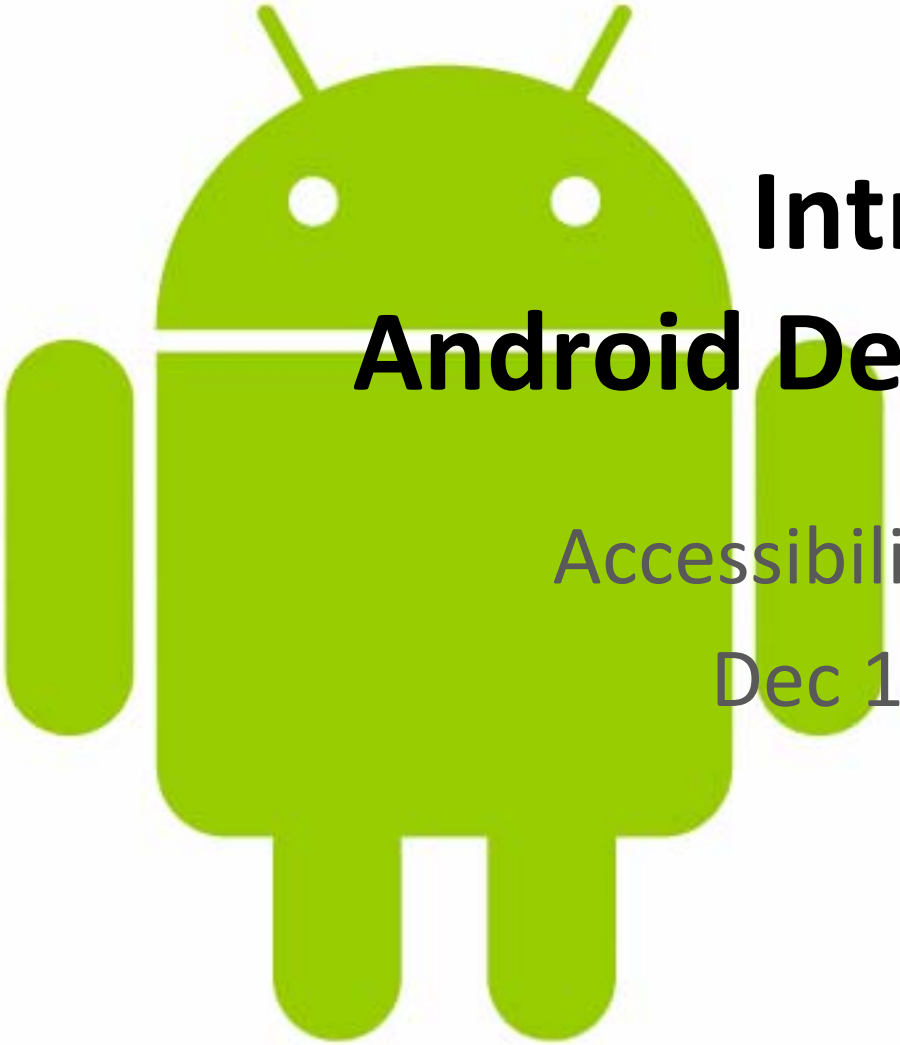
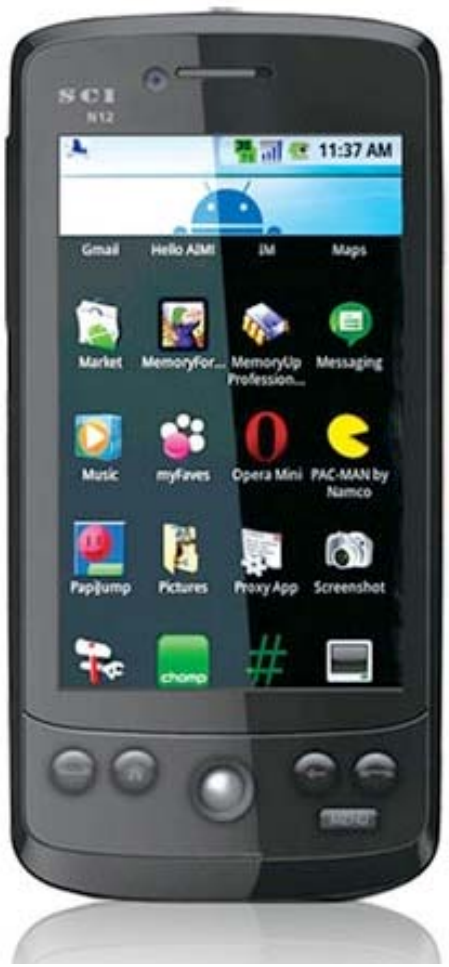


Intro to Android Development 3

Accessibility Capstone

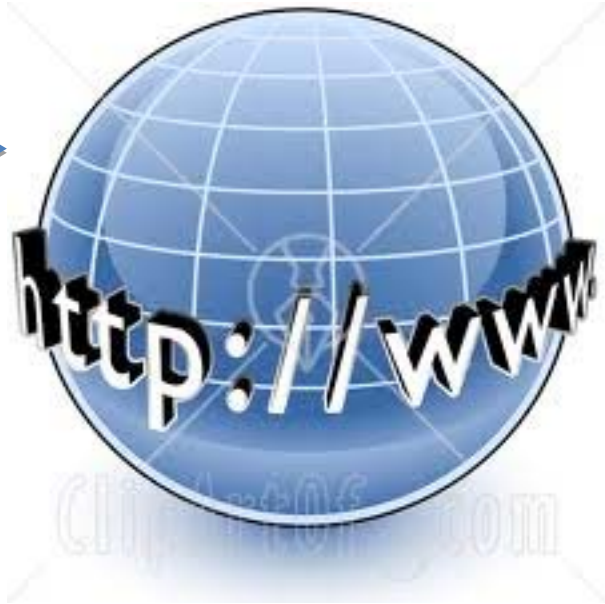
Dec 10, 2010

Using Web Services

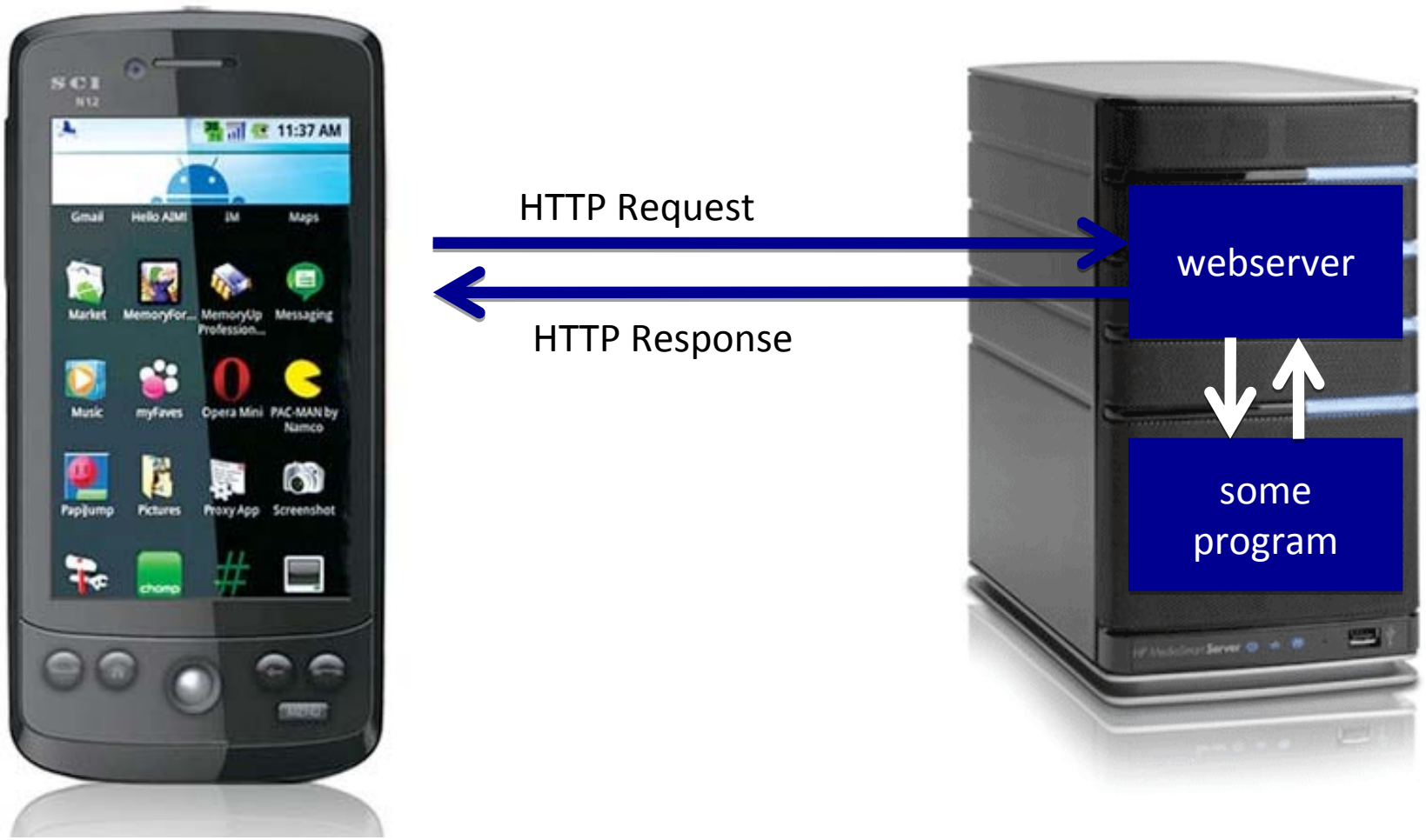


HTTP Request

HTTP Response



Using Web Services



Why use a web service?

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

- 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

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

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

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

        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?

1. We don’t parse the response.
2. We execute the request synchronously (should be asynchronous).
3. We don’t handle errors.

Assignment

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.