# Programación avanzada en MIDP

**Florina Almenárez Mendoza**
**Celeste Campo**
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
{florina, celeste}@it.uc3m.es

# Contexto

- Conocer otras APIs de MIDP y paquetes opcionales
  - temporizadores
  - gestión de certificados y conexiones seguras
  - acceso a ficheros
- Entender el modelo de seguridad en MIDP 2.0

## Bibliografía:


- ***Especificación de MIDP 2.0*** (JSR 118). Disponible en http://www.jcp.org
- http://developers.sun.com/techtopics/mobility/apis/articles/fileconnection/index.html
- http://developers.sun.com/techtopics/mobility/midp/articles/permissions/
- ***J2ME : Java 2 micro edition : manual de usuario y tutorial***. Froufe, Agustín y Jorge, Patricia. Ra-Ma. [2004]. L/S 004.438 JAVA FRO, L/D 004.438 JAVA FRO. Capítulo 12, 17

# Índice

- **Temporizadores**
- Modelo de seguridad
- Acceso a ficheros



# Temporizadores

- Clases heredadas del API de J2SE
  - `Timer`
  - `TimerTask`
- Pasos para definir un temporizador
  1. **Definir la tarea:** Crear una subclase de `TimerTask` que implemente el método `run`
  2. **Preparar su ejecución:** Crear una instancia de la clase `Timer` e invocar su método `schelude`

     **`schedule(TimerTask tt, long delay, long period)`**
- Un objeto `Timer` es un único hilo en "background" que ejecuta todas las tareas (objeto `TimerTask`)
- Finalizar la tarea en cualquier momento ⇨ método `cancel()`



# Ejemplo TimerMidlet

```
private class MyTimerTask extends TimerTask {
 public void run() {
  num++;
  myGauge.setValue(goUp ? GAUGE_MAX-(num%GAUGE_MAX):num%GAUGE_MAX);
 }
}

public class TimerMIDlet extends MIDlet  {
 // Changes the state of timers to/from active to/from not-active
 private void flipFlop() {

    if (timersRunning) {
       timerTaskOne.cancel();
       timerTaskTwo.cancel();
       timer.cancel();
       timersRunning = false;
    } else {
       timer = new Timer();
       timerTaskOne = new MyTimerTask(gaugeOne, false);
       timerTaskTwo = new MyTimerTask(gaugeTwo, true);
       timer.schedule(timerTaskOne, 0, 1000);
       timer.schedule(timerTaskTwo, 0, 1500);
       timersRunning = true;
    }
 }
```


http://www.java-samples.com/showtutorial.php?tutorialid=728

# Índice

- Temporizadores
- **Modelo de seguridad**
- Acceso a ficheros



# Modelo de seguridad

- **MIDP 1.0:**
  - Modelo de seguridad basado en “sand-box”, garantiza que el código java no dañe al dispositivo:
    - verificación del bytecode.
    - el programador tiene acceso a un conjunto limitado y predefinido de APIs.
    - no se soporta JNI, ni la definición de cargadores de clase.
  - No proporciona seguridad a nivel de aplicación

- **MIDP 2.0:**
  - Presenta como una de las mejoras más importantes el soporte de **seguridad a nivel aplicación**
  - Un nuevo **paquete `javax.microedition.pki`**
  - Soporte a **comunicaciones seguras** ⇨ **HTTPS, `SecureConnection`**



# Seguridad a nivel de aplicación

- MIDlet Suite **fiable** (“trusted”) y **no fiable** (“untrusted”)
- MIDlet Suite no fiable
  - origen e integridad del fichero JAR no es fiable para el dispositivo
  - entorno restringido donde el acceso a APIs protegidos no es permitido o es permitido con permisos explícitos del usuario
    - Por ejemplo, permiso para acceder a `javax.microedition.io.HttpConnection`
- MIDlet Suite fiable
  - autenticación e integridad del fichero JAR puede ser fiable
  - asociado con un dominio de protección



# Dominios de seguridad

- Un **dominio de seguridad** define un conjunto de permisos que pueden ser concedidos y los modos de operación ⇨ **políticas**
- Por defecto, el dominio de seguridad es "**`untrusted`**" lo que implica que cualquier operación que esté controlada por un permiso, se consultará previamente al usuario.
- Cuando se consulta al usuario existen 3 modos de operación:
  - "**`oneshot`**": no se mantiene ningún estado y por lo tanto siempre es necesario consultar al usuario.
  - "**`session`**": se mantiene el estado mientras el MIDlet esté en ejecución.
  - "**`blanket`**": se mantiene el estado mientras el MIDlet esté instalado.
- La gestión de estos dominios se realiza a nivel de implementación de MIDP, no a nivel programador



# Permisos

- Mecanismo para **proteger el acceso a los APIs** o funciones que requieren autorización explícita
- Permisos en un dominio son:
  - `Allowed`: conjunto de permisos que están permitidos
  - `User`: conjunto de permisos que el usuario debe autorizar
- Nombres de permisos tienen una estructura jerárquica y son "case-sensitive"
- Atributos `MIDlet-Permissions` y `MIDlet-Permission-opt` definen el conjunto de permisos requeridos por el MIDlet suite (programador)
  - Críticos o no críticos (indispensables u opcionales)
  - El AMS puede decidir no instalarlo si no le puede proporcionar los permisos que solicita



# Permisos (II)

- Los permisos están asociados a operaciones de conectividad:
  - **javax.microedition.io.Connector.http**
  - **javax.microedition.io.Connector.socket**
  - **javax.microedition.io.Connector.https**
  - **javax.microedition.io.Connector.ssl**
  - **javax.microedition.io.Connector.datagram**
  - **javax.microedition.io.Connector.serversocket**
  - **javax.microedition.io.Connector.datagramreceiver**
  - **javax.microedition.io.Connector.comm**
  - **javax.microedition.io.PushRegistry**
- Ejemplo de atributos en el descriptor:

```
MIDlet-Permissions: javax.microedition.io.Connector.socket,
    javax.microedition.io.PushRegistry
MIDlet-Permissions-Opt: javax.microedition.io.Connector.http
```

- Los MIDlets adquieren estos permisos según el dominio de seguridad al que pertenecen



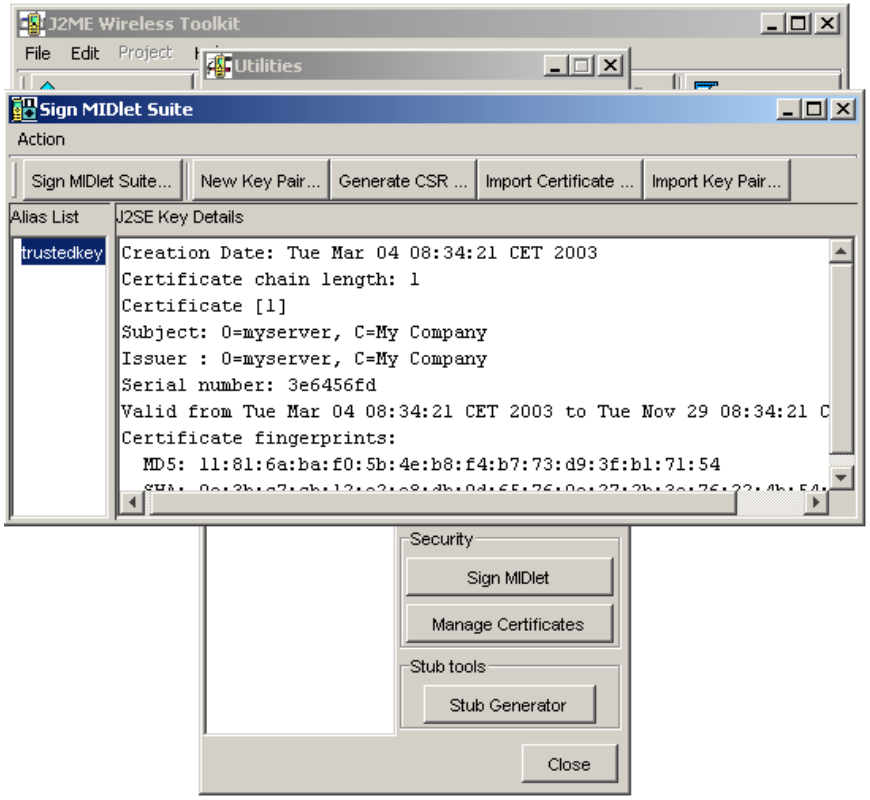
# Ejemplo fichero de políticas

```
domain: O="MIDlet Underwriters, Inc.", C=US
allow: javax.microedition.io.HttpConnection
oneshot(oneshot): javax.microedition.io.CommConnection
alias: client_connections
   javax.microedition.io.SocketConnection,
   javax.microedition.io.SecureConnection,
   javax.microedition.io.HttpConnection,
   javax.microedition.io.HttpsConnection
domain: O=Acme Wireless, OU=Software Assurance
allow: client_connections
allow: javax.microedition.io.ServerSocketConnection,
    javax.microedition.io.UDPDatagramConnection
oneshot(oneshot): javax.microedition.io.CommConnection
```



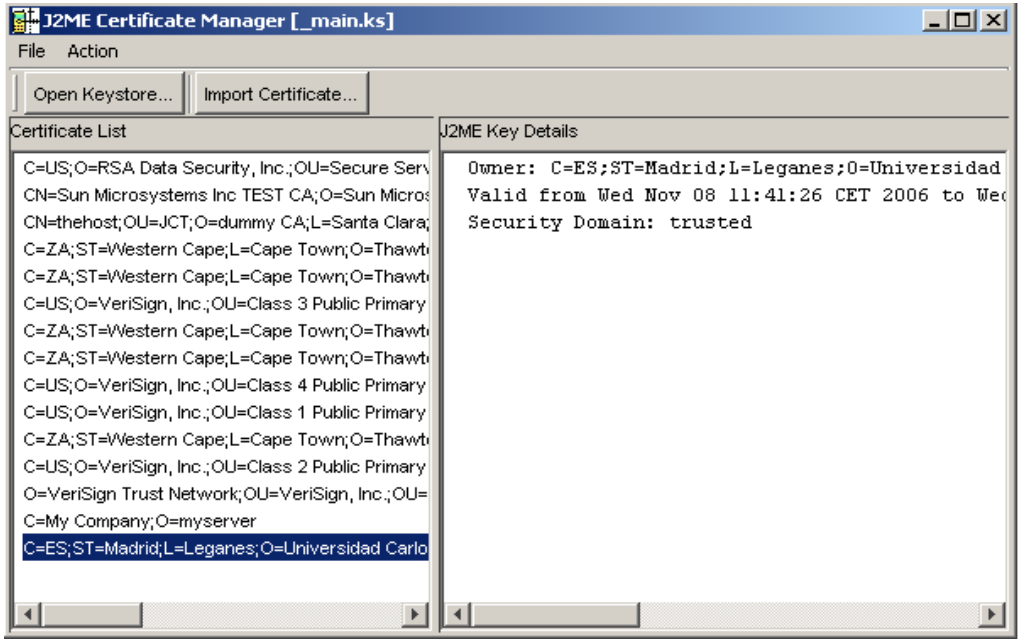
# Firma de código

- **¿Cómo se asocia un MIDlet Suite a un dominio de seguridad?**
  - Se puede realizar a través de **firmado de código** basado en **certificados de clave pública**
  - **Verificación de la firma**: Si el código proviene de un proveedor fiable (fabricante del móvil, operador,…) ese MIDlet Suite al instalarse pertenecerá a un dominio de seguridad determinado, por ejemplo, "**`trusted`**"



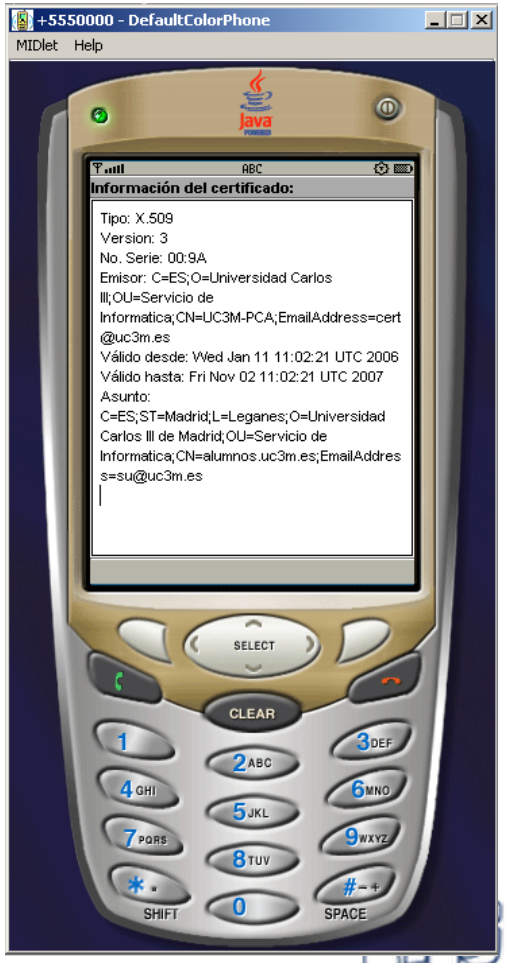
# Paquete `javax.microedition.pki` Gestión de certificados

- Los certificados son usados para gestionar
  - la instalación de MIDlet suites fiables y
  - a nivel de programación conexiones seguras
- Soporte obligatorio a certificados en formato X.509
- Interfaz `Certificate`
  - `getIssuer()`
  - `getNotAfter()`
  - `getNotBefore()`
  - `getSerialNumber()`
  - `getSubject()`
  - `getType()`
  - `getVersion()`



# Conexiones seguras
# Ejemplo HTTPS

```
private void verCertificado (String url) throws Exception {
  StringBuffer sb = new StringBuffer();
  HttpsConnection c = null;
  TextBox t = null;
  try {
   c = (HttpsConnection)Connector.open(url);
   SecurityInfo info = c.getSecurityInfo();
   Certificate cert = info.getServerCertificate();
   sb.append("Tipo: " + cert.getType() + "\n");
   sb.append("Version: " + cert.getVersion() + "\n");
   sb.append("No. Serie: " + cert.getSerialNumber() + "\n");
   sb.append("Emisor: " + cert.getIssuer() + "\n");
   sb.append("Válido desde: " + cert.getNotBefore() + "\n");
   sb.append("Válido hasta: " + cert.getNotAfter() + "\n");
   sb.append("Asunto: " + cert.getSubject() + "\n");
   t = new TextBox("Certificado: ", sb.toString(), 1024, 0);
  } finally {
     if (c != null) c.close();
  }
}
```



# Índice

- Temporizadores
- Modelo de seguridad
- **Acceso a ficheros**



# Nuevas APIs y extensiones de MIDP

- ¿Cómo puedo saber si un dispositivo soporta una extensión?
  - necesario ir a las especificaciones técnicas de los fabricantes del dispositivo.
  - consulta de las propiedades (`System.getProperty(String)`)
  - realizar pequeños programas de prueba para saber cuales o qué parte de las extensiones soporta.
- ¿Cómo puedo encontrar ejemplos y ayuda para desarrollar un MIDlet utilizando estas extensiones?
  - `http://developers.sun.com/techtopics/mobility/reference/techart/index.html`



# Acceso a ficheros

- MIDP no proporciona un mecanismo especifico para cargar ficheros que no sean imágenes
- ¿Qué opciones tenemos?
  - **empaquetar el fichero dentro del JAR** que forma el "MIDlet suite"
    - abrir un `InputStream` al fichero en el directorio `res`.
    - leer el archivo usando el método `getResourceAsStream( )`
  - utilizar el **paquete opcional PDA** (JSR 75)
    - `FileConnection` (FC) APIs (memoria interna o memorias de almacenamiento desmontables)
    - PIM APIs (contacto, calendario, tareas, etc.)

http://developers.sun.com/mobility/midp/questions/res/

# Sistema de ficheros
# Paquete `javax.microedition.io.file`

- Ni la configuración ni el perfil proporcionan acceso al sistema de ficheros o tarjetas de memoria externas
- Paquete opcional **PDA** (JSR 75), basado en el *Generic Connection Framework (GFC):*
  - Dos interfaces: `FileConnection`, `FileSystemListener`
  - Tres clases: `FileSystemRegistry`, `ConnectionClosedException`, `IllegalModeException`
- Propiedades
  - `microedition.io.file.FileConnection.version`
  - `file.separator`
- Seguridad
  - `SecurityException`



# Sistema de ficheros (II)
# Paquete `javax.microedition.io.file`

- Establecimiento de conexiones
  - `Conector.open(String url)`

    `url = file://<host>/<root>/<path>`
    - `Connector.open("file:///CFCard/")`
    - `Connector.open("file:///SDCard/")`
    - `Connector.open("file:///C:/")`
    - `Connector.open("file:////")`
  - `FileSystemRegistry.listRoots()`

- Procesamiento de *Stream*
  - `Conector.openInputStream`
  - `Connector.openDataInputStream`

  } lectura

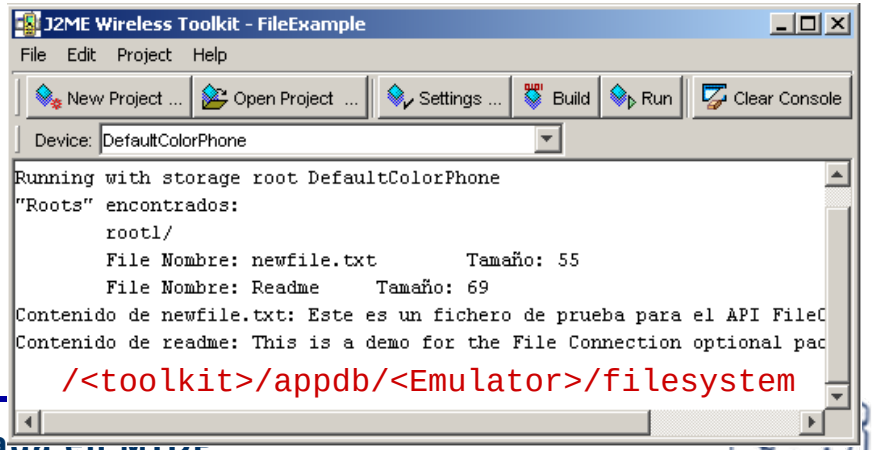
  - `Connector.openOutputStream`
  - `Connector.openDataOutputStream`

  } escritura

# Ejemplo FileConnection

```
private void getContent() {
 try {
   FileConnection fc = (FileConnection)Connector.open("file:///"+ root + "/");
   Enumeration filelist = fc.list("*", true);
   while(filelist.hasMoreElements()) {
     String fileName = (String) filelist.nextElement();
     fc = (FileConnection)Connector.open("file:///"+ root + "/" + fileName);
     if (fc.isDirectory()) {
         System.out.println("Directorio: " + fileName);
     } else {
         System.out.println("File Nombre: " + fileName + "Tamaño: "
                                                    +fc.fileSize());
     }
   }
   fc.close();
 } catch (IOException ioe) {
     System.out.println(ioe.getMessage());
 }
}
```

**FileDemo.java**



# Ejemplo FileManagement

```
public void crearArchivo() {
 try {
   FileConnection newfileconn = (FileConnection)
                                  Connector.open("file:///SDCard/newfile.txt");
   if(!newfileconn.exists())
      newfileconn.create();
   newfileconn.close();
 } catch(IOException ioe) { }
}

public void mostrarArchivo(String fileName) {
 try {
  FileConnection ofc = (FileConnection)
                               Connector.open("file:///SDCard/" + fileName);
  if(ofc.exists()) {
   InputStream is = ofc.openInputStream();
   byte b[] = new byte[1024];
   int len = is.read(b, 0, 1024);
   System.out.println("Contenido de "+ fileName + ": "+ new String(b,0,len));
  } catch (Exception e) { }
}
```

# Otros paquetes opcionales

- **Más información sobre PDA**

  http://developers.sun.com/mobility/allarticles/#pda

- **Mobile Media API**

  http://developers.sun.com/mobility/allarticles/#mmapi

- **Localización**

  http://developers.sun.com/mobility/allarticles/#location



# Integración Java EE y Java ME

**Florina Almenárez Mendoza**
**Celeste Campo**
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
{florina, celeste}@it.uc3m.es

# Objetivo

- Identificar los mecanismos de integración entre J2EE y J2ME

## Bibliografía


- ***J2ME : Java 2 micro edition : manual de usuario y tutorial***. Froufe, Agustín y Jorge, Patricia. Ra-Ma. [2004]. L/S 004.438 JAVA FRO, L/D 004.438 JAVA FRO. Capítulo 16
- ***Enterprise J2ME: Developing Mobile Java Applications***. Michael Juntao Yuan. Prentice Hall. 2003.
- http://java.sun.com/blueprints/wireless/




# Introducción

- J2ME representa un nuevo paradigma de aplicaciones móviles ⇨ “clientes inteligentes”
  - personalización, reducción de tráfico de red, hilos, interacción con otros módulos, seguridad, integración con otras tecnologías
- Comunicación de datos a un servidor J2EE ⇨ contenedor Web (servlets, páginas JSP, …) y contenedor EJBs
  - **HTTP**(S) o (Secure)**Socket**
    - **peticiones GET o POST** mediante un formato propio
  - **servicios Web** a través de un paquete opcional ⇨ **JSR 172**
  - MIDP “no” proporciona conectividad RMI
    - paquete opcional para CDC
- No debe existir comunicación directa con la capa de recursos EIS (RDBMS, ERS, …) ⇨ excepcionalmente por gestión
  - requiere JDBC API para manipulación de datos



# Comunicación HTTP o Socket

- cualquier **tipo de datos** aceptado en la cabecera `Content-type`
  - texto plano con formato específico, por ejemplo:
    - valores separados por coma: 1,valor1,2,valor2,3,valor3
    - parejas clave-valor: id=1,title="valor1" id=2,title="valor2" id=3,title="tres"
  - lenguajes de marcado: XML, (X)HTML
- **métodos HTTP**
  - **GET**, los valores son enviados como parte de la URL en la variable de entorno `QUERY_STRING`
  - **POST**, los datos son enviados como un flujo y su longitud es almacenada en la variable de entorno `CONTENT_LENGTH`
- tener en cuenta técnicas de **gestión de sesiones**
  - URL *rewriting*, por ejemplo
    - `http://www.somesite.com/servlet/product.html;`**`jsessionid=cl98373673At`**
  - Cookies



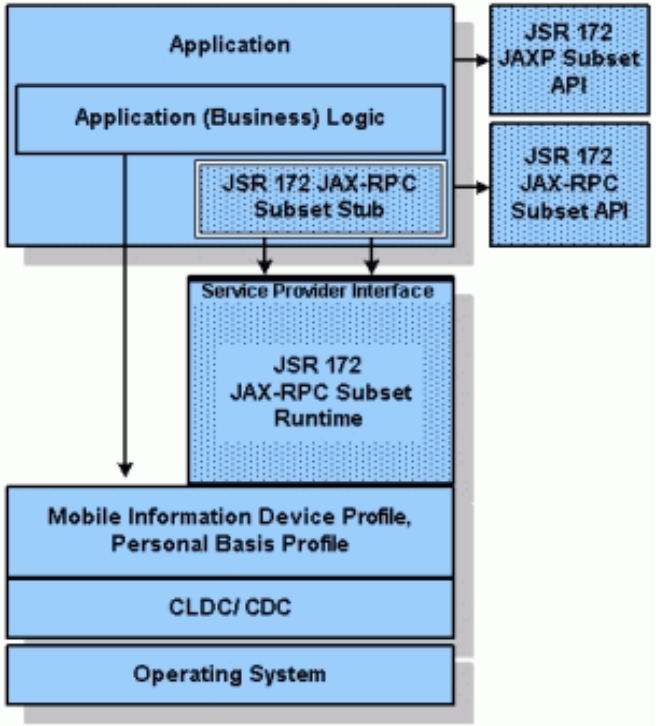
# Servicios Web

- Procesamiento XML y soporte RPC (*Remote Procedure Call*)
  - **WSA** (*Web Services API*)
  - bibliotecas de código abierto (k*)
- **XML y XML Schema**
  - MIDP no proporciona soporte nativo para “parseo” XML (**kXML**)
- **XML-RPC**
  - MIDP no tienen soporte nativo para XML-RPC (**kXML-RPC**)
- **SOAP (*Simple Object Access Protocol*)**
  - proporciona un protocolo basado en XML para codificación de datos
    - mayor versatilidad, mayor sobrecarga ⇨ información en cabeceras sobre seguridad, transacción, …
  - MIDP no proporciona soporte nativo para SOAP (**kSOAP**)



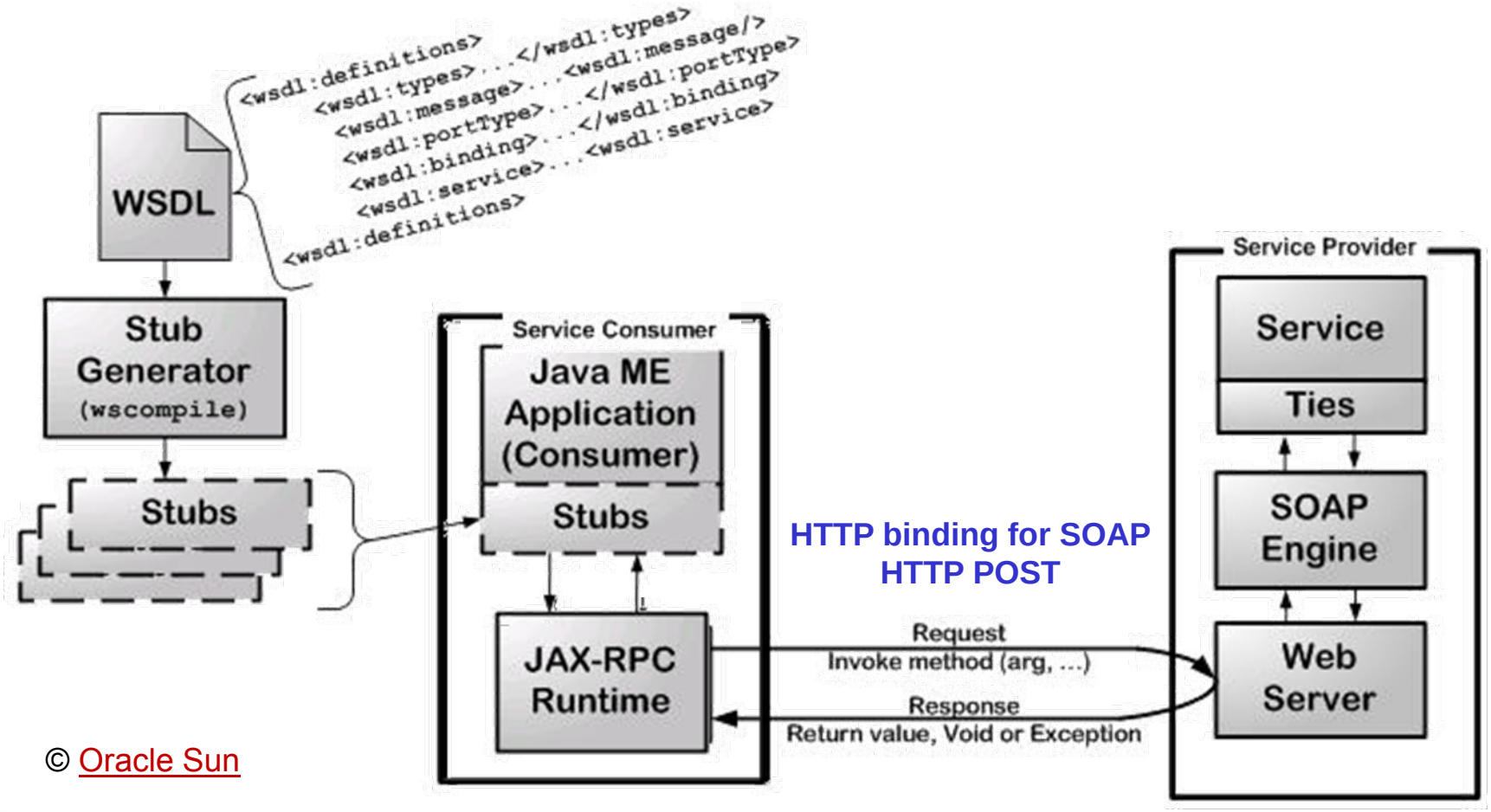
# WSA (JSR 172)

- Incluye dos paquetes:
  - **procesamiento XML** basado en la API JAVA (**JAXP**) 1.1 y API simple para XML versión 2 (**SAX2**)
  - **invocación remota de servicios web** como un subconjunto estricto de la API Java para RPC basado en XML (**JAX-RPC**) 1.1
    - conforme con el perfil básico de *WS-Interoperability* (WS-I)
    - evita tener que tratar con detalles de bajo nivel de HTTP/SOAP y cálculo de referencia de datos
- Descripción y descubrimiento de servicios remotos
  - WSDL (*Web Services Definition Language*) 1.1
  - UDDI (*Universal Description, Discovery and Integration*) 2.0

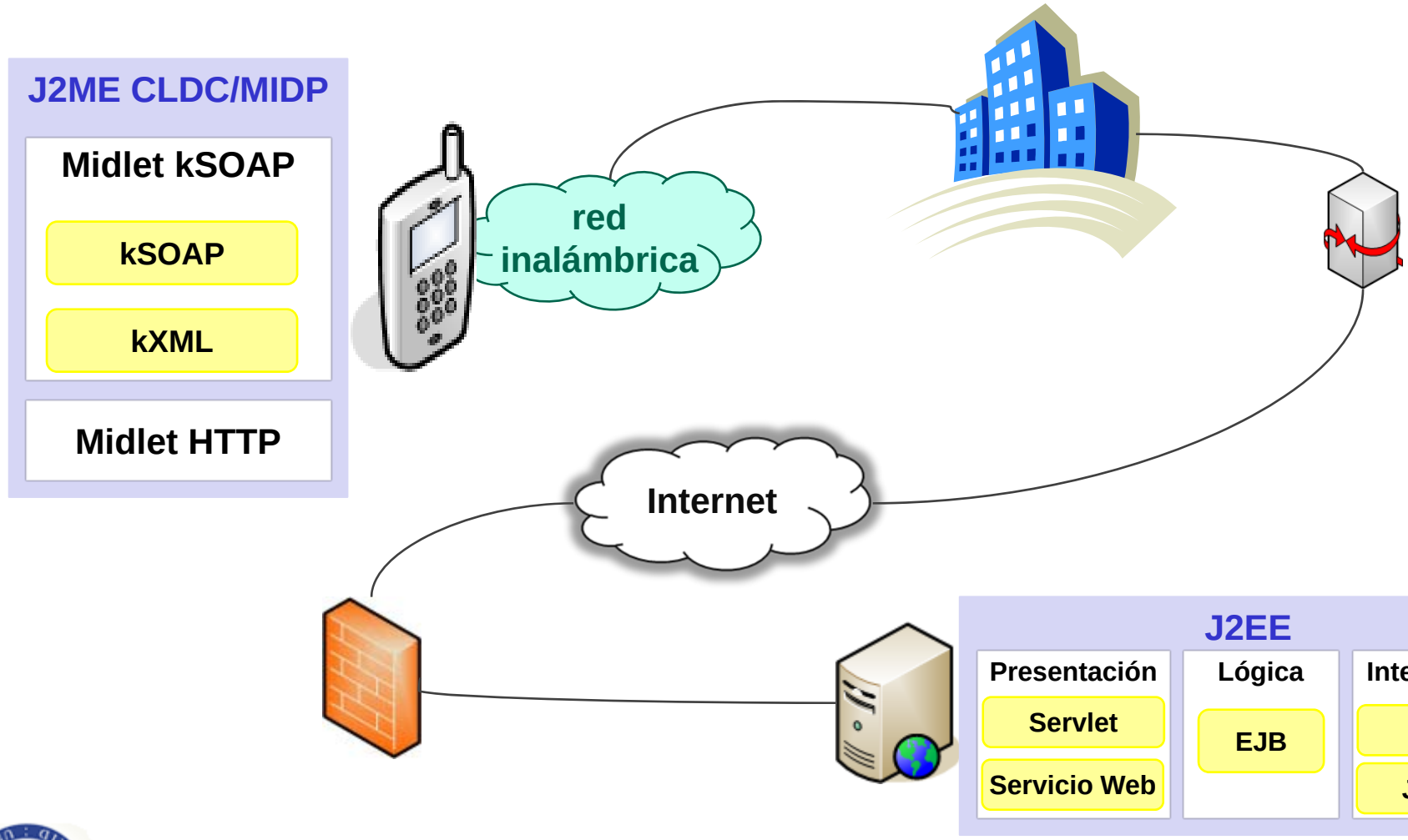



© Sun Oracle

# WSA
# JAX-RPC



# Arquitectura de servicios Web



# Ejemplo HTTP: Midlet

```
String url = "http://www.somesite.com/TestServlet";
String name = "Mike Taylor";
HttpConnection conn = (HttpConnection) Connector.open(url);
conn.setRequestMethod(HttpConnection.POST);
conn.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
conn.setRequestProperty("User-Agent", "Profile/MIDP-2.0 Configuration/CLDC-1.1");
conn.setRequestProperty("Content-Language", "en-US");
conn.setRequestProperty("Accept", "application/octet-stream");
String formData = "name=" + name;
byte[] data = formData.getBytes();
conn.setRequestProperty("Content-Length", Integer.toString(data.length));
OutputStream os = conn.openOutputStream();
os.write(data);
os.close();
int rc = conn.getResponseCode();
if (rc == HttpConnection.HTTP_OK) {
   DataInputStream dis = new DataInputStream(conn.openInputStream());
   String reg = dis.readUTF();
   dis.close();
}
```



# Ejemplo Socket: Midlet

```
String url = "socket://" + serverName + ":" + portNumber;
StreamConnection socket  = (StreamConnection)Connector.open(url, Connector.READ_WRITE);

// Send a message to the server
String request = "POST / HTTP/1.0\n\n";
OutputStream os = socket.openOutputStream();
os.write(request.getBytes());
os.write(new String("name=Mike Taylor"+"\r\n").getBytes());
os.close();

// Read the server's reply, up to a maximum of 128 bytes
InputStream is = socket.openInputStream();
final int MAX_LENGTH = 128;
byte[] buf = new byte[MAX_LENGTH];
int total = 0;
while (total < MAX_LENGTH) {
  int cnt = is.read(buf, total, MAX_LENGTH - total);
  if (cnt < 0) break;
  total += cnt;
}
is.close();
socket.close();
```

# Ejemplo HTTP: TestServlet

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {

   String name = request.getParameter("name");
   ByteArrayOutputStream baos = new ByteArrayOutputStream();
   DataOutputStream dos = new DataOutputStream(baos);
   dos.writeUTF("Hello " + name);
   byte[] data = baos.toByteArray();
   response.setStatus(response.SC_OK);
   response.setContentLength(data.length);
   response.setContentType("application/octet-stream");
   OutputStream os = response.getOutputStream();
   os.write(data);
   os.close();
}
```



# Ejemplo WSA

```
// JAX-RPC
String serviceURL = "www.j2medeveloper.com/pubwebservice";
String articleID = "IntroJSR172";

// Instantiate the service stub
PubService_Stub service = new PubService_Stub();

// Set up the stub
service._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, serviceURL);
service._setProperty(Stub.SESSION_MAINTAIN_PROPERTY, new Boolean(true));

...

try {
  // Invoke the PubService method getArticleByID() to retrieve a specific article by ID
  WirelessArticle article = service.getArticleByID(articleID);

  // Create a Form to display the article
  javax.microedition.lcdui.Form form = new Form(article.getShortTitle());
  form.append(wrap(article.getSummary()));
  ... ... display.setCurrent(form);

} catch (RemoteException e) { // Handle RMI exception

} catch (Exception e) { // Handle exception }

...
```

Fuente: http://developers.sun.com/mobility/apis/articles/wsa/index.html

# Más ejemplos y artículos

- Ejemplos de MIDlets sobre comunicaciones
  - http://www.java2s.com/Code/Java/J2ME/Networks.htm
- Revisar los siguientes artículos:
  - http://developers.sun.com/mobility/allarticles/#networking
  - http://developers.sun.com/mobility/allarticles/#bluetooth
  - http://developers.sun.com/mobility/allarticles/#ws
  - http://www.ibm.com/developerworks/ssa/webservices/tutorials/ws-soa-securesoap1/section2.html