Anexo IV – Documentación Técnica de Programación

**Lista de cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** | **Autor** |
| 1.1 | 11/10/2011 | Esqueleto inicial | Romina Liuzzi |

Tabla de contenidos

[Introducción 5](#_Toc306139286)

[Documentación de las bibliotecas 6](#_Toc306139287)

[Código fuente 7](#_Toc306139288)

[Manual del programador 8](#_Toc306139289)

[Pruebas unitarias 9](#_Toc306139290)

# Introducción

Este documento suele asociar al código fuente de la aplicación, aunque puede llegar a ser más amplio dependiendo del proyecto.

# Documentación de las bibliotecas

# Código fuente

# Manual del programador

Para el desarrollo y testeo de la parte móvil fue preciso contar con una versión de Eclipse soportada tanto por el SDK de Android como por el respectivo ADT (plugin) de Android para Eclipse.

La distribución de Eclipse utilizada para este proyecto fue:

Eclipse IDE for Java Developers

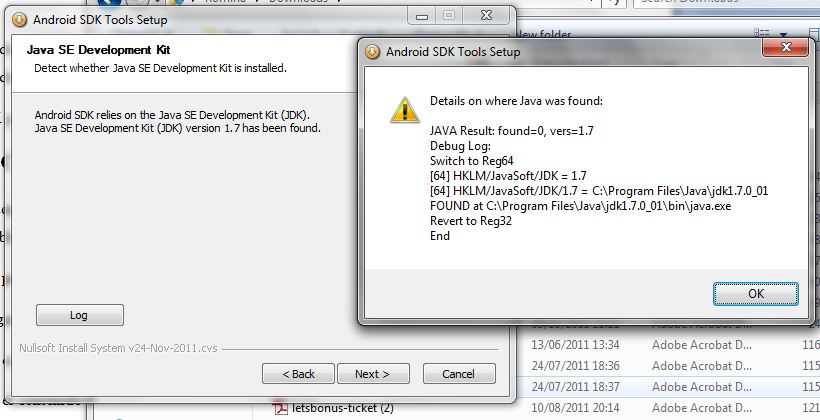
Version: Indigo Service Release 1

Plataforma de Eclipse: 3.7.1.r37

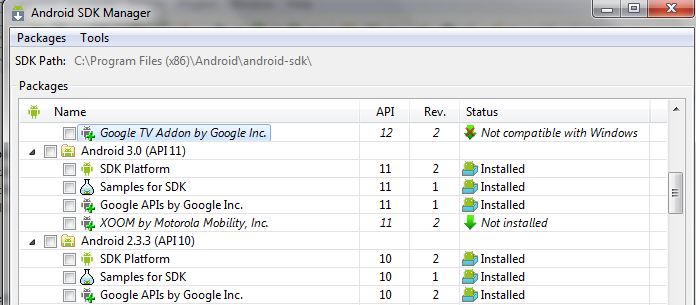
Se puede descargar directamente de la página oficial de [Eclipse](http://www.eclipse.org/downloads/), se incluye el fichero comprimido en formato .zip para Windows de 64-bits. Su instalación consiste en descomprimir el fichero y a abrir la aplicación.

Por otra parte el SDK de Android permite disponer de varias plataformas de desarrollo, es posible descargar el ejecutable desde el [Android Developers Site](http://developer.android.com/sdk/index.html), se incluye una versión para Windows del ejecutable.

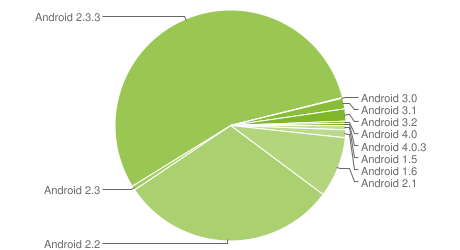
Es preciso tener instalada una versión del JDK compatible con la del SDK.



El SDK de Android nos permite seleccionar las plataformas de Android que nos interesa incorporar.



Tomando en cuenta la tendencia actual, se decide trabajar sobre la distribución Gingerbread (Android 2.3.3/Android2.3.7).



Finalmente se procede a instalar el plugin de eclipse:

Desde Eclipse navegar a: Help > Install New Software.

Agregar repositorio con los valores:

ADT Plugin

https://dl-ssl.google.com/android/eclipse/

**TOMCAT:**

Desde tomcat.apache.org acceder a la sección de “Downloads” y descargar la última versión disponible.

En este proyecto hemos utilizado Tomcat 7.0.23.

Descargar el core correspondiente.

En este caso 64-bit Windows.zip.

Extraer el contenido del fichero comprimido en un directorio fácilmente accesible, es recomendable que el nombre incluya la versión desplegada (por ejemplo: apache-tomcat-7.0.23).

Asegurarse de que se cuenta con una versión de jre o jdk superior a 1.6. Comprobar que las variables de entorno JAVA\_HOME y JRE\_HOME están definidas y apuntan al directorio de instalación del JDK, y JRE respectivamente.

Abrir una ventana de comandos y ejecutar startup.bat.

C:\Users\Romina\Desktop\apache-tomcat-7.0.23\bin>startup.bat

Using CATALINA\_BASE: "C:\Users\Romina\Desktop\apache-tomcat-7.0.23"

Using CATALINA\_HOME: "C:\Users\Romina\Desktop\apache-tomcat-7.0.23"

Using CATALINA\_TMPDIR: "C:\Users\Romina\Desktop\apache-tomcat-7.0.23\temp"

Using JRE\_HOME: "C:\Program Files\Java\jre7"

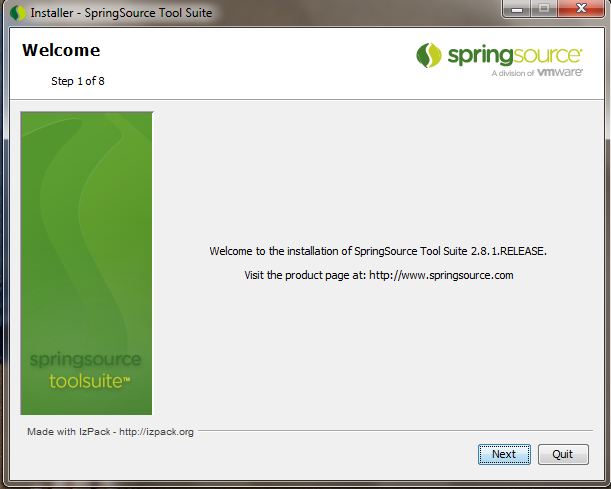
Using CLASSPATH: "C:\Users\Romina\Desktop\apache-tomcat-7.0.23\bin\bootstr

ap.jar;C:\Users\Romina\Desktop\apache-tomcat-7.0.23\bin\tomcat-juli.jar"

SPRINGSOURCE

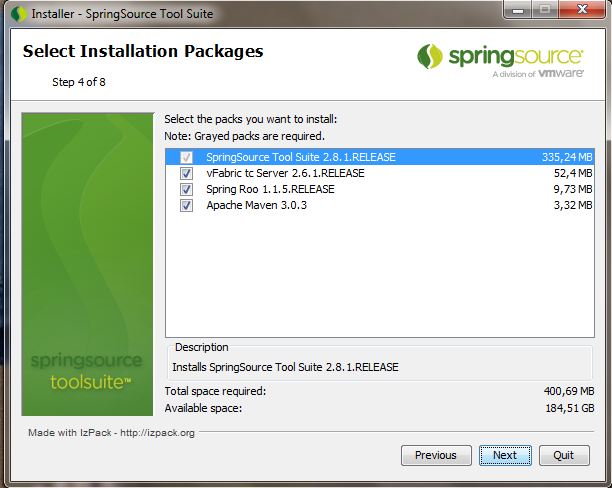
Es preciso instalar Eclipse SpringSource Tool Suite y configurar un server para conectarse a tomcat.

1. Descar el ejecutable para la versión del sistema operativo correspondiente (en este caso Windows-64bits) y seguir los pasos de instalación tal como se describe a continuación.



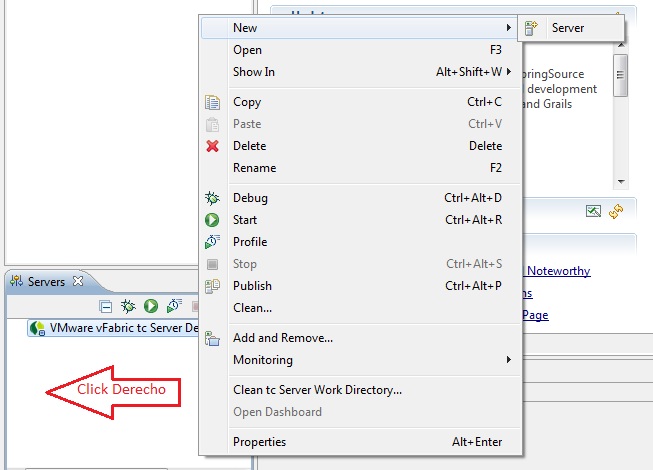


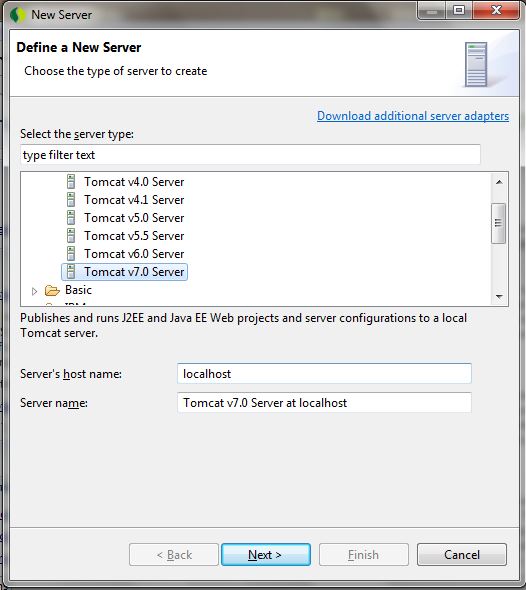


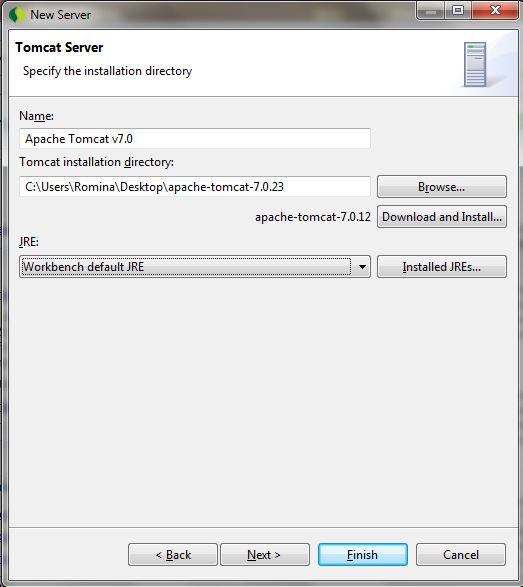


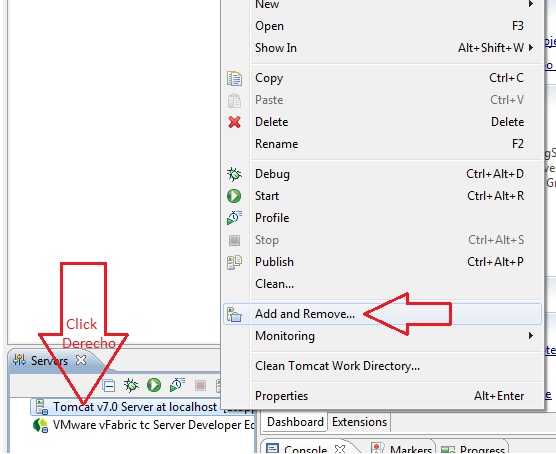


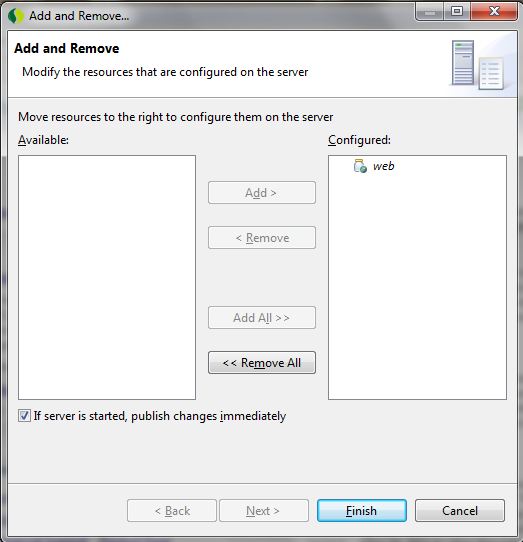
1. Definir un directorio para alojar el espacio de trabajo.
2. Crear un nuevo server para tomcat siguiendo los pasos detallados a continuación.

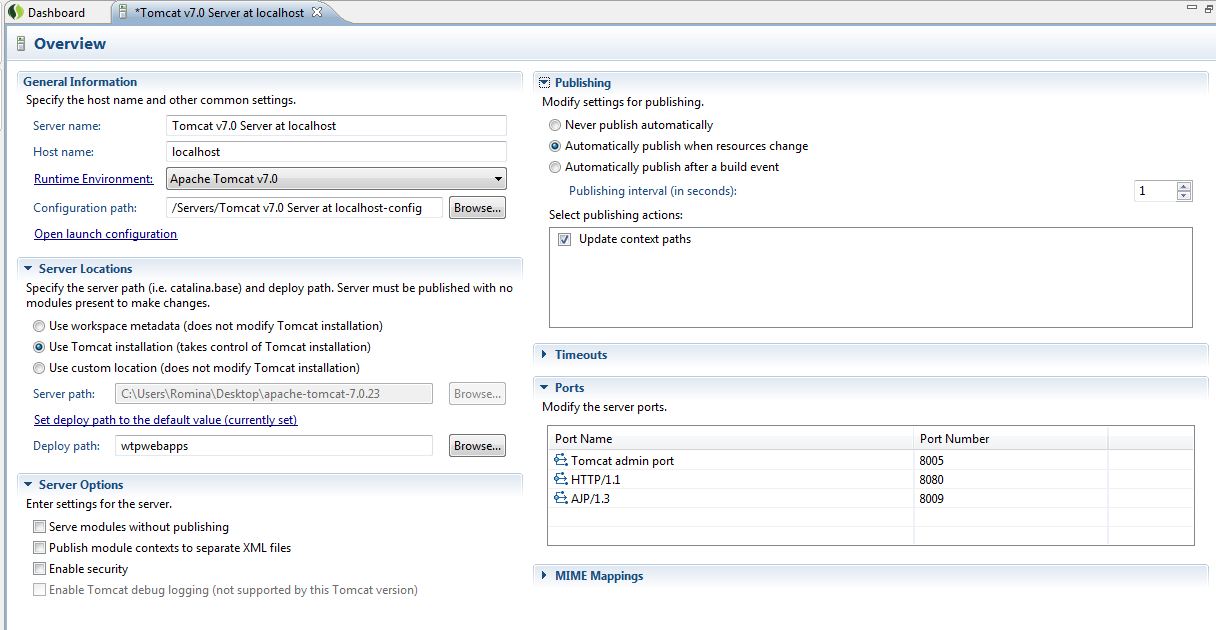












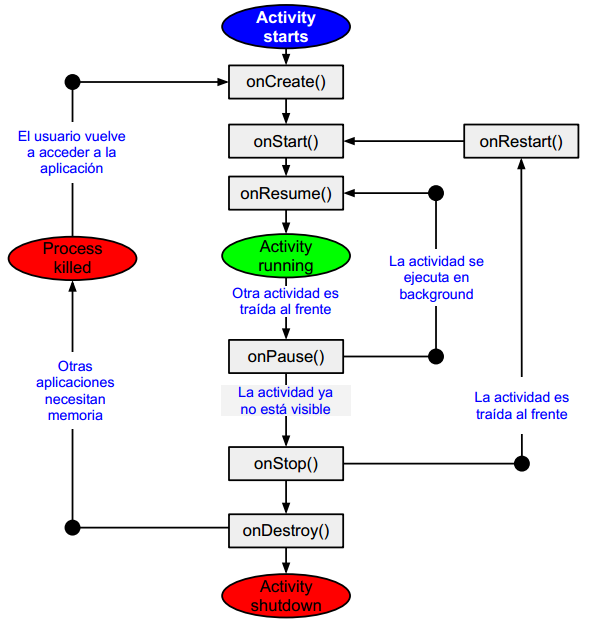
Ciclo de vida de una aplicación Android

Figura 3. Ciclo de vida de una actividad

El esquema anterior representa todos los posibles estados que por los que puede pasar una actividad durante su vida. Estos métodos están implementados en la clase Activity y son heredados en cada una de nuestras actividades de forma transparente. De forma que, en la mayoría de los casos, no necesitamos sobrecargar el método del constructor.

Un caso aparte, es el método onCreate() que debemos implementar siempre; es llamado cuando se inicia la actividad.

Dentro de éste diagrama de flujo es importante notar que existen tres posibles estados en los que se puede “terminar/iniciar” una actividad.

onPause() 🡪 onResume()

onStop() 🡪 onRestart() 🡪 onStart()

onDestroy() 🡪 onCreate()

Estructura de un proyecto Android

En este apartado se explica la estructura de ficheros de un proyecto Android, sus drectorios y ficheros más relevantes.

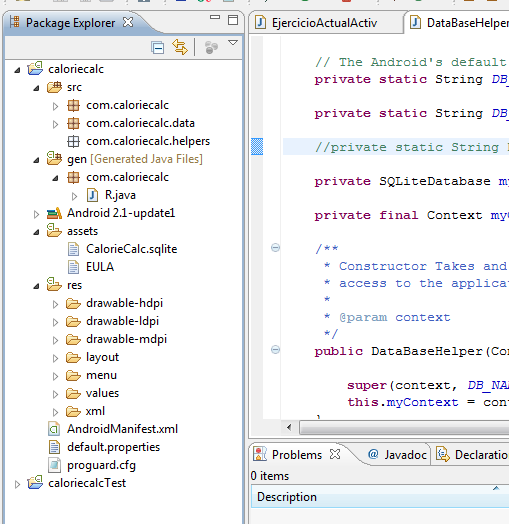


Figura 3. Estructura Proyecto Android

src/

Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Inicialmente, Eclipse creará por nosotros el código básico de la pantalla (Activity) principal de la aplicación, siempre bajo la estructura del paquete java definido.

gen/

Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente en Java dirigidos al control de los recursos de la aplicación.

assets/

Contiene todos los demás ficheros auxiliares necesarios para la aplicación (y que se incluirán en su propio paquete), como por ejemplo ficheros de configuración, de datos, etc.

*NOTA: La diferencia entre los recursos incluidos en la carpeta /res/raw/ y los incluidos en la carpeta /assets/es que para los primeros se generará un ID en la clase R y se deberá acceder a ellos con los diferentes métodos de acceso a recursos. Para los segundos sin embargo no se generarán ID y se podrá acceder a ellos por su ruta como a cualquier otro fichero del sistema. Usaremos uno u otro según las necesidades de nuestra aplicación.*

res/

Contiente todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos de deberán distribuir entre las siguientes carpetas:

res/drawable/

Contienen las imágenes de la aplicación. Se puede dividir en /drawable-ldpi, /drawable-mdpi y /drawable-hdpi para utilizar diferentes recursos dependiendo de la resolución del dispositivo.

res/layout/

Contienen los ficheros de definición de las diferentes pantallas de la interfaz gráfica. Se puede dividir en /layout y /layout-land para definir distintos layouts dependiendo de la orientación del dispositivo.

res/anim/

Contiene la definición de las animaciones utilizadas por la aplicación.

res/menu/

Contiene la definición de los menús de la aplicación.

res/values/

Contiene otros recursos de la aplicación como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), etc.

res/xml/

Contiene los ficheros XML utilizados por la aplicación.

res/raw/

Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

AndroidManifest.xml

Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono, …), sus componentes (pantallas, mensajes, …), o los permisos necesarios para su ejecución.

R.java

Esta clase R contendrá en todo momento una serie de constantes con los ID de todos los recursos de la aplicación incluidos en la carpeta /res/, de forma que podamos acceder fácilmente a ellos.

# Pruebas unitarias

Pruebas de stress sobre la aplicación:

<http://developer.android.com/guide/developing/tools/monkey.html>