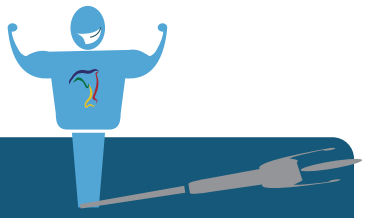
autentia

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...

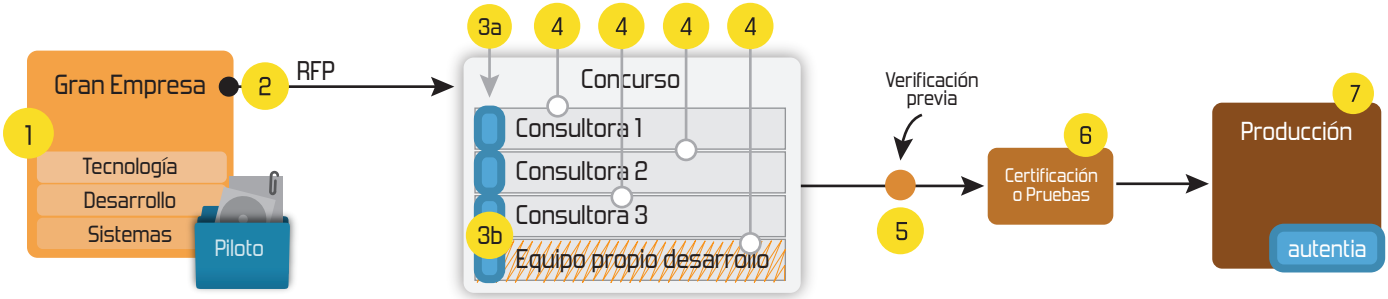
1. Desarrollo de componentes y proyectos a medida



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



E-mail:

Contraseña:

Entrar

Deseo registrarme
He olvidado mis datos de acceso



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: 2011-09-13

77

Share | Regístrate para votar

Ejemplo de viewpager para Android

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Configuración del entorno.
- 4. Creación de la aplicación
- 5. Conclusiones

1. Introducción

Si sois usuarios de móviles con Android, quizás os hayáis fijado en un componente que a mí me llama especialmente la atención. El **viewpager** es un componente que se encuentra por ejemplo en las aplicación de youtube, gtalk y google+ y nos permite desplazarnos entre pantallas deslizando el dedo horizontalmente.

Hace unos días, en la cuenta de twitter de AndroidDev, se publicó un tweet sobre este componente y un enlace a horizontal-view-swiping-with-viewpager.html hablando sobre este componente. Así que me he puesto manos a la obra para hacer algo con él, y ya de paso os cuento cómo.

2. Entorno

- Hardware
  - Mackbook Pro
    - Intel Core i7 2Ghz
    - 8GB RAM
    - 500GB HD
    - Sistema Operativo: Mac OS X (10.6.8)
  - Nexus S
    - Android 2.3.5 (Gingerbread).
- Software
  - Eclipse Indigo

Catálogo de servicios Autentia

Últimas Noticias

XPWeek en Madrid del 19 al 23 de septiembre.

Autentia en La Vuelta a España

Autentia se va de "Vuelta"

Pirineos on Tour

VII Autentia Cycling Day

Histórico de NOTICIAS

Últimos Tutoriales

CAS REST: Cómo logarnos en CAS sin ir a la pantalla de login por defecto

Usando el componente PickList de Primefaces

Clean Code: Impresiones

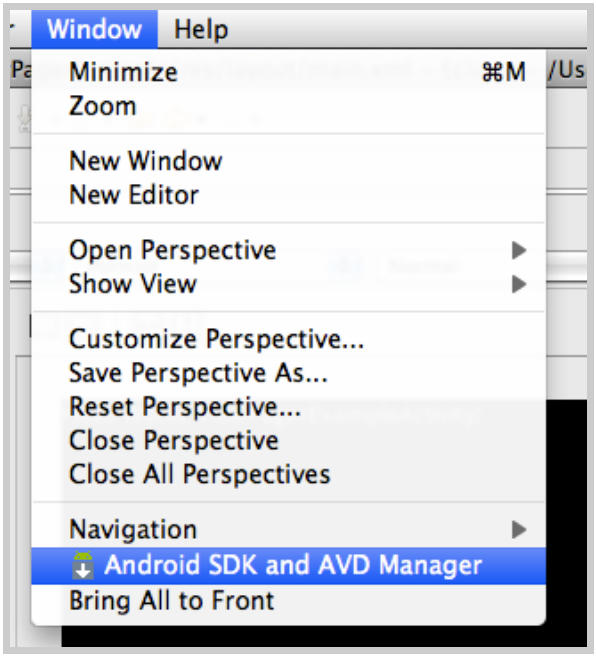
Creación de un componente en JSF2.

Uso de las anotaciones @Embeddable, @Embedded, @AttributeOverrides,

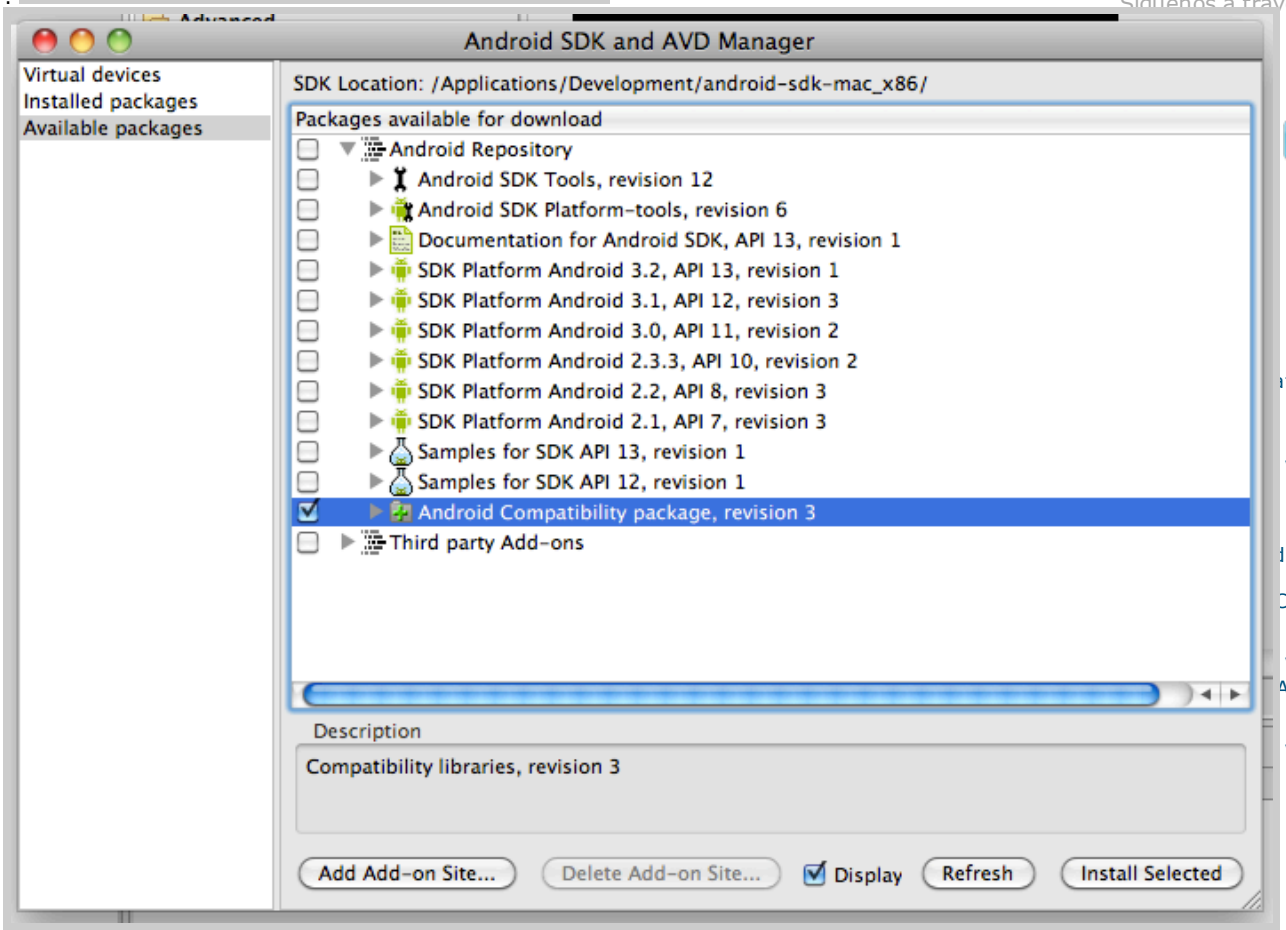
@AssociationOverrides

3. Configuración del entorno.

El primer paso es incluir el paquete de compatibilidad. Una vez que tenemos el sdk de android instalado y configurado, para descargar el paquete de compatibilidad debemos



:



Posiblemente nos salga un aviso de que va a reiniciar "adb", pulsamos en si y dejamos que continúe el proceso.

Últimos Tutoriales del Autor

- Uso de las anotaciones @Embeddable, @Embedded, @AttributeOverrides, @AssociationOverrides
- Como intentar averiguar el juego de caracteres de un archivo
- Como desarrollar un plugin para Eclipse
- Google Custom Search Api desde Android
- Instalación de ntfs-3g para Mac OS X

Síguenos a través

de

talogar

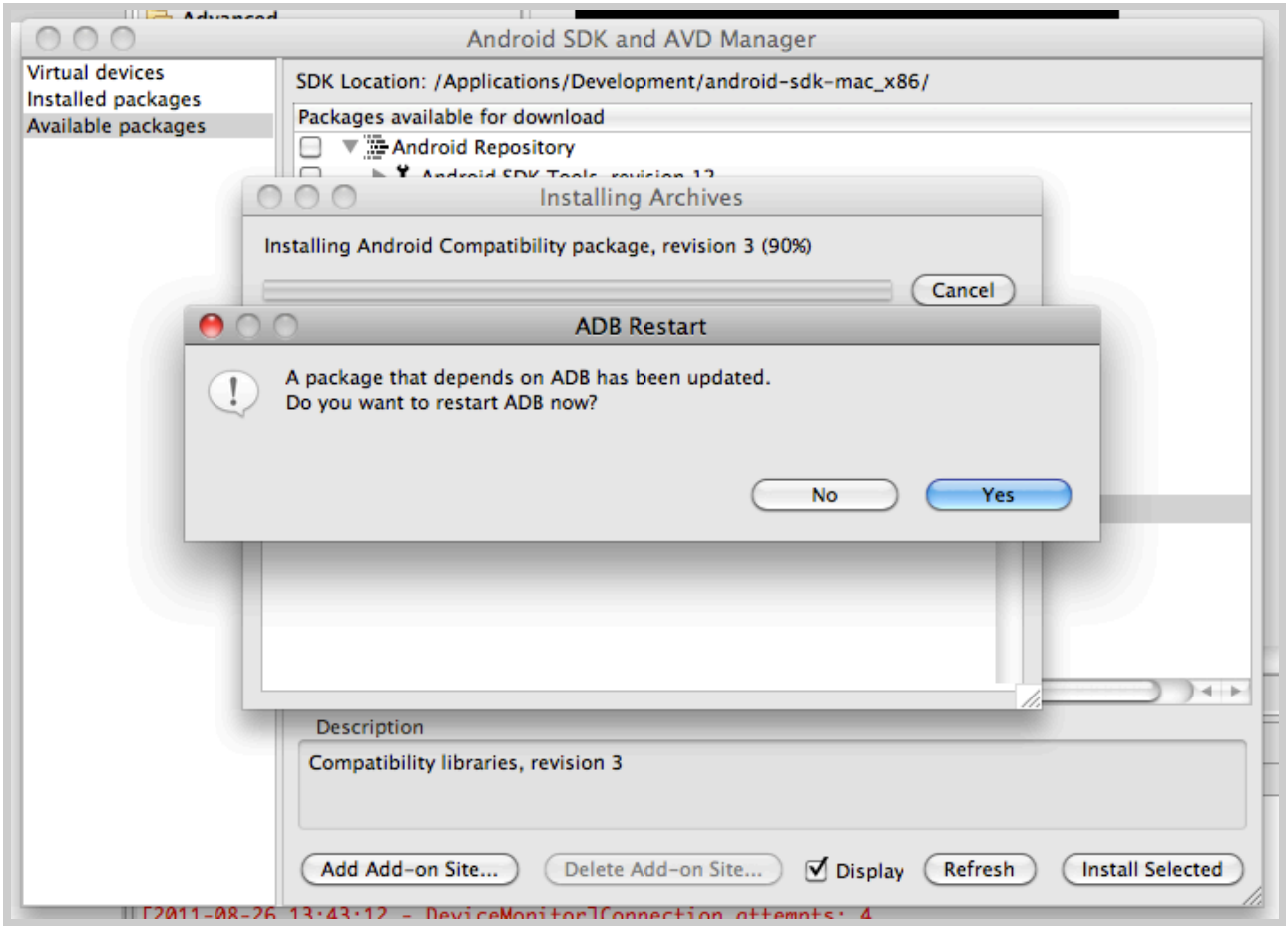
Ventas -

l -

CELONA.

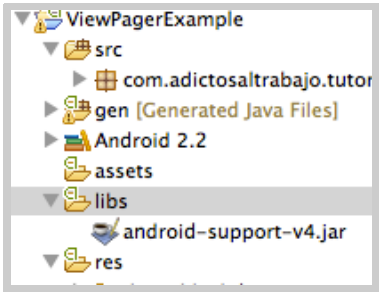
Ventas -
A.

Ventas -

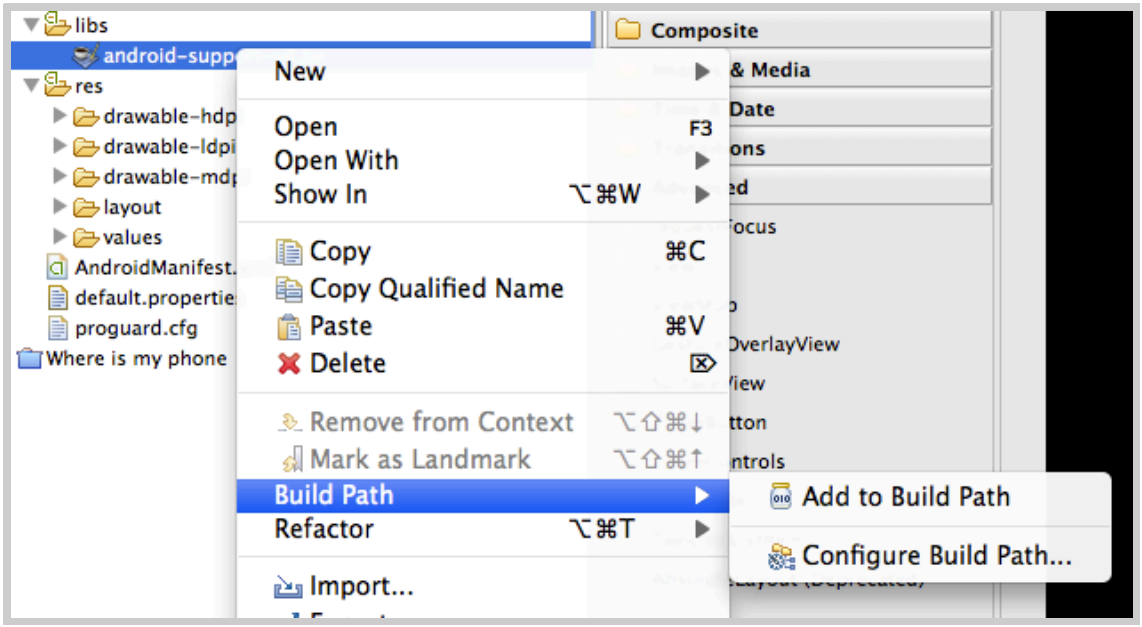


Una vez que hemos obtenido las librerías, configuramos el proyecto incluirlas.

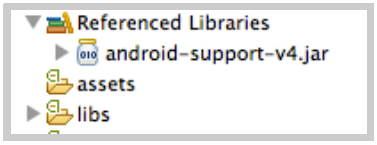
El primer paso es crear una carpeta con el nombre libs en el proyecto e incluir el **archivo android-support-v4.jar** que nos acabamos de descargar.



y a continuación añadimos la libreria al build_path



para saber si ya hemos incluido la librería, debemos tener algo como esto en el proyecto



y ya estamos listos para usarla

4. Creación de la aplicación

La aplicación consta principalmente de un **activity** encargado de mostrar el componente por pantalla y de un **PagerAdapter** encargado de alimentar al componente.

Nuestro activity mostrará un cuadro de texto y botón para realizar una búsqueda en picassaweb.

view plain print ?

```
01.   package com.adictosaltrabajo.tutoriales.android;
02.
03. import android.app.Activity;
04. import android.content.Context;
05. import android.os.Bundle;
06. import android.support.v4.view.PagerAdapter;
07. import android.support.v4.view.ViewPager;
08. import android.view.View;
09. import android.view.View.OnClickListener;
10. import android.view.inputmethod.InputMethodManager;
11. import android.widget.Button;
12. import android.widget.TextView;
13.
14. public class ViewPagerExampleActivity extends Activity {
15.
16.     private PagerAdapter pageAdapter;
17.     private ViewPager pager;
18.     private TextView tvSearch;
19.     private Context context;
20.
21.     /** Called when the activity is first created. */
22.     @Override
23.     public void onCreate(Bundle savedInstanceState) {
24.         super.onCreate(savedInstanceState);
25.         setContentView(R.layout.main);
26.         context = this;
27.
28.         configureTvSearch();
29.         configureButton();
30.
31.     }
32.
33.     private void configureTvSearch() {
34.         tvSearch = (TextView) findViewById(R.id.etQuery);
35.     }
36.
37.     private void configureButton() {
38.         final Button btSearch = (Button) findViewById(R.id.btSearch);
39.         btSearch.setOnClickListener(new OnClickListener() {
40.
41.             @Override
42.             public void onClick(View v) {
43.                 final String textToSearch = tvSearch.getText().toString().t
44.                 try {
45.                     pageAdapter = new CustomPageAdapter(context, Search.dos
46.                     pager = (ViewPager) findViewById(R.id.viewPagerId);
47.                     pager.setAdapter(pageAdapter);
48.                     hideInputMethod();
49.                 } catch (Exception e) {
50.                     // TODO Auto-generated catch block
51.                     e.printStackTrace();
52.                 }
53.
54.             }
55.         });
56.     }
57.
58.     private void hideInputMethod(){
59.         InputMethodManager imm = (InputMethodManager)getSystemService(Conte
60.         imm.hideSoftInputFromWindow(tvSearch.getWindowToken(), 0);
61.     }
62.
63. }
```

y este es el layout que vamos a utilizar

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"
    >
    <LinearLayout android:layout_height="wrap_content" android:layout_width
        <EditText android:layout_weight="1" android:layout_height="wrap_cor
        <Button android:layout_weight="1" android:layout_height="wrap_conte
    </LinearLayout>
    <android.support.v4.view.ViewPager
        android:layout_width="match_parent"
        android:id="@+id/viewPagerId" android:layout_height="match_parent"/
</LinearLayout>
```

Luego creamos nuestro custom adapter, que lo que hace es devolver las vistas según vamos desplazando el dedo horizontalmente.

view plain print ?

```
package com.adictosaltrabajo.tutoriales.android;

import java.util.List;

import android.content.Context;
import android.os.Parcelable;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.ViewPager;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

public class CustomPageAdapter extends PagerAdapter{

    private final List<String> urls;
    private final Context context;


    public CustomPageAdapter(Context context, List<String> urls) {
        super();
        this.urls = urls;
        this.context = context;
    }

    @Override
    public void destroyItem(View collection, int position, Object view) {
        ((ViewPager) collection).removeView((LinearLayout) view);
    }

    @Override
    public void finishUpdate(View arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public int getCount() {
        return urls.size();
    }

    @Override
    public Object instantiateItem(View collection, int position) {

        LinearLayout linearLayout = new LinearLayout(context);
        linearLayout.setOrientation(1);

        final TextView textView = new TextView(context);
        textView.setText("Position: " + position);

        final ImageView image = new ImageView(context);
        image.setImageBitmap(FetchImage.fetchImage(urls.get(position)));


        linearLayout.addView(textView);
        linearLayout.addView(image);

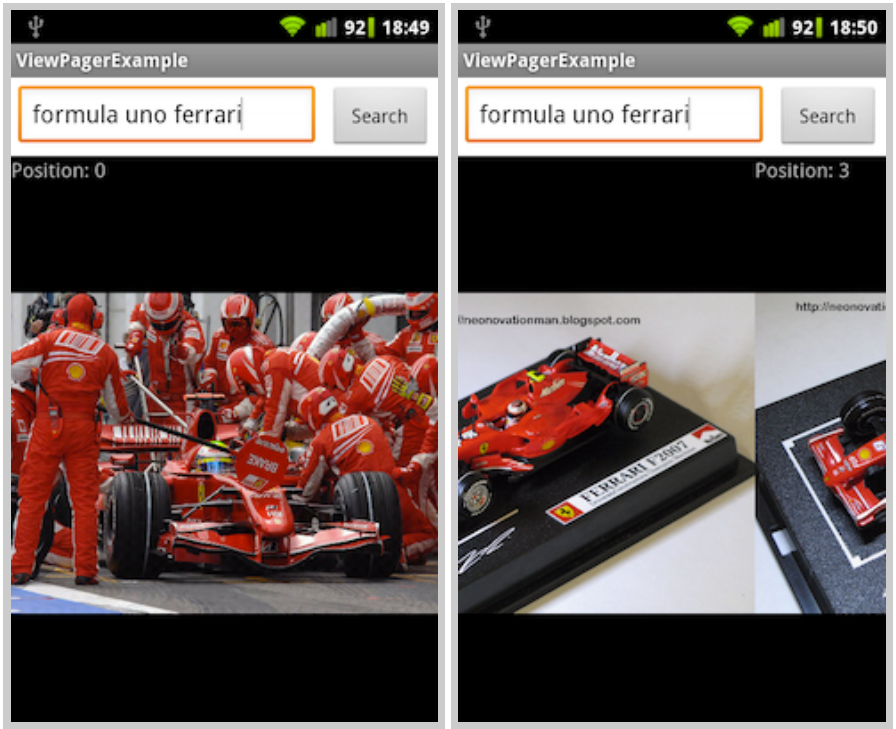
        ((ViewPager) collection).addView(linearLayout, 0);
```

```
58.          ((ViewPager) collection).addView(linearLayout,0);
59.          return linearLayout;
60.      }
61.
62.      @Override
63.      public boolean isViewFromObject(View view, Object object) {
64.           return view==((LinearLayout)object);
65.      }
66.
67.      @Override
68.      public void restoreState(Parcelable arg0, ClassLoader arg1) {
69.          // TODO Auto-generated method stub
70.
71.      }
72.
73.      @Override
74.      public Parcelable saveState() {
75.          // TODO Auto-generated method stub
76.          return null;
77.      }
78.
79.      @Override
80.      public void startUpdate(View arg0) {
81.          // TODO Auto-generated method stub
82.
83.      }
84.
85.  }
```

Y con esto ya tenemos las clases necesarias para nuestro **ViewPager** :).

5. Conclusiones

Vemos como es muy fácil y con muy poquito código somos capaces de crear interfaces bastante complejas. Como siempre, si os quereis bajar el proyecto, lo podéis hacer desde aquí





Para cualquier comentario, duda o sugerencia, tenéis el formulario que aparece a continuación.

Un saludo.

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

Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

Copyright 2003-2011 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto