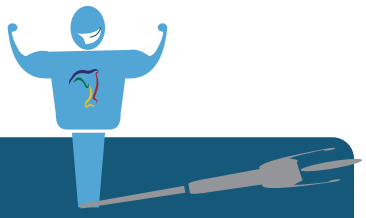
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

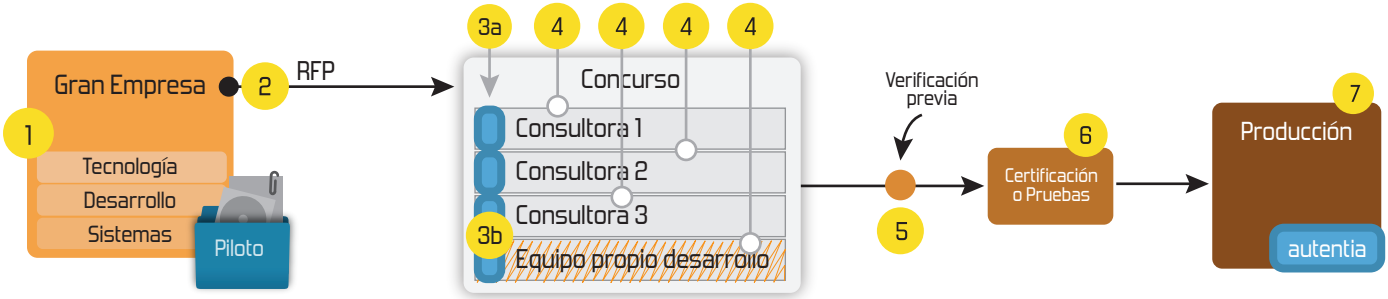
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

DESARROLLADO POR:
Francisco J. Arroyo

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE

Fecha de publicación del tutorial: 2009-02-26

12

Google Custom Search Api desde Android

0. Índice de contenidos.

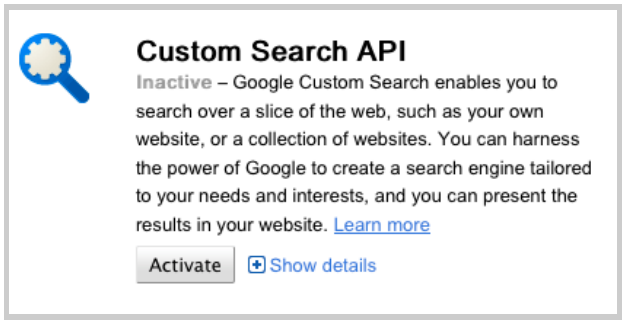
1. Introducción

Debido a un correo que recibí hace poco, tuve que buscar información sobre como podía realizar búsquedas en internet desde un disposivo movil y poder tratar esos resultados sin tener que abrir un navegador.

Entonces pensé, que buscador importante puede haber que devuelva buenos resultados y además provea una api para poder utilizar sus servicios. El primer buscador que viene a la mente es como no Google, que además proporciona apis para la gran mayoría de sus servicios. Buscando información dí con esta página http://code.google.com/intl/es/apis/customsearch/v1/overview.html, que es justo lo que estaba buscando ;). El servicio nos permite realizar 100 consultas gratuitas, si tenemos previsto realizar más consultas tendremos que utilizar otro servicio o pasar por caja.

2. Requisitos previos

Como requisito indispensable es disponer de una cuenta de gmail. Además necesitamos una clave que nos identificará a la hora de realizar las consultas. Para conseguir la clave nos dirigimos a https://code.google.com/apis/console/?api=customsearch&pli=1#welcome , y aceptamos la licencia. A continuación nos saldrá una ventana como la que aparece a continuación



Si en el botón aparece "Activate", lo pulsamos para activar, y si aparece "Deactivate" quiere decir que ya tenemos el servicio habilitado.

Para conocer nuestra clave, nos dirigimos a "API Access" en el menú de la derecha, y en el apartado "Access key" aparecerá nuestra clave.

Últimas Noticias

Preparando el sexto "Autentia Cycling Day"

XV Charla Autentia - web2py (y Google App Engine) - RECORDATORIO

XV Charla Autentia - web2py (y Google App Engine)

XIV Charla Autentia - ZK - Vídeos y Material

Hablando de coaching ágil, milagro nocturo y pruebas de vida

Histórico de NOTICIAS

Últimos Tutoriales

Hibernate - Como definir la forma de persistir nuestros objetos mediante la interfaz CompositeUserType.

RVM y como actualizar Ruby a la versión 1.9.2 en Snow Leopard 10.6.7

toi18n, Traduce tus aplicaciones de forma rápida Online

Instalación de ntfs-3g para Mac OS X

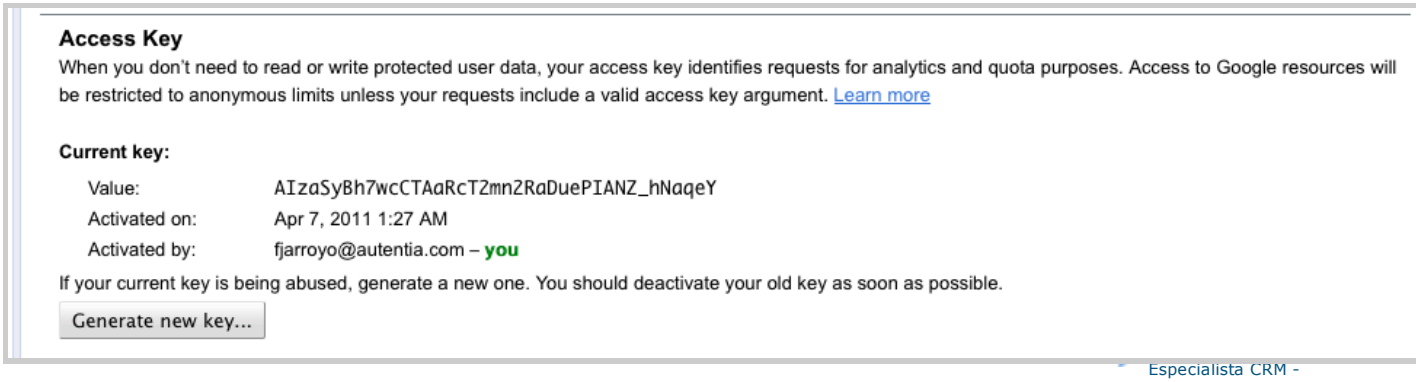
Consumir un servicio web Axis con Android

Últimos Tutoriales del Autor

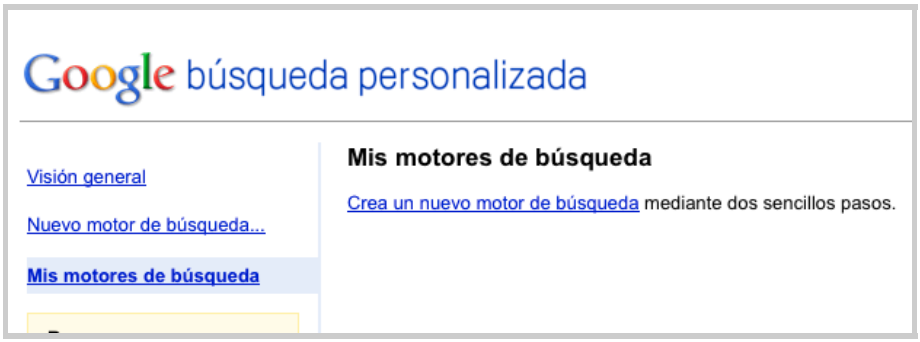
Instalación de ntfs-3g para Mac OS X

Consumir un servicio web Axis con Android

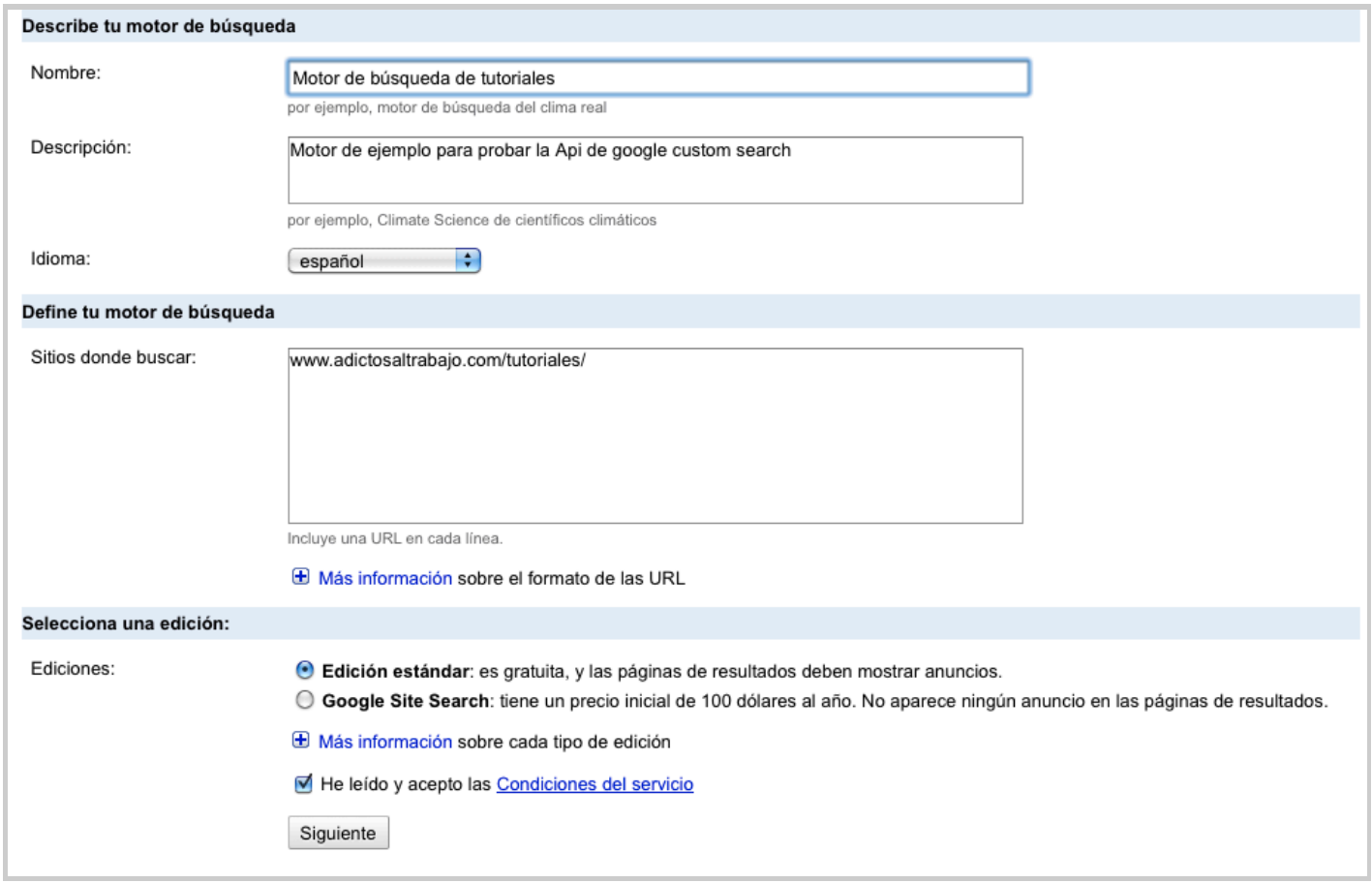
Síguenos a través



Además vamos a crear un motor de búsqueda personalizado. Nos dirigimos a http://www.google.com/cse/ y pinchamos en "Crear motor de búsqueda".



Rellenamos la página como aparece a continuación. Lo único que tenéis que cambiar es las páginas en donde se van a realizar las búsquedas.



A continuación pulsamos en siguiente. Nos mostrará una pantalla en donde podemos utilizar un cuadro de texto para probar la configuración de nuestro buscador personalizado. Pulsamos en siguiente y ya tendremos creado nuestro motor de búsqueda personalizado.

Ahora que tenemos el motor de búsqueda creado, necesitamos anotar un valor que más adelante vamos a necesitar. Entonces nos dirigimos a http://www.google.com/cse/manage/all?hl=es y clickeamos en el "Panel de control" del motor de búsqueda que acabamos de crear. Nos fijamos en el apartado que pone "ID único del motor de búsqueda" y anotamos el valor que aparece. En mi caso es 010306700704176754880:j_bbeym3x0g.

3. Uso de la API

Para utilizar esté servicio, tenemos que realizar una petición a una URL que deberemos componer nosotros. En el ejemplo que nos proporciona la documentación podemos ver

https://www.googleapis.com/customsearch/v1?key=INSERT-YOUR-KEY&cx=017576662512468239146:omuauf_lfve&q=lectures.

Si desglosamos la url tenemos:

- https://www.googleapis.com/customsearch/v1
- **key**=INSERT-YOUR-KEY
- **cx**=017576662512468239146:omuauf_lfve
- **q**=lectures

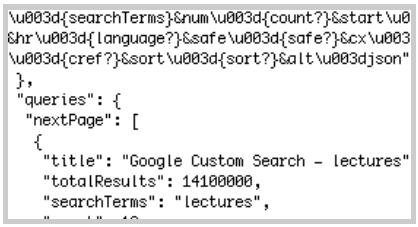
Como podemos ver, tenemos pasamos tres parámetros mediente la url. También es posible pasar un parámetro "callback" para especificarle una función de callback (http://code.google.com/intl/es-ES/apis/customsearch/v1/getting_started.html#JSONP).

El parámetro "key", hace referencia a la clave que hemos creado anteriormente.

El parámetro "cx", hace referencia al contexto con el que se realizarán las búsquedas. Es el identificador único que hemos obtenido al crear nuestro motor de búsqueda personalizado.

El parámetro "q", hace referencia a la búsqueda que vamos a realizar. Por ejemplo "hibernate", "spring", y en caso del ejemplo es "lectures".

Si abrimos un navegador e introducimos en la barra de direcciones la url formada con nuestros campos, obtendremos una cadena json con los resultados obtenidos.



4. Creando el cliente.

Ahora vamos a crear nuestro cliente para mostrar los resultados. Abrimos eclipse y creamos un nuevo proyecto de Android. Nuestra aplicación de Android va a constar de un activity que estará compuesto de un TextView en donde se pondrá lo que se quiere buscar, un botón para realizar la búsqueda y un ListView para mostrar los resultados que nos devuelva la búsqueda.

Las clases principales que vamos a necesitar para realizar nuestra aplicación son:

- HttpRequest: Clase que va a realizar la conexión con el servicio de google.

view plain print ?

```
01. package com.adictosaltrabajo.adictostutoriales.util;
02.
03. import java.io.BufferedReader;
04. import java.io.InputStream;
05. import java.io.InputStreamReader;
06. import java.net.HttpURLConnection;
07. import java.net.URL;
08.
09. public class HttpRequest {
10.
11.     public static synchronized String doRequest(String urlRequest) throws Exception
12.     {
13.         String jsonRespuesta = "";
14.
15.         HttpURLConnection urlConnection= null;
16.         URL url = new URL(urlRequest);
17.         urlConnection=(HttpURLConnection)url.openConnection();
18.         urlConnection.setRequestMethod("GET");
19.         jsonRespuesta = convertStreamToString(urlConnection.getInputStream());
20.
21.         return jsonRespuesta;
22.     }
23.
24.     private static String convertStreamToString(InputStream is) throws Exception {
25.         BufferedReader reader = new BufferedReader(new InputStreamReader(is));
26.         StringBuilder sb = new StringBuilder();
27.         String line = null;
28.         while ((line = reader.readLine()) != null) {
29.           sb.append(line + "\n");
30.         }
31.         is.close();
32.         return sb.toString();
33.       }
34.
35. }
36.
```

- UrlFactory: Clase que nos va crear una url con nuestros parámetros para realizar la petición

```
view plain  print  ?
01. package com.adictosaltrabajo.adictostutoriales.util;
02.
03. import java.net.URLEncoder;
04.
05. public class UrlFactory {
06.
07.     private static final String url = "https://www.googleapis.com/customsearch/v1";
08.     private static final String cx = "010306700704176754880:j_bbeym3x0g";
09.     private static final String key = "INSERT YOUR KEY HERE";
10.
11.     public static String buildUrl(String keywords) throws Exception{
12.         return url + "?
key=" + key + "&cx=" + cx + "&q=" + URLEncoder.encode(keywords);
13.     }
14.
15. }
16.
```

- ResultDeserializer: Clase que nos va a transformar el objeto jSon devuelto por el servicio en una lista de items con los resultados de la búsqueda

```
view plain  print  ?
01. package com.adictosaltrabajo.adictostutoriales.util;
02.
03. import java.util.ArrayList;
04. import java.util.List;
05.
06. import org.json.JSONArray;
07. import org.json.JSONException;
08. import org.json.JSONObject;
09.
10. import com.adictosaltrabajo.adictostutoriales.bean.Item;
11. import com.adictosaltrabajo.adictostutoriales.bean.Result;
12.
13.
14. public class ResultDeserializer{
15.
16.     public static Result deserialize(String object) throws JSONException
17.     {
18.         Result result = new Result();
19.
20.         final List<Item> items = new ArrayList<Item>();
21.
22.         JSONObject jSonObject = new JSONObject(object);
23.         JSONArray jSonObjectArray = jSonObject.getJSONArray("items");
24.         for(int count = 0; count < jSonObjectArray.length();count++){
25.             JSONObject jsonItem = (JSONObject) jSonObjectArray.get(count);
26.
27.             Item item = new Item();
28.             item.setTitle(jsonItem.getString("title"));
29.             item.setHtmlTitle(jsonItem.getString("htmlTitle"));
30.             item.setLink(jsonItem.getString("link"));
31.             item.setDisplayLink(jsonItem.getString("displayLink"));
32.             item.setSnippet(jsonItem.getString("snippet"));
33.             item.setHtmlSnippet(jsonItem.getString("htmlSnippet"));
34.
35.             items.add(item);
36.         }
37.
38.         result.setItems(items);
39.
40.         return result;
41.     }
42. }
43.
```

- Item: Clase que contendrá la información de cada resultado devuelto por el servicio

view plain print ?

```
01. package com.adictosaltrabajo.adictostutoriales.bean;
02. 
03. import java.util.List;
04. 
05. public class Item {
06. 
07.     String kind;
08.     String title;
09.     String htmlTitle;
10.     String link;
11.     String displayLink;
12.     String snippet;
13.     String htmlSnippet;
14.     List<Metatag> metatags;
15.     Pagemap pagemap;
16. 
17.     public Item(){
18.     }
19. 
20.     public String getKind() {
21.         return kind;
22.     }
23. 
24.     public void setKind(String kind) {
25.         this.kind = kind;
26.     }
27. 
28.     public String getTitle() {
29.         return title;
30.     }
31. 
32.     public void setTitle(String title) {
33.         this.title = title;
34.     }
35. 
36.     public String getHtmlTitle() {
37.         return htmlTitle;
38.     }
39. 
40.     public void setHtmlTitle(String htmlTitle) {
41.         this.htmlTitle = htmlTitle;
42.     }
43. 
44.     public String getLink() {
45.         return link;
46.     }
47. 
48.     public void setLink(String link) {
49.         this.link = link;
50.     }
51. 
52.     public String getDisplayLink() {
53.         return displayLink;
54.     }
55. 
56.     public void setDisplayLink(String displayLink) {
57.         this.displayLink = displayLink;
58.     }
59. 
60.     public String getSnippet() {
61.         return snippet;
62.     }
63. 
64.     public void setSnippet(String snippet) {
65.         this.snippet = snippet;
66.     }
67. 
68.     public String getHtmlSnippet() {
69.         return htmlSnippet;
70.     }
71. 
72.     public void setHtmlSnippet(String htmlSnippet) {
73.         this.htmlSnippet = htmlSnippet;
74.     }
75. 
76.     public List<Metatag> getMetatags() {
77.         return metatags;
78.     }
79. 
80.     public void setMetatags(List<Metatag> metatags) {
81.         this.metatags = metatags;
82.     }
83. 
84. 
85. }
86. 
87. 
```

Modificamos la vista que nos crea Eclipse y añadimos un TextView, un ImageButton y un Listview para poder mostrar los resultados de la búsqueda.

view plain print ?

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffff">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_height="wrap_content" an
        <EditText android:layout_height="wrap_content" android:layout_centerVertical="true" a
        <ImageButton android:layout_height="wrap_content" android:layout_width="wrap_content"
    </RelativeLayout>
    <ListView android:layout_height="wrap_content" android:layout_width="fill_parent" androi
</LinearLayout>
```

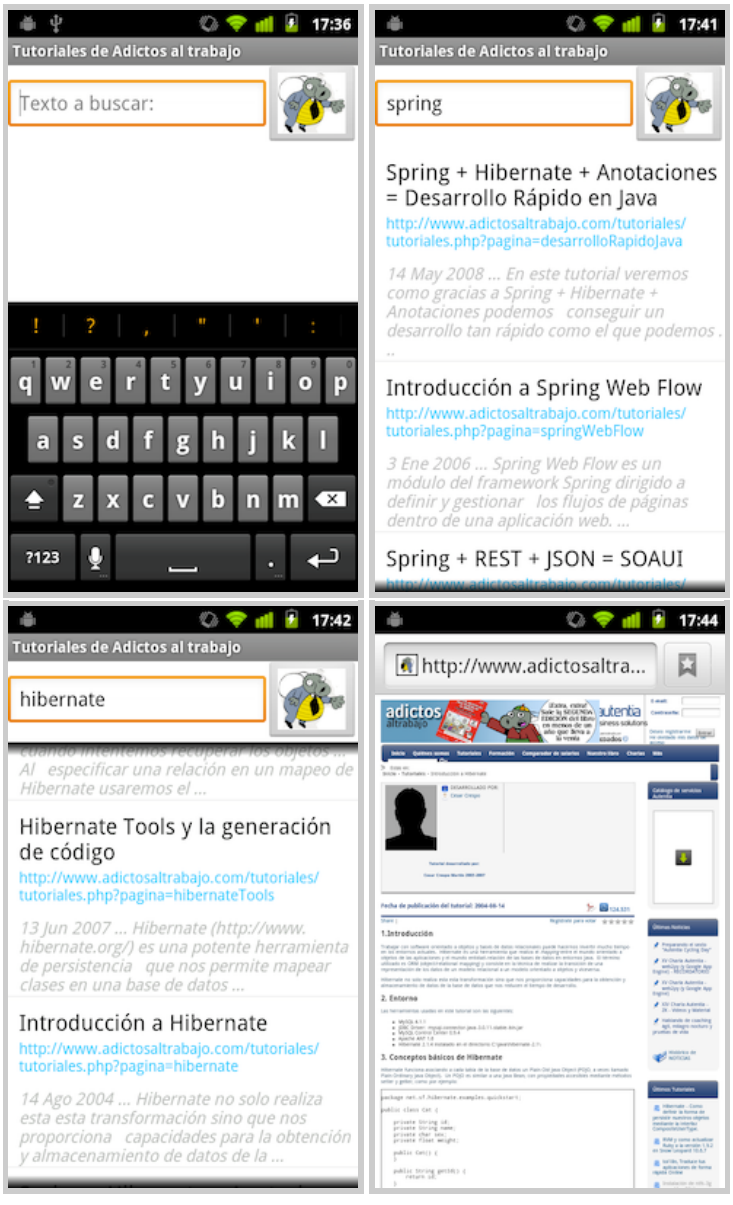
y por último creamos nuestra clase principal.

view plain print ?

```
package com.adictosaltrabajo.adictostutoriales;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

import com.adictosaltrabajo.adictostutoriales.bean.Result;

public class Main extends Activity implements OnClickListener, OnItemClickListener{

    private final Search search = new Search();
    private ImageButton ibDoSearch;
    private TextView tvSearch;
    private ListView lvResults;
    private Result result = new Result();

    final Handler handler = new Handler();
    final Runnable updateItems = new Runnable(){
        public void run(){
            updateItemsInUI();
        }
    };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        init();
    }

    protected Result updateResult(String keywords){
        Result newResult = new Result();
        try {
            newResult = search.doSearch(keywords);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return newResult;
    }

    protected void updateItemsInUI() {
        if(0 == result.getItems().size()){
            Toast.makeText(this, "No se ha encontrado ningún resultado", Toast.LENGTH_SHORT).
        }
        TutorialsInfoAdapter tutorialsInfoAdapter = new TutorialsInfoAdapter(this, result.get
        lvResults.setAdapter(tutorialsInfoAdapter);
    }

    public void init(){
        lvResults = (ListView) findViewById(R.id.lvResults);
        lvResults.setOnItemClickListener(this);
        tvSearch = (TextView) findViewById(R.id.tvSeach);
        ibDoSearch = (ImageButton) findViewById(R.id.ibSearch);
        ibDoSearch.setOnClickListener((android.view.View.OnClickListener) this);
    }

    @Override
    public void onClick(View arg0) {
        switch(arg0.getId())
        {
            case R.id.ibSearch :
                final String textToSearch = tvSearch.getText().toString().trim();
                Thread t = new Thread(){
                    public void run(){
                        result = updateResult(textToSearch);
                        handler.post(updateItems);
                    }
```

```
80.                 };
81.                 Toast.makeText(this,"Realizando búsqueda",Toast.LENGTH_SHORT).show();
82.                 t.start();
83.                 hideInputMethod();
84.                 break;
85.             }
86.         }
87.
88.         private void hideInputMethod(){
89.             InputMethodManager imm = (InputMethodManager)getSystemService(Context.INPUT_METHOD_SE
90.             imm.hideSoftInputFromWindow(tvSearch.getWindowToken(), 0);
91.         }
92.
93.         @Override
94.         public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
95.
96.             Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(result.getItems().get(arg2).
97.             startActivity(intent);
98.         }
99.
100. }
```

Aquí tenéis unas capturas para ver como funciona el invento :)



Aquí tenéis el código fuente del proyecto y el programa por si alguien quiere probarlo ;)

Proyecto para Eclipse

Apk con el programa.

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**
Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

COMENTARIOS