Soporte XSL para herramientas que manipulan contenido XML. Una experiencia de desarrollo

XSL Support for Tools which Manipulates XML Contain: An Experience of Development

Luis Felipe Wanumen Silva

Ingeniero de sistemas y especialista en Ingeniería de Software de la Universidad Distrital Francisco José de Caldas. Docente catedrático de la Universidad Distrital adscrito a la Facultad Tecnológica. Director del Grupo de Investigación en Desarrollo de Herramientas para la Creación y Manipulación de Contenido XML, adscrito al Grupo *Metis* de la misma Facultad.
lwanumen@udistrital.edu.co

Clasificación del artículo: reflexión

Fecha de recepción: 18 de noviembre de 2005

Fecha de aceptación: 2 de junio de 2006

***Palabras clave*:** contenido XML, soporte para XSL, lenguaje XSL.
***Key words*:** XML contain, support to XSL, XSL languaje.

Resumen

En este artículo se desarrollan los aspectos fundamentales que deben tenerse en cuenta para añadir a una herramienta software que manipula contenido XML en Java, otorgando el soporte requerido para que pueda utilizar las potencialidades del lenguaje XSL. El proceso de soporte propuesto fue aplicado a una herramienta específica de software denominada *Desarrollo XML,* producto del trabajo del Grupo de Investigación en Desarrollo de Herramientas para la Creación y Manipulación de Contenido XML.

Abstract

This paper describes the principal aspects to take into account to add assistance in XLS for a software tool which manipulates XML contain in Java. The proposal support process applied to an specific software tool named «XML Development»; it is a product of the Researching Group in Development of Tools to Create and Handle XML Content.

1. Introducción

El lenguaje XSL (Extensible Stylesheet Language o lenguaje de hojas de estilo extensible) fue desarrollado por el W3C[1] con la finalidad de crear estilos y dar formato a los documentos XML, de tal forma que ellos puedan ser publicados en cualquier medio. Es muy similar a CSS[2]; de hecho, en parte está basado en las recomendaciones de CSS2. Con XSL se pretende proporcionar un mecanismo basado en tecnología XML; por esto el primero está basado en el segundo [1].

El lenguaje XSL se divide en dos, complementarios entre sí; cada uno de ellos está definido en su correspondiente recomendación[3]; ellos son:

- XSLT (*Extensible Stylesheet Language Transformations* o lenguaje de hojas de estilo extensible de transformaciones): define cómo efectuar las transformaciones necesarias para aplicar los estilos a los documentos XML.
- XSL-FO (*Formatting Object* u objetos de formateo): permite definir los estilos utilizando los objetos de formato [1].

Las herramientas que manipulan contenido XML usan la tecnología XSL en grados alto o bajo; las que la usan en grado bajo no permiten la creación de archivos XSLT, pero al menos permiten aplicar estas hojas de transformación a archivos XML para convertirlos a HTML; otras en cambio no sólo incorporan la posibilidad de usar XSLT ya establecidas, sino que también permiten la elaboración de hojas de estilo a partir de documentos XML, lo cual tiene un grado de complejidad alto: una buena lectura del documento XML es requerida para crear hojas de estilo que se adapten realmente a él.

Finalmente, existen algunas herramientas que permiten la utilización del lenguaje XQL[4], el cual permite acceder a información contenida en documentos XML, soportado por algunas herramientas de manipulación de este tipo de contenido.

2. Importancia del soporte para XLS en una herramienta que manipule contenido XML

Empleando hojas de estilo XSL pueden transformarse documentos XML y darles formato, por ejemplo para mostrarlos en un navegador *Web*. El componente de formateo XSL no se soporta en Internet Explorer 5.0; por tanto es necesario usar un XSLT, para transformar documentos XML en HTML y luego poderlos mostrar [2].

Si en un navegador es complicado encontrar componentes de formateo XSL, desarrollar herramientas que manipulen contenido XML e incluyan este componente es también difícil, en especial si se tiene en cuenta que la especificación XSL se encuentra aún en desarrollo en muchos aspectos.

Entornos de desarrollo como XML-SPY[5] tienen un grado alto de soporte para XSL; no obstante, otras herramientas como XMETAL[6] brindan un bajo y casi nulo soporte para el uso de este lenguaje, y a pesar de ello siguen siendo consideradas herramientas para creación y manipulación de contenido XML [3].

El soporte de XSLT en las últimas herramientas disponibles en el mercado es bastante grande; así, hoy los grupos de herramientas C++, XML-Oracle, Microsoft, Xalan y Unicorn incluyen soporte para XSLT 1.0. Todos presentan una interfaz similar, de

[1] El consorcio de la *World Wide Web*.
[2] *Cascading Style Sheets*. Los estilos creados con CSS también pueden ser utilizados por documentos XML para darles formato y publicarlos.
[3] La intención del W3C al hacer una recomendación es llamar la atención sobre la especificación y promover su difusión generalizada.
[4] XML Query Language. Se considera un derivado del trabajo sobre XSL [2].
[5] Es posible descargar una versión de esta herramienta, al igual que obtener documentación de la misma en: http://www.xmlspy.com
[6] Una versión de XMETAL, así como información sobre la misma se puede obtener en: http://www.softquad.com/products/xmetal/xml-intro.html

alto nivel, para transformación orientada a nombres de archivo (se trata de una llamada a un API[7], que parece una llamada a la línea de comandos) y una interfaz de bajo nivel que permite el ajuste final de las opciones de configuración para su transformación particular [4].

3. El uso de SAX en el proceso de soporte

Empleando SAX[8] en forma directa pueden llevarse a cabo funciones como:

- Configurar un analizador SAX y analizar un documento XML.
- Manipular elementos, listas de atributos, datos de caracteres e instrucciones de procesamiento.
- Construir un mecanismo de asignación «Elemento XML-método Java», utilizando reflexión.
- Obtener todas las declaraciones de una DTD[9]: elementos, listas de atributos, entidades y notaciones.
- Buscar fuentes de entrada en las rutas de clases CLASSPATH y leer su contenido.
- Construir manipuladores sencillos de errores, resoluciones de entidades, manipuladores de DTD y manipuladores de léxico [5].

Si se quieren efectuar otros procesos más sofisticados se hace necesario un trabajo de programación más amplio, también factible debido a que cualquier manipulación puede hacerse con el DOM[10] y éste se basa en SAX. Queda entonces claro que cualquier manipulación y creación de XSL podría hacerse con SAX, pero con un trabajo arduo de programación.

La creación, por parte de un usuario de un archivo XSL acorde con el archivo XML original, debe reflejarse en una transformación. Para ello, deberá hacerse uso del paquete *javax.xml.transform*, el cual tiene una clase llamada *TransformerFactory* que puede ser usada para crear un objeto tipo *Transformer*. Este último, desde su misma creación, tiene asociada una hoja de transformación que puede realizar transformaciones en cualquier momento invocando el método *transform* (*xmlSource, outputTarget*); éste recibe un archivo XML original y genera una archivo producto de la transformación deseada.

Un ejemplo típico de una transformación en Java es el siguiente:

```
Try{
   File hoja_estilo = new File ("estilo.css");
   StreamSource estilo_fuente = new StreamSource(hoja_estilo);
   Transformer t = TransformerFactory.newInstance().newTransformer(estilo_fuente);
   t.transform(xmlSource, outputTarget);
}
catch(Exception e)}
   System.out.println("Ha ocurrido errores con la transformación");
}
```

4. Clasificación de herramientas oferentes de soporte XLS para diferentes propósitos

4.1. Herramientas que permiten el uso del lenguaje XQL

Las siguientes herramientas permiten el uso del lenguaje XQL desde las herramientas de manipulación de contenido XML, soportando de esta forma el uso del XSL:

- **Xtable**. Su funcionamiento es interesante. Los documentos XML deben registrarse previamente en ella para que los pueda procesar; internamente, la herramienta convierte los documentos XML en tablas; los datos son vinculados con tablas nuevas o existentes, dependiendo de la lógica del archivo XML y el DTD; porque también se hace necesario pasarle su DTD. Así se logra que al hacer una consulta la herramienta consulte información mixta, esto es, perteneciente a los XML originales o de la base de datos relacional. Esto es posible, porque, inicialmente, se hace una búsqueda jerárquica en los docu-

[7] Del inglés *Application Programming Interface*, o Interfaz de Programación de Aplicaciones.
[8] *Simple API for XML*.
[9] *Document Type Definitions*.
[10] *Document Object Model*, o Modelo de Objetos de Documento.

mentos XML previamente registrados, y una asociación entre el nodo padre y el nodo hoja tipo *foreign key*. Debe recordarse que una DTD puede especificar algunos elementos como opcionales y de gran multiplicidad, como en el caso del signo «+»; esta característica debe manejarse con sumo cuidado, para asegurarse que su traslado al modelo relacional sea lo más limpio y coherente posible.

- **Silkroute.** Con esta herramienta los usuarios hacen consultas usando el lenguaje XML-QL, mientras que la vista se crea usando el lenguaje RXL [6]. Dado que usar varios lenguajes causa traumatismos, los fabricantes no tardaron en permitir que las dos funciones se hiciera en XQL.
- **Xbd.** *Xquery* no es el único lenguaje utilizado para acceder a la vista de una base de datos empleando herramientas con soporte XML; en *Xbd* el acceso a la vista se hace mediante los lenguajes *XSL mejorado* y *XQuery mejorado*, que son mejoras a los lenguajes originales hechas específicamente por esta herramienta [7].

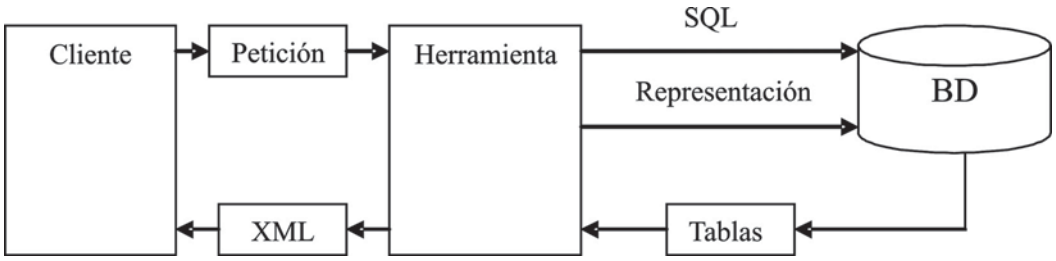

Figura 1. Herramientas que transforman el modelo relacional a XML.
Fuente: propuesta del autor.

4.2. Herramientas que transforman el resultado de consultas a XML.

La figura 2 describe en forma general el funcionamiento de estas herramientas:

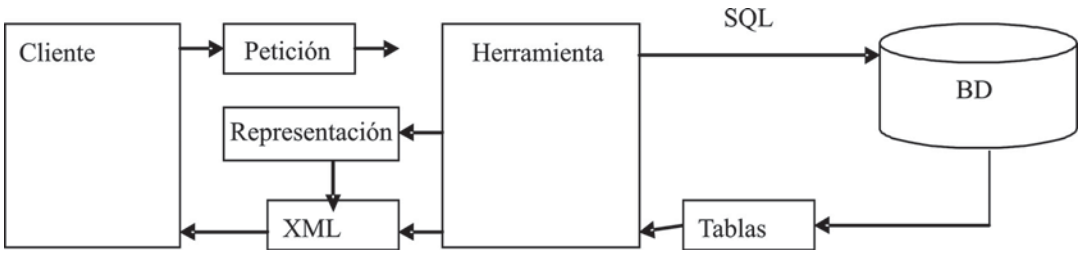

Figura 2. Herramientas que transforman el resultado de consultas a XML.
Fuente: propuesta del autor.

En comparación con el grupo anterior, la ventaja de este tipo de herramientas es que la consulta es mucho más rápida, dado que los datos están almacenados en tablas relacionales [8]. De acuerdo con la figura 2, el problema es que el XML se genera después de tener los resultados; por tanto, se presentan inconvenientes si los datos no provienen de tablas con relaciones que cumplan la segunda forma normal. *XML SQL Utility (XSU)* y *Oracle XSQL Pages* son herramientas que pueden clasificarse dentro de este tipo [9]. Una clasificación más detallada de este tipo de herramientas ha sido obtenida en la Universidad de Deusto[11].

4.3. Herramientas para la generación de documentos en formatos distintos a HTML

Para desempeñar esta función existe *FOP*[12], de Apache XML Project. Dispone de un procesador de XSL-FO para plataformas Java de carácter gratuito[13]; además incorpora una versión de los procesadores *XML Xerces* y *XSLT Xalan*. Utilizando *Xalan*, la generación de archivos PDF se puede efectuar en dos pasos; con FOP puede hacerse solo en uno.

- Transformación en dos pasos:
- Generación del archivo de objetos de formato (FO) con Xalan

En la figura 3 se muestra el proceso respectivo.

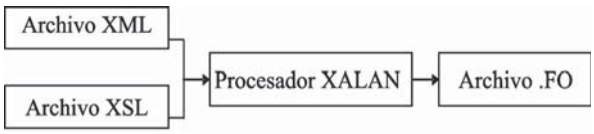

Figura 3. Generación de un archivo de objetos de formato (FO) empleando Xalan.

Fuente: Rodríguez, A. (2004). Publicación en Internet y tecnologías XML. Ed. Alfaomega, 2004, p. 415.

- Generación del archivo PDF con FOP. Véase figura 4.

Figura 4. Generación de un archivo PDF con FOP
Fuente: Rodríguez, A., op. cit., p. 415.

- Transformación en un paso. Véase figura 5.

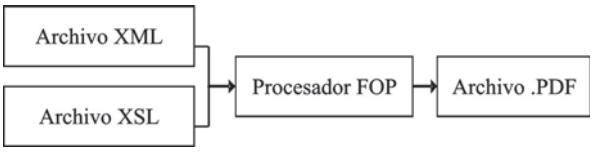

Figura 5. Proceso de transformación en un paso
Fuente: Rodríguez, A., *Op.Cit*., p. 415.

5. Soporte XSL usado en la herramienta *Desarrollo XML*

A continuación se muestra el soporte que otorga a la tecnología XSL la herramienta software *Desarrollo XML*[14] y el grado en el que se usa.

5.1. Implementación del proceso

Teniendo en cuenta que las hojas XSL cumplen la estructura de un documento XML bien formado es posible la construcción de un analizador de hojas de estilo basado en SAX.

El documento XML se lee con SAX desde el comienzo hasta el fin y en forma secuencial, lanzando sucesos a medida que ello sucede; tales sucesos pueden usarse para ir construyendo un árbol que posea los elementos XML. Disponiendo de este árbol en memoria se establecen sus niveles; para cada nivel se identifican los nombres de las etiquetas y se crea una plantilla para cada una.

[11] En [10] se presenta una clasificación mucho más detallada que la mostrada en este documento.
[12] Formatting Object to PDF.
[13] Disponible en: http://xml.apache.org/dist/fop

[14] Herramienta software obtenida por el Grupo de Investigación en Desarrollo de Herramientas para la Creación y Manipulación de Contenido XML, de la Universidad Distrital Francisco José de Caldas.

Al final se tienen todas las plantillas y se procede a la creación de un patrón que seleccione las porciones del documento XML que no se encuentran ausentes, inicialmente para los niveles uno y dos del árbol, indicados por las etiquetas de primer orden del archivo original XML. Finalmente, el patrón y las plantillas se unen en un documento XSL; así se adiciona soporte para la generación automática de XSL a partir de la lectura de un documento XML inicial.

5.2. Proceso de soporte

Desarrollo XML brinda soporte a archivos XSL creados por el usuario haciendo uso del paquete *javax.xml.transform*. Para manipular los errores registra un oyente de errores tipo *ErrorListener* a un objeto tipo *Transformer*; este oyente manejará los errores ocurridos durante la transformación. Al mismo objeto se aplica uno que contenga las propiedades de la transformación; finalmente, *Transformer* se usa para realizar la transformación deseada. En este caso particular la herramienta está prediseñada para generar documentos en formatos planos; aún no genera salidas en formatos no planos, como el PDF.

5.3. Generación de documentos PDF a partir de documentos XML

Para este propósito, el método empleado por la herramienta se encuentra en un estado de desarrollo incipiente. Aunque debería ser un método basado nativamente en XML, lo que hace es usar paquetes libres (*com.lowagie.text.pdf* y *com. lowagie. text*, los cuales permiten la creación del documento requerido casi en forma manual. Así, después de analizar el documento XML, verificar su buena formación y su validez con respecto a la DTD asociada, si esta última existe, la herramienta hace una lectura SAX del documento y va creando, mediante programación, cada uno de los elementos del documento PDF.

6. Trabajos futuros y mejoras potenciales a *Desarrollo XML*

En relación con el soporte de XSL que brinda la herramienta *Desarrollo XML* pueden hacerse mejoras en la creación de los XSL, en el soporte a los XSL creados originariamente por el usuario sin la ayuda de la herramienta, y en el uso de XSL para generación de salidas de archivos en formatos distintos al HTML.

Una mejora fundamental en este sentido es agregar la opción de definir plantillas para todos los niveles del árbol DOM. La carencia de esta implementación en la fase inicial para todos los niveles del árbol hace que se presentan problemas cuando, por ejemplo, se hace un ciclo iterativo en el patrón y no todos los elementos se encuentran en todos los nodos del árbol como sus hijos.

Desarrollo XML inicialmente genera salidas en formato HTML, producto de la transformación de un documento aplicándole XSL; se hace necesario ampliar su funcionalidad añadiendo soporte para salidas XML usando esta técnica. Lo anterior demanda el manejo de procesadores como *Xalan* para generar archivos .FO, que a su vez puedan ser transferidos a procesadores FOP para culminar con la generación de archivos PDF.

Para integrar un procesador *Xalan* en la herramienta se plantea la siguiente metodología, la cual otorgará flexibilidad de conversión de documentos en formato PDF. Los pasos propuestos son: [4]

- Incluir las cabeceras Xerces
- Inicializar Xerces
- Inicializar Xalan XSLT
- Crear fuentes y soportes para enlazarlos
- Crear el procesador
- Crear los contextos de ejecución (específico de Xalan)
- Crear las fuentes de entrada

- Definir la salida
- Invocar los procesos
- Finalizar Xerces.

7. Conclusiones

- No todas las herramientas que manipulan contenido XML brindan soporte para la creación de XSL; incluso en algunos casos se requiere crearlos en otras herramientas e incorporarlos luego para poder visualizarlos. A pesar de lo anterior debe decirse que las últimas herramientas que manipulan contenido XML tienen este soporte; el proyecto en curso exige que se implemente esta funcionalidad adicional a la herramienta.
- SAX no brinda soporte para hacer transformaciones XSL en forma directa, dado que no es aplicación ni analizador de XML. Sin embargo, para obtener funcionalidades sobre documentos XML pueden emplearse un juego de interfaces utilizables por los analizadores. Implementar soporte para transformaciones XSL usando SAX implica un gran trabajo de programación, pero se alcanzan importantes ventajas de desempeño; las XSLT pueden ser tan grandes que el solo hecho de verificar que estén bien formadas empleando DOM podría consumir muchos recursos; empleando SAX, las complicaciones del algoritmo se compensan con el beneficio en rendimiento de operación.
- El grupo de investigación desarrolla actualmente una herramienta software a la cual debe agregarse XSLT empleando alguna de las API *Xalan* o *Xerces*, para implementar más funcionalidades de conversión de documentos.

Referencias bibliográficas

[1] Rodríguez A. (2004). *Publicación en Internet y tecnología XML*. Ed. Alfaomega Ra-ma.

[2] Morrison, M. et al. (2002) *XML al descubierto. La solución más completa*. Ed.: Prentice Hall.

[3] Vásquez, A. (2001). *Navegar en Internet. XML*. Ed. Alfaomega Ra-ma.

[4] Arciniegas, F. (2002) *C++ XML*. *Guía Avanzada*. Ed. Prentice-Hall.

[5] Sjöberg C. (1998) *Critical Factors in Legal Document Management: A Study of Standardised Markup Languages*. Stockholm: Jure AB.

[6] Aqueveque, J.P. *Sindicación XML, RDF, RSS y Atom*. Disponible en: http://www.maestrosdelweb.com/editorial/sindicacion/ [consultado en: 6 septiembre de 2005].

[7] Fermoso A. (2003) *XBD: Sistema de consulta basado en XML a bases de datos relacionales.* PhD thesis, Facultad de Ingeniería ESIDE, Universidad de Deusto.

[8] Page W. Jr. *et al*. (2001). *Oracle 8 /8i*. Ed. Prentice Hall.

[9] Oracle (2002). *Oracle 9I Release 2. XML Database Developers's Guide,* Oracle Corp.

[10] Berjón R., Fermoso A. y Gil M. (2005) *Obtención de datos XML a partir de información almacenada en bases de datos.* En: XATA2005-XML: Aplicações e Tecnologias Associadas-3ª Conferência Nacional, Braga: Universidade do Minho, pp. 16-27.

Bibliografía recomendada

[11] Baumeister H., Koch N. y Mandel L. (1999). *Towards a UML Extension for Hypermedia Design*. In: Proceedings of the UML'99. The Unified Modelling Language-Beyond the Standard. Ed. Robert France y Bernhard Rumpe, LNCS 1723, Springer Verlag, Fort Collins, USA, pp. 614-629.

[12] Bray, T., Paoli, J, Sperberg-Mc.Queen, C. M. y Maler, E. (2000) *Extensible Markup Language (XML) 1.0* (2.nd ed.), W3C Recommendation. Disponible en: http://www.w3.org/TR/2000/REC-xml-20001006/.

[13] Cover, R. (1998) *XML and Semantic Transparency*. OASIS, rev. November 24, 1998. Disponible en: http://www.oasis-pen.org/cover/xmlAndSemantics.html [consultado el 5 de octubre de 2004].

[14] Deytel y Deytel (2001). *Cómo programar en Java*. Ed. Prentice Hall.

[15] Eve, M. y Andaloussi J. (1996) Developing *SGML DTDs: From Text to Model to Markup*. Upper Saddle River, NJ: Prentice Hall.

[16] Froufe, A. (2000). *Java 2 Manual de usuario y tutorial*. 3ª. ed. Ed. Alfa Omega.

[17] González, O. (2001). *XML*. Ed. Anaya Multimedia.

[18] Hanna, P. (2001). *JSP Manual de Referencia*. Ed. Mc Graw Hill.

[19] Haque, I. and O'Connor, B. (2002). *J2ME Enterprise Development*. Ed. John Wiley & Sons.

[20] Hunter, J. y Lagoze, Cl. (2000) *Combining RDF and XML Schemas to Enhance Interoperability between Metadata Application Profiles*. Queensland: DSTC, University of Queensland, Disponible en: http://archive.dstc.edu.au/RDU/staff/jane-hunter/www10/paper.html [consultado el 14 de septiembre de 2002].

[21] Horstmann C. y Cornell, G. (2002) *Java 2 características avanzadas*. Ed. Prentice Hall.

[22] Ioannides, D. (2000) *XML Schema Languages: beyond DTD*. Library Hit-Tech, Vol. 18, núm. 1, pp. 9-14.

[23] Keogh, J. (2003). *Manual de referencia J2EE*. Ed. Mc Graw Hill.

[24] Koch, N., Baumeister, H., Mandel, L. (2000) *Extending UML to Model Navigation and Presentation in Web Applications*. En: Modeling Web Applications, Workshop of the UML´2000. Ed. Geri Winters and Jason Winters, York (England).

[25] Kontio, M. (2002), *Professional Mobile Java*, Ed. It Press.

[26] Mamad A., Bordead S., Cioroianu A., Hart J., Jung , E. y Writz D. (2004). *Java y XML. Referencia para programadores*. Ed. Anaya Multimedia.

[27] Marchall, B. (2000) *XML by Example*. Indianapolis: Que.

[28] Martín B. (2002) *Tratamiento y difusión en Internet de información jurisprudencial mediante tecnologías XML: Aplicación al caso del Tribunal Constitucional* [Tesis doctoral]. Dpto. de Biblioteconomía y Documentación, Universidad Carlos III de Madrid.

[29] Martín, B. y Rodríguez D. (2000) *Estructuración de la información mediante XML: un nuevo reto para la gestión documental.* En: VII Jornadas Españolas de Documentación, Bilbao: Universidad del País Vasco, pp. 113-123.

[30] Michael, Y. (2000), *Aprenda XML ya*. Ed. Mc Graw Hill.

[31] Nogales T., Martín B., Arellano M.C. (2003) *Informática, Derecho y Documentación. Experiencias y posibilidades de aplicación de los lenguajes de marcado de texto (SGML, HTML y XML) a los documentos jurídicos*. En: Memorias XVI Encuentro sobre Informática y Derecho, Madrid: Instituto de Informática Jurídica, Facultad de Derecho, UPCO (en prensa).

[32] Ortiz C. y Giguere, E. (2002). *Mobile Information Device Profile for Java 2 Micro Edition*, Ed. John Wiley & Sons.

[33] Tremblett, P. (2002). *Instant Wireless Java with J»ME*. Ed. Mc Graw Hill.

[34] White, J. y Hemphill (2002), *Java 2 Micro Edition*, Ed. Manning.