The Internets

HttpUrlConnection

Http what?

- Class used to send and receive data from the web
- HttpsURLConnection for secure connections
- Fetch raw data which can be easily converted to an array of Bytes

Steps for using HttpURLConnection

1. Obtain a new HttpURLConnection by calling URL.openConnection() and casting the result to HttpURLConnection.
2. Prepare request - URI, HTTPHeaders, Credentials, Content-Type, Session Cookies
3. Upload a request body (if necessary)
4. Read the Response

Code

```
private byte[] getURLBytes(String urlString) throws IOException{
            URL url = new URL(urlString);
            HttpURLConnection connection = (HttpURLConnection)url.openConnection();

            try{
                ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
                BufferedInputStream in = new BufferedInputStream(connection.getInputStream());

                if (connection.getResponseCode() != HttpURLConnection.HTTP_OK) {
                    return null;
                }

                int bytesRead = 0;
                byte[]buffer = new byte[1024];

                while ((bytesRead = in.read(buffer)) > 0) {
                    outputStream.write(buffer, 0, bytesRead);
                }
                outputStream.close();
                return outputStream.toByteArray();
            } finally {
                connection.disconnect();
            }
        }
```

InputStream / BufferedInputStream

- InputStream - A Readable source of bytes
- BufferedInputStream - Wraps an InputStream and buffers the input.

```
BufferedInputStream in = new BufferedInputStream(connection.getInputStream());
```

ByteArrayOutputStream

A specialized OutputStream for writing content to an (internal) byte array. As bytes are written to this stream, the byte array may be expanded to hold more bytes. When the writing is considered to be finished, a copy of the byte array can be requested from the class.

AsyncTask

UI Thread

- In Android, all UI work happens on the Main thread or "UI Thread"
- Main is responsible for dispatching events to Widgets
- Long running operations should happen on worker threads or your app risks crashing
- Web requests should always be on worker threads as they will block event dispatch to UI

AsyncTask

- Allows access to worker threads with a simple callback to the UI thread
- Must be subclassed to be used
- Configured with 3 generic types to enable additional functionality....

AsyncTask Generic Types

- Params - The type of the parameters sent to the task upon execution.
- Progress - The type of the progress units published during the background computation.
- Result - The type of the result of the background computation

Code

```
 private class MyTask extends AsyncTask<Void, Void, Void> { ... } // an async task with Void types

  private class MyTask extends AsyncTask<URL, Integer, String> {  // Params, Progress, Result

    @Override
    protected void onPreExecute(){
        //perform work on UI Thread after background operation completes
    }

    @Override
    protected String doInBackground(URL... urls) {
        //Network Ops invoked on worker thread
    }

    @Override
    protected void onProgressUpdate(Integer... progress) {
        //Invoked on UI thread after call to publicProgress(Progress ...)
    }

    @Override
    protected void onPostExecute(String result){
        //Invoked on UI Thread after background operation completes, result passed in
    }
}
```

DEMO