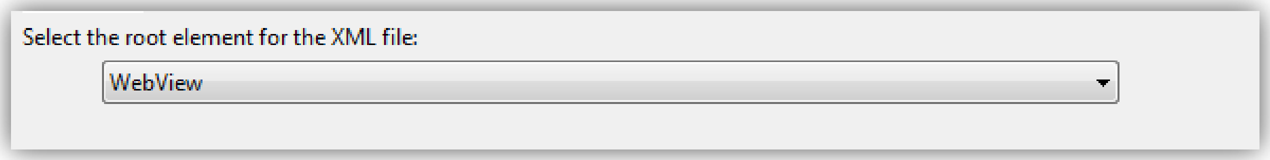
# Lesson 6 – User Interfaces Part 6

In this part, we're going to take a look at the WebView control. The WebView is a web browser component that can be utilized by our application.

Open the existing Lesson5_UI project, and create a new Layout file (see Part 2, if you're not sure how) called **webview_example.xml**. Unlike our other examples, it will simply contain a single WebView instance. So make sure you select WebView for the root element selection.



Select the root element for the XML file:

WebView

Modify the WebView's properties and make sure that it's ID is set to **@+id/WebView01**.

Save and close the XML file.

Open the WebView_example.java file. Add the following code:

```
public class webview_example extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.webview_example);

        final WebView oWeb = (WebView) findViewById(R.id.WebView01);
        oWeb.getSettings().setJavaScriptEnabled(true);
        oWeb.loadUrl("http://www.google.com");
        oWeb.setWebViewClient(new oWebViewClient());

    }

    private class oWebViewClient extends WebViewClient {
      @Override
      public boolean shouldOverrideUrlLoading(WebView view, String url) {
            view.loadUrl(url);
            return true;
      }
    }

}
```

Here’s what it does:

In the onCreate event, we get an instance of our WebView01 control from the XML file, and enable its ability to work with JavaScript (setJavaScriptEnabled()). Then, we tell it to load a URL (loadUrl()). This can be any URL, including an internal one to our application.

The next line, oWeb.setWebViewClient(new oWebViewClient(), is necessary to capture the click events of the browser. If we didn’t do this, clicking on a link inside of our application would trigger the default browser on our device, outside of our application. By overriding the shouldOverrideUrlLoading method, we are able to capture and process links within our application, and send them to our WebView, instead of the outside browser.