

# Background Tasks

Lesson 7



Internet  
connection

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).



# 7.2 Internet connection

# Steps to connect to the Internet

1. Add permissions to Android Manifest
2. Check Network Connection
3. Create Worker Thread
4. Implement background task
  - a. Create URI
  - b. Make HTTP Connection
  - c. Connect and GET Data
5. Process results
  - a. Parse Results



# Permissions



# Permissions in AndroidManifest

## Internet

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

## Check Network State

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



# Manage Network Connection



# Getting Network information

- [ConnectivityManager](#)
  - Answers queries about the state of network connectivity
  - Notifies applications when network connectivity changes
- [NetworkInfo](#)
  - Describes status of a network interface of a given type
  - Mobile or Wi-Fi



# Check if network is available

```
ConnectivityManager connMgr = (ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

if (networkInfo != null && networkInfo.isConnected()) {
    // Create background thread to connect and get data
    new DownloadWebpageTask().execute(stringUrl);
} else {
    textView.setText("No network connection available.");
}
```

# Check for WiFi & Mobile

```
NetworkInfo networkInfo =  
    connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);  
boolean isWifiConn = networkInfo.isConnected();  
  
networkInfo =  
    connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);  
boolean isMobileConn = networkInfo.isConnected();
```

# Worker Thread



# Use Worker Thread

- AsyncTask—very short task, or no result returned to UI
- AsyncTaskLoader—for longer tasks, returns result to UI
- Background Service—later chapter



# Background work

In the background task (for example in `doInBackground()`)

1. Create URI
2. Make HTTP Connection
3. Download Data



# Create URI



# URI = Uniform Resource Identifier

String that names or locates a particular resource

- file://
- http:// and https://
- content://



# Sample URL for Google Books API

[https://www.googleapis.com/books/v1/volumes?  
q=pride+prejudice&maxResults=5&printType=books](https://www.googleapis.com/books/v1/volumes?q=pride+prejudice&maxResults=5&printType=books)

## Constants for Parameters

```
final String BASE_URL =  
    "https://www.googleapis.com/books/v1/volumes?";  
  
final String QUERY_PARAM = "q";  
  
final String MAX_RESULTS = "maxResults";  
  
final String PRINT_TYPE = "printType";
```



# Build a URI for the request

```
Uri builtURI = Uri.parse(BASE_URL).buildUpon()  
    .appendQueryParameter(QUERY_PARAM, "pride+prejudice")  
    .appendQueryParameter(MAX_RESULTS, "10")  
    .appendQueryParameter(PRINT_TYPE, "books")  
    .build();  
  
URL requestURL = new URL(builtURI.toString());
```



# HTTP Client Connection

Internet  
connection

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).



# Make a connection from scratch

- Use [HttpURLConnection](#)
- Must be done on a separate thread
- Requires InputStreams and try/catch blocks



# Create a HttpURLConnection

```
HttpURLConnection conn =  
    (HttpURLConnection) requestURL.openConnection();
```

# Configure connection

```
conn.setReadTimeout(10000 /* milliseconds */);  
conn.setConnectTimeout(15000 /* milliseconds */);  
conn.setRequestMethod("GET");  
conn.setDoInput(true);
```



# Connect and get response

```
conn.connect();
int response = conn.getResponseCode();

InputStream is = conn.getInputStream();
String contentAsString = convertIsToString(is, len);
return contentAsString;
```

# Close connection and stream

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



# Convert Response to String



# Convert input stream into a string

```
public String convertIsToString(InputStream stream, int len)
    throws IOException, UnsupportedEncodingException {

    Reader reader = null;
    reader = new InputStreamReader(stream, "UTF-8");
    char[] buffer = new char[len];
    reader.read(buffer);
    return new String(buffer);
}
```



# BufferedReader is more efficient

```
StringBuilder builder = new StringBuilder();
BufferedReader reader =
    new BufferedReader(new InputStreamReader(inputStream));
String line;
while ((line = reader.readLine()) != null) {
    builder.append(line + "\n");
}
if (builder.length() == 0) {
    return null;
}
resultString = builder.toString();
```



# HTTP Client Connection Libraries



# Make a connection using libraries

- Use a third party library like [OkHttp](#) or [Volley](#)
- Can be called on the main thread
- Much less code



# Volley

```
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";

StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        // Do something with response
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {}
});
queue.add(stringRequest);
```



# OkHttp

```
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder()
    .url("http://publicobject.com/helloworld.txt").build();
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onResponse(Call call, final Response response)
        throws IOException {
        try {
            String responseData = response.body().string();
            JSONObject json = new JSONObject(responseData);
            final String owner = json.getString("name");
        } catch (JSONException e) {}
    }
});
```



# Parse Results



# Parsing the results

- Implement method to receive and handle results  
(`onPostExecute()`)
- Response is often JSON or XML

Parse results using helper classes

- [JSONObject](#), [JSONArray](#)
- [XMLPullParser](#)—parses XML



# JSON basics

```
{  
  "population":1,252,000,000,  
  "country":"India",  
  "cities":["New Delhi","Mumbai","Kolkata","Chennai"]  
}
```

# JSONObject basics

```
JSONObject json0bject = new JSONObject(response);
String nameOfCountry = (String) json0bject.get("country");
long population = (Long) json0bject.get("population");
JSONArray list0fCities = (JSONArray) json0bject.get("cities");
Iterator<String> iterator = list0fCities.iterator();
while (iterator.hasNext()) {
    // do something
}
```



# Another JSON example

```
{"menu": {  
    "id": "file",  
    "value": "File",  
    "popup": {  
        "menuitem": [  
            {"value": "New", "onclick": "CreateNewDoc()"},  
            {"value": "Open", "onclick": "OpenDoc()"},  
            {"value": "Close", "onclick": "CloseDoc()"}  
        ]  
    }  
}
```

# Another JSON example

Get "onclick" value of the 3rd item in the "menuitem" array

```
JSONObject data = new JSONObject(responseString);
JSONArray menuItemArray =
    data.getJSONArray("menuitem");
JSONObject thirdItem =
    menuItemArray.getJSONObject(2);
String onClick = thirdItem.getString("onclick");
```

# Learn more

- [Connect to the Network Guide](#)
- [Managing Network Usage Guide](#)
- [HttpURLConnection reference](#)
- [ConnectivityManager reference](#)
- [InputStream reference](#)



# What's Next?

- Concept Chapter: [7.2 Internet connection](#)
- Practical: [7.2 AsyncTask and AsyncTaskLoader](#)



# END



# Exercise 01

