February 2005

# Approaches for Device-Independent Content Delivery to Mobile Devices

Bozena Jankowska
*European University Viadrina*

In: Ferstl, Otto K, u.a. (Hg) 2005. *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*; 7. Internationale Tagung Wirtschaftsinformatik 2005. Heidelberg: Physica-Verlag

ISBN: 3-7908-1574-8

# Approaches for Device-Independent Content Delivery to Mobile Devices

**Bożena Jankowska**
European University Viadrina

*Abstract: The Web and enterprise information systems are gradually increasing their reach to a wide range of mobile devices. These devices, however, support different formats and possess device-specific features. This raises the problem of creating separate presentations for each device type or, at least, for each class of devices. Instead of designing many pages, content authors may use a device-independent technique which will help them to render suitable content automatically. This paper introduces the principles of device-independent content delivery and outlines various mechanisms and technologies for generation of device-independent applications. It also describes a developed Mobile Interfaces Tag Library (MITL) – a solution that automatically produces appropriate content depending on the detected device characteristics.*

*Keywords: device independence, mobile delivery context, mobile computing, adaptability of mobile content*

## 1 Introduction

With the increasing popularity and falling prices for various types of wireless devices such as PDAs, SIMpads, Palmtops or mobile phones, the needs and expectations of consumers regarding access, availability and consumption of information are continuously evolving. More and more users take advantage of the enticing promise of mobile computing - to obtain data through different access mechanisms from any location and at any time.

According to the Trend Research [Lehn02, pp. 8-9] mobile users are mainly interested in reading messages such as e-mails or SMS (81%), accessing ERP functionality (72%), viewing Intranet information from their own company (68%), checking typical Web information such as news, stock prices, travel or weather services (57%) and communicating with different databases (44%)[1]. Established vendors of information systems such as SAP [SAP01] or Baan [Forr03] recog-

[1] In this survey multiple answers were possible.

nized the possibility of added business value through mobile access to desktop- or Web-based applications and developed mobile front-ends to their systems. Worldwide, most of the online newspapers, information services, market trading services and travel agencies also offer slimmed, mobile versions of their Web sites.[2] It is estimated that in 2004 at least half of the Fortune2000 companies will support wireless systems [Dula03]. However, from the German top 500 enterprises listed under http://www.top500.de, in 2003 only 10% had WAP-sites [CeKuNo04, p. 59]. Enterprises without wireless presence probably do not expect to gain additional customers by presenting their business offer in a way that will enable mobile users to access it or the development and maintenance costs for mobile pages are higher than possible benefits from such sites.

Gartner Group predicts that in 2004 nearly 800 million people will use mobile devices and IDC estimates that mobile subscribers will grow from 395 million to 1 billion in 2005 [Dula03]. Despite this rapid growth in the number of mobile users and increasing use of wireless devices in private and professional life, the acceptance of mobile Web sites is still low [Hueb+03, p. 43; Reck01]. The missing approval can be partially attributed to relatively high transmission costs, large download times due to low bandwidths and small screen sizes, but the crucial reason remains the lack of well-designed, high-quality mobile applications. A recent study on 77,506 WML pages showed for example that 19% of them were faulty. 31% of incorrect sites referred to dead links or links that could not be displayed in mobile browsers. 27% could not be shown because they exceeded the allowed deck-size of 1,400 octets. 32% of the sites were not displayed due to various violations of WML-DTD specification [CeKuNo04, pp. 61-62]. In the study, navigation and pagination issues were not analyzed, but they are also considered as typical problems of tailoring Web sites to mobile appliances.

Nowadays, more and more enterprises decide to reach customers equipped with wireless appliances. Consequently, their programmers are forced to develop content that is targeted for various devices and takes into account their characteristics. The efforts and expenses involved in manual adaptation of the content (e.g. by developing many different views of the same data and applying them according to the device-specific presentation features) are unreasonably high. They encompass rewriting applications for various browsers, markup languages and device types, maintaining large code bases and continuous market research on characteristics of novel devices. Ideal solution would therefore consist in creating of only one version of the content for all appliances. Special adaptation software should then take care of a respective presentation which would match the device's capabilities. Although this vision may be very tempting, there are some core questions that have to be addressed by approaches for automatic generation of device-independent content. They encompass the following issues:

2 See the directories of WAP-sites under: http://wapjag.com, http://mobile.yahoo.com, http://www.2thumbswap.com, http://www.mopilot.com.

a) How to express an application independently of the targeted devices?

b) How to adapt device-independent applications to suit device-specific characteristics?

c) How to give authors the possibility to retain some control over the final presentation of their content?

Current Web technologies such as Cascading Style Sheets [W3C04] or XForms [W3C04e] can answer the questions only partially. Therefore developers and researchers worldwide work on new solutions that would enable device-independent content delivery to different devices. The focus of the research is on matching the content with the needs, capabilities and limitations of the delivery environment. A further goal is to maximize the number of devices to which the content can be delivered by proposing authoring techniques that do not apply only to limited set of appliances. Solutions for device-independent content delivery usually fall into one of three broad adaptation categories: client-side, intermediate or server-side adaptation. This paper describes the existing approaches and introduces a server-side adaptation method called Mobile Interfaces Tag Library (MITL) that automatically adapts the content to the features of mobile appliances. It detects devices characteristics using information included in CC/PP profiles or, alternatively, HTTP headers [W3C03] and delivers data in HTML [W3C99], WML [WAP02] and XHTML [W3C03b], taking into account also the supported graphical formats.

The paper is organized as follows: Section two describes the principles for device-independent content delivery. Subsequent section provides an overview of existing methods for the delivery of contextual information with regard to different devices. In section four common technologies and specific approaches for device-independent content delivery are described. Next section introduces the developed Mobile Interfaces Tag Library, provides a detailed description of its functionality and presents an exemplary application. It gives also an overview of the Integrated Development Environment (IDE) for MITL that facilitates the development of MITL-based documents and presents an evaluation of MITL. In last section issues for further research are discussed and some concluding remarks are provided.

## 2 Principles for Device-Independent Content Delivery

To enhance user experience with the content delivered to variety of devices, W3C consortium specified principles for device independence [W3C03]. These principles should not be regarded as strict set of requirements. They should rather serve as guidelines for the design of applications based on existing markup languages, during the development of adaptation tools, in the process of extending existing markup languages or for the evolution of new markups. According to these princi-

ples, content adaptation aspects can be considered from three different perspectives: the user's perspective, the author's perspective and the delivery perspective.

Generally speaking, the user would like to interact with the Web using various devices and via many kinds of access mechanisms. From the user's perspective two most important aspects of content adaptation are the device-independent access to the same functional presentation and device-independent Web page identifiers. Device independent access means that the user is able to obtain equivalent functionality of an application regardless of the device. Obviously, the presentation will not be the same on every device and its quality may vary. The intended function of a page should be associated with only one Web page identifier, and should apply to all presentations obtained from the Web page identifier, no matter what the access mechanism is. For example, a user who enters one URL for a weather service in a browser on his PDA and in a browser on WAP-enabled mobile phone should see the forecasts for a chosen city in form of texts and/or images. On a screen of the mobile phone the temperatures and WBMP images symbolizing the weather conditions will be displayed. On the PDA the user will get colored JPEG pictures and more detailed information about weather due to the larger screen. Navigation structures and pagination will vary but the user will be able to see the weather forecast for the same cities. User experience in form of messages "cannot display image" or "deck's size too large" are considered as delivering a non-equivalent functionality.

From the author's perspective it should be possible to provide a functional presentation, in response to a request associated with a specific Web page identifier, in any given delivery context that has an adequate access mechanism. This principle is simply a restatement from the author's perspective of the principles mentioned above. The adaptation process should provide a presentation that allows a user to successfully access a Web page to get the information or complete the interaction intended. If a functional presentation of an application cannot be delivered due to inherent limitations in the access mechanism, an appropriate error message should be shown to the user. Additionally, it should be possible to provide a customized and harmonized presentation depending on the device capabilities. The author should be able to control the presentation on different devices. It is unrealistic to expect from an author to create different presentation data for every delivery context. Whenever possible, authored source content should be reused across multiple delivery contexts.

Delivery-related adaptation principles encompass the characterization of delivery context and delivery preferences. The term delivery context refers to the set of attributes that are given for particular delivery environment. The adaptation software should be able to associate the characteristics of the delivery context with a request of a particular Web page identifier. Unless the characteristics of the delivery context can be made available to the adaptation process, it will not be possible to know whether a specific presentation of content can be delivered in that context, or how to generate a suitable presentation. It should be possible for a user to provide or update any presentation preferences as part of the delivery context

(context-aware applications). If the user provides presentation preferences, they may be used by the adaptation process to offer a more suitable presentation, after taking into account the constraints of the network and device. The process may allow the user to obtain the most appropriate presentation for their abilities and circumstances.

# 3 Delivery Context

Content adaptation has to be based on information about the delivery context. This can include the relevant features of devices such as supported markup languages or graphical formats, recognized browser types, screen sizes, maximally allowable deck size, etc. Additionally, delivery context may provide data about network's characteristics, user preferences or application-specific parameters such as user's location. Currently, there are six commonly-known standards that help to obtain the delivery context and some further specifications are in progress. The popular approaches include the HTTP headers, the W3C Composite Capabilities/Preference Profiles (CC/PP), the WAP User Agent Profile (UAProf), the SyncML Device Information standard (DevInf), the Media Queries and the Universal Plug and Play Standard (UPnP) [see W3C02 for more details].

Delivery context is usually extracted from the HTTP standard Accept headers [W3C02]. These headers contain information about the character sets, supported media types, languages and content encoding. Additionally, the User-Agent header comprises data about the device manufacturer, the version number, the device hardware and the browser used. Crucial contextual information included in the User-Agent header is not standardized and depends on the good will of device manufacturers.

The Composite Capability/Preferences Profile (CC/PP) [W3C04a] recommended by the W3C consortium delivers contextual information related to device specifications and user preferences. The standard is based on the Resource Description Framework (RDF) [W3C04c] serialized to Extensible Markup Language (XML) [W3C04b]. CC/PP is vocabulary independent and allows using vocabularies which may be optionally described by RDF Schema [W3C04d]. It is composable/decomposable and enables the dynamic creation of context profiles from fragments of capabilities information that are distributed among multiple repositories on the web. A CC/PP profile can describe a number of components (e.g. a mobile browser) by specifying their attributes (e.g. type, version and name of the browser, supported MIME types, its screen size, etc.).

The WAP Forum has defined User Agent Profiles (UAProf ) as part of its WAP specification. [WAP02] UAProf is an implementation of CC/PP for WAP-enabled mobile terminals and is transported with the help of CC/PP headers over HTTP. It

has a two level hierarchy composed of components and attributes with specified vocabulary and can be written in RDF serialized to XML. Profiles using the UAProf vocabulary consist of six components: HardwarePlatform, SoftwarePlatform, NetworkCharacteristics, BrowserUA, WapCharacteristics and PushCharacteristics. UAProf proposes specific transfer syntax for profile and profile differences, as well as resolution rules for default values of properties and difference values. First commercial implementations of UAProf are already available for WAP proxies as well as browsers and the Mobile execution Environment group within the European Telecommunications Standards Institute (ETSI) has agreed to adopt UAProf for asserting device capabilities in future wireless appliances [SaHj01, p. 42].

The SyncML initiative (http://www.syncml.org) strives to develop a common synchronization protocol for data exchange between servers and devices such as mobile phones, PDAs and desktop PCs. Before the synchronization can take place, the devices have to exchange information about their capabilities. For this purpose SyncML Device Information standard (DevInf), implemented in XML, is used. The DevInf description delivers data about the device, its data storage and supported extensions as well as accepted content types.

Universal Plug and Play (http://www.upnp.org) is a standard supported by Microsoft. It is targeted at device independent interconnection and uses XML to describe device capabilities. UPnP assumes that the devices will use XML together with XSL [W3C03c] to manipulate the device description and display for example only certain information about its capabilities.

Media Queries [W3C02a] are a standard that is usually processed directly on a client using local delivery context. Media Queries are based on the media types defined in CSS2 specification [W3C04]. In CSS, specific styles can be applied according to a number of named categories of devices, e.g. aural, embossed, handheld, print, tv, etc. The style used to present an element of HTML, XHTML or XML depends therefore on the attributes of the delivery device. With help of the CSS “display” property, it is possible to include or exclude certain elements from the presentation.

For the delivery of mobile context, HTTP headers and CC/PP are the most commonly used approaches [W3C02, SpGo02]. Interestingly, CC/PP standard enjoys increasing popularity among developers, but device vendors are still reluctant to implement it.

# 4 Approaches for Device Independence

## 4.1 Methods for Content Adaptation

Four categories of methods are used for fitting the same content to different categories of devices: scaling, manual authoring, transducing and transforming [cf. Schi+02, pp. 38-40].

Scaling is a relatively old approach that allows scaling down the content in a viewer using some kind of a "fit-to-screen" feature or alternating the portion of page displayed at any given time (e.g. by using zooming functionality). Scaling can be applied for high-resolution color displays (e.g. Pocket PC) equipped with browsers with vertical and horizontal scrolling capability. It reduces the readability of a site and makes the interaction and navigation more difficult. Zooming was applied for viewing information at multiple scales in different devices by Pad++ already in 1994 [BeHo94].

Manual authoring is a very work-intensive method consisting in creation of separate content that fits particular devices. This approach leads to well-designed pages with good layout and navigation possibilities but is only feasible if the amount of pages is small and enterprises can afford high development and maintenance costs.

Since manual authoring is not a cost-effective solution, techniques that allow automatic device-independent authoring become more and more popular. Transducing is one of such approaches. It translates HTML and images into supported formats (e.g. WML, XHTML) using a proxy. This method falls into the intermediate adaptation category, where an intermediary controls the adaptation process. A mobile device requests a Web site through a proxy such as Mobile Google (http://mobile.google.com) or Wingman [Fox+98]. The proxy retrieves the content associated with a particular URL, transduces it into appropriate formats and compresses or converts images to supported types. In this approach, adaptation abilities are limited because intermediaries usually lack specific information about the pages. Proxies cannot guarantee that the transcoded content will be usable on the device which required it. For example, compressed WAP pages cannot exceed the size of 1.4 KByte and the transcoding proxy has to split up the HTML pages so that they fulfill this requirement. This may lead to unusable navigation structures or damaged layouts.

Transforming is a method that gained increased recognition in last years. In this approach content is changed to the appropriate format and the transcoding proxy modifies the content structure (e.g. replacing tables with lists or altering navigation elements) and the way of interacting with the content. Transcoding proxies use different techniques to render a Web page on a small-screen device. Existing approaches apply various transcoding heuristics [HwSeKi03], semantic annota-

tions [HoAbOn03] or text summarization techniques [BuMoPa01]. For example the m-Links system completely changes the Web page layout [Schi+01]. It splits one page into multiple small sub-pages and displays it as a collection of links, which lead to appropriate textual descriptions.

## 4.2 Exemplary Adaptation Approaches

Recently, the number of approaches for tailoring data to various devices increased rapidly, proving the growing interest in device-independent content delivery. The proposed methods belong to three categories: intermediate, client-side and server-side adaptation.

Some examples of intermediate adaptation that requires an additional middle layer called proxy were described in the previous section. Client-side adaptation is not a commonly used method because the adaptation code needs an access to device's capabilities. As an example of client-side adaptation Cascading Style Sheets (CSS) are usually named. CSS allow styling of HTML, XHTML, XML and Scalable Vector Graphics [W3C03a]. Authors can use appropriate style sheets to define different presentation rules for various media types or to omit certain fragments of pages, for example on mobile devices.

Server-side adaptation, in which a server is in charge of content delivery, gained the widest acceptance of authors. The server can respond to a user's request and deliver radically different content, depending on the obtained delivery context. The differences relate to the amount of information, presentation style, navigation and layout.

A common server-side adaptation method consists of retrieving data from an information system in XML format and converting them on the server-side to the appropriate markup language with eXtensible Style Sheet Language Transformations and an XSLT processor. In this technique content is separated from presentation and the same data can be presented in different ways, depending on a style sheet. However, if the view changes, every style sheet has to be updated separately.

Cocoon [MoAs02] is another server-side adaptation approach. It is a publishing framework written in Java language to provide content in multiple markup languages such as HTML, XHTML, WML, etc. It is based on the concept of complete division between three layers: document content, style and logic. Content, logic and style are separated out into different XML files, XSL transformation capabilities are then used to merge them. The Cocoon model allows sites to be highly structured and well-designed, reduces duplication and site management costs. It permits different presentations of the same data depending on the requesting client. The main disadvantage of this framework is slow transformation of

XML files into desired markup language and separate Web identifiers for each markup language.

Another approach for delivering information to different devices in a device-independent way is the User Interface Markup Language (UIML) [Abra+99]. UIML permits a declarative description of a user interface and separates the interface from the application logic and device type. UIML document can be mapped to any type of user interface (e.g. Java AWT, WML, HTML). The developed language is not an open-source project and the technology for generating specific output cannot be evaluated.

To address the needs of web developers who build applications for a variety of devices, the W3C has invented XForms [W3C04e]. XForms is characterized by the separation of content, presentation, and logic, similarly to Cocoon. The content is the fundamental data model, the presentation describes the appearance on different platforms, and the logic defines dependencies between form elements and additional functionality. XForms enables the validation of user input at the client within the browser. This reduces client-server interactions and server load. Furthermore, XForms provides means to calculate values from user input at the client. Most mobile devices, however, do not currently support special browsers for viewing XForms.

Mobile Application Markup Language [ScKo03] is a language based on XHTML syntax. It defines main elements of the document, which are then written in XML and converted to the end format using XSLT. The architecture of MAML is based on Java servlets with chains of filters. Filters are responsible for identifying the required content type and for transforming the XML documents with help of XSLT. The main drawback of the MAML approach is scarce device context and missing fragmentation method for creating user-friendly mobile interfaces.

Renderer Independent Markup Language (RIML) is part of a Consensus project, and it aims at the development of highly-usable mobile applications [Gras+02]. The documents are written in RIML, which is based on the XForms standard and then transformed by an adaptation engine to a specific device at runtime. The adaptation of large data-sets and navigation structures to devices with limited interaction capabilities are also taken into account by this approach. This method is relatively new and is still under development.

Further adaptation approaches include new languages such as Dialog Description Language [SpGo02], specific architectures such as MVC [John+02] or Struts (http://jakarta.apache.org/struts) or extensions to existing languages such as JavaServer Faces [Sun03].

# 5 Mobile Interfaces Tag Library Approach

## 5.1 Description of the Mobile Interfaces Tag Library (MITL)

The developed library for device-independent content delivery to mobile devices is a server-side approach, especially useful for designing pages with medium complexity. It takes care of the pagination and creation of appropriate navigation elements and avoids the high complication level of languages such as UIML or RIML. It is based on JavaServer Pages feature called JSP tag libraries [ShChaRy01]. Tags are Java classes which are interpreted on the server side. They encapsulate functionality that can be used from a JSP page. Examples might be conditional logic, database access, internationalization, etc.

The Mobile Interfaces Tag Library (MITL) generates appropriate markup elements in WML, XHTML and HTML depending on device context. The library was developed taking into account the requirements for device-independent content delivery specified by the W3C Consortium. MITL is easy to use and enables rapid prototyping of applications designed for different devices. Authors have to include only one tag for displaying a particular element in three different formats. Applying this tag library divides the work of developing multiple views of the same data between JSP programmers responsible for the functionalities encapsulated in tag libraries (presentation logic) and developers of pages who are in charge of the layout and maintenance of Web sites.

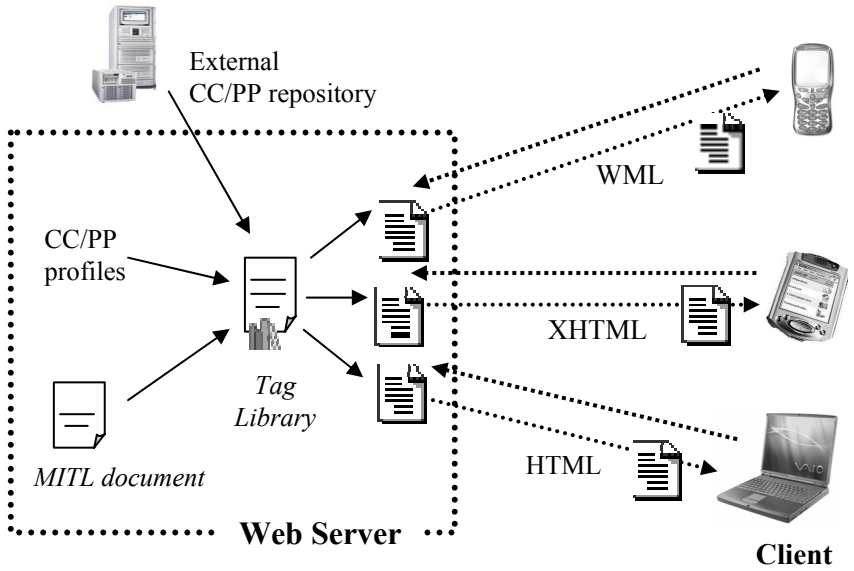



Figure 1: Document processing with MITL

Figure 1 illustrates the overall structure of request processing with help of MITL. A client sends a request for a JSP page and delivers information about device capabilities in form of HTTP headers or CC/PP profiles. A Web server (e.g. Apache Tomcat) looks for a document which was requested and processes it on the server side. The MITL tag library retrieves information about the device context (supported formats, screen size, etc.) using the data from HTTP headers or extracting relevant information from a local or external repository of CC/PP profiles. Then it converts each tag in the document to the appropriate element in a particular format. For example the "<mitl:doc title="Welcome">" tag is converted into "<HTML> <HEAD> <TITLE> Welcome </TITLE> </HEAD> </BODY> [...] </BODY> </HTML>" for browsers supporting HTML. The information about the Java classes responsible for the conversion is stored in XML format in the tag library descriptor files (TLD). The output is sent back as a response to the client and displayed in the appropriate browser.

The MITL library contains the following tags:

- DocTag – depending on the detected browser type generates the root elements for HTML, WML or XHTML, the title of a page and specifies the link to an appropriate CSS stylesheet in XHTML and HTML. Alternatively, it is able to generate CSS stylesheets automatically, basing on the retrieved device characteristics. Since the applications are served by the server, the tag also sets a correct MIME type for each device (e.g. text/html for html, text/xml+xhtml for XHTML, etc.). Information about the devices' features is retrieved in a DocTag which is always the first tag used in a page. The tags are able to communicate with other tags on the same page. Therefore, the DocTag detects the browser type as well as its capabilities and all other tags use this information for content adaptation. This decreases the number of data exchanged between the client and server and enables faster processing of requests. The DocTag determines the form of presentation depending on the relevant features of a device. It uses the CC/PP profiles to obtain the delivery context, or, if no CC/PP are provided, the HTTP headers. For processing of CC/PP profiles, DELI is used [Butl02].
- WmlYes/WmlNoTag - WmlYesTag indicates that elements enclosed in the WmlYes tags should be included in the output for browsers supporting WML. WmlNoTag enables the elements enclosed by these tags to be excluded in the output for browsers supporting WML and presented only in browsers with XHTML or HTML support.
- DeckTag - produces a WML card or a deck of many cards. For each card a card title and card name is generated. It also offers the possibility to specify the tasks (such as „forward", „back", „accept",etc.) or the amount of time after which the user is redirected to a new page.

- FragmntOn, Title, Desc, Details – these tags identify that automatic fragmentation of text according to the screen size should be performed and categorize basic elements of the text such as its title, general description and details.
- PaginationTag – is in charge of pagination of large data sets and emulates different pagination and indexing styles (e.g. "previous item-next item"- style).
- TableTag – draws a table with such optional attributes as background color, border size as well as values for cellpading and cellspacing elements. For WML it also gives the number of columns.
- RowTag – draws a row in a table. The developer can specify a color for a row, its height and width.
- CellTag – draws cells in a row with specified height, width and background color.
- HyperlinkTag – produces link elements in HTML, WML and XHTML.
- ImageTag – creates appropriate images depending on device features. By default an image is converted to the first format supported by a browser (e.g. WBMP, JPG, PNG). The conversion takes into account the information about device's displaying capabilities such as the size of the screen, the supported color depth, etc. The developer can also explicitly specify to which format the image should be converted for a particular device as well as specify exactly the characteristics of the image such as its resolution and size. If no graphical format is supported, textual description is rendered. For image conversion a class library called JIMI (http://java.sun.com/developer/products/jimi) is applied. This is the Sun development kit for reading and writing several formats including GIF, JPEG, TIFF, PNG, BMP and ICO.
- FormTag and FieldTag – generate a form with different field tags such as input text boxes, text areas, buttons, etc.
- ListTag, ListElement Tag – generate lists and their elements.
- LConvTag – converts lists into tables and tables into lists.
- TConvTag – converts one table into another with different characteristics such as different number of columns and rows.

The defined tags offer a web application developer the possibility to control how data is processed in back-end Java components without including any Java code in the JSP page. The tags allow furthermore for generation of completely different presentation layers. Content transformed by the MITL can be in form of regular text or XML data. In the second case the XML source has to be transformed with the help of XSLT in order to extract specific data or modify them. MITL tags can be combined with any Java code or markup-specific tags.

## 5.2 Integrated Development Environment for MITL

The Integrated Development Environment (IDE) for MITL was developed in order to facilitate the programming of MITL applications (cf. figure 2). It was programmed with help of Oracle JDeveloper 10g [Orac04] using Java AWT and Swing components. It can be therefore used on a wide variety of computing platforms, working the same way no matter where it runs. The IDE enables inexperienced programmers to write an application without knowing the MITL structure in detail and validates the syntax, decreasing the possibility of application's errors. For all tags the attributes are displayed explicitly in form of panels with text boxes, which reduces significantly the time needed for writing a MITL document, because tag elements do not have to be learned by heart. For some of the tags a set of the allowed attributes is provided in a drop-down list (cf. figure 2). Additionally, obligatory attributes are marked with red color and an asterisk - the application will prompt user to complete the tag if these are left empty.

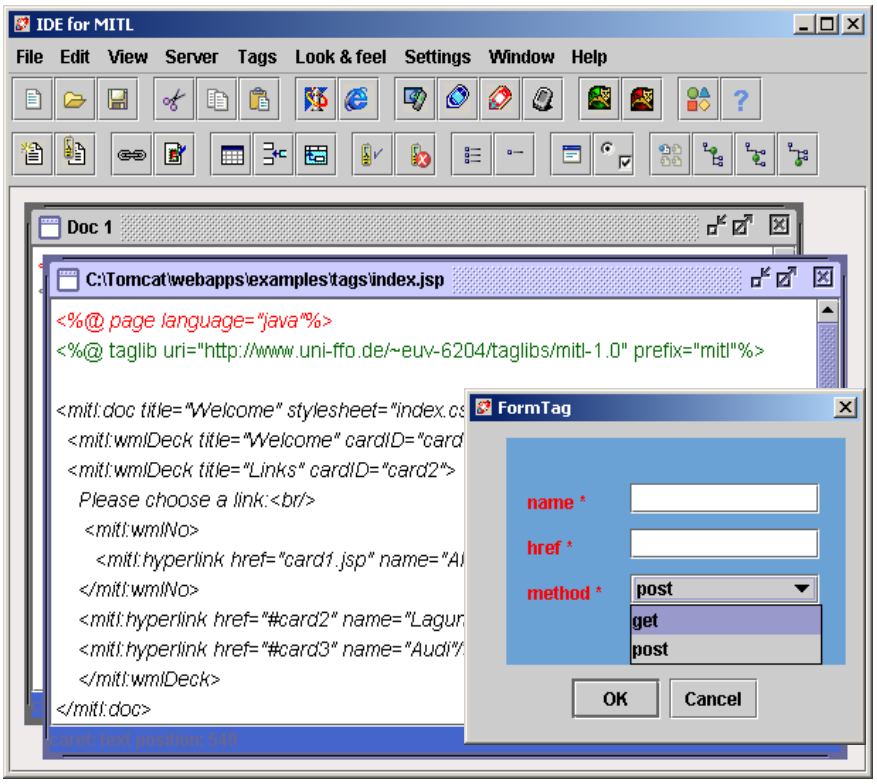



Figure 2: Integrated Development Environment for MITL

The IDE is integrated with some mobile simulators, Internet Explorer and Netscape and provides the possibility to view the resulting output on a specified device. The environment is highly configurable – the user can choose the server on which the libraries should run as well as the simulators. It also possesses the standard functionalities such as cut/copy/paste, text formatting, windows manipulation or operations on files.

The icons representing available tags are grouped regarding their functionality on one taskbar (e.g. the TableTag, RowTag, CellTag are placed together and distinctly separated from other tags so that it is easy for a user to locate and apply them), all other icons (e.g. responsible for starting the web server, viewing the file in various browsers, saving it, etc.) can be found on another taskbar. All preprogrammed tags and features of the developed IDE can also be accessed by using the drop-down menus. Extensive help with introduction on how to use the environment, examples of simple and sophisticated documents written using the MITL, and answers to typical problems are provided, so that even a new user can quickly write an own MITL page.

## 5.3 Example of MITL Application

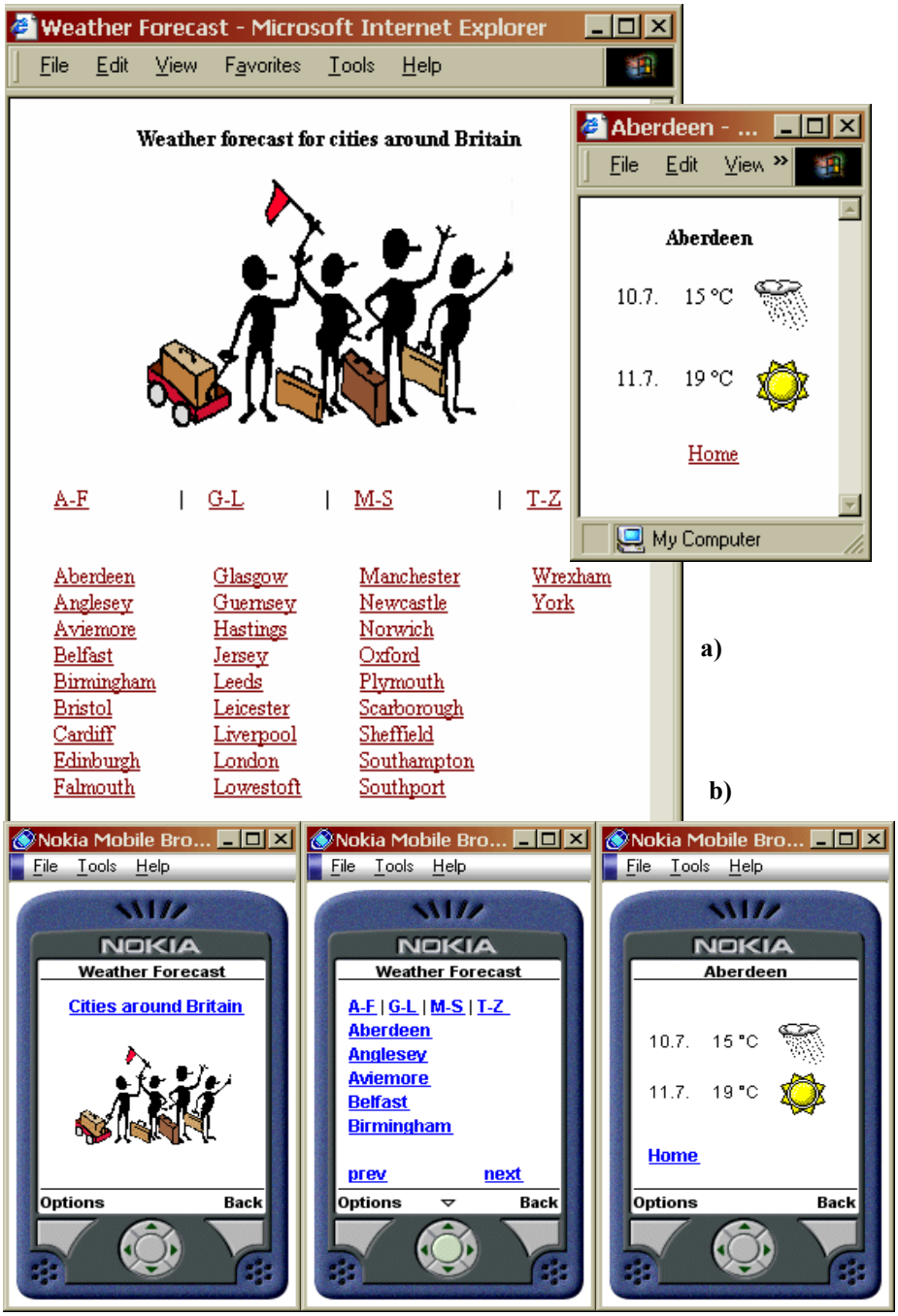
Figure 3 demonstrates a simple Web page of a weather forecasting service generated with the MITL library and rendered differently in various browsers. The content of one big HTML page was split into many pages for devices supporting XHTML and WML with regard to the screen size. Appropriate navigation elements were generated on each page in order to obtain detailed characteristics of the displayed item. Depending on supported image formats, the images were converted to appropriate types such as GIF or WBMP.

The data was retrieved from a database with help of DBTags library [Apac04]. This library can be applied to build a connection with the database and to retrieve appropriate data relying on the JDBC mechanism. Instead of writing Java code embedded in a JavaServer Page, author can make use of specific tags for building a connection to database, sending queries or retrieving results.

The pagination was performed with the PaginationTag, whereas the number of displayed elements (cities) depends on the screen size of a device and calculated number of visible lines. Two different styles were applied for pagination: the "previous item – next item"-style for mobile devices and alphabetical division of items (e.g. cities starting with letters from A to F were grouped together).

The HTML page is split into subpages on mobile devices due to the use of the DeckTags. They are ignored in the case of Web browsers and allow to split one page into a chosen number of smaller pages (so-called cards). Content may be also included or excluded in certain browsers with the WMLYes/WMLNoTags. Images were converted to appropriate formats (WBMP and GIF) and scaled using

JIMI features implemented in ImageTags. Although this example is very simple, it shows some of the possibilities of the MITL library.




**c)**

Figure 3: The same page presented in different browsers on different devices. a) Internet browser supporting HTML b) Nokia Mobile Browser Simulator 4.0 with a browser supporting XHTML c) Nokia 5100 with a browser supporting WML and WBMP

## 5.4 Evaluation of MITL

The Mobile Interfaces Tag Library was designed for creating simple and intermediately complex mobile applications from scratch. Alternatively, it can be used to convert existing HTML pages into wireless sites by extracting data from Web pages using wrappers [KuTr02] but the entire control over the conversion process is left to authors. MITL enables communication with different databases (with DBTag library) and provides support for structural tasks such as iteration and conditionals, text inclusion or internationalization (JSTL library).

The most important advantage of MITL is, however, the fact that the language can be used by non-professional programmers and occasional users because the syntax is similar to other, commonly-known markup languages. The user does not have to know the elements of MITL since the pages can be generated using the IDE. Compared with other languages, e.g. UIML or RIML, the user is able to develop device-independent applications after a short introductory course, because of limited number of elements implemented and transparent structure of the language. As the history of SGML and HTML showed, these are very important features for the wide acceptance of new solutions.

Most mobile pages possess a relatively simple structure. Content authors do not use for example tables as layout elements and do not squeeze many different elements in one row. They rather place them in a sequential manner and the user has to scroll down to see more content [SAP00; Jone+99]. Therefore creating one page for various devices in different markup languages does not require complicated transcoding heuristics or automatic summarization techniques [BuMoPa01;

HwSeKi03]. Problems arise if existing Web sites should be converted to mobile pages because their navigational structures, amount of content per screen and layout strongly diverge. It is much easier to assemble a Web page from mobile cards then convert existing HTML pages to wireless sites because of insufficient separation between logic, data and presentation. MITL is not applicable for automatic conversion of Web pages because it lacks mechanisms for recognizing document structure, generating summaries or extracting keywords from large fragments of text.

Another drawback of MITL results from insufficient support of delivery context by device manufacturers. MITL does not use its own database with profiles of devices; instead of this it relies on the information about device characteristics delivered automatically by the mobile appliance. Since many producers do not offer such support but usually deliver data about the phone type, it would be reasonable to create own database with device profiles or to use available collections of profiles such as Wireless Universal Resource (WURFL)[3]. Still, the quality of such databases strongly depends on the amount of information collected.

# 6 Summary and Further Work

The MITL library described in this paper should facilitate the development of mobile applications for different mobile devices. It provides an extensible framework for automated, device-independent content delivery. Since MITL is based on Java and is an open-source approach, it can be further enhanced by experienced tag programmers and easily extended with new tags. The author has already started to develop tags for converting the content to J2ME (Java 2 Mobile Edition) code [Topl02].

According to the survey conducted amongst students from the European University Viadrina (inexperienced in programming for mobile devices) and some developers from different programming firms, the approach is easy to learn and use, extensible and faster than manual authoring. It does not, however, support features such as inheritance of properties, templates reusing or automatic user's input validation. These shortcomings will be eliminated in the second, more sophisticated approach – the Mobile Content Adaptation Language (MCAL) built upon XForms concept. It will provide separation of content, presentation and logic. The content will be in form of XML, the presentation elements will describe the appearance on different devices and the logic will define dependencies between form elements. The adaptation from the language to specific end-formats will be performed by Java servlets with chain of filters. It will be possible to convert the information to

[3] See http://wurfl.sourceforge.net/index.php for more details.

the same formats as in the first approach but this method will be intended for rather experienced developers. For authors who would like to develop device-independent content for mobile devices without spending a lot of time on framework configuration and learning a completely new language, MITL is highly recommended.

# References

[Abra+99] Abrams M. et al.: UIML: An Appliance-Independent XML User Interface Language. In: Proceedings of the 8th International World Wide Web Conference, Toronto, 1999, pp. 1695-1708.

[Apac04] Apache Software Foundation: DBTags Library. http://jakarta.apache.org/taglibs/doc/dbtags-doc/index.html, 2004. Download 2004-06-28.

[BeHo94] Bederson, B.; Hollan, J.: Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. In: Proceedings of the 7th Annual Symposium on User Interface Software and Technology, Marina del Rey, California, 1994, pp. 17 – 26.

[BuMoPa01] Buyukkokten, O.; Garcia-Molina, H.; Paepcke, A.: Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices. In: Proceedings of the 10th International World Wide Web Conference, Hong Kong, 2001, pp. 652-662.

[Butl02] Butler, M.H.: DELI: A Delivery Context Library for CC/PP and UAProf, External Technical Report HPL-2001-260. http://www-uk.hpl.hp.com/people/marbut/DeliUserGuideWEB.htm, 2002, Download 2004-06-23.

[CeKuNo04] Ceska, T.; Kuhlins, S.; Nösekabel, H.: Programmgestützte Analyse von WAP-Sites. In: Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI) 2004, Essen, 2004, Band 3, pp. 54-64.

[Dula03] Dulaney, K.: Predicts 2004: Mobile and Wireless. http://www4.gartner.com/ DisplayDocument?doc_cd=118831, 2003, Download 2004-07-01.

[Forr03] Forrester Research: Choosing The Right Mobile Enterprise Application. http://www.forrester.com/go?docid=16663, 2003, Download 2004-07-03.

[Fox+98] Fox, A. et al.: Experience with Top Gun Wingman, a Proxy-Based Graphical Web Browser for the 3Com PalmPilot. In: Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, The Lake District, 1998, pp. 407-424.

[Gras+02] Grassel, G. et al.: Definition and Prototyping of a Renderer-Independent Markup Language for Enterprise Applications, W3C DIAT Workshop, St. Leon-Rot, 2002.

[HoAbOn03] Hori, M., Abe, M., Ono, K.: Extensible Framework of Authoring Tools for Web Document Annotation. In: Proceedings of the International Workshop on Semantic Web Foundations and Application Technologies, Nara, 2003. http://www-kasm.nii.ac.jp/ SWFAT/PAPERS/SWFAT20R.PDF. Download 2004-09-23.

[Hueb+03] Hübsch, G.; Springer, T.; Schill, A.; Spriesterbach, A.; Ziegert, T.: Systemlösungen für die Entwicklung adaptiver Anwendungen für mobile und ubiquitäre Infrastrukturen. In: HMD, Praxis der Wirtschaftsinformatik, Heft 229, 2003, pp. 42-55.

[HwSeKi03] Hwang, Y.; Seo, E.; Kim, J.: Structure-Aware Web Transcoding for Mobile Devices. In: IEEE Internet Computing, 7 (5), 2003, pp. 14-21.

[John+02] Johnson, M. et al.: Designing Enterprise Applications with the J2EE Platform, Addison-Wesley Publishing Company, Boston, 2002.

[Jone+99] Jones, M., Marsden, G., Mohd-Nasir, N., Boone, K., Buchanan, G.: Improving Web Interaction on Small Displays. In: Proceedings of the 8th World Wide Web Conference, Toronto, 1999, pp. 51-59.

[KuTr02] Kuhlins, S.; Tredwell, R.: Toolkits for Generating Wrappers - A Survey of Software Toolkits for Automated Data Extraction from Websites. In: Proceedings of the NetObjectDays Conference, Erfurt, 2002, pp. 184-198.

[Lehn02] Lehner, F.: Mobile Business, Mobile Services, Forschungsbericht der Universität Regensburg, Nr. 49. http://www.uni-regensburg.de, 2002, Download 2004-06-30.

[MoAs02] Moczar, L.; Aston, J.: Cocoon Developer's Handbook, SAMS Publishing, Indianapolis, 2002.

[Orac04] Oracle: JDeveloper 10g, http://otn.oracle.com/products/jdev/index.html, 2004, Download 2004-06-20.

[Reck01] Recklier, O.: M-Commerce: The Next Hype. http://www.themanager.org /resources/M-Commerce.htm#_ToC510089434, 2001, Download 2004-07-01.

[SaHj01] Saryanarayana, L.; Hjelm, J.: CC/PP for Context Negotiation and Contextualization. In: Proceedings of the 2nd International Conference on Mobile Data Management (MDM), Hong Kong, pp. 239-245.

[SAP00] SAP: SAP Interaction Design Guide for WAP Applications. http://www.sapdesignguild.org/resources/wap_guidelines/wap_guidelines.pdf, 2000. Download 2004-10-12.

[SAP04] SAP: SAP Mobile Engine. http://www.sap.co.kr/solutions/mobilebusiness/pdf/50057072s.pdf. Download 2004-09-20.

[Schi+01] Schilit, B., N.; Trevor, J.; Hilbert, D.; Koh, T.: m-Links: An Infrastructure for Very Small Internet Devices, in: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, 2001, Rome, pp. 122 – 131.

[Schi+02] Schilit, B., N.; Trevor, J.; Hilbert, D.; Koh, T.: Web Interaction Using Very Small Internet Devices. In: Computer, Vol. 35, Issue 10, 2002, pp. 37 – 45.

[ScKo03] Scharf, D.; Koch, A.: A New Technology-Driven Approach for Extended Mobile Applications: MAML - Mobile Application Markup Language. In: Proceedings of the 8th International Workshop on Mobile Multimedia Communications (MoMuC), Munich, 2003, pp. 23-27.

[ShChaRy01] Shachor, G.; Chace, A.; Rydin, M.: JSP Tag Libraries, Manning Publications, Greenwich, 2001.

[SpGo02] Springer, T.; Göbel, S.: A Modular Adaptation Framework for Single Source Publishing. In: Proceedings of the IADIS International Conference WWW/Internet 2002, Lisabon. Portugal, pp. 11-19.

[Sun03] Sun Microsystems: JavaServer Faces Technology. http://java.sun.com/j2ee/javaserverfaces, 2003, Download 2004-06-29.

[Topl02] Topley, K.: J2ME in a Nutshell, O'Reilly & Associates, Sebastopol, 2002.

[W3C02] W3C: Delivery Context Overview for Device Independence. http://www.w3c.org/2001/di/public/dco, 2002, Download 2004-06-17.

[W3C02a] W3C: Media Queries W3C Candidate Recommendation. http://www.w3.org/TR/css3-mediaqueries/, 2002, Download 2004-07-10.

[W3C03] W3C: Device Independence Principles, W3C Working Group Note. http://www.w3.org/TR/di-princ, 2003. Download 2004-07-10.

[W3C03a] W3C: Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR /SVG11, 2003, Download 2004-07-03.

[W3C03b] W3C: XHTML 2.0 The Extensible HyperText Markup Language Specification. http://www.w3.org/TR/xhtml2/, 2003, Download 2004-06-25.

[W3C03c] W3C: XSL Transformations (XSLT) Version 2.0. http://www.w3.org/TR/xslt20, 2003, Download 2004-06-23.

[W3C03d] W3C: Voice Extensible Markup Language (VoiceXML) Version 2.0. http://www.w3.org/TR/voicexml20/, 2003, Download: 2004-06-20.

[W3C04] Cascading Style Sheets Level 2 (CSS 2.1). http://www.w3.org/TR/CSS21, 2004, Download 2004-07-12.

[W3C04a] W3C: Composite Capabilities/Preference Profiles Structure and Vocabularies 1.0. http://www.w3.org/TR/CCPP-struct-vocab, 2004, Download 2004-07-12.

[W3C04b] W3C: Extensible Markup Language (XML) 1.0 (Third Edition). http://www.w3.org/TR/REC-xml, 2004, Download 2004-06-23.

[W3C04c] W3C: RDF Primer. http://www.w3.org/TR/rdf-primer, 2004. Download 2004-06-27.

[W3C04d] W3C: RDF Schema Specification 1.0. http://www.w3.org/TR/rdf-schema, Download 2004-07-03.

[W3C04e] W3C: XForms 1.1. http://www.w3.org/TR/xforms-11-req, 2004, Download 2004-07-06.

[W3C99] W3C: HTML 4.01 Specification. http://www.w3.org/TR/html4/,1999, Download 2004-06-26.

[WAP02] WAP Forum: Wireless Application Protocol WAP 2.0, Technical White Paper. http://www.wapforum.org/what/WAPWhite_Paper1.pdf, 2002, Download 2004-06-28.