

Ein "Walkthrough"

JavaFX und die Google API (Blogger & Calendar)

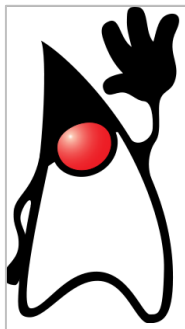
Projekt: gConnect v0.01 / 2009-01-28

Zielsetzung: Google Services vom Desktop aus nutzen

 Google Blogger Data API

 Google Calendar Data API

 **mittels JavaFX**
als Rich Internet Application (Desktop Client)



Arbeitsumgebung:

Windows XP SP3, NetBeans 6.5,
JDK 6 und JRE 6 Update 11

Aufwand: ca. 70 Stunden
für das gesamte Projekt inkl. Dokumentation

Inhalte

Die Gliederung folgt dem Entwicklungsprozess und ist daher nicht in inhaltliche Blöcke unterteilt.

Lizenz für Inhalte + Software	3
Einstieg ins Thema	4
Vorwissen	4
Auf geht's	5
Erster Kontakt: Ein Beispielprogramm	5
Google API in NetBeans 6.5 nutzen	6
Verbindung zu Google Calendar	8
Benutzer-Login (Interface + Logik)	10
Passwort-Eingabe	12
Daten aus Kalender auslesen und in JavaFX darstellen	13
Such-Anfragen an Google Calendar	14
Doppelklick in JavaFX	17
Eintrag in den Kalender (Timezone + Location)	18
Bindings in JavaFX fürs Design	22
Binding einer SwingComponent	23
Let's do it Blogger.com!	24
Elemente für das Blogger Interface	26
Anbindung des Interfaces an die Blogger API	27
Wechseln zwischen Blogger- und Calendar-Interface	31
Mehrere Blogs ansprechen	32
Feedback im Interface	36
Login-Daten speichern	37
Falschen Login abfangen	38
Problem mit dem Password-Field (JavaFX Bug)	41
Sortierung der Kalender-Einträge	42
Letzter Schritt: Deployment	43
Eigene Start-Verknüpfung vom Desktop	43
Java Web Start	44
Variante 1: Einbinden mehrerer Zertifikate	46
Variante 2: Eigenes Signieren von JARs	50
Applikation online stellen	52
Anstehende Features + Implementierungen	53

Anlage: Quellcodes

Lizenz für Inhalte + Software

Projekt: **gConnect** (Anbindung an Google Blogger + Calendar via JavaFX)

Autor: Kai Kajus Noack

Webseite: <http://media-it.blogspot.com/>

Creative Commons BY-NC-SA

<http://creativecommons.org/licenses/by-nc-sa/3.0/de/>

Es ist Ihnen gestattet:



das Werk zu vervielfältigen, es zu verbreiten + öffentlich zugänglich zu machen



Abwandlungen bzw. Bearbeitungen des Inhaltes anzufertigen

Zu den folgenden Bedingungen:



Namensnennung

Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.



Keine kommerzielle Nutzung

Dieses Werk darf nicht für kommerzielle Zwecke verwendet werden.



Weitergabe unter gleichen Bedingungen

Wenn Sie den lizenzierten Inhalt bearbeiten oder in anderer Weise umgestalten, verändern oder als Grundlage für einen anderen Inhalt verwenden, dürfen Sie den neu entstandenen Inhalt nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch oder vergleichbar sind.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter welche dieses Werk fällt, mitteilen. Am Einfachsten ist es, einen Link auf diese Seite einzubinden.
- Jede der vorgenannten Bedingungen kann aufgehoben werden, sofern Sie die Einwilligung des Rechteinhabers dazu erhalten.
- Diese Lizenz lässt die Urheberpersönlichkeitsrechte unberührt.

Einstieg ins Thema

Es ist oft ein Glücksspiel, ob man den richtigen Start in ein Thema „erwischt“ oder an einer falschen Stelle ansetzt und womöglich scheitert. Ich hatte bei dem gewählten Thema Glück, da Sun Microsystems und Google viel daran setzen, ihre Produkte für die Nutzer verfügbar zu machen und entsprechend gute Dokumentationen bereitzustellen.

Ich bin der Hoffnung, dass diese Dokumentation vielen Interessierten den Einstieg ins Thema erleichtert und die eine oder andere Frage geklärt wird, die im Entwicklungsprozess auftaucht. Es wäre natürlich schön, wenn sich aus dem Prototypen ein richtiges Projekt entwickeln würde. Doch dies liegt letztendlich nicht nur an mir, sondern auch am Mitwirken einer Community.

Genug der Worte! - Lasst uns beginnen:

Vorwissen

Als erstes ist es sinnvoll, sich mit dem Thema Google Services + API (Application Programming Interface bzw. Programmierschnittstelle) vertraut zu machen. Hierzu empfehle ich die **Videos zu Google Data** (GData ist das Austauschformat für Googles Services), die meines Erachtens gut geeignet sind, einen Überblick zu erlangen und die Zusammenhänge zu verstehen:

1. Introduction to Google Data [12:20] www.youtube.com/watch?v=ADos_xW4_J0
2. Google Developer Day London: GData APIs - Part 1 [34:06] www.youtube.com/watch?v=qF5fbkURJ98
3. Google Developer Day London: GData APIs - Part 2 [39:22] www.youtube.com/watch?v=vZ2wiSibcC4

Außerdem sollte man bereits einige Kenntnisse über die Funktionsweise von Google Data APIs besitzen. Hierzu empfiehlt sich der Einstieg über: „What are the Google Data APIs?“¹ sowie ein Blick in die dazugehörige JavaDoc² für die APIs.

Der „Developer's Guide: Protocol“³ für die Google Data API Feeds in Blogger sollte ebenfalls überflogen werden. Jedoch sei schon jetzt erwähnt, dass uns hier viel von der „harten“ Programmier-Arbeit abgenommen wird, wenn wir Googles **Java Client Library**⁴ nutzen.

Grundlegendes

Man sollte eine objektorientierte Programmiersprache (bestenfalls Java) und eine Skriptsprache (wie z. B. Actionscript) beherrschen, um den Einstieg in JavaFX zu verstehen, so wie er hier dokumentiert wurde.

¹ <http://code.google.com/apis/qdata/index.html>

² <http://code.google.com/apis/qdata/javadoc/>

³ http://code.google.com/apis/blogger/docs/2.0/developers_guide_protocol.html

⁴ <http://code.google.com/apis/qdata/client-java.html>

Auf geht's

Um etwas bauen zu können, benötigt man Werkzeuge. Diese werden mit dem **JavaFX Paket** bereitgestellt, das auf der Seite www.javafx.net zum Download bereitsteht. In diesem Download befindet sich ebenfalls NetBeans 6.5 - eine Entwicklungsumgebung (IDE) für Java- und JavaFX-Projekte. Von den drei Downloads muss der „NetBeans IDE 6.5 for JavaFX 1.0“ gewählt werden.

Der beste Einstieg, den ich gefunden hatte, war Googles Artikel „**Getting Started with the Google Data Java Client Library**“⁵. Hier wird ausführlich erklärt, welche Einstellungen und Abhängigkeiten (sogenannte „Dependencies“) unter Windows, Mac oder Linux notwendig sind. Außerdem wird eine Beispielapplikation unter Nutzung der Google Data Client Library gezeigt.

Entwicklungsumgebung NetBeans

Mit der Installation der bei javafx.net heruntergeladenen Datei „netbeans-6.5-javafx-windows.exe“ werden JavaFX und NetBeans 6.5 installiert.

Wer sich noch nicht mit NetBeans auskennt, sollte die Kurzanleitung lesen unter: http://www.mathematik.hu-berlin.de/~lamour/WR_I WR_II_Num_I/ext/NetBeans%20for%20beginners.htm

Erster Kontakt: Ein Beispielprogramm

Das erste Beispielprogramm aus dem oben genannten Artikel⁵ von Google lief ohne Probleme, auch wenn nur über einen Aufruf in der Kommandozeile von Windows, unter Nutzung von Apache ANT (einer Software für das automatische Erstellen von Applikationen). Googles Beispiel-Programm legte einen Post in einem Blog an, schrieb einen Kommentar und löschte beides danach wieder.

Um das Beispiel zu benutzen, war folgendes Vorgehen notwendig:

Zuerst muss die Datei *gdata-samples.java-1.29.0.java.zip*⁶ heruntergeladen und entpackt werden (am besten in C:\). Das Beispielprogramm ruft man dann in der Windows-Kommandozeile wie folgt auf:

```
c:\gdata\java>ant.bat -f build-samples.xml sample.blogger.run
```

Jedoch klappte es nicht auf Anhieb, da der Path nicht gesetzt war.

Für dieses Beispiel war es notwendig, dass man:

1. zuerst den Path auf das Ant-Verzeichnis (path C:\Programme\NetBeans 6.5\java2\ant\bin\) legt und
2. in der Datei **gdata\java\build-samples\build.properties** ein paar Änderungen vornimmt: bei `sample.credentials.username=` muss ein Loginname (Email) und bei `sample.credentials.password=` das entsprechende Passwort eingetragen werden.

⁵ http://code.google.com/apis/gdata/articles/java_client_lib.html

⁶ <http://gdata-java-client.googlecode.com/files/gdata-samples.java-1.29.0.java.zip>

In der nachstehenden Grafik ist die Konsolenausgabe für das funktionierende Programm zu sehen.

```

C:\>path C:\Programme\NetBeans 6.5\java2\ant\bin\
C:\>cd gdata\java
C:\gdata\java>ant.bat -f build-samples.xml sample.blogger.run
Buildfile: build-samples.xml

sample.blogger.dependencies:
template.require.service.jar:

sample.blogger.build:
[java] Compiling 1 source file to C:\gdata\java\sample\blogger\classes
[jar] Building jar: C:\gdata\java\sample\blogger\lib\BloggerClient.jar

sample.blogger.run:
[java] Benar Cinta's Blogs
[java] Benar Cinta Blog
[java]
[java] Successfully created draft post: Snorkling in Aruba
[java] Successfully created public post: Back from vacation
[java] Benar Cinta Blog
[java] Back from vacation
[java] Snorkling in Aruba
[java] This is my first Blog
[java]
[java] Benar Cinta Blog posts between 2007-04-04 and 2007-04-06
[java] Back from vacation
[java] Tue, 13 Jan 2009 18:14:44 +0000
[java] Snorkling in Aruba
[java] Tue, 13 Jan 2009 18:14:44 +0000
[java] This is my first Blog
[java] Tue, 13 Jan 2009 13:59:10 +0000
[java]
[java] Post's new title is "Swimming with the fish".
[java]
[java] Post's new title is "The party's over".
[java]
[java] Creating comment
[java] Comments on Benar Cinta Blog: The party's over
[java] Did you see any sharks?
[java] Tue, 13 Jan 2009 18:14:00 +0000
[java]
[java] Deleting comment
[java] Deleting draft post
[java] Deleting published post

BUILD SUCCESSFUL
Total time: 35 seconds
C:\gdata\java>

```

Screenshot der Ausgabe (cmd, Kommandozeile unter Windows XP)

Google API in NetBeans 6.5 nutzen

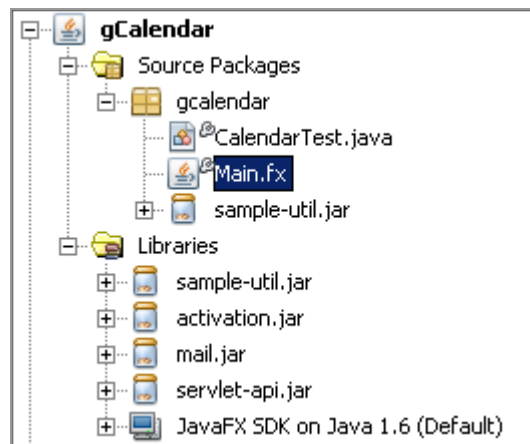
Nach dem Start von NetBeans und dem Anlegen eines neuen JavaFX-Projektes musste ich mir neben Googles Client Library auch die Libraries für die Dependencies besorgen.

Eine Übersicht über alle Downloads an notwendigen JAR-Dateien findet sich in „Getting Started“ für Windows⁷. Auf die Java- und die ANT-Installation kann verzichtet werden, da diese bereits mit NetBeans (netbeans-6.5-javafx-windows.exe) installiert wurden.

Wir benötigten: mail.jar | activation.jar | servlet.jar

Diese können in das neu erstellte Projekt unter NetBeans eingebunden werden (rechter Mausklick aufs Projekt >Libraries >Add JAR/Folder).

⁷ http://code.google.com/apis/gdata/articles/java_client_lib.html#windows



Ins Projekt eingebunden JAR-Dateien

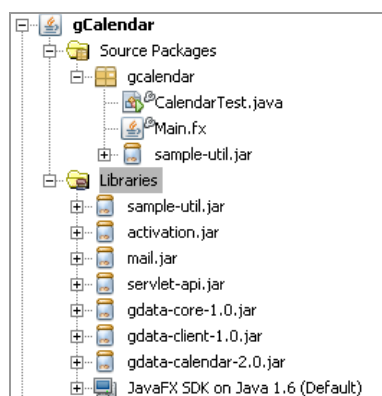
Notwendige Google Client Libraries für Blogger + Calendar

Die für unser Projekt notwendigen JAR-Dateien von Google waren:

mail.jar | activation.jar | servlet.jar // wie soeben erwähnt
 +
 gdata-blogger-2.0.jar
 gdata-calendar-2.0.jar
 gdata-client-1.0.jar
 gdata-core-1.0.jar
 sample-util.jar
 gdata-media-1.0.jar // später hinzugefügt, siehe weiter hinten im Text

Sie befinden sich in der Datei *gdata-samples.java-1.29.0.java.zip*.
 Siehe dort im Verzeichnis „java/lib“.

Für den Test-Kalender waren zuerst nur gdata-core + gdata-client + gdata-calendar zusätzlich einzubinden:



Eingebundene JARs

Verbindung zu Google Calendar

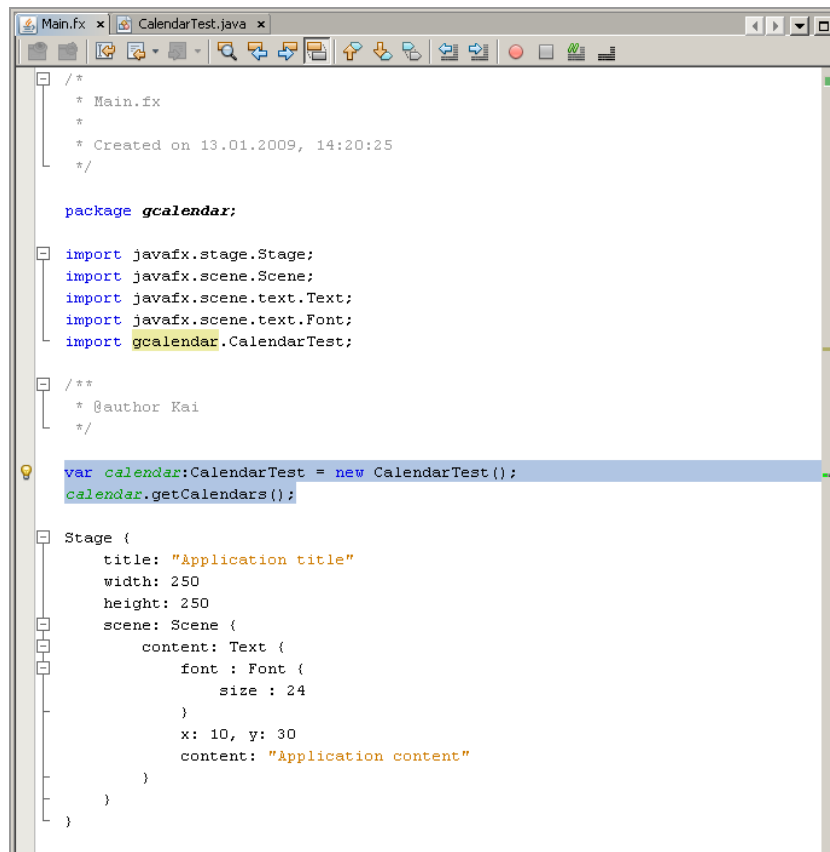
Um beginnen zu können, erstellte ich eine javaFX-Datei in unserem Projekt unter NetBeans und fügte Code hinzu, um ein Standard-Fenster zu öffnen. Darüber hinaus testete ich die Einbindung einer Java-Klasse in die JavaFX-Datei (siehe Screenshot).

 In diesem Dokument ist mit „JavaFX-Datei“ stets die „Main.fx“ gemeint.

Aufruf von Java-Klassen in JavaFX

Wie einem Java-Programmierer bereits bekannt ist, wird auch in JavaFX eine java-Datei in ein Package eingebunden. Der Unterschied besteht im Aufruf der Klasse in der JavaFX-Syntax, z.B.

```
var calendar:CalendarTest = new CalendarTest();
```



```

/*
 * Main.fx
 *
 * Created on 13.01.2009, 14:20:25
 */

package gcalendar;

import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import gcalendar.CalendarTest;

/**
 * @author Kai
 */
var calendar:CalendarTest = new CalendarTest();
calendar.getCalendars();

Stage {
    title: "Application title"
    width: 250
    height: 250
    scene: Scene {
        content: Text {
            font : Font {
                size : 24
            }
            x: 10, y: 30
            content: "Application content"
        }
    }
}

```

CalendarTest ist die aus JavaFX aufgerufene Java-Datei

Im Beispiel haben wir ein Kalender-Objekt, dessen Methode *getCalendars()* wir mit *calendar.getCalendars();* aufrufen.

Die **CalendarTest.java** enthielt Code, der aus Googles Artikel „Getting Started“⁵ entnommen wurde:

```

package gcalendar;

import com.google.gdata.client.*;
import com.google.gdata.client.calendar.*;
import com.google.gdata.data.*;
import com.google.gdata.data.acl.*;
import com.google.gdata.data.calendar.*;
import com.google.gdata.data.extensions.*;
import com.google.gdata.util.*;

import java.net.*;
import java.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;

// import sample.util.*;

public class CalendarTest {

    public void getCalendars() {
        CalendarService myService = new CalendarService("exampleCo-exampleApp-1.0");
        try {
            myService.setUserCredentials("username@gmail.com", "myPassword");
        } catch (AuthenticationException ex) {
            Logger.getLogger(CalendarTest.class.getName()).log(Level.SEVERE, null, ex);
        }

        URL feedUrl = null;
        try {
            // request the list of all calendars from the authenticated user
            feedUrl = new
                URL("http://www.google.com/calendar/feeds/default/allcalendars/full");
        } catch (MalformedURLException ex) {
            Logger.getLogger(CalendarTest.class.getName()).log(Level.SEVERE, null, ex);
        }
        CalendarFeed resultFeed = null;
        try {
            // execute GET command on the URL + put resultant feed into a tidy object
            resultFeed = myService.getFeed(feedUrl, CalendarFeed.class);
        } catch (IOException ex) {
            Logger.getLogger(CalendarTest.class.getName()).log(Level.SEVERE, null, ex);
        } catch (ServiceException ex) {
            Logger.getLogger(CalendarTest.class.getName()).log(Level.SEVERE, null, ex);
        }

        System.out.println("Your calendars:");
        System.out.println();

        // iterate through each entry and print the title
        // (stored as a TextConstruct - getPlainText for text)
        for (int i = 0; i < resultFeed.getEntries().size(); i++) {
            CalendarEntry entry = resultFeed.getEntries().get(i);
            System.out.println("\t" + entry.getTitle().getPlainText());
        }
    }
}

```

Beim Kompilieren erschien in der NetBeans-Konsole folgende Fehlermeldung:

```

13.01.2009 15:58:50 gcalendar.CalendarTest getCalendars
SCHWERWIEGEND: null
com.google.gdata.util.ServiceForbiddenException: Forbidden
You must be a calendar user to use private feeds.

at
com.google.gdata.client.http.HttpGDataRequest.handleErrorResponse(HttpGDataRequest.java
:505)

at
com.google.gdata.client.http.GoogleGDataRequest.handleErrorResponse(GoogleGDataRequest.
java:555)

(...)

```

Da hatte ich doch offensichtlich vergessen, dass der User mindestens einen Kalender besitzen muss, um auf ihn zugreifen zu können! ☺

Nachdem ich mich also mit dem Browser Firefox bei Google Calendar anmeldete und gleich zwei Kalender erstellte, startete ich die JFX-Applikation nochmals und erhielt folgende Konsolenausgabe:

```
compile:
jar:
standard-run:
Your calendars:

    benarcinta@googlemail.com
    Mein Zweiter Kalender
```

Ich konnte also auf die Kalender zugreifen und freute mich, dass der Zugriff auf die Google API funktionierte! (Hier mit Verweis auf mein „Abenteuer“ beim Entwickeln mit Adobe AIR).

Benutzer-Login (Interface + Logik)

Die Eingabe für Benutzernamen (meist die GMail-Adresse) und das Passwort sollten als nächstes in der Oberfläche unserer JavaFX-Applikation implementiert werden. Benötigt wurden zwei Textinput-Felder und ein Login-Button.

Bei der Recherche im Netz bin ich auf „Building GUI Applications With JavaFX - Tutorial Overview“⁸ gestoßen, wo sich auch Beispiele für benötigte Elemente wiederfinden ließen. Hier sei besonders auf „Lesson 4: Creating Graphical Objects“⁹ hingewiesen. Informationen und ein Beispiel für das Data-Binding in JavaFX erhält man in „Lesson 5: Applying Data Binding to UI Objects“¹⁰.

Ich nutzte das vorhanden Wissen und erstellte einen funktionierenden Button sowie die zwei Textfelder.

```
var loginButton = Rectangle {
    x: bind ourStage.width / 2 - 70
    y: 200
    width: 140
    height: 35
    arcWidth: 20
    arcHeight: 55
    stroke: Color.BLACK
    fill: LinearGradient {
        startX: 0.0,
        startY: 0.0,
        endX: 0.0,
        endY: 1.0,
        proportional: true
        stops: [
            Stop {
                offset: 0.0
                color: Color.WHITE},
            Stop {
                offset: 1.0
                color: Color.BLACK}
        ]
    }
}
```

⁸ <http://java.sun.com/javafx/1/tutorials/ui/>

⁹ <http://java.sun.com/javafx/1/tutorials/ui/graphics/index.html>

¹⁰ <http://java.sun.com/javafx/1/tutorials/ui/binding/index.html>

```

var loginButtonLabel = Text {
    fill: Color.WHITE
    font: Font {
        size: 18
    }
    x: bind ourStage.width / 2 - 30
    y: 225
    content: "Connect"
}
var usernameTextfield = SwingTextField { // password-field analog
    width: 200
    foreground: Color.BLACK
    background:Color.WHITE
    translateX: bind ourStage.width / 2 - 100
    translateY: 100
    text: "Your Google-Account (Email)"
};

```

Die vorgenannten Tutorials von Sun erwiesen sich als exzellent. Mit ihnen und der JavaFX API¹¹ war es mir möglich, einen ansprechenden Login-Screen zu erstellen. Inklusive logischer Anbindung ans Interface und Mouse-Funktionalitäten.

```

content: [
    Group {
        content: [
            loginButton,
            loginButtonLabel
        ]
        effect: Reflection {
            fraction: 0.9
            topOpacity: 0.5
            topOffset: 2.5
        }
        cursor: Cursor.HAND
        onMouseEntered: function(evt: MouseEvent):Void {
            loginButton.scaleX = 1.2;
            loginButtonLabel.underline = true;
        }
        onMouseExited: function(evt: MouseEvent):Void {
            loginButton.scaleX = 1.0;
            loginButtonLabel.underline = false;
        }
        onMouseClicked: function(evt: MouseEvent):Void {
            username = usernameTextfield.text;
            password = passwordField.getText();
            calendar.authenticate(username,password);
            //... weitere Funktionalitäten
        }
    }
]

```

Als nächstes galt es, die Daten als Strings an die Kalender-Klasse zu übergeben. Hierfür änderte ich einfach die Methode:

```
calendar.getCalendars();
```

ZU:

```
calendar.getCalendars(username, password);
```

Und fing die Daten in der getCalendars() in Java auf:

```
this.username = user;
this.password = pwd;
```

Bingo! Es klappte mit der Datenübergabe sowie der Verbindung zu Google Calendar ☺

¹¹ <http://java.sun.com/javafx/1/docs/api/>

Passwort-Eingabe

Mir fiel an dieser Stelle auf, dass das Passwort während der Eingabe als Klartext sichtbar war. Ich glaubte, das wäre schnell zu erledigen (mit `setEchoChar` wie in Java), doch weit gefehlt. Diese Funktion war nicht in JavaFX enthalten. Also musste ein Workaround über Java Swing her:

```
var password = new javax.swing.JPasswordField();
password.setColumns(10);

// und in der Scene hinzugefügt
Group {
    content: [ SwingComponent.wrap(password) ]
}
```

Jetzt bestand jedoch das Problem, dass das Passwort-Feld anstatt mit „Password“ mit „*****“ beschriftet war. Da ich dem User aber direkt das „Password“ als Text anzeigen wollte (wie man es von Webseiten her gewöhnt ist), so bestand die nächste Aufgabe darin, 1. das „Password“ beim Start anzuzeigen und 2. beim Klick auf das Feld den Text zu löschen und alle weiteren Eingaben mit „*“ anzugeben. Das war nicht so einfach, wie sich zeigte.

Folgender Code löste das Problem:

```
var password = new javax.swing.JPasswordField("Password");
// remember the char sign for the password
var theChar = password.getEchoChar();
// disable password asteriks at start up password.setEchoChar(0);

// außerdem schuf ich eine Referenz die SwingComponent,
// um auf sie zugreifen zu können + zu positionieren
var passwordComponent = SwingComponent.wrap(password);
passwordComponent.translateX = stageWidth / 2 - 100;
passwordComponent.translateY = 140;
passwordComponent.width = 200;

// und in der Scene stand nun:
Group {
    content: [ passwordComponent ]
    cursor: Cursor.TEXT
    onMouseClicked: function(evt: MouseEvent):Void {
        // delete "Password" from textfield
        password.setText("");
        // set * again for entered password chars
        password.setEchoChar(theChar);
    }
}
```

Darüber hinaus stellte ich fest, dass ich einen leeren Nutzernamen und ein leeres Passwort-Feld noch nicht abgefangen hatte. Hier ergab sich, dass JavaFX eine simple Notation für Vergleiche benutzt:

Java	JavaFX
&&	and
	or
!	not

```
if ( (username != "") and (password != "") and (password != "Password") ) { ... }
```

Hiermit war ein zufriedenstellendes Erst-Ergebnis erreicht:



Anbindung Interface mit Login-Daten zur API

Daten aus Kalender auslesen und in JavaFX darstellen

Für die Darstellung der Daten sollte nach dem Login ein neuer Screen erscheinen. Hierfür wurde eine neue Oberfläche notwendig.

Ich dachte mir, eine neue JavaFX-Datei mit einem neuen Screen wäre hilfreich. Ich wollte jeden neuen Screen in eine separate JavaFX-Datei legen, sodass bessere Übersichtlichkeit vorherrscht und die Main.FX klein bleibt, und nur die Screens beinhaltet.

Beim Erstellen der Screen-Dateien stieß ich aber auf das Problem, das JavaFX-Klassen keine Konstruktoren kennen:

Since JavaFX classes are initialized with object literal expressions, there are no constructors in JavaFX classes. Instead, two specially named blocks of code, the `init` block and the `postinit` block, may be declared in a JavaFX class that fulfil some of the functionalities of constructors in Java.

<http://www.ocjweb.com/jnb/jnbDec2008.html>

Da ich nach mehrmaligem Probieren nicht so einfach auf ein vernünftiges Ergebnis stieß, entschloss ich mich erst einmal für jedes Interface jeweils eine *Group* als Referenz anzugeben. So konnte ich über „visible“ entscheiden, welche Group mit welchem Inhalt angezeigt werden sollte.

Feedback zum Status der Authentifizierung

Anschließend testete ich die Applikation und stellte fest, dass die Authentifizierung bei Google (in der java-Datei) ausgegliedert werden sollte sowie eine Status-Rückmeldung benötigt wird (erfolgreicher Login). Ansonsten wusste weder der Nutzer, noch der Programmierer, ob korrekt eingeloggt wurde.

Die Authentifizierung wurde gestartet mit dem `CalendarService`:

```
myService.setUserCredentials(user, pwd);
```

Hier galt es, den Status abzufangen. Doch leider war kein „getStatus“ (wie z.B. bei dem `CalendarEventEntry`) als Methode enthalten.

Da `CalendarService` von zwei Klassen erbt, musste ich in der JavaDoc unter *com.google.gdata.client.Service* nachschauen, um näher an die Lösung heranzukommen. Jedoch ohne Erfolg. Eine direkte Status-Abfrage war auch hier nicht zu finden.

Schließlich nutzte ich die simpelste Lösung, die mir einfiel: Ein try-catch-Block, der nur bei fehlgeschlagener Verbindung eine Exception wirft. In der Java-Datei:

```
void authenticate(String user, String pwd) {
    this.username = user;
    this.password = pwd;
    try {
        myService.setUserCredentials(user, pwd);
        System.out.println("CONNECTED");
    } catch (AuthenticationException ex) {
        Logger.getLogger(CalendarTest.class.getName()).log(Level.SEVERE, null, ex);
        System.out.println("CANNOT ESTABLISH CONNECTION!");
    }
}
```

Dies funktionierte wie gewünscht.

Nun musste ich noch die Calendar-Group gestalten, die alle grafischen Elemente für die Kommunikation mit dem Kalender enthalten sollte.

Such-Anfragen an Google Calendar

Dann galt es, Kalenderabfragen zu implementieren. Ein Button ließ sich über die Bibliothek „import `javafx.ext.swing.SwingButton;`“ einbinden und einfügen:

```
var buttonSearch = SwingButton{text: "Search"};
```

Ein Textfield namens „searchTextfield“ für die Eingabe des Suchwortes habe ich zusätzlich erstellt und auf der Bühne positioniert. Dazu kam noch eine `SwingList`, die die Ergebnisse unserer Suchanfrage (Methode *doquery()*) darstellen sollte.

Um **mehrere Ergebnisse abzufangen**, wollte ich auf ein String-Array zurückgreifen.

Die Java-Datei warf ein String-Array (String[] results;) zurück, wenn man die doquery()-Methode aufrief. Es hieß nun, das String-Array in JavaFx anzunehmen. Jedoch stellte sich heraus, dass JavaFX keine Arrays kennt! Sondern nur Sequenzen, siehe „Using JavaFX sequences in Java“¹².

Meine erste Variante in der FX-Datei:

```
var recentResults = calendar.doQuery(searchTerm,
    "http://www.google.com/calendar/feeds/username@gmail.com/private/full",
    username, password); // usernameTextfield.text, password.getText();
```

schien zu funktionieren, als ich jedoch auf die Elemente via recentResults[0] zugreifen wollte, klappte es nicht. Offensichtlich können String-Arrays von Java nicht ohne Weiteres nach JavaFX kommuniziert werden. Ein Lösungsversuch bestand in Reflections, ich benutzte die javafx.reflect **FXJavaArrayType**, doch diese Variante funktionierte ebenfalls nicht.

Anschließend entschloss ich mich, das Array in Java zu behalten und stattdessen über eine zweite Java-Methode „getQueryResults“ (die in JavaFX per for-Schleife aufgerufen wird), die einzelnen Werte als Strings auszulesen und in einem String-Array in JavaFX zu speichern. Das klappte reibungslos:

In Java:

```
public String getQueryResults(int elementAt) {
    // iterate over all results and save them in string array, maximal 10 entries
    resultString =
        myResultsFeed.getEntries().get(elementAt).getTitle().getPlainText();
    return resultString;
}
```

In Java FX (SearchButton):

```
onMouseClicked: function(evt: MouseEvent):Void {
    // clear SwingList
    resultsList.items = null;

    // avoid empty search
    if (searchTextfield.text != "") {
        var searchTerm = searchTextfield.text;
        entriesFound = calendar.doQuery(searchTerm,
            "http://www.google.com/calendar/feeds/username@gmail.com/private/full",
            "username@gmail.com", "myPassword"); // usernameTextfield.text,
            password.getText());

        if ( entriesFound > 0) {
            for (i in [0..(entriesFound-1)]) {
                insert calendar.getQueryResults(i) into recentResults;
                // insert into list
                var item = SwingListItem {
                    text: recentResults[i] // listing of events
                };
                insert item into resultsList.items;
                // println("# array-size is {sizeof recentResults}");
            }
        }
        else {
            // insert into list
            var item = SwingListItem {
                text: "Sorry, no Entries found!" // here listing of events
            };
            insert item into resultsList.items;
        }
    }
}
```

// Hinweis:

// Das Problem mit dem String-Array wurde gelöst, siehe weiter unten im Dokument bei „Sortierung der Kalender-Einträge“

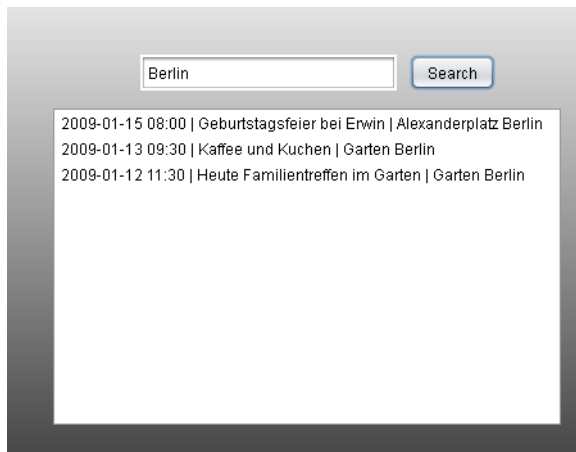
¹² http://java.sun.com/developer/technicalArticles/scripting/javafx/javafx_and_java/#6

Soweit so gut. Jetzt galt es noch Zeit und Ort für jeden Termin anzugeben.

In Java erweiterte ich hierzu die Anweisung bei der Methode „getQueryResults“:

```
resultString =
    myResultsFeed.getEntries().get(elementAt).getTimes().get(0).getStartTime().toUiString()
    + " | " +
    myResultsFeed.getEntries().get(elementAt).getTitle().getPlainText()
    + " | " +
    myResultsFeed.getEntries().get(elementAt).getLocations().get(0).getValueString().toString();
```

Nun sah das Ergebnis der Abfrage wie folgt aus:



Beim mehrfachen Testen der Applikation fand ich noch einen kleinen Fehler, und zwar dass bei mehrfachen Suchanfragen die alten Ergebnisse in der Liste stehen blieben. Dies bereinigte ich, indem ich nicht nur die „resultsList.items“ (SwingList -> SwingListItem) löschte, sondern ebenso die „recentResults“ (also das String-Array):

```
// clear SwingList
resultsList.items = null;
// clear String Array
recentResults = null;
```

Danach fügte ich dem Interface noch einen Button „Show My Events“ hinzu, der alle aktuellen Termine zurückgibt, indem an doQuery(“”) ein leerer String übergeben wird, sodass Google alle Einträge ausliest und an uns zurückgibt.

Doppelklick in JavaFX

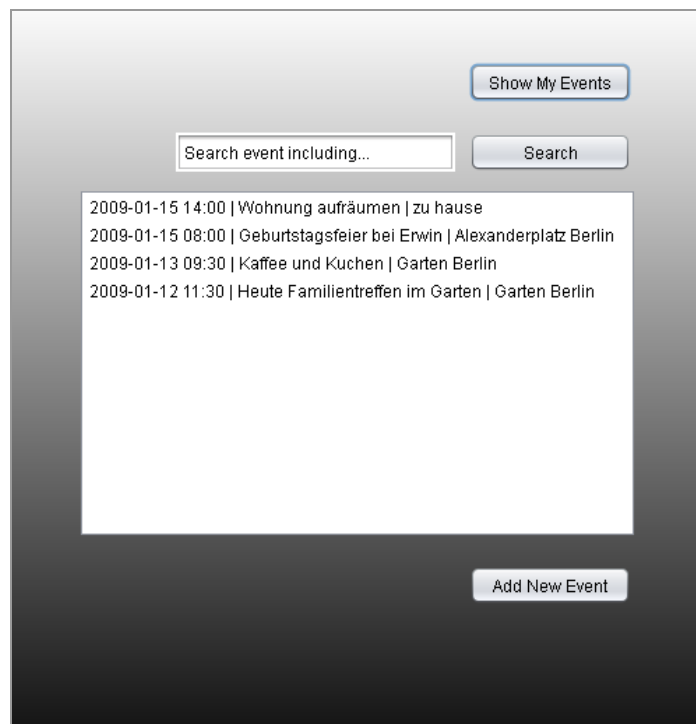
Eine Editier-Möglichkeit wollte ich zuerst über einen „Edit“-Button implementieren, entschied mich dann jedoch für einen Doppelklick per Maus auf einen Eintrag in der Liste. Der Doppelklick wurde wie nachfolgend beschrieben implementiert, die echte Editier-Funktionalität des Eintrags ist aber in dieser Version noch nicht verwirklicht.

Die SwingList enthielt eine Methode „onMouseClicked“, mit der sich ein Event abfeuern ließ. Ein Doppelklick kann derzeit noch nicht direkt in JavaFX angesprochen werden, ein Workaround hierfür ist das Zählen des Klicks auf ein Element mit folgendem Code:

```
onMouseClicked: function(e: MouseEvent):Void {
    if(e.clickCount >= 2) {
        resultsList.selectedItem.text = "editable";
    }
}
```

Beim Recherchieren bin ich auf einen weiteren Workaround mittels Timeline gestoßen¹³. Den ich aber nicht implementierte.

Nachstehend der aktuelle Screenshot, der auch die geänderten Buttons beinhaltet (alle width-Parameter der Buttons sind an den breitesten Button, den show-my-events-Button geknüpft (via „width: **bind** buttonShowEvents.width“):



Interface der bis hier entwickelten Kalender-Applikation

Ein Hinweis für das Editieren per Doppelklick fügte ich ebenfalls nachträglich als Text-Element hinzu.

¹³ <http://www.lexique-du-net.com/blog/index.php?post/2008/12/14/how-do-i-detect-double-click-in-JavaFX>

Eintrag in den Kalender (Timezone + Location)

Als nächstes galt es, Felder für die Daten eines Termins zu erstellen. Hierzu genügten einfache `SwingTextFields` (`fieldEventTitle`, `fieldEventContent`, `fieldEventStartDate`, `fieldEventStartTime`, `fieldEventEndDate`, `fieldEventEndTime`), die ausgelesen werden konnten.

Die Logik für die Daten stellte sich als schwierig dar: Zunächst musste ich das richtige Format an Google Calendar übergeben. Ich stieß auf einen Beispiel-Code, bei dem ein eigener Calendar inklusive Date-Parser verwendet wird.¹⁴

Jedoch dachte ich mir, es müsste doch wesentlich leichter gehen. Nach dem Prinzip der „Einfachheit“ baute ich mir die Strings mit wenigen Zeilen selbst zusammen:

```
// date format is ("yyyy-MM-ddTHH:mm:ss-SSSZ")
// e.g. "2009-01-15T17:00:00-08:00"
var offsetTimezone = "+01:00"; // Germany .344Z
var eventTitle = "Stagnight!";
var eventContent = "with all the folks";

var eventStartDate = "2009-01-15";
var eventStartTime = "15:00";
var eventStart = "{eventStartDate}T{eventStartTime}:00{offsetTimezone}";

var eventEndDate = "2009-01-15";
var eventEndTime = "18:00";
var eventEnd = "{eventEndDate}T{eventEndTime}:00{offsetTimezone}";

calendar.setEvent("http://www.google.com/calendar/feeds/{username}/private/full",
eventTitle, eventContent, eventStart, eventEnd);
```

* Am obigen Beispiel ist auch das erste Mal die Einbindung einer Variable in einen String zu erkennen, der im Gegensatz zu Java hier so funktioniert: `var meinString = "Beispiel { stringVar }";`

Leider musste ich, nachdem ich mit der Applikation einen Eintrag in Google Calendar anlegte, feststellen, dass meine Zeiten im Kalender um einige Stunden verschoben waren. Ein Ändern des Wertes in Richtung der deutschen Zeit (bei `offsetTimezone`) brachte jedoch auch nichts, sondern warf letztlich die Exception bei +8 Stunden offset:

```
gcalendar.Calendar setEvent
SCHWERWIEGEND: null
com.google.gdata.util.ServiceException: Method Not Allowed
```

Schließlich forschte ich im Internet und fand heraus, dass unser Format als `xs:date` string geparsed wird → Coordinated Universal Time (UTC, auch "Greenwich Mean Time"). Spezifikationen des W3C sind zu finden unter: <http://www.w3.org/TR/xmlschema-2/#dateTime>

Nach einer langen Recherche bin ich endlich darauf gestoßen, weshalb mein Offset für die Timezone von "+01:00" für Deutschland nicht funktionierte...

Ich hatte beim Einrichten meines Google Calendar unklugerweise eine *GMT+7 (Indonesien)* angegeben. Dieser Fehler hatte mich viel Zeit gekostet ☹ - Merken wir uns also den Workaround für später:

Benutze `google.gdata.calendar.TimeZoneProperty`, um die korrekte Timezone auszulesen!

Darüber hinaus muss der User darauf achten, grundsätzlich bei seinen Einstellungen in Google für seinen Haupt-Account ebenfalls die richtige Zeitzone einzustellen!

¹⁴ http://learnjavafx.typepad.com/weblog/multitier_javafx_script_apps/

Nun durfte ich die TextFields im Interface an meine `setEvent()` anbinden, um den neuen Termin an Google abzusenden. Meine `onMouseClicked()` auf den `addEventButton` sah schließlich so aus:

```
onMouseClicked: function(evt: MouseEvent):Void {

    // http://code.google.com/apis/calendar/docs/2.0/reference.html#Calendar_feeds
    // writes into Calendar

    // date format is ("yyyy-MM-ddTHH:mm:ss-SSSZ")
    // e.g. "2009-01-15T17:00:00-08:00"
    var offsetTimezone = "+01:00"; // Germany .344Z
    var eventTitle = fieldEventTitle.text; // "Stagnight!";
    var eventContent = fieldEventContent.text; // "with all the folks"

    var eventStartDate = fieldEventStartDate.text; // "2009-01-15"
    var eventStartTime = fieldEventStartTime.text; // "15:00"
    var eventStart = "{eventStartDate}T{eventStartTime}:00{offsetTimezone}";

    var eventEndDate = fieldEventEndDate.text; // "2009-01-15"
    var eventEndTime = fieldEventEndTime.text; // "18:00"
    var eventEnd = "{eventEndDate}T{eventEndTime}:00{offsetTimezone}";

    calendar.setEvent("http://www.google.com/calendar/feeds/{username}/private/full",
        eventTitle, eventContent, eventStart, eventEnd);

}
```

Das erweiterte Interface, dessen Anbindung an Google nunmehr funktionierte, sah nun wie folgt aus:



Ich testete das Interface weiter und erhielt eine Exception, wenn ich einen Termin hinzugefügt hatte (add new event) und dann „Show My Events“ betätigte, um den Feed mit allen Terminen abzurufen:

```
java: FEEDS WITH FOUND: 4
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
```

Da der Code vorher auch fehlerfrei lief, schaute ich mir den Termin im Browser in Google Calendar an und musste feststellen, dass der Ort (Location) nicht übergeben wurde.

Wo griffen wir also auf den Feed, speziell den Ort zu? In der Java-Datei wurde ich fündig:

```
resultString =
    myResultsFeed.getEntries().get(elementAt).getTimes().get(0).getStartTime().toUiString()
    + " | " +
    myResultsFeed.getEntries().get(elementAt).getTitle().getPlainText() + " | " +
    myResultsFeed.getEntries().get(elementAt).getLocations().get(0).getValueString().toString();
```

Damit wir auch ohne Ort in Ruhe weiterarbeiten können, habe ich diesen Teil in einen try-catch-Block gesetzt. Um dem Abhilfe zu schaffen, fügte ich beim „setEvent“ die folgenden Zeilen für ein When-Objekt hinzu:

```
Where eventPlace = new Where();
eventPlace.setValueString(eventLocation);
myEvent.addLocation(eventPlace);
```

Jetzt lief es reibungslos!

Nachtrag: Mit der vorgenannten Variante werden Objekte im try-Block ignoriert, bei denen keine Orte spezifiziert worden sind. Stattdessen ist es natürlich sinnvoller, von vornherein einen Ort festzulegen, damit alle Termine erscheinen. Daher fügte ich noch folgende Code-Zeile hinzu, sodass leere Strings vermieden werden.

```
Where eventPlace = new Where();
// eventPlace.setLabel(eventLocation);
// prevent empty string (no location set)
if (eventLocation == "") eventLocation = "----";
eventPlace.setValueString(eventLocation);
myEvent.addLocation(eventPlace);
```

Und falls der User bereits Termine direkt im Kalender angelegt hatte, die keinen Ort beinhalten, musste ich den try-catch-Block zu folgender Version vervollständigen:

```
public String getQueryResults(int elementAt) {

    resultString =
        myResultsFeed.getEntries().get(elementAt).getTimes().get(0).getStartTime().toUiString() + " | " +
        myResultsFeed.getEntries().get(elementAt).getTitle().getPlainText();

    String location;
    try {
        location =
            myResultsFeed.getEntries().get(elementAt).getLocations().get(0).
            getValueString().toString();
    }
    catch (Exception e) {
        System.out.println("java: INVALID Location specified!");
        return "INCOMPLETE ENTRY IS: " +
            myResultsFeed.getEntries().get(elementAt).getTitle().getPlainText();
    }
    if (location == "") {
        location = "----";
    }
    resultString += " | " + location;
    return resultString;
}
```

Anschließend gestaltete ich das Interface um, sodass es etwas kompakter war und Platz für ein zukünftiges Menü geschaffen wurde.



Dann sollte im Start- und End-Feld jeweils das aktuelle Datum stehen. Hierfür griff ich auf `DateTime` zurück und definierte in der Java-Datei:

```
// returns recent date in format: 2009-01-15
public String getToday() {
    // "2009-01-19T11:21:26.256Z"
    String today = DateTime.now().toUiString();
    today = today.substring(0, 10);
    return today;
}
```

Und rief die Funktion bei den Text-Parametern der Textfelder auf:

```
text: calendar.getToday(); // "2009-01-16"
```

Als letztes fügte ich dem Hintergrund noch eine Grafik hinzu, sowie die Google-Logos für Calendar (später gleiches für Blogger). Hierzu lernte ich das Objekt „ImageView“ kennen (vgl. Code bei *imageBackground* und *imageCalendarIcon*).



Grafiken via ImageView hinzugefügt
Grafik: by Malabooboo¹⁵

Bindings in JavaFX fürs Design

Aus designtechnischen Gründen sollte man das Binding in JavaFX in Betracht ziehen. Wenn Positionierungsangaben beispielsweise an die Gesamtbreite gebunden sind, so kann das Layout bei Veränderungen der Hauptfenstergröße ("Stage") beibehalten werden; mit Orientierung an der Mitte (*ourStage.width/2*).

Ich versuchte dies mit der Startseite, musste jedoch feststellen, dass unbedingt auf die Reihenfolge der Initiierung zu achten ist. Sofern die Breite noch nicht spezifiziert wurde, erhält sie den Wert 0, was wiederum dazu führt, dass die Objekte an falschen Positionen stehen.

Da die Stage in meinem Code als letztes initiiert wurde, da vorher alle Sceen-Elemente existieren mussten, konnte auch keines der Elemente auf ihre *width* zugreifen. Ein Kunstgriff war notwendig: Ich initiierte die Stage Basisdaten gleich zu Beginn mit:

```
var ourStage:Stage = Stage {
    title: "GDATA Connect"
    width: 800
    height: 800
    visible: true
}
```

Fügte jedoch die Szene erst ganz am Ende des Codes hinzu:

```
ourStage.scene = ourScreen;
```

So klappte schließlich das dynamische Binding an die *ourStage.width* ohne Fehler!

¹⁵ <http://www.flickr.com/photos/malabooboo/2405993368/> (CC by-nc)

Binding einer SwingComponent

Das Binden der passwordComponent (die das Password-Field enthielt), stellte sich als schwierig heraus:

```
var passwordComponent = SwingComponent.wrap(passwordField);
// geht so nicht!
passwordComponent.translateX = bind (ourStage.width / 2) - 100;
passwordComponent.translateY = 140;
```

Da die SwingComponent eine abstrakte Klasse ist (d. h. man kann von ihr kein Objekt erstellen, sie ist eine Superklasse, von der geerbt werden muss), konnte ich kein direktes Binding vornehmen.

Der Umweg lief über das Erstellen einer neuen Klasse (!), die von der SwingComponent-Klasse erbt und die `createJComponent()` implementiert bzw. überschreibt.

Den alten Code:

```
var passwordComponent = SwingComponent.wrap(passwordField);
passwordComponent.translateX = ourStage.width / 2 - 100;
passwordComponent.translateY = 140;
passwordComponent.width = 200;
passwordComponent.wrap(passwordField);
```

ersetzte ich mit einem neuen Klassenkörper, der als JComponent das zuvor initiierte JPasswordField zurückwarf:

```
class swingHelper extends SwingComponent {
    override function createJComponent() {
        return passwordField;
    }
}
var passwordComponent:swingHelper = swingHelper {
    translateX: bind ourStage.width / 2 - 100
    translateY: 140
    width: 200
}
passwordComponent.wrap(passwordField);
```

Und endlich funktionierte auch das Binding mit der Stage-Breite.

Fazit an dieser Stelle: Warum also einfach, wenn es auch kompliziert geht?!

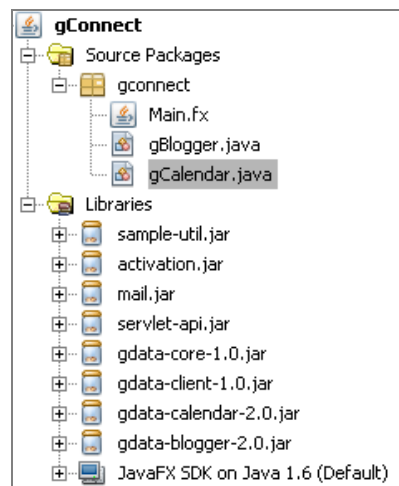
Let's do it Blogger.com!

Als nächstes sollte die Anbindung an Blogger inklusive Interface erstellt werden.

Zu Beginn waren die notwendige JAR-Datei (Library) "gdata-blogger-2.0.jar" in unser Projekt in NetBeans einzubinden.

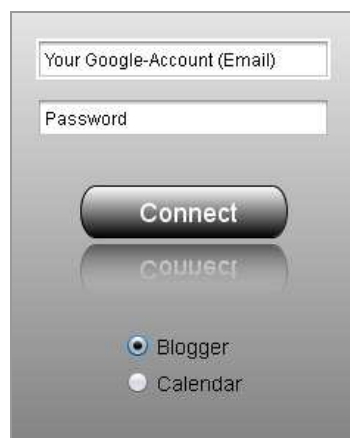
Anschließend erstellte ich eine JAVA-Datei namens „gBlogger.java“, die die Logik übernehmen sollte.

** Anmerkung: An dieser Stelle habe ich unser Projekt von „gCalendar“ in „gConnect“ umbenannt. Gleiches gilt für den Package-Namen. Das automatische Refactoring von NetBeans war hierbei eine große Hilfe (es vergaß nur den Package-Namen innerhalb der Main.fx und den dortigen Import der Calendar.java, alles andere klappte auf Anhieb).



Aktueller Workspace

Um es dem User schon zu Beginn auswählen zu lassen, welchen Service er als erstes sehen möchte (Blogger oder Calendar), fügte ich dem Startscreen eine ToggleGroup mit zwar RadioButtons hinzu:



Als nächstes legte ich in der Java FX eine Group „bloggerGroup“ an, die die Interface-Felder aufnehmen sollte.

Zufällig bin ich auf den Beispiel-Code der Google Developers gestoßen, der sich in der Java Client Library versteckte, unter `\gdata\java\sample\blogger\BloggerClient.java`.

Dort befanden sich einige Anbindungsmöglichkeiten, sodass ich den Code teilweise in meine `gBlogger.java` einbinden konnte.

Jedoch ergab sich die folgende Fehlermeldung, wenn ich ein `gBlogger`-Objekt erzeugen wollte:

```
java.lang.NoClassDefFoundError: com/google/gdata/data/media/mediarss/MediaThumbnail
    at com.google.gdata.data.blogger.PostEntry.declareExtensions(PostEntry.java:57)
    at com.google.gdata.data.ExtensionProfile.addDeclarations(ExtensionProfile.java:71)
    at com.google.gdata.data.BaseFeed.declareExtensions(BaseFeed.java:226)
    at
com.google.gdata.client.blogger.BloggerService.declareExtensions(BloggerService.java:145)
    at com.google.gdata.client.blogger.BloggerService.<init>(BloggerService.java:84)
    at gconnect.gBlogger.<init>(gBlogger.java:56)
    at gconnect.Main.javafx$run$(Main.fx:288)
    at gconnect.Main.javafx$run$(Main.fx:288)
```

In der `gBlogger.java` stand:

```
BloggerService myService = new BloggerService("exampleCo-exampleApp-1");
```

In der `Main.Fx`:

```
var blogservice:gBlogger = new gBlogger();
```

Die Fehlersuche begann. Nach etwa einer Stunde Suche (!) war ich immer noch nicht auf die Lösung gestoßen. Der Compiler konnte das Objekt nicht erstellen, weil ihm offensichtlich die `media/mediarss/MediaThumbnail` fehlte...

Schließlich erinnerte ich mich, dass das erste ANT-Beispielprogramm (das ich am Anfang des Dokuments im Zusammenhang mit Blogger zeigte) ja lauffähig war. So suchte ich also dort in den Build-Parametern und schaute mir noch einmal in der „`core.xml`“ die Dependencies an:

```
<!-- Dependency path for all media enabled services -->
<path id="build.service.media.classpath">
  <pathelement location="{mail.jar}"/>
  <pathelement location="{activation.jar}"/>
  <pathelement location="{gdata-media.jar}"/>
</path>
```

Und da war der Fehler! In unserem Projekt gab es in den Libraries für die Google Calendar Anbindung keine `gdata-media.jar`, die jedoch nun für Blogger notwendig wurde!

Sie fehlte dem Compiler also! So war dann die **Lösung** zu simpel: Einfach in NetBeans die Datei „`gdata-media-1.0.jar`“ (die sich bei den Samples in „`gdata\java\lib`“ befindet) bei den Libraries hinzufügen!

Die Ausgabe der Blogdaten in der Console funktionierte einwandfrei, die Verbindung war hergestellt!

Elemente für das Blogger Interface

Das Blogger-Interface im Browser sieht online wie folgt aus:

The screenshot shows the Blogger 'Create Post' interface. At the top, there are tabs for 'Create', 'Edit Posts', and 'Moderate Comments'. Below the tabs is a 'Title:' text input field. To the right of the title field are 'Edit HTML' and 'Compose' buttons. Below the title field is a rich text editor toolbar with icons for bold (b), italic (i), link, unlink, undo, redo, and image. A 'Preview' button is located to the right of the toolbar. The main content area is a large empty text box. Below the content area is a 'Post Options' section with a dropdown arrow. To the right of the dropdown is a 'Labels for this post:' field with the example text 'e.g. scooters, vacation, fall' and a 'Show all' link. Below the labels field are 'Reader Comments' options with radio buttons for 'Allow' (selected) and 'Don't allow'. To the right of the comments options is a 'Post date and time' section with input fields for '19.01.09' and '07:24'. At the bottom, there are 'PUBLISH POST' and 'SAVE NOW' buttons, and a 'Return to list of posts' link. A small text block at the bottom right contains shortcuts: 'Shortcuts: press Ctrl with: B = Bold, I = Italic, P = Publish, S = Save, D = Draft more'.

Das heißt, wir brauchten Text-Felder für TITLE, CONTENT, LABELS, POST-DATE, POST-TIME, sowie einer Checkbox oder Radiobuttons für ALLOW_COMMENTS.

The design draft shows a simplified version of the Blogger interface. It features a 'Google Blog Search BETA' logo in the top right corner. Below the logo is a 'Post Title' text input field. Underneath the title field is a large empty text box labeled 'CONTENT'. Below the content box is a 'Labels' text input field. To the right of the labels field are two input fields for '2009-01-19' and '18:00'. At the bottom left, there are radio buttons for 'Allow' (selected) and 'Don't allow'. At the bottom right, there is a 'Post now!' button.

Erster Entwurf für das Blogger-Interface

Auf den ersten Blick erschien das Interface sehr nüchtern. Mit einem Blogger-Icon links oben habe ich es schließlich aufgelockert.



Blogger-Icon

Außerdem nutzte ich meine Photoshop-Fähigkeiten und änderte die Beschriftung des Google-Logos von „Blog Search“ zu „Blogger“. Dies ist auf den Screenshots der folgenden Seiten zu sehen.

Merkwürdig war auch, dass der Text im `SwingTextField` „Content“ standardmäßig vertikal zentriert war. Leider gab es hier keine ad-hoc-Lösung in JavaFX. Das „vertical alignment“ ist nur für statischen Text vorgesehen (!) Auch mithilfe der `TextBox` im JavaFX-Paket konnte ich nichts erreichen, da sie die gleichen Eigenschaften aufzuweisen schien.

Nach einer Recherche bin ich auf „How to create a multi line text area using JavaFx“¹⁶ gestoßen, wo das gleiche Problem diskutiert wurde. Dies (also eine `TextArea` Komponente) ist bis dato nicht in JavaFX implementiert worden! Stattdessen wurde auf der genannten Seite eine JavaFX-Datei vorgestellt, deren Klasse Abhilfe schaffte.

Ich implementierte den Code als `TextArea.fx` (kommentierte noch die `translateX/Y` aus) und änderte die

```
var fieldBlogContent = SwingTextField {
```

```
ZU: var fieldBlogContent = TextArea {
```

Einziges Manko für Shortcut-Benutzer:

Die Tabulator-Taste zum Weiterspringen funktioniert aus diesem Feld heraus nicht, stattdessen wird ein Tab-Sprung im Text eingefügt. Wenn man ins nächste Feld gelangen möchte, so ist die Tastenkombination „Strg + Tab“ zu benutzen.

Anbindung des Interfaces an die Blogger API

Jetzt hieß es, sich erst einmal mit den Möglichkeiten vertraut zu machen. Das Interface fürs Blogging war gebaut, nun musste es nur noch logisch verknüpft werden.

Durch die Beispieldatei `BloggerClient.java` von Google bestanden bereits diverse Methoden:

- `createComment`
- `createPost`
- `deleteComment`
- `deletePost`
- `getBlogId`
- `printAllComments`
- `printAllPosts`
- `printDateRangeQueryResults`
- `printUserBlogs`

¹⁶ <http://forums.sun.com/thread.jspa?threadID=5345065>

- updatePostTitle

Auf *createPost* lag hierbei der Fokus. Diese Methode musste abgeändert werden, da sie ein Entry-Element zurückwarf. In unserer Java-Datei entfernte ich den Service, der bei den Parametern übergeben wurde, sowie das Return-Statement für den Entry:

```
public void createPost(String title, String content, String authorName,
    String userName, Boolean isDraft) throws ServiceException, IOException {
    // Create the entry to insert
    Entry myEntry = new Entry();
    myEntry.setTitle(new PlainTextConstruct(title));
    myEntry.setContent(new PlainTextConstruct(content));
    Person author = new Person(authorName, null, userName);
    myEntry.getAuthors().add(author);
    myEntry.setDraft(isDraft);

    // Ask the service to insert the new entry
    URL postUrl = new URL(feedUri + POSTS_FEED_URI_SUFFIX);
    Entry currentPost = myService.insert(postUrl, myEntry);
    System.out.println("Created post: "+currentPost.getTitle().getPlainText());
}
```

Ich schickte einen Testeintrag in JavaFX ab:

```
blogservice.createPost("Test Title", "<p>This is all the content!</p>", "Benar Cita",
    username, false)
```

und es funktionierte tatsächlich auf Anhieb! Im Browser öffnete ich meinen Blog und der Eintrag war vorzufinden.

Es fiel jedoch auf, dass es keinen Parameter für die Zeit des Posts gab, sodass ich hier nochmals in die Dokumentation zur API schauen musste. Dies konnte mit „Entry.setPublished“ festgelegt werden.

Darüber hinaus fehlten die Parameter für die Labels (in der API als Categories benannt) und der Boolean-Wert für die Erlaubnis von Kommentaren.

Auf der Suche nach dem Festlegen von Kommentaren bin ich bei der Entry-Klasse nur auf `getRights()` gestoßen, welche vermutlich jedoch für andere Zwecke gedacht ist.

Da ich keine andere Möglichkeit in der JavaDoc entdecken konnte, das (Nicht)Erlauben von Kommentaren je Post zu bestimmen, fragte ich bei den Google-Entwicklern für die Blogger API an (bisher jedoch ohne Antwort, 28.01.2009):

http://groups.google.com/group/bloggerDev/browse_thread/thread/d0fe17faa0378cd0?pli=1

http://groups.google.com/group/blogger-help-howdoi/browse_thread/thread/9c248201190c1013?pli=1

Bei den Labels sah es besser aus. Sie ließen sich zwar nicht direkt initiieren, doch mit Nutzung der Category-Klasse konnten sie gesetzt werden.

```
import com.google.gdata.data.Category; // for Labels

// (...)

Category category = new Category();
category.setScheme("http://www.blogger.com/atom/ns#");
category.setTerm(postLabels);
myEntry.getCategories().add(category);
```

Netterweise konnte ein kompletter String übergeben werden, der alle Labels enthält. So entfiel die Arbeit des Zerlegens in Substrings.

Ein leeres Label darf im Übrigen nicht übergeben werden, sonst tritt eine Exception auf. Dies habe ich mit einer if-Bedingung verhindert, die prüft, ob der String Inhalt hat.

Hier war noch darauf zu achten, wie die Strings verglichen werden. Ein einfaches if (x == "") funktionierte nicht, sondern warf die Exception:

```
Exception in thread "AWT-EventQueue-0" java.lang.RuntimeException:
com.google.gdata.util.InvalidEntryException: Bad Request
Invalid category term
```

So verglich ich die **Strings** (als Objekte) mit der **equals**-Methode:

```
if (postLabels.equals("")) { ... }
```

Gleichermaßen änderte ich in der gCalendar.java die Zeile

```
if (location == "") {
mit
    if (location.equals("")) {
```

trotzdem diese bisher merkwürdigerweise funktioniert hatte.

Beim Testen stellte ich ebenfalls fest, dass der **Autor** nicht explizit angegeben werden muss, wenn er mit dem User (bzw. Inhaber des Blogs) übereinstimmt. Für eine spätere Implementierung ließ ich diesen Code-Teil stehen und kommentierte ihn erst einmal nur aus, siehe unten.

Nachdem ich alles eingearbeitet und die Werte der Text-Felder mit den String-Variablen verknüpft hatte, sah der Code folgendermaßen aus:

In Java (Methode):

```
public void createPost(String title, String content,
    String postLabels, Boolean commentsAllowed, String postPublish,
    String authorName, String userName, Boolean isDraft)
    throws ServiceException, IOException {
    // create the entry to insert
    Entry myEntry = new Entry();
    // set title of post
    myEntry.setTitle(new PlainTextConstruct(title));
    // set content of post
    myEntry.setContent(new PlainTextConstruct(content));
    // set author of post (not necessary if user is author)
    // Person author = new Person(authorName, null, userName);
    // myEntry.getAuthors().add(author);
    // set draft or non-draft
    myEntry.setDraft(isDraft);
    // add time of post
    DateTime timePublished = DateTime.parseDateTime(postPublish);
    myEntry.setPublished(timePublished);
    // add category (i.e. label)
    Category category = new Category();
    category.setScheme("http://www.blogger.com/atom/ns#");
    category.setTerm(postLabels);
    myEntry.getCategories().add(category);
    // set allow or disallow of comments
    // commentsAllowed - implementation later

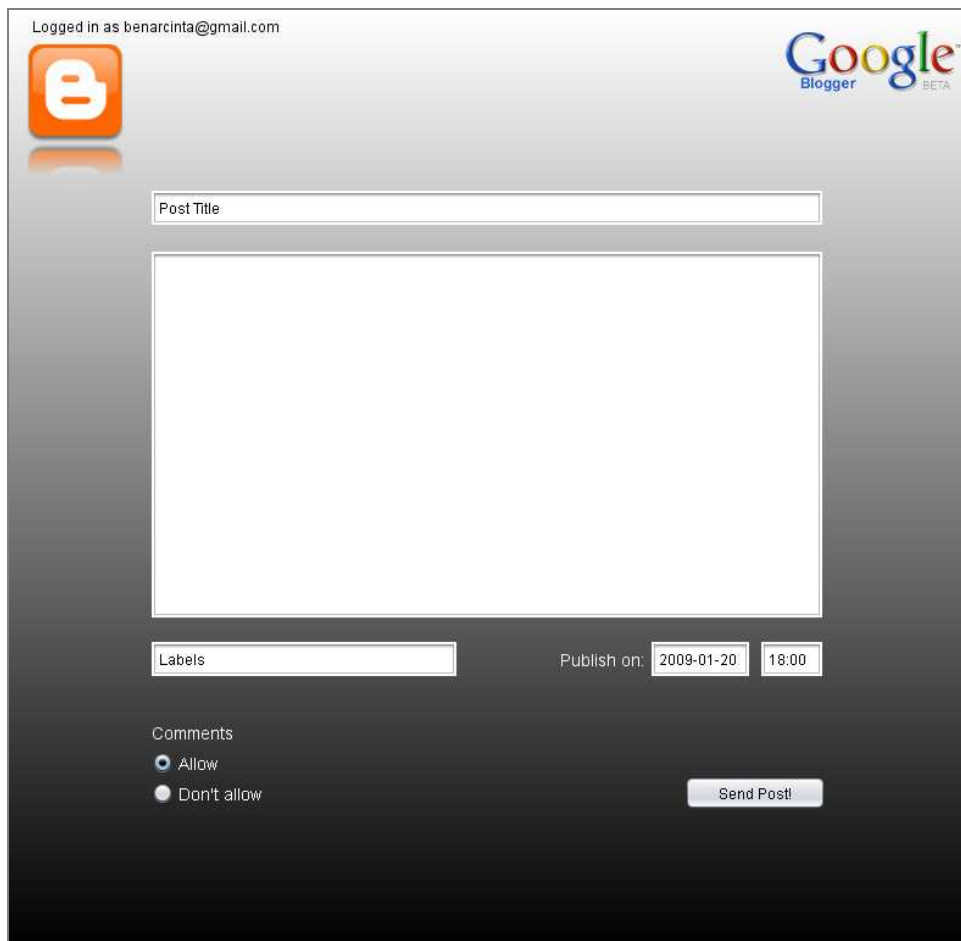
    // ask the service to insert the new entry
    URL postUrl = new URL(feedUri + POSTS_FEED_URI_SUFFIX);
    Entry currentPost = myService.insert(postUrl, myEntry);
    System.out.println("Created post: " + currentPost.getTitle().getPlainText());
}
```

Es ist noch zu erwähnen, dass ich in der javaFX-Datei die Variable „offsetTimezone“ global und static gemacht hatte, da sie sich nicht ändern sollte.

In Java FX sah die Übergabe an createPost() jetzt so aus:

```
Group {
    content: [
        buttonAddPost
    ]
    cursor: Cursor.HAND
    onMouseClicked: function(evt: MouseEvent):Void {
        // date format is ("yyyy-MM-ddTHH:mm:ss-SSSZ")
        var postPublish =
            "{fieldBlogPostDate.text}T{fieldBlogPostTime.text}:00{offsetTimezone}";
        // create our post
        blogservice.createPost(fieldBlogTitle.text, fieldBlogContent.text,
            fieldBlogLabels.text, ( if(bloggerCommentsAllowed.selected) true
            else false ), postPublish, blogservice.username, username,
            false); // false = nodraft
    }
}
```

Mit Blick auf das Interface bemerkte ich, dass Beschriftungen fehlten (Comments und PublishOn). Diese fügte ich als Text-Elemente in der Main.fx hinzu:



An dieser Stelle fügte ich noch onMouseClicked-Funktionen hinzu, und zwar dem Feld „Labels“ sowie dem Feld „Post Title“, die dafür sorgen, dass die Inhalte gelöscht werden, wenn der User auf sie klickt.

Wechseln zwischen Blogger- und Calendar-Interface

Für einen Wechsel zwischen den beiden Services musste ein Hinweis im Interface eingearbeitet werden. Mir kam die Idee, die zwei Icon-Grafiken verkleinert und in Graustufe zu nutzen, und zwar als Indikator links unten im Interface. Und statt Adobe Photoshop zu bemühen und zusätzliche Grafiken zu erstellen, wollte ich die Möglichkeiten der Klasse `javafx.scene.effect.Effect.ColorAdjust` nutzen.

Anfangs dachte ich, dass deren Methoden zur Veränderung von brightness, contrast, hue und saturation nicht geeignet sind (ich testete die Saturation von 0..1), doch als ich ein Bild in Photoshop testweise bearbeitete, verstand ich, dass die Saturation einen negativen Wert annehmen muss.



unverändert



Saturation -1.0

Zusätzlich machte ich das Objekt zu 50 % transparent.

```
imageCalendarIcon.opacity = 0.5;
```

Da ich zu Beginn die initiierten ImageView-Objekte (in JavaFx: `imageCalendarIcon` und `imageBloggerIcon`) für diesen Zweck verwendete und modifizierte, wurde es unübersichtlich und problematisch, da ich neben den Bildwerten auch die Positionen jedes Mal verändern musste (beim Umschalten des Services).

Da sich die Code-Zeilen stark vermehrten, sah ich ein, dass es an dieser Stelle tatsächlich einfacher ist, zwei zusätzliche ImageViews mit diesen Grafiken zu erstellen und dann einzubinden (siehe `imageBloggerIconSwap` und `imageCalendarIconSwap`).

Ein Bug trat auf: Die Blogger-Icon-Grafik (eine PNG) wurde nur als grau-weißes Quadrat dargestellt, wenn ich sie verkleinerte und die Saturation veränderte. Eine andere png-Grafik wurde mit dem gleichen Code problemlos initiiert.

Da es also offensichtlich Probleme mit der Saturation gab, entschied ich, auf den grafischen Effekt des Ausblässens via Saturation zu verzichten. Der Code sah nun wie folgt aus:

```
var imageBloggerIconSwap = ImageView {
    x: 16;
    y: bind ourStage.height - 140;
    opacity: 0.5
    scaleX: 0.5
    scaleY: 0.5
    image:
        Image {
            url: "{__DIR__}images/google_blogger_by_BrunoAbreu.png"
        }
}

// ...
// ...

Group {
    content: [
        imageBloggerIconSwap
    ]
    cursor: Cursor.HAND
}
```

```

onMouseEntered: function(evt: MouseEvent):Void {
    imageBloggerIconSwap.opacity = 1.0;
    imageBloggerIconSwap.scaleX = imageBloggerIconSwap.scaleY = 0.7;
}
onMouseExited: function(evt: MouseEvent):Void {
    imageBloggerIconSwap.opacity = 0.5;
    imageBloggerIconSwap.scaleX = imageBloggerIconSwap.scaleY = 0.5;
}
onMouseClicked: function(evt: MouseEvent):Void {
    blogservice.authenticate(username,password);
    calendarGroup.visible = false;
    bloggerGroup.visible = true;
}
}

```

Das Springen von einem Service zum anderen funktionierte nun, indem man auf das Icon unten links klickte.



Mehrere Blogs ansprechen

Wir hatten bisher nur einen Blog, den wir ansprachen. Ich nutzte Firefox, meldete mich bei Blogger an und erstellte einen zweiten Blog. Nun war ich gespannt, welcher Blog gewählt wurde.

Im Code wurde der Blog über die Blog-ID angesprochen:

```

// String
blogId = getBlogId(myService);

```

Die Methode `getBlogId` sollte uns via

```

return entry.getId().split("blog-")[1];

```

eigentlich nur eine Blog-ID zurückwerfen.

So war es dann auch, und zwar die vom zuletzt angelegten Blog (in unserem Fall der zweite).

Nun wurde es erforderlich, dass der User den Blog, in den gepostet wird, auswählen kann. Hierfür konnte eine Combo Box in das Interface aufgenommen werden, die alle Blogs aufnehmen sollte:

```
import javafx.ext.swing.SwingComboBox;
import javafx.ext.swing.SwingComboBoxItem;

// ...

//A combo box with two items
var combobox = SwingComboBox{
    translateX: 100
    translateY: 50
    width: 200
    items:[
        SwingComboBoxItem{ text: "1st Blog" },
        SwingComboBoxItem{ text: "2nd Blog" },
    ]
    selectedIndex: 0
};
```

Als Funktionalität musste vorgesehen werden, dass alle Blogs des Users abgefragt und als SwingComboBoxItem in die SwingComboBox gelegt werden. Dies erreichte ich wie folgt:

in Java FX:

```
// array to hold blogs of the user
var blogList: String[];
var blogsFound:Integer = 0;

// ...

if(choseStartWith.getSelection().text == "Blogger") {
    blogservice.authenticate(username,password);
    bloggerGroup.visible = true;
    // swapper to Calendar
    imageCalendarIconSwap.visible = true;
    // get all blogs of the user and fill combobox
    blogsFound = blogservice.howmanyUserBlogs();
    println("Total Blogs: {blogsFound}") ;

    if ( blogsFound > 0) {
        for (i in [0..(blogsFound - 1)]) {

            // if true then the full blog-name is returned
            insert blogservice.getUserBlogId(i, false) into blogList;

            // insert into combobox
            var item = SwingComboBoxItem {
                text: blogservice.getUserBlogId(i, true) // listing of Blogs
            };
            insert item into comboboxBlogs.items;
        }
        // necessary to focus a newly inserted blog in the list
        comboboxBlogs.selectedItem = comboboxBlogs.items[blogsFound-1];
    }
    else {
        // no blogs found
        var item = SwingComboBoxItem {
            text: "Sorry, no Blog found!"
        };
        insert item into comboboxBlogs.items;
    }
}
```

In Java verwendete ich Teile der „public static void printUserBlogs()“ und erstellte neue Methoden:

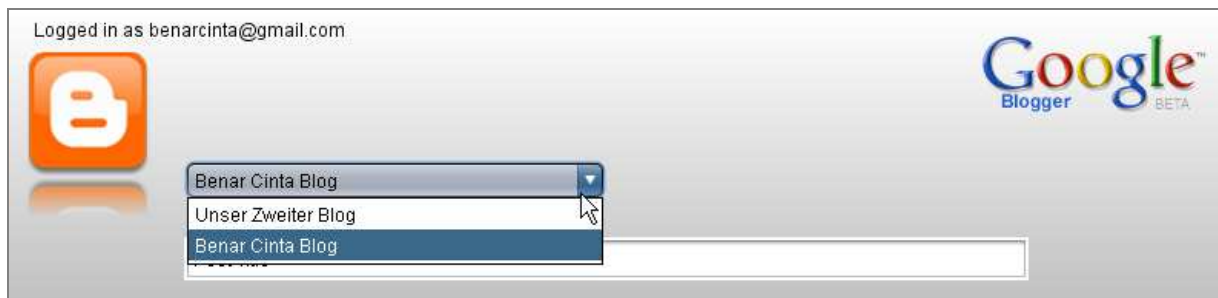
```
public String getUserBlogId(int position, Boolean wantName)
    throws ServiceException, IOException {
    // Request the feed
    final URL feedUrl = new URL(METAFEED_URL);
    Feed resultFeed = myService.getFeed(feedUrl, Feed.class);
    if (wantName) {
        // return Name of Blog
        return resultFeed.getEntries().get(position).getTitle().getPlainText();
    }
    else {
        // return ID
        return resultFeed.getEntries().get(position).getId();
    }
}

//returns the number of blogs found
public int howmanyUserBlogs()
    throws ServiceException, IOException {

    // Request the feed
    final URL feedUrl = new URL(METAFEED_URL);
    Feed resultFeed = myService.getFeed(feedUrl, Feed.class);

    // Print the results
    return resultFeed.getEntries().size();
}
```

Damit wurde die Anbindung hergestellt und die Blogs waren in der ComboList zu finden:



Leider musste ich jedoch feststellen, dass die Blog-Ids, die ich im String-Array „blogList“ (über die Abfrage `resultFeed.getEntries().get(position).getId()`); speicherte, nicht wie für den Feed benötigt formatiert waren.

Ich erhielt über `getId()` die feedURI:

```
http://www.blogger.com/feeds/tag:blogger.com,1999:user-753016354307.blog-7831192593990669774/posts/default
```

brauchte jedoch den String im Format:

```
http://www.blogger.com/feeds/7831192593990669774/posts/default
```

Ich ersetzte `getId()` mit der korrekten Variante, die mir nur den String nach „blog-“ zurückwarf:

```
// returns the ID which comes after 'blog-'
Entry entry = resultFeed.getEntries().get(position);
return entry.getId().split("blog-")[1];
```

Anschließend ergänzte ich die Parameter der createPost()-Methode mit der ID des aktiven Blogs:

```
public void createPost(String title, String content, String postLabels,
    Boolean commentsAllowed, String postPublish, String authorName,
    String userName, String activeBlogId, Boolean isDraft)
    throws ServiceException, IOException {

    // create the entry to insert
    Entry myEntry = new Entry();
    // set title of post
    myEntry.setTitle(new PlainTextConstruct(title));
    // set content of post
    myEntry.setContent(new PlainTextConstruct(content));
    // set author of post (not necessary if user is author)
    // Person author = new Person(authorName, null, userName);
    // myEntry.getAuthors().add(author);
    // set draft or non-draft
    myEntry.setDraft(isDraft);
    // add time of post
    DateTime timePublished = DateTime.parseDateTime(postPublish);
    myEntry.setPublished(timePublished);
    // add category (i.e. label)
    if (postLabels.equals("")) {
        // do nothing
    }
    else {
        Category category = new Category();
        category.setScheme("http://www.blogger.com/atom/ns#");
        category.setTerm(postLabels);
        myEntry.getCategories().add(category);
    }
    // insert the new entry according to the active blog
    URL postUrl = new URL(FEED_URI_BASE + "/" + activeBlogId +
        POSTS_FEED_URI_SUFFIX);
    Entry currentPost = myService.insert(postUrl, myEntry);
}
```

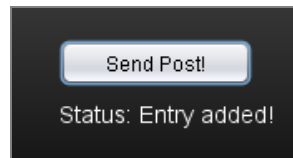
Und ergänzte die Funktion onMouseClicked() in der JavaFX-Datei beim buttonAddPost um die Zeile:

```
onMouseClicked: function(evt: MouseEvent):Void {
    var activeBlogId = blogList[comboboxBlogs.selectedIndex];
    // date format is ("yyyy-MM-ddTHH:mm:ss-SSSZ")
    var publishPostOn =
        "{fieldBlogPostDate.text}T{fieldBlogPostTime.text}:00{offsetTimezone}";
    // create our post
    blogservice.createPost(fieldBlogTitle.text, fieldBlogContent.text,
        fieldBlogLabels.text, (if(bloggerCommentsAllowed.selected) true else false),
        publishPostOn, blogservice.username, username, activeBlogId, false);
}
```

Feedback im Interface

Da ich bisher das erfolgreiche Hinzufügen eines Events bzw. Posts nur in der Java-Console ausgeben ließ, musste ich nun einen Weg finden, auch dem User im Interface ein Feedback zu geben. Dass beispielsweise ein Eintrag hinzugefügt wurde oder dass etwas nicht funktionierte.

Für diesen Zweck erstellte ich ein Text-Element „feedbackBloggerText“, das den jeweiligen Status berichtet.



Bestätigung für den hinzugefügten Post (Blogger)

Ich änderte die createPost() Methode in der gBlogger.java so, dass sie mir ein True zurückwarf, sofern es keinen Fehler beim Posting gab.

```
public Boolean createPost
```

Und den Code in der JavaFx änderte ich zu:

```
var posted:Boolean = blogservice.createPost(fieldBlogTitle.text, fieldBlogContent.text,
    fieldBlogLabels.text, (if(bloggerCommentsAllowed.selected) true else false ),
    publishPostOn, blogservice.username, username, activeBlogId, false);
if (posted == true) {
    feedbackBloggerText.content = "Status: Entry added!"
}
else {
    feedbackBloggerText.content = "Status: Error!"
}
```

Bei dieser Gelegenheit fügte ich der createPost() einen Try-Catch-Block hinzu, der noch fehlte, um Exceptions aufzufangen:

```
Entry currentPost;
try {
    currentPost = myService.insert(postUrl, myEntry);
}
catch (IOException ex) {
    Logger.getLogger(BloggerService.class.getName()).log(Level.SEVERE, null, ex);
    return false;
} catch (ServiceException ex) {
    Logger.getLogger(BloggerService.class.getName()).log(Level.SEVERE, null, ex);
    return false;
}
System.out.println("java: Created post: " + currentPost.getTitle().getPlainText());
// return true if posting worked without an error
return true;
```

Analog implementierte ich den Code für Interface und Logik der Kalender-Applikation!

Mir fiel beim Testen auf, dass noch ein Problem bestand: Startet man mit dem Kalender und wechselt dann über das Blogger-Icon links unten zu Blogger, so wurde nur die Authentifizierung durchgeführt, jedoch nicht die ComboBox gefüllt.

Daher baute ich den Code etwas um und verlagerte die Authentifizierung inkl. das Füllen der ComboBox in die Funktion `doBlogger()`. Zusätzlich fügte ich die Boolean-Variable „`blogActivated`“ hinzu, um sicherzustellen, dass nicht mehrfach initiiert wird.

```
// imageBloggerIconSwap
onMouseClicked: function(evt: MouseEvent):Void {
    if (blogActivated == false) {
        doBlogger();
        blogActivated = true;
    }
    calendarGroup.visible = false;
    bloggerGroup.visible = true;
}
```

Selbiges implementierte ich für den Kalenderteil unserer Applikation. Eine Mehrfach-Initiierung war nun ausgeschlossen.

Login-Daten speichern

Da es für den User mühselig ist, jedes Mal den Login-Namen neu einzugeben, entschloss ich mich, diesen nach der ersten Eingabe zu speichern.

Um dies zu erreichen, musste ich die `FileOutputStream` / `FileInputStream` Klassen von Java einbinden. Hier halfen mir Tipps von „Java 103: File-handling under Java“¹⁷.

Ich erstellte eine Funktion namens `findUsername()`, die ich zu Beginn aufrief, um festzustellen, ob es bereits einen bestehenden User gab (ansonsten wurde „*Your Google-Account (Email)*“ in das Username-Feld geschrieben). Außerdem erstellte ich eine Funktion `saveUsername()` für die Speicherung des eingegebenen Namens.

```
function findUsername():String {
    try {
        // open the file
        var fstream:FileInputStream = new FileInputStream("lastuser.txt");
        // convert our input stream to a DataInputStream
        var readIn:DataInputStream = new DataInputStream(fstream);
        var readUsername = readIn.readLine();
        readIn.close();
        // if no data is present (file is empty)
        if (readUsername.equals("") or readUsername == null) {
            readUsername = "Your Google-Account (Email)";
        }
        else {
            // if user has been found, focus the password field
            passwordComponent.requestFocus();
            // and delete the text in the field
            passwordField.setText("");
            // set up password char *
            passwordField.setEchoChar(theChar);
        }
        // return the user name fetched
        return readUsername;
    }
}
```

¹⁷ <http://www.javacoffeebreak.com/java103/java103.html>

```

catch (ex:IOException) {
    println ("Error reading file: {ex}");
};

// if no file has been found
return "Your Google-Account (Email)"
}

function saveUsername():Void {
    // Create a new file output stream connected to "myfile.txt"
    var outputStream:FileOutputStream;
    var p:PrintStream; // declare a print stream object

    try {
        // file output object
        outputStream = new FileOutputStream("lastuser.txt");
        p = new PrintStream(outputStream);
        // write user to file
        p.println (user);
        p.close();
    }
    catch (ex:IOException) {
        println ("Error writing to file: {ex}");
    };
}
}

```

Falschen Login abfangen

Ich bemerkte ebenfalls, dass ein falscher Login im derzeitigen Code nicht abgefangen wurde, sondern nur den Startscreen ausblendete. Ich musste also abfragen, ob ein Service zu Google etabliert wurde. So lagerte ich Authentifizierungs-/Initiierungscode für Blogger und Calendar in JavaFX in die separaten Methoden *doBlogger()* und die *doCalendar()* aus und ließ jeweils „true“ zurückwerfen, wenn eine Verbindung hergestellt wurde:

Änderungen für Calendar in der JavaFX:

```

obsolet:    function doCalendar() {
              calendar.authenticate(username,password);
            }

neu:        function doCalendar():Boolean {
              if( calendar.authenticate(username,password) ) {
                calendarActivated = true;
                calendarGroup.visible = true;
                // swapper to Blogger
                imageBloggerIconSwap.visible = true;
                return true;
              }
              else {
                return false;
              }
            }
}

```

Die **gCalendar.java** musste darauf einen Boolean-Wert zurückgeben, Ergänzung:

```

Boolean authenticate(String user, String pwd) {
    this.username = user;
    this.password = pwd;
    try {
        myService.setUserCredentials(user, pwd);
    }
}

```

```

    } catch (AuthenticationException ex) {
        Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE,
            null, ex);
        System.out.println("java: CANNOT ESTABLISH CONNECTION!");
        return false;
    }
    System.out.println("java: CONNECTED");
    return true;
}

```

Änderungen für Blogger in der JavaFX:

obsolet:

```

function doBlogger() {
    blogservice.authenticate(username,password);
    // ...
    insert item into comboboxBlogs.items;
}

```

neu:

```

function doBlogger():Boolean {
    if ( blogservice.authenticate(username,password) ) {
        blogActivated = true;
        // ...
        insert item into comboboxBlogs.items;
    }
    return true;
}
else {
    // login wrong or connection not working
    return false;
}
}

```

Die authenticate() in **gBlogger.java** musste nun auch einen Boolean-Wert zurückgeben, es wurden ergänzt:

```

Boolean authenticate(String user, String pwd) {
    System.out.println("java: TRYING TO AUTHENTICATE");
    this.username = user;
    this.password = pwd;
    try {
        // Authenticate using ClientLogin
        myService.setUserCredentials(user, pwd);
    } catch (AuthenticationException ex) {
        Logger.getLogger(BloggerService.class.getName()).log(
            Level.SEVERE, null, ex);
        System.out.println("java: CANNOT ESTABLISH CONNECTION!");
        return false;
    }
    System.out.println("java: CONNECTED");
    try {
        // Get the blog ID from the metatfeed
        blogId = getBlogId(myService);
        System.out.println("java: blogId: " + blogId);
        feedUri = FEED_URI_BASE + "/" + blogId;
    } catch (ServiceException se) {
        se.printStackTrace();
    } catch (IOException ioe) {
        ioe.printStackTrace();
        System.out.println("java: CANNOT ESTABLISH CONNECTION!");
    }
    return true;
}

```

In der Java-FX-Datei fügte ich beim onMouseClicked() für den Login-Button hinzu:

```
// connect to google
var connected:Boolean = false;

// BLOGGER chosen at STARTUP
if(choseStartWith.getSelection().text == "Blogger") {
    connected = doBlogger();
}
// CALENDER chosen at STARTUP
else if (choseStartWith.getSelection().text == "Calendar") {
    connected = doCalendar();
    calendarActivated = true;
    calendarGroup.visible = true;
    // swapper to Blogger
    imageBloggerIconSwap.visible = true;
}
if (connected) {
    startGroup.visible = false;
}
else {
    // login failed
    loggedInText.content = "Your Login failed !!!"
}
}
```

So erschien bei falschem Login als Feedback links oben im Startscreen eine Nachricht an den User!

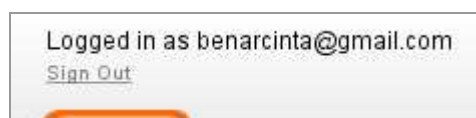
Für die Textfelder Username und Password des Startscreens ergänzte ich bei den onMouseClicked()-Methoden zwei Zeilen, die dafür sorgten, dass bei der Neueingabe der Daten, das Label „Your Login failed !!!“ wieder verschwindet:

```
// if there was a wrong login beforehand, hide this label again
loggedInText.content = "";
```

Logout-Button

Als ich einen Freund als „Normal-User“ ☺ den Prototyp der Applikation testen ließ, fragte er mich, wo der Logout-Button sei. Aus diesem Grund entschloss ich mich, ein „Sign Out“ direkt unter dem Label „Logged in as user@gmail.com“ (siehe *signoutText*) zu positionieren. Funktionalität:

```
Group {
    content: [
        signoutText
    ]
    cursor: Cursor.HAND
    onMouseClicked: function(evt: MouseEvent):Void {
        java.lang.System.exit(0);
    }
}
```



Sign Out implementiert

Problem mit dem Password-Field (JavaFX Bug)

Als ich den User, der meist mit Keyboard navigiert, berücksichtigen wollte und Tabulator-Sprünge aufs Password-Feld beim Startscreen und die Enter-Taste als Bestätigung abfangen wollte, stellte sich leider heraus, dass JavaFX in der aktuellen Version 1.0 für ein `SwingTextField` keine Key-Events empfangen kann.

Siehe Bug: **“Unable to catch key event from a `SwingTextField`”**¹⁸ (2009-01-09)

Das hieß also, für unsere Applikation konnten wir das Password-Feld nicht registrieren, sofern wir mit dem Tabulator vom Usernamen zum Password-Feld sprangen. So war der aktuelle Zustand der, dass man mit dem Tabulator ins Feld springt, ohne dass es das Feld bemerkt und den default-Inhalt „Password“ löschen würde.

Hinweis: Dies geschieht nur, wenn die Applikation das erste Mal startet (ohne Inhalt in der „lastuser.txt“), sodass der Username für die Eingabe fokussiert wird. Steht ein Username bereits fest, wird der Fokus gleich auf das Password-Feld gesetzt und es steht ordnungsgemäß bereit.

Für dieses Problemchen wollte ich trotzdem einen Workaround bauen.

Am Ende der JavaFX-Datei `Main.fx` fügte ich eine Timeline ein:

```
import javafx.animation.Timeline;
import javafx.animation.KeyFrame;
import javafx.animation.Interpolator;

// if no username has been specified
if (usernameTextfield.text == "Your Google-Account (Email)") {
    Timeline {
        keyFrames: [
            KeyFrame {
                time: 0s
            }
            KeyFrame {
                time: 5s
                action: function():Void {
                    passwordField.setText("");
                    // set up password char *
                    passwordField.setEchoChar(theChar);
                }
            }
        ]
    }.play();
}
```

So wurde der Hinweis „Password“ aus dem Passwortfeld nach 5 Sekunden automatisch gelöscht, das heißt in der Zeit, in der der User seine Email-Adresse als Login eingibt.

¹⁸ <http://forums.sun.com/thread.jspa?threadID=5359494&tstart=0>

Sortierung der Kalender-Einträge

Da ich bisher die Events des Kalenders aus der JavaFX über eine Vorschleife abrief, entschied ich mich,

```
for (i in [0..(entriesFound - 1)]) {
    // for (var i = 0; i < entriesFound; i++) {
    insert calendar.getQueryResults(i) into recentResults;
    // insert into list
    var item = SwingListItem {
        text: recentResults[i] // listing of events
    };
    insert item into resultsList.items;
```

zu ersetzen durch:

```
// gives a String Array
var entryListing = calendar.getQueryResultsArray();

for (i in [0..(entryListing.size() - 1)]) {
    insert entryListing[i] into recentResults;
    // insert into list
    var item = SwingListItem {
        // listing of events
        text: entryListing[i].toString();
    };
    insert item into resultsList.items;
}
```

Das String-Array konnte nun doch verwendet werden, was merkwürdigerweise im Gegensatz zu den zuvor erwähnten Problemen mit Arrays in Java / JavaFX stand.

Die Methode **getQueryResultsArray()** in der `gCalendar.java` sortierte das String-Array, das die Kalender-Einträge aus der Query enthielt, nun mit folgendem Code:

```
public String[] getQueryResultsArray() {
    // iterate over all results and save them in string array
    String[] strArray = new String[myResultsFeed.getEntries().size()];
    for (int i = 0; i < myResultsFeed.getEntries().size(); i++) {
        strArray[i] =
            myResultsFeed.getEntries().get(i).getTimes().get(0).
                getStartTime().toUiString()
            + " | " +
            myResultsFeed.getEntries().get(i).getTitle().getPlainText();

        String location = null;
        try {
            location =
                myResultsFeed.getEntries().get(i).getLocations().get(0).
                    getValueString().toString();
        }
        catch (Exception e) {
            System.out.println("java: INVALID Location specified!!");
        }
        if (location.equals("")) {
            location = "---";
        }
        strArray[i] += " | " + location;
    }

    // sort the string array
    java.util.Arrays.sort(strArray);

    return strArray;
}
```

Letzter Schritt: Deployment

Entsprechend des Status der Applikation habe ich die Versionsnummer 0.01 hinzugefügt und beschäftigte mich danach mit dem Deployment. Dafür warf ich einen Blick ins Tutorial „Deploying the JavaFX Application Outside of the IDE“¹⁹.

Auf dieser Seite findet man Auswahlmöglichkeiten zu „Standard Execution“, „Web Start Execution“ und „Run in Browser“. Bei den ersten beiden steht „stand-alone desktop application“, bei letzterer steht „applet-based desktop application“.

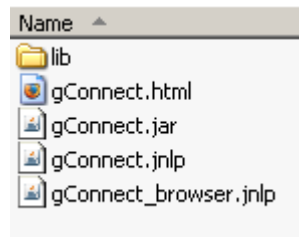
Eigene Start-Verknüpfung vom Desktop

Für den Start aus der **Windows-Konsole** (Command Line) gab ich aus dem Verzeichnis:

```
C:\Dokumente und Einstellungen\Kai\Eigene Dateien\workspace\gConnect\dist\
```

heraus folgenden Befehl ein (man beachte die Anführungszeichen unter Windows XP):

```
"C:\Programme\NetBeans 6.5\javafx2\javafx-sdk1.0\bin\javafx.exe" -classpath gConnect.jar gconnect.Main
```



Listung der Dateien im Verzeichnis

Die Applikation startete und funktionierte soweit ohne Probleme.

Natürlich ist es so möglich, einfach eine Verknüpfung auf dem Desktop zu legen, sofern JavaFX schon installiert wurde, und dort die notwendigen Parameter anzuhängen. Wichtig ist es hierfür nur, den richtigen Pfad zur javafx.exe zu kennen. Außerdem sind sämtliche Jar-Dateien mitzuliefern. Meine Verknüpfung:

```
"C:\Programme\NetBeans 6.5\javafx2\javafx-sdk1.0\bin\javafx.exe" -classpath gConnect.jar gconnect.Main
```

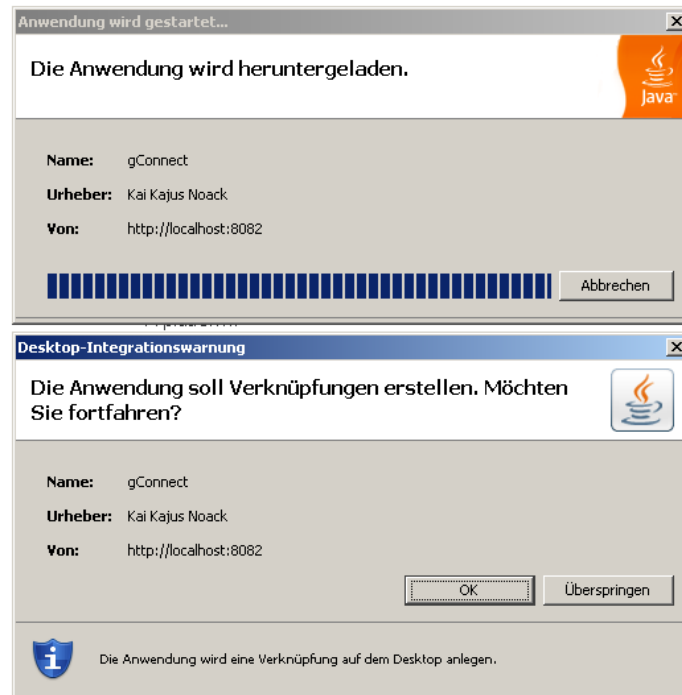


Einziges Manko: Beim Starten öffnet sich zusätzlich das Kommandofenster von Windows, das als Java-Console dient und einigen Usern Skepsis abverlangt.

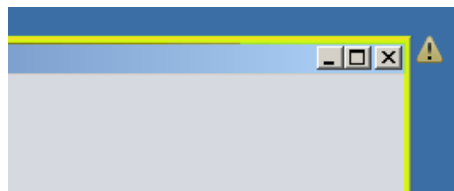
¹⁹ <http://javafx.com/docs/tutorials/deployment/>

Java Web Start

Danach versuchte ich die Variante über **Java Web Start** und folgte dem Tutorial für das Deployment in NetBeans²⁰



Dies schien auf Anhieb zu funktionieren, die Applikation wurde eingerichtet.



Beim Start erschien jedoch nur ein graues Fenster mit einem Warnzeichen.

Um zu prüfen, warum die Applikation nicht angezeigt wurde, startete ich die Java Konsole (im Tray-Icon rechten Mausklick > Konsole anzeigen; oder Systemsteuerung >Java >Erweitert +Java-Konsole >Konsole einblenden).

Die Fehlerausgabe besagte:

```
java.security.AccessControlException: access denied (java.io.FilePermission lastuser.txt read)
    at java.security.AccessControlContext.checkPermission(Unknown Source)
    at java.security.AccessController.checkPermission(Unknown Source)
    at java.lang.SecurityManager.checkPermission(Unknown Source)
    at java.lang.SecurityManager.checkRead(Unknown Source)
    at java.io.FileInputStream.<init>(Unknown Source)
    at java.io.FileInputStream.<init>(Unknown Source)
    at gconnect.Main.findUsername(Main.fx:539)
    at gconnect.Main.javafx$run$(Main.fx:1030)
```

²⁰ <http://javafx.com/docs/tutorials/deployment/configure-for-deploy.jsp#jwsmodel>

Es war also die Datei *lastuser.txt*, die Sicherheitsprobleme bereitete.

Kein Wunder! Ich kompilierte ja auch für den Web-Start und aus einem Applet heraus konnte ich, ohne gesonderte Rechte zu vergeben, keine Dateien schreiben.

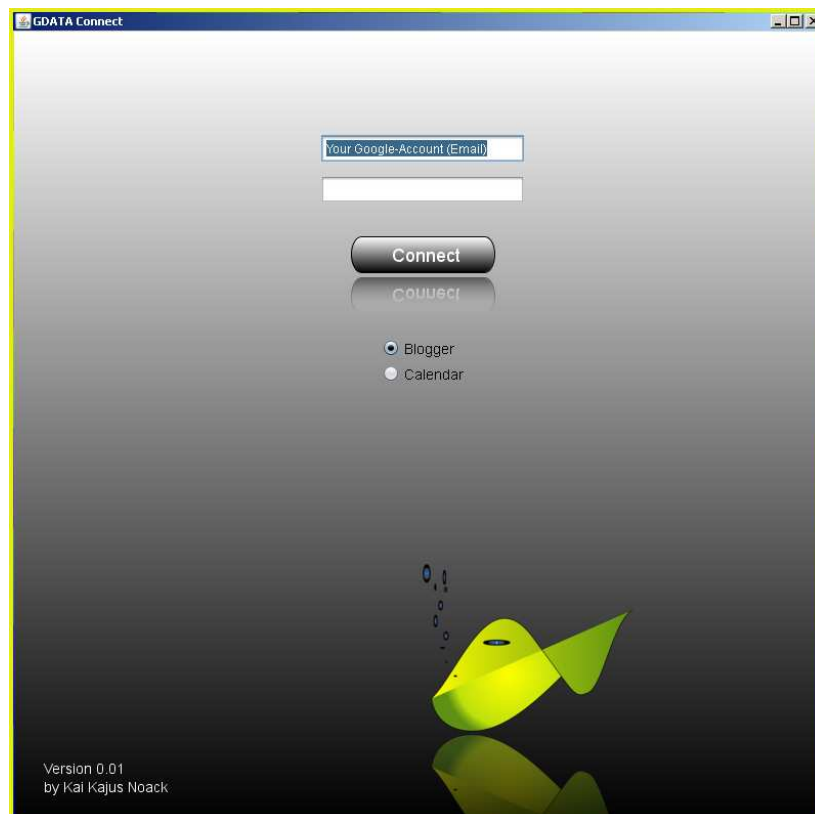
In der JNLP Datei im Verzeichnis `\dist` fügte ich daher hinter dem Tag „`</information>`“ folgende Zeilen hinzu:

```
<security>
  <all-permissions/>
</security>
```

Nun startete die Applikation mit Doppelklick auf die *gConnect.jnlp*.
Als erstes erschien das Java 6-Logo:

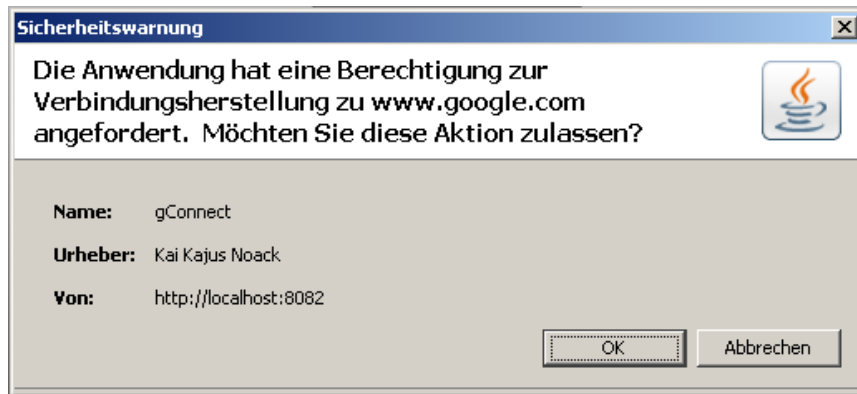


Danach öffnete sich meine Applikation:



Als nächstes gab ich die Login-Daten ein und klickte auf Connect. Unsere Rich Internet Application sollte nun eine Verbindung zu Google.com herstellen, um die Services anzusprechen.

Es erschien eine Sicherheitswarnung:



Nachdem ich OK betätigte, tauchte in der Java Konsole eine Fehlermeldung auf:

```
Exception in thread "AWT-EventQueue-0" java.lang.ExceptionInInitializerError
    at com.google.gdata.client.http.GoogleGDataRequest$Factory.getRequest
      (GoogleGDataRequest.java:77)
    at com.google.gdata.client.Service.createRequest(Service.java:600)
    at com.google.gdata.client.GoogleService.createRequest(GoogleService.java:486)
    at com.google.gdata.client.Service.createFeedRequest(Service.java:988)
    at com.google.gdata.client.Service.getFeed(Service.java:818)
    at com.google.gdata.client.GoogleService.getFeed(GoogleService.java:592)
    at com.google.gdata.client.Service.getFeed(Service.java:838)
    at gconnect.gBlogger.getBlogId(gBlogger.java:89)
    at gconnect.gBlogger.authenticate(gBlogger.java:60)
    at gconnect.Main.doBlogger(Main.fx:991)
    at gconnect.Main$27.lambda(Main.fx:624)
    at gconnect.Main$27.invoke(Main.fx:612)
    at gconnect.Main$27.invoke(Main.fx:612)

Caused by: java.security.AccessControlException: access denied
    (java.util.PropertyPermission com.google.gdata.DisableCookieHandler read)
    at java.security.AccessControlContext.checkPermission(Unknown Source)
    at java.security.AccessController.checkPermission(Unknown Source)
    at java.lang.SecurityManager.checkPermission(Unknown Source)
    at java.lang.SecurityManager.checkPropertyAccess(Unknown Source)
    at java.lang.System.getProperty(Unknown Source)
    at java.lang.Boolean.getBoolean(Unknown Source)
```

Variante 1: Einbinden mehrerer Zertifikate

Eine Sicherheitseinstellung war folglich noch nicht richtig gewählt. Meine Recherche ergab, dass es hierfür zwei Lösungen geben soll:

1. Applet signieren oder
2. java.policy Datei ändern (in C:\Programme\Java\jre6\lib\security\java.policy)

Das heißt es lagen wahrscheinlich unterschiedliche Zertifikate bzw. Signierungen in den Libraries (den JAR-Dateien) vor.

Im Internet fand ich: „Java Web Start can use multiple JAR files signed by different certificates, by using the component extension mechanism and **multiple JNLP files**. The only requirement is that the JAR files contain code from different packages.”²¹

²¹ <http://java.sun.com/j2se/1.5.0/docs/guide/javaws/developersguide/faq.html#213>

Ich verlagerte demnach jede JAR-Datei, die wir benutzten, in eine separate JNLP und fügte diese über eine `<extension... >` in die Hauptdatei **gConnect.JNLP** ein.

Aus:

```
<resources>
  <j2se version="1.5+"/>
  <extension name="JavaFX Runtime"
    href="http://dl.javafx.com/javafx-rt.jnlp"/>
  <jar href="gConnect.jar" main="true"/>
  <jar href="lib/sample-util.jar"/>
  <jar href="lib/activation.jar"/>
  <jar href="lib/mail.jar"/>
  <jar href="lib/servlet-api.jar"/>
  <jar href="lib/gdata-core-1.0.jar"/>
  <jar href="lib/gdata-client-1.0.jar"/>
  <jar href="lib/gdata-calendar-2.0.jar"/>
  <jar href="lib/gdata-blogger-2.0.jar"/>
  <jar href="lib/gdata-media-1.0.jar"/>
</resources>
```

wurde:

```
<resources>
  <j2se version="1.5+"/>
  <extension name="JavaFX Runtime"
    href="http://dl.javafx.com/javafx-rt.jnlp"/>
  <jar href="gConnect.jar" main="true"/>
  <extension name="Java Helper" href="help-sample-util.jnlp"/>
  <extension name="Java Helper" href="help-activation.jnlp"/>
  <extension name="Java Helper" href="help-mail.jnlp"/>
  <extension name="Java Helper" href="help-servlet-api.jnlp"/>
  <extension name="Java Helper" href="help-gdata-core-1.0.jnlp"/>
  <extension name="Java Helper" href="help-gdata-client-1.0.jnlp"/>
  <extension name="Java Helper" href="help-gdata-calendar-2.0.jnlp"/>
  <extension name="Java Helper" href="help-gdata-blogger-2.0.jnlp"/>
  <extension name="Java Helper" href="help-gdata-media-1.0.jnlp"/>
</resources>
```

Somit wurden die verschiedenen Zertifikate für die einzelnen JAR-Dateien über verschiedene JNLPs „umgeleitet“. Die neuen „Umleitungs“-JNLP hatten jeweils folgende Struktur:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" codebase="http://ws503" href="Help.jnlp">
  <information>
    <title>gConnect Helper</title>
    <vendor>Kai Kajus Noack</vendor>
  </information>
  <resources>
    <jar href="lib/activation.jar"/>
  </resources>
  <component-desc/>
</jnlp>
```

Nach dem Programmstart erschien erneut ein Fehler:

```
com.sun.deploy.net.FailedDownloadException:
Ressource konnte nicht geladen werden: http://ws503/sample-util.jar

// ...
```

Dieser Fehler wurde verursacht, da ich oben in der help-sample-util.jnlp zu stehen hatte:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" codebase="http://ws503" href="Help.jnlp">
```

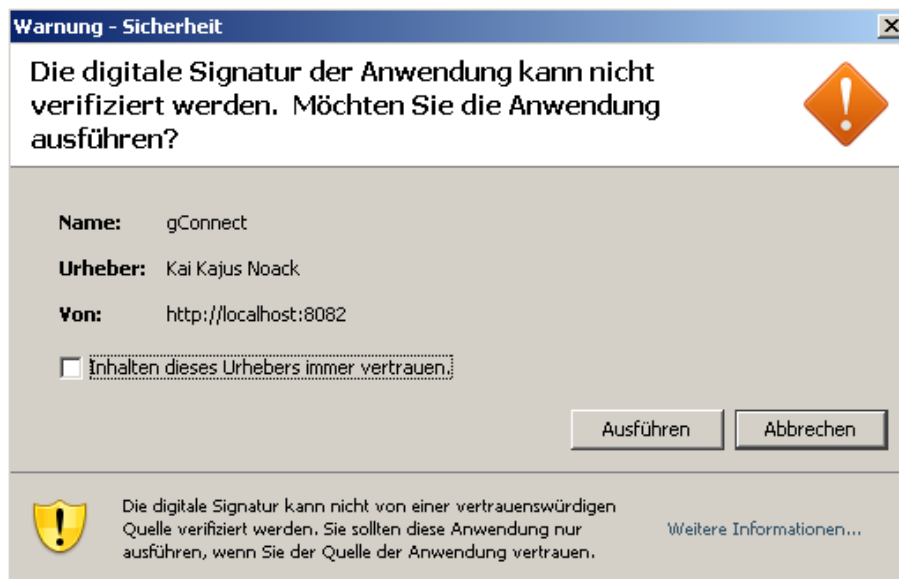
Und somit die Pfadangabe falsch war! Es musste also analog zur gConnect.jnlp heißen:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+"
codebase="http://localhost:8082/servlet/org.netbeans.modules.javafx.project.J
nlpDownloadServlet/C%3A/Dokumente+und+Einstellungen/Kai/Eigene+Dateien/worksp
ace/NetBeansProjects/gConnect/dist/" href="gConnect.jnlp">
```

Nun wurde die Anwendung beim Start „heruntergeladen“:

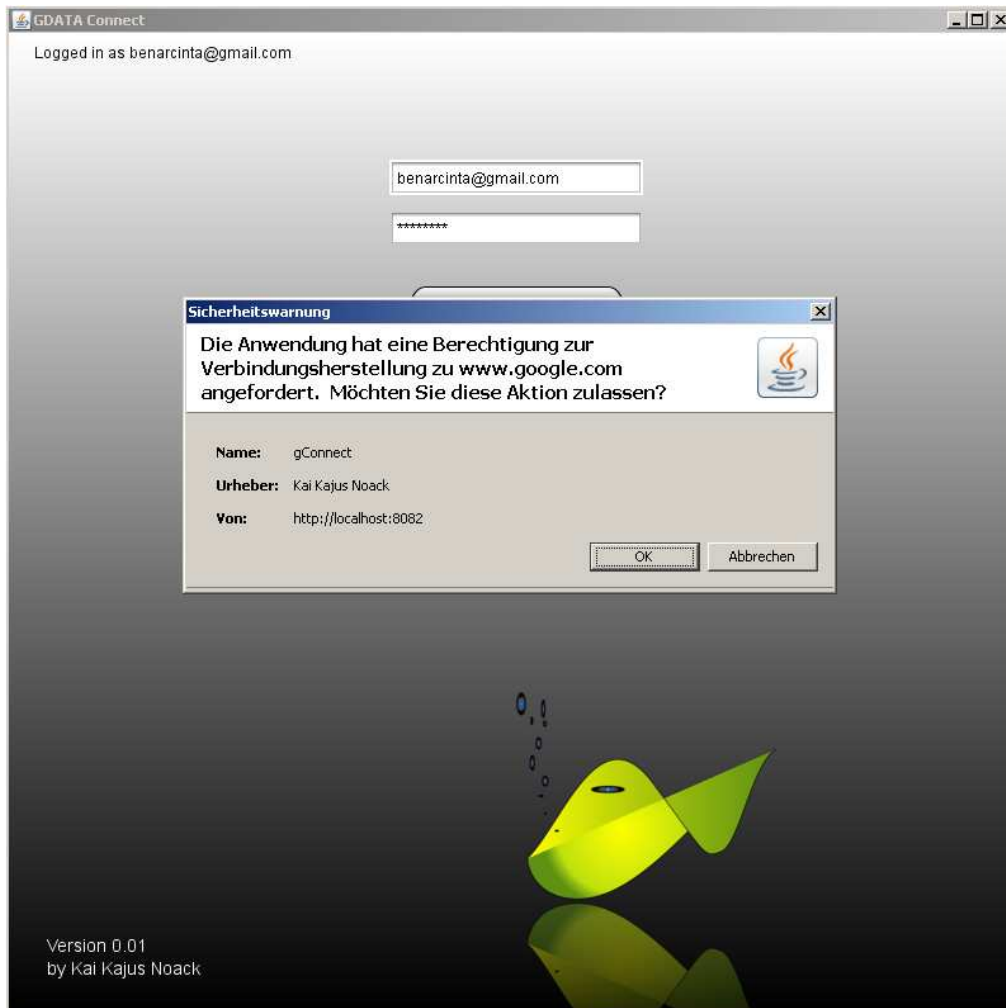


und ein Bestätigungsfenster erschien:



Nachdem ich mir vertraute, klickte ich auf „Ausführen“.

Die Applikation startete, ich loggte mich ein, danach erschien die Abfrage:



Ich bestätigte und es erschien wieder eine Exception in der Konsole mit der Meldung:

```
Exception in thread "AWT-EventQueue-0" java.lang.ExceptionInInitializerError
at
com.google.gdata.client.http.GoogleGDataRequest$Factory.getRequest(GoogleGDataRequest.java:77)
at com.google.gdata.client.Service.createRequest(Service.java:600)
at com.google.gdata.client.GoogleService.createRequest(GoogleService.java:486)
at com.google.gdata.client.Service.createFeedRequest(Service.java:988)
at com.google.gdata.client.Service.getFeed(Service.java:818)
at com.google.gdata.client.GoogleService.getFeed(GoogleService.java:592)
at com.google.gdata.client.Service.getFeed(Service.java:838)
at gconnect.gBlogger.getBlogId(gBlogger.java:89)
// ...

Caused by: java.security.AccessControlException: access denied
(java.util.PropertyPermission com.google.gdata.DisableCookieHandler read)
```

Eine Recherche bei Google nach „AccessControlException java.util.PropertyPermission com.google.gdata.DisableCookieHandler read“ ergab nur einen Eintrag²² (Googlehacking).

²² <http://markmail.org/message/klh3jgqxgawethh5y>

Die Fehlermeldung besagte, dass in Zeile 89 der `gBlogger.java` etwas nicht stimmt. Dort stand:

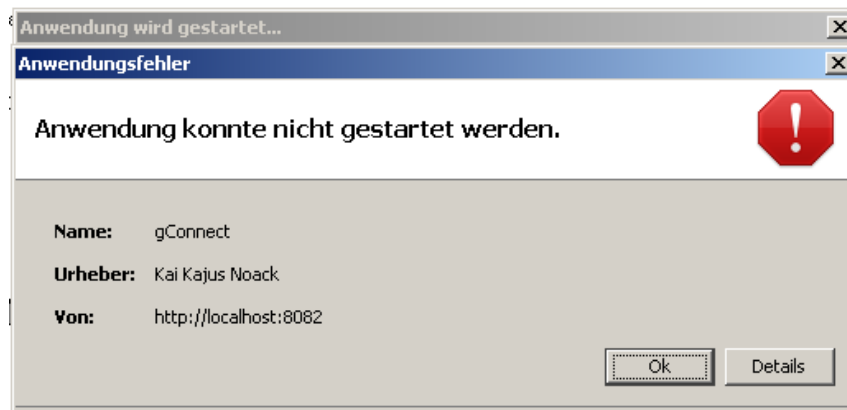
```
Feed resultFeed = myService.getFeed(feedUrl, Feed.class);
```

Der Service konnte also nicht kommunizieren, aufgrund von nicht vergebenen Rechten!

Variante 2: Eigenes Signieren von JARs

Als nächstes prüfte ich nochmals die Projekt-Einstellungen unter NetBeans und stellte fest, dass das „Self-Signed“ noch nicht aktiviert war. Ich stellte unser Projekt (unter den Properties / NetBeans) also auf „selbst-signiert“ um und kompilierte neu (Versuchte nochmals den Modus „Java Web-Start“).

Mit Start der `gConnect.jnlp` aus dem Windows Explorer heraus, startete die Applikation, doch gab die Fehlermeldung aus „Anwendung konnte nicht gestartet werden“:



In der Java-Konsole war zu lesen:

```
#### Java Web Start Error:
#### JAR-Ressourcen in JNLP-Datei sind nicht von demselben Zertifikat signiert
```

Schließlich änderte ich die Projekt-Einstellungen in NetBeans zu „**Standard Execution**“ zurück, um das Signieren der JARs selbst zu übernehmen!

Ich erstellte eine `_signJARs.bat` (die das Signieren automatisiert) mit folgendem Inhalt:

```
PATH C:\Programme\Java\jdk1.6.0_11\bin
keytool -genkey -keystore myKeystore -alias myself
keytool -selfcert -alias myself -keystore myKeystore
keytool -list -keystore myKeystore
PAUSE
jarsigner -keystore myKeystore gConnect.jar myself
jarsigner -keystore myKeystore lib\activation.jar myself
jarsigner -keystore myKeystore lib\gdata-blogger-2.0.jar myself
jarsigner -keystore myKeystore lib\gdata-calendar-2.0.jar myself
jarsigner -keystore myKeystore lib\gdata-client-1.0.jar myself
jarsigner -keystore myKeystore lib\gdata-core-1.0.jar myself
jarsigner -keystore myKeystore lib\gdata-media-1.0.jar myself
jarsigner -keystore myKeystore lib\mail.jar myself
jarsigner -keystore myKeystore lib\sample-util.jar myself
jarsigner -keystore myKeystore lib\servlet-api.jar myself
PAUSE
```

Vorgehensweise fürs Deployment:

1. "Clean and Build Project" in NetBeans
2. Kopiere die Datei `_signJARs.bat` ins Verzeichnis `\dist`
3. Starte `_signJARs.bat` und gib alle notwendigen Daten an (keyStore etc.)
4. Öffne `gConnect.jnlp` und füge hinter dem Tag `</information>` hinzu:

```
<security>
  <all-permissions/>
</security>
```

Mit dem Start der `gConnect.jnlp` erschien erneut:

```
#### Java Web Start Error:
#### JAR-Ressourcen in JNLP-Datei sind nicht von demselben Zertifikat signiert
```

An dieser Stelle unternahm ich einen letzten Versuch. Ich suchte in den einzelnen JAR-Dateien, die ich via **WinRAR öffnete**, um mir die Verzeichnisse „**META-INF**“ (in denen sich die Keys befinden) anzusehen.

5. Öffne „`activation.jar`“, gehe ins Verzeichnis „`META-INF`“ und lösche `SUN_MICR.RSA` und „`SUN_MICR.SF`“
6. Öffne „`mail.jar`“, gehe ins Verzeichnis „`META-INF`“ und lösche „`SUN_MICR.RSA`“ und „`SUN_MICR.SF`“
7. Starte die `gConnect.jnlp`

Daran lag es! Es klappte ENDLICH!

Abschließende Hinweise

1. Nach jedem neuen Kompilieren muss das Signieren erneut vollzogen werden.
2. Beim Hochladen und Testen auf einem Server muss natürlich noch darauf geachtet werden, dass die URL in der JNLP-Datei von „`localhost:8082 ...`“ zum neuen Pfad geändert wird. Einmal bei „`<jnlp spec="1.0+" codebase="`“ und bei „`<homepage href="`“.

Fazit fürs Deployment einer JavaFX-Applikation für den Desktop:

Herauszufinden, wie die Zertifizierung fremder JAR-Dateien von statten geht und die Stolpersteine zu umgehen, kostet einfach zu viel Kraft und Zeit ☹

Applikation online stellen

Um die Applikation online zum Download bereitzustellen oder von dort aus startbar zu machen, muss man alle Dateien aus dem Verzeichnis \dist\ uploaden und eine einfache HTML-Datei (wie index.html) erstellen, mit entsprechendem Link:

```
<div style="padding-top: 20px; padding-left: 40px;">
<p>Starte gConnect</p>
<p>
<a href="gConnect.jnlp"></a>
</p>
<br />
<p>
Für eine Offline-Nutzung muss die Datei
<a href="gConnect.jnlp">gConnect.jnlp</a>
auf dem Desktop gespeichert und dann gestartet werden.
</p>
```

Im Browser erscheint auf der Webseite:



Anstehende Features + Implementierungen

gCalendar

- Verwalten mehrerer Kalender
- ein tatsächliches Editieren der Events ermöglichen
- Abfrage von heutigen und zukünftigen Events

gBlogger

- Option für das (Nicht)Zulassen für Kommentare einbauen
- Upload von Bildern erlauben
- WYSIWYG Editor einbinden

Weiters

- keyListener für Enter-Taste bei Fields und Buttons
- bessere Feedbacks im Interface für die Services
- Animationseffekte wie Fading etc. über die Timeline
- gesamtes Fenster zum System-Tray minimieren (in JavaFX 1.0 noch nicht möglich)
- ...

Letzte Hinweise

Wenn direkt aus dem Browser heraus (<http://firstai.de/ext/gconnect/>) gestartet wird, so wird die Datei „lastuser.txt“ im Verzeichnis „C:\Dokumente und Einstellungen\UserFolder“ gespeichert.

Wird die gConnect.jnlp vorher auf dem Desktop gespeichert und von dort gestartet, so wird eine Desktop-Verknüpfung angelegt. Startet man diese Verknüpfung, so wird die „lastuser.txt“ ebenfalls auf dem Desktop gespeichert.

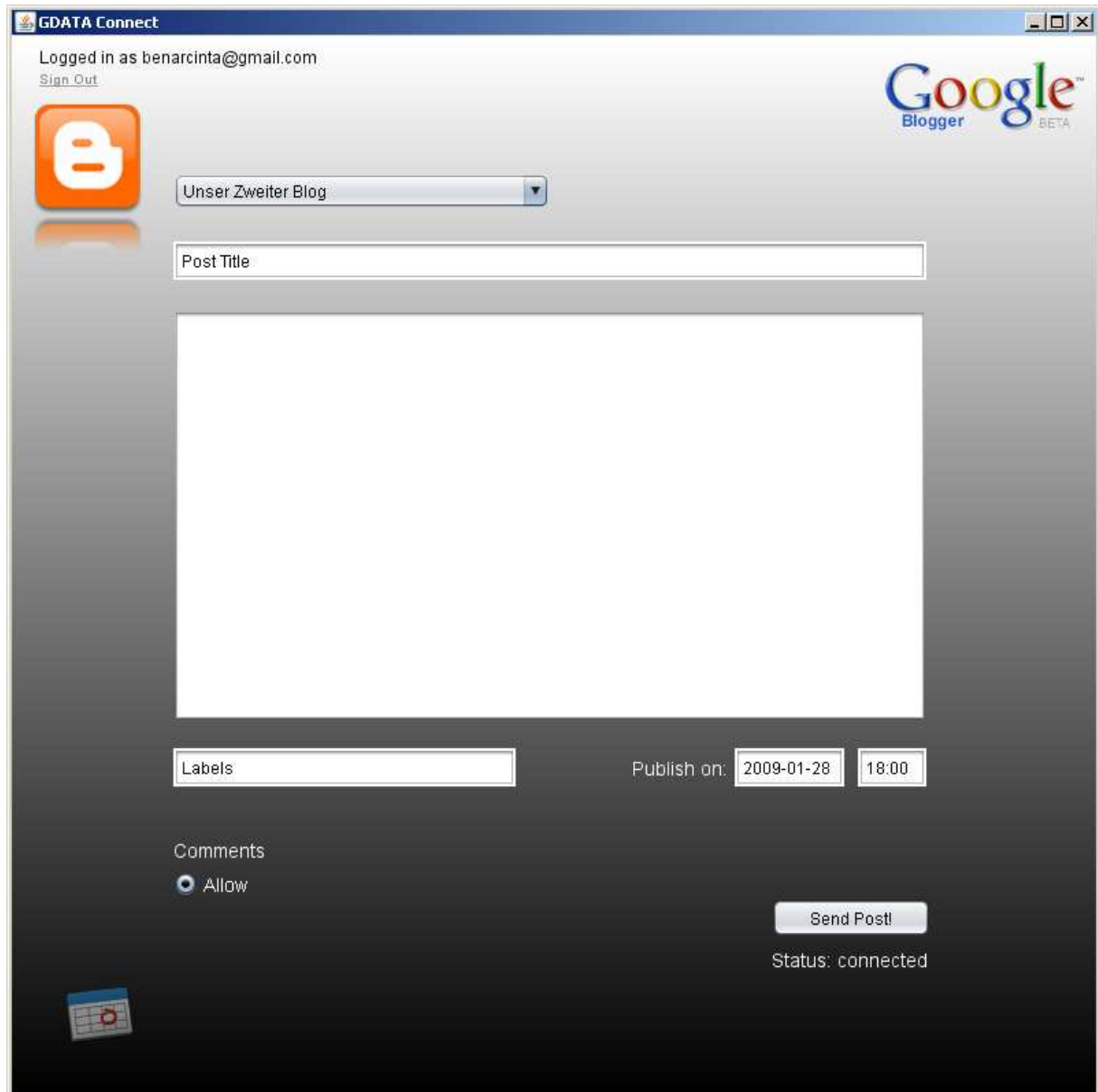
Wenn eine Firewall (wie McAfee) angeschaltet ist, kann die Applikation keine Verbindung zu Google herstellen. Die Firewall muss diese Applikation freigeben bzw. deaktiviert werden.

Kai Kajus Noack
Ende Januar 2009

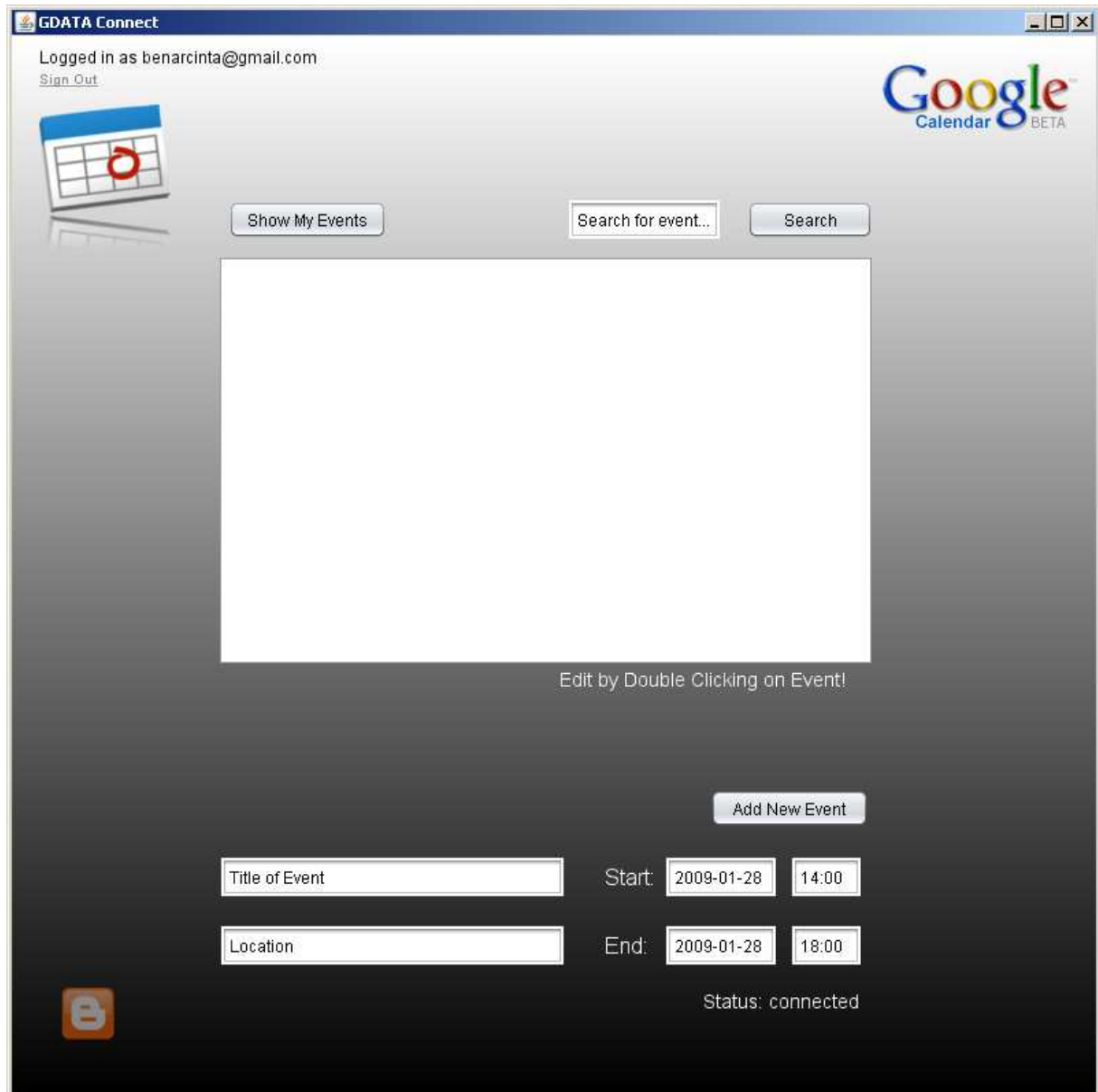
Screenshots der Applikation in Version 0.01



StartScreen



BloggerScreen



CalendarScreen

Main.fx

```

1 /*
2  * Main.fx
3  * Created in January 2009
4  * Kai Kajus Noack
5  *
6  * licence: BY-NC-SA http://creativecommons.org/licenses/by-nc-sa/3.0/de/deed.de
7  * Thank you.
8  *
9  */
10
11
12 /**
13  * @author Kai Kajus Noack
14  * @version 0.01
15  * @project gConnect
16  * @website http://media-it.blogspot.com
17  */
18
19
20 package gconnect;
21
22 import javafx.stage.Stage;           //the main window
23 import javafx.scene.Scene;           //required to display objects of Node type such as a circle and rectangle
24 import javafx.scene.paint.LinearGradient; //required to fill objects with a linear gradient
25 import javafx.scene.paint.Stop;      //required to specify colors and offset of the linear gradient
26 import javafx.scene.paint.Color;     //required to fill and stroke objects with color
27 import javafx.scene.Group;           //required to group objects to be able to operate with them as a unit
28 import javafx.scene.shape.Rectangle; //required to render a rectangle
29 import javafx.scene.effect.Reflection; //required to apply a reflection effect
30 import javafx.scene.text.Text;       //required to show text labels
31 import javafx.scene.text.TextAlignment; //required to align text of the labels
32 import javafx.scene.text.Font;
33 import javafx.ext.swing.SwingTextField; //for the textfields
34 import javafx.scene.Cursor;           //required to display a hand-cursor
35 import javafx.scene.input.MouseEvent; //required to handle mouse events
36 import javafx.scene.input.KeyEvent;  //required to handle key events
37
38 import javax.swing.JPasswordField;    //required for password asteriks
39 import javafx.ext.swing.SwingComponent; //and this one too to wrap the swing component
40
41 import javafx.ext.swing.SwingButton;   //for a very simple button
42 import javafx.ext.swing.SwingList;    //list for the results of the search query
43 import javafx.ext.swing.SwingListItem; //items (results)
44 import javafx.ext.swing.SwingToggleGroup; //for start up choice (blogger, calendar,...)
45 import javafx.ext.swing.SwingRadioButton; //same
46
47 import javafx.scene.image.*;          //for images esp. background
48
49 import java.util.Date;                 //write date into specific format

```

```

50
51 import javafx.scene.control.*;           //to get the textbox (and the vertical alignment)
52
53 import javafx.ext.swing.SwingComboBox;   //ComboBox (pull down list) to chose the blog from
54 import javafx.ext.swing.SwingComboBoxItem;
55
56 import java.io.FileOutputStream;           //create file and write to file,
57 import java.io.PrintStream;            //saving the username (not the password)
58 import java.io.IOException;            //catch possible exceptions
59 import java.io.FileInputStream;        //read file object
60 import java.io.DataInputStream;        //read username
61
62 import gconnect.gCalendar;             // our gCalendar java class
63 import gconnect.gBlogger;             // our gBlogger java class
64 import gconnect.TextArea;             // area to write our post in blogger
65
66 // import javafx.util.Sequences;
67 // import com.google.gdata.data.calendar.CalendarEntry;
68
69
70 var ourStage:Stage = Stage {
71     title: "GDATA Connect"
72     width: 800
73     height: 800
74     visible: true
75 }
76
77 var username = "";
78 var password = "";
79 var offsetTimezone = "+01:00"; // Germany .344Z
80
81 var backgroundFilling = LinearGradient {
82     startX: 0.0,
83     startY: 0.0,
84     endX: 0.0,
85     endY: 1.0,
86     proportional: true
87     stops: [
88         Stop {
89             offset: 0.0
90             color: Color.WHITE},
91         Stop {
92             offset: 1.0
93             color: Color.BLACK}
94     ]
95 }
96
97
98 var startGroup:Group;
99
100 var usernameTextfield = SwingTextField{
101     width: 200

```

```

102     foreground: Color.BLACK
103     background:Color.WHITE
104     translateX: bind ourStage.width / 2 - 100
105     translateY: 100
106     text: "Your Google-Account (Email)"
107 };
108
109 var passwordField = new JPasswordField("Password");
110 // remember the char for password
111 var theChar = passwordField.getEchoChar();
112 // disable password asteriks (at start up we want to show "Password" and not "*****")
113 passwordField.setEchoChar(0);
114
115 // necessary class that extends the abstract class
116 class swingHelper extends SwingComponent {
117     override function createJComponent() {
118         return passwordField;
119     }
120 }
121 var passwordComponent:swingHelper = swingHelper {
122     translateX: bind ourStage.width / 2 - 100
123     translateY: 140
124     width: 200
125 }
126 passwordComponent.wrap(passwordField);
127
128
129 var loginButton = Rectangle {
130     x: bind ourStage.width / 2 - 70
131     y: 200
132     width: 140
133     height: 35
134     arcWidth: 20
135     arcHeight: 55
136     stroke: Color.BLACK
137     fill: LinearGradient {
138         startX: 0.0,
139         startY: 0.0,
140         endX: 0.0,
141         endY: 1.0,
142         proportional: true
143         stops: [
144             Stop {
145                 offset: 0.0
146                 color: Color.WHITE},
147             Stop {
148                 offset: 1.0
149                 color: Color.BLACK}
150         ]
151     }
152 }
153 var loginButtonLabel = Text {

```

```
154     fill: Color.WHITE
155     font: Font {
156         size: 18
157     }
158     x: bind ourStage.width / 2 - 30
159     y: 225
160     content: "Connect"
161 }
162 var versionNumber = Text {
163     fill: Color.WHITE
164     font: Font {
165         size: 14
166     }
167     x: 30
168     y: 725
169     content: "gConnect Version 0.01\nby Kai Kajus Noack"
170 }
171
172
173 // A toggle group to switch between radio buttons
174 var choseStartWith = SwingToggleGroup{};
175 var radioButtonBlogger = SwingRadioButton{
176     text: "Blogger"
177     font: Font {
178         size: 14
179     }
180     toggleGroup: choseStartWith
181     translateX: bind ourStage.width / 2 - 40
182     translateY: 300
183     selected: true
184 };
185 var radioButtonCalendar = SwingRadioButton{
186     text: "Calendar"
187     font: Font {
188         size: 14
189     }
190     toggleGroup: choseStartWith
191     translateX: bind ourStage.width / 2 - 40
192     translateY: radioButtonBlogger.translateY + 25
193     selected: false
194 };
195
196
197 var imageBackground = ImageView {
198     x: bind ourStage.width / 2;
199     y: 520;
200     image: Image {
201         url: "{__DIR__}images/background_by_kaikajusnoack.png"
202     }
203     effect: Reflection {
204         fraction: 0.9
205         topOpacity: 0.5
```

```
206         topOffset: 2.5
207     }
208 }
209
210 var loggedInText:Text = Text {
211     fill: Color.BLACK
212     font: Font {
213         size: 12
214     }
215     x: 20
216     y: 20
217     content: ""
218 }
219 var signoutText:Text = Text {
220     visible: false
221     fill: Color.GREY
222     font: Font {
223         size: 10
224     }
225     underline: true
226     x: 20
227     y: 35
228     content: "Sign Out"
229 }
230
231
232 var calendarGroup:Group;
233
234 var imageCalendarIcon = ImageView {
235     x: 20;
236     y: 50;
237     image: Image {
238         url: "{__DIR__}images/google_calendar_fluid_by_Malabooboo.png"
239     }
240 }
241 var imageCalendarGoogleLogo = ImageView {
242     x: 635;
243     y: 15;
244     image: Image {
245         url: "{__DIR__}images/google_calendar.png"
246     }
247 }
248
249 // hold the query results of our search
250 var resultList = SwingList {
251     name: "Calendars"
252     width: 480 // thewidth
253     height: 300
254     translateX: ourStage.width / 2 - 250
255     translateY: 160
256 }
257
```

```
258 var buttonShowEvents = SwingButton{
259     text: "Show My Events"
260     translateX: bind fieldEventAnchorX + 40 // stageWidth / 2 + 110
261     translateY: 120
262 };
263
264 var buttonSearch = SwingButton{
265     text: "Search"
266     translateX: (resultsList.translateX + resultsList.width - buttonShowEvents.width * 0.8)
267     translateY: bind buttonShowEvents.translateY
268     width: bind buttonShowEvents.width * 0.8
269 };
270
271 var searchTextfield = SwingTextField{
272     width: 110
273     foreground: Color.BLACK
274     background:Color.WHITE
275     translateX: buttonSearch.translateX - 130
276     translateY: buttonShowEvents.translateY
277     text: "Search for event..."
278 };
279
280 var buttonAddEvent = SwingButton{
281     text: "Add New Event"
282     translateX: ourStage.width / 2 + 110
283     translateY: bind fieldEventAnchorY - 50
284     width: bind buttonShowEvents.width
285 };
286
287 var hintEditText = Text {
288     fill: Color.WHITE
289     font: Font {
290         size: 14
291     }
292     x: ourStage.width / 2
293     y: 475
294     content: "Edit by Double Clicking on Event!"
295 }
296
297 // array to hold results for our search as text
298 var recentResults: String[];
299 var entriesFound:Integer = 0;
300
301 // create a calendar object
302 var calendar:gCalendar = new gCalendar();
303
304 var calendarActivated:Boolean = false;
305
306
307 /***** Fields for Adding a New Entry *****/
308 var fieldEventAnchorX = 118;
309 var fieldEventAnchorY = 600;
```

```
310 var fieldEventTitle = SwingTextField{
311     width: 250
312     translateX: bind fieldEventAnchorX + 35
313     translateY: bind fieldEventAnchorY
314     text: "Title of Event"
315 };
316 var fieldEventLocation = SwingTextField{
317     width: 250
318     translateX: bind fieldEventAnchorX + 35
319     translateY: bind fieldEventAnchorY + 50
320     text: "Location"
321 };
322 var fieldEventStartDate = SwingTextField{
323     width: 80
324     translateX: bind fieldEventAnchorX + 360
325     translateY: bind fieldEventAnchorY
326     text: calendar.getToday(); // "yyyy-mm-dd"
327 };
328 var fieldEventEndDate = SwingTextField{
329     width: 80
330     translateX: bind fieldEventAnchorX + 360
331     translateY: bind fieldEventAnchorY + 50
332     text: calendar.getToday(); // "2009-01-16"
333 };
334 var fieldEventStartTime = SwingTextField{
335     width: 50
336     translateX: bind fieldEventAnchorX + 452
337     translateY: bind fieldEventAnchorY
338     text: "14:00"
339 };
340 var fieldEventEndTime = SwingTextField{
341     width: 50
342     translateX: bind fieldEventAnchorX + 452
343     translateY: bind fieldEventAnchorY + 50
344     text: "18:00"
345 };
346 var labelEventStart = Text {
347     fill: Color.WHITE
348     font: Font {
349         size: 16
350     }
351     x: bind fieldEventAnchorX + 315
352     y: bind fieldEventAnchorY + 20
353     content: "Start:"
354 }
355 var labelEventEnd = Text {
356     fill: Color.WHITE
357     font: Font {
358         size: 16
359     }
360     x: bind fieldEventAnchorX + 315
361     y: bind fieldEventAnchorY + 70
```

```
362     content: "End:"
363 }
364
365
366 /*****  BLOGGER  *****/
367
368 // array to hold blogs of the user
369 var blogList: String[];
370 var blogsFound: Integer = 0;
371
372 var blogservice:gBlogger = new gBlogger();
373
374 var blogActivated:Boolean = false;
375
376
377 var bloggerGroup:Group;
378
379 var imageBloggerIcon = ImageView {
380     x: 16;
381     y: 50;
382     image: Image {
383         url: "{__DIR__}images/google_blogger_by_BrunoAbreu.png"
384     }
385 }
386 var imageBloggerGoogleLogo = ImageView {
387     x: 635;
388     y: 15;
389     image: Image {
390         url: "{__DIR__}images/google_blogger.png"
391     }
392 }
393
394 /*****  Fields for Adding a New Entry to  BLOGGER  *****/
395 var fieldBlogAnchorX = 118;
396 var fieldBlogAnchorY = 150;
397
398 // a combo box can hold items
399 var comboboxBlogs = SwingComboBox{
400     translateX: bind fieldBlogAnchorX
401     translateY: 100
402     width: 275
403     /* items:[
404         SwingComboBoxItem{ text: "1st Blog" },
405         SwingComboBoxItem{ text: "2nd Blog" },
406     ]*/
407     selectedIndex: 0
408 };
409
410 var fieldBlogTitle = SwingTextField{
411     width: 550
412     translateX: bind fieldBlogAnchorX
413     translateY: bind fieldBlogAnchorY
```

```

414     text: "Post Title"
415 };
416 var fieldBlogContent = TextArea {
417     width: 550
418     height: 300
419     translateX: bind fieldBlogAnchorX
420     translateY: bind fieldBlogAnchorY + 50
421     text: ""
422 };
423 var fieldBlogLabels = SwingTextField{
424     width: 250
425     translateX: bind fieldBlogAnchorX
426     translateY: bind fieldBlogAnchorY + 370
427     text: "Labels"
428 };
429     var fieldBlogPostDate = SwingTextField{ //Post-DATE 19.01.09
430     width: 80
431     translateX: bind fieldBlogAnchorX + 410
432     translateY: bind fieldBlogAnchorY + 370
433     text: calendar.getToday(); // "yyyy-mm-dd"
434 };
435     var fieldBlogPostTime = SwingTextField{ //Post-TIME 05:40
436     width: 50
437     translateX: bind fieldBlogAnchorX + 500
438     translateY: bind fieldBlogAnchorY + 370
439     text: "18:00"
440 };
441 var feedbackBloggerText = Text {
442     fill: Color.WHITE
443     font: Font {
444         size: 14
445     }
446     textAlignment: TextAlignment.RIGHT;
447     x: bind buttonAddPost.translateX;
448     y: bind buttonAddPost.translateY + 50;
449     content: "Status: connected"
450 }
451 var feedbackCalendarText = Text {
452     fill: Color.WHITE
453     font: Font {
454         size: 14
455     }
456     textAlignment: TextAlignment.RIGHT;
457     x: bind buttonAddEvent.translateX - 5;
458     y: bind buttonAddEvent.translateY + 160;
459     content: "Status: connected"
460 }
461
462 // allow or disallow Reader Comments (logic not yet implemented)
463 var bloggerComments = SwingToggleGroup{};
464 var bloggerCommentsAllowed = SwingRadioButton{
465     toggleGroup: bloggerComments

```

```
466     foreground: Color.WHITE
467     text: "Allow"
468     font: Font {
469         size: 14
470     }
471     translateX: bind fieldBlogAnchorX
472     translateY: bind fieldBlogAnchorY + 460
473     selected: true
474 };
475 var bloggerCommentsNotAllowed = SwingRadioButton{
476     toggleGroup: bloggerComments
477     foreground: Color.WHITE
478     text: "Don't allow"
479     font: Font {
480         size: 14
481     }
482     translateX: bind bloggerCommentsAllowed.translateX
483     translateY: bind bloggerCommentsAllowed.translateY + 25
484     selected: false
485 };
486
487 var buttonAddPost = SwingButton{
488     text: "Send Post!"
489     translateX: bind ourStage.width / 2 + 155
490     translateY: bind fieldBlogAnchorY + 480
491     width: bind buttonShowEvents.width
492 };
493
494 var blogLabelComment = Text {
495     fill: Color.WHITE
496     font: Font {
497         size: 14
498     }
499     x: bind bloggerCommentsAllowed.translateX
500     y: bind bloggerCommentsAllowed.translateY - 10
501     content: "Comments"
502 }
503 var blogLabelPublishOn = Text {
504     fill: Color.WHITE
505     font: Font {
506         size: 14
507     }
508     x: bind fieldBlogPostDate.translateX - 75
509     y: bind fieldBlogPostDate.translateY + 20
510     content: "Publish on:"
511 }
512
513 // SWAPPERS
514 var imageBloggerIconSwap = ImageView {
515     x: 16;
516     y: bind ourStage.height - 125;
517     opacity: 0.5
```

```

518     scaleX: 0.5
519     scaleY: 0.5
520     image: Image {
521         url: "{__DIR__}images/google_blogger_by_BrunoAbreu.png" //flickr.com
522     }
523 }
524 }
525 var imageCalendarIconSwap = ImageView {
526     x: 16;
527     y: bind imageBloggerIconSwap.y
528     opacity: 0.5
529     scaleX: 0.5
530     scaleY: 0.5
531     image: Image {
532         url: "{__DIR__}images/google_calendar_fluid_by_Malabooboo.png" //flickr.com
533     }
534 }
535 }
536 }
537 function findUsername():String {
538     try {
539         // open the file
540         var fstream:FileInputStream = new FileInputStream("lastuser.txt");
541         // convert our input stream to a DataInputStream
542         var readIn:DataInputStream = new DataInputStream(fstream);
543         var readUsername = readIn.readLine();
544         readIn.close();
545         // if no data is present (file is empty)
546         if (readUsername.equals("") or readUsername == null) {
547             readUsername = "Your Google-Account (Email)";
548         }
549         else {
550             // if user has been found, focus the password field
551             passwordComponent.requestFocus();
552             // and delete the text in the field
553             passwordField.setText("");
554             // set up password char *
555             passwordField.setEchoChar(theChar);
556         }
557         // return the user name fetched
558         return readUsername;
559     }
560     catch (ex:IOException) {
561         println ("Error reading file: {ex}");
562     };
563     // if no file has been found
564     return "Your Google-Account (Email)"
565 }
566 }
567 function saveUsername(user:String):Void {
568     // Create a new file output stream connected to "myfile.txt"
569     var outputStream:FileOutputStream;

```

```

570 var p:PrintStream; // declare a print stream object
571 try {
572     outputStream = new FileOutputStream("lastuser.txt"); // file output object
573     p = new PrintStream(outputStream);
574     // write user to file
575     p.println (user);
576     p.close();
577 } catch (ex:IOException) {
578     println ("Error writing to file: {ex}");
579 };
580 }
581
582
583 /***** SCENE OF THE STAGE *****/
584 var ourScreen:Scene = Scene {
585     fill: backgroundFilling
586     width: 800;
587     //objects that appear on the scene
588     content: [
589         /***** STARTSCREEN *****/
590         loggedInText,
591         startGroup = Group {
592             visible: true; // TRUE
593             content: [
594                 Group {
595                     content: [
596                         loginButton,
597                         loginButtonLabel
598                     ]
599                     effect: Reflection {
600                         fraction: 0.9
601                         topOpacity: 0.5
602                         topOffset: 2.5
603                     }
604                     cursor: Cursor.HAND
605                     onMouseEntered: function(evt: MouseEvent):Void {
606                         loginButton.scaleX = 1.2;
607                         loginButtonLabel.underline = true;
608                     }
609                     onMouseExited: function(evt: MouseEvent):Void {
610                         loginButton.scaleX = 1.0;
611                         loginButtonLabel.underline = false;
612                     }
613                     onMouseClicked: function(evt: MouseEvent):Void {
614                         username = usernameTextfield.text;
615                         password = passwordField.getText();
616                         // save user name to file for next start of application
617                         saveUsername(username);
618                         if ( (username != "") and (password != "") and (password != "Password") and (username != "Your Google-
619                             Account (Email)") ) {
620                             // connect to google
621                             var connected:Boolean = false;

```

```

621
622         // BLOGGER chosen at STARTUP
623         if(choseStartWith.getSelection().text == "Blogger") {
624             connected = doBlogger();
625         }
626         // CALENDER chosen at STARTUP
627         else if (choseStartWith.getSelection().text == "Calendar") {
628             connected = doCalendar();
629         }
630
631         if (connected) {
632             startGroup.visible = false;
633             // show login name
634             loggedInText.content = "Logged in as {username}";
635             // show option to logout
636             signoutText.visible = true;
637         }
638         else {
639             // login failed
640             loggedInText.content = "Your Login failed !!!"
641         }
642     }
643     else {
644         println("# NO CONNECTION ESTABLISHED DUE TO WRONG LOGIN DATA!")
645     }
646 }
647 },
648 Group {
649     content: [
650         usernameTextfield
651     ]
652     cursor: Cursor.TEXT
653     onMouseClicked: function(evt: MouseEvent):Void {
654         usernameTextfield.text = "";
655         // if there was a wrong login beforehand, hide this label again
656         loggedInText.content = "";
657     }
658 },
659 Group {
660     content: [
661         passwordComponent
662     ]
663     cursor: Cursor.TEXT
664     onMouseClicked: function(evt: MouseEvent):Void {
665         // delete "Password" from textfield
666         passwordField.setText("");
667         // set * again for entered password chars
668         passwordField.setEchoChar(theChar);
669         // if there was a wrong login beforehand, hide this label again
670         loggedInText.content = "";
671     }
672 },

```

```

673     Group {
674         content: [
675             radioButtonBlogger,
676             radioButtonCalendar
677         ]
678         cursor: Cursor.HAND
679         onMouseClicked: function(evt: MouseEvent):Void {
680             // set * again for entered password chars (actually not necessary)
681             passwordField.setEchoChar(theChar);
682         }
683     },
684     imageBackground,
685     versionNumber
686 ]
687 }
688
689 /***** CALENDAR *****/
690 calendarGroup = Group {
691     visible: false;
692     content: [
693         imageCalendarGoogleLogo,
694         Group {
695             content: [
696                 imageCalendarIcon
697             ]
698             effect: Reflection {
699                 fraction: 0.3
700                 topOpacity: 0.8
701                 topOffset: 2.5
702             }
703         },
704         Group {
705             content: [
706                 searchTextfield
707             ]
708             cursor: Cursor.TEXT
709             onMouseClicked: function(evt: MouseEvent):Void {
710                 searchTextfield.text = "";
711             }
712         },
713         Group {
714             content: [
715                 buttonSearch
716             ]
717             cursor: Cursor.HAND
718             onMouseClicked: function(evt: MouseEvent):Void {
719                 // clear SwingList
720                 resultsList.items = null;
721                 // clear String Array
722                 recentResults = null;
723             }
724             // avoid empty search

```

```

725         if (searchTextfield.text != "") {
726             var searchTerm = searchTextfield.text;
727             entriesFound = calendar.doQuery(searchTerm,
728                 "http://www.google.com/calendar/feeds/{username}/private/full", username, password);
729             if ( entriesFound > 0 ) {
730                 for (i in [0..(entriesFound - 1)]) {
731                     // fetch calendar entries from service
732                     insert calendar.getQueryResults(i) into recentResults;
733                     // create SwingListItem that holds the String
734                     var item = SwingListItem {
735                         text: recentResults[i] // listing of events
736                     };
737                     // and insert it into the list on the screen
738                     insert item into resultsList.items;
739                     // println("# array-size is {sizeof recentResults}");
740                 }
741             } else {
742                 // if nothing found, show negative feedback
743                 var item = SwingListItem {
744                     text: "Sorry, no Entries found!" // here listing of events
745                 };
746                 insert item into resultsList.items;
747             }
748         }
749     } else {
750         // if searchfield is empty, show negative feedback
751         var item = SwingListItem {
752             text: "Sorry, no Entries found!" // here listing of events
753         };
754         insert item into resultsList.items;
755     }
756 }
757 },
758 Group {
759     content: [
760         buttonShowEvents
761     ]
762     cursor: Cursor.HAND
763     onMouseClicked: function(evt: MouseEvent):Void {
764         // clear SwingList
765         resultsList.items = null;
766         // clear String Array
767         recentResults = null;
768         // get all entries by sending an empty string!
769         entriesFound = calendar.doQuery("", "http://www.google.com/calendar/feeds/{username}/private/full",
770             username, password);
771         if ( entriesFound > 0 ) {
772             // returns a String Array
773             var entryListing = calendar.getQueryResultsArray();
774             for (i in [0..(entryListing.size() - 1)]) {

```

```

775         insert entryListing[i] into recentResults;
776         // insert into list
777         var item = SwingListItem {
778             text: entryListing[i].toString(); // listing of events
779         };
780         insert item into resultsList.items;
781     }
782 }
783 else {
784     // if nothing found, show negative feedback
785     var item = SwingListItem {
786         text: "Sorry, no Entries found!"
787     };
788     insert item into resultsList.items;
789 }
790 }
791 },
792 Group {
793     content: [
794         fieldEventStartDate,
795         fieldEventStartTime,
796         fieldEventEndDate,
797         fieldEventEndTime,
798         labelEventStart,
799         labelEventEnd
800     ]
801     cursor: Cursor.TEXT
802     onMouseClicked: function(evt: MouseEvent):Void {
803     }
804 },
805 Group {
806     content: [
807         fieldEventTitle
808     ]
809     cursor: Cursor.TEXT
810     onMouseClicked: function(evt: MouseEvent):Void {
811         fieldEventTitle.text = "";
812     }
813 },
814 Group {
815     content: [
816         fieldEventLocation
817     ]
818     cursor: Cursor.TEXT
819     onMouseClicked: function(evt: MouseEvent):Void {
820         fieldEventLocation.text = "";
821     }
822 },
823 Group {
824     content: [
825         buttonAddEvent
826     ]

```

```

827     cursor: Cursor.HAND
828     onMouseClicked: function(evt: MouseEvent):Void {
829         // sends event to Calendar in Google
830
831         // date format is ("yyyy-MM-ddTHH:mm:ss-SSSZ")
832         // e.g. "2009-01-15T17:00:00-08:00"
833         var eventTitle = fieldEventTitle.text;
834         var eventLocation = fieldEventLocation.text;
835
836         var eventStartDate = fieldEventStartDate.text; // "2009-01-15"
837         var eventStartTime = fieldEventStartTime.text; // "15:00"
838         var eventStart = "{eventStartDate}T{eventStartTime}:00{offsetTimezone}";
839
840         var eventEndDate = fieldEventEndDate.text; // "2009-01-15"
841         var eventEndTime = fieldEventEndTime.text; // "18:00"
842         var eventEnd = "{eventEndDate}T{eventEndTime}:00{offsetTimezone}";
843
844         // send Event to google, returns true if without any errors, i.e. accepted
845         var accepted:Boolean =
            calendar.setEvent("http://www.google.com/calendar/feeds/{username}/private/full", eventTitle,
                eventLocation, eventStart, eventEnd);
846         if (accepted == true) {
847             feedbackCalendarText.content = "Status: Event added!"
848         }
849         else {
850             feedbackCalendarText.content = "Status: Error!"
851         }
852     }
853 },
854 Group {
855     content: [
856         resultsList
857     ]
858     onMouseClicked: function(e: MouseEvent):Void {
859         // implementation to receive a mouse double-click
860         if(e.clickCount >= 2) {
861             resultsList.selectedItem.text = "editable";
862         }
863     }
864 },
865 hintEditText,
866 Group {
867     content: [
868         imageBloggerIconSwap
869     ]
870     cursor: Cursor.HAND
871     onMouseEntered: function(evt: MouseEvent):Void {
872         imageBloggerIconSwap.opacity = 1.0;
873         imageBloggerIconSwap.scaleX =
874         imageBloggerIconSwap.scaleY = 0.7;
875     }
876     onMouseExited: function(evt: MouseEvent):Void {

```

```

877         imageBloggerIconSwap.opacity = 0.5;
878         imageBloggerIconSwap.scaleX =
879         imageBloggerIconSwap.scaleY = 0.5;
880     }
881     onMouseClicked: function(evt: MouseEvent):Void {
882         // establish service, if not yet done
883         if (blogActivated == false) {
884             doBlogger();
885             blogActivated = true;
886         }
887         calendarGroup.visible = false;
888         bloggerGroup.visible = true;
889     }
890 },
891     feedbackCalendarText
892 ]
893 }
894
895 /***** BLOGGER *****/
896 bloggerGroup = Group {
897     visible: false;
898     content: [
899         imageBloggerGoogleLogo,
900         comboboxBlogs,
901         Group {
902             content: [
903                 imageBloggerIcon
904             ]
905             effect: Reflection {
906                 fraction: 0.3
907                 topOpacity: 0.8
908                 topOffset: 2.5
909             }
910         },
911         Group {
912             content: [
913                 fieldBlogTitle
914             ]
915             cursor: Cursor.TEXT
916             onMouseClicked: function(evt: MouseEvent):Void {
917                 fieldBlogTitle.text = "";
918             }
919         },
920         Group {
921             content: [
922                 fieldBlogContent,
923                 fieldBlogPostDate, //Post-DATE 19.01.09
924                 fieldBlogPostTime //Post-TIME 05:40
925             ]
926             cursor: Cursor.TEXT
927             onMouseClicked: function(evt: MouseEvent):Void {
928

```

```

929     },
930     Group {
931         content: [
932             fieldBlogLabels
933         ]
934         cursor: Cursor.TEXT
935         onMouseClicked: function(evt: MouseEvent):Void {
936             // delete content if clicked on field
937             fieldBlogLabels.text = "";
938         }
939     },
940     Group {
941         content: [
942             buttonAddPost
943         ]
944         cursor: Cursor.HAND
945         onMouseClicked: function(evt: MouseEvent):Void {
946             // receive blog chosen from blog list in interface
947             var activeBlogId = blogList[comboboxBlogs.selectedIndex];
948
949             // date format is ("yyyy-MM-ddTHH:mm:ss-SSSZ")
950             var publishPostOn = "{fieldBlogPostDate.text}T{fieldBlogPostTime.text}:00{offsetTimezone}";
951             // create our post and returns true if posted without any errors
952             // last parameter is false = 'no draft'
953             var posted:Boolean = blogservice.createPost(fieldBlogTitle.text, fieldBlogContent.text,
954                 fieldBlogLabels.text, ( if(bloggerCommentsAllowed.selected) true else false ),
955                 publishPostOn, blogservice.username, username, activeBlogId, false);
956             // if accepted
957             if (posted == true) {
958                 feedbackBloggerText.content = "Status: Entry added!"
959             }
960             else {
961                 feedbackBloggerText.content = "Status: Error!"
962             }
963         }
964     }
965     Group {
966         content: [
967             bloggerCommentsAllowed
968             // bloggerCommentsNotAllowed // (implementation not yet done)
969         ]
970         cursor: Cursor.HAND
971     },
972     blogLabelComment,
973     blogLabelPublishOn,
974     Group {
975         content: [
976             imageCalendarIconSwap
977         ]
978         cursor: Cursor.HAND
979         onMouseEntered: function(evt: MouseEvent):Void {
980             imageCalendarIconSwap.opacity = 1.0;

```

```

981         imageCalendarIconSwap.scaleX =
982         imageCalendarIconSwap.scaleY = 0.7;
983     }
984     onMouseExited: function(evt: MouseEvent):Void {
985         imageCalendarIconSwap.opacity = 0.5;
986         imageCalendarIconSwap.scaleX =
987         imageCalendarIconSwap.scaleY = 0.5;
988     }
989     onMouseClicked: function(evt: MouseEvent):Void {
990         if (calendarActivated == false) {
991             calendar.authenticate(username,password);
992             calendarActivated = true;
993         }
994         bloggerGroup.visible = false;
995         calendarGroup.visible = true;
996     }
997     },
998     feedbackBloggerText
999 ]
1000 },
1001 Group {
1002     content: [
1003         signoutText
1004     ]
1005     cursor: Cursor.HAND
1006     onMouseClicked: function(evt: MouseEvent):Void {
1007         java.lang.System.exit(0);
1008     }
1009 }
1010 ]
1011 }
1012 }
1013
1014
1015 function doBlogger():Boolean {
1016     if ( blogservice.authenticate(username,password) ) {
1017         blogActivated = true;
1018         bloggerGroup.visible = true;
1019         // swapper to Calendar
1020         imageCalendarIconSwap.visible = true;
1021         // get all blogs of the user and fill combobox
1022         blogsFound = blogservice.howmanyUserBlogs();
1023         println("Total Blogs: {blogsFound}");
1024         if ( blogsFound > 0 ) {
1025             for (i in [0..(blogsFound - 1)]) {
1026                 // add blog name into array, remember position for later posting
1027                 insert blogservice.getUserBlogId(i, false) into blogList;
1028                 // create item named with current blog
1029                 var item = SwingComboBoxItem {
1030                     text: blogservice.getUserBlogId(i, true)
1031                 };
1032                 // insert name of blog into combobox

```

```

1033         insert item into comboboxBlogs.items;
1034     }
1035     // necessary to focus a newly inserted blog in the list
1036     comboboxBlogs.selectedItem = comboboxBlogs.items[blogsFound - 1];
1037 }
1038 else {
1039     // no blogs found
1040     var item = SwingComboBoxItem {
1041         text: "Sorry, no Blog found!";
1042     };
1043     insert item into comboboxBlogs.items;
1044 }
1045 return true;
1046 }
1047 else {
1048     // // authentication failed (login wrong, connection not working or another error)
1049     return false;
1050 }
1051 }
1052
1053 function doCalendar():Boolean {
1054     if( calendar.authenticate(username,password) ) {
1055         calendarActivated = true;
1056         calendarGroup.visible = true;
1057         // swapper to Blogger
1058         imageBloggerIconSwap.visible = true;
1059         return true;
1060     }
1061     else {
1062         // authentication failed
1063         return false;
1064     }
1065 }
1066
1067
1068
1069 /**** INIT ****/
1070 usernameTextfield.text = findUsername();
1071
1072 /**** Helper for Password Field ****/
1073 import javafx.animation.Timeline;
1074 import javafx.animation.KeyFrame;
1075 import javafx.animation.Interpolator;
1076
1077 // if username has not been specified yet then init password field after 5 sec
1078 if (usernameTextfield.text == "Your Google-Account (Email)") {
1079     Timeline {
1080         keyFrames: [
1081             KeyFrame {
1082                 time: 0s
1083             }
1084             KeyFrame {

```

```
1085         time: 5s
1086         action: function():Void {
1087             passwordField.setText("");
1088             // set up password char *
1089             passwordField.setEchoChar(theChar);
1090         }
1091     }
1092 }
1093 }.play();
1094 }
1095
1096 // finally add the scene to our stage!
1097 ourStage.scene = ourScreen;
1098
1099
1100 /*
1101  * Credits Graphics:
1102  *
1103  * Google icons by Google
1104  * Calendar icon by Malabooboo
1105  * Blogger icon by BrunoAbreu
1106  * Fish graphics by Kai Kajus Noack
1107  *
1108  * @ all: thanks for sharing your great work!
1109  *
1110  */
1111
1112
```

gCalendar.java

```
1 /*
2  * gCalendar.java
3  * Created January 2009
4  *
5  * @author: Kai Kajus Noack
6  * @project: gConnect
7  * @version: 0.01
8  * @release: 2008-01-28
9  * @website: http://media-it.blogspot.com
10 *
11 * Some code snippets are taken from Google's Samples:
12 * http://code.google.com/apis/gdata/articles/java_client_lib.html
13 *
14 */
15
16 package gconnect;
17
18 import com.google.gdata.client.*;
19 import com.google.gdata.client.calendar.*;
20 import com.google.gdata.data.*;
21 import com.google.gdata.data.calendar.*;
22 import com.google.gdata.data.extensions.*;
23 import com.google.gdata.util.*;
24
25 import java.net.*;
26 import java.io.*;
27 import java.util.Locale;
28 import java.util.logging.Level;
29 import java.util.logging.Logger;
30
31
32 public class gCalendar {
33
34     String username = "";
35     String password = "";
36
37     // create a calendar service
38     CalendarService myService = new CalendarService("gCalendar Service 0.1");
39
40     Boolean authenticate(String user, String pwd) {
41         System.out.println("java: TRYING TO AUTHENTICATE");
42         this.username = user;
43         this.password = pwd;
44         try {
45             // Authenticate using ClientLogin
46             myService.setUserCredentials(user, pwd);
47         } catch (AuthenticationException ex) {
48             Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
49             System.out.println("java: CANNOT ESTABLISH CONNECTION!");
50         }
51     }
52 }
```

```

50         return false;
51     }
52     System.out.println("java: CONNECTED");
53     return true;
54 }
55
56 // returns in format: 2009-01-15
57 public String getToday() {
58     String today = DateTime.now().toUiString(); // "2009-01-19T11:21:26.256Z"
59     today = today.substring(0, 10);
60     return today;
61 }
62
63 // new calendar event entry
64 public Boolean setEvent(String feedAdress, String eventTitle, String eventLocation, String eventStart, String eventEnd) {
65     System.out.println("Setting event: " + eventTitle + eventLocation + eventStart + eventEnd);
66     URL postURL = null;
67     try {
68         postURL = new URL(feedAdress);
69     } catch (MalformedURLException ex) {
70         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
71     }
72
73     CalendarEventEntry myEvent = new CalendarEventEntry();
74
75     // set title and description
76     myEvent.setTitle(new PlainTextConstruct(eventTitle));
77
78     // create DateTime events and create a When object to hold them
79     // then add the When event to the event
80     DateTime startTime = DateTime.parseDateTime(eventStart); // "2009-01-15T15:00:00-08:00"
81     DateTime endTime = DateTime.parseDateTime(eventEnd); // "2009-01-15T17:00:00-08:00"
82     When eventTimes = new When();
83     eventTimes.setStartTime(startTime);
84     eventTimes.setEndTime(endTime);
85     myEvent.addTime(eventTimes);
86     // add a Where object to add the location
87     Where eventPlace = new Where();
88     eventPlace.setValueString(eventLocation);
89     myEvent.addLocation(eventPlace);
90
91     try {
92         // POST the request and receive the response
93         myService.insert(postURL, myEvent);
94     } catch (IOException ex) {
95         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
96         return false;
97     } catch (ServiceException ex) {
98         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
99         return false;
100    }
101    System.out.println("java: ENTRY ADDED to Calendar!");

```

```

102     // return true if posting worked without any errors
103     return true;
104 }
105
106 String resultString = "";
107 CalendarEventFeed myResultsFeed = null;
108 int entriesFound = 0;
109
110 // do a query to the calendar service
111 public Integer doQuery(String searchTerm, String feedAdress, String user, String pwd) {
112
113     URL feedURL = null;
114     try {
115         // request the list of all calendars from the authenticated user
116         feedURL = new URL(feedAdress);
117     } catch (MalformedURLException ex) {
118         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
119     }
120     // create a new query object and set the parameters
121     Query myQuery = new Query(feedURL);
122     myQuery.setFullTextQuery(searchTerm);
123
124     // send the request with the built query URL
125     try {
126         myResultsFeed = myService.query(myQuery, CalendarEventFeed.class);
127     } catch (IOException ex) {
128         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
129     } catch (ServiceException ex) {
130         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
131     }
132
133     entriesFound = myResultsFeed.getEntries().size();
134     System.out.println("java: FEEDS WITH " + searchTerm + " FOUND: " + entriesFound);
135     return entriesFound;
136 }
137
138 // return the whole event data (time | title | location) at given position
139 public String getQueryResults(int elementAt) {
140     // get event time + title, elements of results feed
141     resultString = myResultsFeed.getEntries().get(elementAt).getTimes().get(0).getStartTime().toUiString() + " | " +
142         myResultsFeed.getEntries().get(elementAt).getTitle().getPlainText();
143     // try to receive location if set
144     String location;
145     try {
146         location = myResultsFeed.getEntries().get(elementAt).getLocations().get(0).getValueString().toString();
147     } catch (Exception e) {
148         System.out.println("java: INVALID Location specified!!");
149         return "INCOMPLETE ENTRY IS: " + myResultsFeed.getEntries().get(elementAt).getTitle().getPlainText();
150     }
151     // if location is not available add ---
152     if (location.equals("")) {
153         location = "---";

```

```

154     }
155     resultString += " | " + location;
156     System.out.println("java: " + resultString);
157     return resultString;
158 }
159
160 // return all the events found as String Array
161 public String[] getQueryResultsArray() {
162     // create a new array with number of feeds found
163     String[] strArray = new String[myResultsFeed.getEntries().size()];
164     // iterate of all events and save data in string array
165     for (int i = 0; i < myResultsFeed.getEntries().size(); i++) {
166         strArray[i] = myResultsFeed.getEntries().get(i).getTimes().get(0).getStartTime().toUiString() + " | " +
167             myResultsFeed.getEntries().get(i).getTitle().getPlainText();
168         // care for locations
169         String location = null;
170         try {
171             location = myResultsFeed.getEntries().get(i).getLocations().get(0).getValueString().toString();
172         } catch (Exception e) {
173             System.out.println("java: INVALID Location specified!!");
174         }
175         if (location.equals("")) {
176             location = "---";
177         }
178         // write to array
179         strArray[i] += " | " + location;
180     }
181
182     // sort the string array!
183     java.util.Arrays.sort(strArray);
184     // to set special Locale chose: (strArray, java.text.Collator.getInstance(Locale.ENGLISH));
185
186     // return all the data fetched
187     return strArray;
188 }
189
190
191 // method not yet used in the interface, we are handling only one calendar for version 0.01
192 public void getCalendars() {
193     URL feedUrl = null;
194     try {
195         // request the list of all calendars from the authenticated user
196         feedUrl = new URL("http://www.google.com/calendar/feeds/default/allcalendars/full");
197     } catch (MalformedURLException ex) {
198         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
199     }
200     CalendarFeed resultFeed = null;
201     try {
202         // execute GET command on the URL and put the resultant feed into a tidy object
203         resultFeed = myService.getFeed(feedUrl, CalendarFeed.class);
204     } catch (IOException ex) {
205         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);

```

```
206     } catch (ServiceException ex) {
207         Logger.getLogger(CalendarService.class.getName()).log(Level.SEVERE, null, ex);
208     }
209
210     System.out.println("Your calendars:");
211     System.out.println();
212
213     // iterate through each entry and print the title (stored as a TextConstruct - getPlainText for text)
214     for (int i = 0; i < resultFeed.getEntries().size(); i++) {
215         CalendarEntry entry = resultFeed.getEntries().get(i);
216         System.out.println("\t" + entry.getTitle().getPlainText());
217     }
218 }
219
220 }
221
222
223
```

gBlogger.java

```
1 /* Copyright (c) 2008 Google Inc.
2  *
3  * Licensed under the Apache License, Version 2.0 (the "License");
4  * you may not use this file except in compliance with the License.
5  * You may obtain a copy of the License at
6  *
7  *     http://www.apache.org/licenses/LICENSE-2.0
8  *
9  * Unless required by applicable law or agreed to in writing, software
10 * distributed under the License is distributed on an "AS IS" BASIS,
11 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 * See the License for the specific language governing permissions and
13 * limitations under the License.
14 *
15 * Code from Samples \gdata\java\sample\blogger\BloggerClient.java
16 *
17 *
18 *** All Modifications by ***
19 *
20 * @author: Kai Kajus Noack
21 * @project: gConnect
22 * @version: 0.01
23 * @release: 2008-01-28
24 * @website: http://media-it.blogspot.com
25 *
26 * gBlogger.java
27 * Created January 2009
28 *
29 */
30
31 package gconnect;
32
33 import com.google.gdata.client.Query;
34 import com.google.gdata.client.blogger.BloggerService;
35 import com.google.gdata.data.DateTime;
36 import com.google.gdata.data.Entry;
37 import com.google.gdata.data.Feed;
38 //import com.google.gdata.data.Person;
39 import com.google.gdata.data.PlainTextConstruct;
40 import com.google.gdata.data.TextContent;
41 import com.google.gdata.util.*;
42 import java.util.logging.Level;
43 import java.util.logging.Logger;
44
45 import java.io.IOException;
46 import java.net.URL;
47
48 // necessary for Labeling of a post
49 import com.google.gdata.data.Category;
```

```

50
51
52 public class gBlogger {
53
54     String username = "";
55     String password = "";
56     String blogId = "";
57
58     BloggerService myService = new BloggerService("gBlogger Service 0.1");
59
60     Boolean authenticate(String user, String pwd) {
61         System.out.println("java: TRYING TO AUTHENTICATE");
62         this.username = user;
63         this.password = pwd;
64         try {
65             // Authenticate using ClientLogin
66             myService.setUserCredentials(user, pwd);
67         } catch (AuthenticationException ex) {
68             Logger.getLogger(BloggerService.class.getName()).log(Level.SEVERE, null, ex);
69             System.out.println("java: CANNOT ESTABLISH CONNECTION!");
70             return false;
71         }
72         System.out.println("java: CONNECTED");
73         try {
74             // get the blog ID from the metatfeed
75             blogId = getBlogId(myService);
76             System.out.println("java: blogId: " + blogId);
77             feedUri = FEED_URI_BASE + "/" + blogId;
78         } catch (ServiceException se) {
79             se.printStackTrace();
80         } catch (IOException ioe) {
81             ioe.printStackTrace();
82             System.out.println("java: CANNOT ESTABLISH CONNECTION!");
83         }
84         // if everything worked
85         return true;
86     }
87
88     static final String METAFEED_URL = "http://www.blogger.com/feeds/default/blogs";
89     static final String FEED_URI_BASE = "http://www.blogger.com/feeds";
90     static final String POSTS_FEED_URI_SUFFIX = "/posts/default";
91     static final String COMMENTS_FEED_URI_SUFFIX = "/comments/default";
92     static String feedUri;
93
94     // Parses the metafeed to get the blog ID for the authenticated user's default blog.
95     private static String getBlogId(BloggerService myService)
96         throws ServiceException, IOException {
97         // Get the metafeed
98         final URL feedUrl = new URL(METAFEED_URL);
99         Feed resultFeed = myService.getFeed(feedUrl, Feed.class);
100
101         // If the user has a blog then return the id (which comes after 'blog-')

```

```

102     if (resultFeed.getEntries().size() > 0) {
103         Entry entry = resultFeed.getEntries().get(0);
104         return entry.getId().split("blog-")[1];
105     }
106     throw new IOException("User has no blogs!");
107 }
108
109 // returns the name of the blog of a given position
110 public String getUserBlogId(int position, Boolean wantName) throws ServiceException, IOException {
111     // Request the feed
112     final URL feedUrl = new URL(METAFEED_URL);
113     Feed resultFeed = myService.getFeed(feedUrl, Feed.class);
114     if (wantName) {
115         // return Name of Blog
116         return resultFeed.getEntries().get(position).getTitle().getPlainText();
117     }
118     else {
119         // returns the ID which comes after 'blog-'
120         Entry entry = resultFeed.getEntries().get(position);
121         return entry.getId().split("blog-")[1];
122     }
123 }
124
125 //returns the number of blogs found
126 public int howmanyUserBlogs()
127     throws ServiceException, IOException {
128     // request the feed
129     final URL feedUrl = new URL(METAFEED_URL);
130     Feed resultFeed = myService.getFeed(feedUrl, Feed.class);
131     // return the number of blogs
132     return resultFeed.getEntries().size();
133 }
134
135 // creates a new post on a blog by using a GoogleService
136 public Boolean createPost(String title, String content,
137     String postLabels, Boolean commentsAllowed, String postPublish,
138     String authorName, String userName, String activeBlogId, Boolean isDraft)
139     throws ServiceException, IOException {
140     // create the entry to insert
141     Entry myEntry = new Entry();
142     // set title of post
143     myEntry.setTitle(new PlainTextConstruct(title));
144     // set content of post
145     myEntry.setContent(new PlainTextConstruct(content));
146     // (not yet implemented)
147     // set author of post (not necessary if user is author)
148     // Person author = new Person(authorName, null, userName);
149     // myEntry.getAuthors().add(author);
150     // set draft or non-draft
151     myEntry.setDraft(isDraft);
152     // add time of post
153     DateTime timePublished = DateTime.parseDateTime(postPublish);

```

```

154     myEntry.setPublished(timePublished);
155     // add category i.e. the label
156     if (postLabels.equals("")) {
157         // do nothing
158     }
159     else {
160         Category category = new Category();
161         category.setScheme("http://www.blogger.com/atom/ns#");
162         category.setTerm(postLabels);
163         myEntry.getCategories().add(category);
164     }
165     // set allow or disallow of comments
166     // (implementation not yet done)
167
168     // insert the new entry according to the active blog
169     URL postUrl = new URL(FEED_URI_BASE + "/" + activeBlogId + POSTS_FEED_URI_SUFFIX);
170     Entry currentPost;
171     try {
172         currentPost = myService.insert(postUrl, myEntry);
173     }
174     catch (IOException ex) {
175         Logger.getLogger(BloggerService.class.getName()).log(Level.SEVERE, null, ex);
176         return false;
177     } catch (ServiceException ex) {
178         Logger.getLogger(BloggerService.class.getName()).log(Level.SEVERE, null, ex);
179         return false;
180     }
181     System.out.println("java: Created post: " + currentPost.getTitle().getPlainText());
182     // return true if posting worked without any error
183     return true;
184 }
185
186
187
188 /** THE FOLLOWING HAS NOT BEEN MODIFIED AND IS TAKEN FROM GOOGLE SAMPLES */
189
190 /**
191  * Displays the titles of all the posts in a blog. First it requests the posts
192  * feed for the blogs and then is prints the results.
193  *
194  * @param myService An authenticated GoogleService object.
195  * @throws ServiceException If the service is unable to handle the request.
196  * @throws IOException If the URL is malformed.
197  */
198 public static void printAllPosts(BloggerService myService)
199     throws ServiceException, IOException {
200     // Request the feed
201     URL feedUrl = new URL(feedUri + POSTS_FEED_URI_SUFFIX);
202     Feed resultFeed = myService.getFeed(feedUrl, Feed.class);
203
204     // Print the results
205     System.out.println(resultFeed.getTitle().getPlainText());

```

```

206     for (int i = 0; i < resultFeed.getEntries().size(); i++) {
207         Entry entry = resultFeed.getEntries().get(i);
208         System.out.println("\t" + entry.getTitle().getPlainText());
209     }
210     System.out.println();
211 }
212
213 /**
214  * Displays the title and modification time for any posts that have been
215  * created or updated in the period between the startTime and endTime
216  * parameters. The method creates the query, submits it to the GoogleService,
217  * then displays the results.
218  *
219  * Note that while the startTime is inclusive, the endTime is exclusive, so
220  * specifying an endTime of '2007-7-1' will include those posts up until
221  * 2007-6-30 11:59:59PM.
222  *
223  * @param myService An authenticated GoogleService object.
224  * @param startTime DateTime object specifying the beginning of the search
225  *        period (inclusive).
226  * @param endTime DateTime object specifying the end of the search period
227  *        (exclusive).
228  * @throws ServiceException If the service is unable to handle the request.
229  * @throws IOException If the URL is malformed.
230  */
231 public static void printDateRangeQueryResults(BloggerService myService,
232     DateTime startTime, DateTime endTime) throws ServiceException,
233     IOException {
234     // Create query and submit a request
235     URL feedUrl = new URL(feedUri + POSTS_FEED_URI_SUFFIX);
236     Query myQuery = new Query(feedUrl);
237     myQuery.setUpdatedMin(startTime);
238     myQuery.setUpdatedMax(endTime);
239     Feed resultFeed = myService.query(myQuery, Feed.class);
240
241     // Print the results
242     System.out.println(resultFeed.getTitle().getPlainText() + " posts between " + startTime + " and " + endTime);
243     for (int i = 0; i < resultFeed.getEntries().size(); i++) {
244         Entry entry = resultFeed.getEntries().get(i);
245         System.out.println("\t" + entry.getTitle().getPlainText());
246         System.out.println("\t" + entry.getUpdated().toStringRfc822());
247     }
248     System.out.println();
249 }
250
251 /**
252  * Updates the title of the given post. The Entry object is updated with the
253  * new title, then a request is sent to the GoogleService. If the insertion is
254  * successful, the updated post will be returned.
255  *
256  * Note that other characteristics of the post can also be modified by
257  * updating the values of the entry object before submitting the request.

```

```

258     *
259     * @param myService An authenticated GoogleService object.
260     * @param entryToUpdate An Entry containing the post to update.
261     * @param newTitle Text to use for the post's new title.
262     * @return An Entry containing the newly-updated post.
263     * @throws ServiceException If the service is unable to handle the request.
264     * @throws IOException If the URL is malformed.
265     */
266     public static Entry updatePostTitle(BloggerService myService,
267         Entry entryToUpdate, String newTitle) throws ServiceException,
268         IOException {
269         entryToUpdate.setTitle(new PlainTextConstruct(newTitle));
270         URL editUrl = new URL(entryToUpdate.getEditLink().getHref());
271         return myService.update(editUrl, entryToUpdate);
272     }
273
274     /**
275     * Adds a comment to the specified post. First the comment feed's URI is built
276     * using the given post ID. Then an Entry is created for the comment and
277     * submitted to the GoogleService.
278     *
279     * NOTE: This functionality is not officially supported yet.
280     *
281     * @param myService An authenticated GoogleService object.
282     * @param postId The ID of the post to comment on.
283     * @param commentText Text to store in the comment.
284     * @return An entry containing the newly-created comment.
285     * @throws ServiceException If the service is unable to handle the request.
286     * @throws IOException If the URL is malformed.
287     */
288     public static Entry createComment(BloggerService myService, String postId,
289         String commentText) throws ServiceException, IOException {
290         // Build the comment feed URI
291         String commentsFeedUri = feedUri + "/" + postId + COMMENTS_FEED_URI_SUFFIX;
292         URL feedUrl = new URL(commentsFeedUri);
293
294         // Create a new entry for the comment and submit it to the GoogleService
295         Entry myEntry = new Entry();
296         myEntry.setContent(new PlainTextConstruct(commentText));
297         return myService.insert(feedUrl, myEntry);
298     }
299
300     /**
301     * Displays all the comments for the given post. First the comment feed's URI
302     * is built using the given post ID. Then the method requests the comments
303     * feed and displays the results.
304     *
305     * @param myService An authenticated GoogleService object.
306     * @param postId The ID of the post to view comments on.
307     * @throws ServiceException If the service is unable to handle the request.
308     * @throws IOException If there is an error communicating with the server.
309     */

```

```

310 public static void printAllComments(BloggerService myService, String postId)
311     throws ServiceException, IOException {
312     // Build comment feed URI and request comments on the specified post
313     String commentsFeedUri = feedUri + "/" + postId + COMMENTS_FEED_URI_SUFFIX;
314     URL feedUrl = new URL(commentsFeedUri);
315     Feed resultFeed = myService.getFeed(feedUrl, Feed.class);
316
317     // Display the results
318     System.out.println(resultFeed.getTitle().getPlainText());
319     for (int i = 0; i < resultFeed.getEntries().size(); i++) {
320         Entry entry = resultFeed.getEntries().get(i);
321         System.out.println("\t" +
322             ((TextContent) entry.getContent()).getContent().getPlainText());
323         System.out.println("\t" + entry.getUpdated().toStringRfc822());
324     }
325     System.out.println();
326 }
327
328 /**
329  * Removes the comment specified by the given editLinkHref.
330  *
331  * @param myService An authenticated GoogleService object.
332  * @param editLinkHref The URI given for editing the comment.
333  * @throws ServiceException If the service is unable to handle the request.
334  * @throws IOException If there is an error communicating with the server.
335  */
336 public static void deleteComment(BloggerService myService, String editLinkHref)
337     throws ServiceException, IOException {
338     URL deleteUrl = new URL(editLinkHref);
339     myService.delete(deleteUrl);
340 }
341
342 /**
343  * Removes the post specified by the given editLinkHref.
344  *
345  * @param myService An authenticated GoogleService object.
346  * @param editLinkHref The URI given for editing the post.
347  * @throws ServiceException If the service is unable to handle the request.
348  * @throws IOException If there is an error communicating with the server.
349  */
350 public static void deletePost(BloggerService myService, String editLinkHref)
351     throws ServiceException, IOException {
352     URL deleteUrl = new URL(editLinkHref);
353     myService.delete(deleteUrl);
354 }
355 }
356
357 /** Code from Google Samples:
358  *
359  * // Demonstrate how to publish a public post.
360  * Entry publicPost = createPost(myService, "Back from vacation", "<p>I didn't want to leave.<p>", "Post author",
361     *                               userName, false);

```

```
361 System.out.println("Successfully created public post: " + publicPost.getTitle().getPlainText());
362
363 // Demonstrate various feed queries.
364 printAllPosts(myService);
365 printDateRangeQueryResults(myService, DateTime.parseDate("2007-04-04"),
366 DateTime.parseDate("2007-04-06"));
367
368 // Demonstrate two ways of updating posts.
369 draftPost.setTitle(new PlainTextConstruct("Swimming with the fish"));
370 draftPost.update();
371 System.out.println("Post's new title is \"
372 + draftPost.getTitle().getPlainText() + \"\".\n");
373 publicPost = updatePostTitle(myService, publicPost, "The party's over");
374 System.out.println("Post's new title is \"
375 + publicPost.getTitle().getPlainText() + \"\".\n");
376
377 // Demonstrate how to add a comment on a post
378 // Get the post ID and build the comments feed URI for the specified post
379 System.out.println("Creating comment");
380 String selfLinkHref = publicPost.getSelfLink().getHref();
381 String[] tokens = selfLinkHref.split("/");
382 String postId = tokens[tokens.length - 1];
383 Entry comment = createComment(myService, postId, "Did you see any sharks?");
384
385 // Demonstrate how to retrieve the comments for a post
386 printAllComments(myService, postId);
387
388 // Demonstrate how to delete a comment from a post
389 System.out.println("Deleting comment");
390 deleteComment(myService, comment.getEditLink().getHref());
391
392 // Demonstrate two ways of deleting posts.
393 System.out.println("Deleting draft post");
394 draftPost.delete();
395 System.out.println("Deleting published post");
396 deletePost(myService, publicPost.getEditLink().getHref());
397
398 }
399 */
400
401
402
```

TextArea.fx

```
1 /*
2  * MyTextArea.fx
3  * Created on 2009-01-26
4  *
5  * @author Steven Herod
6  * http://forums.sun.com/thread.jspa?threadID=5357725&messageID=10558050#10558050
7  *
8  * thanks Steve!
9  * partly outcommented/modified by Kai Kajus Noack
10 *
11 */
12
13 package gconnect;
14
15 import java.awt.event.KeyEvent;
16 import java.awt.event.KeyListener;
17 import java.awt.Font;
18 import javafx.ext.swing.SwingComponent;
19 import javax.swing.JComponent;
20 import javax.swing.JTextArea;
21 import java.awt.Color;
22
23 public class TextArea extends SwingComponent {
24
25     var myComponent: JTextArea;
26
27     public var length: Integer;
28     public var readText:String;
29     public var text: String on replace{
30         myComponent.setText(text);
31     }
32
33     public var tooltipText: String on replace{
34         myComponent.setToolTipText(tooltipText);
35     }
36
37
38     public override function createJComponent():JComponent{
39         // translateX = 15;
40         // translateY = 5;
41         // var f:Font = new Font("sanserif",Font.PLAIN, 11);
42
43         myComponent = new JTextArea(4, 33);
44         myComponent.setOpaque(false);
45         // myComponent.setFont(f);
46         myComponent.setWrapStyleWord(true);
47         myComponent.setLineWrap(true);
48         myComponent.addListener( KeyListener{
49             public override function keyPressed(keyEvent:KeyEvent) {
```

```
50         if (keyEvent.VK_PASTE == keyEvent.getKeyCode()) {
51             myComponent.paste();
52         }
53     }
54     public override function keyReleased( keyEvent:KeyEvent) {
55         var pos = myComponent.getCaretPosition();
56         text = myComponent.getText();
57         myComponent.setCaretPosition(pos);
58     }
59     public override function keyTyped(keyEvent:KeyEvent) {
60         length = myComponent.getDocument().getLength();
61     }
62 }
63 );
64 return myComponent;
65 }
66 }
67
68
```