

## Intelligent Finance – XML Web Services In Action.

Halifax Bank in England recently launched an online bank called Intelligent Finance ([www.if.com](http://www.if.com)). An XML application, Intelligent Finance represents a great example of how to use the conversion of legacy systems into online applications as an opportunity to add functionality while extending reach. XML Web Services have been used to integrate client browsers and mobile phones with legacy mainframe banking systems to create a new type of online bank, offering financial services never seen before.

All too often developers charged with the task of web-enabling systems simply transfer the pre-existing functionality across to the internet, giving no thought to possible opportunities to enhance the system. The conversion of an existing system should provide developers with the opportunity to extend systems in new and innovative ways. Intelligent Finance represents a classic example of how to use XML to provide more intelligence and better functionality when web-enabling systems.

Basically, Intelligent Finance works by amalgamating all of your accounts for the purposes of interest calculations. Under the normal banking system each account is treated as a separate item, with interest calculated at a rate specific to that account type. What Intelligence Finance does is link all of your accounts and provide you with opportunities to connect debts to savings in various ways. For example, if you have borrowed £50,000, and have £10,000 in your saving account, you could amalgamate the two accounts for interest calculations, paying interest on only £40,000. The system is quite straightforward if you think about it, and you have to wonder why banks haven't done it before.

Linking customer accounts to the internet in order to create an online facility for an existing bank requires the creation of a layer of middleware between the legacy mainframe banking systems and the web server. What Halifax Bank developers saw was that the need to create this layer represented an opportunity to add functionality to the overall system.

Dirk Marwinski, of Shinka Technologies, worked on the technical architecture;

"The middle tier exposes the existing functionality through XML. Interfaces are exposed in XML as Web Services, while data is formatted through XML Schemas," said Dirk.

Thus, the Intelligent Finance banking system lies primarily at the XML Web Services layer. Web Services are mechanisms which permit developers to create distributed computing environments. Similar functionality has been offered by DCOM, RPC, and EJB, but many people believe XML Web Services are better. Web Services enable an application to find and access another application, using XML as a common interface so as to avoid having to know anything about the internal requirements of the application being called. In a sense, XML Web Services enable developers to treat entire applications like platform-independent objects. In theory, a web service written in C++, running on IIS, could access another written in Perl, running on Linux/Apache.

In order to achieve this, Web Services must provide standardized mechanisms to search for and find a service, and then to request and access that service. In addition, Web Services must provide a standard mechanism to define the input and output parameters of a service.

Technical Architect Dirk Marwinski came to XML from CORBA.

"The advantage of XML Web Services as opposed to other middleware languages is that you can decouple the clients from the servers best under XML. You also get better testing and error messages than, for example, CORBA," he said.

The system was written using EJB, though tools were first created to shield the developers from directly working with XML. Developers found they had to use a wide range of Java developer's tools, right down to Vi. DOM was used during compilation to convert XML objects into their corresponding java objects, while SAX was used for parsing during runtime. DOM is generally heavier on resources than SAX because it represents complete XML structures in memory, and is not as fast. However, Stefan said that resource usage was not as important

as error messages during compile time. On the other hand, SAX's speed and resource efficiency made it the best candidate for use during runtime.

XML Schemas were used in preference to DTD's. According to Shinka Technology's Stefan Havenstein, who was responsible for coordinating the different XML channels, schemas were found to be more powerful, especially when it came to handling multiple elements of the same type, and for creating compound elements. It was also felt that Schemas provided a more future-proof system than DTD's. Future-proofing was also a key factor in determining the use of SOAP for data exchange.

SOAP is an XML protocol which is used in distributed environments for information exchange, and for invoking methods in applications. SOAP is a light-weight protocol that uses HTTP, which avoids a number of issues, such as dealing directly with firewalls. SOAP is still in development at W3C, so Intelligent Finance uses what the developers on the project refer to as "early SOAP dialects."

One of the most surprising lessons development managers found when creating the Intelligent Finance system was that XML significantly improves debugging. The fact that all XML documents must be human-readable made it much easier to expose errors, since messages between two applications were not meaningless binary, but could be read (and, if necessary, edited) by the developers. However, it was also discovered that XML Web Services also retain some of the problems common to any distributed computing environment.

"You get time and performance issues around sending messages and then waiting for a reply," said Stefan Havenstein. "So you want to avoid making your messages too granular."

Stefan was one of those responsible for coordinating the various developer teams. His conclusion from watching them use XML Web Services is that the tricks from other distributed computing environments still apply, and this can present a steep learning curve for developers who aren't familiar with coding for distributed environments.

"Don't call ten methods ten times when you can call them all at once in a single message," he said, "And you need to understand that you are working in a document-oriented model, not an object-oriented one. However, in my experience, XML Web Services can do everything CORBA can do."

BRANDT DAINOW

bd@internet-etc.com

Brandt Dainow is Senior XML tutor and lecturer with Internet Etc ([www.internet-etc.com](http://www.internet-etc.com)), the internet training company.

**LINKS:**

**XML Web Services:** <http://www.w3.org/TR/wsdl>

**SOAP Part 1:** <http://www.w3.org/TR/soap12-part1/>

**SOAP Part 2:** <http://www.w3.org/TR/soap12-part2/>

**Intelligent Finance:** <http://www.if.com>

**Shinka Technologies:** [www.shinkatech.com/](http://www.shinkatech.com/)