Seminar aus Informatik

# Google Web Toolkit

11.01.2009


Praktikanten:

Alexander Brosch
alexander.brosch@sbg.ac.at

Martin Mitterbauer
martin.mitterbauer@sbg.ac.at

Supervisor:

O.Univ.-Prof. Dipl.-Ing. Dr. Wolfgang PREE
wolfgang.pree@sbg.ac.at

Akademie:

Department of Computer Sciences
Universität Salzburg
Jakob-Haringer-Str. 2
5020 Salzburg
AUSTRIA

# Inhaltsverzeichnis

# Was ist das Google(TM) Web Toolkit ?

## Framework für Webanwendungen

Das Google(TM) Web Toolkit, abgeküzrt als GWT, ist ein Framework zur Entwicklung von Webanwendungen.
Eine mit dem Google(TM) Web Toolkit erstellte Webanwendung kann man in Client und Server unterteilen, wobei beide Teile komplett in der Programmiersprache Java geschrieben werden können. Der Client ist der Teil der Webapplikation, der im Webbrowser „ausgeführt“, also dargestellt wird. Der Server ist jener Teil, der zum Beispiel Daten bereitstellt, welche dann vom Client zum richtigen Zeitpunkt geladen werden können. Wir werden hier kurz auf beide Teile eingehen:

## Der Client

Alles was im Browser dargestellt werden soll, d.h. Text, Bilder, aber auch die UI-Logik (zum Beispiel was passieren soll wenn man auf einen Button drückt) wird in Java programmiert. Man kann sich dabei den Widgets bedienen, die von GWT mitgeliefert werden. Sie sind sehr ähnlich wie die Komponenten von Swing zu verwenden. Man kann also den Client ähnlich wie eine Desktop-Applikation programmieren.

### Java → JavaScript Compiler

Dieser Java-Code wird dann vom mitgelieferten Compiler nach JavaScript kompiliert, und dieser Code kann dann im Webbrowser ausgeführt werden.
Daraus ergeben sich große Vorteile von GWT:

- Erfahrene Java – Programmierer können Webanwendungen schreiben, ohne JavaScript beherrschen zu müssen.

- **Eclipse**: Bekannte Java-Entwicklungstools können verwendet werden, wie zum Beispiel die IDE Eclipse, die die Effizienz beim Programmieren deutlich erhöhen.
- **Browserunabhängigkeit**: Wer schon einmal JavaScript programmiert hat, weiß wie unterschiedlich die einzelnen Browser diesen Code interpretieren, es wäre also viel Wissen um die einzelnen Browser und ihre JavaScript-Interpreter notwendig, damit das JavaScript-Programm auf allen Browsern gleich aussieht.
  Das GWT verpackt dieses Wissen in seinen Java-zu-JavaScript-Compiler, und der Programmierer muss sich darum nicht mehr kümmern.
- **Debugging**: Mit dem GWT wird ein „Hosted Browser“ mitgeliefert, der es ermöglicht, die Webanwendung zu debuggen, d.h. die Applikation kann Schritt für Schritt ausgeführt werden, während der Programmierer die Werte der Java-Variablen betrachtet. Auch hier kann man also in der gewohnten Java-Welt bleiben.
- **JUnit**: Das Framework, das automatisiertes Testen von kleinen Codeeinheiten ermöglicht, ist bereits in GWT integriert.
- **Server-Entlastung**: Da jegliche Interaktion mit dem Benutzer im Browser (Client) verarbeitet wird, wird die Belastung des Webservers sowie des Netzwerkes verringert. Eine Kommunikation mit dem Webserver findet nur mehr selten statt (siehe nächstes Kapitel).

Eines sei noch zu erwähnen: Da alles an Java Code kompiliert wird, ist es nicht möglich, den gesamten Sprachumfang von Java zu verwenden, es muss nämlich zu jeder Methode ein Pendant in der JavaScript-Welt geben. Es steht aber ein großes Subset des Packages java.lang zur Verfügung.

### Native JavaScript

Natives JavaScript kann aber trotzdem auch in einer GWT-Applikation verwendet werden. Zum Beispiel können schon fertige JavaScript-Programme über *JSNI* (Javascript Native Interface) in die GWT-Applikation einbauen.

**Code: Beispiel für JSNI** [1]

```
public static native void alert(String msg) /*-{
   $wnd.alert(msg);
 }-*/;
```

JSNI kann als „web equivalent of inline assembly code“[2] angesehen werden.

## Der Server

Im Client, also im Webbrowser, läuft also ein gewisses Programm ab. Dieses Programm behandelt alle Benutzerinteraktionen. Manchmal ist es aber notwendig mit einem Server zu kommunizieren, sei es um Daten von einer Datenbank zu lesen oder um einen Warenkorb zu aktualisieren. Für die Serverkommunikation verwendet GWT AJAX („**A**synchronous **J**avaScript **a**nd **X**ML“).
Zum besseren Verständnis folgt hier eine kurze Erklärung des Konzepts AJAX:

### AJAX

„... bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Server und dem Browser, das es ermöglicht, innerhalb einer HTML-Seite eine HTTP-Anfrage durchzuführen, ohne die Seite komplett neu laden zu müssen. Das eigentliche Novum besteht in der Tatsache, dass nur gewisse Teile einer HTML-Seite oder auch reine Nutzdaten sukzessiv bei Bedarf nachgeladen werden, womit Ajax eine Schlüsseltechnik zur Realisierung des Web 2.0 darstellt.“[3]

Das GWT bietet dem Client nun 2 Möglichkeiten an, um mit dem Server zu kommunizieren. Beide arbeiten mit dem AJAX-Konzept.

### Serverkommunikation durch HTTP-Calls

Diese – einfachere – Möglichkeit der Serverkommunikation läuft so ab:

1. Der Client sendet einen HTTP Request, und fährt mit der Programmausführung fort.
2. Der Server verarbeitet die Anfrage und sendet eine Antwort.
3. Der Client verarbeitet die Antwort.

Man beachte, dass die Kommunikation asynchron abläuft, d.h. der Browser fährt nach dem Absenden des HTTP Requests mit der Programmausführung fort, und kann somit nicht durch Netzwerkprobleme einfrieren. Wenn die Antwort des Servers schließlich beim Client ankommt, wird eine Callback-Methode ausgeführt, die beim Absenden des HTTP-Calls mit angegeben wurde.

Die Serverseitige Implementierung kann in diesem Fall in einer beliebigen Sprache stattfinden, sei es mithilfe von PHP, Perl, Python oder Java Servlets.

### Serverkommunikation durch GWT-RPC

Eine weitere Möglichkeit der Serverkommunikation ist die der Remote Procedure Calls. GWT liefert bereits eine Schnittstelle für diese Kommunikation mit. Die Technik basiert auf der der Java Servlets, der Server muss also hier in Java programmiert werden. Ein klarer Vorteil dieser Technik stellt die Möglichkeit des Austauschs von Java-Objekten zwischen Client und Server dar.

Folgendes Klassen-Diagramm stellt die Zusammenhänge zwischen Client und Server in Bezug auf GWT-RPC dar.

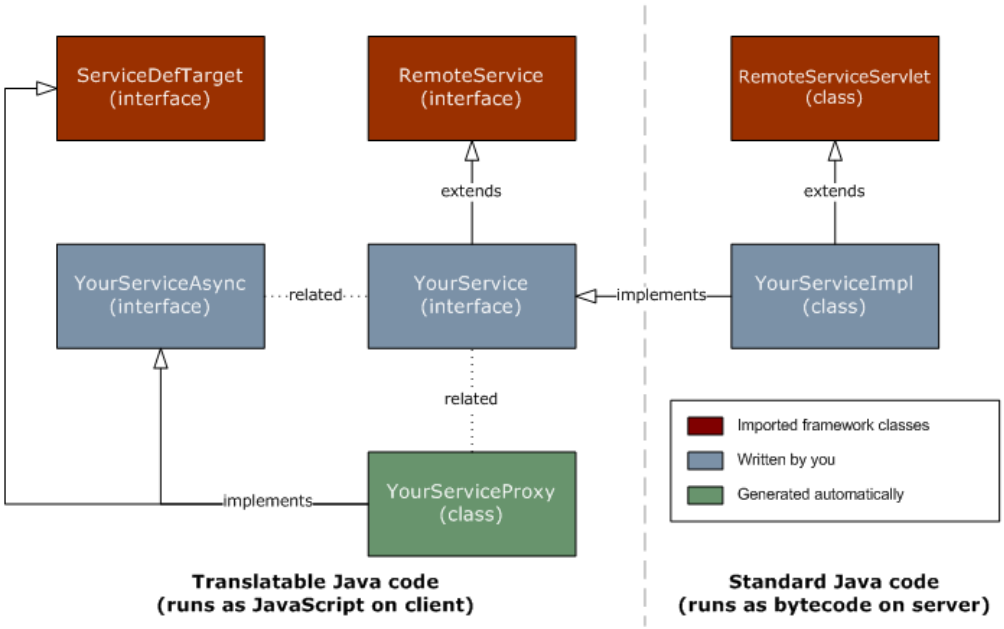



**Abbildung 1: Das GWT-RPC Klassendiagramm** [4]

Die beiden Interfaces, die für eine GWT-RPC Kommunikation am Client verfügbar sein müssen, könnten zum Beispiel so aussehen:

**Code: GWT-RPC Interface Definition** [5]

```
public interface MyEmailService extends RemoteService {
  void emptyMyInbox(String username, String password);
}


public interface MyEmailServiceAsync {
  void emptyMyInbox(String username, String password,
      AsyncCallback<Void> callback);
}
```

Die folgende Methode verwendet dieses soeben erzeugte Interface und löst dadurch einen Server Call aus:

**Code: GWT-RPC Server Call** [6]

```
public void menuCommandEmptyInbox() {
        // (1) Create the client proxy. Note that although you are creating the
        // service interface proper, you cast the result to the asynchronous
        // version of the interface. The cast is always safe because the
        // generated proxy implements the asynchronous interface automatically.
        //
        MyEmailServiceAsync emailService = (MyEmailServiceAsync)
          GWT.create(MyEmailService.class);

        // (2) Create an asynchronous callback to handle the result.
        //
        AsyncCallback callback = new AsyncCallback() {
            public void onSuccess(Void result) {
              // do some UI stuff to show success
            }

            public void onFailure(Throwable caught) {
              // do some UI stuff to show failure
            }
          };

          // (3) Make the call. Control flow will continue immediately and later
          // 'callback' will be invoked when the RPC completes.
          //
          emailService.emptyMyInbox(fUsername, fPassword, callback);
        }
```

## Vergleichbare Projekte

Es existieren zahlreiche Projekte, die ebenfalls einen Crosscompiler-Ansatz verfolgen.

- **XML11:** Das Projekt der San Francisco State University bietet ebenfalls einen Java-zu-JavaScript-Compiler, verwendet allerdings direkt die Widgets von Swing anstatt der proprietären Google-Widgets. Ein weiterer Unterschied zu GWT ist, dass vom Java Byte Code, anstatt vom Java Source Code nach JavaScript kompiliert wird.
- **WebCream** (Firma CreamTec)
- **WebOnSwing** (OpenSource)
- **RAP (Rich Ajax Platform):** Ein Teil des Eclipse Frameworks. [7]

# Kapitel Installation und Generierung eines GWT Projektes

## Installation von Google Web Toolkit

Vor der Installation von Google Web Toolkit solle überprüft werden, ob eine Java SDK Version 1.5 oder höher installiert ist.

Die Installation von Google Web Toolkit verläuft folgendermaßen:
Zuerst müssen die erforderlichen Daten der GWT-Distribution von der http://code.google.com herunter geladen werden. Diese Daten müssen anschließend in einen beliebigen Ordner entpackt werden. Diese einfachen Schritte sind an sich schon die ganze Installation. Google Web Toolkit benötigt keinen eigenen Installer, alle Dateien die benötigt werden, befinden sich in einem Ordner.

### Eintrag in der Pfadvariable

Zur einfacheren Handhabung bei der Generierung eines neuen Google Web Toolkit Projekts, ist es weiters möglich den Installationsordner in die Pfadvariable einzutragen.
Durch diesen Eintrag ist es in einem Kommando Zeilen Orientierten Tool nicht mehr nötig den vollständigen Pfad zum Installationsordner anzugeben.

Unter Windows sind folgende Einstellungen vorzunehmen:

1. Öffnen Sie die Systemsteuerung.
2. Öffnen Sie die Dialogbox „Systemeigenschaften“ mit einem Klick auf System.

3. Klicken Sie in dieser Dialogbox auf den Button „Umgebungsvariablen“.
4. In der Dialogbox „Umgebungsvariablen“ navigieren Sie zu den Benutzervariablen und dort zur „PATH“ Variable.
5. Erweitern Sie Ihre „PATH“ Variable mit einer absoluten Pfadangabe ihres Installationsordners.

Unter Mac oder Linux sind folgende Einstellungen vorzunehmen:

1. Editieren Sie das File namens .profile oder bash_profile in Ihrem „Home“ Verzeichnis.
2. Falls ihr Installationsordner den Pfad „/home/user/gwt-linux-1.5.3/“ aufweist, muss die Datei folgendermaßen angepasst werden:
   PATH=$PATH:/home/user/gwt-linux-1.5.3/
   export PATH.
3. Diesen Änderungen greifen erst nachdem Sie sich an Ihrem System mit Ihrem Account ab und neu angemeldet haben.

# Generierung eines Google Web Toolkit Projekts

Zu Beginn sollte ein Ordner erstellt werden, in dem das Projekt erzeugt wird. In unserem Fall heißt der Ordner „Stackpanel“.
Zum Erstellen des Projektes werden die mitgelieferten „Commandline Tools“ verwendet. Diese Tools (Utilities) erstellen die nötige Unterordnerstruktur und Files, die benötigt werden.

## Erstellung eines Eclipse Projekts mittels projectCreator

Das erste Tool, das zum Erzeugen eines Projekts benötigt wird, ist der projectCreator. Der projectCreator erzeugt ein Shell Eclipse Projekt.
Dazu muss ein Shell geöffnet werden in der man zu seinem Projektunterodner navigiert (Stackpanel), um folgenden Befehl abzusetzen:
projectCreator -eclipse Stackpanel -out Stackpanel

Nachdem der Befehl abgesetzt worden ist, sollte in dem Projektunterordner die generierte Projektunterordner-struktur vorhanden sein. Der Ordner sollte zum Speichern des Project Source Codes z.B. die Unterordner src und test sowie auch einige Eclipse Files (.projekt, .classpath) beinhalten.

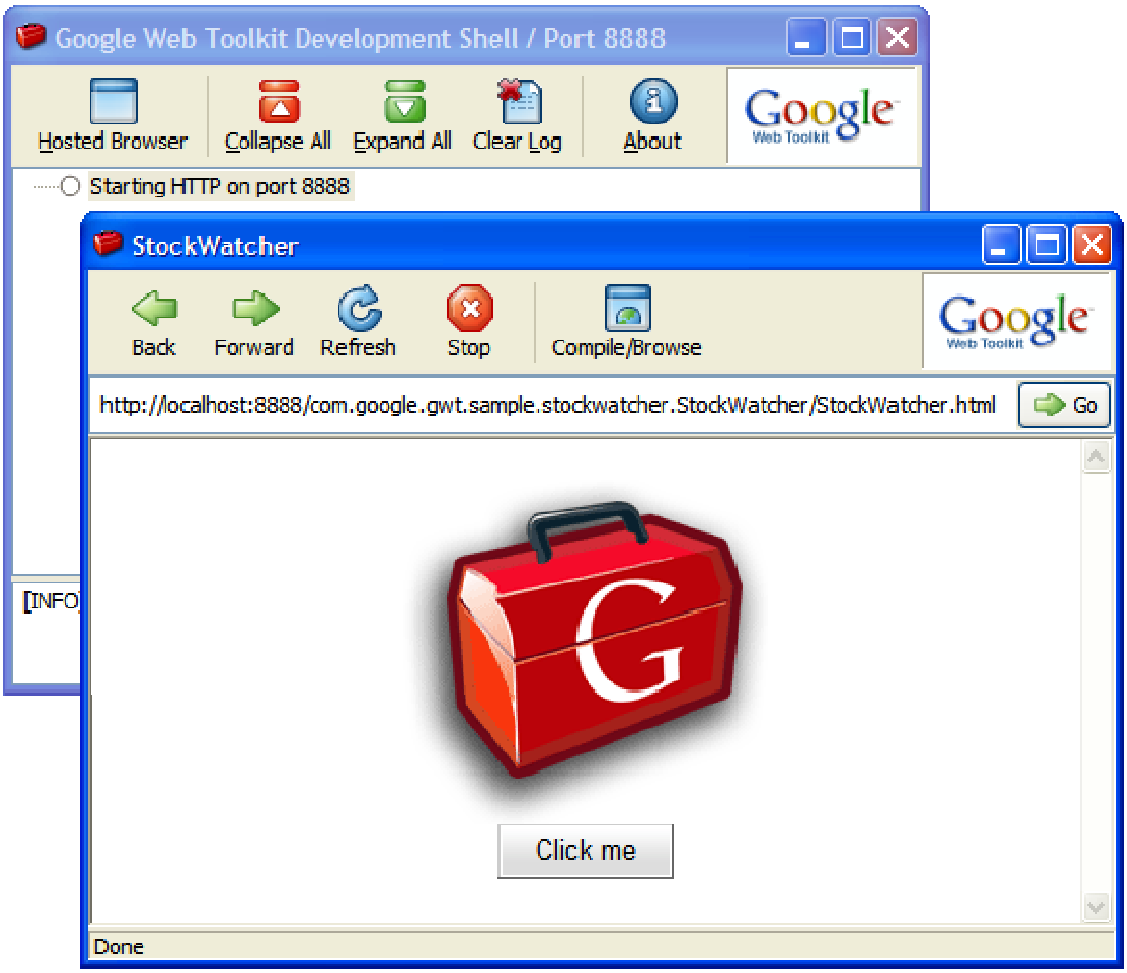
## Erstellung eines GWT Projekts mittels applicationCreator

Im nächsten Schritt wird das Tool applicationCreator verwendet. Mit diesem Tool wird ein minimales, aber funktionierendes GWT Projekt erzeugt.
Für unser Stackpanel Projekt geben wir der Main Class folgenden Namen:

com.google.gwt.sample.stackpanel.client.StackWatcher

Der applicationCreator kann auch “Eclipse launch config files” erstellen, mit denen der „host mode“ einfach debugt werden kann. Navigieren Sie hierfür in einer Shell wieder in Ihren Projektunterordner und führen Sie folgenden Befehl aus:
applicationCreator –eclipse StockWatcher –out StockWatcher com.google.gwt.sample.stockwatcher.client.StockWatcher

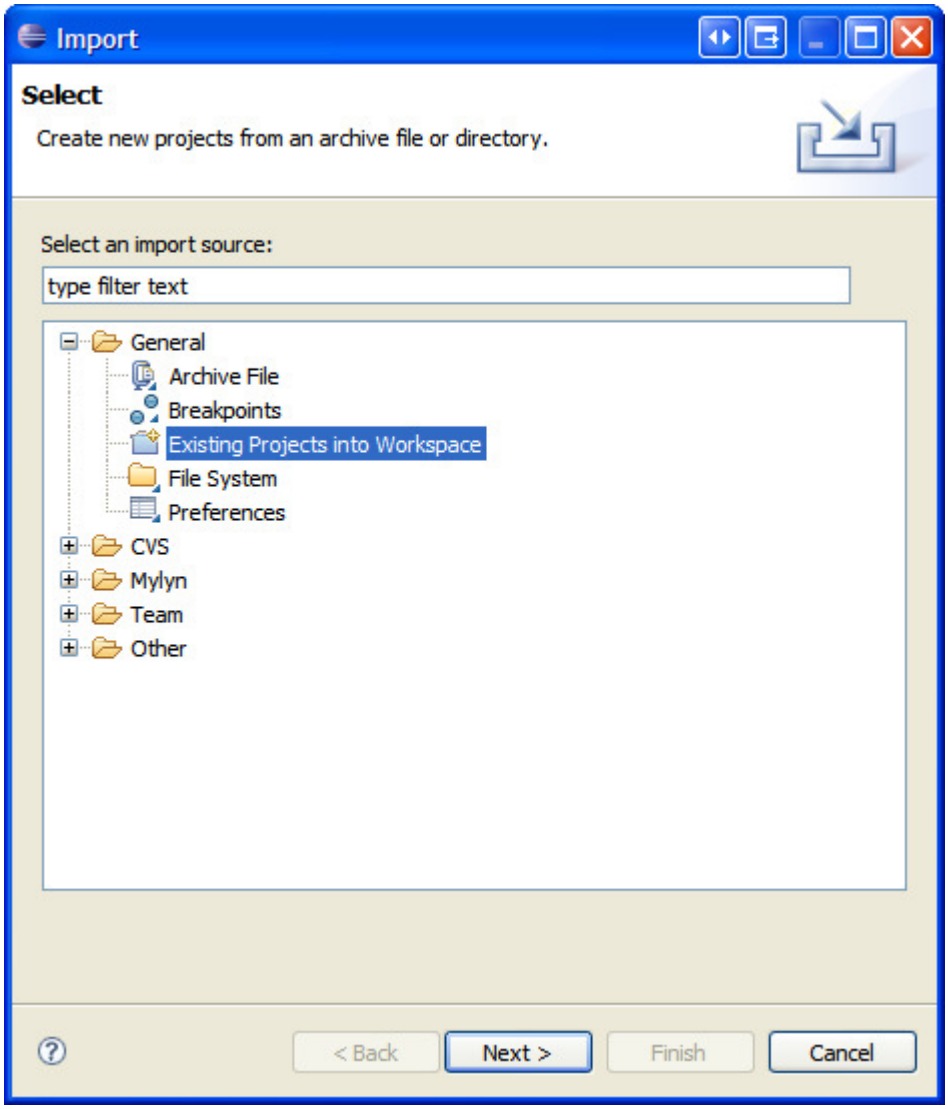
Nachdem der Befehl ausgeführt wurde, befinden sich im Projektunterordner zwei Scripts. In unserem Fall sind es die Files „Stackpanel-compile.cmd“ und „Stackpanel-shell.cmd“.
Wenn das Sackpanel-compile.cmd aufgerufen wird, wird das Projekt im sozusagenden Hostmode gestartet. Es wird somit der GWT Interne Browser gestartet, in dem das Projekt in der Java Virtual Machine läuft.




Mit diesem Browser ist es möglich im Projekt mittels Eclipse etwas zu ändern und durch einen einfachen Refresh im GWT Browser die Veränderungen zu testen.
Hiezu muss betont werden, dass im Host Mode das GWT Projekt nicht in Java Script Kompiliert wird.

## Importieren eines GWT Projekts in Eclipse

Nach dem öffnen der Eclipse Entwicklungsumgebung klicken Sie bitte auf die Menüstruktur File -> Import. Es sollte sich folgende Dialogbox öffnen:




Wählen Sie in dieser Dialogbox „Existing Projects into Workspace“ aus. In der darauf folgenden Dialogbox wählen Sie Ihr Projekt aus dem Projektunterordner aus und durch einen Klick auf „Finish“ wird das GWT Projekt in den Eclipse Workspace geladen.

# Aufbau eins GWT Projekts


## Module

Im Unterordner Stackpanel\src\de\ace\gwt\app\stackpanel befindet sich das GWT Projekt XML File, mit welchem man die Konfigurationseinstellungen vornehmen kann.
Das neu erstellte XML File enthält <inherits> Tags, <entry-point> Tags und <stylesheet> Tags. Die <inherits> Tags definieren die Abhängigkeiten des GWT Projektes, das könnte „build-in module“ von GWT sein oder selbst erstellte Module. In einem neu erstellten Projekt sind die Module com.google.gwt.user.User und com.google.get.user.theme.standard.Standard standardmäßig hinterlegt. Diese Module enthalten die „core GWT libraries“ und die jeweiligen Styles der Widgets, die mittels CSS hinterlegt werden können.

Der <entry-point> beschreibt die Mainklasse, welche beim Start des Projektes angestoßen wird. Diese Klasse muss das EntryPoint Interface beinhalten, welche eine Methode namens „onModuleLoad()“ beschreibt. In der „onModuleLoad()“ Methode werden alle Komponenten des GWT Projektes initialisiert.

Der <stylesheet> Tag beschreibt, welche .css Datei inkludiert werden soll. In unserem Fall ist es die Stackpanel.css, welche im Ordner „public“ hinterlegt ist.

### Java Source Code

Im Unterordner Stackpanel\bin\de\ace\gwt\app\stackpanel\client befinden sich die Source Files (Stackpanel.class), welche bei der Veröffentlichung in java Files kompiliert werden. In diesem File befindet sich die Grundfunktionalität des GWT Projektes. Im Grunde beinhaltet der „client“ Ordner alle alle Source Files und Subpackages.

### Static Resources

Wir wollen nun den Unterordner src\de\ace\gwt\app\stackpanel\public betrachten. Dieses „public“ Ordner ist der Behälter, in dem sich alle statischen Ressourcen befinden, die in der GWT Applikation inkludiert werden. Dies können CSS Files oder Images sein. Nach Fertigstellung des Projekts muss das Projekt kompiliert werden, hierbei werden alle Dateien die sich im „public“ Ordner befinden vom GWT Compiler in den Output Ordner kopiert. Die dabei generierte Java Script Datei wird ebenfalls in den Output Ordner kopiert.

## HTML Host Page

Die Datei Stackpanel.html ist unsere “host page”. Eine solche „host page“ ist der Container einer GWT Applikation. Diese Datei ist eine gewöhnliche HTML Datei, welche im <script> Tag auf unsere Applikation verweist. Der Name unserer Datei ist der vollständige Name unseres Projektes gefolgt von .nocache.js:

```
<script language="javascript"
src="de.ace.gwt.app.stackpanel.Stackpanel.nocache.js">
</script>
```

## GWT Projekt in eine bestehende Webseite inkludieren

In unserem Fall gehen wir davon aus, dass wir eine PHP Webseite erstellen wollen, die clientseitiges Verhalten mittels einem GWT Projekt zur Verfügung stellt.
Dies kann mit folgendem <script> verwirklicht werden:

```
<html><head>
<title>Wrapper PHP for StockWatcher</title>
<script language='javascript'
src=de.ace.gwt.app.stackpanel.Stackpanel.nocache.js'>
</script> </head> <body>
<?php echo 'PHP version: ' . phpversion();?> </body></html>
```

Durch diesen einen <script> Tag wird diese PHP Webseite zu einer „host page“ für die GWT Applikation. GWT selbst muss nicht explizit PHP unterstützen. Somit ist die clientseitig verarbeitete GWT Applikation unabhängig von dynamisch generiertem PHP Code. Sie ist grundsätzlich unabhängig von jeglichen „Server Side Frameworks“.

# Quellen


Robert Hanson, Adam Tacy: **GWT im Einsatz: Ajax-Anwendungen entwickeln mit dem Google Web Toolkit**
*(Hanser Fachbuch; 1. Auflage, 2007)*

Ryan Dewsbury: **Google Web Toolkit Applications**
*(Prentice Hall, Auflage 2007)*

Arno Puder: **A cross-language framework for developing AJAX applications**
(ACM International Conference Proceeding Series; Vol. 272)
*http://portal.acm.org/citation.cfm?id=1294325.1294340&coll=ACM&dl=ACM&CFID=7124962&CFTOKEN=46758090*

Wikipedia: **Google Web Toolkit**
*http://de.wikipedia.org/wiki/Google_Web_Toolkit*


---

[1] *http://code.google.com/intl/de-DE/docreader/#p=google-web-toolkit-doc-1-5&s=google-web-toolkit-doc-1-5&t=DevGuideJavaScriptFromJava*

[2] *http://code.google.com/intl/de-DE/docreader/#p=google-web-toolkit-doc-1-5&s=google-web-toolkit-doc-1-5&t=DevGuideJavaScriptNativeInterface*

[3] *http://de.wikipedia.org/wiki/Ajax_(Programmierung)*

[4] *http://google-web-toolkit-doc-1-5.googlecode.com/svn/wiki/AnatomyOfServices.gif*

[5] *http://code.google.com/intl/de-DE/docreader/#p=google-web-toolkit-doc-1-5&s=google-web-toolkit-doc-1-5&t=DevGuideMakingACall*

http://code.google.com/docreader/#p=google-web-toolkit-doc-1-5&s=google-web-toolkit-doc-1-5&t=GettingStartedCreateProject

[6] *http://code.google.com/intl/de-DE/docreader/#p=google-web-toolkit-doc-1-5&s=google-web-toolkit-doc-1-5&t=DevGuideMakingACall*

[7] Arno Puder: **A cross-language framework for developing AJAX applications**
(ACM International Conference Proceeding Series; Vol. 272)
*http://portal.acm.org/citation.cfm?id=1294325.1294340&coll=ACM&dl=ACM&CFID=7124962&CFTOKEN=46758090*