Paper 177-27

Using Java™, SAS® and XML to Integrate Handheld Data Acquisition with Data-management Web Services

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

ABSTRACT

Java™, SAS® and XML are justly viewed as complimentary, synergetic technologies. Integrating data from hand-held devices into a data management system can be accomplished by effectively combining these tools. The platform-independence of Java provides portability to small devices as well as to server processes and traditional client interfaces. SAS data structures and programming constructs can be used effectively to cooperate with Java objects and XML. The conceptual purpose of a web service is to simplify the problem of interoperation by using existing protocols and standardized data. Data management, redefined as a web service, achieves access via XML messaging and meta-data. The service it provides is "interoperability", whether with another application, another server, a browser, or even a handheld device.

INTRODUCTION

Integrating data from handheld computers into a structured data management system can be a difficult and vexing problem. Ideally, hand-held devices should provide a seamless extension to information systems, but without careful design and planning, the integration can be far from simple. The hardware, operating systems and software used in these devices is evolving rapidly, presenting hard decisions for IT managers. The pace of this change can make deployment criteria short-lived, rendering yesterdays choices obsolete by today's technologies. Current efforts to integrate data management systems with handheld devices are constrained by several factors that can be stated in PC[1] terms: device-challenged, data-challenged, portability-challenged and integration challenged.

DEVICE CHALLENGE

Until recently, there were limited options for deploying small hand-held computers for serious data acquisition purposes. Constrained by hardware limitations, subnotebook sized computers were under-powered, clumsy, fragile and idiosyncratic. Notably, an early exception was the Apple Newton.

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 hand-held 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 are now incorporating integrated wireless connectivity as core functionality rather than 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.

For the moment however, 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.

DATA CHALLENGE

As intriguing as these devices are, it is the information they process which is of real interest to the user. Even though PDA's are marketed as personal (multi-media) organizers, there is little insight required to imagine exploiting them for serious work. The asymmetrical requirements of data presentation and 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 collection is complex, then providing the forms, validation logic and appropriate data structures can be daunting on a small footprint device. Complicating the situation is the reality that the device might be connected, or disconnected (requiring data synchronization).

[1] Politically correct, not Personal Computer.

PORTABILITY CHALLENGE

Although hand-held systems have been used for data acquisition in the past, many of them have suffered from intractable problems. There are many issues which can complicate the successful use of handhelds in the field, including hardware design, software inadequacies and lack of user "acceptance". A brief list of complaints about handheld devices would include things such as:

- Hardware peculiarities
- Non-standard or proprietary data formats
- Fragile design
- Unique Ergonomics
- Resource limited programming requirements
- Dubious/unknown future viability in the marketplace

From an IT management perspective, it is always a good idea to standardize; device proliferation is a real problem. If multiple 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 deployment burden can be substantial when supporting 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 device. These purportedly "portable"[2] systems lack portability[3]!

INTEGRATION CHALLENGE

Not surprisingly, the handheld will provide only one of many views 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, it seems, is to incorporate the handheld into a larger architecture for accessing and creating data. Being able to create and maintain models, logic and code which are sufficiently extensible to also 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 data management. The handheld is then just another *thin* client, with some unique characteristics.

PDA PROLIFERATION

Even among PDA's, the choices are perplexing and it is not a simple matter to choose among the options. After all, the term PDA is actually a convenient term which aggregates devices which are of somewhat dissimilar designs. The balkanization of the PDA market appears to have settled along a well defined fault-line between PalmOS and PocketPC.

Model	OS
Compaq iPaq 3650	PocketPC
Compaq iPaq H3670	
HP Jornada 548	
Compaq iPaq H3850	
Compaq iPaq H635	
Palm Vx	PalmOS
Palm m505	
Sony CLIE PEG-N760C	
Handspring Visor Deluxe	
Sony CLIE PEG-S320	

Table 1: Common PDA's sold January 2002.

But a survey of PDA sales isn't the complete picture. 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)[4] of devices. There are a number of Java Virtual Machines available for PDA's, notable among them is Insignia's Jeode (an 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[5] for PalmOS. The J2ME MID Profile addresses many of the limitations of using a microbrowser on a resource limited device.

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

[3] 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.

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

[5] The J2ME Mobile Information Device Profile

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 apps 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.

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 goes by the name of 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[6], layout and menusa 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.

A feature in JDK 1.4 is long term bean persistence, including a downlowdable GUI builder that reads and writes XML documents representing GUI’s. It is this kind of future functionality which holds so much promise. Imagine an automated data-driven server process which generates an up-to-the-minute XML representation of a dataset. A sync operation, initiated from a PDA, then queries the server which responds with a properly formatted, QA-enabled data-entry project specification for that particular device and data collection requirement.

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

By reviewing current technology, such as Fieldworker Data, and developments in Java technology, such as the aforementioned XML GUI builder, it is clear that the software needed to support seamless integration of mobile data collection is rapidly becoming available. Handheld devices will soon be dynamically configurable, automatically synchronized and simple extensions of the information infrastructure.

DATA MANAGEMENT REDEFINED

Data management is fundamentally important to enterprise information. Any experienced SAS professional appreciates SAS’s large-volume data management capabilities. SAS provides not only the core data management functionality found in BASE SAS, but also offers Data Warehousing capabilities, metadata management etc. While not every application needs to be web-enabled, the requirement to connect many SAS applications to thin clients of all types, whether to an automated process, a browser session or a handheld computer, is obvious.

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

Table 2: SAS 8.2 Web Technologies.

Increasingly, SAS provides Web Technologies to extend it’s data management capabilities through information delivery to 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.

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 would seem a natural choice for Web services, which may be accessed by many different client platforms. 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 acceptance 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 (95/98/Me/NT/2000/XP) Linux (Redhat, Debian...), MacOS X, Solaris, SymbianOS, PalmOS, and PocketPC. This wide-spread availability 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 owes its pedigree to its SGML roots but it owes its success to the web. XML's key role is 'data interchange', and it has been referred to as 'Universal Data'. XML provides interoperability because it is structured, extensible and open. 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 transformation 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 provides a mechanism to identify and refer to an arbitrary element or attribute names and values. XPath and XLST work together to define what data to use, and how to transform it.

Of course, an XML parser[7] 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. 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 between various XML parsers, allowing Java programmers to use a consistent API regardless of which parser they use"(Burke 2001).

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

WEB SERVICES

An on-going problem for data managment and information delivery is application integration or interoperation. One of the difficulties of integration is enforcing a data standard, but exposing information as XML now appears to be a basic 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 criticised 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.

XML AND SAS

Clearly, SAS is acquiring the ability to process XML and to expose SAS data as XML documents. As of version 8.2, SAS provides several ways to create XML from SAS datasets. The 'experimental' ODS MARKUP statement can be used to generate many markup languages, but importantly to this discussion are XML, WML, XSL and DTD. Also currently available is the ability to define and use tagset definitions with ODS output and the XML LIBNAME engine.

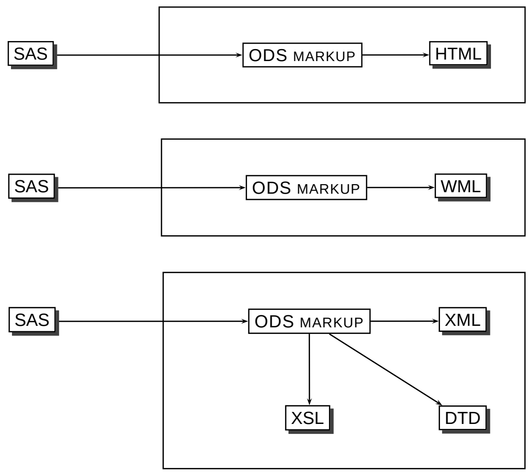

Figure 1: ODS MARKUP in SAS 8.2 can be used to create XML and several other output types.

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 is the ability to import XML into a SAS dataset using the XMLMap 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 may be used to re-shape the XML to a format SAS can accommodate.

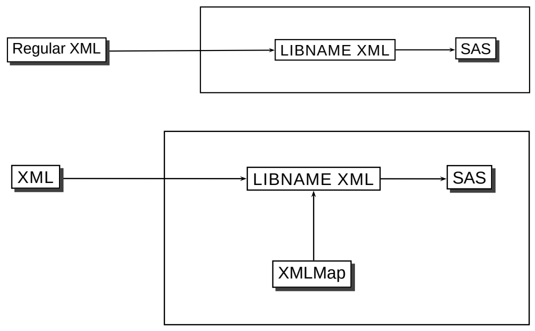

Figure 2: LIBNAME XML functionality in 8.2 uses XMLMap syntax to achieve data binding between the XML input and a SAS data set

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.

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.

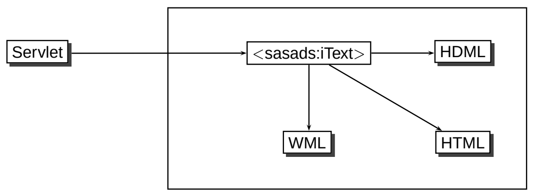

Figure 3: WebAF's iPage TransformationBeans conditionally generate markup from a Servlet via custom JSP tags.

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 . XHTML [Basic] documents are XML conforming and thus are well-formed and can be 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.

OPENNESS

The importance of standardization for data exchange is obvious. The real advantage to using XML is that data are "portable" and thus standards in XML specifications play a vital role. On the other hand, standards, if implemented too early or too broadly, can prevent the adoption of better technologies. The standards on which XML technologies are based, are becoming more important as software is able to understand and process XML universally.

There is a substantial amount of development occurring in open source projects. Much of this open source software is finding its way into developer environments, test suites and even commercial software. Examples are the Apache web server and Tomcat (the reference implementation for Java Servlet and JSP technologies), which have both become defacto standards. SAS includes Apache 1.3 and Tomcat 3.2.1 in AppDevStudio 2.0 as the default Java Application server.

In the Java/XML arena there are several open source projects under the umbrella of `xml.apache.org`. The Xalan processor is an open source implementation of

XSLT and Xpath. Xerces-J is a Java XML Parser, version 2 of which will support SAX 2.0, DOM Level 2, JAXP, Schema and Namespaces. Xalan and Xerces are also included in AppDevStudio.

There are also several open source projects which attempt to provide programming frameworks. Examples of these are MVC frameworks such as Struts (J2EE-compliant) and Turbine which are subprojects of the Apache Jakarta project. Another is the Castor data-binding framework which provides Java to XML binding, Java to SQL/LDAP persistence, etc. There are even open source J2EE-based application servers, for example JBoss (J2EE-compliant) or Enhydra . The standards and specification processes around Java and XML have certainly created a dynamic environment for software development to occur. The interaction of commercial and open source efforts is mutually productive and on-going.

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. Integrating handheld devices into a data management web-service is a goal that Java, SAS and XML can very effectively achieve.

WEB REFERENCES

SAS XML

http://www.sas.com/rnd/base/topics/odsmarkup/

http://www.sas.com/rnd/base/topics/odstagsets/

http://www.sas.com/rnd/base/topics/sxle82/prod82/

http://www.sas.com/rnd/base/topics/sxle82/exp82/

OTHER

http://www.enhydra.org

http://www.fieldworker.com

http://www.handhelds.org

http://www.insignia.com

http://www.jboss.org

http://www.jcp.org

http://www.savaje.com

http://xml.apache.org

REFERENCES

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

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".

XHTML™ Basic (2000). "http://www.w3.org/TR/xhtml-basic/.

ACKNOWLEDGEMENTS

I would like to thank Greg Barnes Nelson for inviting me to participate in the Emerging Technologies 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