Guía de estilo de la aplicación para Android (version1.0)

Francisco Javier Fernández Toro por GeoRemindMe.

1 de abril de 2011

Índice

1. Introducción y objetivos.

Tras los últimos cambios hacia la creación de una pequeña empresa para la gestión, mantenimiento y evolución del software GeoRemindMe, conjuntamente con la comunidad; y la urgente necesidad de la deficinición de una arquitectura de aplicación para que facilite la implementación, testeado y futura ampliación del sistema, he decidido crear una guía de estilo para evitar demasiadas refactorizaciones de código en un futuro y que cualquier programador pueda participar en el proyecto lo antes posible sin que la curva de aprendizaje se incremente.

Este documento no pretende ser un contrato para permitir la colaboración de programadores, si no que pretende ser un documento de referencia al que recurrir en caso de dudas, refactorizaciones de código y argumentación de posibles conflictos. Ni que decir tiene que este es un documento 'vivo', es un documento que ha de cambiar y de modificarse para que se adapte a los cambios que se produzcan y poder tener una guía usable y contemporánea.

2. Estilo general de programación y organización de las fuentes.

Como estrutura básica para el estilado y estructuración de nuestro código se va a usar como referencia el documento Java Code Convention en el cual se indica como nombrar por convenio clases, interfaces, variables, etc. a fin de conseguir un código legible y ordenado que permita su difusión, mantenimiento y desarrollo con poco coste en tiempo, y en algunos casos, dinero.

A la hora de elegir un nombre para los distintos elementos que componen nuestro sistema tomaremos un nombre que describa claramente el cometido de dicha entidad, ya sea una clase, un objeto, una variable o una constante. Debido al proceso de 'pseudo-programación extrema' que vamos a llevar a cabo, la refactorización del código es algo importante y necesario, pero una buena elección del nombre que hace referencia a un ítem, puede ahorrarnos un tiempo muy valioso en algunos casos.

3. Documentación del código fuente.

Somos la comunidad. Necesitamos usar código escrito por distintas personas, con disitintas formas de pensar y distintas formas de codificar, además de añadir el problema de que estamos en lugares distintos con horarios diferentes y en el que la comunicación por medios tradicionales no puede ser llevada a cabo tal y como nos gustaría. El teléfono no está siempre disponible, el correo electrónico es lento y usar un sistema de mensajería instantánea no es viable. Solo nos queda codificar de forma clara y documentar. Si, sé que es un proceso tedioso pero los entornos de programación de hoy en día nos proporcionan templates y recursos que casi automatizan el hecho de documentar nuestro código para más adelante generar una documentación sencilla pero no por esos menos eficaz.

Tras hacer una extensa búsqueda en Internet he llegado a la conclusión de que esta es la mejor forma de documentar el código:

1. Al principio de cada clase (usando Javadoc).
2. Al principio de cada método (usando Javadoc).
3. Ante cada variable de clase (usando Javadoc).
4. Al principio de un fragmento de código no evidente (con una línea).
5. A lo largo de bucles (con una línea).
6. Siempre que el código no sea evidente o tenga un comportamiento poco usual (con una línea).

Debido a que vamos a usar JAVA que ya nos proporciona la herramienta Javadoc, esta va a ser la elegida para documentar nuestro código y además porque el entorno de programación Eclipse que es el usado para el desarrollo de Android integra de forma automática nuestra documentación en los pop-ups que nos muestra con ayuda y sugerencias. Para esto, hagamos un pequeño refresco de como usar Javadoc:

- Comentarios Javadoc. Este tipo de comentarios son los que Javadoc interpreta como suyos y los parsea como información útil. Su estilo es el siguiente:

```
/**
 *
 * Descripción del elemento.
 * Una frase o varios párrafos sin demasiada extensión.
 *
 * @tag Etiqueta que Javadoc entiende y parsea.
 */
```

- Como etiquetas a usar con Javadoc en la declaración de clases e interfaces tenemos las siguientes:

```
@author Indica el autor del código.
@version Version y fecha de la revisión.
@see Referencias a otras clases y métodos.
```

- Las etiquetas que se usan para la documentación de métodos y constructores son estas:

```
@param Una por cada argumento que recibe.
@return El valor de retorno en caso de que no sea void.
@exception o @throws Una por tipo de excepción que pueda lanzar
(exception y throws se usan indistintamente).
```

Como documentación extra se puede visitar el siguiente en documento How to write Doc comments for the Javadoc Tool que indica con mucho más detalle como usar Javadoc.

4. Testeo del código antes de compartirlo.

El principal objetivo del código es funcionar bien, y esto no siempre es así. Por eso debemos de testear nuestro código para asegurarnos de que cuando llegue a manos de un cliente, el software tenga la calidad que el cliente espera del mismo. Esta es una parte básica e imprescindible del proceso de desarrollo de software que vamos a seguir, ya que nos vamos a apoyar en diferentes metodologías ágiles y estas van estrechamente ligadas al *TDD (Test Driven Development)*. Sé que esta sección del documento puede sonar un poco absurda, pero es absolutamente necesario que probemos nuestro código y compartamos las pruebas realizadas. Me explico, el simple hecho de probar ciertas funcionalidades de una clase nueva, no nos garantiza que todo funcione o funcione para todo el conjunto de datos para el que fué diseñada; por eso es necesario compartir los casos de prueba realizados con JUnit como una parte más del paquete de software que estamos desarrollando.

Para obtener información de como usar JUnit en Android recomiendo mirar en primer lugar los siguientes links: Hello, Testing para ver un ejemplo muy simple de como usarlo; y Testing Fundamentals para comprender como funciona el método de testeo en Android.

Como he dicho anteriormente, se va a usar JUnit que está perfectamente integrado en JAVA y en el caso de Android, se ha adaptado un poco para que su manejo sobre elementos de la plataforma como actividades, servicios, etcétera sea más intuitivo. Si quieres saber un poco más sobre JUnit te recomiendo visitar el sitio web JUnit.orgque incluye gran cantidad de información de como usarlo.

5. Internacionalización del código.

Una de las características de Android que más me gusta es la gran potencia que tiene para permitir la internacionalización de la aplicación y la adaptación a gran multitud de dispositivos dispares entre sí sin que para el programador se convierta en una odisea. En esta sección voy a centrarme en principio solo en la internacionalización del texto para que el equipo de traducción pueda ir traduciendo el software incluso antes de que este esté terminado. La forma en al que Android maneja esto es mediante ficheros XML que contienen el texto que se usa en la aplicación y depende de la ubicación de este fichero dentro de la jerarquía de directorios dentro de la carpeta *res* corresponde a un idioma o a otro.

Todas nuestras cadenas de caracteres van a ir dentro de este fichero y como nombre para identificarlas desde el código se va a poner su nombre en inglés. Así de simple. He probado diferentes métodos pero ese es el más intuitivo, el más fácil de recordar y el que puede hacer que se manteja y se traduzca más rápido y mejor.

6. Mantén vivo este documento.

Ya como último elemento de este texto va una petición personal. Quiero que este texto cambie, que me digaís como de malo es y se adapte al equipo de trabajo ya que el único objetivo de todo esto es que podamos trabajar juntos sin tener que estudiar el código de una persona día tras día antes de poder medio entenderlo. Cualquier sugerencia, error y cosas así, un simple email a la siguiente dirección fran at georemindme dot com e intentaré reflejar los cambios lo antes posible.