Llamar a otra Activity

Aplicación Android

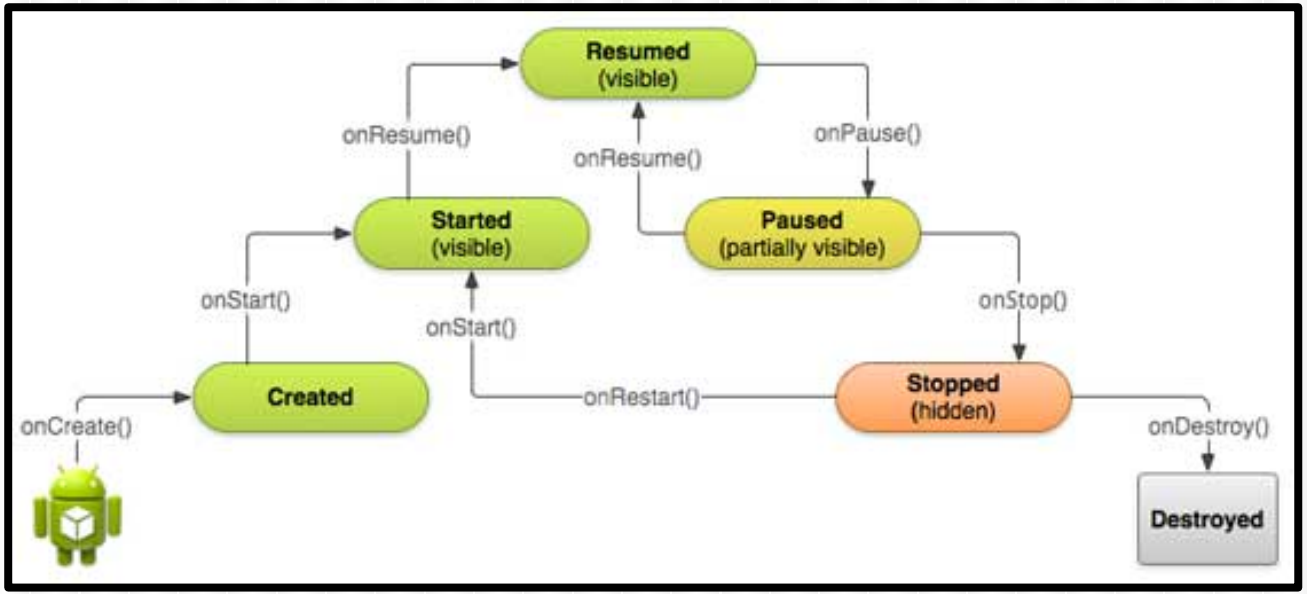
Temas

1. Responder al botón Enviar
2. Construir un “Intent”
3. Iniciar la Segunda Actividad
4. Crear la Segunda Actividad
5. Recibe el “Intent”
6. Visualizar el mensaje

Interfaz Simple de Usuario

Antes de seguir vamos a repasar los conceptos aprendidos hasta el momento:

[Video: Android Activity](#) de [Mito Code](#)



Responder al botón Enviar

Para poder Responder al evento onclick cuando se activa el botón Enviar, Abrir activity_main.xml de la carpeta res/layout/.

Agregar en el objeto Botón el atributo android:onClick. el atributo onClick=" sendMessage", es el nombre de un método en su actividad que el sistema llama cuando el usuario hace clic en el botón.

El resultado se ve así →

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

MainActivity.java

Agregar el método a la Clase.
Abrir MainActivity.java de la carpeta src/.

Incluir el método: ⟶

```
//* Es llamado cuando el usuario Clickea el botón Enviar */
public void sendMessage(View view) {
    // Hace algo en respuesta al botón click
}
```

Para que el sistema vincule este método con el nombre del método dado a android: onClick, la firma debe ser exactamente como se muestra.

En concreto, el método debe:

- Ser público,
- Tiene un valor de retorno vacío,
- Tiene una View con un solo parámetro (este será la View que ha hecho clic)

Construir un Intent

Una Intent es un objeto que proporciona la unión entre los componentes por separado (por ejemplo, dos actividades) en tiempo de ejecución.

Intent representa "la intención de hacer algo." Una aplicación puede usar las Intent para una amplia variedad de tareas, pero generalmente es usado para iniciar otra actividad.

Dentro del método sendMessage (), cree una Intent que inicie una actividad llamada DisplayMessageActivity:

```
Intent intent = new Intent(this, DisplayMessageActivity.class);
```

→ Ojo: Despues hay que importar la clase Intent con Ctrl + Shift + O en Eclipse.

Construir un Intent

```
Intent intent = new Intent(this, DisplayMessageActivity.class);
```

El constructor utilizado aquí tiene dos parámetros:

Un contexto como primer parámetro (esto se usa porque la clase de actividad es una subclase de Contexto)

La clase componente de la aplicación al que el sistema debe entregar la Intent (en este caso, la actividad que debe ser iniciada)

Nota: La referencia a DisplayMessageActivity generará un error si está utilizando un IDE como Eclipse, porque la clase no existe todavía. Ignorar el error por el momento; vamos a crear la clase pronto.

Construir un Intent

Un Intent no sólo le permite iniciar otra actividad, sino que puede llevar un conjunto de datos al la actividad. Dentro del método sendMessage (), utiliza findViewById () para obtener el elemento EditText y sumar su valor de texto al Intent:

```
Intent intent = new Intent(this, DisplayMessageActivity.class);
EditText editText = (EditText) findViewById(R.id.edit_message);
String message = editText.getText().toString();
intent.putExtra(EXTRA_MESSAGE, message);
startActivity(intent);
```

Nota: Ahora se necesita importar la sentencia android.widget.EditText. Ahora vamos a definir una constante EXTRA_MESSAGE.

Construir un Intent

Un Intent puede llevar una colección de varios tipos de datos como pares de valores que son llamados “Extra”(Key Values) valores claves.

El método putExtra () toma del primer parámetro el “Key name” (nombre a visualizar) y del segundo parámetro el valor.

Para que la siguiente actividad pueda examinar los datos “Extra”, debe definir el nombre de la Key utilizando una constante pública.

Así que vamos a agregar la definición de la Key EXTRA_MESSAGE en la parte superior de la clase MainActivity:

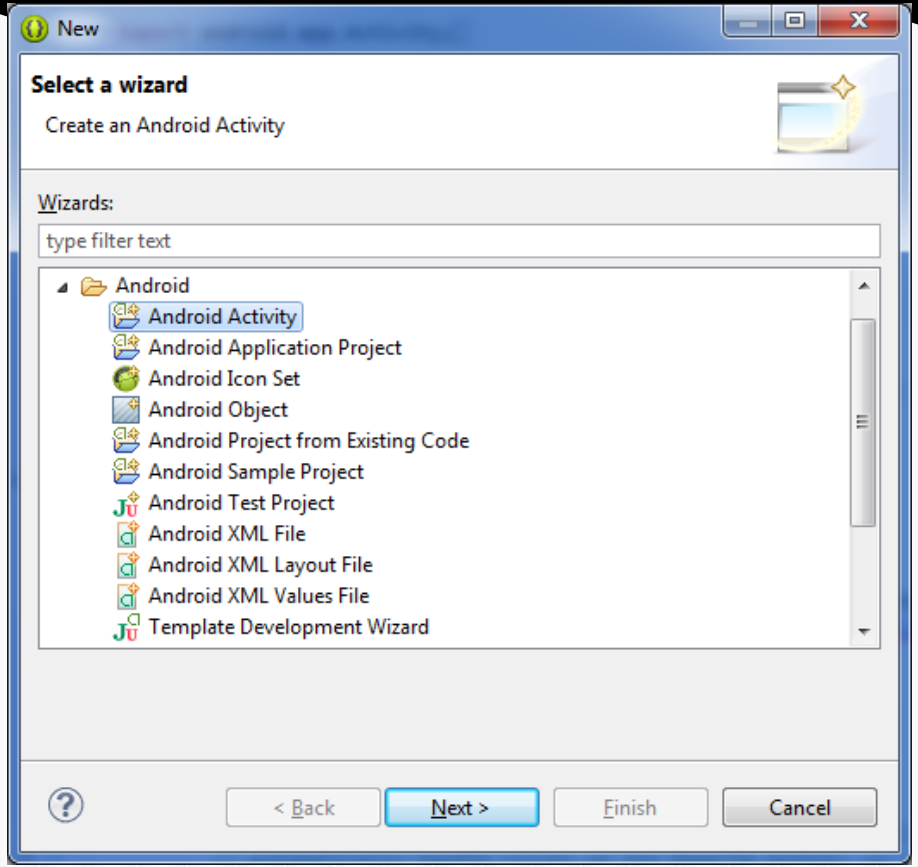
```
public class MainActivity extends ActionBarActivity {

    public final static String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";

    ...
}
```

El namespace que se usa es el del proyecto para que sea único el nombre del Key Extra.

Crear la Segunda Activity

Para crear una nueva actividad utilizando Eclipse:

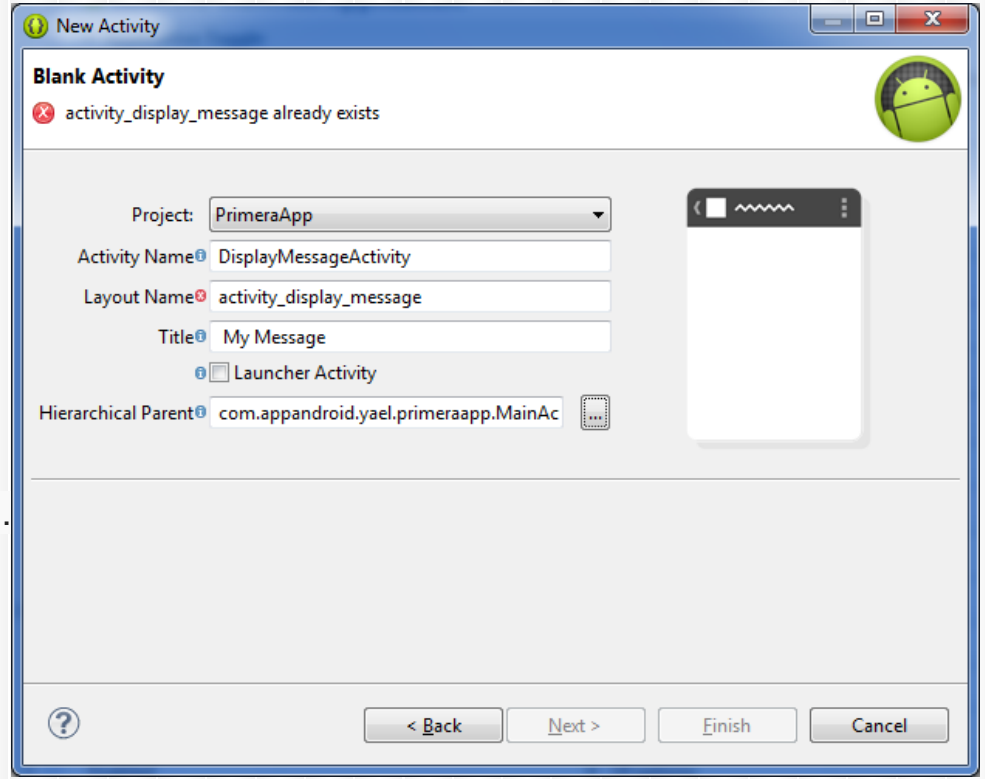
1. Haga clic en **Nuevo** en la barra de herramientas.
2. En la ventana que aparece, abra la **carpeta Android** y seleccione **Actividad Android**. Haga clic en Siguiente.



Crear la Segunda Activity

3. Seleccione **BlankActivity** y haga clic en **Siguiente**.
4. Rellene los detalles de la actividad:
   - **Proyecto**: MyFirstApp
   - **Nombre de la actividad**: DisplayMessageActivity
   - **Nombre Layout**: activity_display_message
   - **Título:** My Message
   - **Tipo de Navegación:** Ninguno
   - **Padres jerárquico:** com.example.myfirstapp.MainActivity
     (namespace del MainActivity definido)

5. Haga clic en **Finalizar**.



Crear la Segunda Activity

Agregar el String del Mensaje: Cuando usamos Eclipse automáticamente al crear la Actividad nueva agregar las cadenas que sean necesarias.
Podemos verificar que el valor que se ingresó en el campo “title_activity_display_message” se asigna “My Message”, se ha agregado al archivo de strings correspondiente:

```
<resources>
    …
        <string name="app_name">PrimeraApp</string>
        <string name="hello_world">Hello world!</string>
        <string name="action_settings">Settings</string>
        <string name="edit_message">Introduzca un mensaje</string>
        <string name="button_send">Enviar</string>
        <string name="title_activity_display_message">My Message</string>
</resources>
```

Crear la Segunda Activity

AndroidManifest.xml contenga la nueva actividad: Cuando usamos Eclipse automáticamente al crear la Actividad nueva agrega la entradas correspondientes en el archivo.
Podemos verificar que el valor que se ingresó dentro del tag <application> una nueva <activity> con los valores que ingresamos.

```
<application ... >
    ...
    <activity
        android:name=".DisplayMessageActivity"
        android:label="@string/title_activity_display_message"
        android:parentActivityName=".MainActivity" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.appandroid.yael.primeraapp.MainActivity" />
    </activity>
</application>
```

Recibir el Intent

Si se ejecuta ahora la aplicación no pasa mucho.
Al hacer clic en el botón Enviar se inicia la segunda actividad pero utiliza por defecto "Hola mundo".

Vamos a actualizar la actividad para mostrar en la View el texto Ingresado.

Recibir el Intent

Cada Activity es invocada por un Intent. Se puede obtener la Intent que comenzó su actividad llamando getIntent () y recuperar los datos contenidos en ella.

En el método **onCreate()** de la clase **DisplayMessageActivity**, recibe el Intent y extre el mensaje entregado por MainActivity:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Get the message from the intent
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.
EXTRA_MESSAGE);

    // Create the text view
    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(message);

    // Set the text view as the activity layout
    setContentView(textView);
}
```

Recibir el Intent

Dentro de **DisplayMessageActivity** se asignan los valores a la View de la segunda Activity:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Get the message from the intent
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.
EXTRA_MESSAGE);

    // Create the text view
    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(message);

    // Set the text view as the activity layout
    setContentView(textView);
}
```

Funcionamiento Final

Ahora se puede ejecutar la aplicación y ver lo que sucede cuando ingresamos el texto y presionamos el botón en la primer actividad, llama a la segunda enviando la cadena.