MIDDLEWARE SERVICE SUPPORT FOR MODERN APPLICATION PRESENTATIONS

Boyan Bontchev[1], *Sylvia Ilieva*[2]

Abstract: Nowadays, with steadily increasing number of various markup languages and their dialects, application developers need more often to rewrite the presentation in order to support more client devices. Java programmers using JSP for dynamic generation of the presentation layer have to create for given content different Java Server Pages (JSP)´s for showing it in different HTTP, WAP and Palm client. Instead of rewriting the same JSP several times, authors demonstrate an approach of separation of content and presentation form, by means of XSLT transformation over XML content. The client device type is detected by a server-side detector, which uses the type for choosing the correct XSLT for the correspondent XML content generated by a JSP. Moreover, for non-XML browsers, the detector invokes a transformer, which generates on-the-fly any-ML presentation using both the XML and XSLT.
Key words: XML, XSLT, JSP, mobile devices

1. INTRODUCTION

Nowadays, the most popular access to Web is via HTTP Web browser. Millions of users surf through the Web space connecting through their Windows PC, Macintosh and Linux/Unix workstations. In the same time, the number of Web surfers using mobile phones and PDA (such as Palm Pilot series) is steadily increasing. The prognostics of W3C (World Wide Web Consortium) foresee in year 2002 the share of wireless device usage will reach 75% from the total numbers of Web users [1]. Delivering and managing Web content for wireless devices is a challenge because the transformation of content originally created for PCs into format compatible with wireless devices is extremely difficult. Web sites written in HTML are not compatible with wireless devices and must undergo some form of transformation. A number of standards such as HTML, Compact HTML, Handled Device Markup Language (HDML), i-mode, Wireless Markup Language (WML), eXtensible Markup Language (XML) and their respective variations have been developed.

XML is a new markup language developed from W3C. It is aiming at responding to different requirements for the future of HTML. On one side, people need more tags. For example both mathematicians and chemists want tags for formulas, but those tags are different. On other side, developers want fewer tags. There is a need of simple markup language because the small devices, like Palm Pilot, are not powerful enough to process HTML pages. To solve this dilemma XML is doing two basic changes in HTML:

- it does not define any tags in advance
- it is more strict.

The XML applications can be classified in the following two types:

- document applications - manipulate information, which is designated mainly for people
- data applications - manipulate information, which is designated mainly for software consumption.

[1] Chair of ICT, Sofia University, Bulgaria
[2] CLPP, Bulgarian Academy of Sciences, Acad. G. Bonchev St., Bl. 25-A, Sofia 1113, Bulgaria

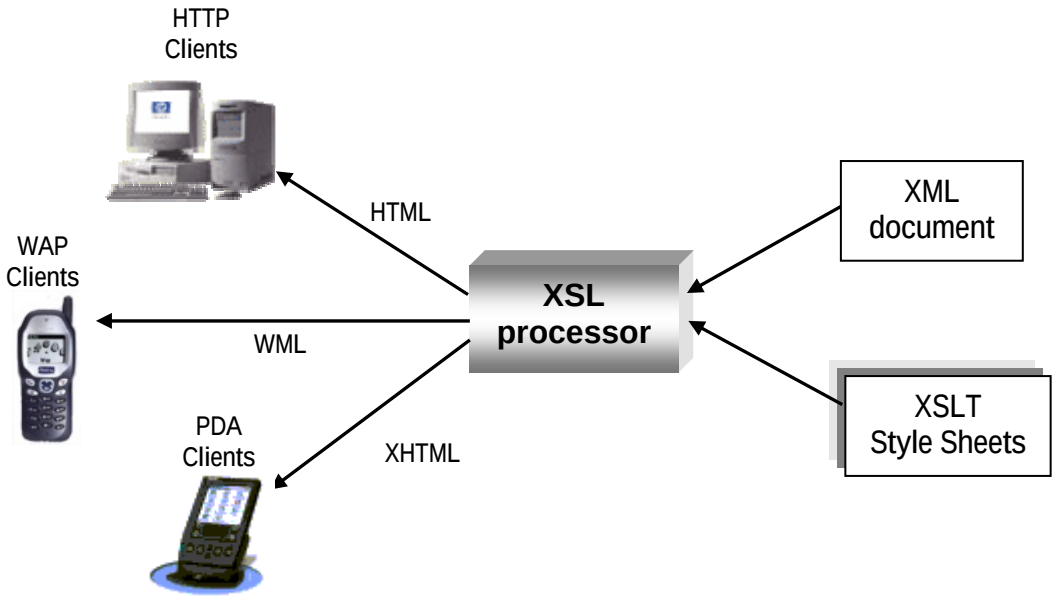

Fig. 1. Conversions of a XML document into several markup's through different XSLT style sheets

XML is valuable not because it is markup language, but because it is a standard language. W3C has developed several standards, which complete XML. One of these additional standards is "Stylesheets" and they define how XML documents have to be presented on display, on paper or on editor. XML supports two languages for stylesheets: XSL (XML Stylesheet Language) and CSS (Cascading Style Sheets). XSL is more powerful that is way we pay attention to it. To use XSL we need XSL processor. There are a lot of XSL processors, for example LotusXSL or from Microsoft. They are just software components that correspond to the XSL standard. [Figure 1]

XSLT (eXtensible STyle sheets for Transformation) is a language, which defines transformation of XML documents. It takes one XML document and transforms it in another XML document.

One of decisions to solve the specific needs of small devices (like mobile phones and Palm Pilots) can be the usage of XHTML, which is XML simpler version of HTML or in other words it is combination of XML and HTML. XHTML is based on HTML but has XML syntax. Also, it is designed to be modular as it is expected that small devices will use only subset of the recommendation.

The challenge lies in creating content for wireless devices that is suitable in format, content and design, and represents an added-value services to end users. The decision is in XSL. It is not dificult to support different browsers and platforms if the original document is keeping in XML and it is transforming in appropriate XHTML subset with XSLT. The conversion may be to HTML or WML or any markup language (fig. 1). The main advantage of XML in this case is that it is concentrated to the structure of the document, which makes it independent from the transporting environment.

The paper is organized as follows: section 2 reviews the common architecture of modern Java server-side applications, and shows the integration XSL transformation inside that architecture. Section 3 presents how XML/XSL is used to generate dynamically different presentation layer according the agent type, and how this idea fits to the middleware architectural later. Section 4 concludes the paper showing the advantages of the proposed approach.

2. MIDDLEWARE ARCHITECTURE OF JAVA SERVER-SIDE APPLICATION

Java server technologies like servlets, JSP (Java Server Pages) and EJB (Enterprise Java Beans) are widely used for creation of to interactive Web pages because of the advantages of Java as a programming language. Developers are attracted by paradigms as "write once, run anywhere" and, specially for JSP usage, by the effective separation of presentation from content. Here, we provide a review of common n-tire Web architecture used widely in modern Java server-side applications and frameworks [5]. The architecture main principles follow the popular Model-View-Controller (MVC) design pattern and are referred as JSP Model 2 [4].

Figure 2 presents a global view on this architecture. The MVC pattern is implemented by separation of the *data model* (relational DB, file system, LDAP server, etc.) from the *controller* (a dispatcher servlet) and from the presentations (i.e., the *views*) done entirely in JSP. The dispatcher is a controller servlet which clients' HTTP requests and calls the proper Action Handler responsible to retrieve and pass to corresponding business components the request parameters. The mapping between the action specified in the HTTP request and its correspondent action handler is provided by configurations files (usually, in XML). The action handlers access request, session and application contexts and, based on that information, verify the user input and invoke business components to perform certain operations for that action. The business components have access to the data model through a persistency manager (collection of data access classes, in some cases with a object-relational mapping) and to a resource pool through a resource manager. The resource files are used for internationalization, setting of application parameters, etc. Business components retrieve data content from the model within Java beans, which are used afterwards by JSP's for dynamic generation of given presentation markup.

Finally, the dispatcher calls correspondent view handler with its specific parameters using again mapping information (action-view handler) from the configuration file. In case of error or unexpected situation the dispatcher forwards the control towards a different view handler. The view handler invokes a JSP which populates data from the beans on proper positions inside a markup pattern and thus produces an any-ML presentation. JSP is not aware where data comes from, i.e. it does not make any difference between database values and internationalization strings read from property files.

As it is obvious from the figure, for generation of different presentation (see fig. 1) we need to redesign the JSP's. Thus, there will be developed several pools of JSP for HTML, XTML, WML, etc., which will reside in different Web applications. Moreover, each mobile phone treats presentation tags in different manner, so we have to design different WML generators. Next chapter shows how to resolve the problem using XML/XLS transformations.

3. USING XML-XSLT TRANSFORMATION FOR DYNAMIC GENERATION OF VARIOUS PRESENTATION FORMS

The basic idea for XML usage in presentation tire is to divide data content from presentation tags, which are glued together. This make design of new presentation more difficult and complex. XML/XLS transformation tries to resolve it by separation between content data and presentation data. Now, instead of generating a markup presentation, JSP produce XML files which incorporate XML content data. For each client device, there are developed a set of XML transformations defining how the correspondent markup will be generated.

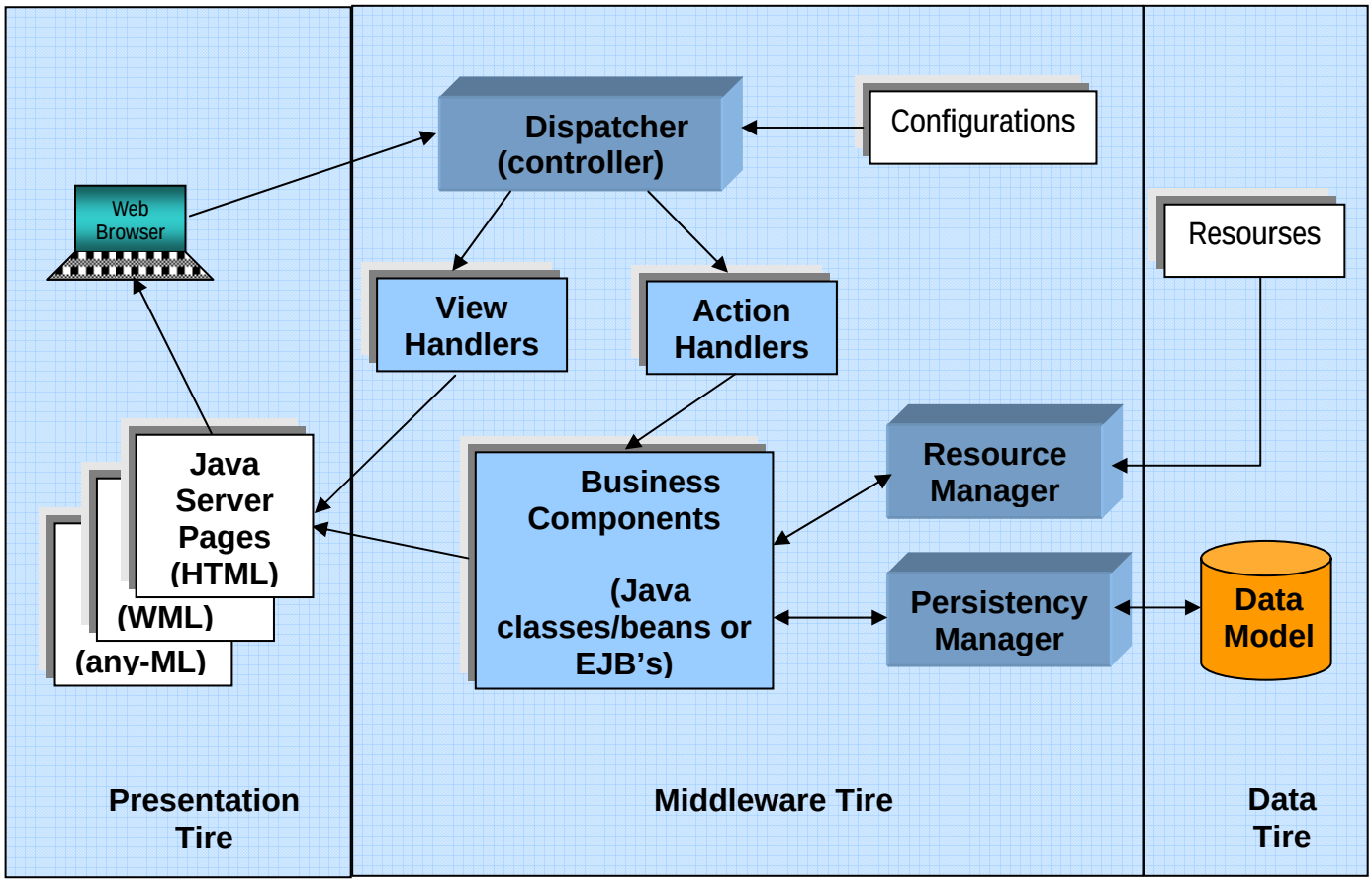

Fig. 2. Basic architectural view of a common
Java server-side application

Figure 3 shows how new components are added to the common architecture to comply the idea explained over. Before accepted by the dispatcher, HTTP requests passed through an agent detector which determines the browser type and reports it to a presentation manager. Invocation of JSP is again done by a view handler but now the JSP generates XML with data content. It is accepted by the presentation manager, which chooses the proper XSLT schema for transformation of XML into presentation corresponded for the detected client.

The XML/XSLT transformation can be done on the client side in cases, where client devices support XML/XSL, e.g. for browsers like MS Internet Explorer 5.* and Netscape Navigator 6.*. Otherwise, the presentation manager sends the par XML/XSLT to a transformer, which realizes the transformation on the server side. The decision where to do the transformation is taken by the presentation manager and is based on the detected agent type. Thus, for clients with no XSLT support the application has to produce the conversion. They exist a plenty of XSLT engines doing this conversion in a pretty rapid way, most of them available as freeware.

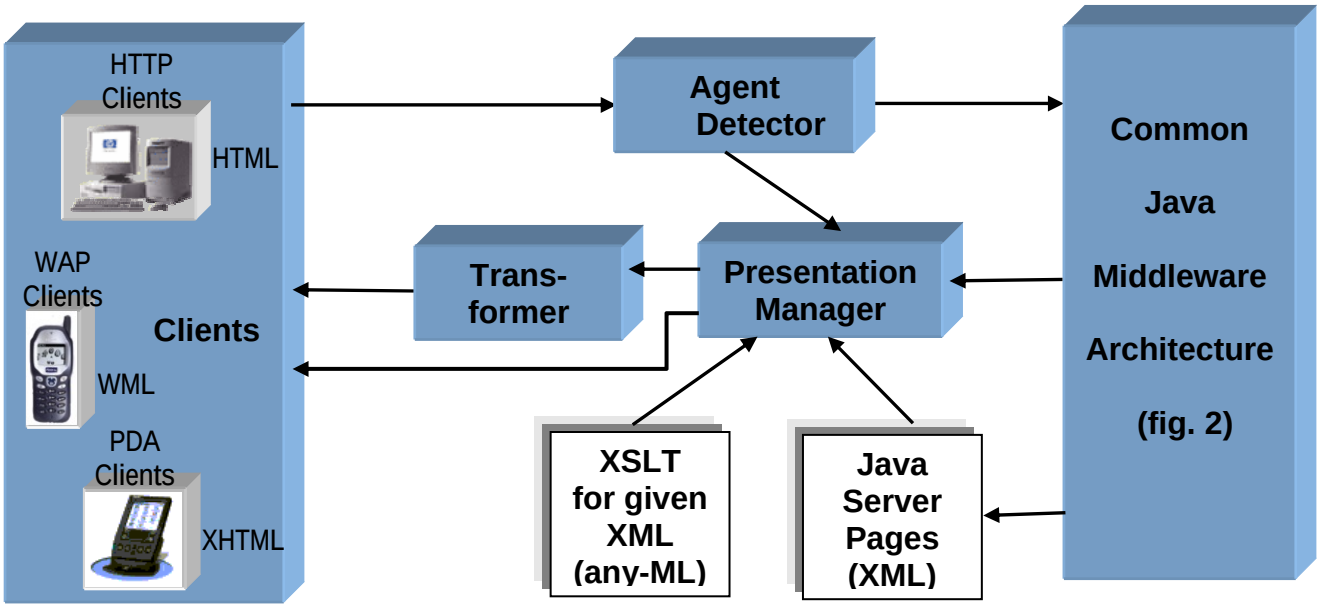

Fig. 3. XSL transformation extension to
the common Java server-side applications

4. CONCLUSIONS

Two alternatives enable Web site applications to be read by wireless Internet devices. The first rewrites the application, requiring the content provider to build at least two or three different sites for the different wireless markup languages (e.g., WML, HDML, i-mode). In the future, this will be even more tedious due to the growing number of handsets with different attributes.

The second alternative relies on some kind of middleware that enables the application to be read by any kind of handset without rewriting the site and, therefore, less development costs. It was discussed in the presented paper.

The main benefits of the suggested approach are: separation of data content from presentation and less development cost. The effectiveness comes from the fact that when companies have to assure support of various client devices for given application, they do not have to redesign (to program) the JSP but only have to do specify new XSL transformations.

REFERENCES

1. Marshal, B. *XML by Example*. Que, 2000, 571 p.
2. XSL Transformations (XSLT) Specification, Version 1.0, W3C Working Draft, 21 Apr 1999, http://www.w3.org/TR/1999/WD-xslt-19990421
3. Pappo, N. Middleware Bridges Internet, Wireless. Telecommunications online, May 2000, http://www.telecoms-mag.com/issues/200005/tcs/middleware.html
4. Seshadri, G. Understanding JavaServer Pages Model 2 architecture. JavaWorld, December 1999, http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html
5. The Jakarta Project: Struts User Guide. http://jakarta.apache.org/struts/userGuide/index.html