Network Communication

It’s time to “talk” to the outer world. In this chapter we will learn how to communicate with services over the Internet.
What is HTTP

- HTTP = Hypertext Transfer Protocol
- Request-Response protocol on Client-Server model
- A response contains completion status information and optional data
What is HTTP
Android Threading Model

- All the application GUI runs on the UI thread
- Trying to access UI not from the UI thread will cause crash
- If the UI thread is stuck, the app is frozen
- If the app is not responding for more than 5 seconds, Android send the user ANR message
- Android does not allow doing network operations on main thread
The ANR (Application not responding) message is disabled by default on the device settings. Turn it on and you will see how many bad applications make ANR.
AsyncTask helps us run tasks in the background

```java
// Execute the task (From the UI thread)
(new DownloadImageTask()).execute("string parameter");

private class DownloadImageTask extends AsyncTask<String, Void, String> {
    protected String doInBackground(String... urls) {
        // This is done in a different thread
        return "string result";
    }

    protected void onPostExecute(String result) {
        // This is done in the UI thread
    }
}
```
HTTP Get

- Retrieve information from a specified URI
- Data in the form of parameters in URI

Example:

String lineRead = "";

try {
    URL url = new URL("http://www.11sheep.com");
    URLConnection connection = url.openConnection();
    DataInputStream dataIn = new DataInputStream(connection.getInputStream());
    lineRead = dataIn.readLine();
} catch (IOException e) {
    e.printStackTrace();
}
Create a new project
In the first step, get all the animals list from the server
Present the animal list in a list
When the user selects animal get animal sound from the server
Advanced – make a preloader while networking for better UI experience

URL to get animals:
Ex14 – Animals Festival

- Sheep
- Cow
- Dog
- Cat

Sheep sound is: Beh, Beh

Close
Ex – Creating applications with parse
Step 1 – Create parse project

- Create account at parse.com
- Create a new application
Step 2 – Get the application key

Your app MyCoolApp has been successfully created. Learn how to use Parse in your app using our Quickstart Guide, or start designing your app in the Data Browser.
Step 3 - Install the SDK

1. Download & unzip the SDK
   Make sure you are targeting Gingerbread (android-9) or higher.
   
   Download the SDK

2. Add the SDK to your app in Android Studio
   Drag the Parse-* jar you downloaded into your existing app's "libs" folder and add the following to your build.gradle

   ```groovy
   dependencies {
       compile 'com.parse.bolts:bolts-android:1.+'
       compile fileTree(dir: 'libs', include: 'Parse-*.*')
   }
   ```
Step 4 – Connect app to Parse

Connect your app to Parse

Before continuing, select your Parse app from the menu at the right. These steps are for your “MyCoolApp” app.

Your app must request the INTERNET and ACCESS_NETWORK_STATE permissions, if it isn’t doing so already. Add the following lines inside the <manifest> tag in your AndroidManifest.xml:

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

Add the following to your Application#onCreate():

```java
// Enable Local Datastore.
Parse.enableLocalDatastore(this);

Parse.initialize(this, "<YOUR_APP_NAME>");
```

Compile and run!
Save data to parse

```java
ParseObject sheepObject = new ParseObject("SheepTable");
sheepObject.put("SheepName", "Shifra");
sheepObject.put("SheepAge", 13);
sheepObject.saveInBackground();
```
Save data to parse + callback

ParseObject sheepObject = new ParseObject("SheepTable");
sheepObject.put("SheepName", "Shifra");
sheepObject.put("SheepAge", 13);
sheepObject.saveInBackground(new SaveCallback() {
    @Override
    public void done(ParseException e) {
        if (null == e) {
            // everything is ok, the data is saved
        }
        else {
            // Something went wrong
            Log.e("MY_TAG", "The error: " + e.getMessage());
        }
    }
});
saveEventually

ParseObject sheepObject = new ParseObject("SheepTable");
sheepObject.put("SheepName", "Shifra");
sheepObject.put("SheepAge", 13);
sheepObject. saveEventually();
ParseQuery<ParseObject> query = ParseQuery.getQuery("SheepTable");
query.findInBackground(new FindCallback<ParseObject>() {
    public void done(List<ParseObject> sheepList, ParseException e) {
        if (e == null) {
            Log.d("MY_TAG", "Retrieved " + sheepList.size() + " sheep");

            for (int index = 0; index < sheepList.size(); index++) {
                Log.d("MY_TAG", "Sheep name: " + sheepList.get(index).get("SheepName"));
                Log.d("MY_TAG", "Sheep age: " + sheepList.get(index).get("SheepAge"));
            }
        } else {
            Log.d("MY_TAG", "Error: " + e.getMessage());
        }
    }
});
Now we can make the sheep dating application with real remote server and database instead of local database.
Android Push Notifications
Parse Push Notifications

- [https://www.parse.com/docs/push_guide](https://www.parse.com/docs/push_guide)