ANDROID platforma
Multimedia & WebKit

Mobilni i distribuirani informacioni sistemi

Mr Bratislav Predić

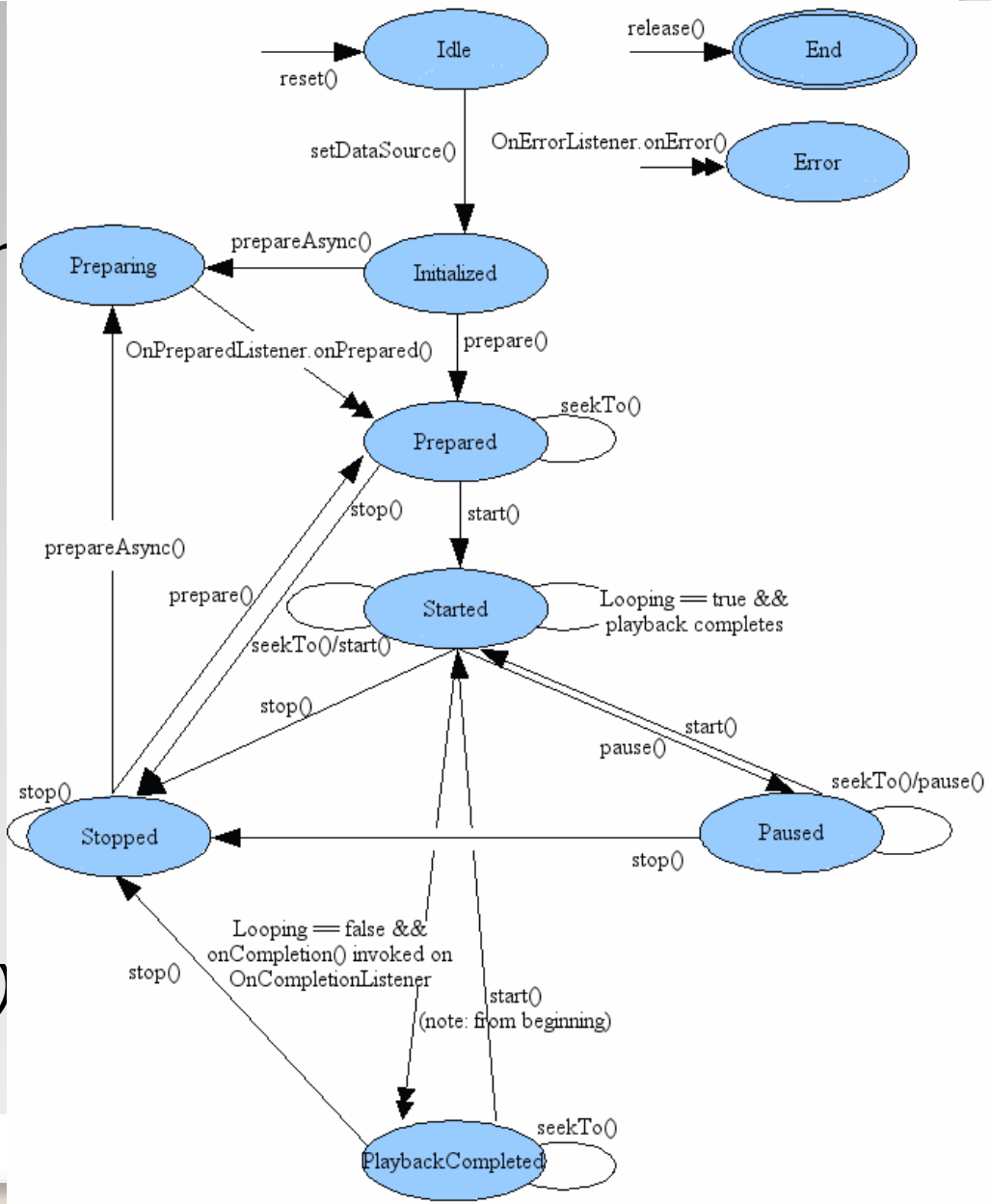
2012. godina

Android multimedia

- Multimedia playback na Android sistemu obavlja klasa *MediaPlayer*
- Tokom upravljanja multimedia fajlova klasa *MediaPlayer* se ponaša kao state-machine (konačni automat) sa stanjima:
  - Inicijalizacija media plejera multimedija sadržajem
  - Priprema multimedijalnog sadržaja
  - Puštanje sadržaja (playback)
  - Pauziranje ili zaustavljanje sadržaja pre kraja
  - Završetak reprodukcije multimedijalnog sadržaja

Android multimedia

- Neophodni koraci pre reprodukcije
  - Kreiranje *MediaPlayer* instance
  - Inicijalizacija sadržajem
  - Priprema za reprodukciju
- Kada više nije potreban obavezno osloboditi resurse
  - *mediaPlayer.release()*



Android multimedia

- Izvori multimedijalnog sadržaja
  - Uključen kao resurs aplikacije
  - Uskladišten u lokalnim datotekama (na memorijskoj kartici)
  - Preko *ContentProvider*-a iz drugih aplikacija
  - Kao stream sa udaljenih lokacija preko mreže
- Multimedia kao lokalni resursi se smeštaju u *res/raw*
- Raw resursi se ne kompresuju prilikom kreiranja apk paketa aplikacije
- Ovakvi resursi se referenciraju jednostavnim imenom, bez ekstenzije

Android multimedia

- Kreiranje *MediaPlayer* objekta
  - Statičkom metodom *create*
    Prosleđujemo aplikacioni kontekst i multimedia sadržaj kao
    - Naziv iz resursa
    - URI do lokalne datoteke *file://*
    - URI do online resursa (URL)
    - URI do lokalnog *ContentProvider*-a

```
Context appContext = getApplicationContext();

MediaPlayer resourcePlayer = MediaPlayer.create(appContext, R.raw.my_audio);
MediaPlayer filePlayer = MediaPlayer.create(appContext,
        Uri.parse("file:///sdcard/localfile.mp3"));
MediaPlayer urlPlayer = MediaPlayer.create(appContext,
        Uri.parse("http://site.com/audio/audio.mp3"));
MediaPlayer contentPlayer = MediaPlayer.create(appContext,
        Settings.System.DEFAULT_RINGTONE_URI);
```

Android multimedia

- *MediaPlayer* instanciran statičkom *create* metodom je automatski pripremljen
- **NE SME** ponovo da se poziva *prepare* metoda
- Alternativno, na postojećem *MediaPlayer* objektu može se pozvati metoda *setDataSource()*
- U ovom slučaju je **OBAVEZNO** da se pozove metoda *prepare* pre reprodukcije
- Moguće je više puta koristiti *MediaPlayer*

```
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setDataSource("/sdcard/test.3gp");
mediaPlayer.prepare();
```

Android kamera i fotografije

- Praktično svi današnji Android telefoni imaju barem jednu (zadnju) integrisanu kameru
- Najjednostavniji način za snimanje fotografije je ako s eiskoristi postojeća (sistemska) aktivnost
- Kreiramo *Intent* sa akcijom *ACTION_IMAGE_CAPTURE* i startujem aktivnost sa *startActivityForResult()*

```
startActivityForResult(
        new Intent(MediaStore.ACTION_IMAGE_CAPTURE),
                        TAKE_PICTURE);
```

- Na dalje se koristi sistemska aktivnost sa svim standardnim funkcionalnostima

Android kamera i fotografije

- Image capture akcija snima fotografije u dva režima
  - Thumbnail – podrazumevano, snima se thumbnail *Bitmap* u *data* promenljivoj extras dela *Intent*-a koji je vratio poziv *onActivityResult()*
  - Full image – ako se u *Intent*-u kojim se poziva aktivnost kamere doda extra URI pod nazivom *MediaStore.EXTRA_OUTPUT* na tu lokaciju se snima velika slika
- Rezultat snimanja fotografije se hvata u callback metodi *onActivityResult()*

Android kamera i fotografije

- Primer

```
private static int TAKE_PICTURE = 1;
private Uri outputFileUri;
private void getThumbailPicture() {
  Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
  startActivityForResult(intent, TAKE_PICTURE);
}
private void saveFullImage() {
  Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
  File file = new File(Environment.getExternalStorageDirectory(), test.jpg");
  outputFileUri = Uri.fromFile(file);
  intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
  startActivityForResult(intent, TAKE_PICTURE);
}
@Override
protected void onActivityResult(int requestCode,int resultCode, Intent data){
  if (requestCode == TAKE_PICTURE) {
    Uri imageUri = null;
    // Check if the result includes a thumbnail Bitmap
    if (data != null) {
      if (data.hasExtra("data")) {
        Bitmap thumbnail = data.getParcelableExtra("data");
        // TODO Do something with the thumbnail
      }
    } else {
      // TODO Do something with the full image stored in outputFileUri
    }
```

Android kamera i fotografije

- Direktan pristup kameri zahteva permission u manifestu

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

- Koristi se klasa *Camera* za zadavanje preferenci kamere, parametara fotografije i iniciranje snimanje fotografije
- Za instanciranje *Camera* klase se koristi statički *open()* metod
- Po završetku rada sa kamerom **OBAVEZNO** je treba osloboditi

```
Camera camera = Camera.open();
[ . . . Do things with the camera . . . ]
camera.release();
```

Parametri kamere

- Podešavanja kamere su snimljena u objektu *Camera.Parameters*
- Ovim parametrima se pristupa metodom *getParameters()* klase *Camera*
- Zatim se radi sa objektom *Parameters* ()
- Na kraju se postavlja modifikovani objekat *Parameters* na kameri metodom *setParameters()*

```
Camera.Parameters parameters = camera.getParameters();
[ . . . make changes . . . ]
camera.setParameters(parameters);
```

- Parametri kamere: *SceneMode, FlashMode, WhiteBalance, ColorEffect, FocusMode*

Parametri kamere

- Parametri kamere takođe zadaju karakteristike snimljene fotografije, thumbnail slike i preview slike
  - *setJpegQuality(), setJpegThumbnailQuality()*
  - *setPictureSize(), setPreviewSize(), setJpegThumbnailSize()*
  - *setPictureFormat(), setPreviewFormat()*
  - *setPreviewFrameRate()*
- Različiti uređaji podržavaju različite parametre, pre postavljanja potrebno je proveriti odgovarajućim metodama:
  - *getSupported...()*

Parametri kamere

- Provera podržanih parametara

```
Camera.Parameters parameters = camera.getParameters();
List<String> colorEffects = parameters.getSupportedColorEffects();
if (colorEffects.contains(Camera.Parameters.EFFECT_SEPIA))
  parameters.setColorEffect(Camera.Parameters.EFFECT_SEPIA);
camera.setParameters(parameters);
```

- Calback za autofokus (kada snimiti fotografiju)
  - *AutoFocusCallback* objekta *Camera*

```
camera.autoFocus(new AutoFocusCallback() {
  public void onAutoFocus(boolean success, Camera camera) {
    // TODO Do something on Auto-Focus success
  }
});
```

- Zove se svaki put kada se fokus promeni
- Parametar *success* indikator uspešnosti fokusiranja

Kamera preview

- Pristup video stream-u kamere (često u augmented reality aplikacijama)
- Preview kamere se prikazuje na *SurfaceHolder*
- U UI aplikacije treba uključiti *SurfaceView*
- Uspešnost kreiranja surface-a hvatamo u callback metodi *SurfaceHolder.Callback*
- Posle toga prosledimo kreirani surface objektu *Camera* korišćenjem metode *setPreviewDisplay()*
- Metode *startPreview()* i *stopPreview()* kontorlišu prikaz preview-a na ekranu

Kamera preview - primer

```
public class MyActivity extends Activity implements SurfaceHolder.Callback {
  private Camera camera;
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    SurfaceView surface = (SurfaceView)findViewById(R.id.surface);
    SurfaceHolder holder = surface.getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    holder.setFixedSize(400, 300);
  }
  public void surfaceCreated(SurfaceHolder holder) {
    if (mediaRecorder == null) {
      try {
        camera = camera.open();
        camera.setPreviewDisplay(holder);
        camera.startPreview();
        [ . . . Draw on the Surface . . . ]
      } catch (IOException e) { Log.d("CAMERA", e.getMessage()); }
    }
  }
  public void surfaceDestroyed(SurfaceHolder holder) {
    camera.stopPreview();
    camera.release();
  }
}
```

Kamera preview

- Moguće je zadati i *PreviewCallback* da se pozove svaki put kada imamo spreman preview frame
- Tako možemo ručno modifikovati i crtati svaku sličicu preview stream-a
- Metodi *setPreviewCalllback()* objekta *Camera* prosleđujemo implementaciju *PreviewCallback* sa override-ovanom metodom *onPreviewFrame()*

```
camera.setPreviewCallback(new PreviewCallback() {
  public void onPreviewFrame(byte[] _data, Camera _camera) {
    // TODO Do something with the preview image.
  }
});
```

Snimanje fotografije

- Implementirano kroz tri callback-a
  - *ShutterCallback*
  - Dva *PictireCallback*-a (za RAW i JPEG slike)
- *ShutterCallback* se poziva čim se zatvori zatvarač
- *PictureCallback* dobija niz bajtova sa sadržajem slike u odgovarajućem formatu
- Primer:
  Manuelno snimamo fotografiju i snimamo je u JPEG formatu na CD karticu

Snimanje fotografije - primer

```
private void takePicture() {
  camera.takePicture(shutterCallback, rawCallback, jpegCallback);
}

ShutterCallback shutterCallback = new ShutterCallback() {
  public void onShutter() {
    // TODO Do something when the shutter closes.
  }
};

PictureCallback rawCallback = new PictureCallback() {
  public void onPictureTaken(byte[] data, Camera camera) {
    // TODO Do something with the image RAW data.
  }
};

PictureCallback jpegCallback = new PictureCallback() {
  public void onPictureTaken(byte[] data, Camera camera) {
    // Save the image JPEG data to the SD card
    FileOutputStream outStream = null;
    try {
      outStream = new FileOutputStream("/sdcard/test.jpg");
      outStream.write(data);
      outStream.close();
    } catch (FileNotFoundException e) { Log.d("CAMERA", e.getMessage()); }
    catch (IOException e) { Log.d("CAMERA", e.getMessage()); }
  }
};
```

JPEG EXIF podaci

- EXIF (EXchangeable Image File format) Sadrži dodatne informacije o snimljenoj fotografiji
  - Datum i vreme
  - Podešavanja kamere
  - Thumbnail
  - Opis
  - **Geolokaciju** (koordinate gde je snimljena fotografija)
- Za pristup EXIF informacijama slike se koristi klasa *ExifInterface*

```
ExifInterface exif = new ExifInterface(filename);
```

JPEG EXIF podaci

- Objekat *ExifInterface* sadrži kolekciju atributa koji ma se pristupa metodama
  - *getAttribute()*
  - *setAttribute()*
- Atributi se identifikuju statičkim konstantama koje su definisane u *ExifInterface (TAG_*)*

```
File file = new File(Environment.getExternalStorageDirectory(),"test.jpg");
try {
  ExifInterface exif = new ExifInterface(file.getCanonicalPath());
  // Read the camera model and location attributes
  String model = exif.getAttribute(ExifInterface.TAG_MODEL);
  float[] latLng = new float[2];
  exif.getLatLong(latLng);
  // Set the camera make
  exif.setAttribute(ExifInterface.TAG_MAKE, "My Phone");
} catch (IOException e) { Log.d("EXIF", e.getMessage()); }
```

Android WebKit browser

- Sistemski browser je moguće koristiti kao widget kreiranim aktivnostima
- Android browser je zasnovan na WebKit engine-u, isto kao i Apple Safari
- *WebView* widget se koristi za prikaz HTML sadržaja unutar aplikacije
- *WebView* će automatski koristiti bilo koji tip dostupne mrežne konekcije za preuzimanje sadržaja

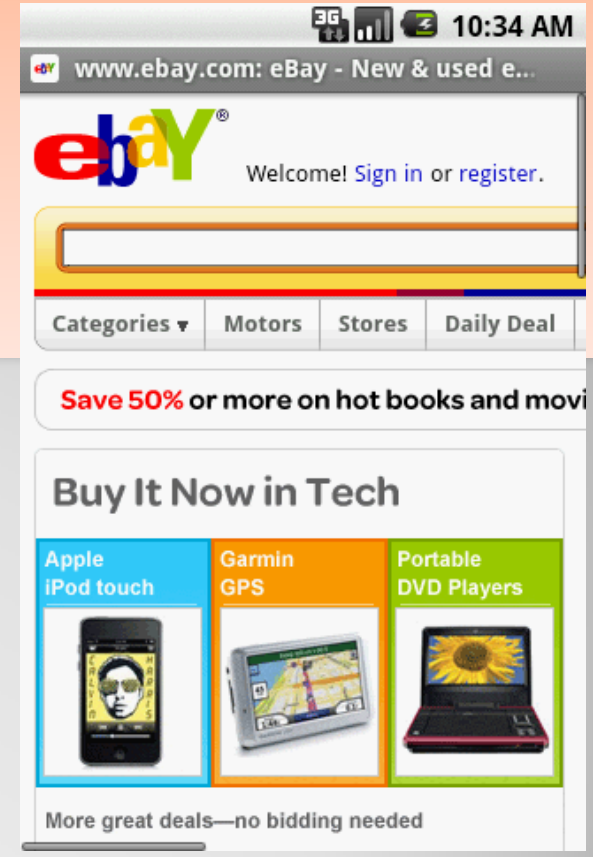
Android WebKit browser

- WebKit rendering engine podržava sve standarne funkcije za navigaciju
  - Navigaciju napred/nazad kroz istoriju
  - Zoom in / out
  - Text search na HTML stranici
  - Učitavanje sadržaja
  - Zaustavljanje učitavanja
  - ...
- Korišćenje WebView komponente zahtva INTERNET permission u manifestu

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

Android WebKit browser

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



- Iz koda aplikacije *(onCreate)* radimo navigaciju

Android WebKit browser

```
public class AndDemoUI extends Activity {
  WebView browser;

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    browser = (WebView)findViewById(R.id.webkit);
    browser.loadUrl("http://eBay.com");
    browser.getSettings().setJavaScriptEnabled(true);
  }
}
```

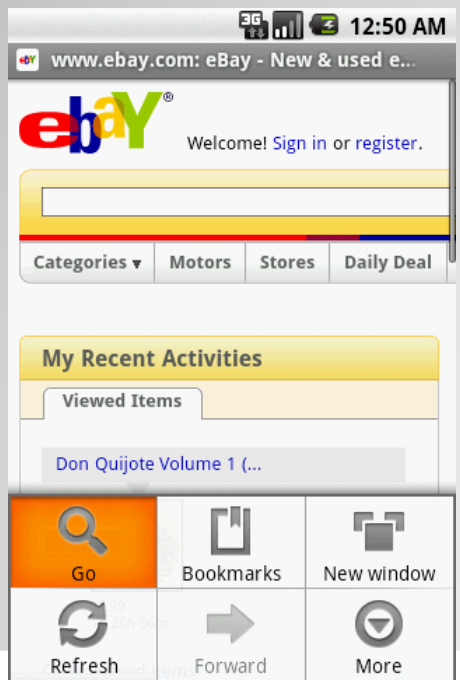
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="cis493.demoui" android:versionCode="1" android:versionName="1.0">

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

  <application android:icon="@drawable/icon"
        android:label="@string/app_name"> <activity
        android:name=".AndDemoUI" android:label="@string/app_name">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter> </activity> </application>
   <uses-sdk android:minSdkVersion="3" />
</manifest>
```

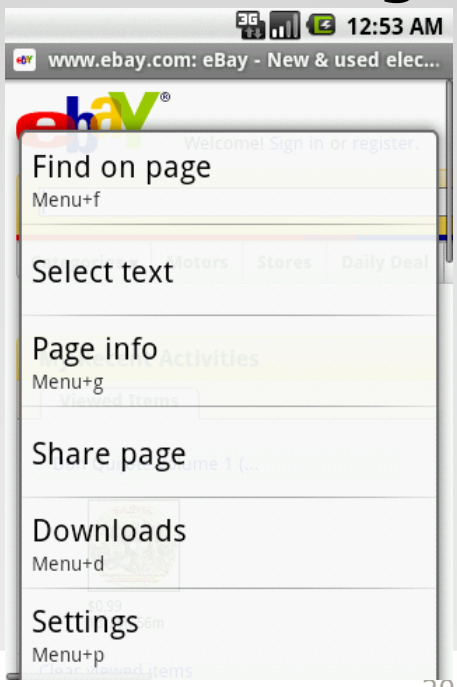
Android WebKit browser

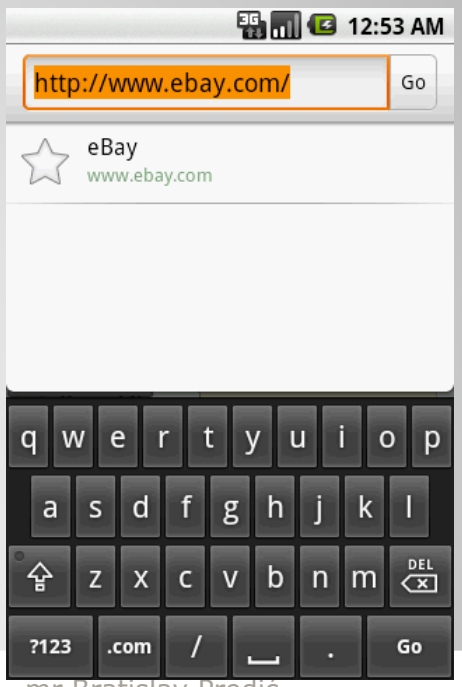
- JavaScript je podrazumevano isključen u *WebView* widget-u
- Može se uključiti sa

```
myWebView.setSettings().setJavaScriptEnabled(true);
```

- *WebView* ima standardan ugrađen meni

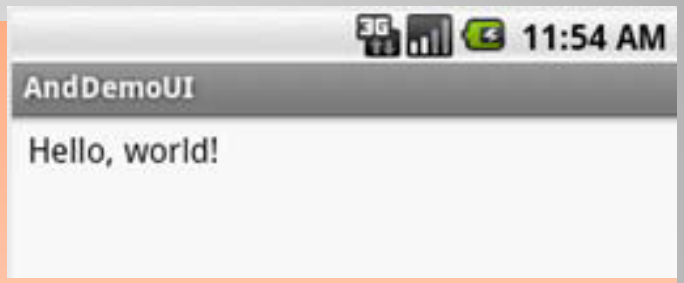






Android WebKit browser

- U *WebView* se može i direktno učitati neki HTML, nije neophodno da se preuzima sa URL adrese preko mreže
  - Npr. Situacija kada prikazujemo HTML help za aplikaciju

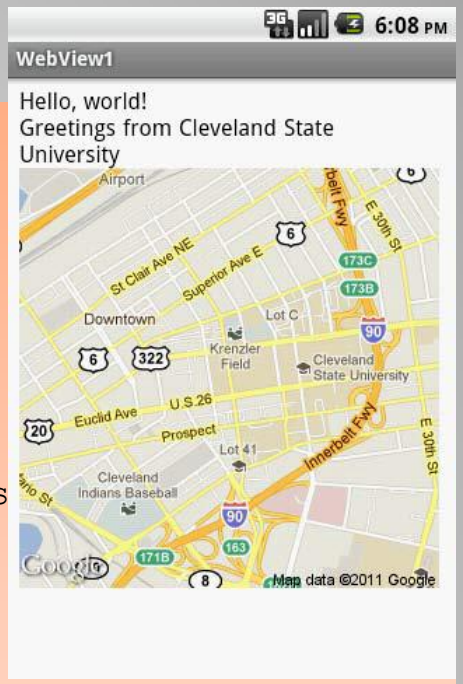



```
public class AndDemoUI extends Activity {
  WebView browser;

  @Override
  public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.main);
  browser=(WebView)findViewById(R.id.webkit);
  browser.loadData(
        "<html><body>Hello, world!</body></html>",
        "text/html",
        "UTF-8");
  }
}
```

Android WebKit browser

- Možemo i kombinovati statički i dinamički sadržaj

```
public class AndDemoUI extends Activity {
  WebView browser;

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    browser = (WebView)findViewById(R.id.webkit);
    // code your htlm in-line page (or store page in app's
    // "assets" folder)
    String aGoogleMap =
        "<img src=\"http://maps.googleapis.com/maps/api/"
        + "staticmap?center=41.5020952,"+
        "-81.6789717&zoom=14&size=300x300&sensor=false\"> ";
    String myHtmlPage = "<html> <body> Hello, world! "
        + "<br> Greetings from Cleveland State University"
        + aGoogleMap
        + " </body> </html>";
    browser.loadData(myHtmlPage, "text/html", "UTF-8");
  }
}
```



Android WebKit browser

- *WebView* widget nema navigation toolbar prikazan na ekranu
- Komande WebKit browser-a
  - *reload()* – refresh trenutno učitane stranice
  - *goBack()* – jedan korak unazad u istoriju
  - *goForward()* – jedan korak unapred
  - *goBackOrForward()* – proizvoljan broj koraka
  - *canGoBackOrForward()* – da li ima istorije
  - *clearCache()* – čisti keš
  - *clearHistory()* – briše istoriju
- Izabrani podskup ovih operacija je moguće dodati u meni aplikacije

Veza HTML+JavaScript+Android

- Moguće je ovom vezom omogućiti JavaScript kodu da upravlja Android aplikacijom (poziva metode nekog Android objekta)
- Ovo može biti jako korisna opcija za integraciju HTML interfejsa sa Android aplikacijom
- Takođe može biti jako veliki bezbednosni propust

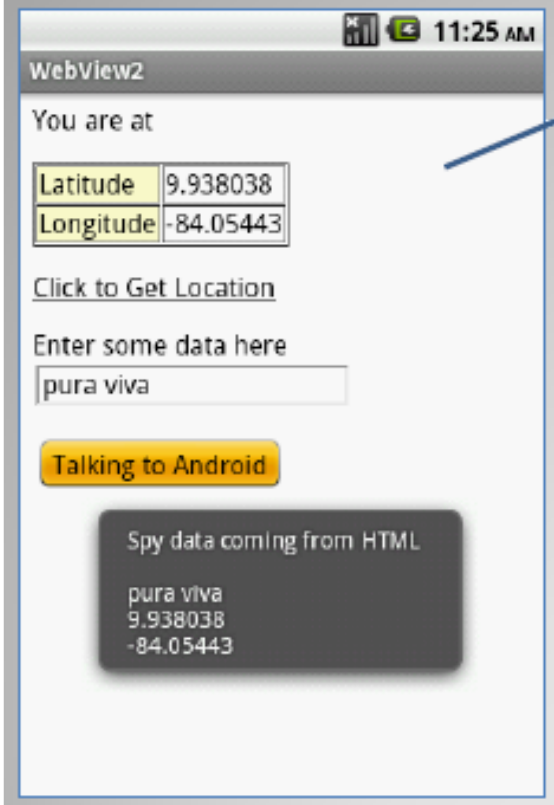
```
public void addJavascriptInterface ( Object obj, String interfaceName )
```

- Primer može biti uključivanje Google Maps API web interfejsa u Android aplikaciju

Veza HTML+JavaScript+Android

- Prednosti
  - Logika aplikacije je na serveru, nikad na klijentima (lakši update)
- Primeri
  - WebView2 – razmena objekata između JS i Android-a
  - WebView3 – Prikaz fiksne lokacije korišćenjem Google Maps API v3 (samo HTML i JS)
  - WebView4 – Slanje realne lokacije sa Android-a u JS (prikaz realne trenutne lokacije)

Veza HTML+JavaScript+Android



HTML

Android OS

locater object

MyLocater
-spy
+getLatitude() +getLongitude() +htmlPassing2Android()

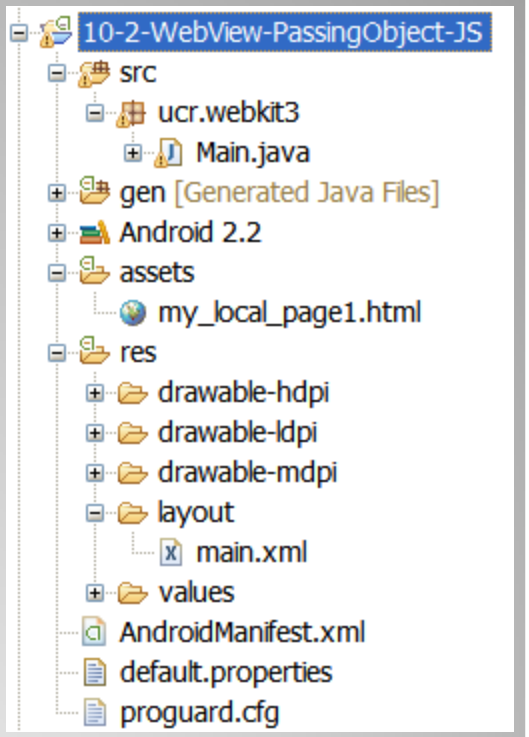
You pass an object whose **methods** you want to expose to JavaScript (*class vars not visible*)

Veza HTML+JavaScript+Android

- Primer interakcije JS+HTML i Android

Veza HTML+JavaScript+Android

- Koraci
  - *WebView* u *Layout*-u
  - HTML stranicu u *assets* folder
  - Kreirati Java objekat koji delimo sa JS

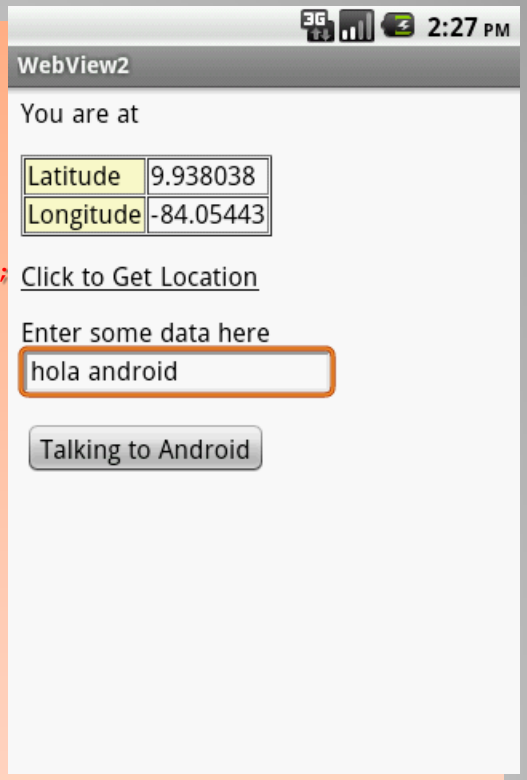



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <WebView
      android:id="@+id/webview"
      android:layout_width="fill_parent"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

Veza HTML+JavaScript+Android

- HTML stranica

```
<html>
<head>
<title>Android_Passing_HTML_JS</title> <head>
<script language="javascript">
function whereami() {
            document.getElementById("lat").innerHTML=locater.getLatitude();
            document.getElementById("lon").innerHTML=locater.getLongitude();
            document.getElementById("myText").value = locater.getCommonData();
            }
function talkBack2Android() {
            locater.setCommonData("Greetings from html");
            var spyHtml = "Spy data coming from HTML\n"
            + "\n" + document.getElementById("myText").value
            + "\n" + document.getElementById("lat").innerHTML
            + "\n" + document.getElementById("lon").innerHTML;
            locater.htmlPassing2Android(spyHtml);
            }
</script>
</head>
<body>
<p> You are at </p>
<table border="1" cellspacing="1" cellpadding="1">
  <tr>
    <td bgcolor="#FFFFCC"> Latitude </td>
    <td><span id="lat"> (unknown) </span></td>
  </tr>
  <tr>
    <td bgcolor="#FFFFCC"> Longitude </td>
    <td><span id="lon"> (unknown) </span></td>
  </tr>
</table>
<p><a onClick="whereami()"><u> Click to Get Location </u></a></p>
<p> Enter some data here <input type="text" id="myText" />
<p> <input type="button" onclick= "talkBack2Android()" value="Talking to Android">
</body>
</html>
```

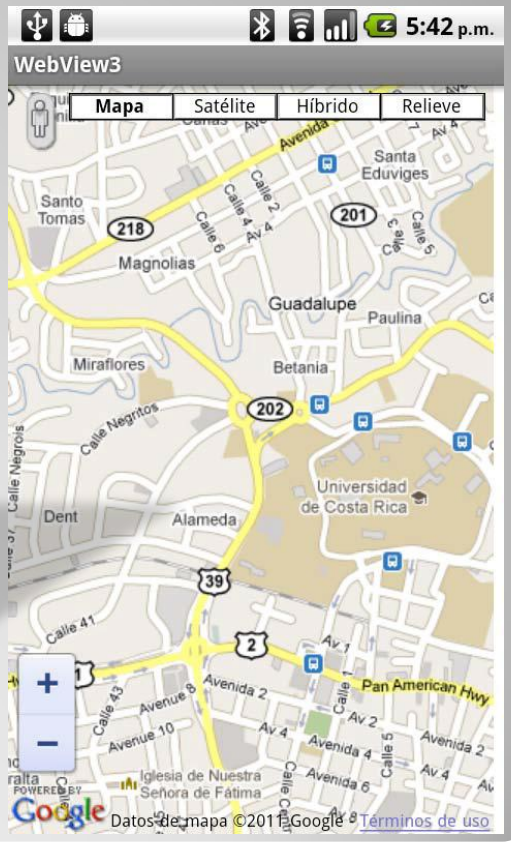


Veza HTML+JavaScript+Android

- U manifest moramo da dodamo permissions

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

```
public class Main extends Activity {
  WebView browser;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // connect browser to local html file showing map
    browser = (WebView) findViewById(R.id.webview);
    browser.getSettings().setJavaScriptEnabled(true);
    browser.loadUrl("file:///android_asset/webview_map.html");
  }
}
```



Veza HTML+JavaScript+Android

- Kombinacija obostrane komunikacije
- Koristimo Android objekat da pošaljemo realnu lokaciju
- HTML stranica sadrži JavaScript kod za prikaz mape centrirane oko poslate lokacije

Veza HTML+JavaScript+Android

- Android aplikacija

```
public class Main extends Activity implements LocationListener {
  private static final String MAP_URL =
"http://gmaps-samples.googlecode.com/svn/trunk/articles-android-webmap/simple-android-
map.html";
  private WebView browser;
  LocationManager locationManager;
  MyLocater locater = new MyLocater();

  @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 must be outside)
    //criteria.setAccuracy(Criteria.ACCURACY_COARSE); // 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: 1 sec. or 0 meter]
    locationManager.requestLocationUpdates(provider, 1, 0, this);
    locater.setNewLocation(locationManager.getLastKnownLocation(provider));
  }
```

Veza HTML+JavaScript+Android

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

/** Sets up the browser object and loads the URL of the page **/
private void setupbrowser() {
  final String centerURL = "javascript:centerAt("
                         + locater.getLatitude() + ","
                         + locater.getLongitude() + ")";
  // set up the browser to show location results
  browser = (WebView) findViewById(R.id.webview);
  browser.getSettings().setJavaScriptEnabled(true);
  browser.addJavascriptInterface(locater, "locater");
  browser.loadUrl("file:///android_asset/webview_map.html");
  // Wait for the page to load then send the location information
  browser.setWebViewClient(new WebViewClient() {
    @Override
    public void onPageFinished(WebView view, String url) {
      browser.loadUrl(centerURL);
    }
  });
}
```

Veza HTML+JavaScript+Android

```
@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();
  locater.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
}
```

Veza HTML+JavaScript+Android

```
// ////////////////////////////////////////////////////////////////
// An object of type "MyLocater" will be used to pass data back and
// forth between the Android app and the JS code behind the html page.
// ////////////////////////////////////////////////////////////////
  public class MyLocater {
    private Location mostRecentLocation;

    public void setNewLocation(Location newCoordinates){
      mostRecentLocation = newCoordinates;
    }

    public double getLatitude() {
      if (mostRecentLocation == null)
        return (0);
      else
        return mostRecentLocation.getLatitude();
    }

    public double getLongitude() {
      if (mostRecentLocation == null)
        return (0);
      else
        return mostRecentLocation.getLongitude();
    }
  }// MyLocater
}//class
```