



Universidad Nacional del Nordeste
Facultad de Ciencias Exactas, Naturales y Agrimensura

Trabajo de Adscripción

Plataformas de Desarrollo de Aplicaciones para Móviles

WebSphere Studio Device Developer



Silvana Daniela Maidana - L.U.: 33.548

Materia: Teleproceso y Sistemas Distribuidos
Director: Mgter. David Luis la Red Martínez

Licenciatura en Sistemas de Información
Corrientes - Argentina

2008

Índice General

1	La Experiencia Pervasiva de IBM	1
1.1	Modelo de Programación de WebSphere	5
1.2	Alcance	5
1.3	Alcance de la Arquitectura PvC	7
1.4	Modelo de Programación Extremo a Extremo	9
1.5	WebSphere y las Aplicaciones para Dispositivos	9
1.6	La plataforma de servicios pervasivos	9
1.7	Modelo de Programación de WebSphere	11
1.8	Ejemplos de Aplicación Pervasiva	13
2	WebSphere Studio Device Developer IDE	15
2.1	Developer IDE	15
2.2	Arquitectura Base de WSDD	15
2.3	Arquitectura de Eclipse	16
2.4	WebSphere Studio Workbench	16
2.5	WebSphere Studio Device Developer	17
2.6	Trabajar con el IDE	19
	2.6.1 Barra de Herramientas	20
	2.6.2 Perspectiva, Editores y Vistas.	21
	2.6.3 Espacio de Trabajo (Workspaces)	22
2.7	Configuración J2ME	23
2.8	Construir y Ejecutar en Dispositivos	23
	2.8.1 Construcción	24
	2.8.2 Dispositivos	25
	2.8.3 Creación de Dispositivos	25
	2.8.4 Lanzar una Aplicación	25
	2.8.5 Realizar un Lanzamiento	28

Bibliografía

29

Índice de Figuras

1.1	Experiencia pervasiva de IBM	2
1.2	Computación ubicua	3
1.3	Experiencia ubicua extendida a la plataforma WebSphere	4
1.4	Alcance	6
1.5	Arquitectura PVC	7
1.6	Modelo de programación extremo a extremo	8
1.7	Paquete de la plataforma de servicios pervasivos.	10
1.8	Modelo de programación de WebSphere.	12
1.9	Ejemplo de una aplicación pervasiva.	13
2.1	WebSphere Studio Device Developer	18
2.2	Barra de herramientas	20
2.3	Constructor MIDP	24
2.4	Creación de dispositivos.	26
2.5	Lanzar una aplicación	27

Capítulo 1

La Experiencia Pervasiva de IBM

La *experiencia pervasiva* cubre múltiples ubicaciones para acceder a aplicaciones y datos a través de diferentes tipos de redes de área local, área personal, y redes de área amplia. Estas pueden ser *cableada* o *inalámbrica* (o una combinación).

El punto final puede ser cualquiera de los pilares pervasivos (*un vehículo, una casa, una máquina o una combinación*).

Las aplicaciones pueden ser comunes a todas las ubicaciones, o específicos para un subconjunto (ver fig. 1.1 de la pág. 2).

Las ubicaciones individuales tienen necesidades únicas, por lo tanto, la computación ubicua también debe acomodarse a las distintas “ubicaciones” móviles que interactúan entre sí.

En la *experiencia pervasiva*, los usuarios tienen la necesidad de acceder a sus aplicaciones desde diversos lugares y las mismas aplicaciones necesitan estar disponibles a través de diversas redes y para cualquier dispositivo (ver fig. 1.2 de la pág. 3).

El *objetivo* es el *acceso a datos y aplicaciones en cualquier momento y en cualquier lugar desde cualquier dispositivo*.

Esto es lo que significa pervasivo: *acceso, interacción, integración*.

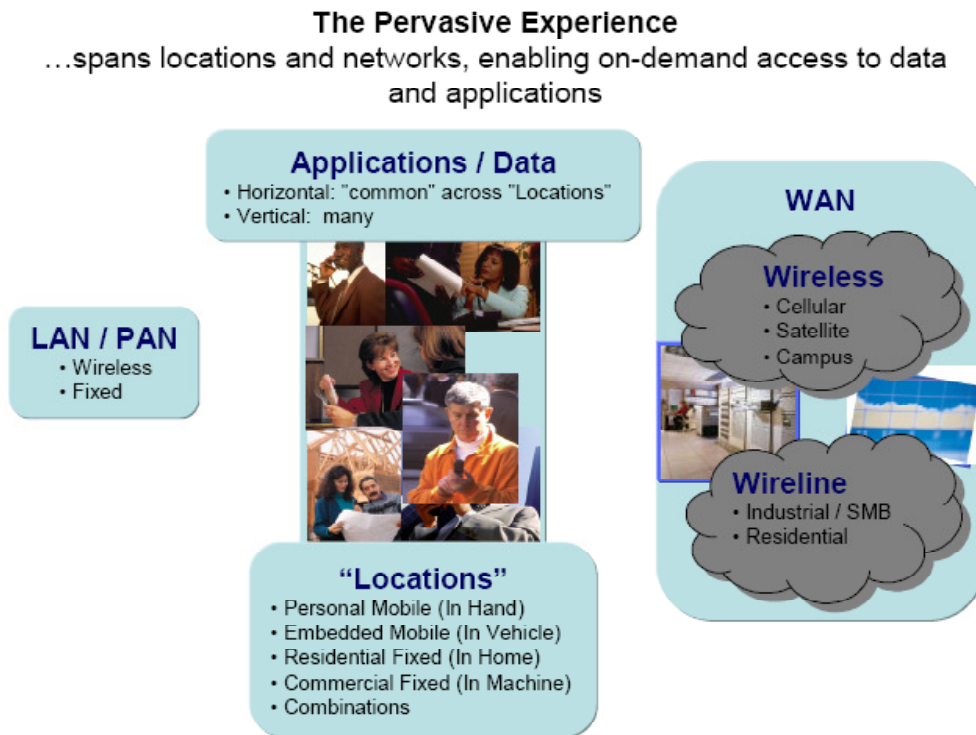


Figura 1.1: Experiencia pervasiva de IBM

Pervasive Computing Spectrum of Access

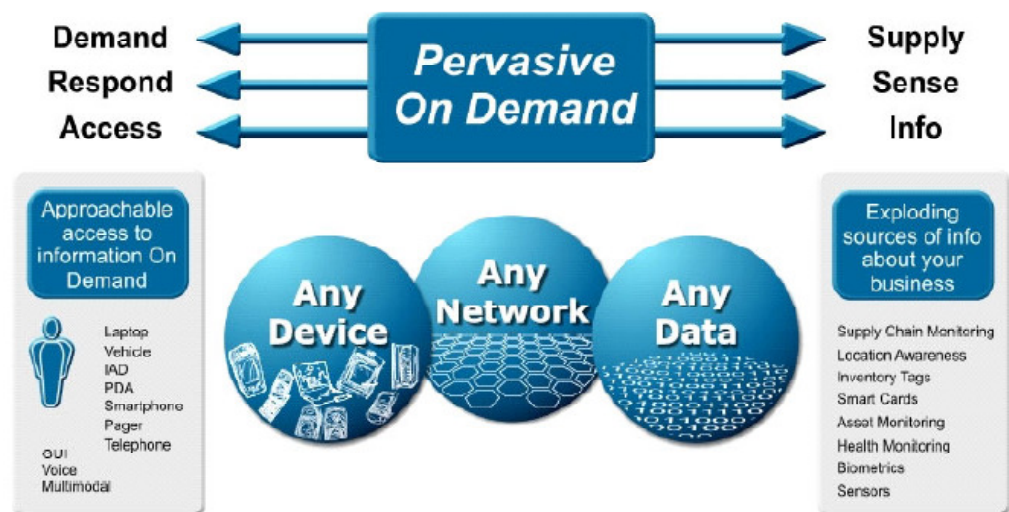


Figura 1.2: Computación ubicua

Extending the WebSphere Platform with Device Software

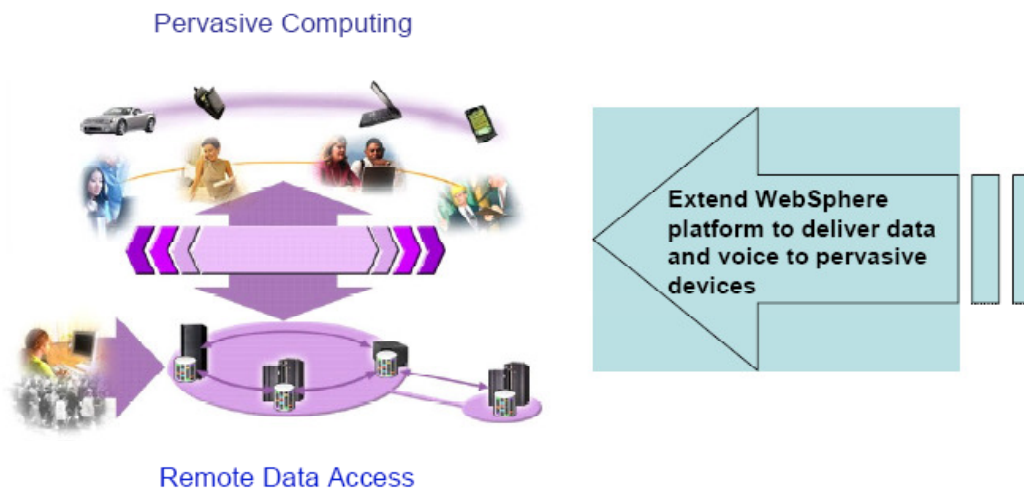


Figura 1.3: Experiencia ubicua extendida a la plataforma WebSphere

La experiencia pervasiva se extiende a la plataforma *WebSphere* para entregar datos y voz a todos los dispositivos ubicuos (ver fig. 1.3 de la pág. 4).

1.1 Modelo de Programación de WebSphere para Dispositivos Pervasivos

El modelo de programación de *WebSphere* para dispositivos pervasivos está diseñado para mover el actual modelo de programación de *WebSphere* a lo más bajo de la cadena, desde que los dispositivos acceden al navegador.

Proporciona un modelo estándar para los dispositivos de programación que sea compatible con el modelo que se está utilizando en el desarrollo *J2EE*, extendiendo el modelo *J2EE* para hacerlo más utilizable en el ambiente de dispositivos [2].

1.2 Alcance

El modelo *J2EE* está diseñado para soportar en general algún tipo de lenguaje de marcas para la conexión a navegadores. La marca es creada por un servlets o Java Server Pages y, a continuación es entregado a un navegador para su interpretación.

El dispositivo de salida es siempre un navegador o un agente similar de prestación.

Este agente está siempre conectado a una red con el fin de alcanzar el servidor. Si bien se trata de un excelente modelo de programación, se excluyen muchos otros métodos de conexión y de salida.

El portal de *WebSphere* extiende este modelo para soportar a otros formatos de salida a través de transcodificación. Por ejemplo, un portlet que produce Voice XML va a interactuar con un teléfono, en lugar de un navegador, pero la lógica básica de la aplicación de negocio seguirá siendo la misma.

Naturalmente, aún mejorando el modelo anterior, todavía existen espacios vacíos.

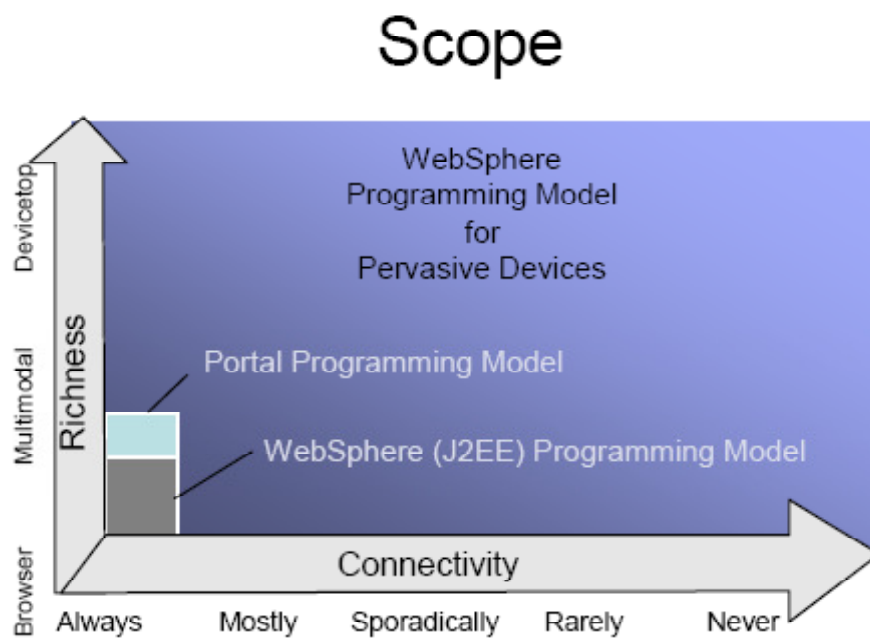
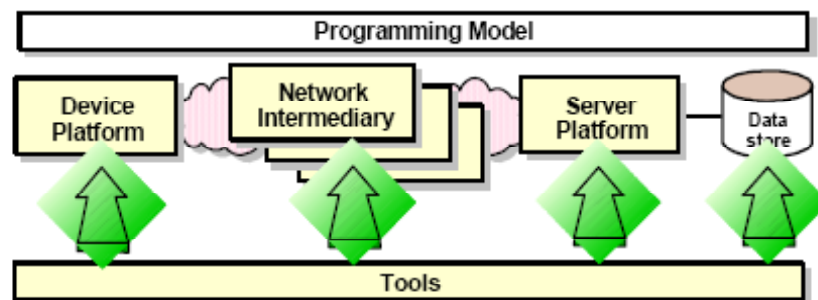


Figura 1.4: Alcance

PvC Architecture Scope



- **Built upon**
 - Existing and emerging standards
 - Industry technologies and products

Figura 1.5: Arquitectura PVC

El modelo de programación de *WebSphere* para dispositivos pervasivos extiende este modelo aún más a los dispositivos que no son necesariamente navegadores, y no pueden estar conectados todo el tiempo.

Por ejemplo, *WebSphere Everyplace Access* proporciona navegación fuera de línea (puede procesar formularios HTML mientras está desconectado, ya que esas operaciones han subido cuando la red estaba disponible).

1.3 Alcance de la Arquitectura PvC

La arquitectura de la computación pervasiva (PVC) es bastante simple: un dispositivo de algún tipo se conecta por alguna red a un servidor que puede tener acceso a datos.

Todas las piezas aquí se basan en estándares (XML, SyncML, JDBC, y así

End to End Programming Model

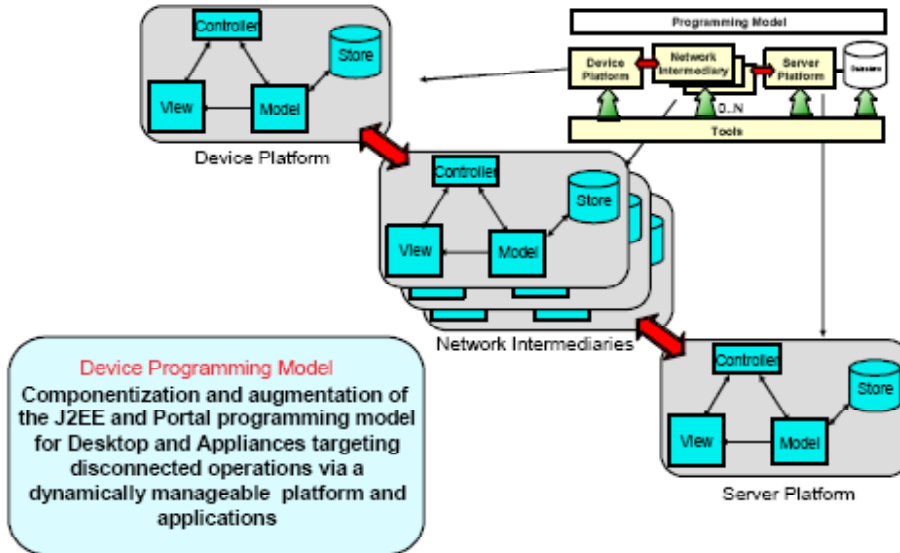


Figura 1.6: Modelo de programación extremo a extremo

sucesivamente), la industria de tecnologías aceptó que no hay estándares todavía por lo que las aplicaciones interoperarán con otras aplicaciones soportando esas normas (ver fig. 1.5 de la pág. 7).

Dentro de este ámbito, se podrá apoyar el modelo basado en navegador de *J2EE*, los dispositivos con navegadores, la navegación fuera de línea, VXML entregado a un teléfono móvil del usuario, y muchas otras opciones.

No se especifican como los datos son entregados o como son interpretados por el cliente final.

El modelo solo establece que los datos se mueven de un servidor a través de una red (o redes) a un dispositivo.

1.4 Modelo de Programación Extremo a Extremo

El extremo a extremo del modelo de programación se basa en la metodología estándar modelo-vista-controlador, generalmente en *J2EE*, un servlet actúa como controlador, Java Server Pages (JSP) proporciona la vista (o la aplicación de salida) y los datos se acceden a través de Enterprise Java Beans (EJBs), que son los modelos de lógica y datos de negocio en la parte final (ver fig. 1.6 de la pág. 8).

Esta arquitectura está siendo empujada a través de la red a nuevos dispositivos.

La desconexión de operaciones permitirá a la lógica de negocio funcionar si el dispositivo está actualmente conectado a la red o no.

Por ejemplo, las transacciones podrán ser almacenadas en un DB2[®] Everyplace (DB2e), base de datos a nivel local en un dispositivo, si la red no está disponible, y fluirá a la parte final una vez que los sistemas de conectividad de la red fue restaurada.

Con esta funcionalidad, los dispositivos comienzan a actuar como puntos finales en el modelo de programación.

1.5 WebSphere y las Aplicaciones para Dispositivos

Todas estas aplicaciones son construidas por las herramientas de *WebSphere Studio*. Todas las solicitudes están basadas en estándares abiertos para permitir la conectividad y la interoperabilidad con otras aplicaciones, plataformas y dispositivos.

Desde un punto de vista del desarrollo, es posible considerar la utilización de *WebSphere Studio Device Developer* como un plug-in para *WebSphere Studio Application Developer*.

1.6 La plataforma de servicios pervasivos

La plataforma de servicios pervasivos abarca una serie de diferentes entornos de desarrollo (ver fig. 1.7 de la pág. 10).

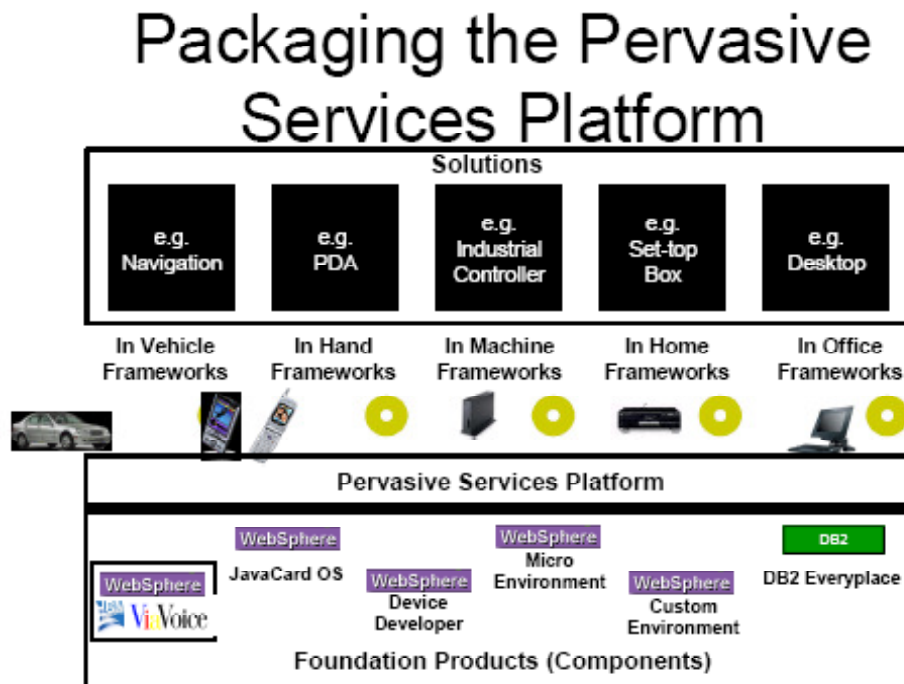


Figura 1.7: Paquete de la plataforma de servicios pervasivos.

Sin embargo, el código debería ser similar a través de los ambientes, y todos los códigos se llevarían a cabo en alguna de las variantes de Java - J2SE, J2ME, etc.

Los pilares soportan diferentes familias de dispositivos, pero la codificación sigue siendo básicamente la misma.

Para teléfonos móviles y PDAs, probablemente todavía se verá la programación de MIDP, ya que es habitual del modelo de programación.

Para set-top boxes y otros dispositivos, se puede tener *SMF* (kit de utilidades de desarrollo que facilita la creación de aplicaciones gestionadas por el servidor implementadas en el Services Management Framework) como marco para entregar paquetes.

1.7 ¿Cuál es el Modelo de Programación de WebSphere para la Computación Ubicua?

En este punto se desarrolla cómo el modelo de programación extiende el modelo *J2EE*: los navegadores son extendidos más allá de HTML a visual (VXML), o una combinación (navegadores y aplicaciones multimodales).

El “Escritorio” cliente podrá ser un ordenador portátil, PDA u otro dispositivo.

La red conecta el cliente a un servidor, pero la conexión de red puede ser esporádica (ver fig. 1.8 de la pág. 12).

Con el fin de hacer que esto funcione, se necesitará de inteligencia en el dispositivo.

Por ejemplo, WebSphere Everyplace Access tiene un cliente que sincroniza Lotus Notes[®] databases, buzones de correo Exchange, y bases de datos DB2 con un solo clic.

DB2 Everyplace podrá ser utilizado como un área de almacenamiento a nivel local, hasta que la red está disponible para fluir las transacciones.

MQ Everyplace podrá ser usado para la cola de mensajes a nivel local hasta que la red está disponible para enviar los mensajes a sus destinos.

What is the WebSphere Programming Model for Pervasive Computing?

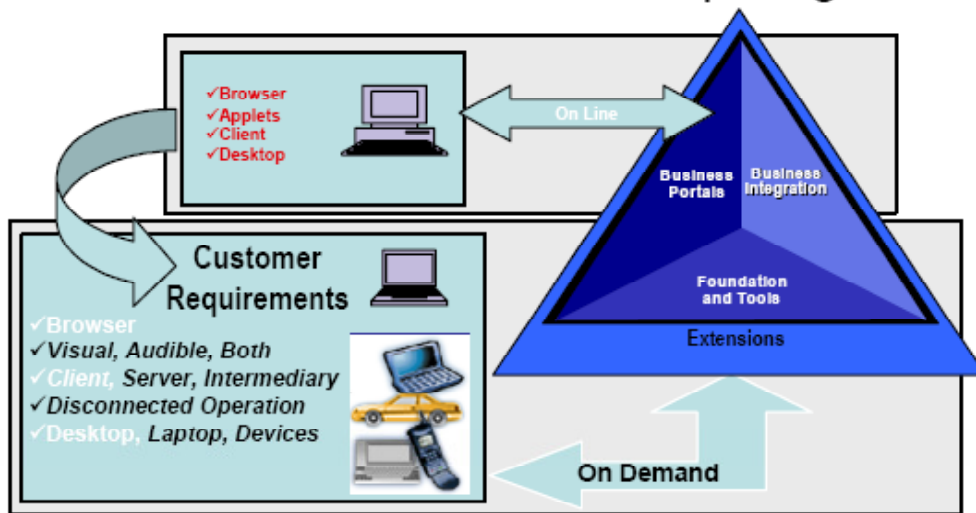


Figura 1.8: Modelo de programación de WebSphere.

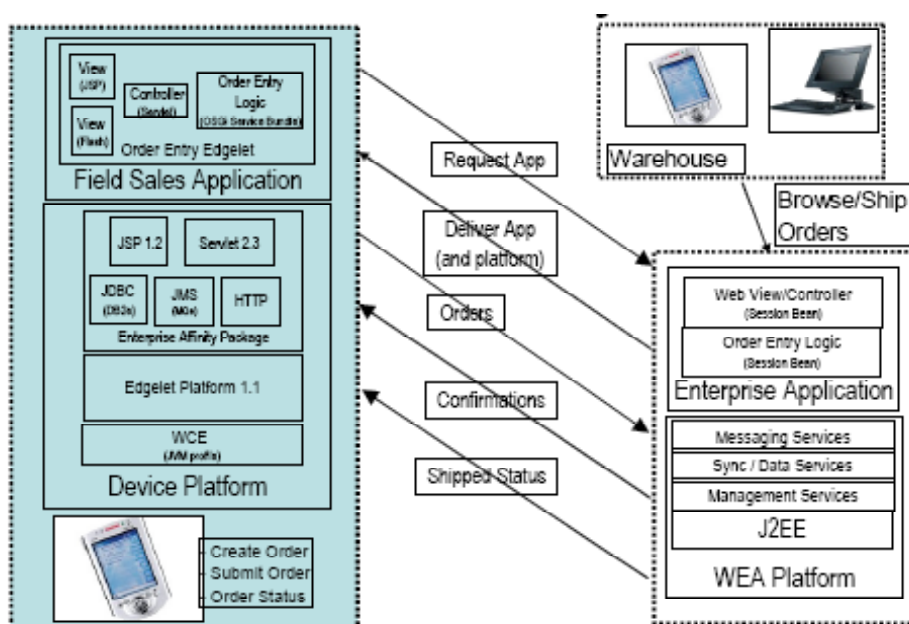


Figura 1.9: Ejemplo de una aplicación pervasiva.

SMF contiene un motor de servlets y procesador JSP que permite el procesamiento en el dispositivo, y también proporciona un conjunto de instrumentos para convertir archivos WAR a paquetes para facilitar la distribución y el despliegue de un código de J2EE a SMF.

1.8 Ejemplos de Aplicación Pervasiva

Se muestra un ejemplo de aplicación, con una posible implementación.

Se requerirá ser capaz de recoger órdenes de forma remota y, a continuación, procesar estas órdenes cuando sea conveniente (ver fig. 1.9 de la pág. 13).

Se tendrá una plataforma sobre la que el dispositivo puede descargar las aplicaciones y, a continuación, procesar las aplicaciones en el dispositivo.

La aplicación local puede recoger los pedidos y, a continuación, pasarlos

cuando la red está disponible. Así mismo, cuando la red está disponible, de cualquier estado anterior las órdenes deben fluir al dispositivo para su revisión.

Esta solicitud se basa en la extensión de servicios de WebSphere Everyplace (antes conocido como Edgelets).

La Extensión de Servicios SMF

En este caso, se está corriendo en el WebSphere Custom Environment. Se tendrá acceso a través de JDBC a DB2 Everyplace, mensajería a través de MQ Everyplace, y Servlets y procesamiento JSP a través del servicio HTTP en SMF.

La aplicación del dispositivo es una variación de la misma aplicación que se ejecute en el back-end de los sistemas, sólo convertidos a un conjunto para el despliegue en SMF.

Se podrá usar DB2 Everyplace para recopilar datos en el dispositivo hasta que la red está disponible, y entonces usar tanto SyncML para sincronizar con el back-end de bases de datos, o MQ para enviar mensajes de transacción entre los dos sistemas.

Capítulo 2

WebSphere Studio Device Developer IDE

2.1 Developer IDE

WebSphere Studio Device Developer es una plataforma para el desarrollo, depuración y despliegue de aplicaciones para dispositivos pequeños. Es miembro de la familia *WebSphere*.

Estos productos son construidos sobre la plataforma *Eclipse*, de manera que utilizar Eclipse es otra buena forma de familiarizarse con la funcionalidad básica del *WebSphere Studio Device Developer*.

El entorno de desarrollo integrado también viene con una copia del *WebSphere Micro Environment* (IBM-compatible con J2ME JVM), con licencia para el desarrollo.

2.2 Arquitectura Base de WSDD

WebSphere Studio Device Developer forma parte de la familia *WebSphere*. Todo producto *WebSphere Studio* tiene un ancestro común, el *WebSphere Studio Workbench* (WSWB), que es la versión de IBM de la plataforma *Eclipse*.

Eclipse es un ambiente de desarrollo de código abierto (open-source). Uti-

liza un plug-in diseñado para ampliar su funcionalidad básica, ya que solo hace muy poco.

Se podrá descargar la plataforma *Eclipse* en <http://www.eclipse.org>, incluyendo plug-ins para desarrollar código Java.

Puesto que es de código abierto, también se podrá contribuir a su desarrollo.

Periódicamente, *IBM* toma una instantánea de *Eclipse* y los distribuye como el *WebSphere Studio Workbench*, que está diseñado para ser una plataforma de desarrollo para socios de negocio para ampliar la arquitectura básica y complementos.

Estas herramientas también deben ser capaces de conectar a *Eclipse*, así como los miembros de la familia de productos de *WebSphere Studio*.

2.3 Arquitectura de Eclipse

La Plataforma *Eclipse* fue diseñado originalmente por *IBM* como base para sus herramientas de desarrollo.

El marco fue donado a la comunidad Open Source.

Se trata de un marco y guía para el desarrollo de entornos de desarrollo integrado.

El marco base (*Eclipse*) puede tener múltiples herramientas de desarrollo adjunto o conectado a ella.

Esto hace el entorno de desarrollo potente y flexible.

2.4 WebSphere Studio Workbench

WebSphere Studio Workbench (WSWB) es la versión de *IBM* de la plataforma *Eclipse*. Está a disposición de *PartnerWorld* para Desarrolladores miembros para su utilización en el desarrollo de plug-ins y sus propios productos (no es un producto de *IBM* para la venta).

WebSphere Studio Workbench es el fundamento de las demás herramientas

IBM WebSphere Studio, que son productos de IBM, y todos ellos extienden la arquitectura básica mediante plug-ins:

- *WebSphere Studio Site Developer* es WebSphere Studio Workbench con herramientas adicionales para el desarrollo de sitios web.
- *WebSphere Studio Application Developer* es WebSphere Studio Site desarrollado con herramientas adicionales para añadir desarrollo J2EE.
- *WebSphere Studio Enterprise Developer* es WebSphere Studio Application Developer con herramientas adicionales para sistemas con agregado de conectividad.

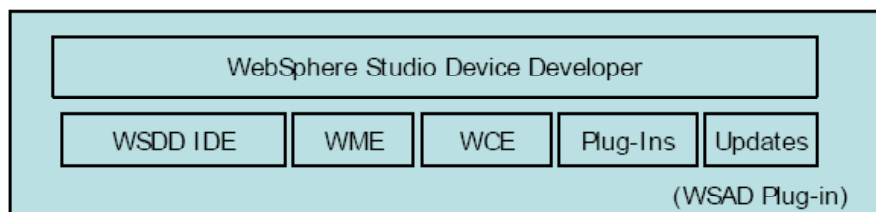
Desde la perspectiva de la arquitectura de Eclipse, WebSphere Studio Site Developer es un conjunto de plug-ins para WebSphere Studio Workbench. WebSphere Studio Application Developer es otro conjunto de plug-ins para WebSphere Studio Workbench que es un superconjunto de WebSphere Studio Site Developer, y así sucesivamente.

WebSphere Studio Device Developer no es parte de esta cadena, sin embargo se basa en *WebSphere Studio Workbench* al igual que los otros. Además, WebSphere Studio Device Developer puede funcionar como un plug-in para WebSphere Studio Site Developer y WebSphere Studio Application Developer para ampliar los productos y permitir que tanto el cliente y el servidor de la programación, estén dentro de una plataforma, o puede ser instalado stand-alone.

2.5 WebSphere Studio Device Developer

Esta es la funcionalidad básica de *WebSphere Studio Device Developer*. *WebSphere Studio Device Developer* contiene un entorno de desarrollo integrado, basado en la plataforma Eclipse, el WebSphere Micro Environment (WME, una potencia de Java en tiempo de ejecución que implementa las especificaciones J2ME), el WebSphere Custom Environment (WCE, Java en tiempo de ejecución que no implementa las especificaciones de J2ME, por lo que es más personalizable), funcionalidad adicional a través de plug-ins, y el gestor de actualizaciones para permitir la actualización del código a través de la Web (ver fig. 2.1 de la pág. 18).

WebSphere Studio Device Developer



WSDD can also act as a plug-in for WSAD to allow developing client and server-side applications from the same IDE

Figura 2.1: WebSphere Studio Device Developer

El *WebSphere Custom Environment* no se entrega con *WebSphere Studio Device Developer*, pero está disponible (para libre uso) en la Web a través del Administrador de Actualizaciones.

El *WebSphere Studio Device Developer* puede también ser instalado como un plug-in si el Site Developer o Application Developer es detectado en la máquina, *WebSphere Studio Device Developer* se instalará como un plug-in en el producto.

En anteriores emisiones, había una opción para instalar un plug-in o stand-alone.

En las actuales emisiones, si WebSphere Studio Application Developer se encuentra, *WebSphere Studio Device Developer* se instala como un plug-in únicamente, aunque esto ocurre al final de la instalación.

Si se desea utilizar WebSphere Studio Application Developer y WebSphere Studio Device Developer por separado en la misma máquina, se deberá instalar el WebSphere Studio Device Developer primero, luego el WebSphere Studio Application Developer.

Si se necesita más funcionalidad, se podrá añadir plug-ins para WebSphe-

re Studio Device Developer (o cualquier miembro de la familia WebSphere Studio).

Las actualizaciones están disponibles a través del gestor de actualizaciones, en la perspectiva Instalación / Actualización.

Se podrá descargar versiones actualizadas de los módulos existentes, así como añadir nuevas funcionalidades al WebSphere Studio Device Developer.

2.6 Trabajar con el IDE

WebSphere Studio Device Developer utiliza un concepto de espacio de trabajo, un directorio que contiene el código de trabajo (un subdirectorio por cada proyecto), y un subdirectorio de metadatos que contienen información sobre el código.

La creación de un acceso directo es importante cuando se desee utilizar múltiples espacios de trabajo, puesto que se puede usar la bandera “-data” para poner un específico espacio de trabajo en ejecución, por ejemplo, `wsdd.exe -data “C:\my_workspace\project1”`, se utilizará el subdirectorio `project1` en el directorio `my_workspace` como su espacio de trabajo.

WebSphere Studio Device Developer también utiliza el concepto de proyecto, una colección de paquetes que componen la totalidad de una aplicación Java (u otra). Por ejemplo, se podrá crear un Java, J2ME, MIDlet, C u otro proyecto.

Se deberá pensar un *proyecto* como un *super-paquete*. Cuando se empieza un desarrollo desde cero, se crea un proyecto. Se deberá escoger el tipo apropiado de proyecto.

Varios espacios de trabajo permiten mantener múltiples proyectos por separado, todos los proyectos en el espacio de trabajo son reconstruidos cuando se hace un “Reconstruir todos”.

En versiones anteriores, esta fue la principal forma para evitar la sobrecarga de la reconstrucción de proyectos.

Sin embargo, en siguientes emisiones, puede cerrar un proyecto, desde la perspectiva de recursos entonces el proyecto no será reconstruido.

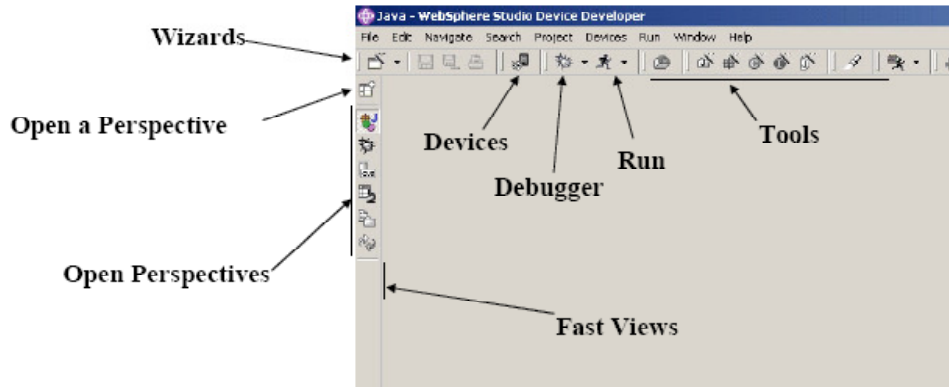


Figura 2.2: Barra de herramientas

Si no se encuentra un proyecto, o se está intentando importar un proyecto y muestra que ya existe, ir a la perspectiva Recursos y asegurarse de que no existe, o si está cerrado.

La pantalla de bienvenida actúa como un archivo readme para la versión de la herramienta, y se muestra automáticamente la primera vez que invoca el producto. También se encuentra en el menú Ayuda.

La herramienta se divide en *Perspectivas*, *barras de herramientas*, *vista*, y *editores*. La pantalla entera es a veces llamado el *Workbench*.

Una *perspectiva* es una colección predefinida de *barras de herramientas*, *vistas* y *editores* [1].

2.6.1 Barra de Herramientas

La barra de herramientas superior (bajo la barra de menús) contiene todos los wizards disponibles de *WebSphere Studio Device Developer*, que cambian para que coincida con la perspectiva actual.

Los Wizards están disponibles para crear aplicaciones, definir los dispositivos, probar código y crear estructuras Java (clases, interfaces, etc) dentro de sus programas (ver fig. 2.2 de la pág. 20).

La barra izquierda de herramientas contiene todas las perspectivas y / o vistas abiertas.

Cuando se abre una perspectiva, su ícono se añadirá a la parte superior derecha de la barra de herramientas.

Para cerrar una Perspectiva cuando se haya terminado con ella, o cuando haya demasiadas abiertas, simplemente hacer clic con el botón derecho sobre su ícono. En el menú contextual que aparece, se podrá cerrar esta perspectiva o todas las perspectivas abiertas.

Para crear una vista rápida, basta con arrastrar cualquier vista de la barra de herramientas.

2.6.2 Perspectiva, Editores y Vistas.

Una *perspectiva* es un grupo de vistas y editores en la ventana del espacio de trabajo diseñadas para centrarse en una determinada tarea.

En una ventana del espacio de trabajo pueden existir una o varias perspectivas.

Cada perspectiva contiene una o varias vistas y editores.

Dentro de una ventana, cada perspectiva puede tener un conjunto distinto de vistas, pero todas las perspectivas comparten el mismo conjunto de editores. También se puede personalizar perspectivas y guardarlas.

Una *vista* es un componente visual del espacio de trabajo.

Se suele utilizar para navegar en una jerarquía de información (como los recursos del espacio de trabajo), abrir un editor o visualizar propiedades del editor activo.

Las modificaciones efectuadas en una vista se guardan inmediatamente.

Sólo puede existir una instancia de un tipo de vista concreto en una ventana del espacio de trabajo.

Un *editor* se utiliza para modificar los archivos, y es específico para el tipo de archivo que está siendo editado.

Con WebSphere Studio Device Developer, se puede especificar editores

externos para determinados tipos de archivo.

Sólo un editor puede estar activo en cualquier momento, varios editores se podrán abrir, pero todos estarán en la misma ventana, disponible a través de pestañas.

Si se cambia algo en un editor, por lo general se guardará antes de que se utilice (antes de ejecutar el código, por ejemplo). Se podrá configurar una opción en preferencias para guardar automáticamente un archivo antes de que sea utilizado.

Si un archivo ha sido modificado en un editor y no se guarda, la ficha contiene un asterisco en el nombre del archivo para recordar que los contenidos deben ser guardados.

Precaución: Algunas vistas son editables. A diferencia de un editor, si se cambia algo en una vista, ese cambio es inmediato y permanente.

2.6.3 Espacio de Trabajo (Workspaces)

Todo el código se almacena en archivos separados en un “espacio de trabajo”, un directorio de repositorios basados en valores predeterminados para el espacio de trabajo dentro del directorio de instalación de *WebSphere Studio Device Developer*.

Se trata de una importante partida de la familia VisualAge[®] de herramientas que almacenan código en un repositorio, en lugar de archivos separados. Esto significa que podrá editar y procesar todos los archivos fuera de *WebSphere Studio Device Developer*, aunque esto no es recomendable.

Se podrá cambiar el espacio de trabajo, para que tome efecto se deberá reiniciar *WebSphere Studio Device Developer* para cambiar al nuevo espacio de trabajo.

Un espacio de trabajo contiene un directorio “. metadata” que *WebSphere Studio Device Developer* utiliza para mantener la información del proyecto.

2.7 Configuración J2ME

Java 2 Micro Edition (J2ME) es un conjunto estándar de configuraciones y perfiles para el desarrollo de Programas Java en dispositivos embebidos.

Si una aplicación puede aprobar una serie de pruebas, es certificada “Java Powered”, lo que significa que cumple con las especificaciones J2ME de las normas.

WebSphere Micro Edition ha sido certificado “Java Powered”.

WebSphere Studio Device Developer por defecto incluye soporte para J2ME en PalmOS y PocketPC. Otras configuraciones y el soporte de otras plataformas están disponibles a través del administrador de actualizaciones.

WebSphere Studio Device Developer soporta la descarga de código a un PDA para su ejecución y depuración.

2.8 Construir y Ejecutar en Dispositivos

WebSphere Studio Device Developer soporta el despliegue de código para varios dispositivos.

Definir los dispositivos para *WebSphere Studio Device Developer* y, a continuación, crear proyectos, se podrá configurar la ejecución de un proyecto en un dispositivo específico.

Dentro de cada proyecto, se es capaz de construir código para una plataforma concreta, o múltiples plataformas. Una vez que el código se construye, se podrá lanzar en un dispositivo específico (ya sea real o un emulador).

WebSphere Studio Device Developer utiliza un emulador MIDP y también se podrá ejecutar código en un JVM separado.

Soporta directamente los dispositivos Palm y el emulador de Palm y PocketPC. Otros dispositivos pueden ser soportados a través de plug-ins desde el fabricante del aparato.

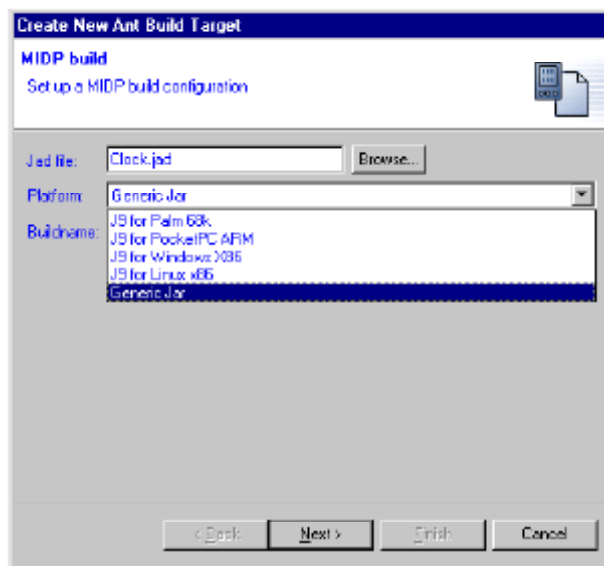


Figura 2.3: Constructor MIDP

2.8.1 Construcción

Al crear una construcción, se deberá especificar una plataforma y un nombre para la construcción.

Una posible convención de nombres es el nombre del proyecto, así como el nombre del dispositivo de destino (ver fig.2.3 de la pág. 24).

Las básicas plataformas son *PalmOS*, *PocketPC*, *WINx86*, *Linux x86*, y *JAR* genéricos. Otros dispositivos se podrán descargar desde el gestor de actualizaciones

Para pruebas sencillas que no requieren un emulador (con código que tiene una método `main ()`), se podrá seleccionar “Ejecutar como -> Java Application”. La aplicación se compila y luego es ejecutada en una JVM. Se verá la salida en la ventana consola de la perspectiva depuración.

Sin embargo, cada vez que se desee ejecutar en un dispositivo físico o ejecutar una aplicación MIDP (que requiere un emulador), se debe crear un constructor aparte.

Un proyecto puede tener múltiples constructores, en función de sus necesidades. Cada constructor debe tener un nombre único dentro del proyecto.

2.8.2 Dispositivos

Cada dispositivo requiere una configuración separada. Todos los dispositivos se comparten, por lo que cualquier proyecto puede ser desplegado en un dispositivo específico.

WebSphere Studio Device Developer soporta dispositivos Palm (a través de HotSync), emulador Palm, y dispositivos PocketPC (a través de ActiveSync) directamente.

WebSphere Studio Device Developer también ejecutará proyectos a nivel local JVM, y aplicaciones MIDP pueden ser desplegados en un emulador MIDP.

Otros proveedores pueden incluir emuladores soportados a través de plug-ins Eclipse. Estos deben estar disponibles a través del gestor de actualizaciones, o directamente desde el proveedor.

2.8.3 Creación de Dispositivos

Hay una función soportada para Palm y dispositivos PocketPC. Por defecto, estas son las únicas opciones para la creación de dispositivo.

Si se añaden plug-ins para otros dispositivos o emuladores, esos nuevos dispositivos deberían aparecer como opciones en esta pantalla (ver fig. 2.4 de la pág. 26).

2.8.4 Lanzar una Aplicación

Cuando se inicia un proyecto, se podrá configurar *WebSphere Studio Device Developer* que automáticamente se puede cambiar a la perspectiva de depuración en preferencias. Si se está ejecutando una línea de comandos de una aplicación Java, se querrá ver la consola para interactuar con él.

Si el dispositivo no es compatible con *WebSphere Studio Device Developer* directamente, se tendrá que exportar el código y luego transferirlo al dispositivo para ejecutarlo.

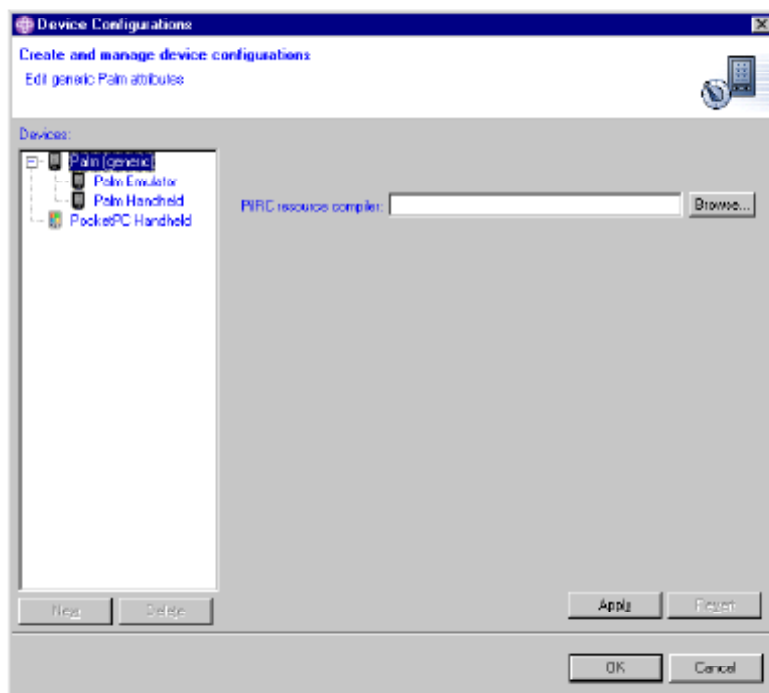


Figura 2.4: Creación de dispositivos.

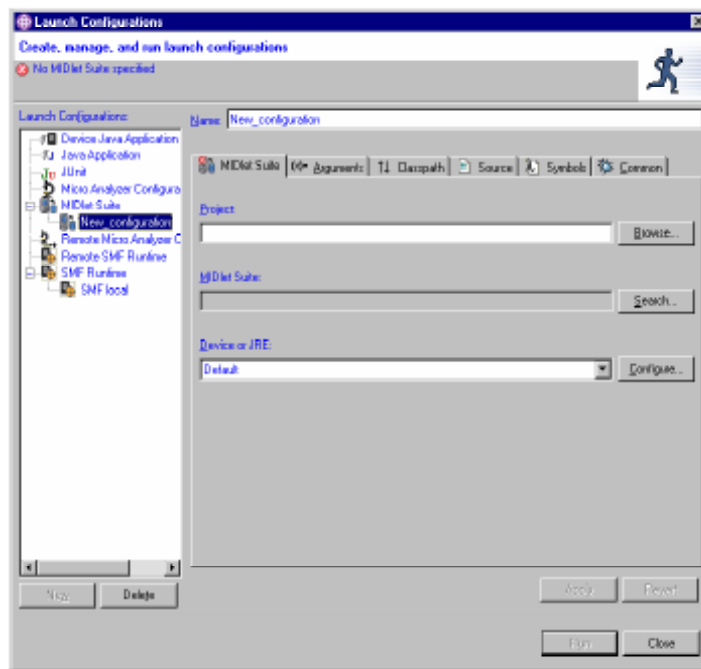


Figura 2.5: Lanzar una aplicación

Al configurar un lanzamiento, en primer lugar hay que elegir el tipo de aplicación que se está construyendo. A continuación, se debe dar un nombre, y seleccionar el proyecto.

Dependiendo del tipo de aplicación, también seleccionar qué constructor usar, por último, a qué dispositivo está dirigido.

Si no se ha definido todos los dispositivos, se podrá pulsar el botón Configurar junto al dispositivos del menú desplegable para definir uno.

Se debe tener un dispositivo definido con el fin de crear un lanzamiento. El nombre del lanzamiento debe ser único (ver fig. 2.5 de la pág. 27).

2.8.5 Realizar un Lanzamiento

Una vez que el lanzamiento ha sido configurado y ejecutado, aparecerá en el menú desplegable “Ejecutar”. Para llegar al menú Ejecutar, se podrá hacer clic en la flecha a la derecha del hombre corriendo, que también muestra el historial de ejecución (lanzamientos que se realizaron recientemente).

Simplemente al hacer clic en el hombre corriendo se volverá a ejecutar el último lanzamiento realizado.

El mismo comentario es aplicable para el menú Debug. Por lo tanto, si se necesita depurar lo que sea que se está trabajando, este es un atajo que se puede utilizar.

Bibliografía

[1] *WebSphere Studio Device Developer Product Documentation*. ibm, 1998-2004.

[2] *Using WebSphere Studio Device Developer to Build Embedded Java Applications*. ibm.com/redbooks, Abril 2004.

Índice de Materias

Arquitectura de Eclipse, 16	WebSphere Studio Site Developer, 17
Arquitectura de WSDD, 15	Workspaces, 22
Arquitectura PvC, 7	WSWB, 16
Barra de Herramientas, 20	
Configuración J2ME, 23	
Construcción, 24	
Creación de Dispositivos, 25	
Dispositivos, 25	
Eclipse, 15	
Experiencia Pervasiva, 1	
IDE, 19	
Lanzar una Aplicación, 25	
Modelo de Programación de WebSphere, 5, 11	
Perspectiva, Editores y Vistas, 21	
Plataforma de servicios pervasivos de WebSphere, 9	
SMF, 11	
WebSphere Studio Application Developer, 17	
WebSphere Studio Device Developer, 15	
WebSphere Studio Enterprise Developer, 17	

