Title: XML Backbones - The Future of N-Tier

Author: Geoff Crawford

XML has already made in roads into corporations in the form of data interchange and EDI replacement. Standards organizations are repositories for the results of group initiatives for industry specific formats. But the most exciting XML features are still emerging in the form of the delivery methods and standardized data sources.

By combining XML formats with the same HTTP and SLL protocols found in ordinary web browsers and web servers, we produce application data components in the form of Web Services. A Web Server is used to accept requests for data, and pages in XML instead of HTML are returned. As a standardized, self-describing format, the data becomes a kind of backbone that any application capable of XML distributed interchange available across the network.

Before we explore the architecture of such backbones, a quick note about the importance of the self-describing aspect of XML. True N-tier applications have each component not only encapsulated from the others, but also available from anywhere on the network. Since data sources to feed those applications can be used in multiple applications written with different products, universality is key. XML allows for identifying each piece of data by name as well as grouping data into a set. These two factors give applications the flexibility to not only mark data they are interested in, but also the data they are not interested in. That way a single XML data source can give information for multiple uses yet not have to have multiple formats.

Here is an example of both named data elements and grouping into a set. The sample is based on two Orders each its Order Items. Notice how each Order Item is surrounded by the Orders tag in order show that relationship. Likewise, each Part Number, Description, Price, and Quantity are encased within an OrderItem tag.

```
<OrderData>
  <Order>
    <OrderNumber>C0200017</OrderNumber>
    <CustomerReference>20020319-0003</CustomerReference>
    <OrderDate>03/21/02</OrderDate>
    <OrderItem>
      <ItemCount>1</ItemCount>
      <PartNumber>1-100-132-06</PartNumber>
      <Description>BLADE</Description>
      <Price>46.25</Price>
      <Qty>100</Qty>
    </OrderItem>
    <OrderItem>
      <ItemCount>2</ItemCount>
```



```
      <PartNumber>1-100-135-01</PartNumber>
      <Description>BOLT</Description>
      <Price>18.00</Price>
      <Qty>50</Qty>
    </OrderItem>
  </Order>
  <Order>
    <OrderNumber>C0200018</OrderNumber>
    <CustomerReference>1456-3292</CustomerReference>
    <OrderDate>03/21/02</OrderDate>
    <OrderItem>
      <ItemCount>1</ItemCount>
      <PartNumber>1-100-135-01</PartNumber>
      <Description>BOLT</Description>
      <Price>17.50</Price>
      <Qty>25</Qty>
    </OrderItem>
  </Order>
</OrderData>
```

The format automatically lets the application know that the same item that is Part Number 1-100-135-02 is also the one that is Quantity 30 in Order Number C0200018. CSV and Flat File formats cannot do that. Neither of those alternatives can allow for each data element to be in any order without causing the application a problem. XML does that gracefully since you refer to elements by name, not location.

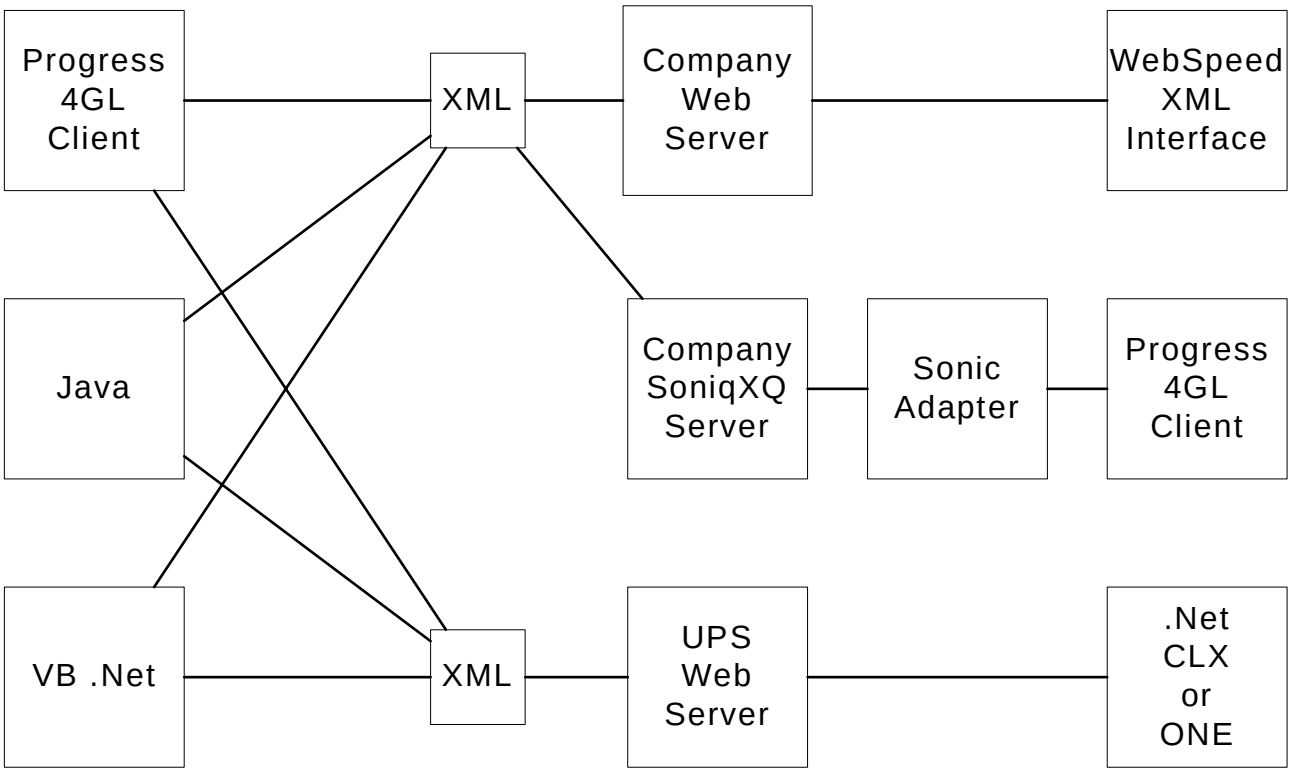
While one application might look at only the data items named Order Number and Customer Reference Number, others might look only at items grouped underneath OrderItem. Since our data might be used in places that are Order Header related as well as Order Item related, we reduce our programming effort by allowing applications to only take information they are interested in.

Many different systems today can already make use of Web Services to provide data sources, everything from Java, to ASP pages with the Microsoft XML parser, and Microsoft Products with the upcoming .NET framework, to Borland CLX applications in c++ or Delphi, to Sun's ONE. By creating one single backbone of XML delivery via Web Services, a multitude of today's applications can seamlessly operate together. CORBA and DCOM have promised that interoperability for distributed applications in the past, Web Services are beginning to actually delivery it.

So how does today's Progress product line fit into the XML backbone world? Today's 4GL client has an XML parser built-in that makes deciphering the XML reasonably easy. TCP/IP sockets are also available in order to generate HTTP requests, however the process is not one step. (see the http://www.freeframework.org for samples of both sockets and generating HTTP requests) SSL access as well as true Web Services client

facilities (SOAP/SAX) are planned for future releases of Progress. On the server side of the back bone, Progress has XML integration with WebSpeed as well as guaranteed delivery with high availability and scalability through the recently announced SonicXQ product line. SonicXQ is the Web Services front end that sits on top of the award winning SonicMQ messaging system.

The possibilities of both front and back ends are endless with the media in between now standardized. Here is a diagram that shows many of the potential scenarios, all of which are available today. Web Services can be used now with UPS shipment data, FAX system integration with VSI-Fax, and many other systems.



For more information on program samples, visit the Free Framework web site as previously mentioned, as well as the XML tool kit from Progressions Number . As an illustration of how other technologies look, here is a Microsoft ASP web page that shows how the MS XML Parser accesses web services.

```
<%@language=JScript%>
<%
  var SrvHTTP;
  var XMLDocument;
  var response;

  // Create instances of the HTTP requestor, the parser and a document
```



```
  SrvHTTP = Server.CreateObject ("MSXML2.ServerXMLHTTP.4.0");
  XMLDoc = Server.CreateObject ("MSXML2.DOMDocument.4.0");
  var XMLParser = new ActiveXObject("MSXML2.DOMDocument.4.0");

  // Build a static HTML document

  XMLDoc.async= false;
  XMLDoc.loadXML ("<Req NAME='Orders'><DATE>3/21/02</DATE></Req>");

  // Send the document to a URL

  SrvHTTP.open ("POST","http://localhost/scripts/wsbroker1.wsc/processxml.p",false);
  SrvHTTP.send (XMLDocument);

  // Write the returned page to the browser

  Response.ContentType = "text/xml";
  Response.Write(SrvHTTP.Response);

%>
```

Phone: (973) 361-4224 Fax (973) 537-6946
E-Mail: info@innov8cs.com Web: www.innov8cs.com

