UA . MASTER MOVILES

MÁSTER UNIVERSITARIO EN DESARROLLO DE SOFTWARE PARA DISPOSITIVOS MÓVILES

PROGRAMACIÓN HIPERMEDIA PARA DISPOSITIVOS MÓVILES

Android – Acceso a servicios Rest

CONTENIDOS

- Acceso a servicios REST
- Códigos de respuesta
- Cabeceras
- Peticiones tipo GET, PUT, POST y DELETE
- Parsing de XML
- Parsing de JSON

ACCESO A SERVICIOS REST

- Para acceder a un servicio tipo REST también vamos a usar la librería `HttpURLConnection`.
- Esta librería nos dará acceso a los códigos de respuesta y cabeceras, además de poder realizar otro tipo de peticiones.
- Para activar las peticiones que envían datos usaremos "`setDoOutput(true)`".
- Para indicar el método de la petición usaremos: `setRequestMethod`
- **Importante**: todas las peticiones siguen teniendo que estar dentro de un hilo.

OBTENER EL CÓDIGO DE RESPUESTA

- Para obtener el código de estado enviado en la cabecera usaremos el método "`getResponseCode`".
- En caso de que la petición sea correcta nos devolverá el código 200.
- En caso de error se nos devolverá un código distinto a 200.
- Para comprobar que la petición es correcta haremos:

```
if( http.getResponseCode() == HttpURLConnection.HTTP_OK ) {
    // Correcto! Ya podemos descargar el cotenido!
}
```

- La lista completa de códigos la podemos ver en https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

ESTABLECER LAS CABECERAS

- Para establecer las cabeceras usaremos el método "`setRequestProperty`"

```
http.setRequestProperty("User-Agent", "...");
http.setRequestProperty("Accept-Charset", "UTF-8");
http.setRequestProperty("Content-Type",
                                "text/plain; charset=utf-8");

// Para indicar que el formato de los datos es JSON...
http.setRequestProperty("Content-Type", "application/json");
http.setRequestProperty("Accept", "application/json");

// Para indicar que vamos a usar XML pondríamos...
http.setRequestProperty("Content-type", "application/xml");
http.setRequestProperty("Accept", "application/xml");
```

ACCESO A LAS CABECERAS DE RESPUESTA

- Para acceder a las cabeceras de respuesta podemos usar:

```
// Para obtener todas las cabeceras
for(Map.Entry<String, List<String>> k :
                    http.getHeaderFields().entrySet()) {
    for (String v : k.getValue()){
        Log.d("Headers", k.getKey() + ":" + v);
    }
}

// Para consultar una cabecera específica...

String type = http.getHeaderField("Content-type");
```

PETICIÓN GET

```
public String peticionGET( String strUrl ) {
    HttpURLConnection http = null;
    String content = null;
    try {
        URL url = new URL( strUrl );
        http = (HttpURLConnection)url.openConnection();
        http.setRequestProperty("Content-Type", "application/json");
        http.setRequestProperty("Accept", "application/json");

        if( http.getResponseCode() == HttpURLConnection.HTTP_OK ) {
            StringBuilder sb = new StringBuilder();
            BufferedReader reader = new BufferedReader(
                    new InputStreamReader( http.getInputStream() ));
            String line;
            while ((line = reader.readLine()) != null) {
                sb.append(line);
            }
            content = sb.toString();
            reader.close();
        }
    }
    catch(Exception e) {  e.printStackTrace();  }
    finally {    if( http != null ) http.disconnect();   }
    return content;
}
```

Contenido de la respuesta leído desde un *inputStream*

PETICIÓN POST

```
public int peticionPOST( String strUrl, String data ) {
    HttpURLConnection http = null;
    int responseCode = -1;
    try {
        URL url = new URL( strUrl );
        http = (HttpURLConnection) url.openConnection();
        http.setRequestMethod("POST");
        http.setRequestProperty("Content-Type", "application/json");
        http.setRequestProperty("Accept", "application/json");
        http.setDoOutput(true);

        PrintWriter writer = new PrintWriter(http.getOutputStream());
        writer.print(data);
        writer.flush();

        responseCode = http.getResponseCode();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (http != null) http.disconnect();
    }
    return responseCode;
}
```

Para enviar datos usamos un **outpusStream**

Además del código de respuesta también podemos leer el contenido igual que en las peticiones GET

PETICIÓN POST

- En las peticiones tipo POST podemos enviar distintos tipos de formatos.
- La función anterior recibe un String que envía en el cuerpo de la petición. Pero también podemos añadir un JSON o enviar más parámetros a la vez:

```
Uri.Builder builder = new Uri.Builder()
        .appendQueryParameter("firstParam", paramValue1)
        .appendQueryParameter("secondParam", paramValue2)
        .appendQueryParameter("thirdParam", paramValue3);
String query = builder.build().getEncodedQuery();

PrintWriter writer = new PrintWriter(http.getOutputStream());
writer.print( query );
writer.flush();
```

PETICIÓN PUT

Muy similares a las peticiones POST

```
public int peticionPUT( String strUrl, String data ) {
    HttpURLConnection http = null;
    int responseCode = -1;
    try {
        URL url = new URL( strUrl );
        http = (HttpURLConnection) url.openConnection();
        http.setRequestMethod("PUT");
        http.setRequestProperty("Content-Type", "application/json");
        http.setRequestProperty("Accept", "application/json");
        http.setDoOutput(true);

        PrintWriter writer = new PrintWriter(http.getOutputStream());
        writer.print(data);
        writer.flush();

        responseCode = http.getResponseCode();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (http != null) http.disconnect();
    }
    return responseCode;
}
```

PETICIÓN DELETE

```
public int peticionDELETE( String strUrl )
{
    HttpURLConnection http = null;
    int responseCode = -1;
    try {
        URL url = new URL( strUrl );
        http = (HttpURLConnection) url.openConnection();

        http.setRequestMethod("DELETE");
        http.setRequestProperty("Content-Type", "application/json");
        http.setRequestProperty("Accept", "application/json");

        // Conectar y obtener el codigo de respuesta
        responseCode = http.getResponseCode();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (http != null) http.disconnect();
    }
    return responseCode;
}
```

PARSING DE ESTRUCTURAS XML

- Trocear el XML en tags, atributos y contenido por medio de librerías.
- En Android tenemos:
  - **SAXParser**: Requiere implementar manejadores o *callbacks* que reaccionan a eventos al encontrar etiquetas o atributos.
  - **XmlPullParser**: Permite iterar sobre el árbol de forma secuencial. Puede ir leyendo de un *buffer* de entrada remoto.

EJEMPLO DE XML

- Vamos a considerar al siguiente ejemplo de documento XML

```
<mensajes>
    <mensaje usuario="pepe">Hola, ¿qué tal?</mensaje>
    <mensaje usuario="ana">Muy bien</mensaje>
</mensajes>
```

- Tenemos una lista de mensajes entre las etiquetas `mensajes`.
- Para cada mensaje tenemos una etiqueta `mensaje`.
- El usuario emisor se indica como atributo de la etiqueta.
- El texto del mensaje se indica como cuerpo entre la etiqueta de apertura y de cierre.

USO DEL XMLPULLPARSER

Inicializamos el parser a partir del flujo de entrada que lee de la URL

```
XmlPullParserFactory parserCreator = XmlPullParserFactory.newInstance();
XmlPullParser parser = parserCreator.newPullParser();
parser.setInput(urlInputStream, null);

int parserEvent = parser.getEventType();
while (parserEvent != XmlPullParser.END_DOCUMENT)
{
    switch (parserEvent) {
       case XmlPullParser.START_DOCUMENT:  break;
       case XmlPullParser.END_DOCUMENT:    break;
       case XmlPullParser.END_TAG:         break;
       case XmlPullParser.START_TAG:
         if(parser.getName()=="mensaje") {
           String usuario = parser.getAttributeValue(null, "usuario");
           String mensaje = parser.nextText();
         }
         break;
    }
    parserEvent = parser.next();
}
```

ESTRUCTURAS JSON

- JSON es una representación muy utilizada para formatear los recursos solicitados a un servicio web RESTful.
- Los elementos están contenidos entre llaves.
- Los valores de los elementos se organizan en pares con la estructura "`nombre:valor`" y separados por comas.
- Las secuencias de elementos están contenidas entre corchetes.
- Ejemplo:

```
[
   {"texto":"Hola, ¿qué tal?", "usuario":"Pepe" },
   {"texto":"Muy bien", "usuario":"Ana" }
]
```

PARSING DE JSON EN ANDROID

- Podemos utilizar los objetos `JSONArray` y `JSONObject`
- Pueden encontrarse anidados. Por ejemplo:

```
JSONArray mensajes = new JSONArray(contenido);
for(int i=0; i<mensajes.length(); i++)
{
    JSONObject mensaje = mensajes.getJSONObject(i);
    String texto = mensaje.getString("texto");
    String usuario = mensaje.getString("usuario");
    //...
}
```

- Lanzará una excepción en caso de error.
- También podemos componer mensajes JSON:
  - Establecer atributos con métodos `put-` (equivalentes a los `get-`)
  - Obtenemos el texto JSON llamando a `.toString()`
- Alternativas más sencillas de utilizar: *GSON, Jackson*

¿PREGUNTAS?