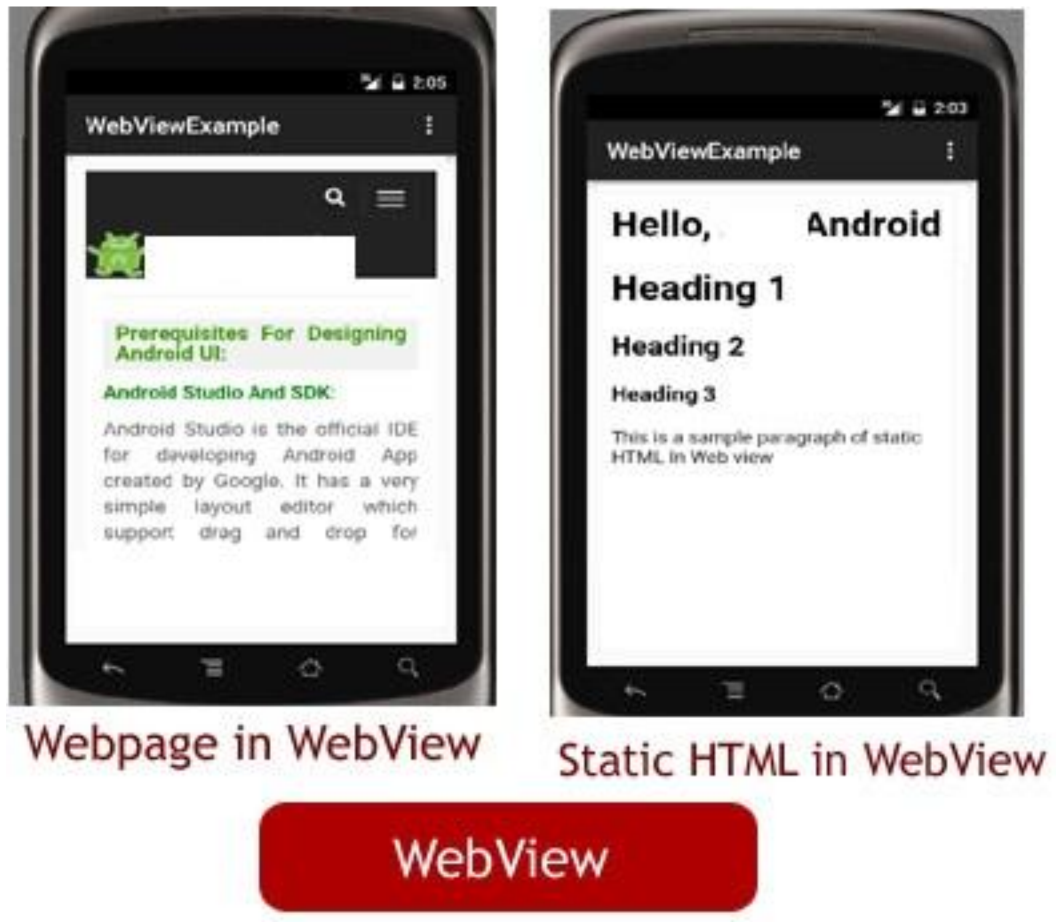
WebView

In Android, WebView is a view used to display the web pages in application. This class is the basis upon which you can roll your own web browser or simply use it to display some online content within your Activity. We can also specify HTML string and can show it inside our application using a WebView. Basically, WebView turns application into a web application.



Android Studio is the official IDE for developing Android App created by Google. It has a very simple layout editor which support drag and drop for

This is a sample paragraph of static HTML in Web view

In order to add Web View in your application, you have to add **<WebView>** element to your XML(layout) file or you can also add it in java class.

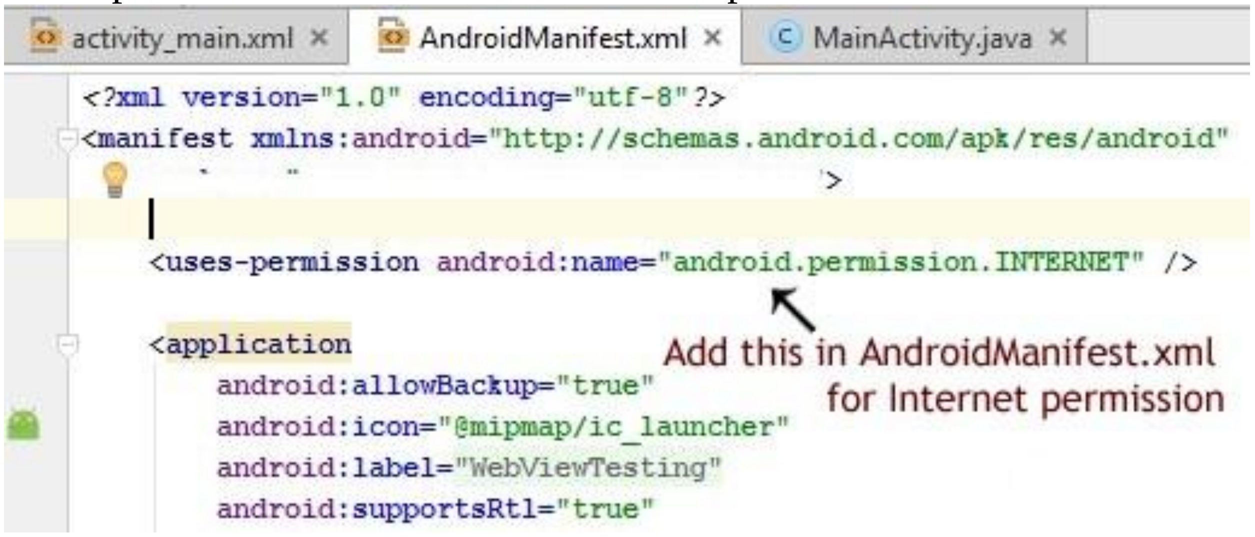
```
<WebView
android:id="@+id/simpleWebView"
android:layout_width="fill_parent"
android:layout_height="fill_parent" />
```

Internet Permission Required For Webview:

Important Note: In order for Activity to access the Internet and load the web pages in a WebView, we must add the internet permissions to our Android Manifest file (Manifest.xml).

Below code define the internet permission in our manifest file to access the internet in our application.

```
<!--Add this before application tag in AndroidManifest.xml-->
<uses-permission android:name="android.permission.INTERNET" />
```



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        >

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="WebViewTesting"
        android:supportsRtl="true"
```

Methods of WebView In Android:

Let’s discuss some common methods of a Webview which are used to configure a web view in our application.

loadUrl() – Load a web page in our WebView

loadUrl(String url)

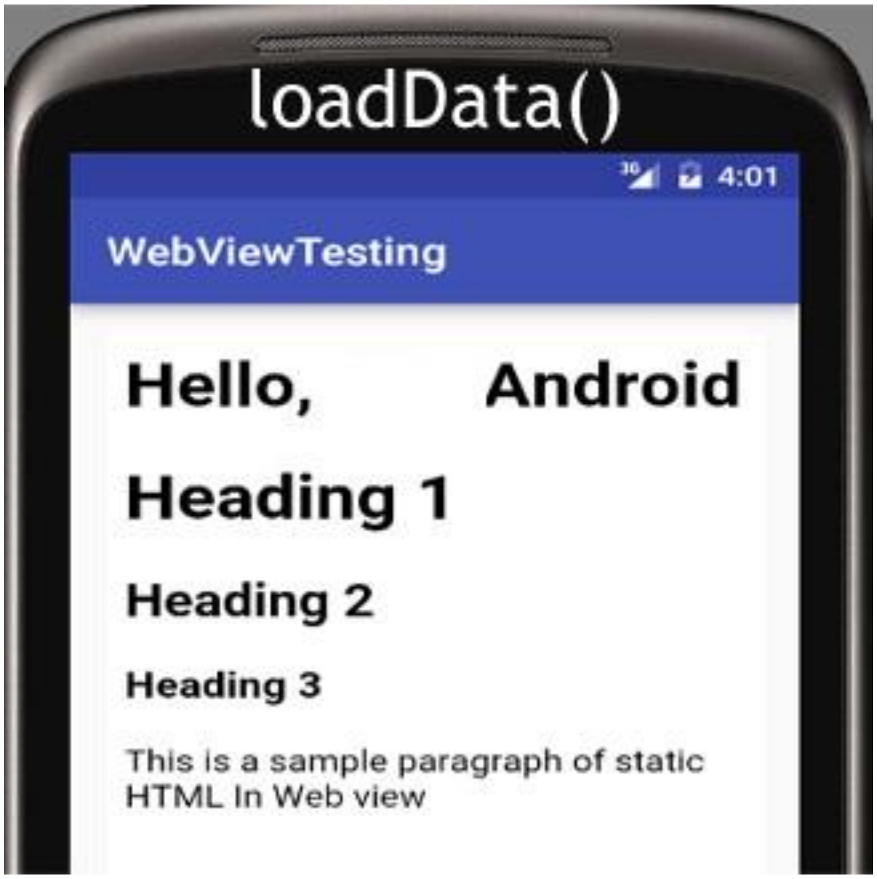
This function is used to load a web page in a web view of our application. In this method we specify the url of the web page that should be loaded in a web view.

```
/*Add in Oncreate() funtion after setContentView()*/
// initiate a web view
WebView simpleWebView=(WebView)
findViewById(R.id.simpleWebView);
// specify the url of the web page in loadUrl function
simpleWebView.loadUrl("https://rpbc.com/ui/");
```

2. loadData() – Load Static Html Data on WebView
loadData(String data, String mimeType, String encoding)

This method is used to load the static HTML string in a web view. loadData() function takes html string data, mime-type and encoding param as three parameters.

```
/*Add in Oncreate() funtion after setContentView()*/
// initiate a web view
WebView webView = (WebView)
findViewById(R.id.simpleWebView);
// static html string data
String customHtml = "<html><body><h1>Hello,       Android</h1>" +
        "<h1>Heading 1</h1><h2>Heading 2</h2><h3>Heading 3</h3>"
+
        "<p>This is a sample paragraph of static HTML In Web view</p>"
+
        "</body></html>";
// load static html data on a web view
webView.loadData(customHtml, "text/html", "UTF-8");
```



Hello, Android

Heading 1

Heading 2

Heading 3

This is a sample paragraph of static HTML In Web view

3. Load Remote URL on WebView using WebViewClient:

WebViewClient help us to monitor event in a WebView. You have to Override the shouldOverrideUrlLoading() method. This method allow us to perform our own action when a particular url is selected. Once you are ready with the WebViewClient, you can set the WebViewClient in your WebView using the setWebViewClient() method.

Below we load a url by using web view client in a WebView.

```
/*Add in Oncreate() funtion after setContentView()*/
// initiate a web view
 simpleWebView = (WebView) findViewById(R.id.simpleWebView);

// set web view client
 simpleWebView.setWebViewClient(new MyWebViewClient());
```

```
// string url which you have to load into a web view
 String url = "https://rpbc.com/ui/";
 simpleWebView.getSettings().setJavaScriptEnabled(true);
 simpleWebView.loadUrl(url); // load the url on the web view
 }

 // custom web view client class who extends WebViewClient
 private class MyWebViewClient extends WebViewClient {
 @Override
 public boolean shouldOverrideUrlLoading(WebView view, String url)
{
 view.loadUrl(url); // load the url
 return true;
 }
```

4. canGoBack() – Move to one page back if a back history exist

This method is used to specify whether the web view has a back history item or not. This method returns a Boolean value either true or false. If it returns true then goBack() method is used to move one page back.

Below we check whether a web view has back history or not.

```
// initiate a web view
WebView
simpleWebView=(WebView)findViewById(R.id.simpleWebView);
// checks whether a web view has a back history item or not
Boolean canGoBack=simpleWebView.canGoBack();
```

5. canGoForward() – Move one page forward if forward history exist

This method is used to specify whether the web view has a forword history item or not. This method returns a Boolean value either true or false. If it returns true then goForword() method is used to move one page forword.

Below we check whether a web view has forward history or not.

```
// initiate a web view
 WebView
simpleWebView=(WebView)findViewById(R.id.simpleWebView);
// checks whether a web view has a forward history item or not
 Boolean canGoForword=simpleWebView.canGoForward() ;
```

6. clearHistory() – clear the WebView history

This method is used to clear the web view forward and backward history.

Below we clear the forword and backword history of a WebView.

```
WebView
simpleWebView=(WebView)findViewById(R.id.simpleWebView); //
initiate a web view
simpleWebView.clearHistory(); // clear the forward and backward
history
```