JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

Mobile Access to Web Services

Systemtechnik Labor
5BHITT 2015/16, Gruppe Y

Christoph Hackenberger

Version 1.0

Note:

Betreuer: M. Borko

Begonnen am 19. Februar 2016

Beendet am 25. Februar 2016

Inhaltsverzeichnis

1. Einführung

Diese Übung gibt einen Einblick in Entwicklungen von mobilen Applikationen.

1.1 Ziele

Das Ziel dieser Übung ist eine Anbindung einer mobilen Applikation an ein Webservices.
Die Anbindung soll mit Hilfe eines RESTful Webservice (Gruppe1) umgesetzt werden.

1.2 Voraussetzungen

- Grundlagen Java und XML
- Grundlegendes Verständnis über Entwicklungs- und Simulationsumgebungen
- Verständnis von RESTful Webservices

1.3 Aufgabenstellung

Es ist eine mobile Anwendung zu implementieren, die sich an das Webservice aus der Übung DezSysLabor-09 "Web Services in Java" anbinden soll. Dabei müssen die entwickelten Schnittstellen entsprechend angesprochen werden.

Es ist freigestellt, welche mobile Implementierungsumgebung dafür gewählt wird. Empfohlen wird aber eine Implementierung auf Android

1.4 Quellen

- "Android Restful Webservice Tutorial – How to call RESTful webservice in Android – Part 3"; Posted By Android Guru on May 27, 2014; online: http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-call-restful-webservice-in-android-part-3/
- "Referenzimplementierung von DezSys09"; Paul Kalauner; online: https://github.com/pkalauner-tgm/dezsys09-java-webservices

Bewertung: 16 Punkte
- Anbindung einer mobilen Applikation an die Webservice-Schnittstelle (6 Punkte)
- Registrierung von Benutzern (3 Punkte)
- Login und Anzeige einer Willkommensnachricht (3 Punkte)
- Simulation bzw. Deployment auf mobilem Gerät (2 Punkte)
- Protokoll (2 Punkte)

2. Entwicklung der iOS App

Zur auffrischen der spärlich vorhandenen Swift Kenntnisse stand die Dokumentation[1] und Tutorial[2] von Apple zur Verfügung.

2.1 Swift Basics

Variablen und Konstanten

Variablen:

```
var a = 1
```

oder mit Typ

```
var userDefinedColorName: String = “red“
```

Konstanten:

```
let defaultColorName = "red"
```

Klassen und Funktionen

In Swift wird zwischen Klassen und Strukturen unterschieden. Wobei Klassen gegenüber Strukturen ein paar zusätzliche Features anbietet zb. Vererbung, Type casting etc. [1]

```
class SomeClass {
    // class definition goes here
}
struct SomeStructure {
    // structure definition goes here
}
```

Vererbung:

```
class Tandem: Bicycle  {
    var currentNumberOfPassengers = 0
}
```

Funktionen:

```
func sayHelloAgain(personName:  String) -> String {
     return "Hello again, " + personName + "!"
}
print(sayHelloAgain("Anna"))
// prints "Hello again, Anna!"
```

Bei Funktionen ohne Rückgabewert muss nicht, wie in anderen Sprachen, () -> Void geschrieben werden, sondern es wird einfach weggelassen.

Closures

„Closures are self-contained blocks of functionality that can be passed around and used in your code. Closures in Swift are similar to blocks in C and Objective-C and to lambdas in other programming languages.“ [1]

```
reversed = names.sort({ (s1: String, s2: String) -> Bool in
    return s1 > s2
})
```

Access Control

Die Access Modifiers in Swift sind File basiert.

1. Public: Zugriff von überall möglich
2. Internal: Zugriff von überall im selben Module
3. Private: Zugriff nur im selben Source File möglich

```
public class SomePublicClass {}
internal class SomeInternalClass {}
private class SomePrivateClass {}

public var somePublicVariable = 0
internal let someInternalConstant = 0
private func somePrivateFunction() {}
```

2.2 Implementierung

Zur Entwicklung der iOS App wurde XCode verwendet. Als Template für die App wurde eine Single View Application gewählt. Danach wurde zuerst das Storyboard entsprechend angepasst und das AppIcon in den Assets festgelegt. Zum Abschluss wurde noch die Anbindung an die, in der Übung DEZSYS09 erstellten, REST Schnittstelle implementiert. Da die Implementierung des HTTP Request standardmäßig keine unverschlüsselten Requests sondern nur HTTPS Requests zulässt wurde die REST Schnittstelle auf OpenShift Cloud deployed. (siehe Punkt 3)

3. Deployment auf OpenShift

Die in der vorherigen Übung (DEZSYS 09) erstellte Übung wurde nun auf OpenShift Deployed. Dazu wurde mittels des Tutorial [3] die OpenShift App erstellt und konfiguriert. Es wurde jedoch, anders als im Tutorial, Tomcat in Version 9 installiert. Außerdem musst anstatt der OpenJDK 7 die Oracle JDK 8 verwendet werden, dazu einfach folgende Schritte befolgen:

export JAVA_HOME=/etc/alternatives/java_sdk_1.8.0
export PATH=$JAVA_HOME/bin:$PATH

Als Datenbank wurde noch der mysql-5.5 cartridge hinzugefügt. Natürlich müssen in der Applikation dann die Zugänge für die DB richtig eingestellt werden und im pom.xml muss die neue Adresse des Tomcat Servers, für das Deployment, konfiguriert werden.

4. Github Repo

https://github.com/chackenberger/Useraccount_App

5. Quellen

[1] The Swift Programming Language (Swift 2.1), iOS Developer Library, verfügbar unter: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Basic

Operators.html#//apple_ref/doc/uid/TP40014097-CH6-ID60 [abgerufen am 20.2.2016]

[2] Start Developing iOS Apps (Swift), iOS Developer Library, verfügbar unter: https://developer.apple.com/library/ios/referencelibrary/GettingStarted/DevelopiOSAppsSwift/ [abgerufen am 20.2.2016]

[3] Free Apache Tomcat Hosting in the Cloud for Java Applications? It's Called OpenShift!, OpenShift Blog, verfügbar unter: https://blog.openshift.com/free-apache-tomcat-hosting-in-the-cloud-for-java-applications-its-called-openshift/ [abgerufen am 23.2.2016]