CMPS 121
Internet and WebApps

Luca de Alfaro

Portions of this presentations are based on content taken from the Android Developers Site. The work was created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.
Opening a data stream

```java
String myFeed = getString(R.string.my_feed);
try {
    URL url = new URL(myFeed);

    URLConnection connection = url.openConnection();
    HttpURLConnection httpConnection = (HttpURLConnection)connection;

    int responseCode = httpConnection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_OK) {
        InputStream in = httpConnection.getInputStream();
        [ ... Process the input stream as required ... ]
    }
}
catch (MalformedURLException e) { }
catch (IOException e) { }
```
Monitoring internet connectivity

There is a number of broadcast events that alert you to when there is a WiFi in range, connected, when it disconnects, etc.
WebViews

WebViews are "browser windows" that you can open in your app.
You can define the size of their "viewport", defining the relationship between remote screen and local web screen (see the developer guide).
Use them if:

- You already have a web application you want to access
- You need to display information that changes
Adding a WebView

Simply include the `<WebView>` element in the layout:

```xml
<?xml version="1.0" encoding="utf-8"?>
<WebView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />
```

To load a page, use `loadUrl()`:

```java
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```

And ask for INTERNET permissions in the manifest:

```xml
<manifest ...
    <uses-permission android:name="android.permission.INTERNET" />
... ...
</manifest>
```
Enabling Javascript in a WebView

```java
WebView myWebView = (WebView) findViewById(R.id.webview);
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```
Define a class in your Android app that will be made visible to the page javascript:

```java
public class JavaScriptInterface {
    Context mContext;

    /** Instantiate the interface and set the context */
    JavaScriptInterface(Context c) {
        mContext = c;
    }

    /** Show a toast from the web page */
    public void showToast(String toast) {
        Toast.makeText(mContext, toast, Toast.LENGTH_SHORT).show();
    }
}
```
Binding JavaScript to Android code

Declare the binding:

```java
WebView webView = (WebView) findViewById(R.id.webview);
webView.addJavascriptInterface(new JavaScriptInterface(this), "Android");
```

And use it from JavaScript:

```html
<input type="button" value="Say hello" onClick="showAndroidToast('Hello Android!')" />
```

```javascript
function showAndroidToast(toast) {
    Android.showToast(toast);
}
</script>
Handling page navigation

Normally, a click on a URL calls an intent to view webpages, and takes you out of your application.
You can override this by providing a WebViewClient:

```java
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebViewClient(new WebViewClient());
```

That makes all links open in the WebView!! (why??)
Handling page navigation

You can also have more control, by creating your own `WebViewClient` and override `shouldOverrideUrlLoading()`:

```java
private class MyWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        if (Uri.parse(url).getHost().equals("www.example.com")) {
            // This is my web site, so do not override; let my WebView load the page
            return false;
        }
        // Otherwise, the link is not for a page on my site, so launch another Activity that handles URLs
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
        startActivity(intent);
        return true;
    }
}

WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebViewClient(new MyWebViewClient());
```
Authentication to web service

A simple solution:
- Store the user's username and password on the device.
- Send the user's username and password via SSL with every request as HTTP Authorization: header.

Another alternative:
- Send username and password once (using SSL).
- Get an authorization token (long, hashed, etc).
- Send the token instead.