mit reinem Java erfolgen, um den Entwickler die Einarbeitung in ein zusätzliches, proprietäres GUI-System zu ersparen und den Einsatz herkömmlicher Entwicklungswerkzeugen zu ermöglichen.

2.2 Konzepte und Architektur

Als zentrales Konzept realisiert GWT eine rein Java-basierte Entwicklung. Dazu wird eine GUI-Komponentenbibliothek definiert, die Komponenten wie `Button`, `RadioButton`, `TextBox` und `PasswordTextBox`, aber auch kompliziertere Komponenten wie `MenuBar`, `Table` und `Tree` enthält. Diese nicht vollständige Aufzählung verdeutlicht, dass ähnliche GUI-Elemente wie in AWT, SWT oder Swing vorhanden sind und daher Entwickler, die mit diesen Bibliotheken vertraut sind, nicht umlernen müssen. Die Komponenten werden dem MVC-Muster entsprechend verwendet und über registrierte Listener mit dem Model, das aus einfachem Java-Code besteht, verbunden.

Der GWT-Compiler übersetzt ein solches Java-Programm in HTML-Seiten mit eingebettetem JavaScript. Der Entwickler kommt weder mit HTML noch mit JavaScript in Berührung. Da sich Java und JavaScript in ihrer Ausdrucksmächtigkeit und bezüglich ihrer Datentypen deutlich unterscheiden, muss sich GWT beschränken und begrenzt die verwendbaren Java-Klassen auf die beiden Packages `java.lang` und `java.util`. Aber auch diese beiden Packages sind nicht vollständig implementiert, so fehlen z.B. `LinkedList` und `Hashtable`. Die grundlegende Funktionsweise stellt Abbildung 1 dar.

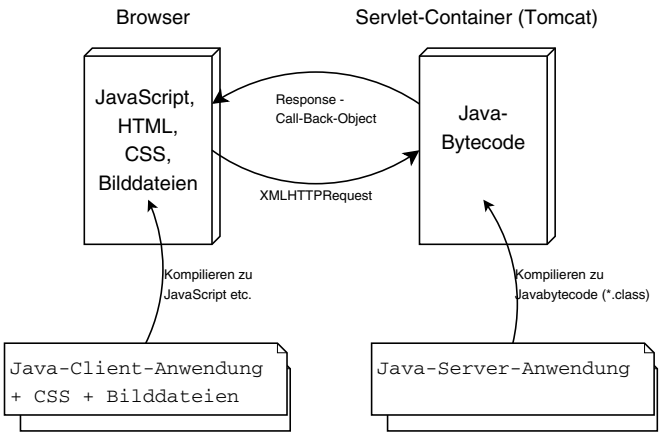

Abbildung 1: GWT-Modell

2.3 Entwicklung von GWT-Anwendungen

Das GWT-System enthält eine Reihe von Kommandozeilenwerkzeugen, die die Erstellung von GWT-Projekten erleichtern. Das Skript `projectCreator` erstellt dazu ein Eclipse-Projekt mit einem Ant-File, das verschiedene Entwicklungsschritte automatisiert. Wir verwenden hier alternativ das Eclipse-Plugin *Googlipse* [goo], das ebenfalls ein lauffähiges Projektskelett erzeugt. Das Kompilieren und Deployen wird über eine Run-Konfiguration

in Eclipse gesteuert, so dass keine Ant-Files verwendet werden müssen.

Ein GWT-Projekt wird nahezu komplett in Java implementiert. Es ist lediglich nötig, den generierten Javascript-Code in eine minimale HTML-Datei einzubinden. Der Sourcecode eines GWT-Modules gliedert sich in drei Java-Packages:

- `client`: Der client-seitige Code, welcher nach Javascript übersetzt wird.
- `server`: Der server-seitige Code, welcher im Application-Server ausgeführt wird.
- `public`: Sonstige Dateien, wie Stylesheets, HTML, Bilder, ...

2.3.1 Die Anwendung auf Client-Seite

Eine GWT-Anwendung besitzt einen definierten Eingangspunkt, vergleichbar den Methoden `init()` und `main()` bei Applets und Standalone-Programmen. Dieser Eingangspunkt implementiert das Interface `EntryPoint` und überschreibt die Methode `onModuleLoad()`. Zur Laufzeit bewirkt der aus dieser Java-Klassen entstandene JavaScript-Code beim Laden in den Browser das Starten der Anwendung:

```
public class MyGwtModule implements EntryPoint {
  MyApplication myApp;
  public void onModuleLoad() {
    Button b = new Button("Start", new ClickListener() {
      public void onClick(Widget w) {
        myApp = new MyApplication();
        myApp.start();
      }
    });
    RootPanel.get().add(b);
  }
}
```

Als Wrapper des JavaScript-Codes muss eine minimale HTML-Seite erstellt werden. Diese hat, falls die Anwendung nur auf GWT basiert, lediglich das erstellte JavaScript im `head`-tag einzubinden:

```
<meta name='gwt:module' content='src.MyGwtModule'>
<script language="javascript" src="gwt.js"></script>
```

Zusätzlich kann in dieser Datei die Unterstützung für das History-Management (Back-Button) konfiguriert werden. Im Source-Verzeichnis des Clients existiert noch eine XML-Konfigurationsdatei die definiert, welche Klasse der Einstiegspunkt der Anwendung ist, von welcher Oberklasse der Einstiegspunkt erbt und welche Klassen Remote-Services implementieren.

2.3.2 Methodenaufrufe auf dem Server

Um einen Methodenaufruf auf dem Server ausführen zu können, müssen client-seitig zwei Interfaces definiert werden. Das erste Interface deklariert die auszuführende Methode mit

ihrem Rückgabewert, das zweite Interface enthält dieselbe Methode mit dem Unterschied eines zusätzlichen Parameters für den asynchronen Call-Back und dem Rückgabetyp `void`:

```
public interface MyRemoteService extends RemoteService {
  public boolean login(String user, String pwd);
}

public interface MyRemoteServiceAsync {
  public void login(String user, String pwd, AsyncCallback callback);
}
```

Diese Methode ist server-seitig zu implementiert. Dafür ist eine eigene Klasse von der Klasse `RemoteServiceServlet` abzuleiten, welche das Interface `MyRemoteService` implementiert:

```
public class MyRemoteServiceImpl extends RemoteServiceServlet
             implements MyRemoteService{
  public boolean login(String user, String pwd) {
    if(user.equals("David") && pwd.equals("geheim")){ return true; }
    return false;
  }
}
```

Um die Implementierung des Remote-Service über ein URL erreichbar zu machen, wird ein Mapping des URLs auf die implementierende Klasse in der Konfigurationsdatei definiert. Die Methode kann nun client-seitig aufgerufen werden. Dazu werden vom GWT-Laufzeitsystem Methoden bereitgestellt, die eine Referenz auf das Service-Objekt zurück geben, im folgenden Beispiel `/mainService`:

```
MyRemoteServiceAsync service = (MyRemoteServiceAsync) GWT
          .create(MyRemoteService.class);
// Ziel url angeben:
ServiceDefTarget endpoint = (ServiceDefTarget) service;
String moduleRelativeURL = GWT.getModuleBaseURL() + "mainService";
endpoint.setServiceEntryPoint(moduleRelativeURL);
```

Um den Methodenrückgabewert zu verarbeiten, stehen die Methoden `onSuccess()` und `onFailure()` des Call-Back-Objektes zur Verfügung. Der Rückgabewert muss in den entsprechenden Typ gecastet werden:

```
// Asynchronen Callback erstellen um das ergebnis zu handlen
AsyncCallback callback = new AsyncCallback() {
  public void onSuccess(Object result) {
    if (((Boolean) result).booleanValue()) {
      Window.alert("Erfolg!");
    }
  }
  public void onFailure(Throwable caught) {}
};
```

Nun kann der Aufruf durchgeführt werden. Die Call-Back-Methoden werden automatisch aufgerufen, wenn die Antwort vom Server eintrifft oder fehlerhafterweise ausbleibt und einen Time-Out erzeugt:

```
service.login("David", "geheim", callback);
```

3 Anwendungen auf Basis von GWT

Google Maps (`maps.google.com`) und *Google Mail* (`gmail.google.com`) sind GWT-basierte Anwendungen. Insbesondere Google Maps hat sicher sehr viel zum Erfolg und Siegeszug von AJAX und des Web 2.0 beigetragen. Mit *Google Docs & Spreadsheets* (`docs.google.com`) versucht sich Google als Anbieter einer web-fähigen Office-Alternative zu etablieren.

Eine Reihe eher spielerischer Anwendungen auf Basis von GWT sind unter dem Stichwort *GWT* im Google-Wiki [gwtb] aufgezählt. Es sind dies unter anderem Spiele wie Tic-Tac-Toe, Sudoku und Hangman. Man findet dort aber auch kleines Computer-Algebra-System oder einen Text-Editor.

Der interessierten Leser findet unter [kit] eine Demo der verschiedenen GUI-Elemente.

4 Zusammenfassung

Der Vorgang, dass Unternehmen Systeme, die unter hohem finanziellem Aufwand entwickelt wurden, zu Open-Source-Systemen erklären, ist relativ selten. Die Motivation dafür ist, zumindest aus betriebswirtschaftlicher Sicht, nicht sofort offensichtlich. Wir haben in diesem Beitrag das jüngste System, das diesen Weg gegangen ist, vorgestellt, das Google Web Toolkit. Mit GWT entwickelte Anwendungen zeichnen sich durch eine sehr hohe Interaktivität aus, die auf die Verwendung von AJAX basiert. Im Gegensatz zu anderen AJAX-Framework ist der GWT-verwendende Entwickler jedoch nicht gezwungen, sich mit JavaScript auseinander zu setzen. GWT erlaubt die Entwicklung von AJAX-Web-Oberflächen in reinem Java. Grundlage ist ein Compiler, der das in Java implementierte Client-GUI nach JavaScript und HTML übersetzt.

Dieser neuartige Ansatz zur Entwicklung von AJAX-basierten Web-Anwendungen kann sich als richtungsweisend etablieren. Es bleibt abzuwarten, inwieweit die Quelloffenheit des GWT diesen Vorgang untersützten kann.

Literatur

[goo] Eclipse-Plugin für GWT. googlipse.

[gwta] Google Web Toolkit. http://code.google.com/webtoolkit.

[gwtb] The Unofficial Google Wiki, Demonstrationen, Tutorials und Tools: GWT. http://google.wikia.com/wiki/Gwt.

[kit] Kitchen Sink. http://code.google.com/webtoolkit/documentation/examples/kitchensink/demo.html.