WebView

Embedding the WebKit Browser

- WebView is a view that display web pages inside your application. You can also specify HTML string and can show it inside your application using WebView. WebView turns your application to a web application.
- You can embed the built-in Web browser as a widget in your own activities, for displaying HTML or full-fledged browsing.
- The Android browser is based on WebKit.
- The Android browser is sufficiently complex that it gets its own Java package **(android.webkit)**, though using the WebView widget itself can be simple or powerful, based upon your requirements.

```
<WebView
  android:id="@+id/webview1"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent” />

public class BrowserDemo1 extends AppCompatActivity {
WebView browser;

@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
browser=(WebView)findViewById(R.id.webview1);
browser.loadUrl(“www.google.com”);
}
```

make one change to AndroidManifest.xml, requesting permission to access the Internet:

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.commonsware.android.webkit">
<uses-permission android:name="android.permission.INTERNET" />
<application>
<activity android:name=".BrowserDemo1"
android:label="BrowserDemo1">
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
</application>
</manifest>
```

Loading Data into webView

- There are two main ways to get content into the WebView.
- One, shown earlier, is to provide the browser with a URL and have the browser display that page via **loadUrl().**
- HTML web page can also be accessed by a local file. For this, the html file must be located inside the asset directory.

```
WebView mywebview = findViewById(R.id.webView1);
mywebview.loadUrl("file:///android_asset/myresource.html");
```

- The browser will access the Internet through whatever means are available to that specific device at the present time (WiFi, cellular network, Bluetooth-tethered phone, well-trained tiny carrier pigeons, etc.).
- The alternative is to use **loadData().**

- **There are two flavors of loadData():**
  - The simpler one allows you to provide the content, the MIME type, and the encoding, all as strings.
  - Typically, your MIME type will be text/html and your encoding will be UTF-8 for ordinary HTML.

```
browser.loadData("<html><body>Hello, world!
</body></html>“, "text/html", "UTF-8");
```

You might use this to:

- display a manual that was installed as a file with your application package
- display snippets of HTML you retrieved as part of other processing, such as the description of an entry.
- generate a whole user interface using HTML, instead of using the Android widget set

Navigating

- WebView offers ways to perform browser navigation, including the following:
  - **reload()** to refresh the currently-viewed Web page
  - **goBack()** to go back one step in the browser history, and canGoBack() to determine if there is any history to go back to
  - **goForward()** to go forward one step in the browser history, and canGoForward() to determine if there is any history to go forward to
  - **goBackOrForward()** to go backward or forward in the browser history, where a negative number as an argument represents a count of steps to go backward, and a positive number represents how many steps to go forward
  - **canGoBackOrForward()** to see if the browser can go backward or forward the stated number of steps (following the same positive/negative convention as goBackOrForward())
  - **clearCache()** to clear the browser resource cache and **clearHistory()** to clear thebrowsing history

Entertaining the Client

- If you are going to use the WebView as a local user interface (vs. browsing the Web), you will want to be able to get control at key times, particularly when users click on links.
- You will want to make sure those links are handled properly, either by loading your own content back into the WebView, by submitting an Intent to Android to open the URL in a full browser, or by some other means.
- That is possible with setWebViewClient().

```
public class BrowserDemo3 extends Activity {
        WebView browser;
        @Override
        public void onCreate(Bundle icicle) {
                super.onCreate(icicle);
                setContentView(R.layout.main);
                browser=(WebView)findViewById(R.id.webkit);
                browser.setWebViewClient(new MyWebViewClient());
}

private class MyWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        if ("www.example.com".equals(Uri.parse(url).getHost())) {
            // This is my website, 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;
    }
}
```