Handheld Computers for Data Entry: A Data Services Approach Using Java ™ and XML

Scott E. Chapal, Ichauway, Inc., Newton, GA

ABSTRACT

Integrating handheld data entry into a data management system can be accomplished elegantly with Java™, SAS® and XML. The availability of Java on both server and client platforms (including handheld devices) provides end-to-end connectivity in support of a SAS service-oriented architecture. XML is rapidly becoming the lingua franca of structured data representation on the web, but the challenge is to use it appropriately.

This paper describes the use of Handheld computers for data entry using disconnected mode for remote deployment. The form-based data entry provides quality assurance on the handheld through validation mechanisms and other constraints. In order to simplify deployment, the project is using a web services model to deliver form definitions and to retrieve data via XML over wireless.

INTRODUCTION

Providing handheld computers for data entry is a compelling goal, especially for remote users in situations where using a computer is impractical or impossible. Ideally, handheld devices should provide a seamless extension to the information systems they are intended to work with. But, without careful design and planning, the integration can be far from simple. The rapid evolution of the hardware and software used in these devices presents hard choices for IT managers. The pace of this change can make deployment criteria short-lived, rendering decisions obsolete quickly.

Until recently, there were limited options for deploying small handheld computers for serious data acquisition purposes. Constrained by hardware limitations, sub-notebook sized computers were under-powered, clumsy, fragile and idiosyncratic. In recent years however, Personal Digital Assistants (PDA) have combined pen-input screen capabilities with operating systems and software that is optimized for these form-factors. Simultaneously, the ubiquitous demand for cell phones and other wireless devices has provided the momentum to miniaturize components. The result has been the rapid development and evolution of a myriad of small handheld devices. “Mobile Computing”, a term which encompasses devices from laptops to PDA’s to cell phones to pagers, also implies the wired/wireless infrastructure needed to support them.

The introduction of these small digital devices is proceeding at a remarkably frenzied pace and the transformation into new hybridized forms is astonishing. PDA’s now incorporate integrated wireless connectivity as core functionality rather than as a retrofitted expansion. Similarly, cell phones are acquiring improved display, memory and processing capabilities. Clearly, there is an obvious convergence of features occurring in the PDA and cell-phone markets. Striking examples of this convergence are the Handspring Treo or the Nokia 9290 Communicator, both projected to be available by mid 2002. The Treo is essentially a Palm with cell-phone capability and the 9290 is basically a cell-phone with PDA features. Ruggedized PDA’s are also becoming available now (Panasonic Toughbook 01) for outdoor and environmentally demanding applications.

PDA’s and similar devices (Web tablets, ruggedized handhelds, etc.) are the main platforms of interest for mobile data acquisition applications, due to their data input mechanisms, stand-alone potential and general computing capacity. Even though PDA’s are marketed as personal (multi-media) organizers, there is little inspiration required to imagine exploiting this form-factor for serious work. The asymmetrical requirements of 1) data presentation and, 2) data-entry validation, present unique problems on small devices.

For a handheld to be useful for data acquisition, the challenge is creating data on the handheld and ultimately getting it into the database (using the appropriate QA in the process). If the data are simple, then simple forms may suffice. But if the data being collected are complex, then providing the forms, validation logic and appropriate data structures can be daunting on a small-footprint device. Complicating deployment is the fact that the device could be connected, or disconnected (requiring data updates or synchronization).

From an IT management perspective standardization is a key goal, and the proliferation of PDA device types is a real potential problem. If several kinds of devices are used (each with it’s own form factor), then each potentially has the need for a separate version of an application or a service. The portability problem with handheld devices is that each type is essentially a distinct “platform” with relatively little similarity guaranteed between types. The resulting support burden can be

substantial when deploying multiple device types or migrating from one device to another. If custom programs are built, they nearly always need to redesigned for another type of handheld. *These purportedly "portable"[1] devices, lack portability[2]!*

Not surprisingly, the handheld will provide only one of many access mechanisms to "enterprise data". Therefore, the need arises to build and maintain custom applications on the PDA, while also maintaining other solutions for a different client or another type of access. The challenge is to incorporate the handheld into the overall architecture for accessing and creating data. Being able to create and maintain models, logic and code which are sufficiently extensible to apply to the handheld, is the objective. So the design goal of separating applications into business logic and presentation components *should* apply to the integration of handheld computers into a data management system. The handheld is just another thin client, with some unique characteristics.

Although current demand is strong for Palm's and PocketPC's, there are also smart phones, web tablets and other device types that are increasingly available and compelling. Many of these devices are using alternatives to Palm or PocketPC operating systems. The Symbian OS is owned by an alliance of Ericsson, Motorola, Nokia, Panasonic, and Psion, and has also been adopted by Fujitsu, Sanyo, Siemen, Sony and others. It already has huge market share in Europe and Asia in mobile communications, but also has the potential of successfully competing with Palm and PocketPC on more 'capable' devices.

Linux has a strong and growing presence in the PDA market. Sharp has introduced the Zaurus SL-5000 which uses Trolltech's Qtopia interface. Tuxia has released a version of Linux for the iPaq and Compaq maintains the site www.handhelds.org where it distributes Linux for the iPaq to developers.

It is by no means certain which operating systems will dominate in the future. *Java is poised to be viable on nearly all combinations of devices and operating systems.*

JAVA ON HANDHELDS

PDA's, from the perspective of Java, are potentially addressable as a class (or classes)[3] of devices. There are a number of Java Virtual Machines available for PDA's, notable among them is Insignia's Jeode (an

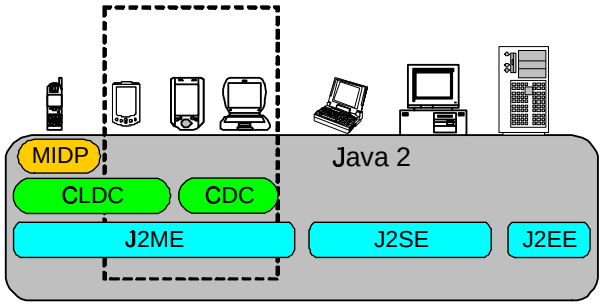

Figure 1: Java 2 provides scalability to almost all device types. The Java 2 Micro Edition J2ME is applicable to PDA's

implementation of PersonalJava and EmbeddedJava), which has been ported to PocketPC, and Linux PDA's. Integral to the Symbian OS is a full Java (J2ME) implementation, which runs on smart phone and PDA device types. For the Palm, there is MIDP[4] for PalmOS. The J2ME MID Profile addresses many of the limitations of using a microbrowser on a resource limited device.

An unorthodox approach is taken in the SavaJE XE OS, which is a "powerful operating system based on Java 2 Platform, Standard Edition (J2SE) technology, for next-generation information appliances and embedded devices - including smart phones, hand-held computers, personal digital assistants, set-top boxes, web pads, home and automotive Internet access devices, and enterprise terminals." (www.savaje.com).

Java has a significant and growing presence on PDA's. Although there are many other options (C, C++, VisualBasic, SmallTalk) for programming these devices, and advantages to using them, Java's secure and platform-independent environment makes it attractive to port Java applications to the PDA. Criticism of Java on small devices has to do mainly with performance, but there are efforts to rectify this. For example, Jeode uses "dynamic adaptive compilation", which attempts to identify frequently used execution paths and compile them from bytecode, leaving the other execution paths to be interpreted. The GCJ compiler and similar efforts are designed to compile Java "ahead of time" to machine code. Also, the PDA's themselves are becoming much faster with increased resource availability.

DATA ENTRY OBJECTIVES

COMPREHENSIVE MOBILE DATA ENTRY

The requirements for data entry on a handheld are straightforward and quite similar to any other data entry need.

[1] Capable of being borne or carried; easily transported; conveyed without difficulty; as, a portable bed, desk, engine.

[2] The ease with which a piece of software (or file format) can be "ported", i.e. made to run on a new platform and/or compile with a new compiler.

[3] CDC vs. CDLC/MIDP variants of J2ME

[4] The J2ME Mobile Information Device Profile

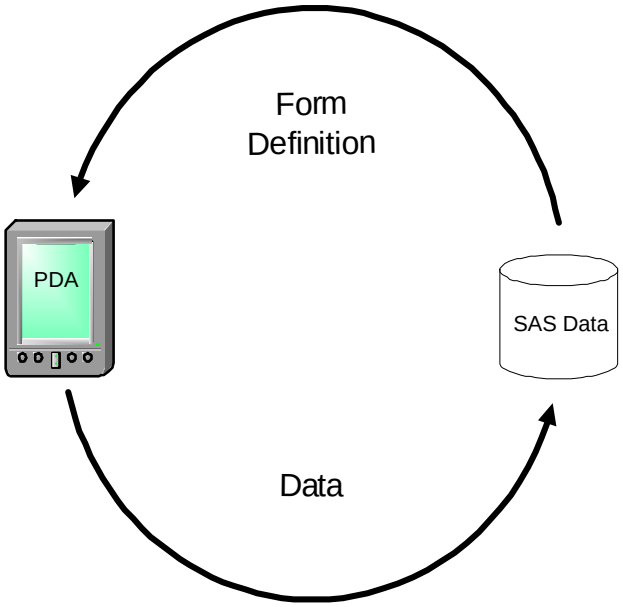

Figure 2: The data entry cycle consists of 1) acquiring a form definition for an interactive data entry session and 2) subsequent update of data to database.

- Simple entry form configuration
- Quality Assurance
  - Data Validation
    - Range Checks
    - Input Masking
    - Value auto-duplication
  - Editing or Re-Entry
- Data Synchronization

SYNCHRONIZATION

The reality for many PDA's is that they are intermittently connected and require some kind of synchronization of data between device and database. This is the well known "HotSync" feature on the Palm Pilot. Similar functionality is provided by "ActiveSync" on the PocketPC. There are numerous applications which use the sync model for data transfer to and from PDA's. The synchronization process can happen through a cradle or via some wireless connection.

An example of an application which is designed to adapt to disparate connection modes and devices is the Fieldworker Data application from Fieldworker Products. Fieldworker's Java-based software runs on a variety of devices including PocketPC, Symbian EPOC32 PDA's and Windows. The software provides a rich interactive interface which is configured via a project file. The project file determines the interface controls[5], layout and menus and populates them with data. What results is an interdependent series of screens which provide for data entry. The data are then stored in a small database on the device and are either exported or synchronized to a server using the Fieldworker Sync product. The Fieldworker software successfully overcomes the need to create customized interfaces for each data collection effort.

DATA MANAGEMENT SERVICES

Increasingly, SAS provides Web Technologies to extend it's data management capabilities through information delivery via the web. With these SAS tools and abilities, presentation of information has graduated from delivery of static HTML to dynamic content and interactivity. SAS Integration Technologies provides tools for administration, Integrated Object Model (IOM), messaging and directory (LDAP) services. SAS understands the need for application integration and is increasingly providing the tools (Java and otherwise) to achieve interoperability.

Product	Technology niche
IntrNet	CGI Java HTML
ODS MARKUP	XML HTML PDF RTF WML etc.
LIBNAME XML	XML import/export
Integration	Middleware
WebEIS	OLAP Java
WebAF	Java Servlet JSP WAP/WML

Table 1: SAS Web-Enablement Technologies.

JAVA - PORTABLE CODE

Java is an ideal language to create distributed enterprise applications. Because of it's object-oriented design and platform independence, it can be used for scalable, distributed systems, and is an obvious platform for Web services, which may be accessed by many different clients. Additionally, the motivation to use Java is enhanced by the fact that a coalition of organizations support the Java 2 Enterprise Edition platform, which includes EJB's, Servlets, JSP's, Java Message Services, etc. This coalition has formally evolved into the Java Community Process which oversees the development of Java technology.

The success of Java is evident on servers as well as on clients and information appliances like PDA's. JVM's are supported on many operating systems including Windows, Linux, MacOS X, Solaris, SymbianOS, PalmOS, and PocketPC. This wide-spread availability

[5] Data collection field types: alpha, number only, picklist, multipick, numeric slider, checkbox, note, sketch, date, time, time stamp, formula and counter.

makes it possible to address multiple platforms with the same or similar code bases, and eases the effort involved in porting. The concept of modularity can then be extended to the choice of platform(s), creating greater choice and potentially simplifying development and migration.

XML - PORTABLE DATA

The Extensible Markup Language may owe its pedigree to SGML, but it owes its success to the web. In its key role of data interchange, XML-based information has come to be referred to as 'Universal Data'. XML provides interoperability because it is structured, extensible and open. The structure is embodied in the concepts of well-formedness and validation. Extensibility is inherent in the design of XML, giving the authors of XML documents and schema control over the choice and implementation of the structure. XML is intrinsically open, in that XML documents are readable text, often with associated DTD's (Document Type Definition) or Schema, providing all the metadata needed to use, or transform, the data.

XML - BENEFITS

The strengths of XML have been detailed extensively, but a typical list of advantageous features would include:

- Plain Text
- Data Identification
- Stylability
- Inline Reusability
- Linkability
- Easily Processed
- Hierarchical

Notably, the hierarchical nature of XML contstructs is diametrically opposed to the 'rectangular' nature of typical SAS data. This hierarchical vs. rectangular method of representing data heightens the need for efficient and automatable means of 'transformation'.

XML TRANSFORMATION

Transformation of XML data concerns principally three XML specifications: XSL, XSLT and XPath. The Extensible Stylesheet Language (XSL) is both a language for transforming XML documents and a vocabulary for formatting XML documents (McLaughlin 2001). XSL Transformations (XSLT) specifies the textual conversion from one document type to another. XPath(Clark & DeRose 1999) provides a mechanism to identify and refer to an arbitrary element or attribute names and values.

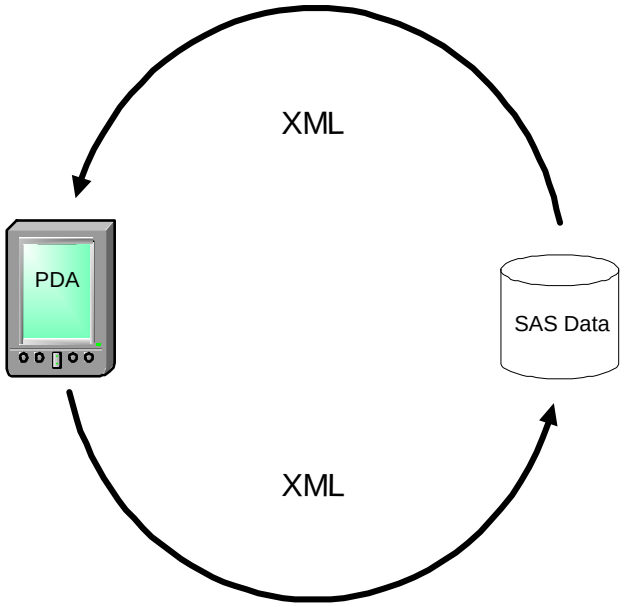

Figure 3: Using XML for form definition and synchronization would standardize the data and help to achieve independence of the PDA device's software implementation and independence from the server implementation.

XPath and XSLT work together to define what data to use, and how to transform it.

Of course, an XML parser[6] is required to read XML data into memory. XML can be read from a file, although Java XML parsers can read from an `InputStream` or a URL. A validating parser uses the documents' DTD or Schema to validate the data during the parsing process, which simplifies the Java validation code required.

XML AND JAVA

Java has deep and multi-faceted XML support. In Java, the Simple API for XML (SAX) is the most commonly used parsing method. Current parsers, (Apache Xerces for example) now implement SAX and DOM (Document Object Model) interfaces and are adding support for Namespaces, Schema and JAXP (see below).

Java, XML, and XSLT are suitable for web applications because of the high degree of modularity they offer (Burke 2001). The Java API for XML Processing (JAXP) supports processing of XML documents using DOM, SAX and XSLT. It is essentially an abstraction which allows the substitution of different parser in an application without changing the application code. This achieves vendor -independence of parsers: "It encapsulates differences

[6] Parsers are available in many languages including C, C++, Perl, Java, etc., both commercial and open-source.

between various XML parsers, allowing Java programmers to use a consistent API regardless of which parser they use"(Burke 2001).

XML AND SAS

Clearly, SAS is acquiring the ability to process XML and to expose SAS data as XML documents. The ODS MARKUP statement (experimental as of 8.2; production support in v9.0) can be given various TAGSET= options to define several markup languages including several XML types. Notably, the XML tagsets generate corresponding XML data, XSL stylesheets and DTD (Document Type Definition) schema. The tagset definitions define ODS MARKUP and LIBNAME XML destinations.

TAGSET	XML Type
DEFAULT	ODSXML (Generic XML)
DOCBOOK	OASIS DocBook
EVENT_MAP	SAS Event Triggers
PYX	PYX[a]
SASIOXML	Generic XML
SASXMOG	Oracle8iXML[b]
SASXMOIM	Open Information Model XML
SASXMOR	Oracle8iXML[c]

[a] Python Pyxie Library
[b] XML LIBNAME XMLTYPE=GENERIC
[c] XML LIBNAME XMLTYPE=ORACLE

Table 2: ODS MARKUP in SAS v9.0 has TAGSET options to support several XML 'dialects'.

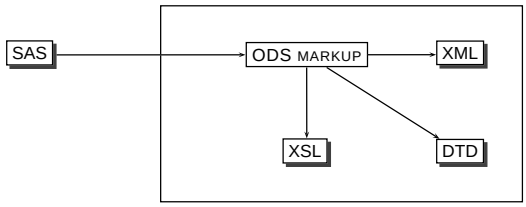

Figure 4: ODS MARKUP can be used to create XML data, stylesheets and schema.

Conversely, there is also the ability to import XML using the XML LIBNAME engine, providing that the XML is "very regular". Also available (as an enhancement to 8.2; production v9.0), is the ability to import XML into a SAS dataset using the XMLMap[7] option with an explicitly defined XML file containing syntax to 'map' variables, observations and variable attributes.

In cases where the XML is not importable, transformation using XSL/XSLT could be used to re-shape the XML to a format SAS can accommodate. However, this highlights a perplexing issue for XML processing and SAS, because the XML capabilities in SAS (ODS MARKUP, LIBNAME XML) overlap with the functionality of XSLT, and Java technologies such as JAXB[8] and JDOM[9]. In this regard, the SAS (Java-based) Atlas[10] product may help in the creation of XMLMAP files, however the need to automate (XML ⇔ SAS) transformation is a requirement beyond an interactive interface.

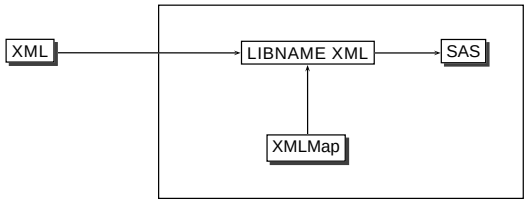

Figure 5: LIBNAME XML uses XMLMap directives (XPath) to map data from XML input to a rectangular SAS data set.

Web components specified by the J2EE platform are servlets and JavaServer (JSP) Pages. Servlets extend the functionality of a Web server by dynamically generating a response to a request from a client. JSP mixes template data (XML, HTML etc.) with content generated dynamically by servlets. Custom tags, which are JSP's most powerful feature (Geary 2001), are themselves XML- compliant and attempt to achieve the separation of presentation and content by encapsulating functionality. WebAF provides a custom tag library specifically for SAS data access via TransformationBeans and linkages to SAS analytical tools. The WebAF component of AppDevStudio 2.0 uses iPage TransformationBeans to generate JavaServer Pages which respond selectively depending on the requesting wireless or other device. This polymorphic behavior is achieved through (micro)browser detection and returns multi-modal markup in WML, HDML (Handheld Device Markup, Language) or HTML.

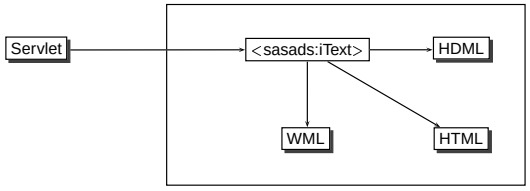

Figure 6: WebAF's iPage TransformationBeans conditionally generate markup from a Servlet via custom JSP tags.

Since XHTML is recommended by the W3C for future web development and XHTML Basic is projected to replace WML, enhancements to this capability would be expected to support XHTML[11]. XHTML [Basic] documents are XML and thus are well-formed and can be

[7] Using XPath methodology to define and retrieve XML elements.

[8] Java API for XML Binding.

[9] A java-centric XML processing approach (JSR-102).

[10] An interactive XMLMAP creation utility projected for release with v9.x.

[11] As of v9.0, CHTML, IMODE and WML are supported.

validated. Also notable is that XHTML Basic uses a wireless version of CSS providing separation of presentation and content as Model-View-Controller (MVC) would encourage.

WEB SERVICES

An on-going problem for data management and information delivery is application integration or interoperation. One of the difficulties of integration is enforcing a data standard, but exposing information as XML is becoming a universal requirement. With the acceptance of XML's key role in data interchange, it is now possible to think of data management as a service, or a "Web Service", as the idea has come to be known.

Although Web-services have been criticized as yet another distributed computing model (such as CORBA, RMI or DCOM), there appears to be momentum and a genuine potential for a higher level application integration. Much of this optimism concerns the fact that the Web Services model is based on Internet standards: HTTP and XML. The idea of remote object access over HTTP via XML is really profoundly transforming, because it simplifies everything. The ubiquitous presence of HTTP makes it ideal for data transfer between client and services, while XML, as an abstract (meta) language, specifies the data and interactions between the parties.

Additionally, the web-services "stack" - SOAP, WSDL, and UDDI provide standardized layers of functionality to ensure interoperability. The Simple Object Access Protocol defines a uniform way to pass XML and perform remote procedure calls. Universal Description, Discovery and Integration provides a registry of SOAP-encoded messages. The UDDI registry uses the Web Services Description Language to describe what a client needs to know to bind to the service. A broad array of support is growing for these ideas and technologies.

The holy grail which web-services is supposed to achieve (and about which there is so much hype), envisions a "semantic-web" wherein self-describing information is produced, advertised, delivered and consumed. This is probably still a long way off, for a variety of technical and not-so-technical reasons. But the vision is a compelling one, especially for data managers and other IT professionals who spend seemingly inordinate effort massaging data to get it "in the right format".

Although a comprehensive web service model may not be achievable for many organizations or projects right now, there is merit in understanding the concepts and incorporating relevant ideas into future project plans. The proposed architecture of the web services style of distributed computing promises greatly enhanced flexibility. Along with this flexibility comes the increased capability to partition and modularize applications into components.

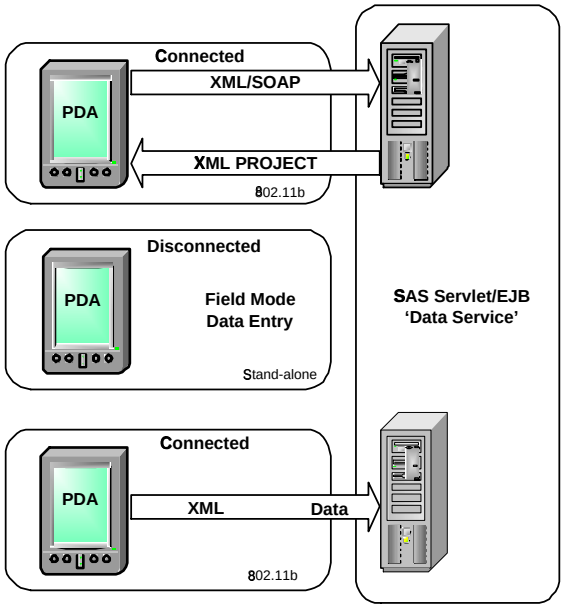

Figure 7: Connecting to a Data Service for Data Entry form definition; Data Entry in stand-alone mode; subsequent data synchronization event after data collection.

COMPONENT-BASED APPLICATIONS

Derived from the web-services model, a SAS-based 'data service' simplifies the deployment of handheld computers in a 'field' situation by standardizing the access method and data delivery. Although not a complete web-service, a prototype model uses SOAP to exchange information between the handheld and the 'data service'. In connected-mode, a message defined on the handheld is delivered to the server (via a SOAP request), and the server responds with the requested form definition in XML. The handheld application then parses the XML encoded form definition to build the form and its behaviors. In disconnected mode data are entered into the form, after which it is reconnected. Finally, the XML data are transferred (SOAP) to the 'data service' which transforms the incoming data for processing or inclusion in the data set(s). A more interactive model could be set up for connected data entry.

Other aspects of a complete web-service will be added to the model to achieve:

- Description: To name and define forms and their data models.
- Discovery: To acquire form definitions from unknown sources.

CONCLUSION

The availability of Java on both server and client platforms (including handheld devices) makes it a natural choice for the next generation of applications. The J2ME, J2SE and J2EE platforms provide a comprehensive spectrum of software to construct and deploy web services. SAS software is quickly acquiring and refining web technologies including Java integration, XML processing and wireless delivery. The analytical, data management and multi-platform capabilities of SAS are very well suited to problems which require a scalable solution. XML's key role is providing open solutions for data interchange and metadata. The web-services architecture is a model which is appealing for providing standardized 'data services' that function independently of a specific client 'type'. Integrating handheld devices into data services is a goal that SAS, Java and XML can effectively achieve.

WEB REFERENCES

SAS XML

http://www.sas.com/rnd/base/index-xml-resources.html

OTHER

http://xml.apache.org
http://www.fieldworker.com
http://www.handhelds.org
http://www.insignia.com
http://www.jboss.org
http://www.jcp.org
http://www.jdom.org
http://www.savaje.com

REFERENCES

Burke, E. M. (2001). Java and XSLT, first edn, O'Reilly.

Clark, J. & DeRose, S. (1999). Xml path language (xpath) version 1.0, "http://www.w3.org/TR/xpath".

Designing Wireless Enterprise Applications Using Java Technology (2001). Technical report, Sun Microsystems, Inc.

Geary, D. M. (2001). "Advanced JavaServer Pages™", Java™ 2 Platform, Enterprise Edition Series, Prentice Hall.

McLaughlin, B. (2001). Java and XML, second edn, O'Reilly.

XHTML™ 1.0: The Extensible HyperText Markup Language (2000). "http://www.w3c.org/TR/xhtml1".

ACKNOWLEDGEMENTS

I would like to thank Heidi Markovitz and Jenine Eason for encouraging me to participate in the Application Development session. I would also like to thank Ichauway, Inc. for providing the opportunity to pursue the projects which allowed me to explore these ideas.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Scott Chapal
Ichauway, Inc.
Rt. 2 Box 2324
Newton GA. 31770
scott.chapal@jonesctr.org