CS371m - Mobile Computing

Persistence - Web Based Storage
CHECK OUT
The Cloud ..........

IBMCloud @IBMcloud · Aug 23
Hamsters: Cute, but dangerous. Trust us—you don't want to share your #cloud with a hamster.
Backend

• No clear definition of backend
• front end - user interface
• backend - data, server, programs the user does not interact with directly
• With 1,000,000s of mobile and web apps ...
• rise of Backend as a Service (Baas)
• Sometimes MBaaS, M for mobile
Back End As a Service - May Provide:

• cloud storage of data
• integration with social networks
• push notifications
  — server initiates communication, not the client
• messaging and chat functions
• user management
• user analysis tools
• abstractions for dealing with the backend
Clicker

• How many Mobile Backend as a Service providers exist?

A. 1 or 2
B. about 5
C. about 10
D. about 20
E. 30 or more

https://github.com/relatedcode/ParseAlternatives
MBaaS

Data

Push

Analytics

Social

Cloud Code

Hosting
Some Examples of MBaaS

- Parse
- Firebase (Google)
- Amazon Web Services
- Google Cloud Platform
- Heroku
- PythonAnywhere
- Rackspace Cloud
- BaasBox (Open Source)
- Usergrid (Open Source)
Examples of Using a MBaaS

• Parse
• [www.parse.com](http://www.parse.com)
• various pricing models
• relatively easy to set up and use
• Going away 1/28/2017
Parse Set Up in AndroidStudio

1. request api key
2. Download Parse SDK
3. Unzip files
4. Create libs directory in app directory (select Project view)
5. Drag jar files to libs directory
Parse Set Up in AndroidStudio

6. add dependencies to gradle build file under app

like so:

```groovy
dependencies {
    compile 'com.android.support:support-v4:18.0.0'
    compile 'com.parse.bolts:bolts-android:1.+'
    compile fileTree(dir: 'libs', include: 'Parse-*.jar')
}
```

https://www.parse.com/apps/quickstart#
parse_data/mobile/android/native/new
Testing Parse

• Add permissions to manifest to access network state and use internet

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

• initialize Parse in onCreate method

```java
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Parse.initialize(this, "GACBq6Jwvf2PL7EIl3IRpvav7GElU...
```
Testing Parse

• at the end of `onCreate()`
• create and send a test object to Parse

```java
private void testParse() {
    ParseObject testObject = new ParseObject("TestObject");
    testObject.put("foo", "bar");
    testObject.saveInBackground();
}
```

• abstraction
  – handles doing this in the background, off the UI thread
Result of Test

Congrats! You saved your first object:

```json
{
  "id": "HQZcs4g5vp",
  "created_at": "2014-11-11T21:34:19Z",
  "updated_at": "2014-11-11T21:34:19Z",
  "foo": "bar"
}
```

- **JSON**
  - JavaScript Object Notation
ParseObject

```java
private void testParse() {
    ParseObject testObject = new ParseObject("TestObject");
    testObject.put("foo", "bar");
    testObject.saveInBackground();
}
```

- Local representation of data (on the device) that can be saved and retrieved from the Parse
- String in constructor is class name
  - like a table in a data base
- put to add key - value pairs
  - String - Object
  - keys must be alphanumerics
  - like a column in the row
ParseObject

private void testParse() {
    ParseObject testObject = new ParseObject("TestObject");
    testObject.put("foo", "bar");
    testObject.saveInBackground();
}

• `saveInBackground` method saves object to Parse in a background thread

• multiple options for saving
  – `saveAll(List)`
  – `saveEventually()` - if server or network not available
  – `saveInBackground(SaveCallback)`
Parse and RandomArt

• add ability to save equations
• save to parse database
• allow multiple users to save equations
• functionality to display a random equation others liked
• up and down votes
onClick for Keep This

```java
public void saveEquation(View v) {
    if (exp != null) {
        // should also check to ensure equation not already saved
        final int[] count = {0};

        ParseQuery<ParseObject> countQuery
            = ParseQuery.getQuery("ArtExpressionCount");

        countQuery.getFirstInBackground(new GetCallback<ParseObject>() {
            @Override
            public void done(ParseObject masterCount, ParseException e) {
                if (e == null) {
                    count[0] = masterCount.getInt("TheCount");
                    Log.d(TAG, "The Count via the master count object: ");
                    masterCount.increment("TheCount");
                    masterCount.saveInBackground();
                }
            }
        });
    }
}
```
masterCount.saveInBackground();

ParseObject currentExpression = new ParseObject("ArtExpression");

currentExpression.put("equation", exp.toString());
currentExpression.put("votes", 1);
currentExpression.put("index", count[0]);
currentExpression.saveInBackground();

} else {
    Log.d(TAG, "Unable to get count, not saving expression.");
}

ParseObject allowed addition of any key value pair. Keys must be Strings.
saveEquation

• Makes a query to get the number of rows in the expression table
  – uses another table with one row with one column (GACK, no auto increment function)
• callback method for completed query
• checks the count
• creates new ParseObject
• makes the index for this new expression the count (0 based indexing)
• saves the object and updates count object
Parse Dashboard

• Examine data uploaded from apps
demo Saving an Equation
Get Random Saved Art

• When user presses button pick a random saved expression and render that image
• We just save the expression so we must recreate image
  – time vs. space trade off
• check count of values and pick random index
public void getRandomGoodArt(View v) {
    pickRandomExpression = false;

    ParseQuery<ParseObject> countQuery
        = ParseQuery.getQuery("ArtExpressionCount");

    countQuery.getFirstInBackground(new GetCallback<ParseObject>() {
        @Override
        public void done(ParseObject masterCount, ParseException e) {
            if (e == null) {
                int count = masterCount.getInt("TheCount");
                int randomIndex = r.nextInt(count);
                Log.d(TAG, "The Count via the master count object: " + count);

                ParseQuery<ParseObject> query
                    = ParseQuery.getQuery("ArtExpression");
                query.whereGreaterThanOrEqualTo("index", randomIndex);
                query.getFirstInBackground(setRandomExpressionFromQuery);
            } else {
                Log.d(TAG, "Unable to get count to get random expression" + e.getMessage());
            }
        }
    });
}
call back object

• pull out the String from the returned object and build expression based on equation

```java
private GetCallback<ParseObject> setRandomExpressionFromQuery
    = new GetCallback<ParseObject>() {
        public void done(ParseObject object, ParseException e) {
            if (e == null) {
                String equation = object.getString("equation");
                RandomExpression exp = new RandomExpression(equation);
                // now draw it
                Log.d(TAG, "equation: " + equation);
                Log.d(TAG, "index of expression: " + object.getInt("index");
                new ArtTaskInner().execute(artImage.getWidth(), artImage.getHeight());
            } else {
                Log.d(TAG, "Unable to get the given random expression");
            }
        }
    };
```
good one logcat

equation: yCQyCxCAMCSCCCxCCCSMS
index of expression: 3

equation: yCQyCxCAMCSCCCxCCCSMS
index of expression: 3

<table>
<thead>
<tr>
<th>objectId</th>
<th>String</th>
<th>createdAt</th>
<th>Date</th>
<th>updatedAt</th>
<th>Date</th>
<th>equation</th>
<th>String</th>
<th>index</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnfckzAUx4</td>
<td></td>
<td>Nov 11, 2014</td>
<td>22:45</td>
<td>Nov 11, 2014</td>
<td>22:45</td>
<td>xCCCSSyCySSSCySSSSMMASSS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>LD17JULs1s</td>
<td></td>
<td>Nov 11, 2014</td>
<td>22:53</td>
<td>Nov 11, 2014</td>
<td>22:53</td>
<td>yCSryxOMOMyQCCAAACxMC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Oi0XTrmtMc</td>
<td></td>
<td>Nov 11, 2014</td>
<td>23:03</td>
<td>Nov 11, 2014</td>
<td>23:03</td>
<td>yCQyCxCAMCSCCCxCCCSMS</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
More Parse

- Includes capability to do local data store
  - save objects on device, save to cloud later
  - abstracts away a lot of the details
  - Kyle Norton from Pariveda: "Assume you WON'T be connected to the network."

- Parse objects meant to be "small"
  - less than 128 kb
  - not for images
  - Parse files for large pieces of data

- Past semesters many groups used Parse successfully
FIREBASE
Firebase

• Yet another Backend as a Service (Baas)
• Designed for web and mobile
• Founded in 2011
• Initial product was backend so websites could easily host chat as part of site
• discovered developers were sending non chat data (such as game state) via the tool
Firebase for Android

- Devices with Android 4.0 (ice cream sandwich) or higher
- Google play services SDK
  - same as fused location
- Android Studio 1.5 or higher
- Your Android studio project and package name
- Firebase Assistant in Android Studio 2.2 or higher
  - Tools -> Firebase
Firebase Project Set up

• Create Firebase project in console
• Just needs name and country
Firebase Project Console

• After creating project, overview page:

Welcome to Firebase! Get started here.

Add Firebase to your Android app
Add Firebase to your iOS app
Add Firebase to your web app
Firebase for Android Project

• Adding Firebase to Android app
• Need package name (easy)
• Debug signing certificate SHA-1 hash (for use of some Firebase features)
• Uses the keytool program included with Java
  – "Manages a keystore (database) of cryptographic keys, X.509 certificate chains, and trusted certificates."
Adding Firebase to Android App

Package name: examples.scottm.hellofirebase


Required for Dynamic Links, Invites, and Google Sign-In support in Auth. Edit SHA-1s in Settings.
Using keytool

• Varies from system to system
• need location of debug signing certificate
  – created automatically when Android Studio installed
• typically,  
  <USER>/.android/debugkeystore
Debug Signing Certificate

- certificate not human readable
Running keytool

• Specifics vary from system to system

To get the debug certificate fingerprint:

<table>
<thead>
<tr>
<th>MAC/LINUX</th>
<th>WINDOWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>keytool -exportcert -list -v \</td>
<td></td>
</tr>
<tr>
<td>-alias androiddebugkey -keystore</td>
<td></td>
</tr>
<tr>
<td>~/.android/debug.keystore</td>
<td></td>
</tr>
</tbody>
</table>

The keytool utility prompts you to enter a password for the keystore. The default password for the debug keystore is `android`. The keytool then prints the fingerprint to the terminal. For example:

Firebase Config File for App

• After providing package name and SHA-1 fingerprint ...
• Firebase generates a JSON file named google-services.json specific for this project
  – multiple projects / apps -> repeat steps
• Download and add file to project
Firebase Config File for App

Add Firebase to your Android app

1. Enter app details
2. Copy config file
3. Add to build.gradle

Switch to the Project view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.

Already added the dependencies? 
Skip to the console
```
{
    "project_info": {
        "project_number": "489833291042",
        "firebase_url": "https://hello-firebase-cb60f.firebaseapp.com",
        "project_id": "hello-firebase-cb60f",
        "storage_bucket": "hello-firebase-cb60f.appspot.com"
    },
    "client": [
        {
            "client_info": {
                "mobilesdk_app_id": "1:489833291042:android:69b93ad9212",
                "android_client_info": {
                    "package_name": "examples.scottm.hellofirebase"
                }
            },
            "oauth_client": [
                {
                    "client_id": "489833291042-eptigvdv48scbc60brllsaq",
                    "client_type": 1,
                    "android_info": {
                        "client_info": {
                            "android_client_info": {
                                "package_name": "examples.scottm.hellofirebase"
                            }
                        }
                    }
                }
            ]
        }
    ]
}
```
Update Gradle Files

The Google services plugin for Gradle loads the google-services.json file you just downloaded. Modify your build.gradle files to use the plugin.

1. Project-level build.gradle (<project>/build.gradle):

```gradle
buildscript {
  dependencies {
    // Add this line
    classpath 'com.google.gms:google-services:3.0.0'
  }
}
```

2. App-level build.gradle (<project>/<app-module>/build.gradle):

```gradle
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

3. Finally, press "Sync now" in the bar that appears.
Firebase Capabilities

• Firebase has a host of capabilities
• User authorization
• database storage
• storage for larger files
• cloud messaging
• push notifications
• analytics
• hosting of web content
Firebase Database

• With Parse offline, migrated Random Art database to Firebase
• The roots of the chat room are somewhat apparent
  – lots of chat examples
  – realtime updates
  – emphasis on authorized users

Default security rules require users to be authenticated
One More Setup Step

• To use Firebase databases in app, after previous setup steps:

Add the Realtime Database to your app

Add the dependency for Firebase Realtime Database to your app-level `build.gradle` file:

```groovy
compile 'com.google.firebase:firebase-database:9.2.0'
```
Firebase Database Rules

• Firebase database rules
• Defines:
  • How data should be structured
  • How data should be indexed
  • When data can be read or written
  • Who can read and write data
// These rules require authentication
{
    "rules": {
        ".read": "auth != null",
        ".write": "auth != null"
    }
}

// These rules give anyone, even people who are not users of your app, // read and write access to your database
{
    "rules": {
        ".read": true,
        ".write": true
    }
}
Hello Firebase

- In app, called from `onCreate` of Activity

```java
private void testFirebase() {
    // Write a message to the database
    FirebaseDatabase database
        = FirebaseDatabase.getInstance();
    DatabaseReference myRef
        = database.getReference("test message 1252");
    myRef.setValue("Hello, Firebase!!!!!!!!");
}
```
Result When App Run

- Immediately writes to database if network connections exists:

```
https://hello-firebase-cb60f.firebaseapp.com/

hello-firebase-cb60f

  test message 1252: "Hello, Firebase!!!!!!!"
```
Firebase database

- Not traditional tables
- "Everything is a JSON! tree"
- Children of main tree are like "tables" in traditional database
- Children of children are typically (but not always) like rows in a traditional table
Random Art Data on Firebase

• equation count child to assign ids and pick random equation

• equation list with children for each equation
Random Art

• App keeps track of current equation count
• First value from database and listener so whenever count changes, local copy is updated
Random Art

• Keep references to parts of JSON tree
• Update values (equation count)
• add children (new, good equations)
• pull random children (old, good equations)
• In main Random Art Activity

```java
private DatabaseReference equationListDatabase;
private DatabaseReference equationCountDatabase;
private int equationCount;
```
Random Art - Count Listener

```java
ValueEventListener postListener = new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        Log.d(TAG, "onDataChanged call for Value Event Listener");
        equationCount = ((Long) dataSnapshot.getValue()).intValue();
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.w(TAG, "loadPost: onCancelled", databaseError.toException());
        // ...
    }
};

equationCountDatabase.addValueEventListener(postListener);
```
Random Art - Save Equation

currentExpressionIsNew = false;
int newCount = equationCount + 1;
Log.d(TAG, "Setting new count. Old count: " + equationCount + ", newCount: " + newCount);
equationCountDatabase.setValue(newCount);
String equation = exp.toString();
// Add current equation to Firebase database
EquationForStorage newExpression = new EquationForStorage(equation, newCount, 1, 0, System.currentTimeMillis());
equationListDatabase.child("" + newCount).setValue(newExpression);

• setValue method to add child
• Firebase data: String, Long, Double, Boolean, Map<String, Object>, List<Object>
• any custom object with 0 argument constructor and public getters for properties
Random Art - Get Equation

• Pick random child based on current number of equations

```java
int randomID = r.nextInt(equationCount);
equationListDatabase.child(randomID + "").addListenerForSingleValueEvent(
    new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            // Get user value
            EquationForStorage eq = dataSnapshot.getValue(EquationForStorage.class);
            Log.d(TAG, "read expression: " + eq.getEquation());
            exp = new RandomExpression(eq.getEquation());
            // now draw it
            Log.d(TAG, "index / id of expression: " + eq.getId());
            new ArtTaskInner().execute(artImage.getWidth(), artImage.getHeight());
        }
    }
    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.w(TAG, "getUser:onCancelled", databaseError.toException());
    }
```
JSON
JSON

• JavaScript Object Notation
• a way to represent JavaScript objects as Strings
• alternative to XML for passing data between servers and clients
• designed for data interchange format that humans can also read and write
JSON Format

• Built on two structures
  – collection of name-value pairs: a.k.a. objects, records, structs, etc.
  – an ordered list of values: a.k.a. an array

• objects
JSON Format

• arrays

• values
  – string, number, object, array, true, false, null
Syntax Diagrams for string and number: [http://www.json.org/](http://www.json.org/)
JSON Strings

Any UNICODE character except " or \ or control character

quotation mark
reverse solidus
solidus
backspace
formfeed
newline
carriage return
horizontal tab
4 hexadecimal digits
JSON Numbers
JSON Examples

• value (String):
  – "Round Rock"

• array:
  – ["Round Rock", "Dallas", "Houston"]

• object
  –{"height":70,"weight":165}