Android
V – Non Java: Web Services & JNI

Stefan BORNHOFEN
EISTI

Web Services

- Method of communication between applications over the World Wide Web
- Exchange of structured data
- Written in various programming languages and running on various platforms
- Web services combine the best aspects of component-based development and the Web
- Complex web services: SOAP
- Lightweight web services: REST

SOAP

- Designed for complex distributed computing environments
- Uses XML for the message format
- Independent of the transport protocol (HTTP, FTP, TCP, UDP, …)
- WSDL service description to describe how the web service works: messages, bindings, operations and location.
- You can discover the service automatically and generate a client proxy from that service description
- *Verbose, lots of overhead*
- *Hard to develop, requires tools*
- *more heavy-weight than REST*

REST

- Point-to-point communication over HTTP
- Invoked by a simple URL
- Human readable results (XML, JSON, plain text…)
- Easy to build, no toolkits required
- Completely stateless (interaction can survive a restart of the server)
- Primary option for mobile applications, no SOAP runtime necessary.
- *No standards: the service producer and service consumer need to have a common understanding of the context*

```
HTTP    CRUD Equivalent
====    ================
GET     read
POST    create
PUT     create, update
DELETE  delete
```

Example: SOAP vs REST

Querying a phonebook application for the details of a given user with ID=12345.

Using SOAP, the request would look something like this:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body pb="http://www.mysite.com/phonebook">
    <pb:GetUserDetails>
      <pb:UserID>12345</pb:UserID>
    </pb:GetUserDetails>
  </soap:Body>
</soap:Envelope>
```

The XML will be sent to the server using an HTTP POST request.
The result is typically an XML file embedded inside a SOAP response envelope.

With REST, the query will typically look like this:
`http://www.mysite.com/phonebook/UserDetails/12345`
This URL is sent to the server using an HTTP GET request.
The HTTP reply is the raw result data, not embedded inside anything, in a way you can directly use

Android REST Call

```
class HTTPTask extends AsyncTask<String, Void, String> {
  @Override
  protected String doInBackground(String... uri) {
    try {
       HttpURLConnection httpsURLConnection = (HttpURLConnection) new URL(url[0]).openConnection();
       httpsURLConnection.setConnectTimeout(2000);
       httpsURLConnection.setRequestMethod("GET");
       httpsURLConnection.connect();
       int mStatus = httpsURLConnection.getResponseCode();
       if (mStatus == 200 || mStatus == 201)
          return readResponse(httpsURLConnection.getInputStream()).toString();
       } catch (IOException E) {}
       return null;
   }
   private StringBuilder readResponse(InputStream inputStream) throws IOException, NullPointerException {
       BufferedReader r = new BufferedReader(new InputStreamReader(inputStream));
       StringBuilder stringBuilder = new StringBuilder();
       String line;
       while ((line = r.readLine()) != null) stringBuilder.append(line);
       return stringBuilder;
   }
   @Override
   protected void onPostExecute(String result) {
       super.onPostExecute(result);
       // do something with the response string
   }
}
```

new HTTPTask().execute("http://my.restful.web.service/parameters");

An asynchronous task is a computation that runs on a background thread and whose result is published on the UI thread.

Set permissions

- Don’t forget to add internet permissions to the AndroidManifest.xml

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

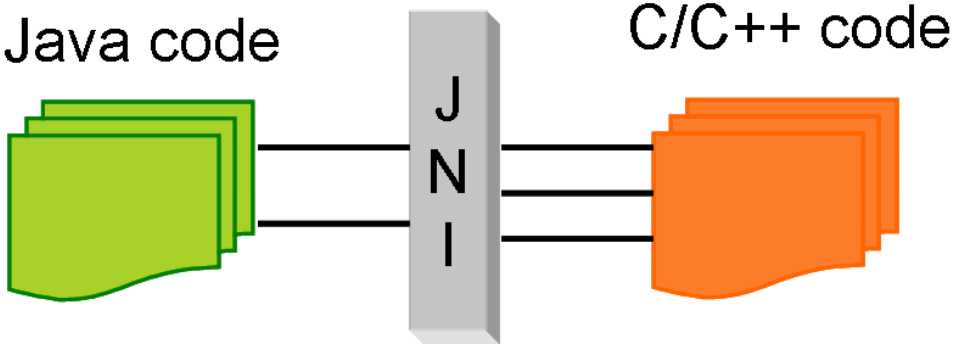
JNI

Programming framework that enables Java applications to call native code (typically C or C++)

- Native code runs faster than JVM code
- You might have some old legacy code written in C/C++ and you don't want to waste your time porting the code to Java

Caution

- An application that relies on JNI loses the platform portability
- The JNI framework does not provide any automatic garbage collection for non-JVM memory resources
- Good candidates for JNI are self-contained, CPU-intensive operations that don't allocate much memory



Android NDK

- Native Development Kit
- Toolset that helps implementing parts of your app using native-code languages

Install NDK, CMake and LLDB
using the Android Studio SDK manager

Android NDK: howto I

- New Project => Include C++ support

```
public class MainActivity extends AppCompatActivity {

    static {
        System.loadLibrary("native-lib");
    }

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

        ... some code
    }

    public native String stringFromJNI();
    public native int hello(int n, int m);
}
```

Android NDK: howto II

- Native functions are implemented in *.c or *.cpp files.
- When the JVM invokes a native function, it passes
  1. *a JNIEnv pointer*: contains the interface to the JVM. It includes all of the functions necessary to interact with the JVM and to work with Java objects (converting native arrays to/from Java arrays, instantiating objects, throwing exceptions, etc.)
  2. *a jobject*: a reference to the Java object inside which this native method has been declared. You can use it to address variables or call other methods of the Java object from the current JNI function.
  3. the Java arguments declared by the Java method.

Android NDK: howto III

Add the native method to your cpp file.

```
#include <jni.h>

extern "C" {

      JNIEXPORT jint JNICALL
      Java_fr_eisti_android_tp5_MainActivity_hello
        (JNIEnv *env, jobject instance, jint n, jint m)
      {
          return n + m;
      }

}
```

Synchronize native-lib.cpp and run your app.

Exercise

Compute Fibonacci by three different methods (2p).

- Android Java
- JNI
- RESTful Web service:
  http://i3.options.eisti.fr/webservices/fibonacci/37

- For Java and JNI, use the naïve Fibonacci algorithm: `fibo(n)=fibo(n-1)+fibo(n-2)`
- Clock the time to see if JNI works faster than Java code

Go back to the Currency Converter and fetch the exchange rates from an online JSON API (2p).