

Giving Java something to do?

XML brings processing power to markup

Developing creative, intelligent custom solutions for our clients has never been easier, yet paradoxically, more challenging. Java has been evolutionary in providing direction for developers and systems integrators looking for better solutions and resolving some long-standing issues. Java makes it easier and faster to develop distributed cross-platform applications. However, Java presents inherent challenges stemming from its immaturity as a language and the very nature of standards based, cross-platform, distributed applications (ensuring interoperability). Java applications, and our clients, stand to benefit a great deal from a new, enabling mark-up language—XML.

First I want to ask rhetorically: what does Java do? OK, (deep breath) Java does distributed cross-platform easily programmed-deployed-supported portable object-oriented dB enabled quick-to-market reusable computing...right? Well, yes and no. Alas, most of the 'things' we think of Java doing (distributed cross-platform easily...) are merely characteristics of the language itself. Not to minimize the benefits of Java, many exist and they are real, but to state the obvious, Java builds programs. Ultimately, what Java really does is process data, more precisely, it processes data from files and dBs.

Yes, Java enables IS departments and Systems Integrators to come to market sooner with custom applications, and yes, Java enables wider client distribution than ever thought possible for powerful applications. But are we really changing computing itself by using Java? Java is, as I mentioned before, an evolution in computing—not a revolution.

Hello eXtensible Markup Language (XML).

XML is also evolutionary—it brings data definition to standardized, compact, distributed, stylized document markup languages. So what? Well, besides the large evolutionary step for document publishing, XML's object characteristics combined with Java's OO strengths make for a true 'revolution' in computing. In short: Java + XML = Processing Power.

XML is a subset of SGML, a widespread, but highly complex markup language. Language designers took some of the best characteristics of SGML and left out a lot of the nice-to-have-but-not-required kind of features and *viola*—XML (sounds a lot like Java's 'creation story,' eh?). The entire Internet community is excited about XML's February 10th debut as an official W3C¹ recommendation because, "XML will be the glue that will integrate electronic data interchange, databases, and even operating systems, making the computer itself an extensible linked document and database."² Sounds impressive. In short, XML makes documents multidimensional—data becomes objects—which means programs can do a lot more with 'simple' data.

XML is actually a family of core language proposals: eXtensible Stylesheet Language, Document Object Model, Resource Description Framework, eXtensible Linking Language, Channel Definition Format, and Meta-Content Format. Some of these core proposals have not even been recommended and are actually competing for industry approval. All in all, XML is very, very young—but it holds a lot of promise.

Gartner Group predicts that rapid deployment of XML standards will spur corporate publishing to out-grow traditional dB publishing by at least 100-to-1³ in the next 4 years. Like it or not, the way we do things today will change. Industry support for this emerging language is even greater than was for Java in its initial state; the need and potential for a distributed, easy-to-learn, object aware markup language is apparent.

Imagine a world where distributed Java applications source their data from non-proprietary XML files (on the Internet), process it, output XML and end users open those same files in Excel, Word, or whatever—it's not so far-fetched! Taken together, the possibilities of Java and XML are truly revolutionary.

¹ World Wide Web Consortium

² John Gage, Sun Microsystems

³ Research IDOM: M-03-6115