

ANDROID >> INTEGRANDO UN RESTFUL WEB SERVICE EN ANDROID



Los servicios del tipo REST son los más utilizados en la actualidad a la hora de crear un servicio Web. Si no estás familiarizado con este tipos de servicios puedes echar un vistazo a [este artículo](#) para una breve introducción.

Creando la aplicación

Vamos a ver en esta entrada como conectar una aplicación Android con un servicio Web RESTful. Para ello vamos a usar la API REST que nos proporciona [My Movie API](#) que nos permite acceder a la información almacenada en la Web [IMDb](#). La aplicación descargará la información de una película desde la Web, para ello realizaremos una petición del tipo **GET** que nos devolverá a su vez un objeto JSON con dicha información.

El diseño de la aplicación va a ser muy sencillo, ya que únicamente contará con un *layout* y una actividad:

res / layout / activity_main.xml

```
1 <LinearLayout
2   xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical"
7     android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
```

```

8      android:paddingTop="@dimen/activity_vertical_margin"
9      tools:context=".MainActivity" >
10
11      <TextView
12          android:layout_width="wrap_content"
13          android:layout_height="wrap_content"
14          android:text="Introduce el nombre de la película" />
15
16      <EditText
17          android:id="@+id/input_película"
18          android:layout_width="match_parent"
19          android:layout_height="wrap_content"
20          android:layout_marginTop="10dp"
21          android:inputType="text" />
22
23      <Button
24          android:layout_width="wrap_content"
25          android:layout_height="wrap_content"
26          android:drawableRight="@android:drawable/ic_menu_search"
27          android:onClick="buscarPelícula"
28          android:text="Buscar" />
29
30  </LinearLayout>
31

```

MainActivity.java

```

1  package com.amatellanes.pelicularest;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.text.TextUtils;
6  import android.view.View;
7  import android.widget.EditText;
8
9  public class MainActivity extends Activity {
10
11      private EditText inputPelícula;
12
13      @Override
14      protected void onCreate(Bundle savedInstanceState) {
15          super.onCreate(savedInstanceState);
16          setContentView(R.layout.activity_main);
17
18          inputPelícula = (EditText)
19              findViewById(R.id.input_película);
20      }
21
22      public void buscarPelícula(View view) {
23          String titulo = inputPelícula.getText().toString();
24          if (!TextUtils.isEmpty(titulo)) {
25
26          }
27      }
28
29  }

```

```
24    }
25
26
27
```

Si ejecutamos ahora la aplicaciones únicamente tendremos una entrada de texto y un botón que al ser pulsado almacenará el valor introducido en dicha entrada.

Realizando la petición GET

Para realizar la petición GET y poder acceder a la información devuelta en la respuesta vamos a usar la librería [http-request](#). Para instalar dicha librería en nuestra aplicación podemos incluir la dependencia Maven en nuestro proyecto o descargar el fichero [HttpRequest.java](#) y añadirlo a nuestro proyecto.

Una vez incluida la librería, lo primero que deberemos hacer para buscar una película es obtener el nombre de la película introducido por el usuario, y a generar la URL para realizar la petición. Si visitamos [la página](#) veremos que se pueden especificar en la URL varios parámetros, pero para este ejemplo únicamente vamos a especificar el nombre, el tipo de la respuesta y el número de registros que queremos obtener:

<http://mymovieapi.com/?title=the+dark+knight&type=json&limit=10>

Únicamente tendremos que parametrizar el valor especificado en el parámetro `title`, ya que el parámetro `type` deberá ser `json` y `limit` será `10`. Para realizar la petición GET deberemos añadir en nuestra actividad una tarea `AsyncTask`. Definimos un `AsyncTask` porque no se debe usar un objeto `HttpRequest` en el hilo principal de la aplicación:

MainActivity.java

```
1
2    public static final String TAG = "com.amatellanes.pelicularest";
3
4    private class LoadFilmTask extends AsyncTask<String, Long, String> {
5        protected String doInBackground(String... urls) {
6            try {
7                return
8                HttpRequest.get(urls[0]).accept("application/json")
9                    .body();
10           } catch (HttpRequestException exception) {
11               return null;
12           }
13       }
14
15       protected void onPostExecute(String response) {
16           Log.i(TAG, response);
17       }
18   }
```

En primer lugar realizamos la petición HTTP en el método `doInBackground()` usando el método `get()` que nos proporciona la librería [http-request](#), especificamos además que se nos devuelva el cuerpo de la respuesta con el método `body()`. La respuesta de la petición será tratada en el método `onPostExecute()`. Ahora actualizamos el método `buscarPelícula()` para que se cree un nuevo hilo que realizarán la petición con la URL que creamos a partir del nombre de la película introducido:

MainActivity.java

```
1 public void buscarPelícula(View view) {
2     String titulo = inputPelícula.getText().toString();
3     if (!TextUtils.isEmpty(titulo)) {
4         String url = String.format(
5             "http://mymovieapi.com/?title=%1$s&type=json&limit=10",
6             titulo);
7         new LoadFilmTask().execute(url);
8     }
9 }
```

Si ejecutamos ahora la aplicación veremos que en nuestro *log* aparecerá el objeto JSON con la información de la película. Antes de hacer la prueba asegúrate que tu dispositivo tenga conexión a Internet.

Parseando el objeto JSON

Ahora vamos a ver como usar la librería [Gson](#) para parsear el objeto JSON que hemos obtenido. Vamos a empezar descargando la librería desde [este enlace](#), además podrás encontrar toda la información y ayuda sobre el uso de esta librería. Descargaremos un fichero .jar que añadiremos a nuestro proyecto.

Vamos a empezar viendo como mostrar el objeto JSON formateado en pantalla, ya que nos resultará útil cuando estemos desarrollando nuestra aplicación. Añadimos un `TextView` a nuestro *layout* `activity_main.xml` y modificamos nuestra actividad principal `MainActivity.java`:

MainActivity.java

```
1 private class LoadFilmTask extends AsyncTask<String, Long, String> {
2     protected String doInBackground(String... urls) {
3         try {
4             return
5                 HttpRequest.get(urls[0]).accept("application/json")
6                     .body();
7         } catch (HttpRequestException exception) {
8             return null;
9         }
10    }
```

```

10     protected void onPostExecute(String response) {
11         Log.i(TAG, response);
12         TextView textView = (TextView) findViewById(R.id.texto);
13         textView.setText(prettyfyJSON(response));
14     }
15 }
16
17     private String prettyfyJSON(String json) {
18         Gson gson = new
19     GsonBuilder().setPrettyPrinting().create();
20         JsonParser parser = new JsonParser();
21         JsonElement element = parser.parse(json);
22         return gson.toJson(element);
23     }
24

```

Ahora vamos a ver como parsear el objeto JSON en un objeto Java. Para ello primero deberemos crear una clase Java que incluya los campos del objeto JSON. Usando esta librería no tendremos que especificar todos los atributos del objeto JSON que vamos a parsear. Si echamos un vistazo a [la información](#) que obtenemos en cada petición, el objeto JSON que obtenemos consta únicamente de *string*, números y arrays de *string*, a excepción del atributo `poster` que es un objeto JSON anidado. Por lo tanto deberemos crear dos clase una para la información de la película y otra para la del póster. El nombre de las propiedades de las clases deben coincidir con el nombre de los atributos del objeto JSON que vamos a seleccionar:

Pelicula.java

```

1     package com.amatellanes.pelicularest;
2
3     public class Pelicula {
4
5         private String title;
6         private int year;
7         private String[] writers;
8         private String[] actors;
9         private String plot_simple;
10        private Poster poster;
11
12        public String getTitle() {
13            return title;
14        }
15
16        public void setTitle(String title) {
17            this.title = title;
18        }
19
20        public int getYear() {
21            return year;
22        }
23    }
24

```

```

21     public void setYear(int year) {
22         this.year = year;
23     }
24     public String[] getWriters() {
25         return writers;
26     }
27
28     public void setWriters(String[] writers) {
29         this.writers = writers;
30     }
31     public String[] getActors() {
32         return actors;
33     }
34
35     public void setActors(String[] actors) {
36         this.actors = actors;
37     }
38     public String getPlot_simple() {
39         return plot_simple;
40     }
41
42     public void setPlot_simple(String plot_simple) {
43         this.plot_simple = plot_simple;
44     }
45     public Poster getPoster() {
46         return poster;
47     }
48
49     public void setPoster(Poster poster) {
50         this.poster = poster;
51     }
52 }
53
54
55
56
57
58
59
60

```

Poster.java

```

1     package com.amatellanes.pelicularest;
2
3     public class Poster {
4
5         private String imdb;
6         private String cover;
7
8         public String getImdb() {
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

8         return imdb;
9     }
10
11     public void setImdb(String imdb) {
12         this.imdb = imdb;
13     }
14
15     public String getCover() {
16         return cover;
17     }
18
19     public void setCover(String cover) {
20         this.cover = cover;
21     }
22
23
24

```

La librería Gson mapeará automáticamente el objeto `Poster` incluido en el objeto `Pelicula`. Si nos fijamos el objeto JSON devuelto es un array donde cada posición se corresponde con un objeto JSON que contiene la información de la película, por lo que el parseo se deberá implementar de la siguiente manera.

MainActivity.java

```

1 private List<Pelicula> getPeliculas(String json) {
2     Gson gson = new Gson();
3     Type type = new TypeToken<ArrayList<Pelicula>>(){}.getType();
4     return gson.fromJson(json, type);
5 }

```

Si quieres ver un ejemplo de un parseo de un objeto JSON simple puedes consultarlo en [esta página](#).

Al método anterior le pasaremos el objeto JSON que obtenemos en la respuesta y éste será convertido una lista de objetos `Pelicula`. La acción opuesta (de objeto Java a objeto JSON) puedes verlo en [la documentación](#) de la librería Gson.

Una vez obtenemos una lista de objetos Java vamos a mostrar como primer ejemplo un único elemento. Modificamos nuestro *layout* principal donde añadiremos los elementos que queremos mostrar de la película:

res / layout / activity_main.xml

```

1 <LinearLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"

```

```

4      android:layout_height="match_parent"
5      android:orientation="vertical"
6      android:paddingBottom="@dimen/activity_vertical_margin"
7      android:paddingLeft="@dimen/activity_horizontal_margin"
8      android:paddingRight="@dimen/activity_horizontal_margin"
9      android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context=".MainActivity" >
11
12     <TextView
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:text="Introduce el nombre de la película" />
16
17     <EditText
18         android:id="@+id/input_pelicula"
19         android:layout_width="match_parent"
20         android:layout_height="wrap_content"
21         android:layout_marginTop="10dp"
22         android:inputType="text"
23         android:text="The Dark Knight" />
24
25     <Button
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content"
28         android:drawableRight="@android:drawable/ic_menu_search"
29         android:onClick="buscarPelicula"
30         android:text="Buscar" />
31
32     <RelativeLayout
33         android:layout_width="match_parent"
34         android:layout_height="wrap_content" >
35
36         <ImageView
37             android:id="@+id/ivPoster"
38             android:layout_width="wrap_content"
39             android:layout_height="wrap_content"
40             android:layout_alignParentLeft="true"
41             android:contentDescription="@string/app_name" />
42
43         <TextView
44             android:id="@+id/tvTitle"
45             android:layout_width="match_parent"
46             android:layout_height="wrap_content"
47             android:layout_alignParentTop="true"
48             android:layout_toRightOf="@id/ivPoster"
49             android:singleLine="true" />
50
51         <TextView
52             android:id="@+id/tvWriters"
53             android:layout_width="match_parent"
54             android:layout_height="wrap_content"
55             android:layout_below="@id/tvTitle"
56             android:layout_toRightOf="@id/ivPoster"
57             android:singleLine="true" />
58
59         <TextView
60             android:id="@+id/tvActors"
61             android:layout_width="match_parent"
62             android:layout_height="wrap_content"
63             android:layout_below="@id/tvWriters"

```



```

54         android:layout_toRightOf="@id/ivPoster"
55         android:singleLine="true" />
56
57         <TextView
58             android:id="@+id/tvPlot"
59             android:layout_width="match_parent"
60             android:layout_height="wrap_content"
61             android:layout_below="@id/tvActors"
62             android:layout_toRightOf="@id/ivPoster"
63             android:maxLines="2" />
64     </RelativeLayout>
65
66 </LinearLayout>
67
68
69
70
71
72
73
74
75
76

```

Tendremos que modificar el inicio de nuestra actividad para preparar el nuevo *layout*:

MainActivity.java

```

1
2     private ImageView ivPoster;
3     private TextView tvTitle, tvWriters, tvActors, tvPlot;
4
5     @Override
6     protected void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.activity_main);
9
10        prepareUI();
11    }
12
13    private void prepareUI() {
14        inputPelicula = (EditText) findViewById(R.id.input_pelicula);
15
16        ivPoster = (ImageView) findViewById(R.id.ivPoster);
17        tvTitle = (TextView) findViewById(R.id.tvTitle);
18        tvWriters = (TextView) findViewById(R.id.tvWriters);
19        tvActors = (TextView) findViewById(R.id.tvActors);
20        tvPlot = (TextView) findViewById(R.id.tvPlot);
21    }
22

```

Ahora definimos una función que muestre la información de la película por pantalla. Como vemos en el objeto JSON que devuelve la Web y que es parseado a un objeto `Poster`, obtenemos la URL de la imagen del póster de la película, por lo que deberemos de cargar la imagen en nuestra aplicación a partir de esta dirección. Para ello vamos a hacer uso de la librería [Picasso](#). Puedes incluirla como una dependencia Maven o descargar el fichero .jar, copiarlo en el directorio `libs` de tu aplicación y añadirlo al *path* de la aplicación. El método quedaría de la siguiente manera:

MainActivity.java

```
1
2 private void mostrarPelicula(Pelicula pelicula) {
3
4     tvTitle.setText(pelicula.getTitle());
5     tvWriters.setText(Arrays.toString(pelicula.getWriters()));
6     tvActors.setText(Arrays.toString(pelicula.getActors()));
7     tvPlot.setText(pelicula.getPlot_simple());
8
9     if (pelicula.getPoster() != null
10         && pelicula.getPoster().getImdb() != null) {
11         Picasso.with(getApplicationContext())
12             .load(pelicula.getPoster().getImdb()).into(ivPoster);
13     }
14 }
```

Como vemos, a la hora de cargar el póster, comprobamos en primer lugar que existe dicho póster y si es así, cargamos la imagen en la aplicación usando la librería Picasso. Por último, modificamos el método `onPostExecute()` de nuestro `AsyncTask`:

MainActivity.java

```
1
2 protected void onPostExecute(String response) {
3     List<Pelicula> peliculas = getPeliculas(response);
4
5     if (!peliculas.isEmpty()) {
6         mostrarPelicula(peliculas.get(0));
7     }
8 }
```

Si arrancamos la aplicación veremos como se carga la información de la película seleccionada correctamente:

El siguiente paso es construir una lista con la información obtenida en la petición. Para ello podemos usar lo visto en [esta entrada](#) donde explico el uso del elemento `ListView` en Android. Para no alargar más esta entrada os dejo el código en mi [GitHub](#) y aquí un vídeo con el resultado:

Descargar código | [GitHub](#)

Share this:

- [Twitter](#)
- [Facebook](#)
-

Relacionado

[Android >> Ejemplo de WebView en Android \(Parte 1 \)](#)En "Android"

[Android >> Crear un splash screen en Android](#)En "Android"

[Android >> Repetir una imagen como fondo en Android](#)En "Android"

Etiquetado [android](#), [api](#), [rest](#), [restful](#)

Deja una respuesta




[← Entrada anterior](#)

[Siguiete entrada →](#)

Buscar:

SÍGUEME EN TWITTER

Mis tuits

 [RSS - Entradas](#)

ENTRADAS RECIENTES

- [Django + Heroku >> Desplegando una aplicación Django en Heroku](#)
- [Android >> Integrando un RESTful Web Service en Android](#)
- [Android >> Ejemplo de Navigation Drawer en Android \(Parte 2 \)](#)
- [Android >> Monitorización de errores usando ACRA](#)
- [DropzoneJs + Django: How to build a file upload form](#)

ARCHIVOS

- [febrero 2014](#)
- [enero 2014](#)
- [diciembre 2013](#)
- [noviembre 2013](#)
- [octubre 2013](#)
- [septiembre 2013](#)
- [agosto 2013](#)

- [julio 2013](#)
- [junio 2013](#)
- [mayo 2013](#)
- [abril 2013](#)
- [marzo 2013](#)

CATEGORÍAS

- [Android](#)
- [Diseño Web](#)
- [Django](#)
- [Git](#)
- [GitHub](#)
- [Java](#)
- [Python](#)
- [Ubuntu](#)
- [Uncategorized](#)
- [WordPress](#)

[Blog de WordPress.com.](#) | [El Tema Chunk.](#)

[Seguir](#)

Seguir “”

Recibe cada nueva publicación en tu buzón de correo electrónico.

Únete a otros 66 seguidores

[Suscribeme](#)

[Construye un sitio web con WordPress.com](#)

☺