2015

PROGRAMACIÓN DE
DISPOSITIVOS MÓVILES

Vicente Cantón Paterna

[TUTORIAL WEBVIEW-JSOUP]

En este documento se expone un pequeño tutorial sobre el uso de webview y parseo HTML mediante JSOUP a través de una aplicación de ejemplo.

INTRODUCCIÓN

A menudo es posible que se necesite obtener información de una determinada página web. Esta información puede ser en forma de texto, imágenes, referencias…por este motivo es muy útil poder recoger dicha información en HTML y llevar a cabo el "parseo" a texto legible por el usuario.

JSOUP es una librería desarrollada con este objetivo. Dispone de métodos para obtener el texto en HTML de una página web y poder posteriormente seleccionar los elementos que se desee. Además es muy fácil de usar.

Por otro lado, y con el fin de exponer un ejemplo de uso de JSOUP, en este tutorial se va a hacer uso de WebView para la visualización de páginas web en aplicaciones Android.

El funcionamiento de la aplicación de ejemplo es simple: en el primer layout se ejecuta un webview con un buscador ya cargado y pulsando el botón "obtener referencias" se obtiene a través de JSOUP todas las referencias que existan en la página que se muestre en el webview.

WEBVIEW

Para poder utilizar un WebView es necesario incluirlo en el layout:

```
<WebView android:id="@+id/webView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"</WebView>
```

Se crea la referencia en la actividad principal:

```
webView = (WebView)findViewById(R.id.webView);
```

Un aspecto importante a tener en cuenta para el uso de WebView es que a menudo es posible que, estando navegando dentro del mismo, se quiera acceder a otra página a través de un link. WebView por defecto está configurado para que al pulsar sobre el enlace se solicite al usuario abrir un navegador para su visualización. Para evitar esto (a menudo molesto) es necesario indicarle al WebView que haga uso de WebClient y haga uso de JavaScript de manera que sea capaz de código JavaScript dentro del propio WebView:

```
webView.getSettings().setJavaScriptEnabled(true);
```

Asimismo hay que crear la interfaz y añadirla al WebView:

```
webView.addJavascriptInterface(new
MyJavaScriptInterface(this.getBaseContext()), "HtmlViewer");
```

La sentencia anterior le indica al WebView que debe utilizar la siguiente interfaz, identificada por el WebView como HtmlViewer :

```
private class MyJavaScriptInterface{

    private Context ctx;

    MyJavaScriptInterface(Context ctx) {
        this.ctx = ctx;
    }

    @JavascriptInterface
    public void showHTML(String html) {
        doc  = Jsoup.parse(html);  //La llamada a parse se encarga de
incluir el string a tipo document

    }
}
```

El cliente Web que se evita que se abra el navegador por defecto del móvil:

```
webView.setWebViewClient(new WebViewClient(){

    //Se obtiene el codigo HTML una vez cargada la pagina
    @Override
    public void onPageFinished(WebView view, String url) {
        webView.loadUrl("javascript:window.HtmlViewer.showHTML" +

"('<html>'+document.getElementsByTagName('html')[0].innerHTML+'</html>');
");
    }
});
```

El método sobreescrito onPageFinished establece que, una vez cargada la página, se haga uso de la interfaz de javascript creada anteriormente . Como se indica arriba, el método showHTML asigna a un objeto de tipo Document todo el código HTML devuelto por el WebView. Obtener dicho código es tan fácil como llamar al método *parse* de Jsoup.

JSOUP

Para poder utilizar JSOUP es necesario añadir “compile ‘org.jsoup:jsou:1.7.3’ a las dependencias Gradle del proyecto en Android Studio:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.0.0'
    compile 'org.jsoup:jsoup:1.7.3'
}
```

Una vez hecho esto ya se puede importar la librería de JSOUP y Document.

En esta aplicación de ejemplo se ha obtenido el html desde el WebView. Sin embargo, desde JSOUP existe un método mucho más fácil mediante el que, dada una URL, obtiene su código HTML:

Document doc = Jsoup.connect(URL).get()

En donde URL es un String con la dirección de la página web de la que obtener el html. Con esta única línea de código ya se tiene todo el código HTML cargado dentro del objeto Document. Ahora es sumamente fácil obtener las partes del documento que nos hagan falta a través de los métodos de la clase Document:

```
Elements links = doc.select("a");
```

Con la línea anterior se crea un objeto Elements que contiene todos los elementos del document con el tag “a”, como por ejemplo los que contienen los links [href]. También podrían seleccionarse otros tags como “div” o “img”.

Una vez obtenidos los Elements se puede acceder a los atributos sin más que llamar al método attr(string):

```
ref = new String[links.size()];
for (int i = 0; i < links.size();i++){
    ref[i] = links.get(i).attr("href").toString();

}
```

De esta forma tan sencilla se tiene un vector de String con todos los vínculos que contiene la página. Si se quiere obtener alguna otra información no se tiene más que jugar con el método *select, attr* y el resto de métodos disponibles en JSOUP.

A modo de ejemplo se incluye parte del código utilizado en otra aplicación:

```
Elements metaElements = doc.select("table");
metaElements = metaElements.select("tr");
```

```
for(Element mElement: metaElements){
    if(mElement.select("th").text().contains("Autor")){
        parsedElements = mElement.select("td");
        autor = parsedElements.first().text().toString();
        int index = autor.indexOf("[");
        autor = autor.substring(0,index);
    }else if(mElement.select("th").text().contains("Fecha")){
        parsedElements = mElement.select("td");
        fecha = parsedElements.first().text().toString();
    }else if(mElement.select("th").text().contains("Publicacion")){
        parsedElements = mElement.select("td");
        editorial = parsedElements.first().text().toString();
    }
}
```

IMPORTANTE: siempre que se trata con temas relacionados con el parseo de código html cabe recordar que es totalmente dependiente de la forma en la que se estructura el mismo y por ello es muy posible que el cambio en la página web implique un fallo en la forma de realizar el parseo por parte de la aplicación.

El código completo de la aplicación de ejemplo puede descargarse aquí:

https://github.com/VicenteCantonPaterna/EjemploParseo