Objetos JSON en Android

Por Juan Vicente Ordás

Introducción

- JSON (JavaScript Object Notation) es un formato para el intercambio de datos.
- Tiene una sintaxis dedicada.
- Nació como una alternativa al XML.
- En la actualidad están sustituyendo a los XML, un ejemplo es Twitter, cuyo modelo de datos está basado en JSON.
- JSON es más eficiente que XML (el volúmen de datos es menor en JSON).

¿Qué son los objetos JSON?

Creación de un proyecto que utilice JSON

¿Qué son los objetos JSONArray?

Bibliografía

¡Gracias por vuestra atención!

Objetos JSON en Android

Por Juan Vicente Ordás

Introducción

- JSON (JavaScript Object Notation) es un formato para el intercambio de datos.
- Tiene una sintaxis dedicada.
- Nació como una alternativa al XML.
- En la actualidad están sustituyendo a los XML, un ejemplo es Twitter, cuyo modelo de datos está basado en JSON.
- JSON es más eficiente que XML (el volúmen de datos es menor en JSON).

¿Qué son los objetos JSON?

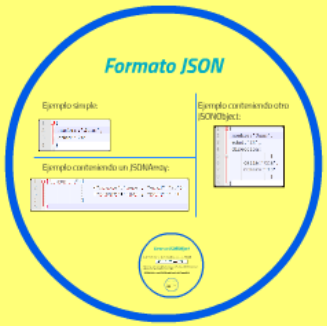
La plataforma Android incluyó las bibliotecas json.org para poder trabajar de manera eficiente con datos JSON. La clase más importante es JSONObject, la cual nos permitirá manejar datos JSON.

*Estos objetos funcionan de manera parecida a los Hashmap, ya que la estructura de sus datos está compuesta por: "clave", valor. La clave es una cadena única que nunca puede ser null.

Valores de un JSONObject

Los tipos de valores que podemos almacenar en un JSONObject son los siguientes:

- Un número (int, float, double / Integer, Float, Double)
- Un String (entre comillas simples)
- Un booleano (true o false)
- Un objeto (entre llaves {})
  - Un JSONObject (entre llaves {})
  - Un JSONArray (entre corchetes [])
- Null



Formato JSON

Ejemplo simple:

```
{
 nombre:'Juan',
 edad:'24'
}
```

Ejemplo conteniendo un JSONArray:

```
{"personas":[
            {"nombre":"pepe", "edad":"20"},
            {"nombre":"juan", "edad":"3"}
           ]
}
```

Ejemplo conteniendo otro JSONObject:

```
{
 nombre:'Juan',
 edad:'24',
 direccion:
           {
  calle:'una',
  numero:'10'
           }
}
```

Crear un JSONObject

La forma de crear un objeto vacío de este tipo es la habitual:

```
//Creación del objeto JSONObject vacío.
JSONObject json = new JSONObject();
```

Hay otra forma de crear objetos de este tipo. El siguiente constructor los crea con datos a partir de un String en formato JSON:

JSONObject json = new JSONObject(String cadenaEnFormatoJSON);

Métodos de JSONObject

Añadir elementos

Como funcionan de manera parecida a los HashMap, la forma de añadir elementos es similar (método ***put("clave", valor)***; sin embargo, estas sentencias para añadir elementos tienen que ir bajo un bloque try-catch, ya que puede producir una excepción JSONException:

```
//Métodos para añadir elementos al objeto JSONObject.
try {
    json.put("nombre", "pepe");
    json.put("edad", 20);
} catch (JSONException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
```

Leer elementos

1. Con el método ***getXXX(String clave)*** conseguiremos cualquier valor que queramos que tenga esa clave. Además, esta sentencia puede producir la excepción JSONException, por lo que tiene que ir en un bloque try/catch. Sin embargo, este método tiene un problema, y es que si la clave no coincide con ningún elemento **no devuelve nada.**

2. Con el método ***optXXX(String clave)*** conseguiremos el valor asociado a la clave que pasamos como parámetro. Si no existe devuelve los siguientes valores dependiendo del caso:

-String: una cadena en blanco.
-boolean: false.
-int, long: 0.
-objeto: null.

Añadir elementos

Como funcionan de manera parecida a los HashMap, la forma de añadir elementos es similar (método ***put("clave", valor)***; sin embargo, estas sentencias para añadir elementos tienen que ir bajo un bloque try-catch, ya que puede producir una excepción JSONException:

```
//Métodos para añadir elementos al objeto JSONObject.
try {
    json.put("nombre", "pepe");
    json.put("edad", 20);
} catch (JSONException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
```

Leer elementos

1. Con el método ***getXXX(String clave)*** conseguiremos cualquier valor que queramos que tenga esa clave. Además, esta sentencia puede producir la excepción JSONException, por lo que tiene que ir en un bloque try/catch. Sin embargo, este método tiene un problema, y es que si la clave no coincide con ningún elemento **no devuelve nada.**

2. Con el método ***optXXX(String clave)*** conseguiremos el valor asociado a la clave que pasamos como parámetro. Si no existe devuelve los siguientes valores dependiendo del caso:

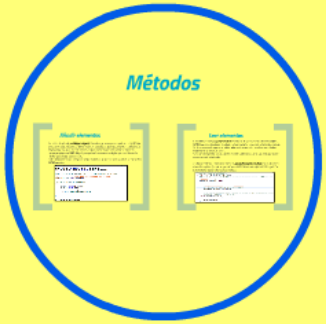
-String: una cadena en blanco.
-boolean: false.
-int, long: 0.
-objeto: null.

¿Qué son los objetos JSONArray?

Los objetos de esta clase se comportan de manera parecida a los ArrayList<> que conocemos pero orientados a la funcionalidad JSON.

```
final JSONArray array = new JSONArray();
```

Este tipo de objetos pueden contener los mismos tipos de valores que un JSONObject.



Métodos

Añadir elementos

Se utiliza el método ***put(Object objeto)***. Sin embargo, lo mejor es construir un JSONObject antes e introducir los pares "clave", valor en este objeto, para posteriormente añadir ese objeto al JSONArray. De esta manera, si queremos recuperar los valores, podremos recuperar primero el JSONObject y después el valor de este objeto que coincida con la "clave" pasada como parámetro.
Éste método tiene que ir bajo un bloque try/catch, ya que se puede producir la excepción JSONException.

```
// Variables auxiliar para cargar la información que necesitamos
String nombre = ed_nombre.getText().toString();
int edad = Integer.parseInt(ed_edad.getText().toString());

//Creación de un objeto JSONObject vacío para insertar los valores
//en el JSONArray y luego poder identificarlos por su clave.
JSONObject json = new JSONObject();

//Añadir elementos al JSONObject.
try {
    json.put("nombre", nombre);
    json.put("edad", edad);
} catch (JSONException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
//JSONArray.put(json.toString());
```

Leer elementos

1. Se utiliza el método ***get(int indice)***. En el caso de que hayamos añadido objetos JSONObject, nos devolverá un objeto correspondiente con el índice indicado, si existe.
En el caso de que hayamos añadido valores directamente, nos devolverá el valor asociado a ese índice, si existe.
Esta sentencia produce la excepción JSONException en caso de que no haya ningún valor en el índice indicado.

2. Otro método que podemos utilizar es ***getJSONObject(int índice)***. Eso sí, tenemos que estar seguros de que lo que contiene el JSONArray son objetos JSONObject. De esta manera, nos ahorramos hacer el cast.

```
//Bucle para recorrer el JSONArray
for(int i = 0; i < array.length(); i++){
    //Creación de una variable JSONObject para poder guardar cada objeto JSONObject que
    //contiene el array jsonarray.
    JSONObject jsonArray = null;

    try {
        //Recogemos cada objeto JSONObject que devuelve
        //el método en la variable creada anteriormente.
        jsonArray = array.getJSONObject(i);

        //Otra manera de guardar en una variable los JSONObject que contiene el JSONArray
        jsonArray = (JSONObject) array.get(i);
    } catch (JSONException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

Añadir elementos

Se utiliza el método ***put(Object objeto)***. Sin embargo, lo mejor es construir un JSONObject antes e introducir los pares "clave", valor en este objeto, para posteriormente añadir ese objeto al JSONArray. De esta manera, si queremos recuperar los valores, podremos recuperar primero el JSONObject y después el valor de este objeto que coincida con la "clave" pasada como parámetro.
Éste método tiene que ir bajo un bloque try/catch, ya que se puede producir la excepción JSONException.

```
// Variables locales para recoger la información que contienen los EditText.
String nombre = ed_nombre.getText().toString();
int edad = Integer.parseInt(ed_edad.getText().toString());

//Creación de un objeto JSONObject vacío para insertar los valores
//en el JSONArray y luego poder identificarlos por su clave.
JSONObject json = new JSONObject();

//Añadir elementos al JSONObject.
try {
    json.put("nombre", nombre);
    json.put("edad", edad);
} catch (JSONException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

JSONarray.put(json.toString());
```

Leer elementos

1. Se utiliza el método ***get(int indice)***. En el caso de que hayamos añadido objetos JSONObject, nos devolverá un objeto correspondiente con el índice indicado, sí existe. En el caso de que hayamos añadido valores directamente, nos devolverá el valor asociado a ese índice, sí existe.
Esta sentencia produce la excepción JSONException en caso de que no haya ningún valor en el índice indicado.

2. Otro método que podemos utilizar es ***getJSONObject(int índice)***. Eso sí, tenemos que estar seguros de que lo que contiene el JSONArray son objetos JSONObject. De esta manera, nos ahorramos hacer el cast.

```
//Bucle para recorrer el JSONArray.
for(int i = 0; i < array.length(); i++){
    //Creación de una variable JSONObject para poder guardar cada objeto JSONObject que
    //contiene el array JSONArray.
    JSONObject jsonTemp = null;

    try {
        //Recojemos cada objeto JSONObject que devuelve
        //el método en la variable creada anteriormente.
        jsonTemp = array.getJSONObject(i);

        //Otra manera de guardar en una variable los JSONObject que contiene el JSONArray.
        jsonTemp = (JSONObject) array.get(i);
    } catch (JSONException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

Creación de un proyecto que utilice JSON

Lo único que hay que hacer para poder manejar objetos JSON, es importar las siguientes librerías:

```
import org.json.JSONObject;
import org.json.JSONArray;
import org.json.JSONException;
```

O también podemos hacer lo siguiente:

```
import org.json.*;
```

Después sólo hay que crear los objetos que necesitemos de las clases JSONObject o JSONArray para manejar los datos JSON.

Uso de los JSONObject y JSONArray

Sirven para el procesado de cadenas JSON que pueden obtenerse de:

- Un fichero.
- Un Web service SOAP.
- Una url.

Bibliografía

Documentación de Android Developers sobre JSONObject y JSONArray.
-http://developer.android.com/reference/org/json/JSONObject.html
-http://developer.android.com/reference/org/json/JSONArray.html
¿Qué es JSON? y su formato.
-http://geekytheory.com/json-i-que-es-y-para-que-sirve-json/
Ejemplo de añadir elementos a un JSONObject y ejemplo leyendo errores de Bugzilla.
-http://www.vogella.com/tutorials/AndroidJSON/article.html
Conectar a Internet y leer documentos JSON (Proyecto JSON para ver imágenes de Flickr).
-http://www.proyectosimio.com/es/programacion-android-asynctask-conectar-a-internet-y-leer-documentos-json/
Porque es mejor JSON que XML para la transferencia de datos.
-http://directoandroid.es/tag/jsonobject/
Para que sirven los JSONObject.
-http://androcode.es/2012/05/parseando-json-en-android/

¡Gracias por vuestra atención!