Programowanie urządzeń mobilnych

Wykorzystanie WebView

Tłumaczenie i adaptacja materiałów: dr Tomasz Xięski.
Na podstawie prezentacji udostępnionych przez Victor Matos, Cleveland State University.

Metody tworzenia aplikacji

Istnieją trzy sposoby realizacji aplikacji na system Android:

1. **Aplikacja natywna**
   Tworzona jest przy wykorzystaniu natywnego SDK i przechowywana jest na urządzeniu. Wykorzystuje wszelkie komponenty platformy Android jak widżety, usługi, aktywności, fragmenty, dostawcy treści, powiadomienia, itp.)

2. **Aplikacja internetowa**
   Zwykle realizowana jest jako aplikacja zdalna i umożliwia pobranie zasobów z serwera z wykorzystaniem zainstalowanej przeglądarki internetowej.

3. **Aplikacja hybrydowa**
   Realizowana jest jako aplikacja internetowa, jednakże prócz znaczników html i wykorzystania zdalnych zasobów możliwa jest interakcja z lokalnymi zasobami i możliwościami urządzenia (np. dostęp do czujników)

WebView

Metody tworzenia aplikacji

Każde podejście ma swoje **wady i zalety**. Przykładowo:

- Opcja 1 pozwala na tworzenie bogatszego interfejsu i wykorzystuje w pełni dostępne zasoby sprzętowe, ale ograniczona jest do platformy Android. Często wykorzystanie innej platformy wymaga przepisania znacznej części bądź całości aplikacji.

- Opcja 2 pozwala uzyskać największy stopień przenośności. Każde urządzenie z dostępem do przeglądarki internetowej może otworzyć taką aplikację, jednakże brak jest bezpośredniego dostępu do zasobów sprzętowych urządzenia mobilnego.

- Opcja 3 stanowi kompromis między pozostałymi podejściami. Wykorzystanie standardu HTML (włącznie z elementami wersji piątej) oraz dostęp z poziomu języka skryptowego do fizycznych zasobów urządzenia zapewnia duże możliwości i zachowanie przyzwoitych możliwości przenośności.

WebView

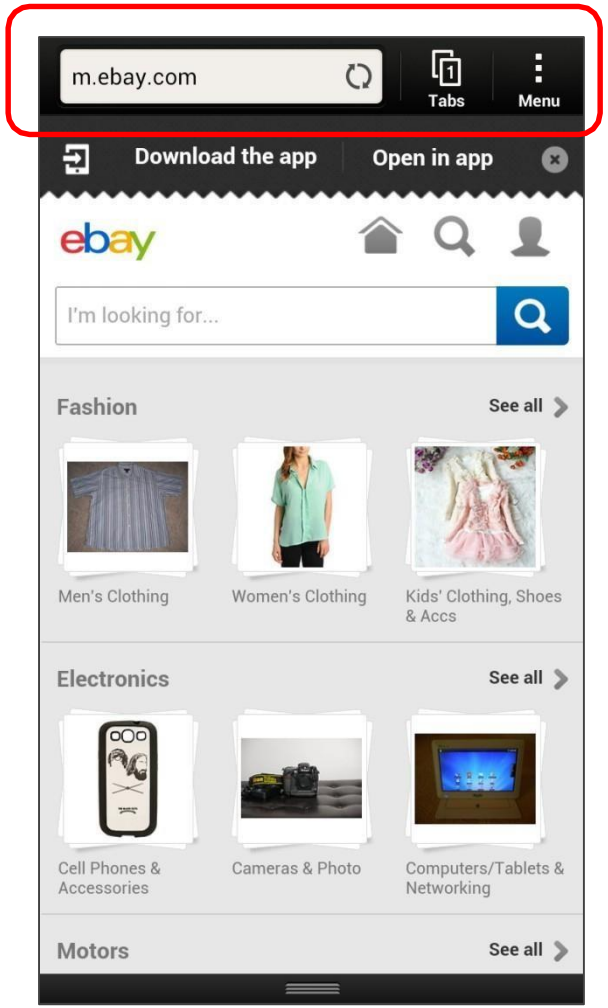
WebView - własności

1. Komponent WebView wykorzystuje otwarto-źródłowy silnik **WebKit** (znany z przeglądarek typu Chrome czy Opera).

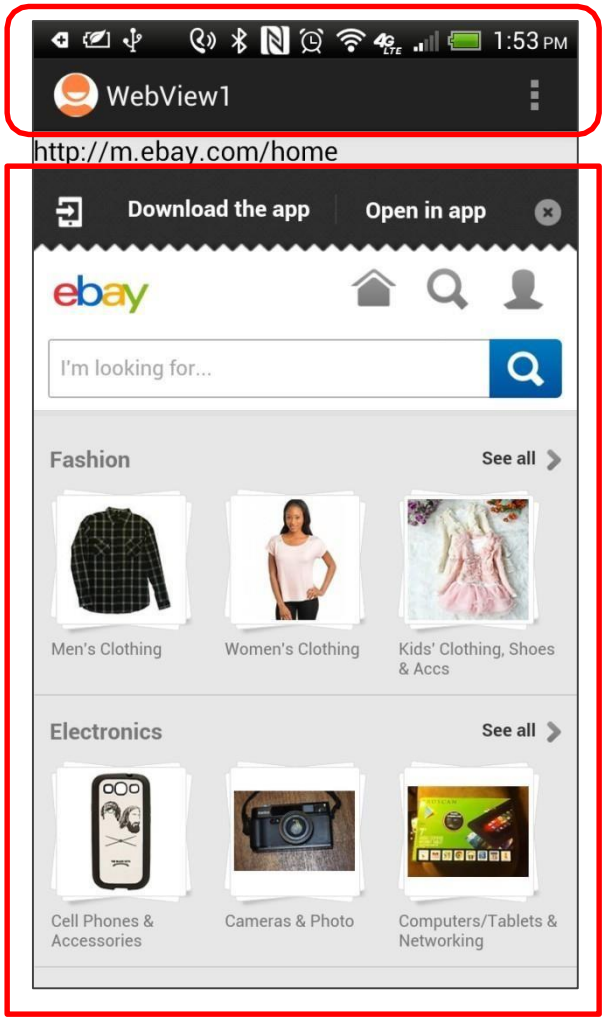
2. Komponent WebView nie jest przeglądarką. Pomimo, że umożliwia wyświetlenie strony internetowej, ale nie posiada standardowych przycisków jak Wstecz, Dalej, Odśwież, itp.

3. Klasa WebView zawiera metody do:
   - Ładowania/Przeładowania strony, nawigacji wprzód lub wstecz poprzez historię przeglądarki**,**
   - Przybliżanie i oddalanie,
   - Wykorzystanie styli CSS oraz skryptów JavaScript by zapewnić podobne GUI na różnych urządzeniach,
   - Wymiana danych z urządzeniem przez interfejs JavaScript,
   - Wyszukiwania tekstu, pobierania zdjęć,

WebView



Przeglądarka prezentująca stronę internetową [(http://m.ebay.com)](http://m.ebay.com)

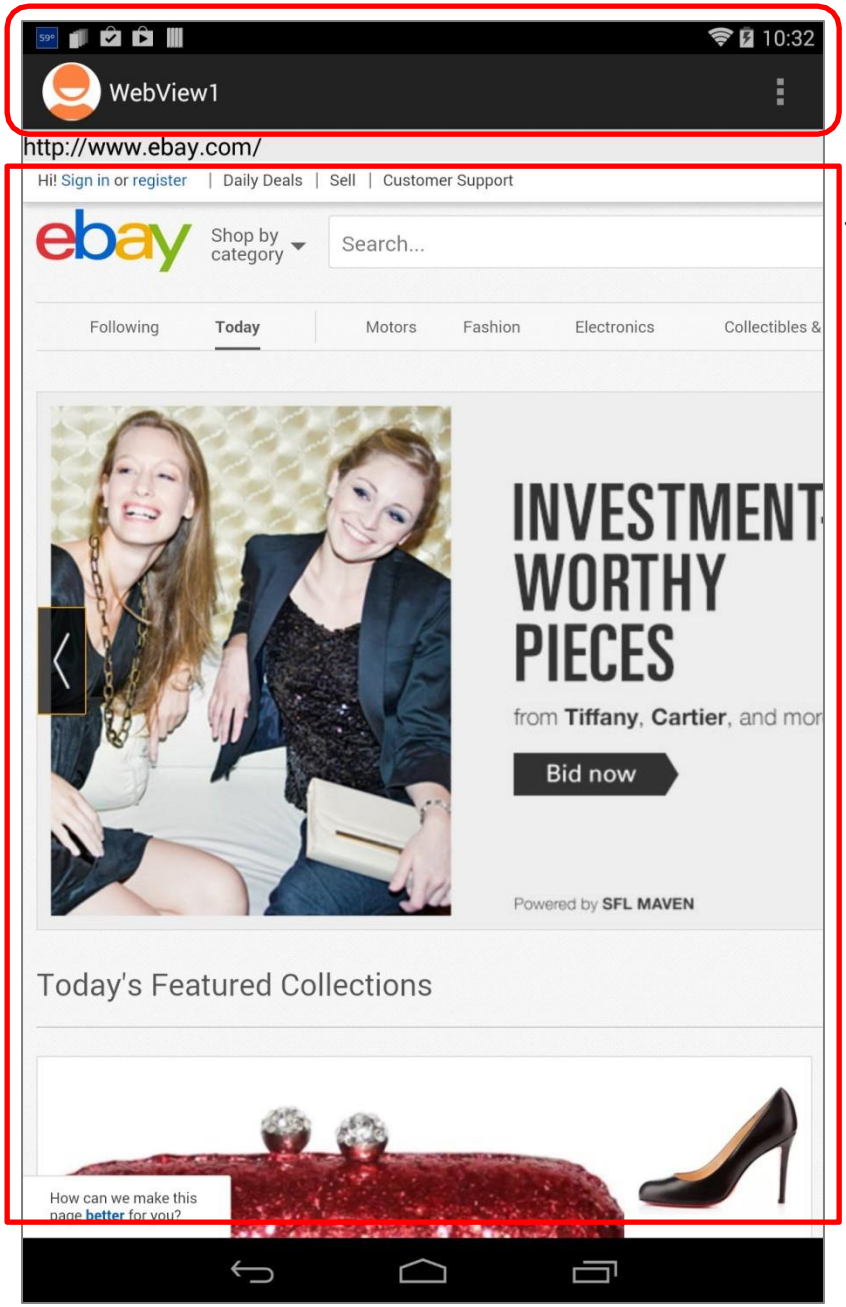


←WebView

Aplikacja Androida wykorzystująca **WebView** by wyświetlić tą samą stronę.

WebView

Aplikacja działająca pod kontrolą tabletu. Oryginalna wersja strony www.ebay.com zostaje zaprezentowana (bez przekierowania jak w przypadku telefonu komórkowego).



←**WebView**

WebView

Uprawnienia

Uwaga! By aktywność miała dostęp do internetu należy dodać uprawnienie **INTERNET** w pliku **Manifestu**. Jeżeli w aplikacji wykorzystywane są dodatkowe biblioteki jak np. **Google Maps**, należy dodać klauzulę `<uses-library…>`.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="csu.matos.webview_demo1"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
          android:minSdkVersion="8"
          android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
          . . .
          <activity
             . . .
          </activity>
          <uses-library android:name="com.google.android.maps" />
    </application>

</manifest>
```

WebView

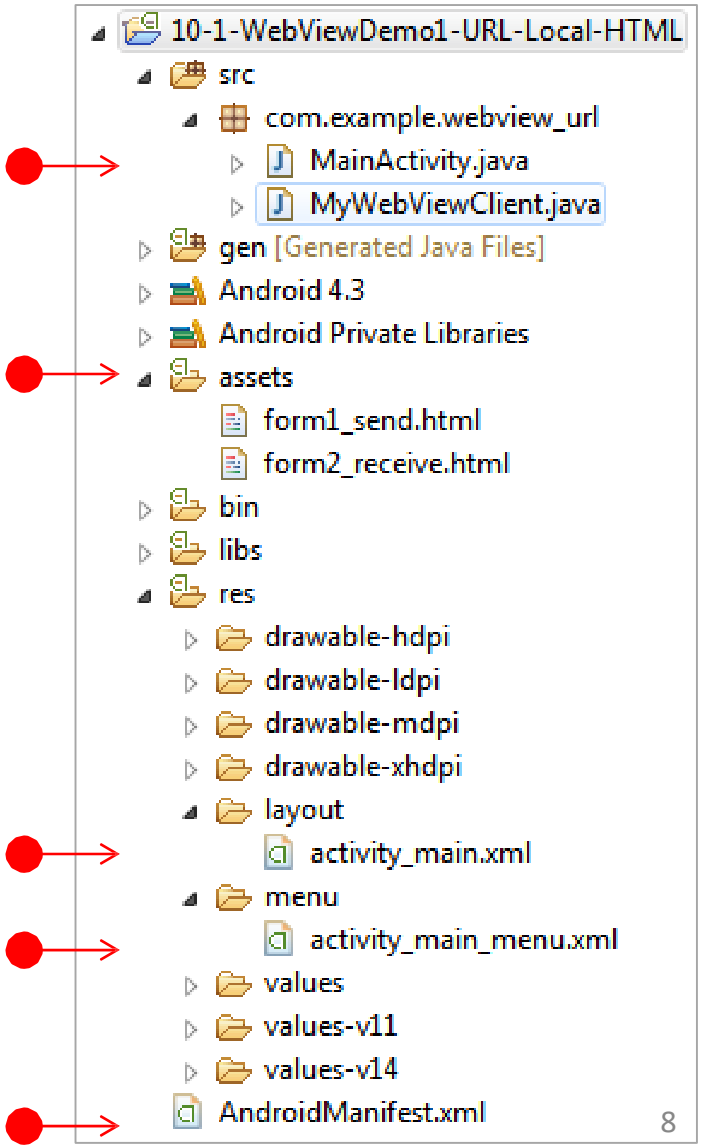
Przykład 1. Wykorzystanie WebView

W tym przykładzie zostaną zaprezentowane 3 sposoby ładowania zasobów do komponentu WebView

1. W pierwszym przypadku, źródłem danych jest strona internetowa znajdująca się na zdalnym serwerze.

2. W drugim przypadku, komponent WebView służy do wyświetlenia danych pochodzących z lokalnie zapisanych stron internetowych.

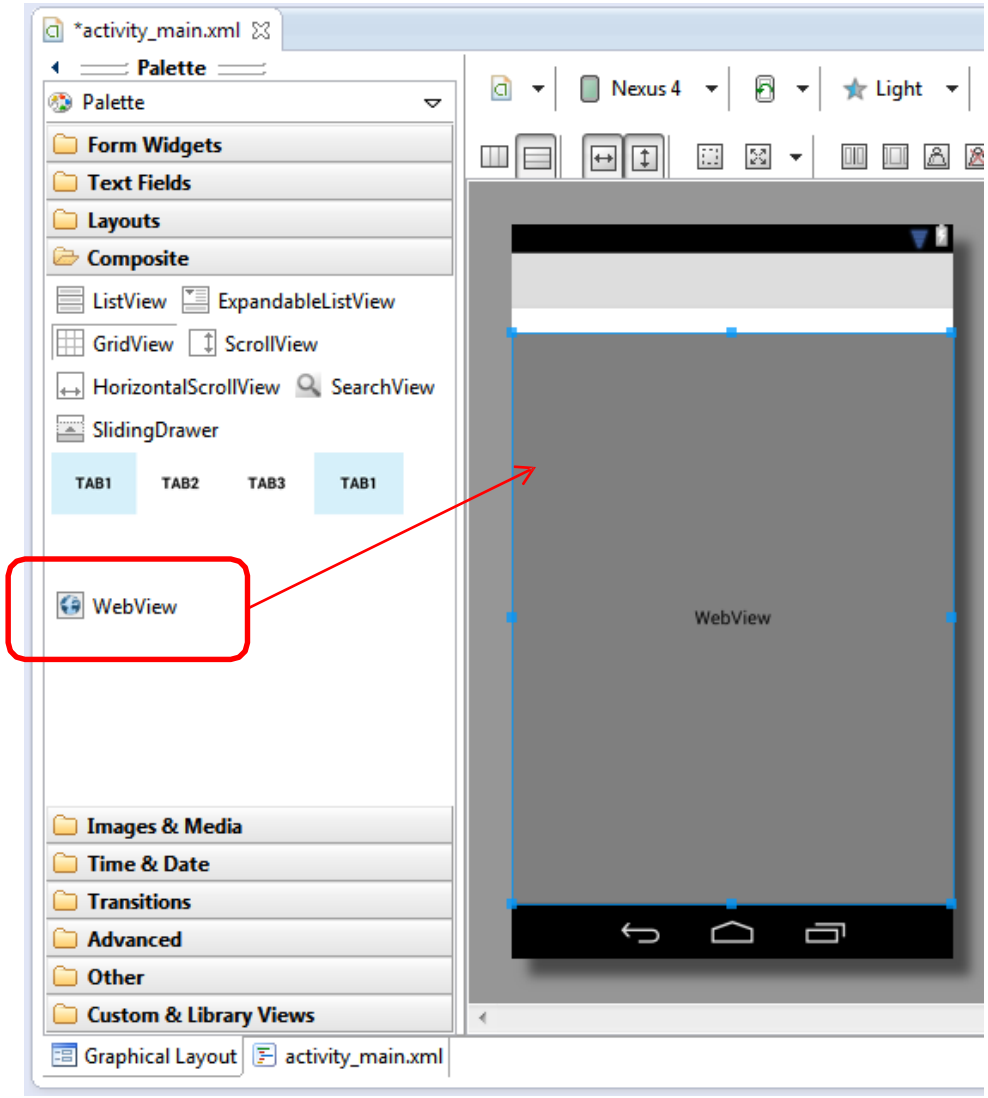
3. Trzeci przykład prezentuje sposób na dynamiczne doładowywanie danych z serwera.



WebView

Przykład 1. Wykorzystanie WebView

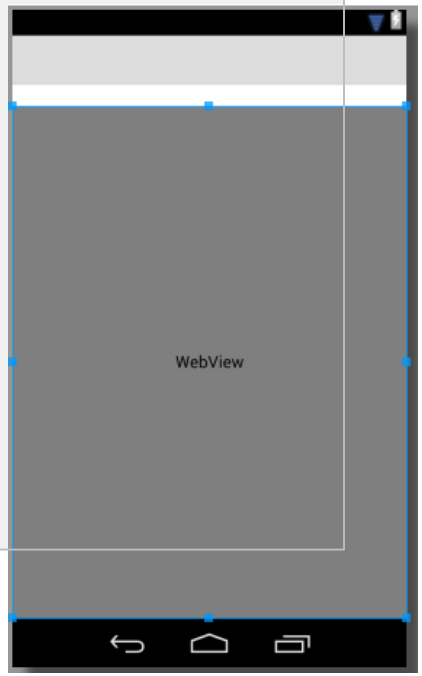
Najprościej rozpocząć od wyboru komponentu **WebView** z dostępnej palety komponentów.



WebView

Przykład 1. Układ – activity_main.xml

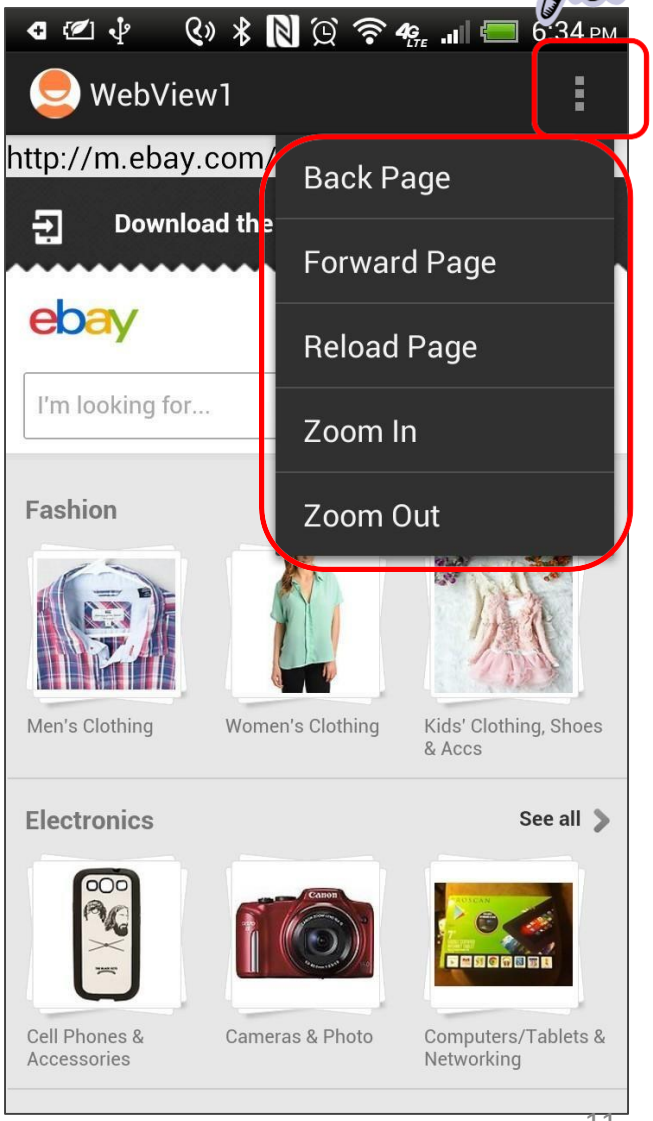
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >

    <TextView android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium" />

        <WebView   android:id="@+id/webview1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
    />
</LinearLayout>
```



WebView

Przykład 1. Menu główne – activity_main_menu.xml

```
<menu
xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/back_page"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="Back Page"/>
    <item
        android:id="@+id/forward_page"
        android:orderInCategory="110"
        android:showAsAction="never"
        android:title="Forward Page"/>
     <item
        android:id="@+id/reload_page"
        android:orderInCategory="120"
        android:showAsAction="never"
        android:title="Reload Page"/>
      <item android:id="@+id/zoom_in"
        android:orderInCategory="130"
        android:showAsAction="never"
        android:title="Zoom In"/>
       <item
        android:id="@+id/zoom_out"
        android:orderInCategory="140"
        android:showAsAction="never"
        android:title="Zoom Out"/>
</menu>
```



WebView

Przykład 1. MainActivity.java

```
public class MainActivity extends Activity {

   TextView txtMsg;
   WebView webview;


   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      txtMsg = (TextView) findViewById(R.id.txtMsg);

      // Try demo1, demo2, or demo3 (please, uncomment one at the time!!!)
       demo1TrySpecificUrl();

      // demo2TryLocallyStoredHtmlPage();
      // demo3TryImmediateHtmlPage();
      // demo4TryRichGoogleMap();

   }// onCreate

   // ----------------------------------------
   // Demo1, Demo2 and Demo3 code goes here...
   // ----------------------------------------

}//MainActivity
```

WebView

Przykład 1. MainActivity.java - Zdalna strona www

```
@SuppressLint("SetJavaScriptEnabled")
private void demo1TrySpecificUrl() {

    webview = (WebView) findViewById(R.id.webview1);
    webview.getSettings().setJavaScriptEnabled(true);

    webview.setWebViewClient(new MyWebViewClient(txtMsg, "ebay.com"));

    webview.loadUrl("http://www.ebay.com");

}
```

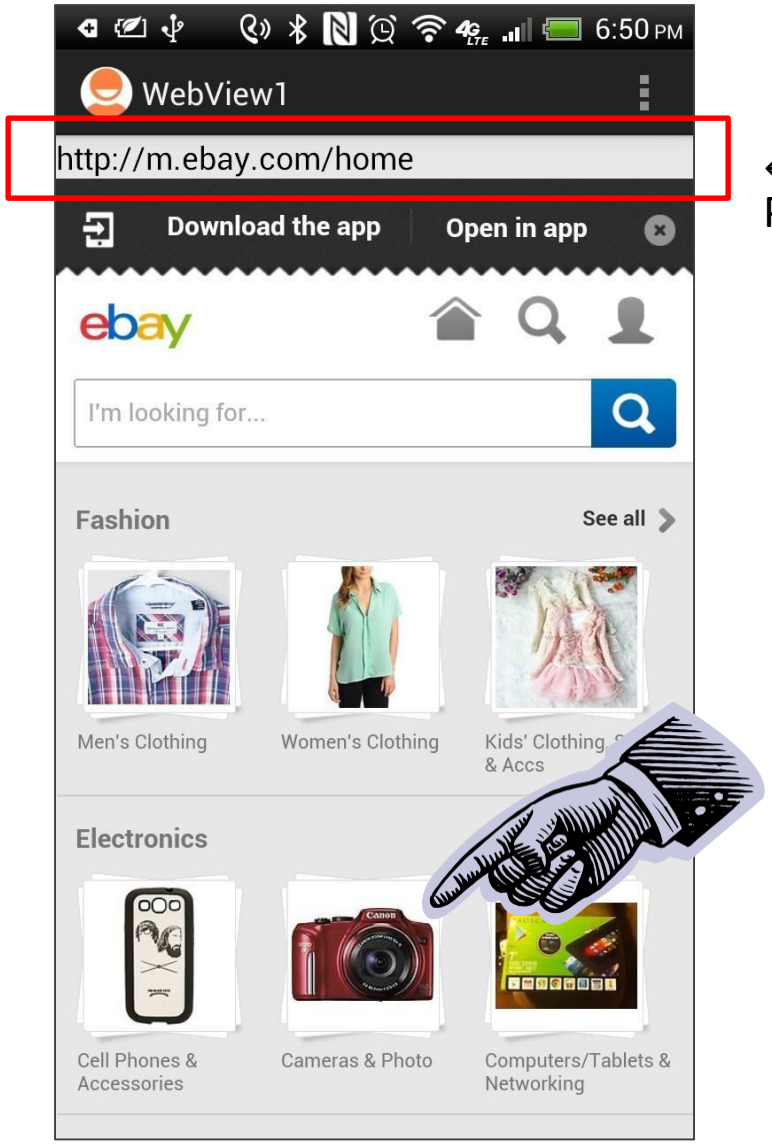
Bezpośrednie podłączenie do ebay.com

Komentarz

1. Zezwól stronie na wykonywanie skryptów JavaScript (jeśli takowe są wymagane)
2. Klasa **MyWebViewClient** odpowiada za kontrolę komunikacji sieciowej i aktualizację GUI wskazując adres URL. Tylko strony z domeny ebay.com mogą być wyświetlane w WebView – pozostałe mogą być pokazane tylko korzystając z wbudowanej przeglądarki internetowej.
3. Zadany **URL** rozpoczyna proces nawigacji. Jednakże ostateczny URL może być inny niż wskazany poprzez wymuszone przekierowanie przez zdalny serwer. Przykładowo wejście na www.ebay.com skutkuje przekierowaniem na stronę m.ebay.com

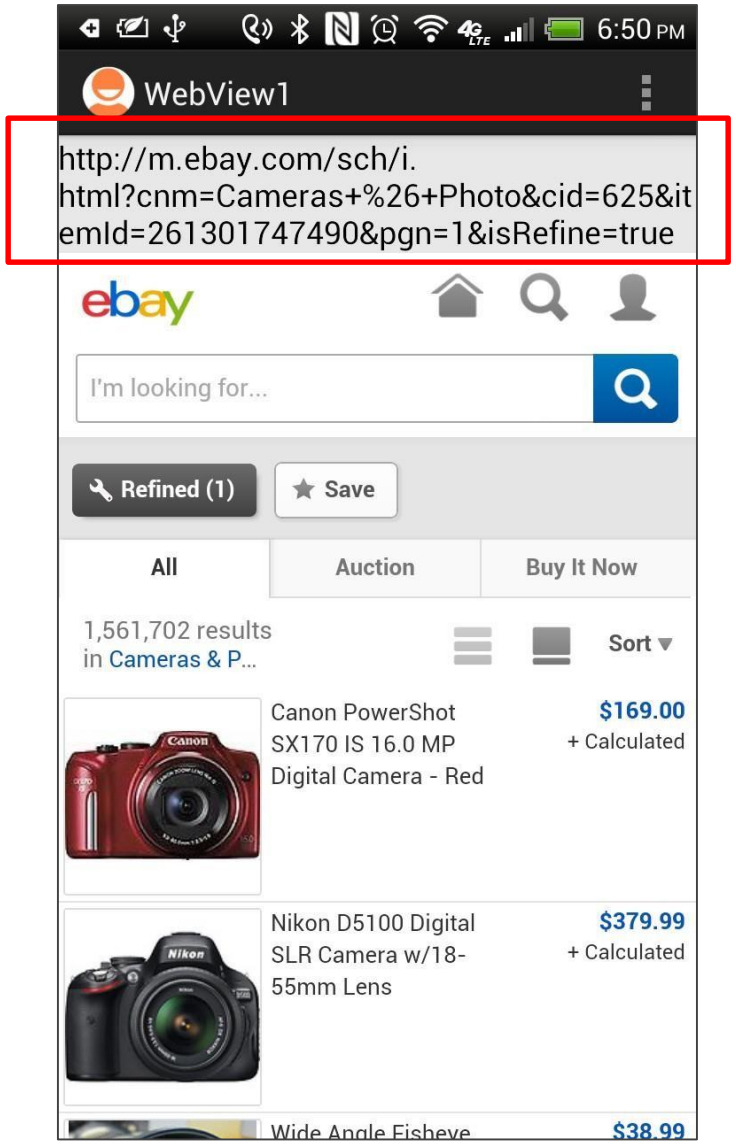
WebView

Przykład 1. MainActivity.java



← Przekierowanie

→ Dalsza nawigacja wymuszona przez użytkownika



WebView

Przykład 1. MainActivity.java - Lokalna strona html

```
@SuppressLint("SetJavaScriptEnabled")
private void demo2TryLocallyStoredHtmlPage() {
    webview = (WebView) findViewById(R.id.webview1);
    webview.getSettings().setJavaScriptEnabled(true);

    webview.setWebViewClient(new WebViewClient());

    webview.loadUrl("file:///android_asset/form1_send.html");

}
```

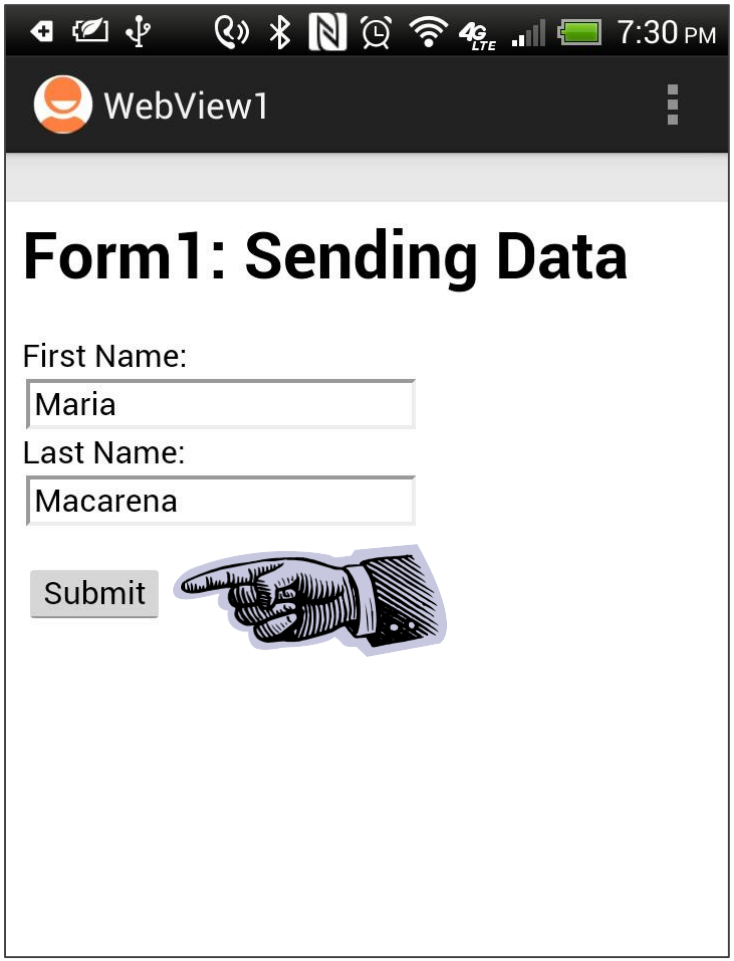
Komentarz

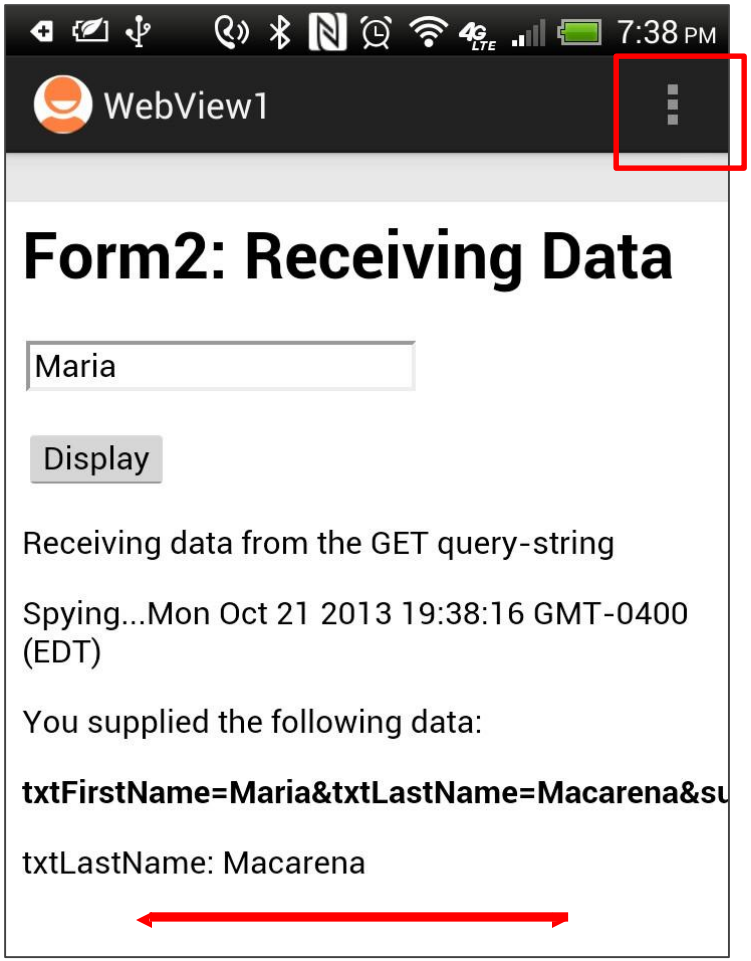
1. Wbudowany obiekt **WebViewClient** używany jest do kontroli nawigacji – dozwolone jest wyświetlanie dowolnych stron internetowych w ramach komponentu WebView.

2. Strona internetowa przechowywana jest w zasobach lokalnych aplikacji (ale może być również przechowywana np. na karcie pamięci). Zwykle strony przechowywane są w katalogu **/assets/** (na tym samym poziomie co /res/ i /java/).

WebView

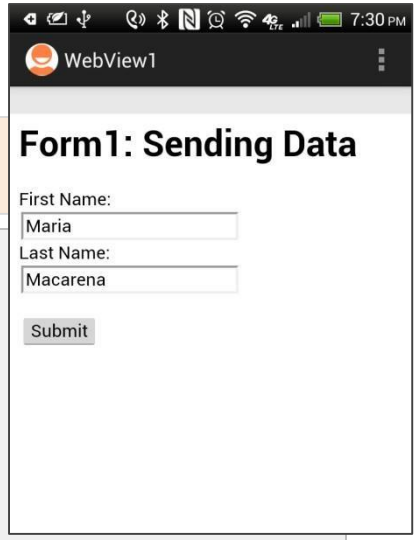
Przykład 1. MainActivity.java - Lokalna strona html





WebView

Przykład 1. form1_send.html - Lokalna strona html




```
<html>
<head>
</head>
<body>

<FORM action="form2_receive.html" id=form1 method=GET name=form1>
        <h1>Form1:  Sending Data</h1>
        First Name:
        <br>
        <INPUT id="txtFirstName" name="txtFirstName" value="Maria">
        <br>
        Last Name:
        <br>
        <INPUT id="txtLastName" name="txtLastName" value="Macarena">
        <p>
        <INPUT type="submit" value="Submit" id=submit1 name=submit1>
</FORM>
</body>
</html>
```

Komentarz

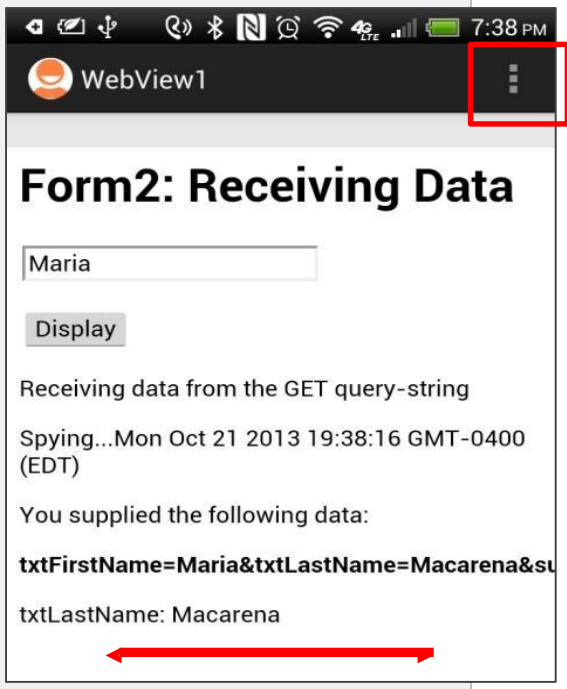
Po wciśnięciu **Submit** wywoływana jest strona *form2_receive.html*. Zawartość formularza zostanie przekazana z wykorzystaniem metody **HTTP GET:**

`txtFirstName=Maria&txtLastName=Macarena&submit1=Submit`

WebView

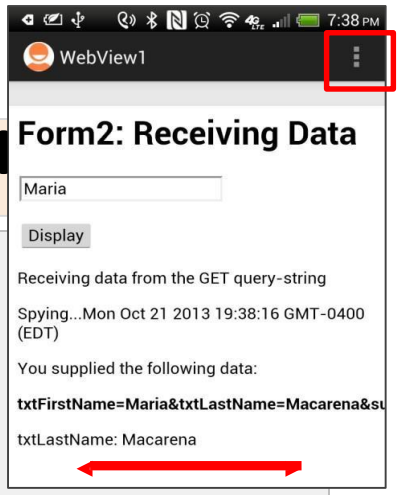
Przykład 1. form2_receive.html - Lokalna strona html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
   <title>Android & Plain HTML Demo</title>
   <script language="javascript">

  function extracting(variable) {
    var query = window.location.search.substring(1);
    alert(query);
    var personName = getQueryVariable(variable)
    document.getElementById("myMsg").value = personName;
  }

  function getQueryVariable(variable) {
      var query = window.location.search.substring(1);
      var vars = query.split('&');
      for (var i=0;i<vars.length;i++) {
          var pair = vars[i].split("=");
          if ((pair[0] == variable) || (variable == 0)) {
            return pair[1];
          }
       }
       alert('Query Variable ' + variable + ' not found');
  }
</script>
```



WebView

Przykład 1.form2_receive.html – Lokalna strona html




```
</head>
 <body>
 <h1>Form2:  Receiving Data</h1>

 <p><input TYPE="text"  id="myMsg" ></p>

 <p><input TYPE="button" value="Display" onClick="extracting('txtFirstName');"></p>


   <p>Receiving data from the GET query-string
   <script>
      document.write("<p>Spying..." + new Date() );
      document.write("<p> You supplied the following data:");
      var querystring = window.location.search.substring(1);
      document.write("<p><b>" + querystring + "</b>");

      document.write("<p>txtLastName: "  + getQueryVariable('txtLastName') );

   </script>


 </body>
</html>
```

WebView

Przykład 1. form2_receive.html – Lokalna strona html

Komentarz

Właściwość **`window.location.search`** zwraca dane przekazane przez formularz (wraz ze znakiem ?):

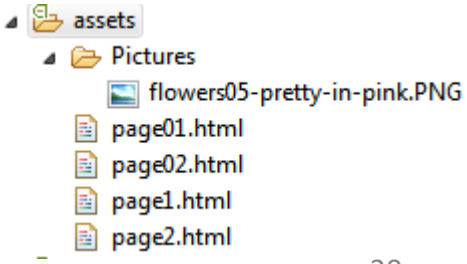
```
window.location.search =
?txtFirstName=Maria&txtLastName=Macarena&Submit1=Submit
```

Zatem

```
window.location.search.substring(0) to '?' natomiast

window.location.search.substring(1) to
txtFirstName=Maria&txtLastName=Macarena&submit1=Submit
```

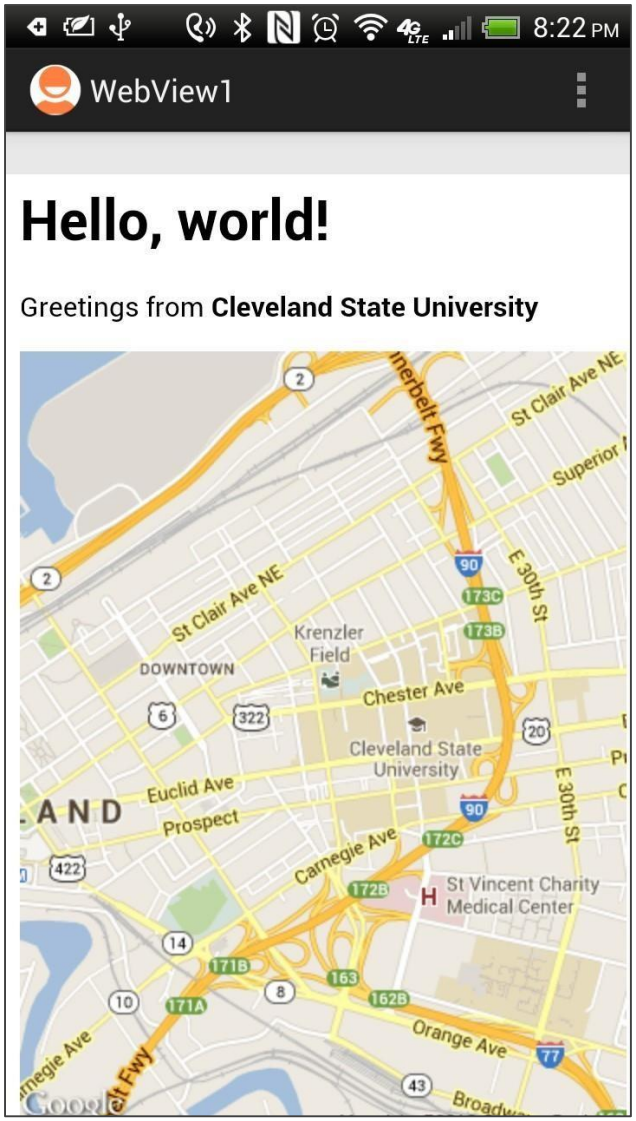
Można dodać *obrazy* do formularza. Dodaj podkatalog **assets/Pictures**. Odwołanie wygląda następująco

```
<img src="Pictures/nazwaObrazka.png" />
```



WebView

Przykład 1. MainActivity.java - Ładowanie dynamiczne



Wykorzystanie Map Google do lokalizowania miejsc na podstawie współrzędnych**:**

http://maps.googleapis.com/maps/api/staticmap?center=41.5020952,-81.6789717&zoom=14&size=350x450&sensor=false

Umożliwia to wyświetlenie statycznej mapy wycentrowanej na podstawie podanych współrzędnych (długości i szerokości geograficznej)

Przykład pokaże jak skopiować to zachowanie korzystając z aplikacji na system Android.

Zdjęcie wygenerowano korzystając z **Google Maps API**. Więcej informacji pod adresem: https://developers.google.com/maps/

WebView

Przykład 1. MainActivity - Ładowanie dynamiczne

Wykorzystanie Map Google do lokalizowania miejsc na podstawie nazwy**:**

http://maps.googleapis.com/maps/api/staticmap
?center=Cleveland+State+University,Ohio
&zoom=15&size=480x700
&maptype=roadmap
&markers=color:green|label:C|41.5020952,-81.6785000
&sensor=false

Umożliwia to wyświetlenie statycznej mapy drogowej wycentrowanej na podstawie podanej nazwy danego miejsca.

Zdjęcie wygenerowano korzystając z **Google Maps API**. Więcej informacji pod adresem: https://developers.google.com/maps/

WebView

Przykład 1. MainActivity.java - Ładowanie dynamiczne

```
@SuppressLint("SetJavaScriptEnabled")
private void demo3TryImmediateHtmlPage() {

    webview = (WebView) findViewById(R.id.webview1);
    webview.getSettings().setJavaScriptEnabled(true);
    webview.setWebViewClient(new WebViewClient());

    String aGoogleMapImage = "<img src=\"http://maps.googleapis.com/maps/api/"
          + "staticmap?center=41.5020952,-81.6789717&"
          + "zoom=14&size=350x450&sensor=false\"> ";

    String myLocalHtmlPage =
            "<html> "
          +    "<body>"
          +     "<h1>Hello, world! </h1>"
          +     "<p> Greetings from <b>Cleveland State University</b>"
          +     "<p>" + aGoogleMapImage
          +    "</body> "
          + "</html>";

    webview.loadData(myLocalHtmlPage, "text/html", "UTF-8");
}
```

WebView

Przykład 1. Ładowanie dynamiczne

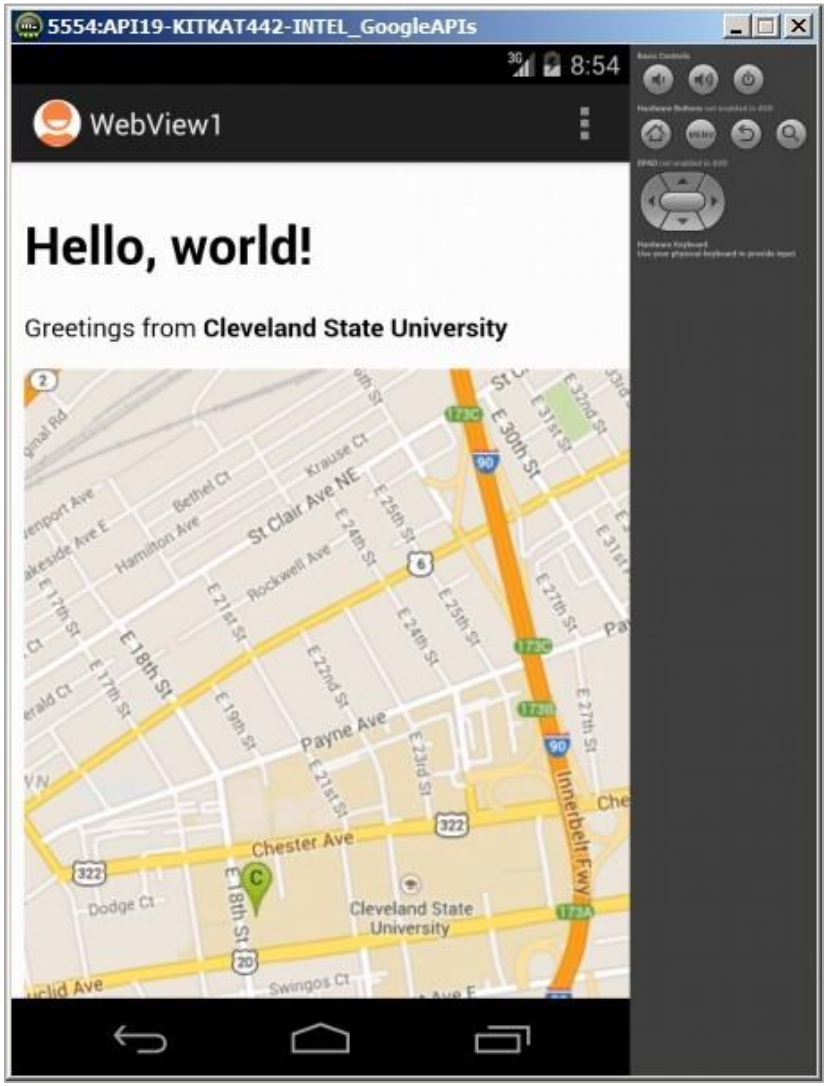
Uwaga!

Sprawdź, czy twoja aplikacja:

1. Ma odpowiednie **uprawnienia** zdefiniowane w pliku **Manifestu** (Internet oraz com.google.android.maps są wymagane)

2. Posiada wkompilowaną bibliotekę Google.API.

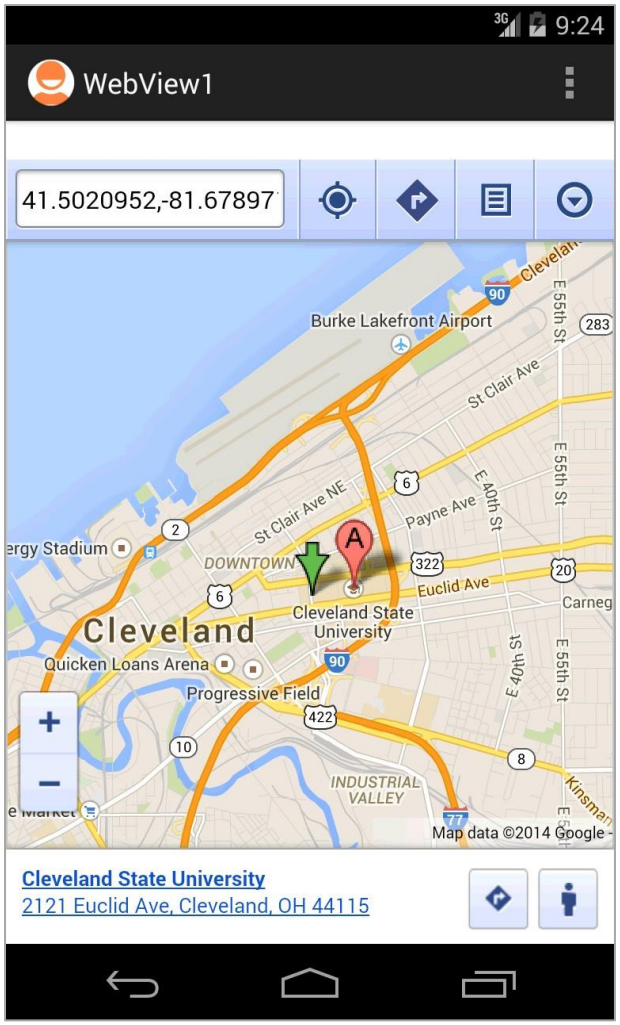
3. Emulator został prawidłowo skonfigurowany (z *Google APIs (x86 System Image))*



WebView

Przykład 1. MainActivity.java - Mapy raz jeszcze

Tym razem zostanie wykorzystana biblioteka **Google Map API V3**




```
@SuppressLint("SetJavaScriptEnabled")
private void demo4TryRichGoogleMap() {
  webview = (WebView) findViewById(R.id.webView1);

  webview.getSettings().setJavaScriptEnabled(true);
  //show only WebViews
  webview.setWebViewClient(new WebViewClient());

  String mapUrl = "https://maps.google.com/maps"
          + "?q=41.5020952,-81.6789717&t=m&z=13";

  webview.loadUrl( mapUrl );

}
```

Tworzona jest „bogatsza” wersja mapy, posiadająca wbudowane kontrolki by przełączyć się na tryb street-view, zmienić powiększenie itp. Więcej informacji pod: https://developers.google.com/maps/documentation/javascript/basics#Mobile

Przykład 1. Mapy raz jeszcze

Uwaga:
Występuje drobna różnica względem poprzedniego przykładu

http://maps.**google**.com/ → Wykorzystuje **Google Maps V2**

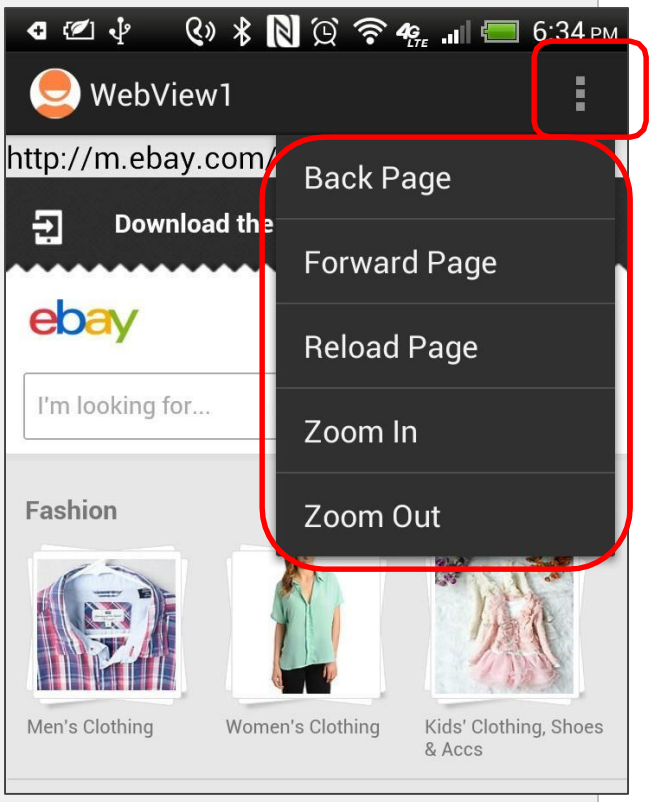
http://maps.**googleapis**.com → Wykorzystuje **Google Maps V3**

Biblioteka V3 jest szybsza od wersji V2 oraz została stworzona z myślą o urządzeniach mobilnych. Dodatkowo biblioteka w starszej wersji nie będzie dalej rozwijana
(zobacz https://www.youtube.com/v/zI8at1EmJjA)

WebView

Przykład 1. MainActivity.java - Menu

```
    // browser navigation implemented through the option-menu
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main_menu, menu);
        return true;
    }// onCreateOptionsMenu

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        String option = item.getTitle().toString();
        if (option.equals("Forward Page"))
            webview.goForward();
        if (option.equals("Back Page"))
            webview.goBack();
        if (option.equals("Reload Page"))
            webview.reload();
        if (option.equals("Zoom In"))
            webview.zoomIn();
        if (option.equals("Zoom Out"))
            webview.zoomOut();
        return true;
    }// onOptionsItemSelected
}// class
```



WebView

Przykład 1. MyWebViewClient.java – Kontrola nawigacji

```
class MyWebViewClient extends WebViewClient {
   // this object keeps a history-log of all visited links
   // and validates url links against a "home-base" server address
   Context context;
   TextView txtMsg;
   String hostServerSuffix; //this is the home-base

   public MyWebViewClient(TextView txtMsg, String hostServerSuffix) {
      this.txtMsg = txtMsg;
      this.hostServerSuffix = hostServerSuffix;
   }

   @Override
   public boolean shouldOverrideUrlLoading(WebView view, String url) {
      context = view.getContext();

      String host = Uri.parse(url).getHost();
      // if 'host' portion extracted above keeps us inside valid
      // base server (hostServerSuffix) then use a WebView to continue
      // navigation, otherwise override navigation by showing requested
      // page inside a full blown-up browser (security issues here...)

      String text = "URL:" + url.toString()
               + "\n\nHOST:" + host
               + "\n\nHOME should be:" + hostServerSuffix;
```

WebView

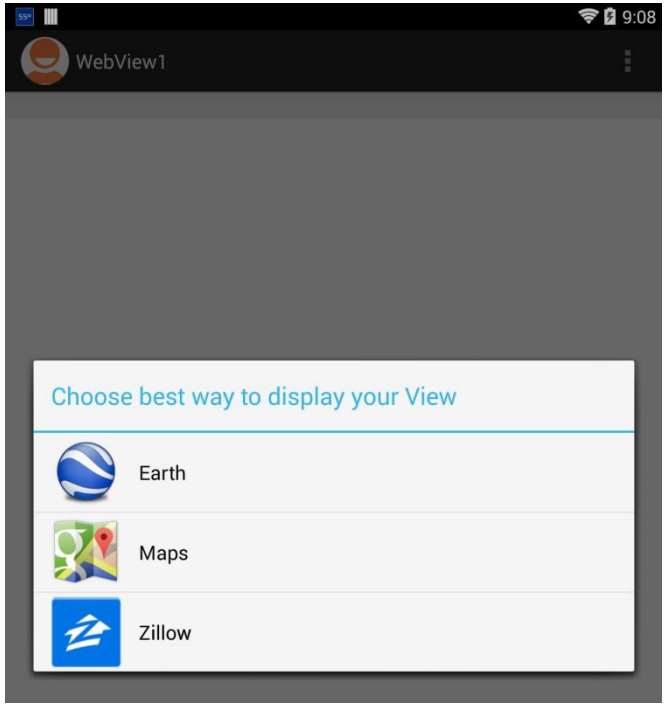
Przykład 1. MyWebViewClient.java – Kontrola nawigacji

```
      Toast.makeText(context, text, Toast.LENGTH_LONG).show();

      if (url.contains(hostServerSuffix)) {
         // do not override navigation (using big web-browser) as long as
         // url points to some page in my defined home-server
         return false;

      } else {
         // The supplied url is not for a page on my 'valid' site,
         // in that case launch another Activity that handles URLs
         Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
         context.startActivity(intent);
         return true;
      }
   }//shouldOverrideUrlLoading

   @Override
   public void onPageStarted(WebView view, String url, Bitmap favicon) {
      // keep user informed - update txtMsg with current URL value
      super.onPageStarted(view, url, favicon);
      txtMsg.setText(url.toString());
   }//onPageStarted

}//MyWebViewClient
```

WebView

Przykład 1. MyWebViewClient.java – Kontrola nawigacji

Co się stanie jeżeli nie zostanie wykorzystany obiekt typu WebViewClient?



Jeżeli **WebViewClient** *nie zostanie wykorzystany,* komponent **WebView** poprosi menedżera aktywności by obsłużyć URL (jeżeli dzieje się to po raz pierwszy, prezentowana jest lista kandydujących aplikacji)

Rysunek po prawej stronie prezentuje taką listę dla przykładu z mapami. Zawiera ona wszystkie zainstalowane aplikacje, dedykowane do obsługi map jak Google-Earth, Google-Maps, czy Zillow.

Zadaniem WebViewClient jest kontrola dostępu do adresów URL:

- jeżeli aktualny adres jest „akceptowany” zwraca on wartość FALSE i komponent WebView przejmuje odpowiedzialność za wyświetlenie strony,
- W przeciwnym przypadku aktywność decyduje o sposobie wyświetlenia danych – zwykle poprzez wykorzystanie przeglądarki internetowej.

WebView

Uwaga – Wykorzystanie JavaScript i animacji Flash

Aplikacja **musi** jawnie nadać uprawnienie by umożliwić wykonanie kodu **JavaScript** na odwiedzanych stronach. Domyślnie interpretacja *Javascript* jest **wyłączona** . Aktywacja wygląda następująco:

```
webview.getSettings().setJavaScriptEnabled(true);
```

By usunąć ostrzeżenie o potencjalnych, niepożądanych skutkach dla bezpieczeństwa przez wykorzystanie JS, do każdej metody wykorzystującej JavaScript można dodać:

```
@SuppressLint("SetJavaScriptEnabled")
```

WebView

Uwaga – Wykorzystanie JavaScript i animacji Flash

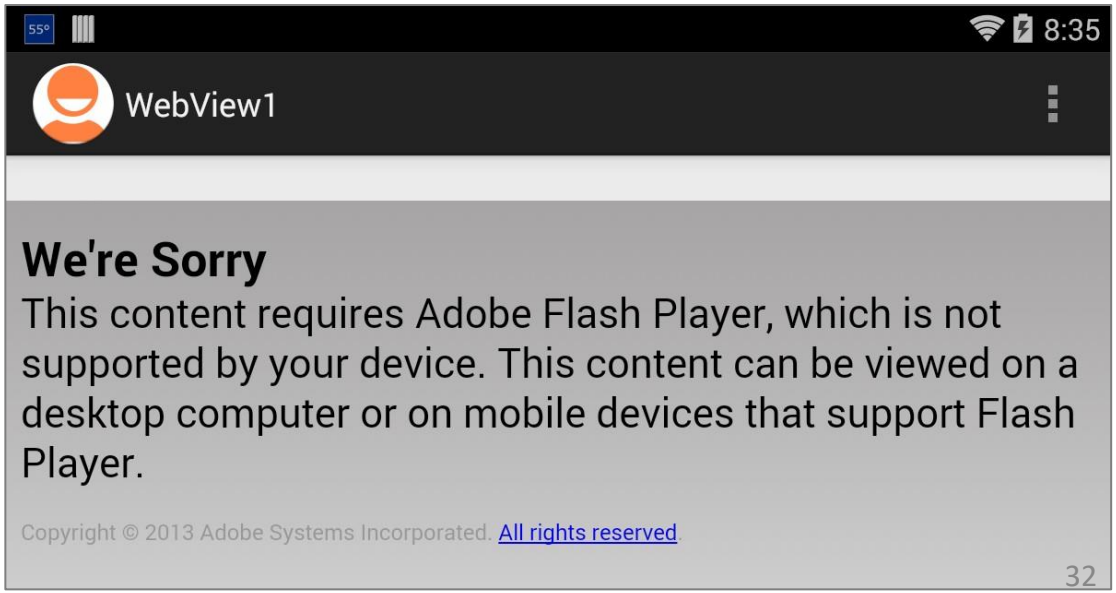
Wykorzystanie wtyczek ma status deprecated, zatem będzie niedozwolone w przyszłości. Należy unikać instrukcji typu:

```
webview.getSettings().setPluginState(PluginState.ON);
```

Przy próbie załadowania strony zawierającej animacje flash:

```
webview.loadUrl("http://get.adobe.com/flashplayer/");
```

najczęściej otrzymamy:



WebView

Często wykorzystywane metody WebView

Podane poniżej metody są przydatne podczas realizacji zadań związanych z przeglądaniem zasobów internetowych. Zazwyczaj są implementowane jako opcje menu głównego bądź inne elementy typu przycisk.

- **reload()** odśwież aktualnie przeglądaną stronę.
- **goBack()** cofnij się o jeden krok w historii (użyj canGoBack() by sprawdzić czy takie postępowanie jest możliwe).
- **goForward()** przesuń się o jedną pozycję do przodu (użyj canGoForward() by określić czy taki krok jest możliwy).
- **goBackOrForward()** przesuń się w przód lub wstecz, gdzie *ujemne/dodanie* wartości numeryczne parametru określają o ile kroków należy się przesunąć.
- Metoda **canGoBackOrForward()** umożliwia określenie czy można cofnąć się/przesunąć dalej o zadaną liczbę kroków.
- **clearCache()** wyczyść pamięć podręczną.
- **clearHistory()** wyczyść historię.
- **ZoomIn() , ZoomOut()** zmień powiększenie lub pomniejszenie.

WebView

Przykład 2. Aplikacje hybrydowe

Poniższy przykład prezentuję kombinację wykorzystania kilku technologii (html/javascript + java) do stworzenia aplikacji mobilnej.

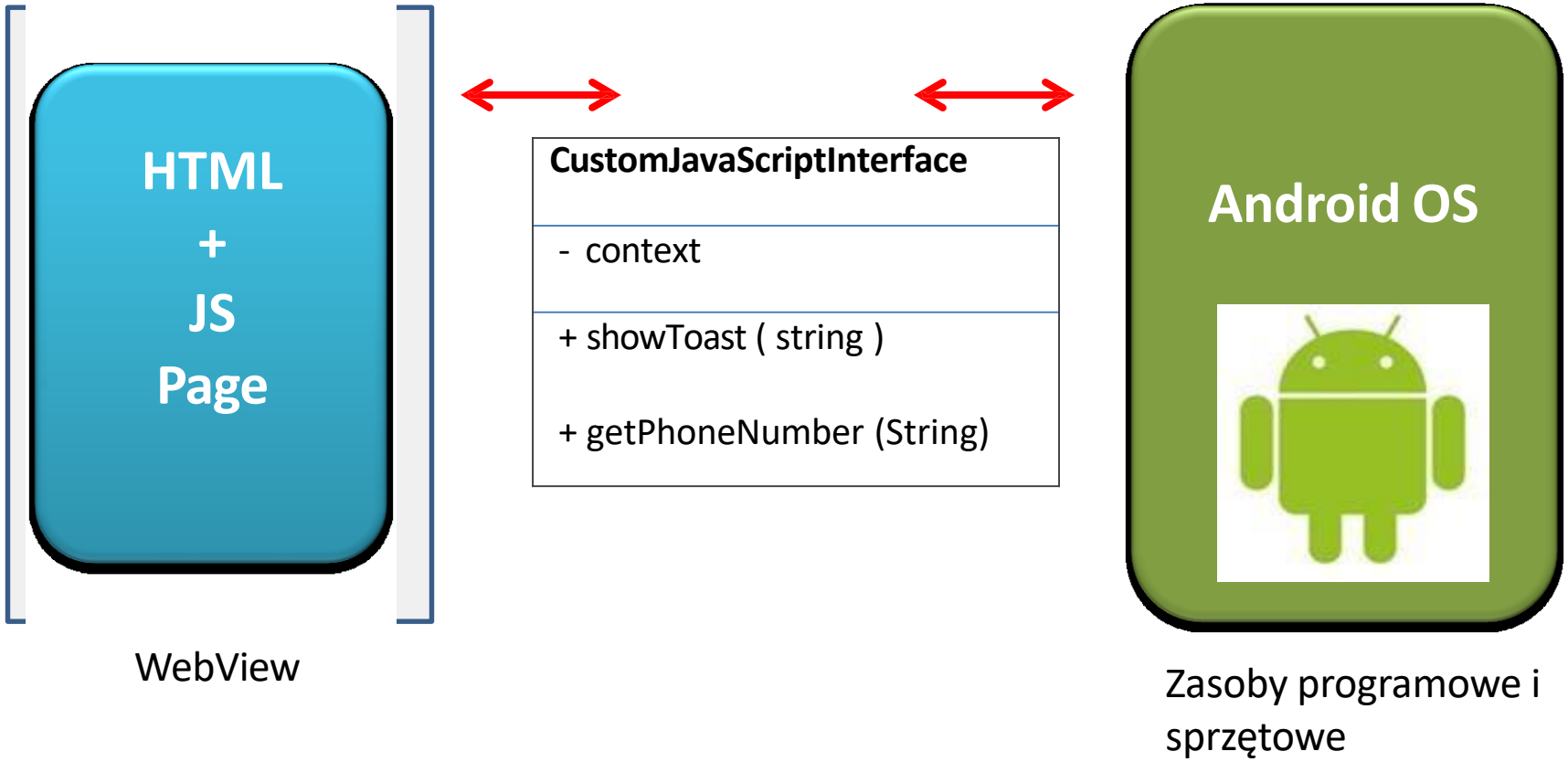
1. Część wizualna aplikacji składa się z dynamicznych stron internetowych w języku HTML. Lokalne strony wykorzystują język JavaScript do implementacji ich działania. Prezentowane są z wykorzystaniem komponentu WebView.

2. Obiekt łączący jest tworzony jako składowa aplikacji Android i przekazywany do wykorzystania w komponencie WebView.

3. Obiekt łączący pozwala na interakcję zasobów html ze wszelkimi metodami czy zasobami sprzętowymi dostępnymi po stronie języka Java. Dostęp odbywa się z poziomu języka skryptowego, poprzez wywołanie odpowiednich metod obiektu łączącego.

Przykład 2. Aplikacje hybrydowe

Metody w obiekcie łączącym powinny być publiczne by kod skryptu mógł je wywoływać.



WebView

Przykład 2. Aplikacje hybrydowe

1. Aplikacja prezentuje przechowywaną lokalnie stronę internetową w komponencie WebView.

2. Aplikacja oczekuje wpisania nazwy użytkownika i przekazuje ją dalej.

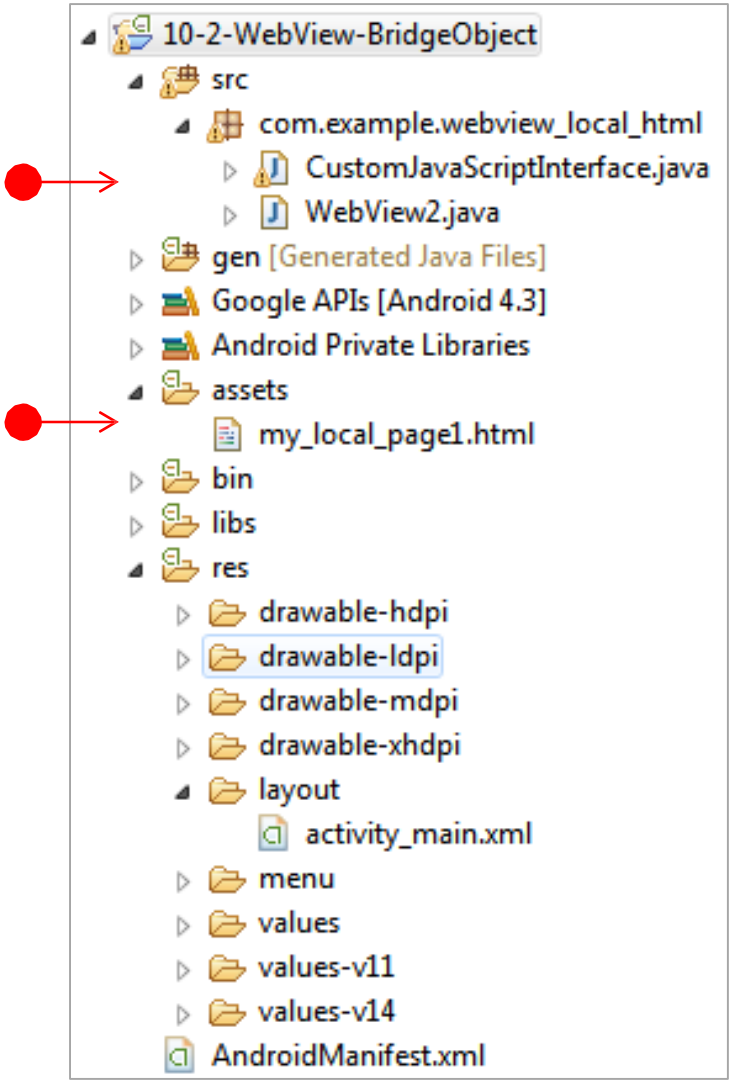
3. Android dokonuje przeszukania bazy kontaktów i zwraca numer telefonu przypisany do danego kontaktu.

4. Użytkownikowi prezentowany jest komunikat typu Toast.

WebView

Przykład 2. Aplikacje hybrydowe

Struktura aplikacji



Układ: activity_main.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
   xmlns:tools="http://schemas.android.com/tools
   " android:layout_width="match_parent"
   android:layout_height="match_parent" >

    <WebView android:id="@+id/webView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

WebView

Przykład 2. MainActivity - WebView2. java

```
public class WebView2 extends Activity {

   @SuppressLint("SetJavaScriptEnabled")
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      WebView webview = (WebView) findViewById(R.id.webView1);
      webview.getSettings().setJavaScriptEnabled(true);

      webview.addJavascriptInterface(new CustomJavaScriptInterface(this),
                                     "HtmlAndroidBridge");

      // if the HTML file is in the app's memory space use:
         webview.loadUrl("file:///android_asset/my_local_page1.html");

      // if the HTML files are stored in the SD card, use:
      // webview.loadUrl("file:///sdcard/my_local_webpage1.html");

      // CAUTION: Manifest must include
      // <uses-permission android:name="android.permission.INTERNET"/>
      // <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

   }//onCreate

}//class
```

Przykład 2. CustomJavaScriptInterface.java

```
public class CustomJavaScriptInterface {
    private Context context;

    CustomJavaScriptInterface(Context context) {
        // remember the application's context
        this.context = context;
    }

    @JavascriptInterface
    public void showToast(String toastMsg) {
        // instead of dull JavaScript alert() use flashy Toasts
        Toast.makeText(context, toastMsg, Toast.LENGTH_SHORT).show();
    }

    @JavascriptInterface
    public String getPhoneNumber(String person) {
        // accept a person's name and fake that you are
        // searching the Android's Contacts Database
        person = person.toLowerCase();
        if (person.equals("jose cuervo"))
            return "555-1111";
        else if (person.equals("macarena"))
            return "555-2222";
        else
            return "555-3333";
    }
}
```

Przykład 2. CustomJavaScriptInterface.java

Komentarz

Klasa **CustomJavaScriptInterface** stanowi most między zawartością HTML oraz systemem Android. Programista decyduje jakie dane są dostępne oraz definiuje odpowiednie akcesory czy modyfikatory.

Obiekt tworzony jest w języku Java a referencja do niego jest przekazywana do środowiska skryptowego poprzez:

```
webview.addJavascriptInterface( new CustomJavaScriptInterface(this),
                                "HtmlAndroidBridge"   );
```

Kod skryptu zawarty jest w stronach HTML (wyświetlanych przez komponent webview) i uzyskuje dostęp do współdzielonych metod poprzez:

```
HtmlAndroidBridge.nazwaMetody(arg)
```

Od API 17, tego typu metody muszą posiadać adnotację `@JavascriptInterface` (inaczej są ignorowane).

WebView

Przykład 2. HTML – my_local_page1.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
 <title>Android GetPhone Demo</title>
 <script>
  function setPerson() {
    // move name 'Jose Cuervo' to input box
    document.getElementById("myName").value = 'Jose Cuervo' ;
  }
  function askAndroidToFindPhone() {
    // bridge object used to send local (html) data to android app
    // passing a person's name, waiting for phone to be returned
    var person =  document.getElementById("myName").value;
    HtmlAndroidBridge.showToast('Android is  busy - Searching phones... ' + person);
    document.getElementById("myPhone").value = HtmlAndroidBridge.getPhoneNumber(person);
  }
</script>
</head>
 <body>
  <p>Interacting: HTML & Android</p>
  <p><a onClick="setPerson()"><u>Click to set Person(Jose Cuervo)</u></a></p>
  <p> Person's name  <input type="text" id="myName" />
  <p> Phone Number   <input type="text" id="myPhone" />
  <p> <input type="button" onclick= "askAndroidToFindPhone()"
            value="Asking Android to Get Phone">
 </body>
</html>
```

Przykład 2. HTML – my_local_page1.html

Komentarz

1. Instrukcja:

```
HtmlAndroidBridge.showToast('Android is busy - Searching phones... ' + person);
```

która jest wywoływana z metody JS, przekazuje ciąg znaków i nakazuje systemowi Android wyświetlić komunikat typu Toast z zadanym tekstem.

2. Wyrażenie:

```
HtmlAndroidBridge.getPhoneNumber(person);
```

umożliwia przeszukanie książki adresowej pod kątem zadanej osoby. Jeżeli wyszukiwanie się powiedzie, numer telefonu tej osoby jest przekazywany do strony internetowej.

Przykład 3. Wykorzystanie istniejących bibliotek

W tym przykładzie, do aplikacji zostanie dołączona biblioteka **Google Maps API,** która umożliwia wzbogacenie poprzednich przykładów.

Dlaczego?

Zalety systemu Android:

1. Dostęp do natywnych mechanizmów (np. lokalizacyjnych).
2. Łatwe dodanie do sklepu Play.
3. Szybsze tworzenie aplikacji korzystając z mechanizmów RAD.

Zalety Google Maps:

1. Główny kod aplikacji znajduje się na serwerze.
2. Dokonane zmiany dostępne są natychmiast dla użytkowników.
3. Częstsze aktualizacje bezpośrednio od Google.
4. Wieloplatformowość: aplikacja działa na wielu różnych platformach.

WebView

Przykład 3. Wykorzystanie istniejących bibliotek

Jakie są założenia?

- Prezentacja mapy danego obszaru z użyciem **Google Maps JavaScript API V3**. (https://developers.google.com/maps/documentation/javascript/)

- Mapa powinna być wyposażona w najnowsze funkcje i być aktualna.

Jak to zrealizować?

- Stwórz standardową aplikację internetową **HTML + JS.** Może być ona tworzona z użyciem standardowych narzędzi wspomagających programowanie aplikacji internetowych.

- Stwórz aplikację działającą pod kontrolą systemu Android, która wyświetla wspomnianą stronę internetową w komponencie typu WebView.

Przykład 3. Wykorzystanie istniejących bibliotek

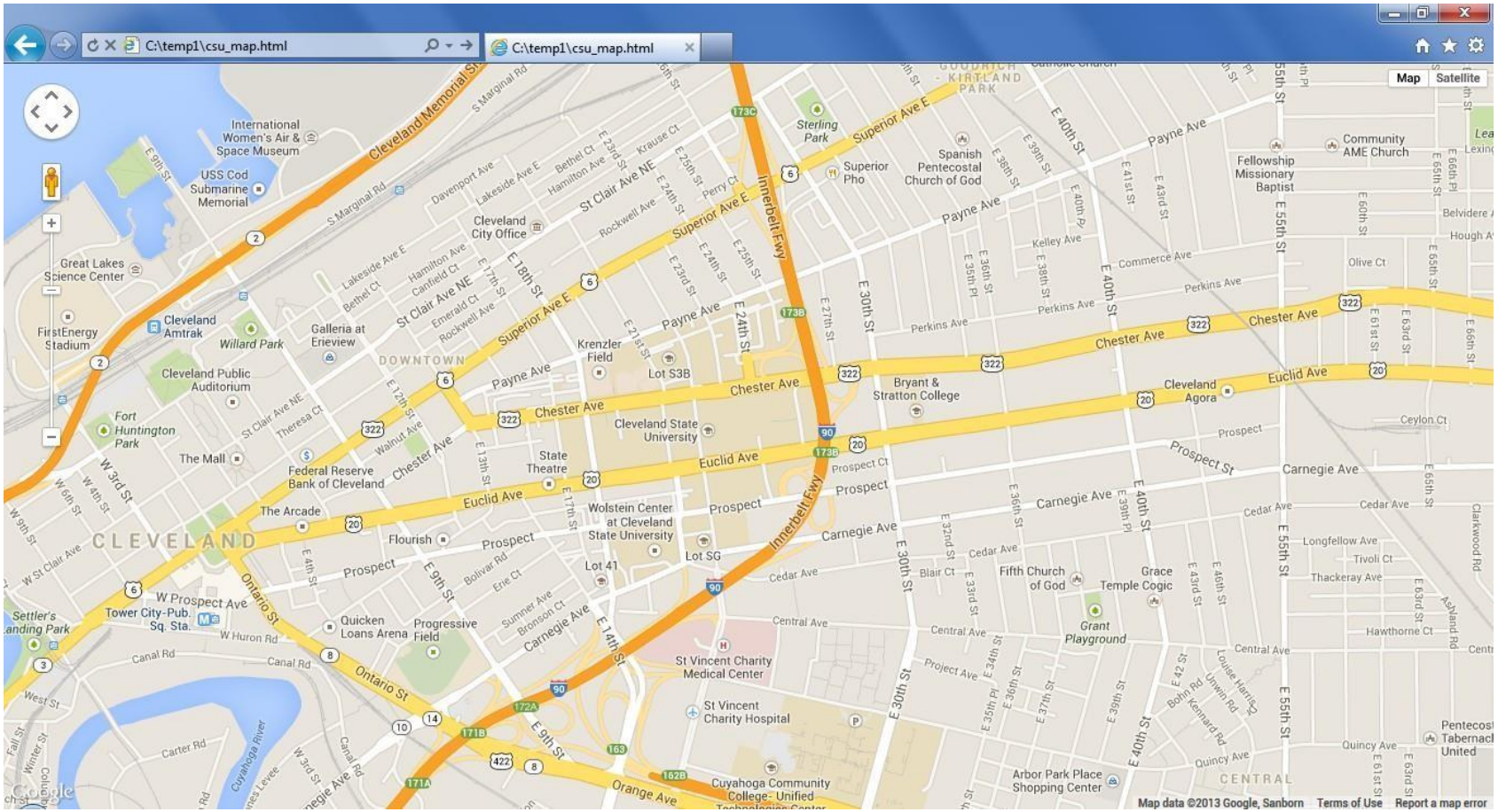
Czym jest Google Maps JavaScript API V3 ?

- **Google Maps Javascript API** jest darmową usługą pozwalającą na osadzenie map google we własnych aplikacjach internetowych.
- Została stworzona z myślą o urządzeniach mobilnych (aczkolwiek sprawdza się również w przypadku urządzeń typu komputer stacjonarny)
- Posiada wiele metod do manipulowania wyświetlaniem tego typu danych (tak jak strona internetowa http://maps.google.com) oraz dodawania własnych warstw graficznych (np. znaczników pozycji).

Link: https://developers.google.com/maps/documentation/javascript/

Przykład 3. Wykorzystanie istniejących bibliotek

Wykorzystanie Google Maps JavaScript API V3



Link: https://developers.google.com/maps/documentation/javascript/basics?csw=1

Przykład 3. Wykorzystanie istniejących bibliotek

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
1 <style type="text/css">
  html { height: 100% }
  body { height: 100%; margin: 0px; padding: 0px }
  #map_canvas { height: 100% }
 </style>
2 <script type="text/javascript"
          src="http://maps.google.com/maps/api/js?sensor=false">
 </script>
 <script type="text/javascript">
3 function initialize() {
    var latlng = new google.maps.LatLng(41.5020952, -81.6789717);
    var myOptions = {
      zoom: 15,
      center: latlng,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
  }
 </script>
 </head>
 <body onload="initialize()">
4      <div id="map_canvas" style="width:100%; height:100%"></div>
 </body>
</html>
```

google_map.html

Strona wyświetlająca mapę google

Przykład 3. Wykorzystanie istniejących bibliotek

Komentarz

google_map.html to strona HTML odpowiedzialna za rysowanie mapy.

1. Znacznik **<style>** ustawia cały ekran (100% szerokości i wysokości) jako płótno na którym rysowania będzie cała mapa.

2. Pierwszy fragment w **<script>** wywołuje metodę API do rysowania mapy. Nie podano jeszcze konkretnej lokalizacji (mapa nie zostanie jeszcze wyświetlona).

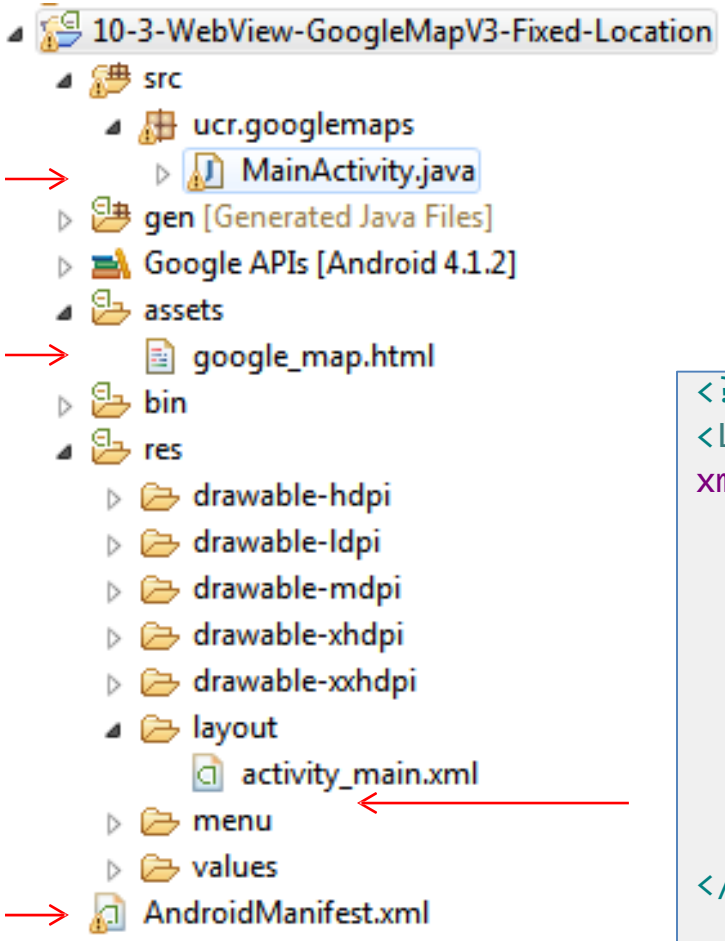
3. Funkcja **Initialize** ustawia przyporządkowanie argumentów. Mapa jest centrowana wokół podanych współrzędnych oraz ustawiany jest poziom powiększenia.

4. Po załadowaniu strony HTML, funkcja **Initialize** jest wywoływana i użytkownikowi prezentowana jest odpowiednia mapa.

WebView

Przykład 3. Wykorzystanie istniejących bibliotek

Struktura aplikacji:



Tworzenie aplikacji:

1. Umieść komponent **WebView** w pliku activity_main.xml.
2. Dodaj stronę internetową do folderu **assets.**
3. Dodaj uprawnienie **Internet** do pliku manifestu.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

  <WebView android:id="@+id/webview"
      android:layout_width="match_parent"
      android:layout_height="match_parent"/>

</LinearLayout>
```

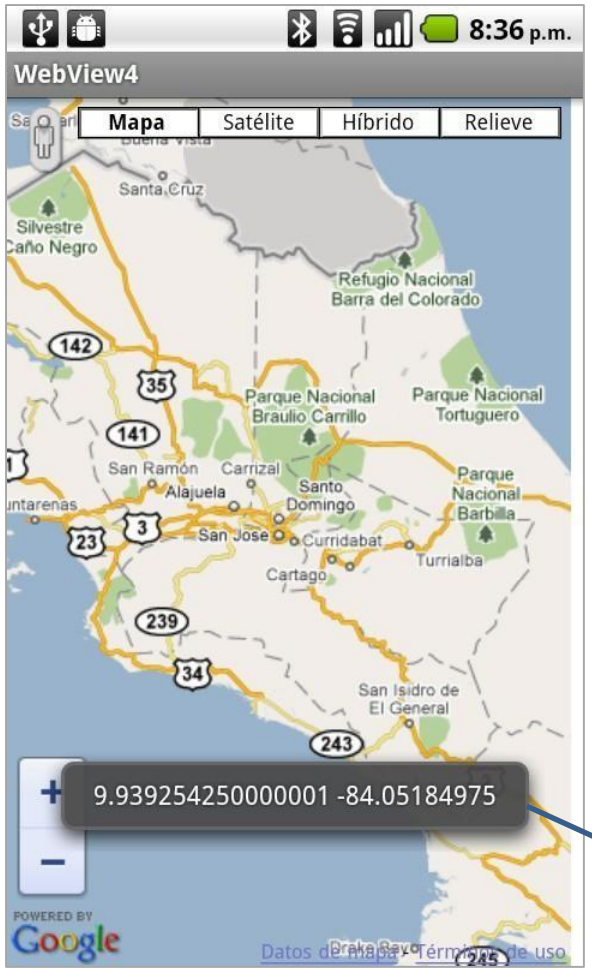
WebView

Przykład 3. Wykorzystanie istniejących bibliotek

```
public class MainActivity extends Activity {
    WebView webview;

    @SuppressLint("SetJavaScriptEnabled")
    @Override
    public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // load into WebView local HTML page defining mapping
        operation webview = (WebView) findViewById(R.id.webview);
        webview.getSettings().setJavaScriptEnabled(true);

        webview.loadUrl("file:///android_asset/google_map.html");
    }//onCreate

}//class
```

Jedynym zadaniem aktywności jest załadowanie określonej strony internetowej.

Przykład 4. Zbieranie danych GPS



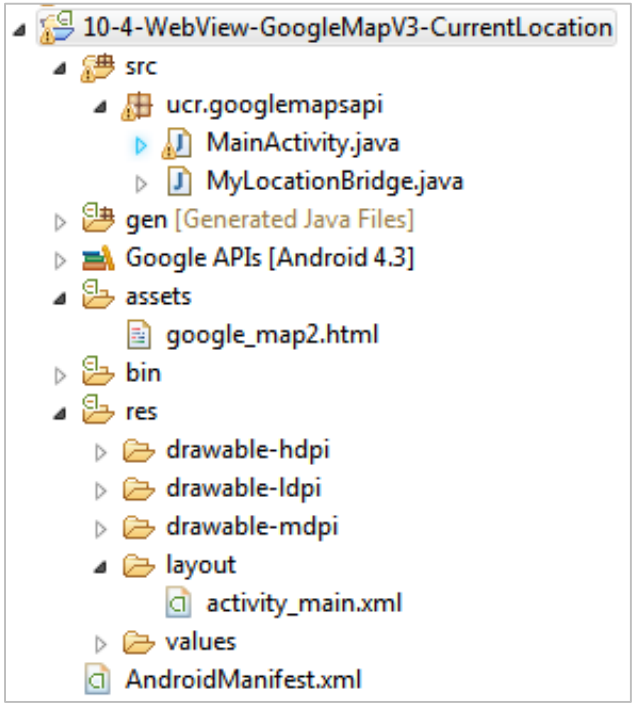
Przykład łączy ze sobą dwa poprzednie:

- Moduł GPS określa rzeczywiste położenie urządzenia mobilnego w danej chwili czasu.
- Obiekt JavaScript przekazuje ustalone współrzędne do strony HTML.
- Strona zawiera kod wyświetlający mapę wycentrowaną na ustalone współrzędne geograficzne.

Rzeczywiste współrzędne geograficzne

Przykład 4. Zbieranie danych GPS

Struktura aplikacji



Układ - activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

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

</LinearLayout>
```

Przykład 4. MainActivity.java

```
public class MainActivity extends Activity implements LocationListener {
   // a LocationListener could use GPS, Cell-ID, and WiFi to establish
   // a user's location. Deciding which to use is based on choices
   // including factors such as accuracy, speed, and battery-efficiency.

   private WebView webview;
     LocationManager locationManager;
     MyLocationBridge locationBridge = new MyLocationBridge();

   @Override
   protected void onDestroy() {
      super.onDestroy();
      // cut location service requests
      locationManager.removeUpdates(this);
   }

   private void getLocation() {
      locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
      Criteria criteria = new Criteria();
      // criteria.setAccuracy(Criteria.ACCURACY_FINE); //use GPS (you should be outside)
      criteria.setAccuracy(Criteria.ACCURACY_COARSE); // cell towers, wifi
      String provider = locationManager.getBestProvider(criteria, true);

      // In order to make sure the device is getting the location, request
      // updates [wakeup after changes of: 5 sec. or 10 meter]
      locationManager.requestLocationUpdates(provider, 5, 10, this);
      locationBridge.setNewLocation(locationManager.getLastKnownLocation(provider));
   }
```

Przykład 4. MainActivity.java

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getLocation();
        setupWebView();
        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    }//onCreate

    // Set up the webview object and load the page's URL
    @SuppressLint("SetJavaScriptEnabled")
    private void setupWebView() {
        final String centerMapURL = "javascript:centerAt("
                                  + locationBridge.getLatitude() + ","
                                  + locationBridge.getLongitude() + ")";
        // set up the webview to show location results webview =
        (WebView) findViewById(R.id.webview);
        webview.getSettings().setJavaScriptEnabled(true);
        webview.addJavascriptInterface(locationBridge, "locationBridge");
        webview.loadUrl("file:///android_asset/google_map2.html");
        // Wait for the page to load then send the location information
        webview.setWebViewClient(new WebViewClient() {
            @Override
            public void onPageFinished(WebView view, String url) {
                webview.loadUrl(centerMapURL);
            }
        });

    }
```

Przykład 4. MainActivity.java

```
    @Override
    public void onLocationChanged(Location location) {
        String lat = String.valueOf(location.getLatitude());
        String lon = String.valueOf(location.getLongitude());
        Toast.makeText(getApplicationContext(), lat + "\n" + lon, 1).show();
        locationBridge.setNewLocation(location);
    }

    @Override
    public void onProviderDisabled(String provider) {
        // needed by Interface. Not used
    }

    @Override
    public void onProviderEnabled(String provider) {
        // needed by Interface. Not used
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        // needed by Interface. Not used
    }


}//class
```

Przykład 4. MyLocationBridge.java

```
// An object of type "MyLocationBridge" will be used to pass data back and
// forth between the Android app and the JS code behind the HTML page.

public class MyLocationBridge {
        private Location mostRecentLocation;

         @JavascriptInterface
        public void setNewLocation(Location newCoordinates){
                mostRecentLocation = newCoordinates;
        }
         @JavascriptInterface
        public double getLatitude() {
                if (mostRecentLocation == null) return (0);
                 else return mostRecentLocation.getLatitude();

        }
         @JavascriptInterface
        public double getLongitude() {
                if (mostRecentLocation == null) return (0);
                 else return mostRecentLocation.getLongitude();
        }

}// MyLocationFinder
```

Przykład 4. google_map2.html

```
<!DOCTYPE html>
<html>
     <head>
     <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
     <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
     <title>Google Maps JavaScript API v3 Example: Marker Simple</title>

     <style type="text/css">
       html { height: 100% }
       body { height: 100%; margin: 0px; padding: 0px }
       #map_canvas { height: 100% }
     </style>

 <script src="http://maps.google.com/maps/api/js?sensor=false"></script>
 <script type="text/javascript">

  function initialize() {
    // get latitude and longitude from the BRIDGE object (JavaScripInterface)
    var myLatlng = new google.maps.LatLng(locationBridge.getLatitude(),
                                          locationBridge.getLongitude());
    var myOptions = {
      zoom: 17,
      center: myLatlng,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    }
```

Przykład 4. google_map2.html

```
    var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);

    var marker = new google.maps.Marker({
        position: myLatlng,
        map: map
    });

  }
</script>

</head>
 <body onload="initialize()">
      <div id="map_canvas"></div>
 </body>
</html>
```

Przykład bazuje na:

http://code.google.com/apis/maps/articles/android_v3.html
http://code.google.com/apis/maps/documentation/javascript/overlays.html#MarkerAnimations
http://gmaps-samples.googlecode.com/svn/trunk/articles-android-webmap

WebView

Przykład 4. Testowanie przy użyciu emulatora

DDMS - 10-4-WebView-GoogleMapV3-CurrentLocation/src/ucr/googlemapsapi/Main.java - Eclipse

File Edit Refactor Source Navigate Search Project Run Window Help

Devices

Name

IceCreamMx-16 [emulator-5554] Onlin

com.android.deskclock 340
com.android.systemui 201
com.android.calendar 313
com.android.providers.calenc 380
com.android.phone 229
com.android.mms 407
com.android.exchange 467
android.process.acore 294
android.process.media 417
com.android.launcher 244
system_process 146
com.android.inputmethod.lat 215
com.android.settings 270
com.android.email 449
com.android.defcontainer 550
com.android.keychain 566
com.svox.pico 580
com.android.quicksearchbox 594
ucr.googlemapsapi 625
com.example.playingwithwel 899

File Explorer | Emulator Control

Telephony Status

Voice: home Speed: Full

Data: home Latency: None

Telephony Actions

Incoming number:

Voice

SMS

Message

Call Hang Up

Location Controls

Manual GPX KML

Decimal

Sexagesimal

Longitude -122.084095

Latitude 37.422006

Send

WebView

Pytania ?

Co dalej?

Google Maps Developers
https://developers.google.com/maps/documentation/

Google Maps API – Webservices
http://code.google.com/apis/maps/documentation/webservices/index.html

Google Maps JavaScript API V3
http://code.google.com/apis/maps/documentation/javascript/tutorial.html

DODATEK A. Sugerowane materiały dodatkowe

Oracle Mobile Application Framework
http://www.oracle.com/technetwork/developer-tools/maf/overview/index.html
Tworzenie aplikacji pod Android oraz iOS wykorzystując platformę Oracle MAF

Intel App Framework
http://app-framework-software.intel.com/
Biblioteka JavaScript do tworzenia wieloplatformowych aplikacji w HTML5

Wskazówki wydajnościowe dla Geo API
https://www.youtube.com/v/zI8at1EmJjA

Dokumentacja Google Maps Developers
https://developers.google.com/maps/documentation/

Kurs Google Maps JavaScript API V3
http://code.google.com/apis/maps/documentation/javascript/tutorial.html

Tworzenie aplikacji wykorzystujących WebView
http://developer.android.com/guide/webapps/webview.html

Debugowanie aplikacji internetowych
http://developer.android.com/guide/webapps/debugging.html

Dobre praktyki dla aplikacji internetowych
http://developer.android.com/guide/webapps/best-practices.html

DODATEK B. Rzeczywista funkcja do przeszukiwania książki adresowej – prosta, choć mało efektywna wersja

Dodaj do pliku manifestu:

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

```
public String findPhoneNumber(String searchName) {
  //look for first contact entry partially matching searchName

  String name = "n.a.";
  String number = "n.a.";
  //defina an iterator to traverse the contact book
  Cursor phones = getContentResolver().query(
         ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
         null, null,  null, null);

  while (phones.moveToNext()) {
      String NAME = ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME;
      name = phones.getString(phones.getColumnIndex(NAME));
      String NUMBER = ContactsContract.CommonDataKinds.Phone.NUMBER;
      number = phones.getString(phones.getColumnIndex(NUMBER));

      if (name.toLowerCase().contains(searchName.toLowerCase())) {
        return number; // a good match, phone number found
    }
  }
  // Not found
  return "n.a.";
}// findPhoneNumber
```