

# **KAPITEL 3: DATENAUSTAUSCH UND SERVERZUGRIFF**

Weiterführende Konzepte mobiler Applikationen  
WWI11SCA - 5.Semester  
Michael Dietz

# AGENDA

- I. Übersicht
- II. Übertragungskanal
- III. Protokolle
- IV. HTTP Aufrufe in Android
- V. Formate
- VI. Zugriffsparadigmen
- VII. Serveralternativen

# DATENAUSTAUSCH

# Datenaustausch

## Übersicht

- Neben dem lokalen Speichern von Daten innerhalb einer App ist der Datenaustausch mit Servern ein häufiges Szenario
- Beispiele
  - Die App lädt aktuelle Wetterdaten vom Server
  - Die Spiele-App verwendet eine übergreifende Highscore Liste
  - Die App kann Nachrichten zwischen Nutzern versenden

# DATENAUSTAUSCH EBENEN

- Zur Realisierung einer Datenübertragung müssen die folgenden Ebenen betrachtet werden
  - Übertragungskanal (Worüber)
  - Übertragungsprotokoll (Wie)
  - Format (Was)
  - Zugriffsparadigma
- Auf allen Ebenen werden nur die bekanntesten, wichtigsten Alternativen vorgestellt



# Übertragungskanal

*Worüber übertragen wir  
Daten?*

# Übertragungskanal

## Übersicht

- Aus Sicht eines mobilen Gerätes können Daten über verschiedene Kanäle gesendet/empfangen werden
  - WLAN
  - Mobiles Internet: LTE, 3G, 2G (GPRS, EDGE, ..)
  - Bluetooth
  - Ethernet (eher selten)
- Moderne APIs abstrahieren die Verbindungskanäle
- Bei Bedarf kann der verfügbare Kanal explizit eingeschränkt werden, z.B. Android nur Wifi

```
ConnectivityManager connMgr = (ConnectivityManager).getSystemService(Context.CONNECTIVITY_SERVICE);  
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();  
if(networkInfo.getType() == ConnectivityManager.TYPE_WIFI)
```

# Übertragungskanal Auswahl

- In welchen Fällen beachtet man welcher Übertragungskanal zur Verfügung steht
  - Bluetooth für lokale Verbindungen zu anderen Geräten
  - WLAN (oder Ethernet) für Apps die große Datenmengen übertragen wollen oder Daten ohne große Latenz übertragen (z.B. Streaming)
  - Internet über das mobile Netz wenn man Funktion zur Abrechnung aus dem Mobilfunknetz verwendet



# Übertragungsprotokoll

## *Wie übertragen wir Daten?*

# Übertragungsprotokoll

## Übersicht HTTP

- Das bedeutendste Übertragungsprotokoll für die Kommunikation zwischen App und Server ist HTTP
- HTTP (Hypertext Transfer Protocol) wurde ursprünglich für die Übertragung von HTML-Seiten geschaffen
- Aufgrund seines flexiblen Aufbaus eignet es sich aber auch für andere Übertragungsszenarien
- Die meisten Plattformen bieten einen HTTPClient der die Nutzung von HTTP abstrahiert
- HTTPS nutzt zusätzlich SSL (Secure Socket Layer) um eine gesicherte Verbindung zu ermöglichen

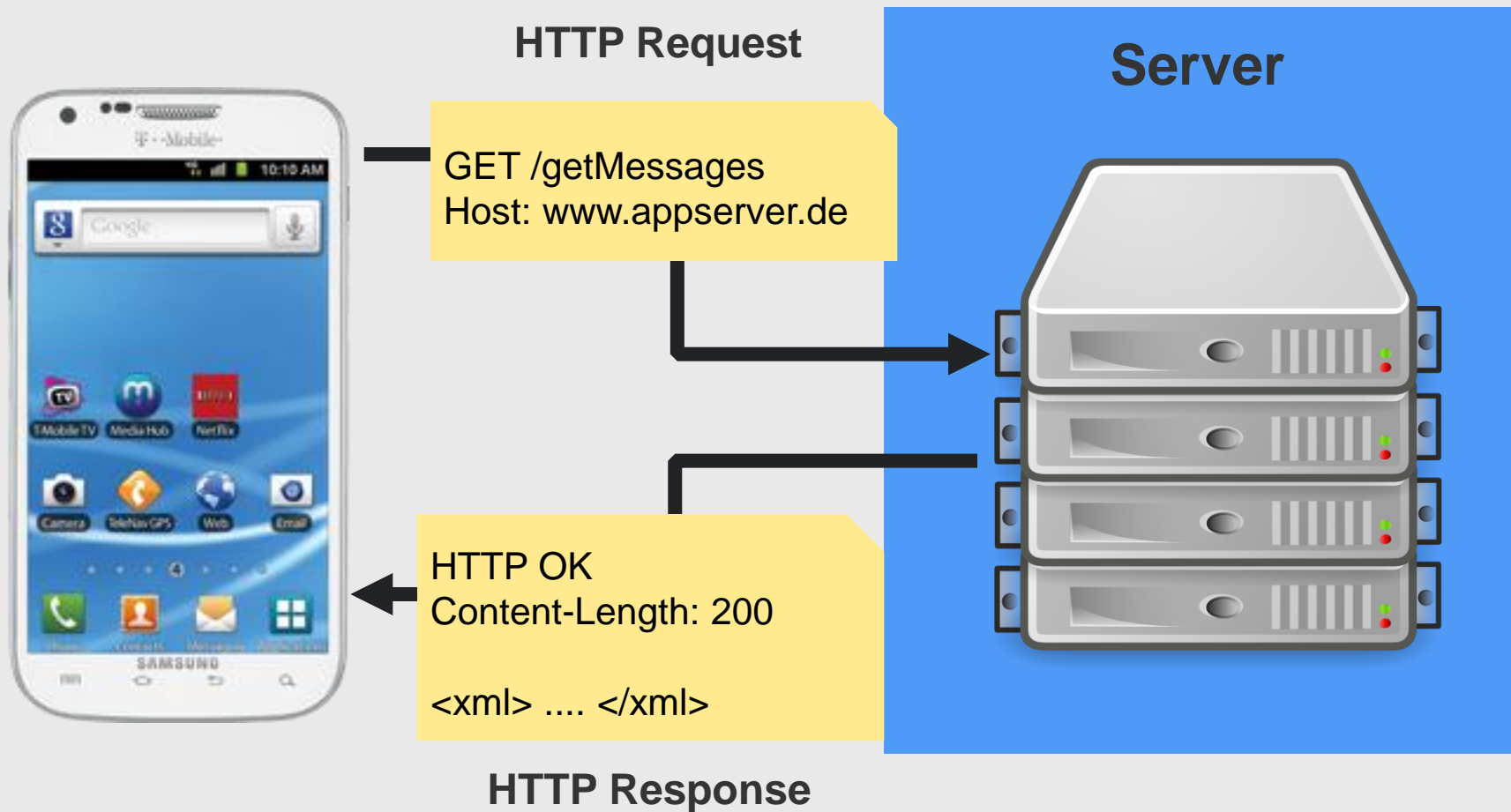
# Übertragungsprotokoll

## HTTP Eigenschaften

- HTTP ist ein zustandsloses Protokoll: D.h. der Server speichert sich keine Daten zum Client
  - Dies kann durch Cookies verbessert werden
- Eine Verbindung zwischen Client und Server wird nicht aufrecht gehalten
  - Der Client sendet den HTTP Request
  - Der Server antwortet mit der HTTP Response
  - Die Verbindung wird geschlossen

# Übertragungsprotokoll

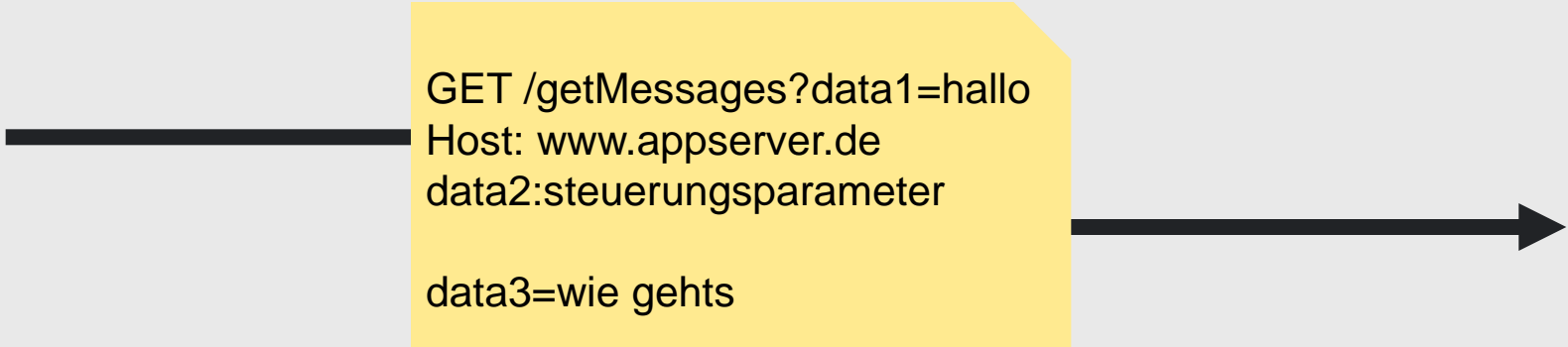
## HTTP Übertragung



# Übertragungsprotokoll

## HTTP Datenübertragung

- Die Übertragung von Daten vom Client an den Server kann auf zwei Wegen erfolgen
  - URL-Parameter: Daten werden durch Parameter in der URL weitergegeben
  - HTTP-Body: Nutzdaten werden im Body des HTTP-Requests mitgegeben
- Steuerungsdaten können zudem auch über den Header mitgegeben werden



The diagram illustrates an HTTP GET request. A thick black horizontal line enters from the left and terminates at a yellow rectangular box with a folded top-right corner. Inside this box, the request details are listed: the method and path 'GET /getMessages?data1=hallo', the host 'Host: www.appserver.de', and two parameters 'data2:steuerungsparameter' and 'data3=wie gehts'. A thick black arrow points from the right side of the yellow box towards the right edge of the slide.

```
GET /getMessages?data1=hallo  
Host: www.appserver.de  
data2:steuerungsparameter  
  
data3=wie gehts
```

# Übertragungsprotokoll

## HTTP Methoden

- Das HTTP Protokoll definiert verschiedene Methoden die aufgerufen werden können
  - GET: Anfordern einer Ressource
  - POST: Übertragung von Daten an den Server
  - HEAD: Nur der HTTP-Header wird zurückgeliefert
  - Eher Selten:
    - PUT: Dient dem Hochladen einer Ressource
    - DELETE: Löscht eine Ressource
    - OPTIONS: Liefert die verfügbaren Methoden

# Übertragungsprotokoll

## Zusammenfassung

- HTTP als Übertragungsprotokoll ist eine der einfachsten und weitverbreitetsten Alternative zum Austausch von Daten zwischen Client und Server
- HTTP hat seine Nachteile
  - Zustandslos
  - Keine Server -> Client Kommunikation
- Alternativ können je nach Plattform auch low-level TCP-IP Socket-Verbindungen oder andere Protokolle verwendet werden

# HTTP Client in Android



# HTTP CLIENT IN ANDROID

## ÜBERSICHT

- In Android stehen komfortable Klassen für den Aufruf per HTTP zur Verfügung
  - HttpClient: Besser in Froyo und Eclair
  - HttpURLConnection: Besser ab Gingerbred
- Um in der App Aufrufe an einen Server durchführen zu können muss eine „Permission“ eingerichtet werden
- Wo macht man das?

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

<http://android-developers.blogspot.de/2011/09/androids-http-clients.html>

# HTTP CLIENT IN ANDROID

## VORBEREITUNG

- Bevor man die Verbindung herstellt sollte man die Verfügbarkeit der Internetverbindung überprüfen
- In einer mobilen App muss immer damit gerechnet werden das keine Verbindung vorhanden ist
- Darauf muss man in der App adäquat reagieren

```
ConnectivityManager con = (ConnectivityManager).getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = con.getActiveNetworkInfo();

if (networkInfo != null && networkInfo.isConnected()) {
    // fetch data
} else {
    // display error
}
```

# HTTP CLIENT IN ANDROID

## AUFRUF

- Der eigentliche Aufruf über die HttpURLConnection ist hingegen einfach

```
InputStream is = null;

try {

    URL url = new URL("http://space-labs.appspot.com/repo/465001/highscore.sjs");
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();

    conn.setRequestMethod("GET");
    conn.connect();

    int response = conn.getResponseCode();
    is = conn.getInputStream();

    // Convert the InputStream into a string
    String contentAsString = convertStreamToString (is); //Nächste Folie

    return contentAsString;

} finally {
    if (is != null) {
        is.close();
    }
}
```

# HTTP CLIENT IN ANDROID

## STREAM ZU STRING

- Um den InputStream in einen String zu machen eignet sich die BufferedReader Klasse
- Die Daten des Streams werden Zeilenweise ausgelesen und mit einem StringBuilder zu einem String konvertiert

```
public static String convertStreamToString(InputStream is) throws Exception {  
    BufferedReader reader = new BufferedReader(new InputStreamReader(is));  
    StringBuilder sb = new StringBuilder();  
    String line = null;  
  
    while ((line = reader.readLine()) != null) {  
        sb.append(line);  
    }  
  
    is.close();  
  
    return sb.toString();  
}
```

# HTTP CLIENT IN ANDROID

## ASYNCHRONE VERARBEITUNG

- Zum Laden von Daten von einem Server wird eine Asynchrone Verarbeitung empfohlen
- Eine synchrone Verarbeitung blockiert den aktuellen Thread und damit die UI-Aktivität

```
private class DownloadWebpageTask extends AsyncTask<String, Void, String> {  
    @Override  
    protected String doInBackground(String... urls) {  
  
        // params comes from the execute() call: params[0] is the url.  
        try {  
            return downloadFromUrl(urls[0]);  
        } catch (IOException e) {  
            return "Unable to retrieve web page. URL may be invalid.";  
        }  
    }  
  
    // onPostExecute displays the results of the AsyncTask.  
    @Override  
    protected void onPostExecute(String result) {  
  
    }  
}
```

# HTTP CLIENT IN ANDROID

## AUFGABE

- Die nachfolgende URL bietet die Möglichkeit Highscore Einträge an den Server zu senden
- [http://space-labs.appspot.com/repo/465001/highscore\\_add.sjs](http://space-labs.appspot.com/repo/465001/highscore_add.sjs)
- Über URL Parameter können die Parameter Name, Level und Points mitgegeben werden. Optional noch ein Client (z.B. Name des Spiels, Plattform, ...)
- Aufgabe: Nach dem Speichern des Highscore-Eintrages in der lokalen Datenbank machen Sie einen HTTP-Aufruf und übermitteln die Daten an diese URL
- Test-URL: <http://space-labs.appspot.com/repo/465001/index.html>

# Übertragungsformat

## Welche Daten werden übertragen?

# Übertragungsformat Überblick

- In welchem Format können Daten effizient übertragen werden?
- Herausforderungen
  - Einfache Verarbeitung
  - Größe / Übertragungsmenge / Overhead
  - Offenheit



# Übertragungsformat XML

- Extensible Markup Language (XML) ist eine Auszeichnungssprache zur Definition hierarchischer Daten in Textform
- Wird für den plattform-unabhängigen Austausch von Daten verwendet
- Syntax (grob):
  - <tagname>
  - Content
  - </tagname>

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CONTACTLIST SYSTEM "contacts.dtd">
<CONTACTLIST>
  <CONTACT>
    <NAME>Michael Naef</NAME>
    <ADDRESS>Rosenstrasse 28, 8001 Zuerich</ADDRESS>
    <PHONE type="mobile">079 123 4567</PHONE>
    <PHONE type="private">01 123 4567</PHONE>
    <MAIL>naef@acm.org</MAIL>
  </CONTACT>
  <CONTACT>
    <NAME>werner Hartmann</NAME>
    <PHONE type="office">01 987 6543</PHONE>
    <PHONE type="fax">01 222 2222</PHONE>
    <MAIL>hartmann@inf.ethz.ch</MAIL>
  </CONTACT>
</CONTACTLIST>
```

# Übertragungsformat XML (Forts.)

- Vorteile
  - Gute Lesbarkeit bei Formatierung
  - Gute Verarbeitung durch verfügbare XML-Parser
  - Abbildung von komplexen Datenstrukturen und Binärdaten (mit Base64)
  - Gute Erweiterbarkeit
  - Kann auf inhaltliche Korrektheit geprüft werden (XML Schema)
- Nachteile
  - XML Parser wird zwingend benötigt und ist nicht auf allen Plattformen eingebaut
  - Verarbeitung/Parsen benötigt Zeit
  - Verarbeitung in Applikationen schwieriger
  - Höherer Daten-Overhead

# Übertragungsformat JSON

- JavaScript Object Notation
- Ursprünglich aus JavaScript heraus entstanden zur Serialisierung von Objekten
- Einfacher Aufbau
- Es gibt Objekte, Arrays und Felder (String, Integer, Float, Boolean)

```
[
  - {
    - list: {
      created_at: "2010-10-04T21:14:57Z",
      title: "test",
      updated_at: "2010-10-04T21:14:57Z",
      owner_id: 1,
      id: 2
    }
  },
  - {
    - list: {
      created_at: "2010-10-06T17:39:09Z",
      title: "hallo welt",
      updated_at: "2010-10-06T17:39:58Z",
      owner_id: 1,
      id: 4
    }
  },
  - {
    - list: {
      created_at: "2010-10-06T17:59:40Z",
      title: "test",
      updated_at: "2010-10-06T17:59:40Z",
      owner_id: 1,
      id: 5
    }
  }
]
```

<http://www.json.org/>

# Übertragungsformat JSON (Forts.)

- Vorteile
  - JSON-Parser auf allen Plattformen vorhanden
  - Kann in einfacher Form auch ohne JSON-Parser verarbeitet werden
  - Abbildung von komplexen Datenstrukturen und Binärdaten (mit Base64)
  - Einfach und schnell zu verstehen
  - Minimaler Daten-Overhead
- Nachteile
  - Schlecht in der Verarbeitung von komplexen Datenstrukturen
  - JSON-Parser muss teilweise noch zusätzlich als Library aufgenommen werden
  - Keine Validierungsmöglichkeit ob Inhalt korrekt (muss Anwendung übernehmen)

# XML vs. JSON

## AUFGABE

- Bilden Sie die folgenden Daten im XML und JSON Format ab
- Validieren Sie die beiden Formate
  - <http://www.xmlvalidation.com/>
  - <http://jsonlint.com/>
- Vergleichen Sie die Größe der beiden Formate. Um welchen Faktor ist das XML-Format größer?

Kunden		
Name	Ort	Umsatz
Peter	Mainz	180
Meier	Berlin	200
Berger	Köln	150

# JSON IN ANDROID

## ÜBERSICHT

- Zur Verwendung von JSON in Android Apps existieren die Klassen JSONObject und JSONArray
  - Können JSON-Texte interpretieren
  - Können JSON-Texte generieren

```
String jsonString = "{\"playername\":\"Michael\"}";

try{

    JSONObject json = new JSONObject(jsonString);

    String playername = json.getString("name");

    System.out.println(playername + „ found in JSON“ + json.toString());

}catch(JSONException e){
    System.out.println("JSON format wrong: " + e.toString());
}
```

# DATENAUSTAUSCH PARADIGMEN

# PARADIGMEN

## SOA / WEB SERVICES

- Web Services sind einzelne Funktionen, die über eine öffentliche Schnittstelle zur Verfügung gestellt werden
- Sie verwenden HTTP als Protokoll
- Eine WSDL-Datei (Web Service Description Language) beschreibt die Schnittstelle im Detail
- Die Datenübertragung erfolgt über SOAP, eine spezielle Definition einer Nachricht basierend auf XML
- Hier spricht man auch von Service-orientierten Architekturen (SOA)
- Beispiel: <http://www.thomas-bayer.com/axis2/services/BLZService?wsdl>



# PARADIGMEN

## SOA / WEB SERVICES (FORTS.)

- Vorteile
  - WSDL beschreibt die Schnittstelle und enthält alle notwendigen Infos für den Verwender
  - Proxy-Klassen können generiert werden
  - Sehr umfangreich mit vielen Möglichkeiten (z.B. Security & Richtlinien, Datentypen, ...)
- Nachteile
  - Komplexer Gesamtstack bei Service-Bereitstellen und beim Aufruf mit viel Overhead
  - Ohne Proxy-Generator schwer zu nutzen

# PARADIGMEN

## REST

- Representational State Transfer (REST) ist ein Paradigma, welches Aussagen/Empfehlungen zur Bereitstellung von Datenzugriffen über HTTP macht
- Es werden 4 Eigenschaften definiert
  - Adressierbarkeit: Eindeutige URL für einen Dienst
  - Repräsentation: JSON oder XML
  - Zustandslosigkeit: Server kennt keine Zustandsinformationen
  - Operationen: HTTP-Methoden zur Definition, welche Operation ausgeführt werden soll. Strikte Einhaltung.

# PARADIGMEN

## REST-BEISPIEL

- Der Zugriff auf eine Liste von Spielern erfolgt über die URL <http://space-labs.appspot.com/repo/465001/player.sjs>
- Unter strikter Einhaltung der HTTP-Methoden kann man über diese URL folgende Aktionen ausführen
  - GET: Lesen der Daten
  - POST: Anlegen eines neuen Spielers
  - PUT: Ändern einen Spielers
  - DELETE: Löschen eines Spielers

<http://www.ibm.com/developerworks/webservices/library/ws-restful/>

# PARADIGMEN

## ODATA

- Das Open Data Protocol (ODATA) ist offizielle Spezifikation zur Umsetzung von APIs
- ODATA bedient sich dazu einer Sammlung von verschiedenen Protokollen und Paradigmen
- Neben dem Zugriffparadigma (REST) definiert ODATA auch die zu erwartenden Formate (Atom, XML oder JSON) und erlaubt eine Navigation in den verfügbaren Objekten
- <http://www.odata.org/>
- Beispiel ODATA-Service <http://services.odata.org/OData/OData.svc/>

# SERVER ZUM DATENAUSTAUSCH

## *WAS VERWENDE ICH ALS SERVER?*

# SERVER ZUM DATENAUSTAUSCH

## ÜBERBLICK

- Frage: Welchen Server kann ich für meine App verwenden?
- Bekannte Optionen
  - Google App Engine: Kostenlos bis zu gewissen Grenzen (Quotas), Java/Python/Go, Umfangreiche Sammlung an API, Verschiedene Datenhaltungs-Optionen
  - Microsoft Azure: Kostenpflichtig, Einfache Bereitstellung von APIs ohne Programmierung
  - CloudBees: Kostenlos für Entwicklung, Integriertes Git, MySQL-Datenbank, nur Java Platform As A Service

# DATENAUSTAUSCH

## AUFGABE

- Unter der folgenden URL ist ein REST-basierter Service verfügbar <http://space-labs.appspot.com/repo/465001/highscore.sjs>
- Rufen Sie den Service auf und schauen Sie in welchem Format die Daten zurückgegeben werden
- Verarbeiten Sie das zurückgegebene Format
- Laden Sie die in der App dargestellte Highscore-Tabelle nun nicht mehr aus der lokalen DB sondern über diesen Service

# QUELLEN



# QUELLEN

- <http://developer.android.com/training/basics/data-storage/index.html>
- <http://www.ibm.com/developerworks/webservices/library/ws-restful/>
- <http://www.cloudbees.com/>
- <http://www.windowsazure.com/>
- <https://developers.google.com/appengine>
- <http://www.json.org/>
- <http://www.odata.org>