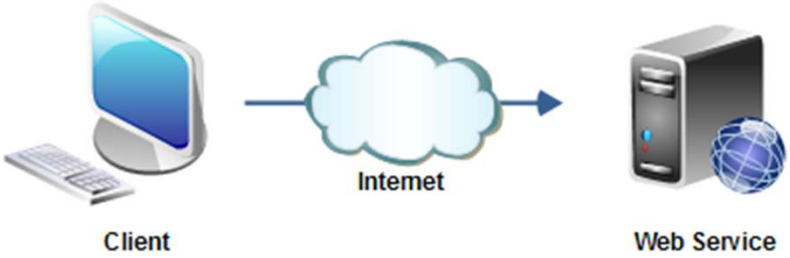
Mobile Application Development

Consuming JSON Web Services

INTRODUCTION TO WEB SERVICES

Web Services

- Think of Web service as:
  - Like calling a method, procedure or function which is running on a different machine than the client.



Why Web Services?

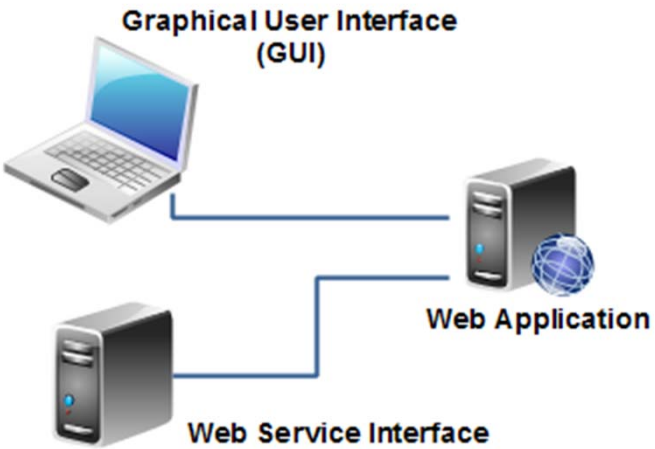
- **Reusable application-components**
  Currency conversion, weather reports, or even language translation as services.

- **Connect existing software**
  Web services can help to solve the interoperability problem by giving different applications a way to link their data. With Web services you can exchange data between different applications and different platforms.

Web Services vs. Web Sites

The main difference between a web service and a web site is, that

- a web site is typically intended for human consumption (human-to-computer interaction), whereas
- web services are typically intended for computer-to-computer interaction.

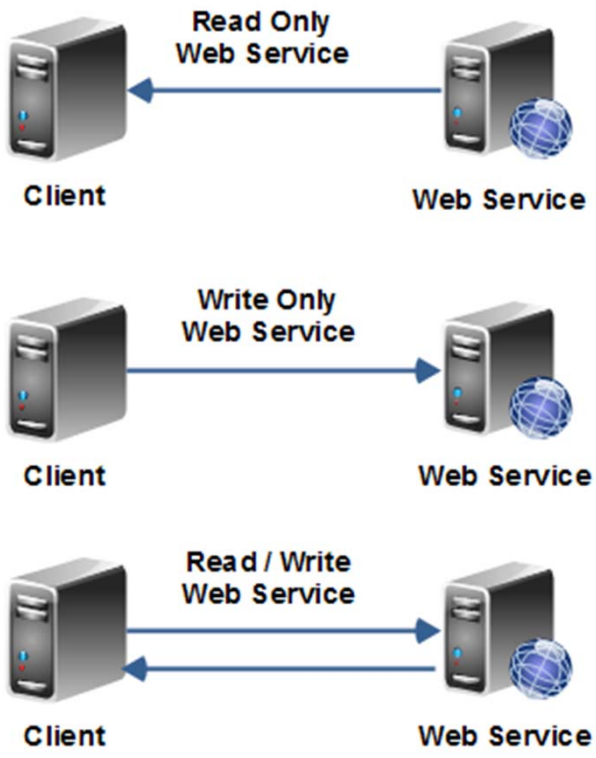


Web Services

- When a client and a web service communicate they exchange messages.
  - A request message is sent from the client to the web service.
  - The web service responds with a response message.
  - This is just like in ordinary HTTP, where a web browser sends an HTTP request to a web server, and the web server replies with an HTTP response.

Web Services

- Web Service Message Exchange Patterns.



Web Service Message Formats

- In the beginning the only web service message format available was SOAP (Simple Object Access Protocol) – an XML based format.

- Later came REST (REpresentational State Transfer) type web services, which uses plain XML, JSON (JavaScript Object Notation), plain text, or even customized formats.

REST (REpresentational State Transfer)

- How it Works?
  - Clients can issue request via HTTP methods (e.g, “GET”, “POST”, etc)
  - Server provides a response which can be representated in either XML, JSON, plain text, or even customized format
  - Status codes indicate what the outcome was.

REST + XML

http://somedomain.com/profiles/bugsbunny

```
<profile>
    <firstName>Bugs</firstName>
    <lastName>Bunny</lastName>
    <address>
        <street>The Road 123</street>
        <zip>12345</zip>
        <city>USA</city>
    </address>
</profile>
```

REST + JSON

http://somedomain.com/profiles/bugsbunny

```
{
    firstName : "Bugs",
    lastName  : "Bunny",
    address   : {
       street : "The Road 123",
       zip    : "12345",
       city   : "USA"
    }
}
```

JSON

- JSON (JavaScript Object Notation), is a lightweight text-based open standard designed for **human-readable data interchange**.
- It is derived from the JavaScript scripting language for **representing simple data structures and associative arrays, called objects**.
- Despite its relationship to JavaScript, it is **language-independent**, with parsers available for many languages.
- See: http://json.org/

Sample JSON Representation

```
{
"person":
  [
    {
      "name":{"first":"John","last":"Adams"},
      "age":"40"
    },
    {
      "name":{"first":"Thomas","last":"Jefferson"},
      "age":"35"
    }
  ]
}
```

JSON Syntax Rules

JSON syntax is a subset of the JavaScript object notation syntax.

- Data is in name/value pairs
- Data is separated by comma
- Curly brackets holds objects
- Square brackets holds arrays
- Contains basic data types: Number, String, Boolean, Object, Array, and null.

Sample JSON Representation

```
{
"person":
  [
    {
      "name":{"first":"John","last":"Adams"},
      "age":"40"
    },
    {
      "name":{"first":"Thomas","last":"Jefferson"},
      "age":"35"
    }
  ]
}
```

Web Service Providers

- Most online businesses, portals & social networks provide web services API to developers. Some free web services available to developers include:
  - http://restcountries.eu/
  - http://www.freecurrencyconverterapi.com/
  - http://openweathermap.org/API
- You can find more web service providers by searching Google or visiting sites like:
  - http://www.programmableweb.com/
  - https://www.mashape.com/

References

- http://en.wikipedia.org/wiki/Web_service
- http://tutorials.jenkov.com/web-services/index.html
- http://json.org/
- http://en.wikipedia.org/wiki/JSON

CREATING JSON WEB SERVICES IN PHP

Setting Content Type with Header

- By default, a PHP script's output is assumed to be HTML.
- Use the **header function** to specify non-HTML output - must appear before any other output generated by the script

```
header("Content-type: text/plain");
echo "This output is plain text now!\n";
```

Content ("MIME") Types

MIME type	Related File Extension
text/plain	.txt
text/html	.html, .htm
text/javascript	.js
text/xml	.xml
Application/json	.json
image/jpeg	.jpg, .jpeg
video/quicktime	.mov
application/octet-stream	.exe

Learn More: http://www.webmaster-toolkit.com/mime-types.shtml

Server: Emitting Text

```
http://localhost/www/server-multiply.php?v1=3&v2=4
```

```
header("Content-type: text/plain");
$a = $_REQUEST["v1"];
$b = $_REQUEST["v2"];
$result = $a * $b;

echo $result;
```

Client: Parsing Text

```
$url="http://localhost/www/web-
  development/webservices-rest/server-
  multiply.php?v1=3&v2=4";

$result=file_get_contents($url);

echo "Multiplication Server Answered: ".$result;
```

Reporting Errors

- Web service should return an HTTP "error code" to the browser, possibly followed by output
- User **headers** for HTTP Error Codes

```
if (($_REQUEST["v1"]==="") || ($_REQUEST["v2"]==""))
{
  header("HTTP/1/1 400 Invalid Request");
  die("An HTTP error 400 (Invalid request) occurred.");
}
```

HTTP Status Codes

Code	Meaning
200	OK
301	Moved
400	Invalid Request
403	Forbidden
404	Not Found
500	Internal Server Error

Learn More: http://en.wikipedia.org/wiki/Http_error_codes

JSON REPRESENTATION

JSON

- The official Internet media type for JSON is **application/json**. The JSON filename extension is **.json**.
- It is used primarily to **transmit data between a server and web application**, serving as an alternative to XML.

Sample JSON

```
{
"person":
  [
    {
      "name":{"first":"John","last":"Adams"},
      "age":"40"
    },
    {
      "name":{"first":"Thomas","last":"Jefferson"},
      "age":"35"
    }
  ]
}
```

JSON Syntax Rules

JSON syntax is a subset of the JavaScript object notation syntax.

- Data is in name/value pairs
- Data is separated by comma
- Curly brackets holds objects
- Square brackets holds arrays

Sample JSON

```
{
"person":
  [
    {
      "name":{"first":"John","last":"Adams"},
      "age":"40"
    },
    {
      "name":{"first":"Thomas","last":"Jefferson"},
      "age":"35"
    }
  ]
}
```

Parse JSON as an Array

```
$string='{"person":
. . .
}';

$json_a=json_decode($string,true);
foreach($json_a[person] as $p)
{
echo "Name: ".$p[name][first]." ".$p[name][last].
   "Age: ".$p[age];
}
```

Parse JSON as an Object

```
$string='{"person":
. . .
}';

$json_o=json_decode($string);
foreach($json_o->person as $p)
{
echo "Name: ".$p->name->first." ". $p->name->last.
   "Age: ".$p->age;
}
```

Server: Emitting JSON Data

```
header('Cache-Control: no-cache, must-revalidate');
header('Content-type: application/json');
$json='
{
"person":
    [
        {
            "name":{"first":"John","last":"Adams"},
            "age":"40"
        },
        {
            "name":{"first":"Thomas","last":"Jefferson"},
            "age":"35"
        }
    ]
}
';
echo $json;
```

Client: Parse JSON From URL

```
$url="http://localhost/www/univeristy/awd
  /webservices-rest/json-server.php";

$string = file_get_contents($url);

$json_o=json_decode($string);
foreach($json_o->person as $p)
{
echo "Name: ".$p->name->first." ". $p->name->last. "
  Age: ".$p->age . "<br />";
}
```

CONSUMING JSON WEB SERVICES IN ANDROID APPS

Consuming JSON Web Services

- In order to consume JSON Web Service in our Android Apps:
  - **Our app needs permission to connect to the Internet.**
    - We can specify **android.permission.INTERNET** in AndroidManifest.xml
  - **We need to be able to send and receive data over the web.**
    - Android provides **HttpURLConnection** class which can be used to send and receive data over the web.
  - **We need to be able to parse JSON data.**
    - Android provides **JSONObject** & **JSONArray** classes, enabling developers to easily parse JSON data
  - **Update UI with received information.**
    - Android provides **AsyncTask** class that allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

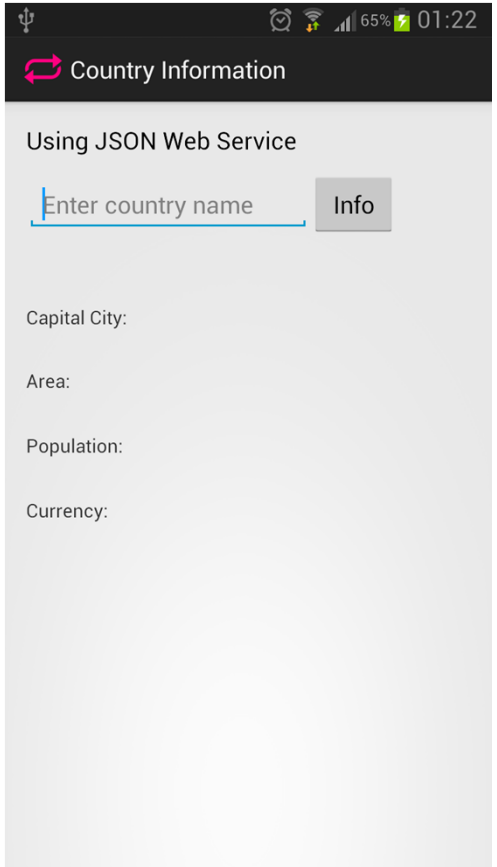
AndroidManifest.xml

Specify Permissions

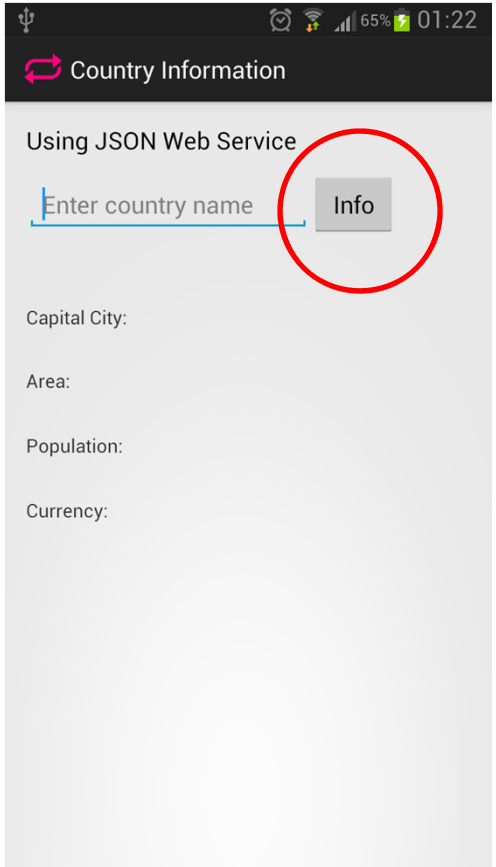
Specify the permissions your application needs, by adding <uses-permission> elements as children of the <manifest> element:

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

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

Create Activity



Activity


65%
01:22
Country Information
Using JSON Web Service
Enter country name
Info
Capital City:
Area:
Population:
Currency:

activity_main.xml

```
. . .
<Button
    android:id="@+id/btnFindInfo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/editCountryName"
    android:layout_alignBottom="@+id/editCountryName"
    android:layout_toRightOf="@+id/editCountryName"
    android:text="@string/btn_get_info"
    android:onClick="loadCountryInfo" />
. . .
```

MainActivity.java

```
. . .
public void loadCountryInfo(View view) {

    EditText cn=(EditText) findViewById(R.id.editCountryName);
    String countryName=cn.getText().toString();
    String strURL="http://restcountries.eu/rest/v1/name/"+countryName;

    if (countryName.length()==0) {
         Toast.makeText(this, "No Country Specified", Toast.LENGTH_SHORT).show();
    }
    else {
         new wsAsyncTask().execute(strURL);
    }

}
. . .
```

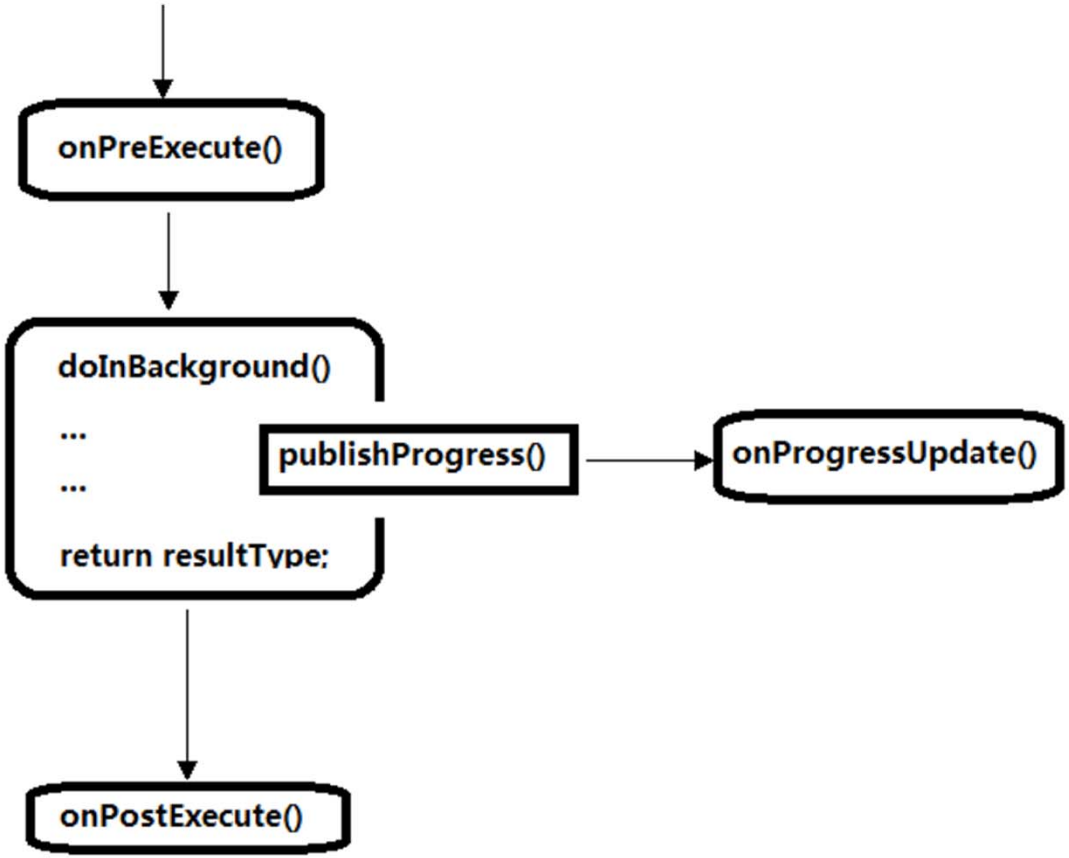
AsyncTask

- AsyncTask class allows to
  - perform background operations and
  - publish results on the UI thread without having to manipulate threads and/or handlers.

AsyncTask

- An asynchronous task is defined by
  - 3 generic types, called Params, Progress and Result, and
  - 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.

AsyncTask


onPreExecute()
doInBackground()
...
...
return resultType;
publishProgress()
onProgressUpdate()
onPostExecute()

AsyncTask

- AsyncTask must be **subclassed** to be used.
  - The subclass will override at least one method (doInBackground(Params...)), and
  - most often will override a second one (onPostExecute(Result)).

MainActivity.java

```
. . .
public void loadCountryInfo(View view) {

    EditText cn=(EditText) findViewById(R.id.editCountryName);
    String countryName=cn.getText().toString();
    String strURL="http://restcountries.eu/rest/v1/name/"+countryName;

    if (countryName.length()==0) {
         Toast.makeText(this, "No Country Specified", Toast.LENGTH_SHORT).show();
    }
    else {
         new wsAsyncTask().execute(strURL);
    }

}
. . .
```

wsAsyncTask()

```
. . .
private class wsAsyncTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... strURL) {
        return requestWebService(strURL[0]);
    }

    @Override
    protected void onPostExecute(String result) {
        . . .
    }

}
. . .
```

requestWebService()

```
public static String requestWebService(String serviceUrl) {
   HttpURLConnection urlConnection = null;
   try {

         . . .

   } catch (MalformedURLException e) {
       e.printStackTrace(); // URL is invalid
   } catch (SocketTimeoutException e) {
       e.printStackTrace(); // data retrieval or connection timed out
   } catch (IOException e) {
       e.printStackTrace(); // could not read response body
       // (could not create input stream)
   } finally {
       if (urlConnection != null) {
           urlConnection.disconnect();
       }
   }
   return null;
}
```

requestWebService()

```
HttpURLConnection urlConnection = null;
  try {
      // create connection
      URL urlToRequest = new URL(serviceUrl);
      urlConnection = (HttpURLConnection) urlToRequest.openConnection();
      urlConnection.setConnectTimeout(15000);
      urlConnection.setReadTimeout(10000);

      // get JSON data
      InputStream in = new BufferedInputStream(urlConnection.getInputStream());

      // converting InputStream into String
      Scanner scanner = new Scanner(in);
      String strJSON = scanner.useDelimiter("\\A").next();
      scanner.close();
      return strJSON;

  }
```

JSON Format

http://restcountries.eu/rest/v1/name/Pakistan

```
[
  {
      . . .
      "capital": "Islamabad",
      . . .
      "population": 184845000,
      . . .
      "area": 881912,
      . . .
      "currencies": [
            "PKR"
      ],
      . . .
  }
]
```

Array

Object

String

Number

Number

Array

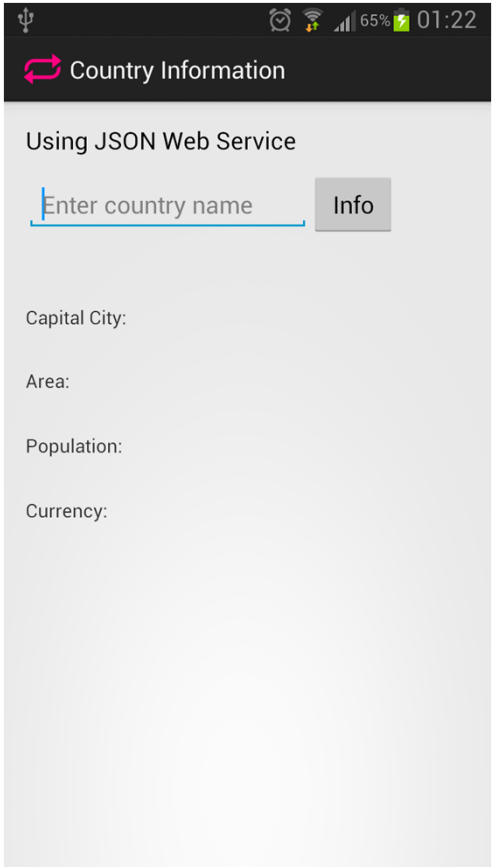
wsAsyncTask()

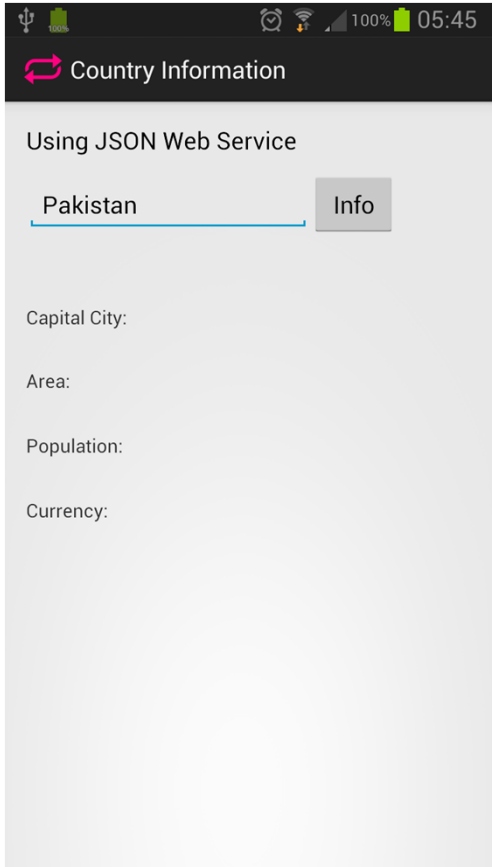
```
. . .
private class wsAsyncTask extends AsyncTask<String, Void, String> {

   @Override
   protected String doInBackground(String... strURL) {
         return requestWebService(strURL[0]);
   }

   @Override
   protected void onPostExecute(String result) {
         . . .
   }

}
. . .
```

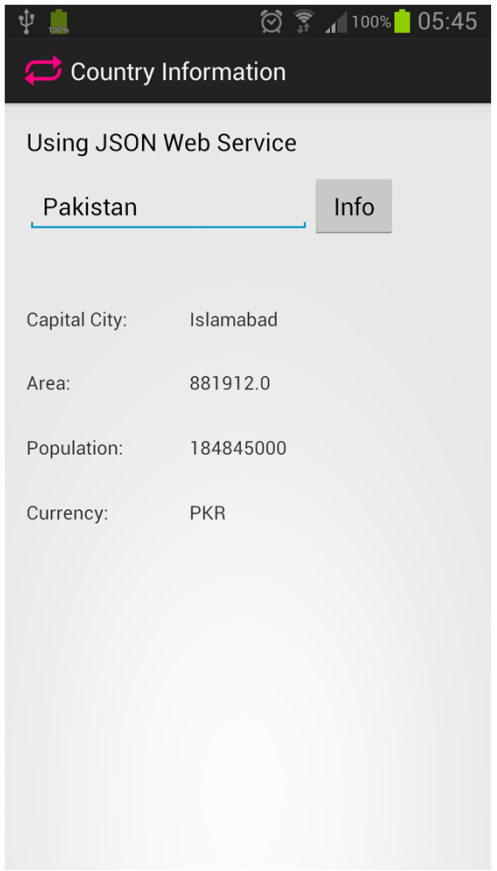
“onPostExecute()” in wsAsyncTask()

```
protected void onPostExecute(String result) {
   txtCapital=(TextView) findViewById(R.id.textCapitalValue);
   txtArea=(TextView) findViewById(R.id.textAreaValue);
   txtPopulation=(TextView) findViewById(R.id.textPopulationValue);
   txtCurrency=(TextView) findViewById(R.id.textCurrencyValue);

   try {
        JSONArray rootArray=new JSONArray(result);
        JSONObject rootObject=rootArray.getJSONObject(0);

        txtCapital.setText(rootObject.optString("capital"));
        txtArea.setText(rootObject.optString("area"));
        txtPopulation.setText(rootObject.optString("population"));
        txtCurrency.setText(rootObject.optJSONArray("currencies").getString(0));
   } catch (JSONException e) {
        e.printStackTrace();
   }
}
```

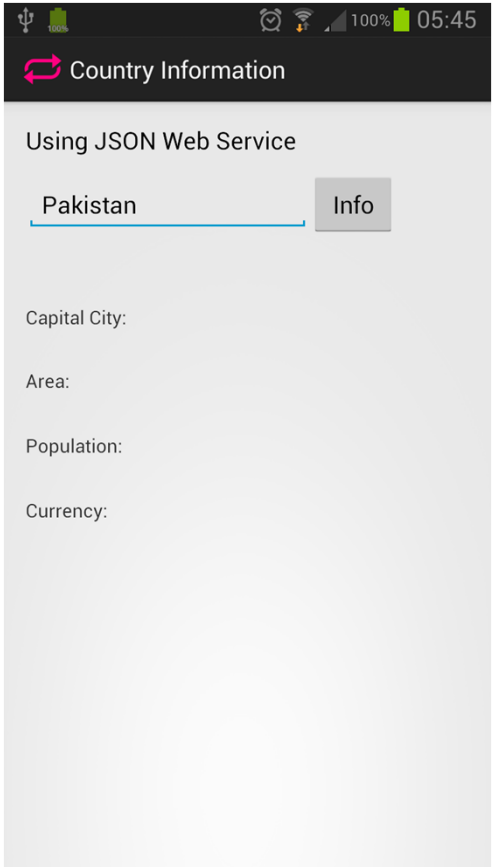
Using JSON Web Service

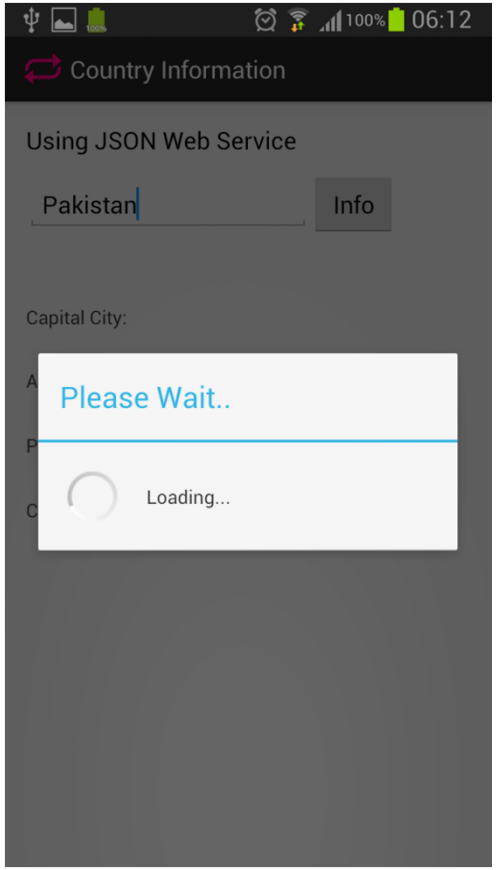


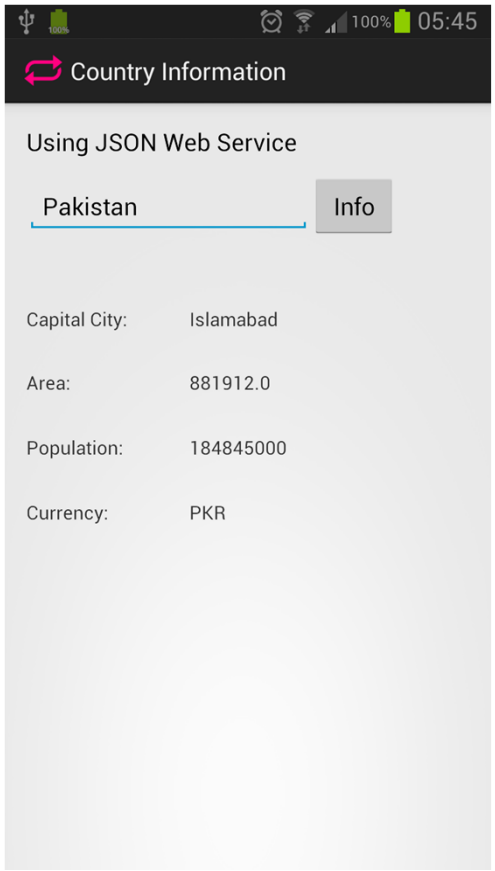




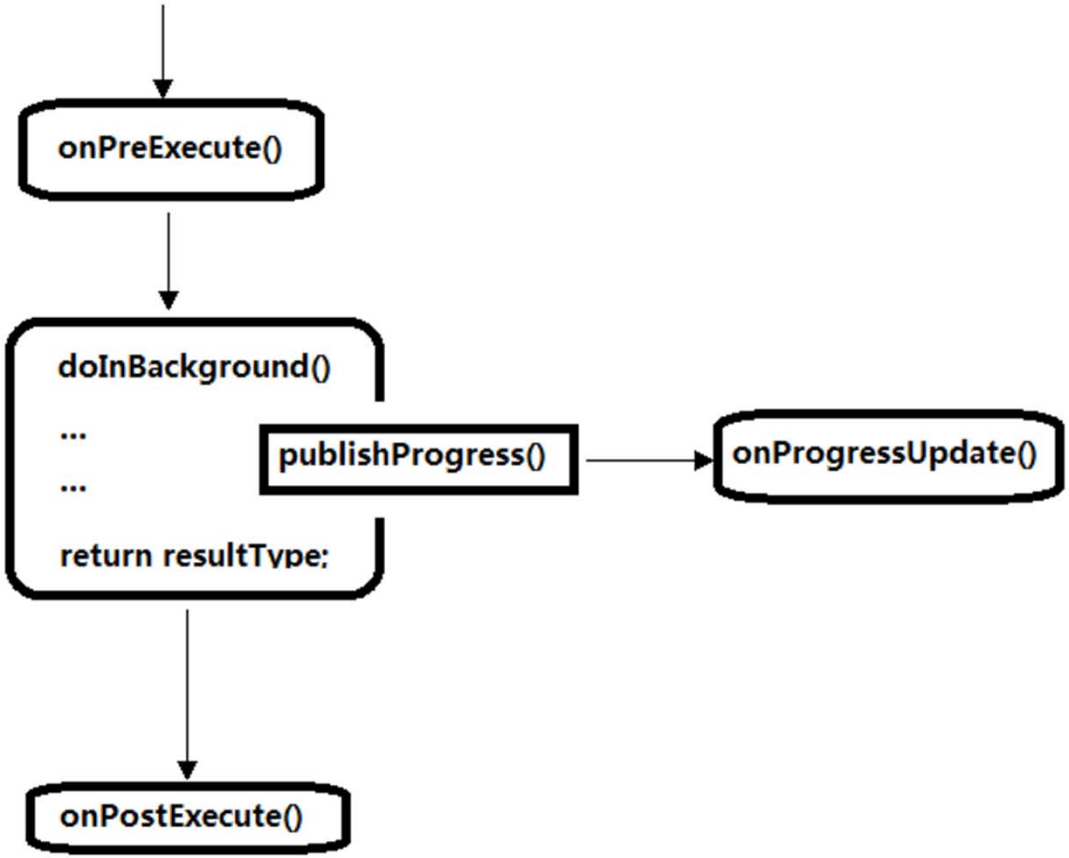
Adding Progress Dialog







AsyncTask


onPreExecute()
doInBackground()
...
...
return resultType;
publishProgress()
onProgressUpdate()
onPostExecute()

MainActivity.java

```
public class MainActivity extends Activity {
    private TextView txtCapital;
    private TextView txtArea;
    private TextView txtPopulation;
    private TextView txtCurrency;

    ProgressDialog dialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);

    }
. . .
```

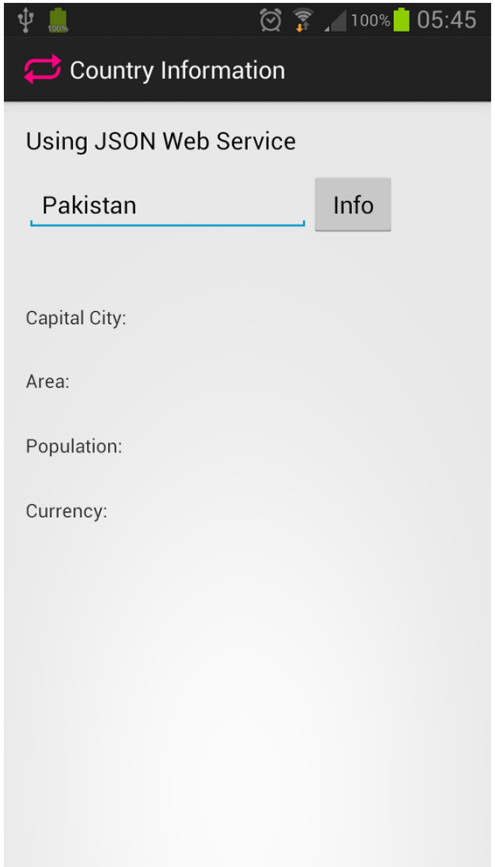
Add “onPreExecute()” in wsAsyncTask()

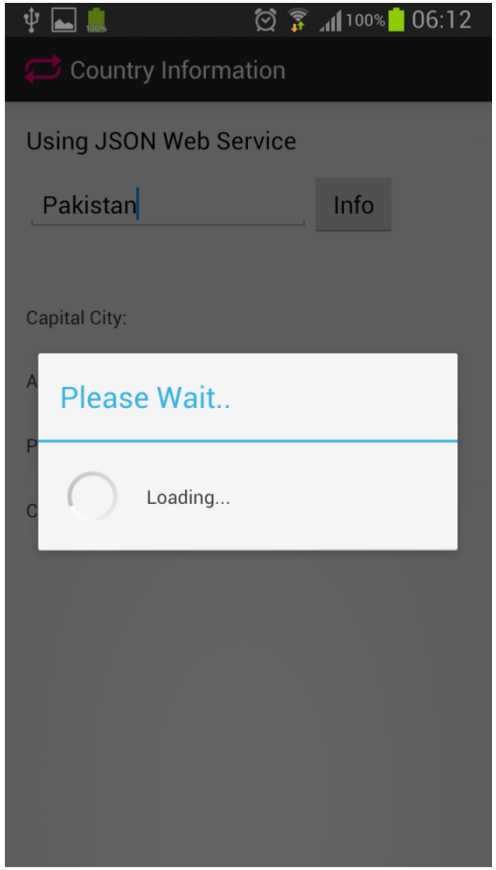
```
. . .
private class wsAsyncTask extends AsyncTask<String, Void, String> {

    . . .

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        dialog = new ProgressDialog(MainActivity.this);
        dialog.setTitle("Please Wait..");
        dialog.setMessage("Loading...");
        dialog.setCancelable(false);
        dialog.show();
    }
}
. . .
```

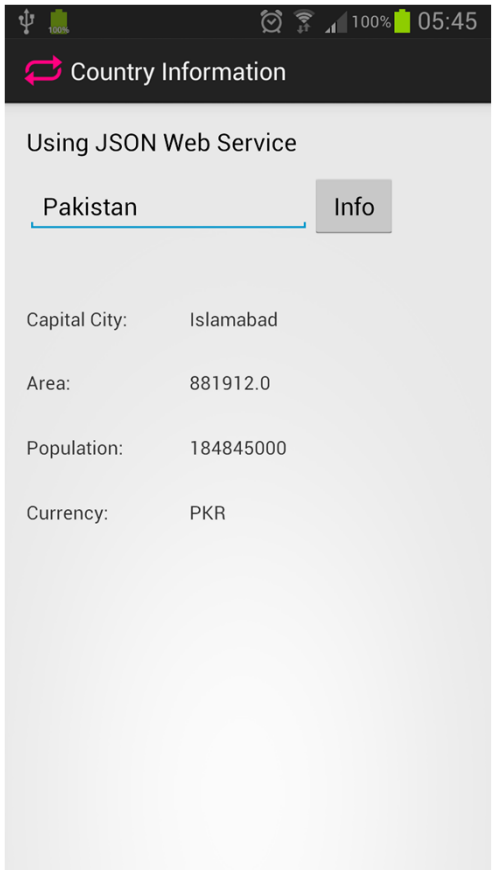
Dismiss Dialog in “onPostExecute()”

```
protected void onPostExecute(String result) {
   txtCapital=(TextView) findViewById(R.id.textCapitalValue);
   txtArea=(TextView) findViewById(R.id.textAreaValue);
   txtPopulation=(TextView) findViewById(R.id.textPopulationValue);
   txtCurrency=(TextView) findViewById(R.id.textCurrencyValue);

   try {
        JSONArray rootArray=new JSONArray(result);
        JSONObject rootObject=rootArray.getJSONObject(0);

        txtCapital.setText(rootObject.optString("capital"));
        txtArea.setText(rootObject.optString("area"));
        txtPopulation.setText(rootObject.optString("population"));
        txtCurrency.setText(rootObject.optJSONArray("currencies").getString(0));
   } catch (JSONException e) {
        e.printStackTrace();
   }
   dialog.dismiss();
}
```

Using JSON Web Service







Consuming JSON Web Services

- In order to consume JSON Web Service in our Android Apps:
  - **Our app needs permission to connect to the Internet.**
    - We can specify **android.permission.INTERNET** in AndroidManifest.xml
  - **We need to be able to send and receive data over the web.**
    - Android provides **HttpURLConnection** class which can be used to send and receive data over the web.
  - **We need to be able to parse JSON data.**
    - Android provides **JSONObject** & **JSONArray** classes, enabling developers to easily parse JSON data
  - **Update UI with received information.**
    - Android provides **AsyncTask** class that allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

HttpURLConnection

- **HttpURLConnection** is used to send and receive data over the web. Data may be of any type and length.
  - Obtain a new HttpURLConnection by calling **URL.openConnection()**.
  - Prepare the request.
    - Optionally upload a request body. Instances must be configured with **setDoOutput(true)** if they include a request body. Transmit data by writing to the stream returned by **getOutputStream()**.
  - Read the response – header & body. The response body may be read from the stream returned by **getInputStream()**. If the response has no body, that method returns an empty stream.
  - Disconnect. Once the response body has been read, the HttpURLConnection should be closed by calling **disconnect()**.

Sending Data With HTTP Request

```
public static String requestWebService(String serviceUrl) {
   HttpURLConnection urlConnection = null;
   try {

         . . .

   } catch (MalformedURLException e) {
       e.printStackTrace(); // URL is invalid
   } catch (SocketTimeoutException e) {
       e.printStackTrace(); // data retrieval or connection timed out
   } catch (IOException e) {
       e.printStackTrace(); // could not read response body
       // (could not create input stream)
   } finally {
       if (urlConnection != null) {
           urlConnection.disconnect();
       }
   }
   return null;
}
```

Sending Data With HTTP Request

```
HttpURLConnection urlConnection = null;
  try {
      // create connection
      URL urlToRequest = new URL(serviceUrl);
      urlConnection = (HttpURLConnection) urlToRequest.openConnection();
      urlConnection.setConnectTimeout(15000);
      urlConnection.setReadTimeout(10000);

      // get JSON data
      InputStream in = new BufferedInputStream(urlConnection.getInputStream());

      // converting InputStream into String
      Scanner scanner = new Scanner(in);
      String strJSON = scanner.useDelimiter("\\A").next();
      scanner.close();
      return strJSON;

  }
```

Sending Data With HTTP Request

```
HttpURLConnection urlConnection = null;
  try {
        . . .

        urlConnection.setRequestMethod("POST");
        urlConnection.setRequestProperty("Content-Type",
                "application/x-www-form-urlencoded");
        urlConnection.setUseCaches (false);

        Uri.Builder builder = new Uri.Builder()
                        .appendQueryParameter("x", "9")
                        .appendQueryParameter("y", "7");
        String data = builder.build().getEncodedQuery();
        byte[] outputInBytes = data.getBytes("UTF-8");
        urlConnection.setRequestProperty("Content-Length", "" +
                          Integer.toString(outputInBytes.length));
        urlConnection.setDoOutput(true);
        OutputStream os = urlConnection.getOutputStream();
        os.write(outputInBytes);
        os.close();


        . . .
  }
```

Consuming Web Services on Localhost

- Each instance of the emulator runs behind a virtual router/firewall service that isolates it from your development machine's network interfaces and settings and from the internet.
  - An emulated device can not see your development machine or other emulator instances on the network.
  - Use **http://10.0.2.2/** instead of http://localhost/ or http://127.0.0.1/

References

- http://developer.android.com/reference/java/net/HttpURLConnection.html
- http://developer.android.com/tools/devices/emulator.html#emulatornetworking
- https://weblogs.java.net/blog/pat/archive/2004/10/stupid_scanner.html
- http://developer.android.com/reference/org/json/JSONObject.html
- http://developer.android.com/reference/org/json/JSONArray.html
- http://developer.android.com/reference/android/os/AsyncTask.html
- http://developer.android.com/training/basics/network-ops/connecting.html
- http://developer.android.com/reference/android/app/ProgressDialog.html

Q & A