Geração de Conteúdos para Plataformas Móveis

Filipe Correia[1], Joel Varanda[1], João Correia Lopes[2], Alexandre Sousa[3]

[1] ParadigmaXis, Av. da Boavista, 1043, 4100-129 Porto.
http://www.paradigmaxis.pt
{filipe.correia,joel.varanda}@paradigmaxis.pt
[2] Faculdade de Engenharia da Universidade do Porto, R. Dr. Roberto Frias, 4200-465 Porto.
http://www.fe.up.pt
jlopes@fe.up.pt
[3] Instituto Superior da Maia, Av. Carlos Oliveira Campos, 4475-690 Castelo da Maia.
http://www.ismai.pt
avs@ismai.pt

Resumo Recentemente têm aparecido novos serviços na Web que disponibilizam informação personalizada eventualmente em dispositivos móveis, que vão muito para além dos tradicionais conteúdos devolvidos em HTML. A construção e manutenção de sistemas de média e grande dimensão que disponibilizem esses serviços, implica custos elevados. Este artigo descreve o trabalho levado a cabo no sentido de facilitar a construção e manutenção de aplicações Web, suportando a apresentação de conteúdos em múltiplas plataformas: PC, PocketPC e WAP. A arquitectura do sistema desenvolvido baseia-se na utilização de uma cadeia de filtros SAX que aplica modificações sucessivas aos conteúdos recolhidos de diversas fontes, sendo algumas destas modificações, transformações XSL. Os dados das diversas fontes são agrupados numa representação intermédia TipML, um dialecto de XML definido no âmbito deste trabalho.

1 Introdução

A World Wide Web tem, tradicionalmente, fornecido informação e serviços sobre HTML [6], tendo este vindo a ser o formato de representação por excelência neste meio. Uma nova geração de serviços de informação personalizada e serviços móveis, entre outros, têm vindo recentemente a disponibilizar conteúdos de modo bem aceite pelo público. No entanto, a construção de tais sistemas implica custos elevados de desenvolvimento e manutenção, nomeadamente quando possuem média ou grande dimensão.

Um problema encontrado frequentemente por quem desenvolve aplicações com uma interface Web, centra-se na necessidade de a tornar acessível pelo maior número possível de utilizadores. Esta necessidade passa, muitas vezes, pelo suporte oferecido a múltiplas plataformas de modo a aumentar a abrangência de difusão.

Hoje em dia existem já variadíssimas plataformas com a capacidade de consultar informação disponível na Web e, com o aumento dessa variedade, maior

se torna a complexidade de desenvolvimento de aplicações de modo a que todas estas plataformas sejam suportadas; em última análise, o desenvolvimento poderá mesmo passar pela elaboração de sítios Web distintos, apropriados a cada uma das plataformas. Como resultado, os custos de desenvolvimento e manutenção tornam-se maiores do que seria de esperar, uma vez que se multiplicam por cada plataforma suportada.

Um outro problema consiste na dificuldade em separar os dados e o modo como estes são apresentados. Quando é conseguida eficazmente, esta separação, torna-se bastante útil, uma vez que permite que cada uma das componentes da aplicação (dados e apresentação) possa ser alterada, facilmente, sem que a outra seja muito influenciada.

Este trabalho tem por objectivo facilitar a construção e manutenção de aplicações Web, suportando a apresentação de conteúdos em múltiplas plataformas: PC, PocketPC e WAP.

Para tal, foi projectado e construído um sistema, seguindo uma metodologia de programação literária, que permite o desenvolvimento de aplicações Web separando em camadas aplicacionais diferentes, a representação dos dados e o modo como estes são apresentados. Para além de HTML e JSP, foram utilizadas sobretudo tecnologias ligadas ao XML (eXtensible Markup Language): XSL, WML, HTML, SAX, DOM.

Na construção e manutenção dos programas desenvolvidos, foi utilizada a ferramenta *Open Source* de suporte à aplicação da programação literária, *dotNoweb* [13].

2 Representação intermédia de um documento

Inicialmente o uso de HTML tinha o objectivo de marcar informação de modo a definir um documento genérico, a ser apresentado na Web. Deveria ser dada ênfase aos elementos que definem o documento (como títulos, parágrafos, etc) e deveria ser dada pouca importância à formatação que essa informação teria junto do utilizador (como cores, tipos de letra, etc). Contudo, face à necessidade que existia, e existe, de especificar o modo como a informação é apresentada ao utilizador, a apresentação de HTML passou a ser rígida, centrada apenas na formatação de informação e não na sua estruturação como documento, independentemente do modo como é apresentado.

Existem várias razões que impedem que um determinado documento HTML, publicado na Web, seja disponibilizado a qualquer plataforma. Factores como capacidade de processamento, área de ecrã disponível e largura de banda limitam as capacidades de dispositivos a partir dos quais seria interessante aceder ao documento em questão.

Na tentativa de suportar o crescente número de plataformas existentes com a capacidade de acesso à Web, surge um aumento da complexidade de desenvolvimento de aplicações que tenham por objectivo ser visualizadas independentemente do dispositivo utilizado [10].

Se o objectivo inicial do HTML fosse mantido existiria uma linguagem mais generalista, que podia ser interpretada correctamente por qualquer plataforma. Deste modo, determinado documento poderia ser consultado em qualquer das plataformas pretendidas sem que, para isso, fosse necessário construir e manter uma versão desse documento para cada uma delas. Contudo, segundo esta abordagem, não existiria qualquer tipo de informação relativa ao modo como a informação seria apresentada.

Na tentativa de estabelecer um compromisso entre a definição de um documento e a definição do modo como é apresentada determinada informação, foi construído um sistema de desenvolvimento de aplicações Web. Este sistema separa, em camadas aplicacionais distintas, a estruturação da informação de um documento, do modo como este será apresentado em cada plataforma. Assim, é permitido a um utilizador do sistema, configurar informação textual e gráfica de modo a que esta possa ser visualizada e permita a interacção em várias plataformas.

Por forma a obter uma representação de um documento independente da plataforma onde este irá ser apresentado, foi definida a linguagem intermédia TipML (*Tip Markup Language*). Esta representação intermédia é baseada num dialecto de XML, que define o posicionamento e ordem de apresentação de fracções de conteúdos, sem incluir informação relativa ao modo como esses conteúdos deverão ser apresentados. Esta linguagem genérica, nesta altura, tem por objectivo suportar as linguagens alvo HTML e WML [11].

Tomando como ponto de partida um documento escrito em TipML, é possível adaptá-lo às necessidades particulares de cada dispositivo. A linguagem é suficientemente rica para ser adaptada a um formato de documento estruturalmente complexo (como uma página HTML), mas também suficientemente simples, para ser adaptada a um documento com muito pouca informação estrutural (como uma página WML).

Esta abordagem permite construir apenas um documento, disponibilizando-o a qualquer das plataformas suportadas. Assim, a manutenção de uma dada aplicação multiplataforma é bastante simplificada; basta ser modificado apenas o documento TipML, em vez de ser necessário modificar todas as versões (para todas as plataformas) do documento. Uma segunda vantagem é a separação entre o conteúdo do documento e o modo como este é apresentado. Assim, a adaptação do documento, a cada formato suportado, pode ser alterada conforme seja desejado.

3 Negociação e geração de conteúdos

Para que um dado documento Web possa ser visualizado na plataforma que lhe acede, em dado momento, é necessário conhecer de que dispositivo se trata exactamente para, deste modo, a informação ser devolvida num formato que o dispositivo possa tratar adequadamente [4].

O protocolo HTTP (*Hyper Text Transfer Protocol*) fornece alguns mecanismos de apoio à negociação de conteúdo. Em cada pedido HTTP, efectuado por um cliente Web, são enviados, no cabeçalho desse mesmo pedido, os tipos de conteúdo (através de MIME (*Multi-Purpose Internet Mail Extensions*) que esse navegador Web aceita [16].

A partir desta informação é seleccionado o tipo de conteúdo a ser apresentado, de acordo com o dispositivo alvo correspondente. Assim, a selecção do tipo de conteúdo a apresentar é feita automaticamente conforme o tipo de dispositivo que aceda ao documento. Por cada dispositivo suportado existe uma folha de estilo XSL [1], preparada para adaptar um documento escrito em TipML, num documento específi co ao dispositivo alvo.

Cada plataforma possui especifi cidades muito concretas. As características, das plataformas consideradas, que constituem maiores limitações à adaptação dos conteúdos a cada plataforma, são apresentadas na Tabela 1.

	PC	**PocketPC**	**WAP**
Formatos	HTML	HTML	WML
Imagens	gif, jpeg	gif, jpeg	WBMP
Dimensões ecrã	800x600, 1024x768	240x268	96x45, 101x65

Tabela1. Capacidades de cada plataforma

A representação em HTML para ser interpretada por um PC, é a que mais se aproxima de um documento escrito em TipML. Assim, torna-se mais simples fazer a adaptação do conteúdo para esta plataforma, tornando, no entanto, mais complexa a adaptação para outras plataformas.

No caso de se procurar uma apresentação em PocketPC, poderá também ser usado HTML, mas neste caso, é necessário obter uma representação diferente da anterior, por forma a não comprometer a usabilidade da aplicação construída. A difi culdade de maior relevância, neste caso, é a representação da mesma quantidade de informação num espaço de ecrã de menores dimensões.

Quando se pretende que a aplicação construída seja acessível via WAP, a adaptação do conteúdo torna-se um pouco mais complexa, do que no caso anterior. Não só é necessário ter em atenção que a área de ecrã disponível é bastante

limitada, como também é necessário ter em consideração as diferenças do formato de codificação da informação textual (WML) e gráfica (WBMP).

4 Arquitectura Lógica do sistema

A arquitectura lógica do sistema de construção de aplicações Web, desenvolvido por forma a atingir os objectivos enunciados, é constituída por três camadas: a camada de dados, a camada de integração e a camada de apresentação.

A arquitectura do sistema baseia-se na utilização de uma cadeia de filtros SAX que aplica modificações sucessivas aos conteúdos recolhidos de diversas fontes [12], sendo algumas destas modificações, transformações XSL.

Os conteúdos podem ser obtidos de bases de dados ou outras fontes de dados e são depois convertidos para XML pela camada de dados sendo, neste formato, introduzidos na cadeia de filtros SAX.

A camada seguinte de integração, trata de colocar os dados das diversas fontes nas partes adequadas de documento que constitui uma representação intermédia em TipML.

A camada de apresentação transforma o documento TipML de modo a que este fique de acordo com as capacidades do dispositivo cliente. Após a negociação de conteúdo está identificada a plataforma de apresentação e basta, por isso, incluir uma folhas de estilos XSL para o formato pretendido.

Após todas as modificações os conteúdos estão adaptados ao dispositivo a que se destinam. Deste modo, o sistema desenvolvido tem a capacidade de efectuar a saída para vários formatos de dados, de acordo com as características de cada uma das plataformas suportadas.

5 Arquitectura Física do sistema

Por forma a ilustrar a arquitectura física do sistema desenvolvido, apresenta-se, usando a linguagem de modelação UML, o Diagrama de Componentes e o Diagrama de Distribuição, respectivamente na Figura 1 e na Figura 2.

O problema em questão não sugeriria, à partida, a utilização de um *parser* de XML uma vez que o objectivo consistia em ler a informação de uma base de dados, para além de ela poder ser obtida de ficheiros XML. Contudo, é possível emular a leitura de dados XML como se de um *parser* (que suporte DOM ou SAX) se tratasse, sendo, na realidade, esses dados lidos de outra fonte.

O modo de funcionamento de um parser SAX, baseado em eventos, favorece a sua adaptação de modo a servir os propósitos pretendidos. Um factor favorável ao uso do SAX neste âmbito, é o processamento sequencial da informação e possibilidade de modularização de diversas componentes pelo recurso a uma

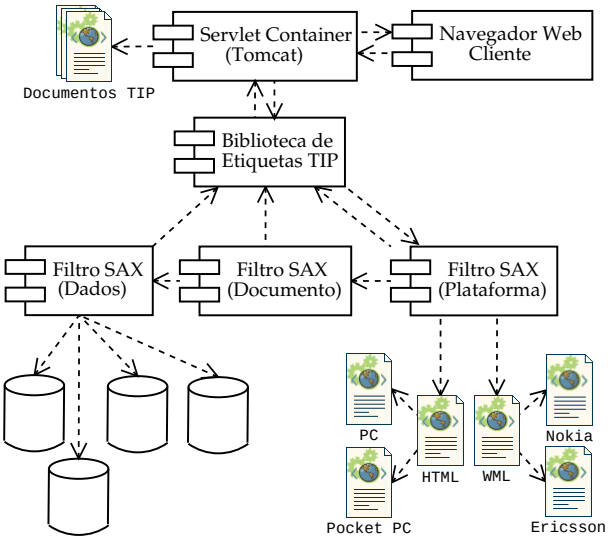

Figura1. Diagrama de Componentes do Sistema

cadeia de filtros. Pelo uso de filtros SAX os dados são alterados em sequência, imediatamente após a leitura e análise gramatical de determinada secção de informação.

O DOM revela-se ideal para armazenar informação quando o modelo de dados pretendido é já uma árvore. Assim, com o seu uso, é possibilitado o acesso aleatório aos dados, mas para este caso não é vantajosa a sua utilização, uma vez que os dados poderão ser acedidos pela sua ordem original. Assim, o tempo e recursos computacionais adicionais necessários para a construção em memória da árvore DOM não seriam, de modo algum, compensados pela possibilidade de acesso aleatório aos dados.

Foi, assim, criado um filtro SAX que se encarrega de fazer a interacção com as bases de dados, obtendo a informação lá contida, como se da leitura de um ficheiro XML se tratasse, tornando-se este processo completamente transparente para o resto da aplicação. A este *leitor* de informação podem ser adicionados os filtros que se pretendam de modo a modificar, de algum modo, os dados em XML dele obtidos. O acesso às base de dados é efectuado por intermédio de uma *API* disponibilizada por um *Driver* JDBC. Será este *driver* que efectuará a interacção propriamente dita com as bases de dados.

Face aos propósitos da aplicação foram utilizados três filtros (ver Figura 1 [7]), sendo o primeiro, como já referido, um filtro de leitura de conteúdos da base de dados. Este filtro, uma vez tendo iniciada a aquisição de informação, irá

lançar eventos correspondentes a uma estruturação em XML dos dados obtidos. Tais eventos são tratados pelo filtro seguinte na cadeia.

O segundo filtro, tem o papel de combinar os conteúdos recebidos (provenientes da base de dados relacional) com a definição do posicionamento de componentes no documento (proveniente do documento *TIP* a ser processado). A combinação referida é feita por meio de uma folha de estilos, já que o filtro SAX não é mais que um transformador XSL. O resultado final da transformação consiste num documento escrito em TipML.

O passo seguinte é dado pelo terceiro e último filtro e consiste na adaptação do documento, agora escrito em TipML, para a plataforma pretendida. Para tal, os eventos do segundo filtro são capturados e, sendo este também um filtro transformador de XSL, é-lhes aplicada uma segunda folha de estilo, específica ao formato de saída em questão (HTML ou WML). A cada folha de estilo que é possível aplicar, são associadas ainda configurações específicas à plataforma em que o formato em questão será utilizado.

O documento *TIP* referido na Figura 1 é uma página JSP adaptada para servir alguns propósitos específicos, como será referido mais à frente. No entanto, este facto permite que a cadeia de filtros já apresentada seja gerida por uma biblioteca de etiquetas de JSP [14].

Estas etiquetas são usadas na definição de certos blocos de informação num documento *TIP* e a elas está associada a lógica que cria cada filtro e faz com que estes funcionem em conjunto. Todos os blocos de informação por elas marcados num documento *TIP* constituem, de algum modo, informação necessária ao funcionamento da cadeia de filtros SAX.

Pelo uso de uma biblioteca de etiquetas JSP é excluída do documento *TIP* qualquer lógica relacionada com a cadeia de transformação e o facto de um documento *TIP* se tratar, no fundo, de uma página JSP possibilita ainda que os conteúdos das etiquetas sejam gerados dinamicamente por meio de *Scriptlets*, inclusive a partir de parâmetros externos (uma vez que o conteúdo das etiquetas acabará por ser código XML, muitos dos princípios aplicados em [9] são aplicáveis).

O funcionamento do sistema, é desencadeado por um pedido por parte de um navegador Web. Esse pedido será dirigido ao *Tomcat*, que servirá simultaneamente como servidor Web e *Servlet Container*. Como tal, o *Tomcat* obtém o documento *TIP* especificado no pedido, lê dele os dados necessários e inicializa a cadeia de filtros SAX, para que o documento seja transformado no formato apropriado.

Quando construída uma aplicação em *TIP*, os componente executáveis de todo o sistema são distribuídos como apresentado na Figura 2.

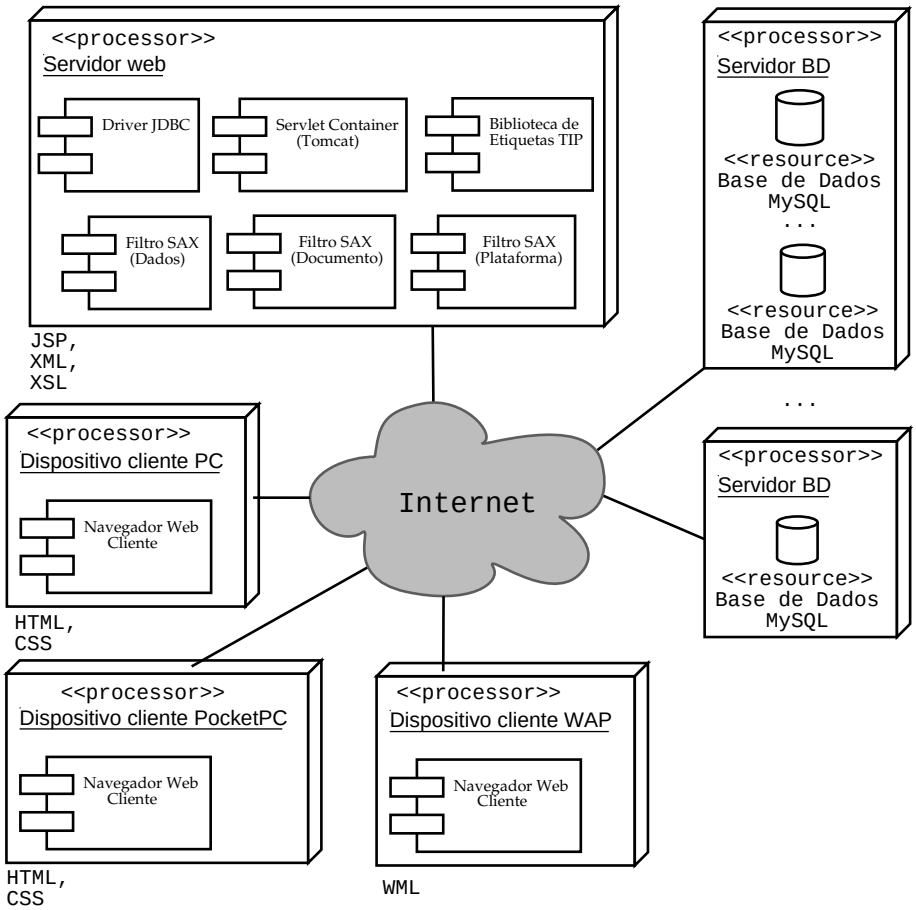

Figura2. Diagrama de Distribuição do Sistema

O *Tomcat* encontra-se a correr numa máquina específica para o efeito onde, num caso limite, poderiam estar também as bases de dados a utilizar. Contudo, esta não é a configuração mais vulgar, uma vez que a possibilidade, oferecida pelo sistema, de serem utilizados dados de várias fontes distintas, permite que as bases de dados se encontrem distribuídas. Assim, todos os nós estão ligados á *Internet*, por intermédio da qual são efectuadas as passagens de informação entre os nós. Os dispositivos cliente existentes constituem nós de *hardware* distintos e, cada um deles, poderá requisitar documentos *TIP* ao servidor Web. Os dispositivos permitidos, como foi já referido, poderão ser de natureza variada (dispositivos WAP, PC e PocketPC), correndo, em cada um deles, um navegador Web apropriado, com a capacidade de requisição de documentos ao servidor.

6 Detalhes de Implementação

Partindo da definição da arquitectura apresentada anteriormente, foi necessário desenvolver todos os componentes necessários ao funcionamento do sistema. As tecnologias utilizadas foram JSP, XML, XML e a plataforma Java [8].

Para acesso às bases de dados foi utilizada a informação do corpo da etiqueta <tip:data>. Tal informação está estruturada segundo um dialecto de XML bem definido, como apresentado no exemplo seguinte.

```
[...]
<tip:data>
   <server>
      <name>localhost</name>
      <database>
         <name>BD</name>
         <query record="{}noticia"{}>
            select * from noticias order by date desc
         </query>
      </database>
   </server>
</tip:data>
[...]
```

A lógica de negociação e adaptação de conteúdo foi implementada com recurso a uma biblioteca de etiquetas JSP, que irá residir no *Tomcat Web container*.

O JSP foi também utilizado para gerar a folha de estilos XSL [15][2] que é usada para transformar o documento da representação intermédia, numa representação com apresentação dos conteúdos para os formato pretendido: WML ou HTML com folhas de estilos CSS externas [3].

Posteriormente é aplicada uma transformação XSL específica à plataforma de visualização alvo; existe por exemplo, uma para HTML num PC e outra para HTML num PocketPC.

7 Teste e exploração

Por forma a avaliar a viabilidade da arquitectura proposta, a adequação da linguagem intermédia desenvolvida e os resultados das apresentações obtidas em HTML e em WML foram construídas duas aplicações de teste. Estas aplicações baseiam-se em aplicações existentes em funcionamento na Web permitindo, deste modo, a validação das funcionalidades disponibilizadas e a detecção de aspectos que podem ser melhorados em desenvolvimentos futuros.

A primeira aplicação baseia-se em `http://www.digito.pt`, que se destina à divulgação de notícias sobre ciência e tecnologia, de onde foram obtidas as notícias inseridas numa base de dados construída para o efeito. A notícia a apresentar é passada como parâmetro a uma consulta à base de dados, sendo aplicadas de seguida as cadeias de filtros SAX por forma a obter uma representação em HTML para PC, uma representação em HTML para PocketPC e, finalmente, uma representação em WML que foi visionada num emulador. O conteúdo foi apresentado com resultados bastante satisfatórios nas várias plataforma alvo.

A segunda aplicação baseia-se em `http://www.newportex.pt`, que se destina à divulgação de notícias sobre têxteis, e permitiu concluir que os resultados são melhores, nomeadamente em termos de navegação, quando os conteúdos são pensados directamente em TipML ao invés de serem adaptados de páginas existentes, como foi o caso. Tal como com a aplicação anterior, esta aplicação de teste também expôs diversos melhoramentos que foram, ou estão em vias de ser, incorporados no TipML.

A título de exemplo, apresentam-se nas Figuras 3, 5 e 4 ilustrações, retiradas do funcionamento do sistema desenvolvido, quando aplicado à segunda aplicação referida.

8 Conclusões e Trabalho Futuro

Os resultados obtidos permitem-nos concluir que a arquitectura proposta e implementada satisfaz globalmente os objectivos, possibilitando a publicação, em formatos apropriados a vários dispositivos, de informação proveniente de uma ou mais fontes, nomeadamente bases de dados relacionais.

Por forma a melhorar a eficiência, as folhas de estilo XSL estáticas podem ser compiladas com JAXP [5] e as folhas de estilo XSL geradas dinamicamente a partir de JSP poderiam ser mantidas numa cache.

Em termos de imagens, a apresentação pode ser muito melhorada através de redimensionamento, uma vez que nesta altura apenas é decidido se são incluídas ou não no documento final, ou mesmo através de conversão de formatos, por exemplo de GIF para WBMP para WAP.

Figura3. Interface do Newportex em HTML

Figura4. Interface do Newportex em WML

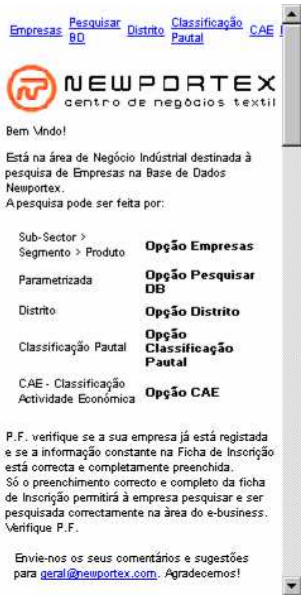

Figura5. Interface do Newportex em HTML para PocketPC

Outros melhoramentos em vista incluem o suporte para sistemas legados, envolvendo a tradução para TipML de páginas existentes em HTML, e a extensão a novas plataformas através da construção de novas folhas de estilo.

Referências

1. Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Tony Graham, Paul Grosso, Eduardo G utentag, Alex Milowski, Scott Parnell, Jeremy Richman, and Steve Zilles. Extensible Stylesheet Language (XSL) version 1.0. Technical report, W3C, October 2001. http://www.w3.org/TR/2001/REC-xsl-20011015/.
2. Iftikhar Ahmed. Enabling web applications for wireless devices. Technical report, Sun Microsystems, Inc., 2001.
3. Bert Bos, Håkon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading style sheets, level 2 - CSS2 specification. Technical report, W3C, May 1998. http://www.w3.org/TR/1998/REC-CSS2-19980512.
4. K. H. Britton, R. Case, A. Citron, R. Floyd, Y. Li, C. Seekamp, B. Topol, and K. Tracey. Transcoding: Extending e-business to new environments. Technical Report 1, 2001.
5. Eric M. Burke. *Java and XSLT*. O'Reilly & Associates, Inc., September 2001.
6. James Clark and Steve DeRose. HTML 4.01 specification. Technical report, W3C, December 1999. http://www.w3.org/TR/html401/.
7. Jim Conallen. *Building Web Applications with UML*. Addison-Wesley, December 1999.
8. Elliotte Rusty Harold. *Processing XML with Java*. Addison-Wesley, 2001.
9. Marshall Lamb. Generate dynamic XML using JavaServer Pages technology. Technical report, ibm.com/developerworks, December 2000.

10. Benoît Marchal. *Applied XML Solutions*. Sams Publishing, August 2000.
11. OMA. Wireless markup language specification version 1.3. Technical report, WAP Forum, Ltd, February 2000. http://www1.wapforum.org/tech/documents/WAP-191-WML-20000219-a.pdf.
12. O'Reilly. *SAX2*. O'Reilly & Associates, Inc., January 2002.
13. Alexandre Valente Sousa. Literate programming applied to software maintenance and teaching computer science. *Perspectivas XXI*, 4(8):101–120, nov 2001.
14. Sun. JavaServer pages, 2002. http://java.sun.com/products/jsp/.
15. Thierry Violleau. eMobile: A sample end-to-end application using the java 2 platform enterprise edition - part 2. Technical report, Sun Microsystems, Inc., December 2000.
16. W3C. HTTP request fields, May 1994. http://www.w3.org/Protocols/HTTP/HTRQ_-Headers.html.