Intro to Android Development 3

Accessibility Capstone

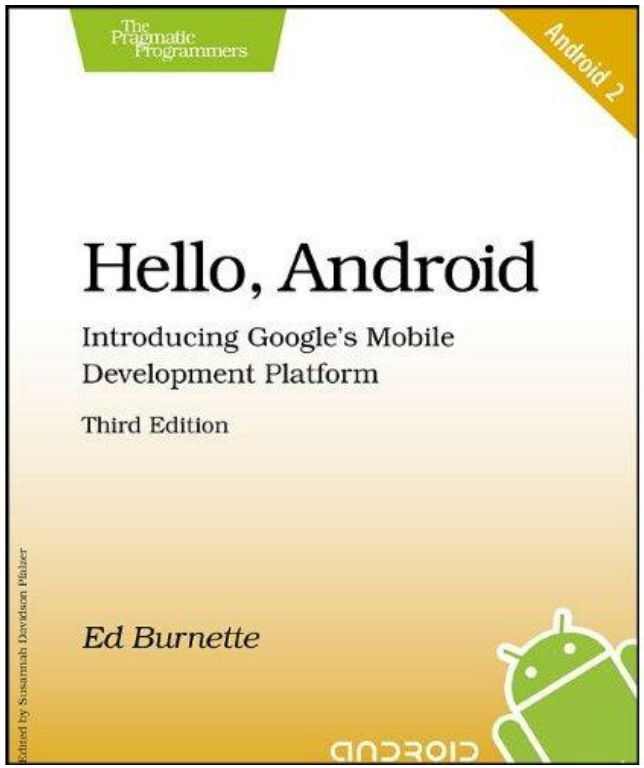
Nov 29, 2011

Outline

- Web Services
- XML vs. JSON
- Your Own Web Service

Hello, Android

The key to the capstone… (sort of)



Using Web Services

HTTP Request

HTTP Response

Using Web Services

Using Web Services

Why use a web service?

An HTTP Request

http://www.google.com

An HTTP Request

http://www.google.com

http://www.geonames.org/

An HTTP Request

http://www.google.com

http://www.geonames.org/

http://ws.geonames.org/wikipediaSearch?q=london&maxRows=10

An HTTP Request

http://www.google.com

http://www.geonames.org/

http://ws.geonames.org/wikipediaSearch?q=london&maxRows=10

Adding parameters:
?param1=val1&param2=val2

An HTTP Response

- Response code
  - 2xx = success! ☺
  - 4xx = client error
  - 5xx = server error

An HTTP Response

- Response code

  2xx = success! ☺

  4xx = client error

  5xx = server error

- Response body (XML, JSON)

<geonames>
<entry>
<lang>en</lang>
<title>London</title>
<summary>
London (; and largest urban area of England and the United Kingdom. At its core, the ancient City of London, to which the name historically belongs, still retains its limited mediaeval boundaries; but since at least the 19th century the name "London" has also referred to the whole metropolis which has developed around

XML and JSON

- Parsers
  - XML: Different traversals include SAX, DOM, etc.
  - JSON: Look at JSONObject/JSONArray classes
- Overviews
  - http://www.w3schools.com/json/
  - http://www.w3schools.com/xml/
- Recommend using JSON: small, easy to parse, fast

XML

```
<?xml version="1.0"?>
<note>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>
```

JSON

```
{
  "employees": [
    { "firstname":"John" , "lastname":"Doe" },
    { "firstname":"Anna" , "lastname":"Smith" },
    { "firstname":"Peter" , “lastname":"Jones" }
  ]
}
```

JSON

- Name/Value Pairs: “name” : “value”
  - "firstname":"John"
- JSON Objects: { … }
  - { "firstname":"John" , "lastname":"Doe" }
- JSON Arrays: […, …, …]

```
[
  { "firstname":"John" , "lastname":"Doe" },
  { "firstname":"Anna" , "lastname":"Smith" },
  …
]
```

How to Use a Webservice

- Find a webservice or write your own

  example: www.geonames.org
- Construct the URL

  add parameters, extra headers (if needed
- Execute the request asynchronously
- Check the response code

  200 = success, 4xx = client error, 5xx = server error
- Parse the response body

Construct the URL

final static String *PLACE_NAME = "Seattle";*

final static String *URL = “http://ws.geonames.org/” +*
“wikipediaSearch?q=" + PLACE_NAME + "&maxRows=4";

Execute the Request

```
public String callWebService(){
	HttpClient httpclient = new DefaultHttpClient();
	HttpGet request = new HttpGet(URL);
	String result = "";

	ResponseHandler<String> handler =
				new BasicResponseHandler();
	try {
		result = httpclient.execute(request, handler);
	} catch (ClientProtocolException e) {
		e.printStackTrace();
	} catch (IOException e) {
		e.printStackTrace();
	}
	httpclient.getConnectionManager().shutdown();

	return result;
}
```

Execute the Request

```
public String callWebService(){
    HttpClient httpclient = new DefaultHttpClient();
    HttpGet request = new HttpGet(URL);
    String result = "";

    ResponseHandler<String> handler =
                        new BasicResponseHandler();
    try {
        result = httpclient.execute(request, handler);
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    httpclient.getConnectionManager().shutdown();

    return result;
}
```

the client will open a connection and executes the request

Execute the Request

```
public String callWebService(){
        HttpClient httpclient = new DefaultHttpClient();
        HttpGet request = new HttpGet(URL);
        String result = "";

        ResponseHandler<String> handler =
                                new BasicResponseHandler();
        try {
                result = httpclient.execute(request, handler);
        } catch (ClientProtocolException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
        httpclient.getConnectionManager().shutdown();

        return result;
}
```

request object contains URL

Execute the Request

```
public String callWebService(){
        ...HttpClient();
        ...RL);
```

this kind of response handler will return the response body as a string if the request was successful

```
        ResponseHandler<String> handler =
                                new BasicResponseHandler();
        try {
                result = httpclient.execute(request, handler);
        } catch (ClientProtocolException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
        httpclient.getConnectionManager().shutdown();

        return result;
}
```

Execute the Request

```
public String callWebService(){
        HttpClient httpclient = new DefaultHttpClient();
        HttpGet request = new HttpGet(URL);
        String result = "";

        ResponseHandler<String> handler =
                                new BasicResponseHandler();
        try {
                result = httpclient.execute(request, handler);
        } catch (ClientProtocolException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
        httpclient.getConnectionManager().shutdown();

        return result;
}
```

executes the request and processes the response using the handler

Execute the Request

```
public String callWebService(){
        HttpClient httpclient = new DefaultHttpClient();
        HttpGet request = new HttpGet(URL);
        String result = "";

        ResponseHandler<String> handler =
                                    sicResponseHandler();
        try {
                result = httpclient.execute(request, handler);
        } catch (ClientProtocolException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
        httpclient.getConnectionManager().shutdown();

        return result;
}
```

generated by the handler

Calling on *callWebService()* and Displaying the Results

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

        TextView tv = (TextView) findViewById(R.id.resultTV);
        String result = callWebService();
        tv.setText(result);
}
```

What’s wrong with the example?

What's wrong with the example?

- result = httpclient.execute(request, handler); **blocks and waits for entire page to download!**
- Solution: threads
  - You *may* have to use another thread in your application!
  - java.util.concurrent package
  - Good example in Hello, Android (web services chapter)
  - Keeping it all simple is key, then program away

Your Own Web Service

- May need if…
  - You need additional processing power
  - You want to tie phone app into a web product
  - Etc…
- If you create your own web service…
  - Server Side Scripting (PHP, Ruby on Rails, etc.)
  - Web server (abstract.cs, etc.)

Your Own Web Service

- Very Basic PHP Overview:
  1. Create .php page
     (http://foo.com/service.php)
  2. Use $_GET[‘param’]
     (http://foo.com/service.php?param=helloworld
  3. Do stuff with the parameters
     ($bar = $_GET[‘param’])
  4. Format output into XML or JSON or whatever you need
     (echo $bar)
- Phone app will visit php page and get its result

Other Topics

- GPS and Sensors
  - LocationManager
- SQL Database
  - SQLiteDatabase
- Camera
- Multitouch

Exercise

Develop an application that uses a web service that is accessible to a blind person.

For extra brownie points: use the GPS!

-Use TTS to make the application accessible.

-Find a web service (you can start from Geonames)

-Find and use a Java XML parser to parse the response body (you can try SAX)

Note: many web services use JSON, not XML.