Desarrollo de aplicaciones informáticas en entornos inalámbricos. Lenguaje Java 2 Micro Edition

Por Alberto José Leira Rejas y José Luis Calvo Rolle

Departamento de Ingeniería Industrial. Universidad de A Coruña

Sin lugar a dudas, uno de los más grandes fenómenos tecnosociales, de finales del siglo XX, ha sido el desarrollo de la telefonía móvil. Tan grande ha sido ese auge, que incluso las personas, empresas e instituciones ligadas a dicho sector, se vieron desbordadas ante la magnitud del evento.

Java 2 Micro Edition (desde ahora J2ME), se ha convertido en un paradigma de desarrollo de aplicaciones inalámbricas, de configuración hardware media-baja y para entornos de computación ubicua.

La interrelación entre las comunicaciones y la computación, de la que Internet es el mejor reflejo, se ha trasladado al ámbito de la telefonía móvil, y otros entornos inalámbricos, tales como PDAS, buscapersonas, etc. La explicación a este boom, puede resumirse en tres apartados :

a) El desarrollo espectacular, de la electrónica integrada, tanto analógica como digital, que permite por poco precio, disponer de circuitería de alto nivel en poco espacio.

b) La cultura informática cada vez más extendida en nuestra sociedad, posibilita el acceso del gran público, a todas las plataformas software.

c) El gran interés por la cultura del ocio y el entretenimiento, especialmente entre los más jóvenes, constituye un amplio segmento del mercado de la telefonía móvil.

Aspectos básicos sobre J2ME

El gráfico 1 muestra los niveles del lenguaje Java.

El J2ME es un entorno ciertamente limitado, pero con una API conceptualmente muy similar a J2SE. Está orientado a sistemas con una capacidad hardware limitada, escasos dispositivos periféricos y enfocados a la conexión. Es decir, en cierta medida nos remonta a los orígenes de Java, orientado a sistemas electrónicos de consumo (microondas, magnetoscopios, etc).

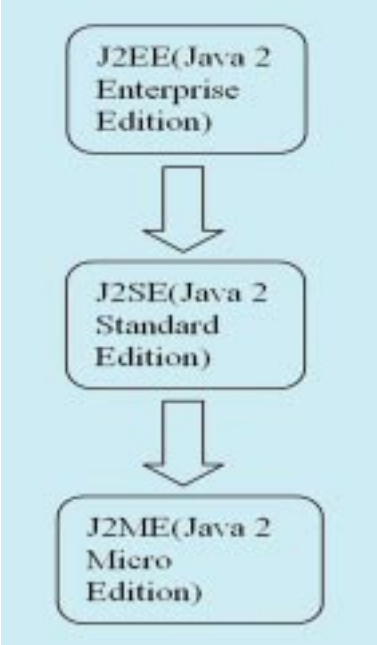

Figura 1. Midlets

Una *configuración* es un conjunto software que engloba las funcionalidades precisas para el desarrollo de aplicaciones en dispositivos similares. Incorpora una máquina virtual Java reducida, un subconjunto de clases y una família de APIS propias. La configuración base de J2ME es la conocida como Configuración de Dispositivo Conectado Limitado (CLDC), para dispositivos con memoria de 160Kb, micro de 16 bits y conexión a 9600 bps.

Un *perfil* es una restricción, mayor que la configuración, ligada a las características particulares de una familia concreta de dispositivos con los pros y los contras que ello conlleva.

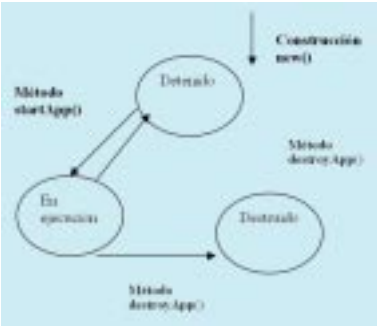

Figura 2. Ciclo de vida de un midlet

La tipología de las aplicaciones Java es muy concreta, (applets, servlets, beans, soporte JSP, etc), y se escapa del objetivo del presente artículo. Aquí destacaremos los llamados midlets (ver fig. 2), por ser propios de J2ME. Los podemos definir como programas, sujetos al perfil MIDP.

Cada dispositivo móvil, compatible con MIDP, cuenta con el llamado Gestor de Aplicaciones, capaz de concretar el ciclo de ejecución del midlet (carga, ejecución y borrado).

startApp(). Se ejecuta al inicializarse o tras detenerse el midlet. En él se implementa el código preciso cuando se inicia o reinicializa el citado midlet.

pauseApp(). Se ejecuta justo antes de detenerse el midlet. En él se implementa código que salva valores, guarda datos para restaurar, etc.

destroyApp(). Se ejecuta antes de destruirse el midlet. En él se implementa el código preciso para liberar recursos.

Diseño de interfaces

Es obvio, que en cualquier aplicación, el diseño de interfaces de usuario, es algo determinante, en tanto que predetermina la amistosidad con el sistema.

En el caso que nos compete, no debemos olvidar, que después de todo, un usuario de telefonía, no tiene por que tener formación informática a nivel de usuario, y así, el éxito de la aplicación depende tanto, de la verificación de los requisitos de la aplicación, como de la facilidad de uso, con unos medios de E/S, muy limitados en la mayor parte de los casos.

Las interfaces pueden construirse a través de una API de alto nivel, o bien de una API de bajo nivel. En primer caso, nos encontramos con la ventaja de tener una interface versátil y multiplataforma, en detrimento de la calidad del segundo caso.

Clases implicadas en la creación de interfaces

La nomenclatura de los métodos, sigue la convención empleada en Java (ver figura 3).

Display	Gestor de recursos gráficos
Displayable	Pantalla genérica de usuario
Screen	Pantalla genérica Api alto nivel
Canvas	Pantalla genérica Api bajo nivel
Graphics	Herramientas de dibujo Canvas
TextBox	Caja de texto editable
List	Lista desplegable
Alert	Pantalla de advertencia
Form	Ventana de formulario
ChoiceGroup	Similar a List, pero insertas en un formulario
Item	Elemento independiente de un formulario
DateField	Elemento de formulario de manejo de fechas
TextField	Elemento de formulario de manejo de textos
Gauge	Elemento de formulario en formato de diagrama de barras
StringItem	Elemento de formulario de manejo de cadena de textos
ImageItem	Elemento de formulario de manejo de imágenes
Choice	Interfaz de clases para selección de opciones
Ticker	Marquesina
Command	Acción realizada por el usuario

Así, para leer un atributo, el método utiliza el prefijo get. Para modificar el atributo, el método utiliza el prefijo set. Por ejemplo, para añadir un elemento, se utiliza el prefifo add, etc. Recordemos, que en Java, los nombres de método comienzan por minúsculas, y cuando constan de más de una palabra, cada una de ellas, a excepción de la primera, comienzan por mayúscula.

API de bajo nivel

La API de bajo nivel, aunque es la más costosa de manejar por el programador, presenta las mejores posibilidades para la construcción de aplicaciones puramente multimedia (p.e. juegos), ya que :

- Permite utilizar al máximo las posibilidades gráficas del dispositivo en concreto. El elemento gráfico por excelencia es el Canvas (véase el repaso sobre Java), un lienzo, sobre el que se dibujan las primitivas correspondientes.
- Es muy potente, a la hora de manejar dispositivos de entrada/salida, y no se constriñe a elementos genéricos. Así se puede gestionar todo el teclado, pantallas táctiles, apuntadores, etc.
- La flexibilidad, dentro de dispositivos de la misma familia es extraordinaria. Ello es gracias a las ventajas de la abstracción de la clase Canvas.
- Es fácil de manejar para los programadores acostumbrados a crear applets.

Gestión de datos con RMS

Nos vamos a encontrar con la necesidad de almacenar permanentemente, en memoria conjuntos de datos, estructurados tabularmente como registros. La herramienta API de J2ME, es la la llamada RMS (*Record Management System).*

Como tres características principales de RMS, destacaríamos :

- Persistencia de los datos entre sesiones y eventos.
- Consistencia del modelo de datos asociado.
- Capacidad para operaciones de E/S (inserción,borrado, etc).

Temporizadores

Los temporizadores, son un recurso fundamental, para el desarrollo de sistemas informáticos, especialmente en gestión de eventos, funciones de bajo nivel y la programación concurrente.

En J2ME, las clases Timer, y TimerTask permiten fijar dos tipos de temporizaciones:

- Eventos que ocurren una sóla vez.
- Eventos que ocurren más veces. Se subdividen en:

o Tareas que se disparan, cuando ha transcurrido el tiempo pasado y las condiciones del sistema lo permitan. Ejemplo: la carga de imágenes para un juego.

o Tareas que se disparan, transcurrido el tiempo citado, independientemente de las condiciones del sistema. Ejemplo: una alarma.

En la segunda clase, los temporizadores pueden contar, en intervalos de tiempo absolutos (*t* milisegundos entre tareas) o a horas determinadas (p.e lanzar un despertador a las 08.00).

Api Multimedia

J2ME proporciona un soporte multimedia básico, que es de esperar vaya in crescendo a medida que surjan las nuevas revisiones de los MIDP y CDLC. Actualmente, las capacidades multimedia de J2ME (sin incluir el manejo de imágenes, similar a J2SE) serían:

- Generación de tonos
- Reproducción de archivos de medios: MPEG, MIDI y WAV

El API MMA incorpora los paquetes media, control y protocol. El primero, permite acceder a las clases e interfaces que permiten reproducir audio y video., encargándose el paquete control de aquellas funciones complementarias de reproducción ligadas al formato.

En el paquete media se encuentra la clase Manager, que permite crear reproductores, generar tonos y gestionar información próxima al sistema.

El Manager, es un intermediario entre la aplicación, que demanda la creación de un reproductor de medios, y éste.

Redes y servicios inalámbricos

Es obvio que un teléfono móvil permite la comunicación inalámbrica. Por tanto, no podemos olvidar la importancia, que tienen en J2ME, aquellas funciones orientadas a la comunicación sin hilos.

a) La Configuración CLDC (Connected Limited Device Configuration) incor-

pora la API GCF (Generic Connection Framework).
b) La filosofía del GCF se sustenta en la idea de la variedad de dispositivos inalámbricos y sus protocolos asociados que se pueden soportar (TCP/IP, i-mode, WAP, Bluetooth, etc),
c) La interoperabilidad entre protocolos, es algo transparente para el programador a través de la clase Connector.

La clase Connector

Perteneciente al GCF, tenemos la clase Connector, cuyo método *open*, permite establecer una conexión (ver fig. 4).

Figura 4. Establecimiento de una conexión

Connector (protocolo: url o dirección, parámetros).	Protocolo : http (protocolo http) file (archivo local) socket (socket) comm (puerto serie) datagram (datagrama)
s	READ (lectura) WRITE(escritura) READ_WRITE(lectura-escritura).
int timeout	Tiempo en milisegundos que espera por la conexión. Si se desborda se lanza una excepción InterruptedIOException.

Interfaz HttpConnection

Http es un protocolo orientado al acceso a páginas Web. Es de tipo TCP/IP y cliente/servidor, con un mecanismo simple de petición respuesta en las transacciones.

El mecanismo de funcionamiento del cliente, se subdivide en tres fases:

- Petición (establecimiento de conexión).
  - oTipo de petición (GET, POST y HEAD).
    - GET (Los datos van como parte de la URL).
    - POST (Los datos van en un stream separado).
  - o Cabecera.
  - o Cuerpo.
- Intercambio de información.
- Cierre de conexión.

La respuesta del servidor se estructura de la manera siguiente:

- Línea de estado. Marca el resultado de la petición.
- Cabecera. Incluye información sobre los datos enviados al cliente.
- Cuerpo. Datos propiamente dichos.

La interfaz HttpConnection, se relaciona con el protocolo http, por medio de un casting y el método open. Los estados se corresponden con las fases ya mencionadas.

¿Por qué http? Es un protocolo universal, fiable y orientado a Internet. Internet es un punto objetivo de los teléfonos móviles, ya sea vía WAP y WML, o bien por medio de UMTS.

No debemos tampoco olvidarnos de la tecnología sin hilos Bluetooth, ni de la mensajería SMS o MMS. Todas éstas tecnologías tienen su soporte J2ME,

J2ME Wireless Toolkit

Es una extensión de J2ME de Sun, de libre distribución, que puede descargarse desde la citada dirección www.sun.com (ver fig. 5 y 6). Nos exige tener instalado, java 2 en la versión 1.4.0 o posteriores.

Incorpora un compilador + emulador, con seis dispositivos básicos (cuatro teléfonos + un dispositivo palm + un dispositivo qwerty).

El autor ha escrito todos los ejemplos con un editor ASCII (el bloc de notas), y los ha compilado y simulado con el citado programa, obteniendo unos resultados satisfactorios.

Este entorno, llama proyecto al midlet o conjunto de midlets que constituyen la aplicación (lo que también se suele llamar suite). Las opciones son lo suficientemente explícitas: New Project, Open Project.

Build permite la compilación, y Run la ejecución sobre un dispositivo concreto. Settings permite asignar parámetros sobre configuración y ubicación de directorios.

Creación de un proyecto usando J2ME Wireless Toolkit

En la opción New Project especificamos el nombre del proyecto y del midlet principal. Tras eso, se genera un fichero explicativo que nos indican la estructura de directorios

Figura 5. J2ME Wireless Toolkit

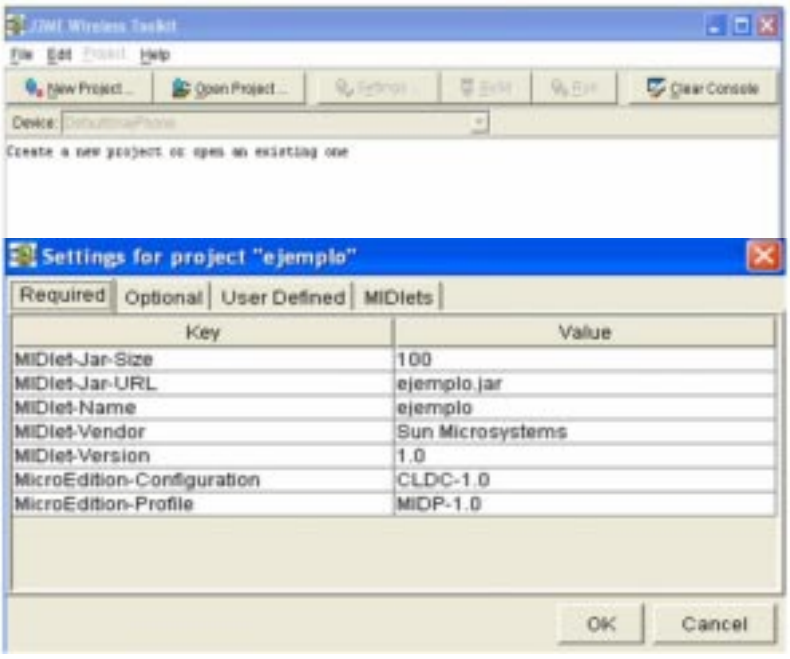

Figura 6. Settings

creada, sobre la que se ubicarán los respectivos archivos (jar, java, etc).

Es responsabilidad del programador, copiar los citados ficheros en la ubicación especificada, ya que no se efectúa de forma automática.

Una vez creado el proyecto, se ejecuta BUILD, que compilará el programa, marcando los potenciales errores. Una vez depurado se puede ejecutar con RUN (que también fuerza la compilación si aún no se ha realizado).

Distribución de un midlet o suite de midlets

Para facilitar la distribución de las aplicaciones J2ME, se utilizan los llamados ficheros de recursos JAR, propios de Java, dónde se "embuten" clases y recursos (imágenes, sonidos, etc). Previamente se crea el fichero de manifiesto, donde se añaden datos de interés (autor, versión,etc).Finalmente se añade un fichero JAD, que incorpora información crítica, para la identificación del programa (nombre, tamaño,etc).

Bibliografía

- Álvarez García & Morales Grela. J2ME. Anaya Multimedia 2002
- Cuenca Jiménez. Programación en Java. Anaya Multimedia 1997
- Leira Rejas. Aplicaciones Informáticas para Telefonía Móvil. Universidad de A Coruña. 2004. ❑