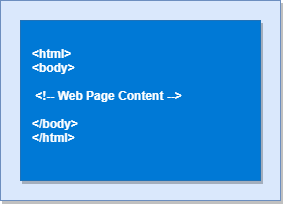
**9. Android WebView with Examples**

In android, **WebView** is an extension of **View** class and it is used to show the static HTML web pages content or remote web pages content with URL in android applications as a part of our activity layout.

Generally, in android the **WebView** will act as an embedded browser to include the web pages content in our activity layout and it won’t contain any features of normal browser, such as address bar, navigation controls, etc.

Following is the pictorial representation of WebView in android applications.



Generally, in android **WebView** is useful to include and show the content of other web pages or application content in our required pages, such as an end-user agreements, etc.

**Android WebView Example**

Following is the example of showing a static HTML content in WebView in android applications.

Create a new android application using android studio and give names as **WebView**. In case if you are not aware of creating an app in android studio check this article [Android Hello World App](https://www.tutlane.com/tutorial/android/android-hello-world-app-example).

Now open an **activity\_main.xml** file from **/res/layout** path and write the code like as shown below

**activity\_main.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" />

Once we are done with adding a **WebView** to the layout, now we will show the static HTML content in **WebView**, for that open main activity file **MainActivity.java** from **\java\com.tutlane.webview**

path and write the code like as shown below.

**MainActivity.java**

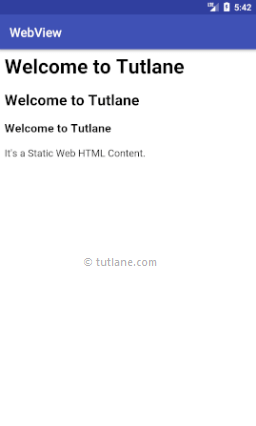
package com.tutlane.webview;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.webkit.WebView;  
  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
        WebView wv = (WebView) findViewById(R.id.webview);  
        String customHtml = "<html><body><h1>Welcome to Tutlane</h1>" +  
                "<h2>Welcome to Tutlane</h2><h3>Welcome to Tutlane</h3>" +  
                "<p>It's a Static Web HTML Content.</p>" +  
                "</body></html>";  
        wv.loadData(customHtml, "text/html", "UTF-8");  
    }  
}

If we observe above code, we are calling our layout using **setContentView** method in the form of **R.layout.layout\_file\_name**. Here our xml file name is **activity\_main.xml** so we used file name **activity\_main** and trying to show the static HTML content in **WebView**.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), onCreate() callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output**

When we run above example using android virtual device (AVD) we will get a result like as shown below.



This is how we can show the static HTML content using WebView in android applications based on our requirements.

**Android Show Web URL Content in WebView Example**

Generally, in android **WebView** will act as an embedded browser to show the static or remote web page content in our android applications.

Now we will see how to load remote URL content in WebView with example in android application.

By using WebView **LoadURL** property we can load remote URL content in our android applications. To show the remote URL content in webview modify **MainActivity.java** file code like as shown below.

**ManiActivity.java**

package com.tutlane.webview;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.webkit.WebView;  
  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
        WebView webView = (WebView) findViewById(R.id.webview);  
        webView.getSettings().setJavaScriptEnabled(true);  
        webView.loadUrl("http://www.juniv.edu");  
    }  
}

If we observe above example, we are trying to load the remote URL (**http://sarker.com**) content in our android application using **WebView** and we set a property **setJavaScriptEnabled()** to enable JavaScript because by default the JavaScript is disabled in WebView.

Generally, the web page which we are loading in **WebView** might use JavaScript. In case if we won’t enable the JavaScript, the functionality which related to JavaScript in web page won’t work that’s the reason we are enabling the JavaScript using setJavaScriptEnabled()

To load the remote web URL content, our application must have an access to the internet. We need to set the internet access permissions like as shown below.

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

……  
</manifest>

Now open our application **AndroidManifest.xml** file in **/manifests** directory and write the code like as shown below

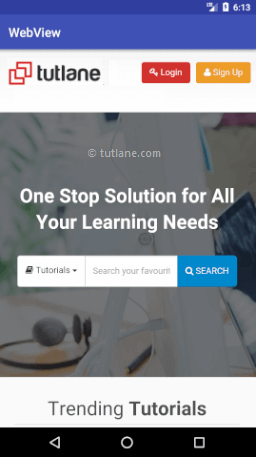
**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.tutlane.webview">  
    <application  
        android:allowBackup="true"  
        android:icon="@mipmap/ic\_launcher"  
        android:label="@string/app\_name"  
        android:roundIcon="@mipmap/ic\_launcher\_round"  
        android:supportsRtl="true"  
        android:theme="@style/AppTheme">  
        <activity android:name=".MainActivity">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
    </application>  
    <uses-permission android:name="android.permission.INTERNET" />  
</manifest>

Once we are done with required settings, now we will run and see the result of our application.

**Output**

When we run above example using android virtual device (AVD) we will get a result like as shown below.



This is how we can show the remote URL web pages content using **WebView** in android applications based on our requirements.