XML for data

Hans C. Arents

senior IT market analyst

I.T. Works

"Guiding the IT Professional"

Innovation Center, Technologiepark 3, B-9052 Gent (Belgium), Tel: +32 (0)9 241 56 21 - Fax: +32 (0)9 241 56 56

E-mail: hca@itworks.be - Site: http://www.itworks.be/ - Home: http://www.arents.be/

XML for data

- Why XML for data?
- The hype and the reality
- XML data design decisions
  - to validate or not to validate?
  - how to use XML data?
    - XML as a simple syntax
    - XML as serialized relational data
    - XML as highly structured hierarchical data
- Your XML data engineering toolkit
  - creating / transforming / storing / accessing
- Typical usage scenarios

Why XML for data?

- Application integration takes up as much as 40% of a typical IT department's development budget
- What does data integration involve?
  - defining custom data exchange formats
  - developing custom parsers to extract data from messages
  - administering custom message exchanges and data connectors to multiple applications and multiple data sources
- Why is XML a natural choice for data integration?
  - open, non-proprietary standard
  - plaintext, lightweight, easily parsed
  - computing platform and programming language independent
  - flexible enough to model both relational and object-oriented data

The hype behind XML for data

- XML has become a runaway success,
  on a *much* greater scale than its designers anticipated
  - not for the reason they had hoped
    - because separation of form from content is *right*
  - but for a reason they barely thought about
    - data needs to be *free* in order to travel the web
- *"What relational databases have done for data storage,*
  XML will do for data exchange over the Internet"
- All data will turn into XML
  - as it is exchanged between applications inside a company
  - as it moves around between services on the Internet
- *"Data integration over the Internet*
  is the killer application for XML"

The reality behind XML for data

- It's still early days for XML data
  - key supporting standards are still being defined
    - have achieved recommendation: **XML Namespaces**, **XSLT** (XPath)
    - are nowhere near recommendation: XLL, XSLFO, **XML Schemas**, **XQL**
  - good XML support in commercial software is still lacking
    - lots of freeware and shareware for XML hackers
    - "take it or leave it" (beta) support in databases and middleware
  - there are already far too many competing XML data solutions
    - e-commerce: CBL, cXML, ebXML, BizTalk, …
- There will not be one single perfect solution
  - no best practices (yet?)
    - everybody is just hacking along
  - no underlying mathematical model
    - relational algebra vs. forest automata

To validate or not to validate?

- Big question: do I consider my XML to be
  - just a funny syntax?
  - real XML?
- It has to be well-formed
  - to have any form of XML processing (SAX, DOM, XSL, …) at all
- It should be valid
  - to perform quality control on the data and the application using it
    - guarantee what is sent / check what is delivered (at development time)
  - to use the XML schema as a means of documenting the application
  - to have each party understand the data exchange content and rules
- Note:
  - *unvalidated* doesn't necessarily mean *not valid*
  - *validated* doesn't necessarily mean *valid*

XML as a simple syntax

- What:
  - use XML as an alternative to CSV (Comma Separated Values) files, fixed length records, *fill-in-your-own-favorite-format-here*, …
- Where:
  - for simple, repetitive data (e.g. measurement data, sampling data)
  - for data with a fixed structure (e.g. configuration files)
  - for speed (avoid COM/Java overhead)
- How:
  - write/parse/read the XML tags yourself
  - (optionally) test well-formedness
  - Unidex *XML Convert* **http://www.unidex.com/xflat.htm**
    - uses XFlat language to describe the layouts of a wide range of flat files
    - converts the flat file in stream mode into an XML data file

Question: what about XML data size?

- Size impact:
  - rule of thumb: *original size x 3* (x 5 if you like really long tags)
  - reading and writing XML tags takes processing time
- Solution: use syntactical tricks
  - use shorthand element names: *exchange* vs. *storage* format

    `<fn>John</fn><ln>Doe</ln>`

    vs. `<firstname>John</firstname><lastname>Doe</lastname>`
  - use (shorthand) attributes instead of (shorthand) elements

    `<n f="John" l="Doe"/>`

    vs. `<firstname>John</firstname><lastname>Doe</lastname>`
- Disadvantages:
  - XML data become less readable / harder to program
  - extra processing required for (de)shortening element/attribute names

Question: what about XML data size?

- Solution: use compression
  - assess impact by comparing
    - take existing data and encode in XML
    - for batch message transmission: use existing compression
  - weigh increased size against increased flexibility and decide
- How:
  - *WinZip* http://www.winzip.com/
  - *XMLZip* http://www.xmlzip.com/
    - XML files can be compressed from a certain node level downwards
    - XML data can be selected & uncompressed at a specific node rather than uncompressing the entire XML file
  - *XMill* http://www.research.att.com/sw/tools/xmill/
    - groups XML element contents with respect to their meaning
    - exploits similarities between those contents to improve compression

, XML as serialized relational data

- What:
  - use XML as a way of dumping relational data to a text-based file
- Where:
  - for exchanging data between relational-enabled data sources
- How:
  - "poor" tabular XML data: rows with fixed number of columns
  - "rich" tabular XML data: including data types, relationships
    - by using special attributes and the ID/IDREF mechanism
      - http://www.extensibility.com/xml_resources/modeling.htm
    - by serializing graphs of data into a canonical form
      - http://www.biztalk.org/Resources/canonical.asp
    - CARD (**C**ommerce **A**ccelerated **R**elational **D**ata)
      - http://www.infoshark.com/card.shtml
  - do it the Microsoft way: ADO persisted recordset

, XML as serialized relational data

- "Poor" tabular XML data: rows with fixed number of columns

```
<DATABASE>
  <EMPLOYEE NUMBER="2361"
            NAME="Lee Marvin"
            DATE="04/04/88"
            SALARY="60000"/>
  ...
  <EMPLOYEE NUMBER="5315"
            NAME="John Buck"
            DATE="12/10/90"
            SALARY="45000"/>
</DATABASE>
```

XQL query

```
/DATABASE/EMPLOYEE[@NUMBER="2361"]/@NAME
```

```
<DATABASE>
  <EMPLOYEE>
    <NUMBER>2361</NUMBER>
    <NAME>Lee Marvin</NAME>
    <DATE>04/04/88</DATE>
    <SALARY>60000</SALARY>
  </EMPLOYEE>
...
  <EMPLOYEE>
    <NUMBER>5315</NUMBER>
    <NAME>John Buck</NAME>
    <DATE>12/10/90</DATE>
    <SALARY>45000</SALARY>
  </EMPLOYEE>
</DATABASE>
```

XQL query

```
/DATABASE/EMPLOYEE[NUMBER="2361"]/NAME
```

Question: use attributes or elements?

- Use attributes
  - for data typing information
  - for data with no further structure
  - for small number of fixed values
- Use elements
  - for data with a lot of structure
  - for data that changes frequently
  - for data that contains multiple lines
- It's a matter of style … but stay consistent!
- Using attributes is more efficient
  - element markup is more verbose than attribute markup
    - start/end tags instead of attribute name and quotes
  - default values can be assigned from the schema
  - attributes can have much stronger constraint checking
- Choice will also depend on the use of the parser
  - DOM: element and attribute information equally accessible
  - SAX: attribute handling has to built into the element handler

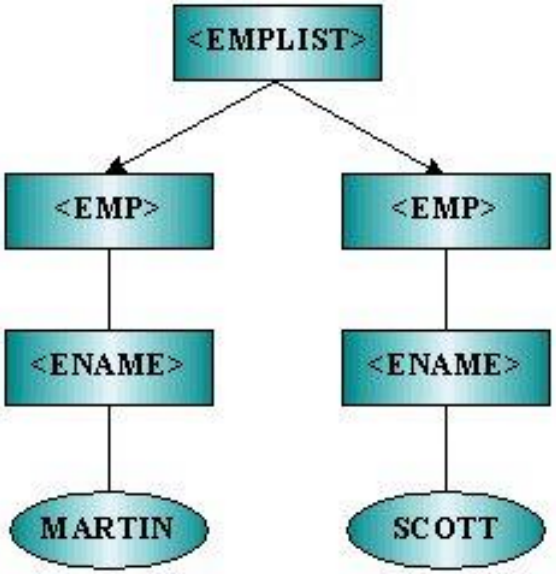
Question: use SAX or DOM processing?

```
<?xml version="1.0"?>
  <EMPLIST>
    <EMP>
      <ENAME>MARTIN</ENAME>
    </EMP>
    <EMP>
      <ENAME>SCOTT</ENAME>
    </EMP>
  </EMPLIST>
```

SAX

```
start document
start element: EMPLIST
start element: EMP
start element: ENAME
characters: MARTIN
end element: EMP
start element: EMP
start element: ENAME
characters: SCOTT
end element: EMP
end element: EMPLIST
end document
```

DOM



Question: use SAX or DOM processing?

- Event-based API: SAX (**S**imple **A**PI for **X**ML)
  - uses *callbacks* to report parsing events to the application
  - application deals with these events through customized *event handlers*

  è use for: - batch processing, retrieving data

  - when the parse tree does not have to manipulated

- Tree-based API: DOM (**D**ocument **O**bject **M**odel)
  - provides *objects* and *methods* to be used by the application
  - application uses these methods to navigate and manipulate the tree

  è use for: - interactive processing, modifying data

  - adding / modifying / deleting elements and attributes

- SAX is faster, better suited for data (development-intensive)

  DOM is more flexible, better suited for documents (memory-intensive)

, XML as serialized relational data

- "Rich" tabular XML data: including data types, relationships

```
...
<!ATTLIST EMPLOYEE.DATE e-dtype NMTOKEN #FIXED 'date'>
<!ATTLIST EMPLOYEE.NAME e-dtype NMTOKEN #FIXED 'string'
                        e-dsize NMTOKEN #FIXED '32'>
...
<!ATTLIST EMPLOYEE pkey_id ID #REQUIRED>
<!ATTLIST EMPLOYEE e-pkey NMTOKEN #FIXED 'EMPLOYEE.NUMBER'>
...
<!ATTLIST REVIEW REVIEW.EMP_NUM_idref IDREF #REQUIRED>
<!ATTLIST REVIEW.EMP_NUM e-fkey NMTOKEN #FIXED 'EMPLOYEE.NUMBER'>
...
```

data types

schema

relation-ships

```
...
<EMPLOYEE pkey_id="EMPLOYEE.2361">
  <EMPLOYEE.NUMBER>2361</EMPLOYEE.NUMBER>
  <EMPLOYEE.NAME>Lee Marvin</EMPLOYEE.NAME>
  <EMPLOYEE.DATE>04/04/88</EMPLOYEE.DATE>
  <EMPLOYEE.SALARY>60000</EMPLOYEE.SALARY>
</EMPLOYEE>
...
<REVIEW REVIEW.EMP_NUM_idref="EMPLOYEE.2361">
  <REVIEW.EMP_NUM>2361</REVIEW.EMP_NUM>
  <REVIEW.DATE>20/12/99</REVIEW.DATE>
  <REVIEW.EVAL>good</REVIEW.EVAL>
</REVIEW>
...
```

data

, XML as serialized relational data

- MS ADO (**A**ctiveX **D**ata **O**bjects) persisted recordset
  - version 2.1: persist recordset as a flat XML file
  - version 2.5: persist recordset
    - as a flat XML file
    - directly into an XML DOM Document object
    - directly into an IIS 5.0 ASP Request or Response object
  - uses the MSXML parser shipped with IE 5
- Advantages:
  - works with any programming language that supports ADO
  - predefined data types with well-defined semantics
- Disadvantages:
  - only works on Microsoft platform
  - only support for XDR schema

, XML as serialized relational data

```
<xml xmlns:s="uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882"
     xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882"
     xmlns:rs="urn:schemas-microsoft-com:rowset"
     xmlns:z="#RowsetSchema">
```

XDR schema
XDR data types
ADO recordset
this recordset

namespaces

```
  <s:Schema id="RowsetSchema">
    <s:ElementType name="row" content="eltOnly" rs:updatable="true">
      <s:AttributeType name="ShipperID" rs:number="1"
        rs:basetable="shippers" rs:basecolumn="ShipperID" rs:keycolumn="true">
        <s:datatype dt:type="int" dt:maxLength="4" rs:precision="10"
          rs:fixedlength="true" rs:maybenull="false"/>
      </s:AttributeType>
      <s:AttributeType name="CompanyName" rs:number="2" rs:nullable="true"
         rs:write="true" rs:basetable="shippers" rs:basecolumn="CompanyName">
        <s:datatype dt:type="string" dt:maxLength="40" />
      </s:AttributeType>
      <s:AttributeType name="Phone" rs:number="3" rs:nullable="true"
        rs:write="true" rs:basetable="shippers" rs:basecolumn="Phone">
        <s:datatype dt:type="string" dt:maxLength="24"/>
      </s:AttributeType>
      <s:extends type="rs:rowbase"/>
    </s:ElementType>
  </s:Schema>
```

schema

```
  <rs:data>
    <z:row ShipperID="1" CompanyName="Speedy&amp;apos;s Express" Phone="(503) 555-9831"/>
    <z:row ShipperID="2" CompanyName="United Package" Phone="(503) 555-3199"/>
    <z:row ShipperID="3" CompanyName="Federal Shipping" Phone="(503) 555-9931"/>
  </rs:data>
</xml>
```

data

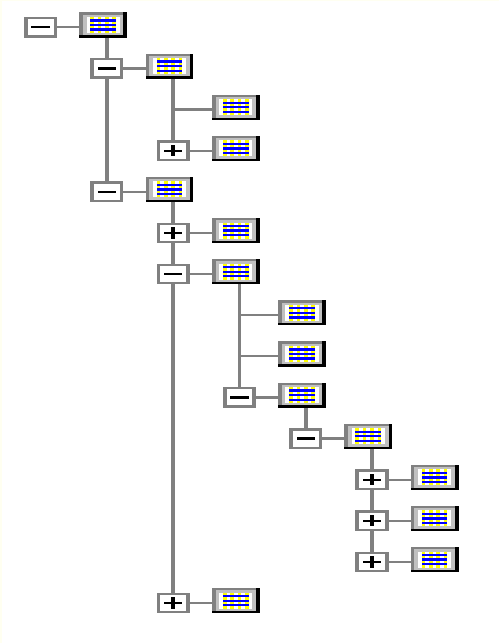
, XML as serialized relational data

- Recordset with pending updates:

```
<rs:data>

  <z:row ShipperID="2" CompanyName="United Package" Phone="(503) 555-3199"/>

<rs:update>
  <rs:original>
    <z:row ShipperID="3" CompanyName="Federal Shipping" Phone="(503) 555-9931"/>
  </rs:original>
  <z:row Phone="(503) 552-7134"/>
</rs:update>

<rs:insert>
  <z:row ShipperID="12" CompanyName="Lightning Shipping" Phone="(505) 111-2222"/>
  <z:row ShipperID="13" CompanyName="Thunder Overnight" Phone="(505) 111-2222"/>
  <z:row ShipperID="14" CompanyName="Blue Angel Air Delivery" Phone="(505) 111-2222"/>
</rs:insert>

<rs:delete>
  <z:row ShipperID="1" CompanyName="Speedy&amp;apos;s Express" Phone="(503) 555-9831"/>
</rs:delete>

</rs:data>
```

ƒ XML as highly structured hierarchical data

- Where:
  - for application integration inside your company
    - not "tight" integration through DCOM/CORBA
    - but "loose" integration through XML data exchange
  - for the exchange of rich business data over the Internet
    - with partners and suppliers
    - with customers
- How:
  - "build or borrow" XML schema
    - do it the Microsoft way: BizTalk.org
    - do it the open standard way: XML.org
  - choose the appropriate data transport protocol
    - synchronous: HTTP / HTTPS
    - asynchronous: MOM (**M**essage **O**riented **M**iddleware)



Question: which XML schema syntax to use?

Data marked up using XML

```
<Person id="40458">
  <Age>37</Age>
  <ZIP>B-9040</ZIP>
</Person>
```

Schema defined using a “classic” DTD

```
<!ELEMENT Person (Age,ZIP)>

<!ELEMENT Age    (#PCDATA)>
<!ELEMENT ZIP    (#PCDATA)>
```

- Problem:
  - XML DTD is in a different syntax than XML documents
  - XML DTD is not expressive enough for rich data modeling
- Solution: **XML Schema**
  - Part 1: **Structure**
    - defining XML DTDs using XML syntax
  - Part 2: **Datatypes**
    - defining datatype constraints to be applied to the contents of XML elements or attributes
- Status: W3C Working Draft (December 17, 1999)

XML Schema

- Enhanced data types
  - 37 versus 10
    - string, boolean, integer, float, double, date, time, …
  - possibility to create custom data types
    - *basetype* restricted through the use of *facets*
    - e.g. for string:
      - enumeration, pattern
      - length, minlength, maxlength
      - minInclusive, maxInclusive, minExclusive, maxExclusive
  - possibility to define custom lexical representations

```
<datatype name="PostalCode"
          source="string">
  <length>6</length>
  <pattern value="B-\d{4}"/>
</datatype>
```

```
<datatype name="PersonAge"
          source="integer">
  <minInclusive>0</minInclusive>
  <maxInclusive>120</maxInclusive>
</datatype>
```

Question: which XML schema syntax to use?

- XML Schema is not yet a standard, so a choice between
  - **DTD** (SGML Document Type Definition)
    - no XML syntax, no data typing, no namespace support
  - **XDR** (XML-Data Reduced, a subset of XML-Data supported by IE 5)
    - XML syntax, basic data typing, simple namespace support
      http://msdn.microsoft.com/xml/XMLGuide/schema-overview.asp
- Choose:
  - DTD
    - if you want to have a choice of tools
    - if you are looking for the open exchange of data
  - XDR
    - if you live in a Microsoft-only world
    - if you want to easily represent relational data
    - caution: will eventually be replaced by XML Schema

Question: how to develop an XML schema?

- Build it yourself
  - not a big technical challenge
    - tough for documents
    - relatively easy for data
  - biggest challenge: getting everybody to agree
    - agree on a shared set of XML document elements
    - agree on the business meanings of different XML schema definitions
- Borrow (and modify) an existing one
  - visit a schema repository (public schema library)
    - do it the Microsoft way: BizTalk.org
    - do it the open standard way: XML.org
  - choose an industry or application schema which suits your needs
  - note: using a schema repository does not mean you will find the one and only schema, there will be a lot of competing schemas

Question: how to build it myself?

- Avoid tags which are
  - obscure
  - inappropriate
- Build a data model as restrictive as possible
  - as little as possible optional
  - most elements can only appear once
  - have a well-defined list of attribute values
  - **è** applications can rely more heavily on XML validation
  - **è** reduces the amount of code the application developer has to write
- Content-oriented DTDs
  - model closely how the data should work
  - use hierarchies to reflect containment relationships

Question: how to borrow an existing one?

	BizTalk.org repository	XML.org repository
Control	Microsoft	Sponsors: Sun, IBM, Oracle, SAP, CommerceOne, …
Advisory members	Membership not published	The 150 members of OASIS
Schema format	XDR with BizTalk wrapper tags, future support for XML Schema	DTDs, future support for XML Schema
Services provided	Discovery and hosting (search for and retrieve schemas and supporting documents from repository)	Current: catalog of links to schema sites Future: discovery and hosting, referral of queries to alternate repositories
Schemas listed	Hosts ±250 schemas in 11 categories from more than 50 organizations	Links to over 100 schema-producing organizations listed in 45 categories
Supporting material	Sample document instance and schema documentation	Future: DTD/schema and supporting files
Search interface	Keyword search or select industry and organization	Browse list organized by industry and organization

Question: how to do it the Microsoft way?

- **BizTalk**
  - Microsoft initiative to define an XML framework for business process integration through the reliable exchange of XML business documents
  - goals:
    - develop a **schema design framework**
      - set of XDR schema conventions for describing data and exchanging messages
    - maintain a **schema repository**
      - public BizTalk schemas validated, registered and stored at the BizTalk.org site
    - deliver a **set of tools**
      - proof of concept BizTalk tools, core of all future Microsoft e-business solutions
  - (very) slow progress:
    - beta version of some parts (promised mid 1999, delivered end of 1999)
      - XML Schema tool, XML Mapping Tool, BizTalk server
    - final product delivery planned for mid 2000, integration has yet to start
      - MSN Portal, MS Commerce Platform, Windows 2000, (Back)Office 2000

Question: what's in a BizTalk message?

BizTalk Message

Transport-specific envelope

BizTalk Document

BizTalk document header

```
<Route>
  <From locationID="…" locationType="…"
        process="…" path="…" handle="…">
  <To locationID="…" locationType="…"
      process="…" path="…" handle="…">
</Route>
```

BizTalk document body

Business document

XDR schema
XML data

- a BizTalk message is an "e-commerce wrapper"

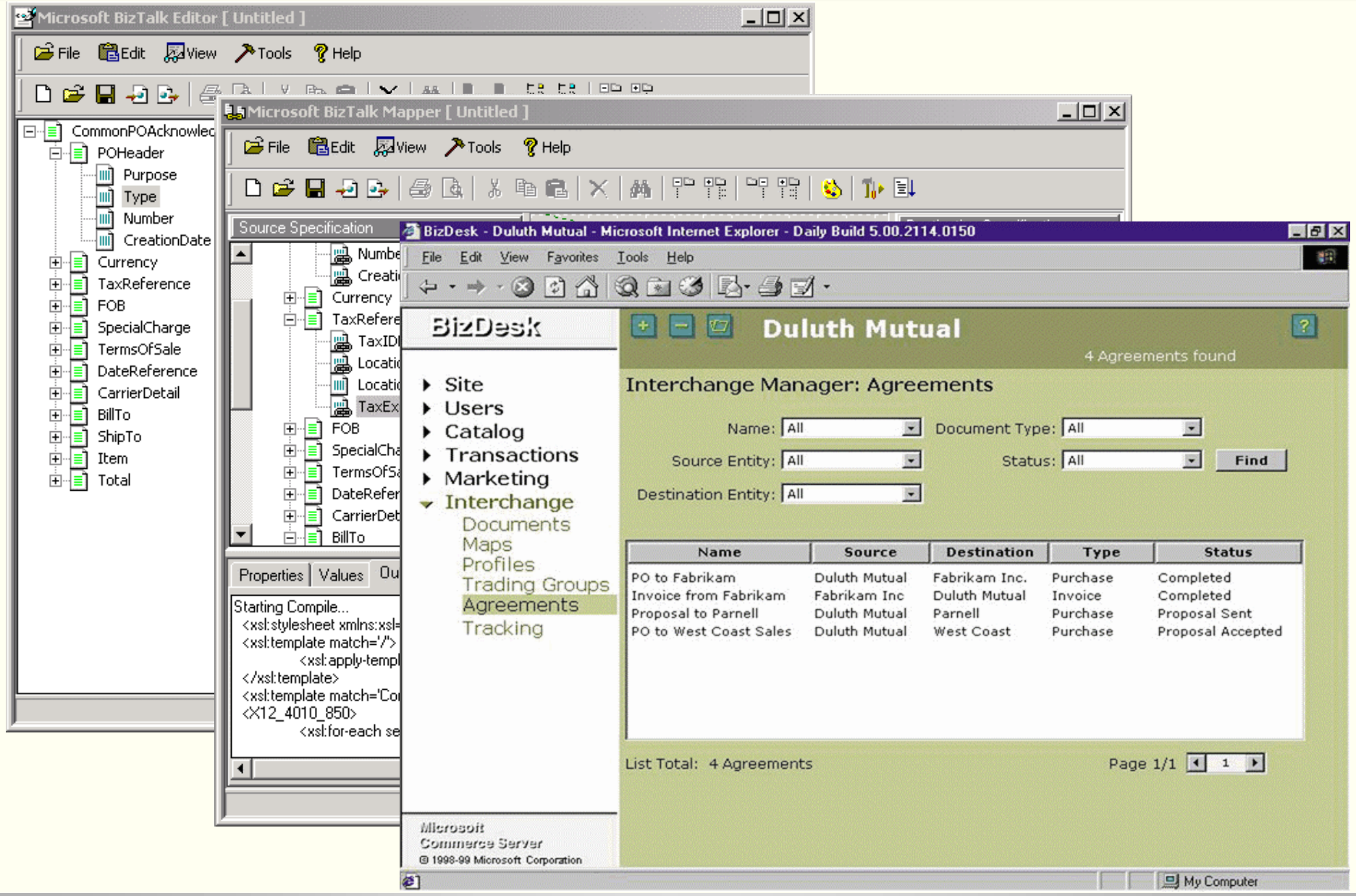
BizTalk terminology

The following are the key components of the BizTalk Framework:

- **Business Document:** The Business Document is a stream of XML containing business transaction data. It can be a purchase order, an invoice, a sales forecast or any other business information. A Business Document forms the payload of a message. The BizTalk Framework does not define the schema of individual documents.
- **Schema:** A schema is the metadata used to describe the content and structure of a Business Document.
- **BizTags:** BizTags are the set of XML tags used to tell a computer how to handle Business Documents. The BizTags are added as an XML envelope or wrapper for a Business Document by an application. They are processed by the BizTalk Server or by other applications.
- **BizTalk Document:** A BizTalk Document is a Business Document that uses the BizTags.
- **BizTalk Message:** BizTalk Messages are used to send BizTalk Documents and any related files between BizTalk Servers.
- **BizTalk Server:** A BizTalk Server handles the processing and functionality defined in the BizTalk Framework specifications.

Source: Microsoft

Question: what's in the BizTalk toolkit?



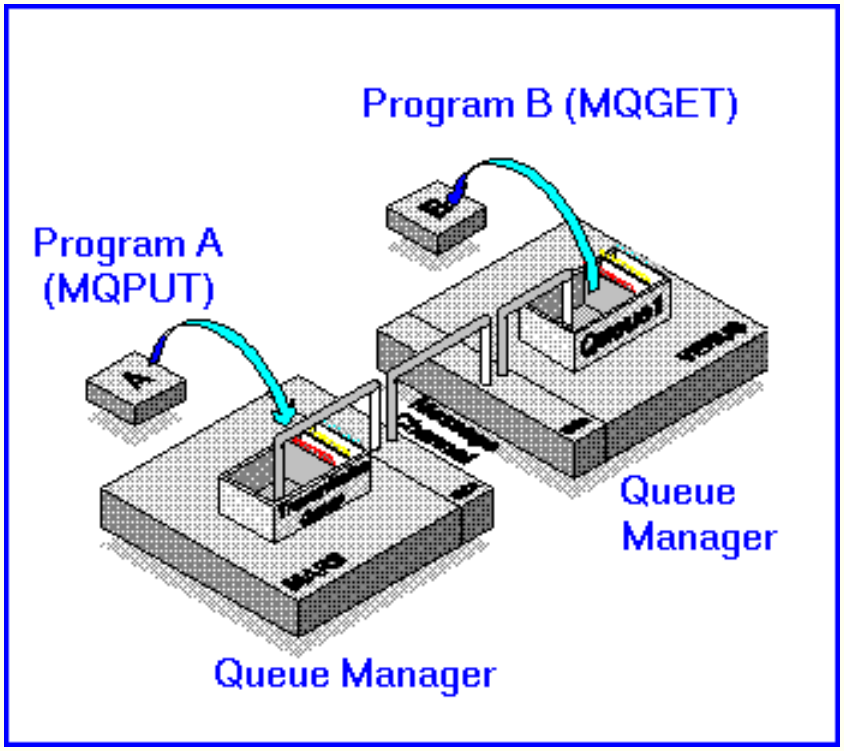
Question: how to cope with 1 XML schemas?

- Solution: transform XML adhering to one schema into another
  - own custom-built application code
  - XSLT (XSL **T**ransformations): stylesheet + processor
- When to use own code:
  - the mapping between the two data formats is straightforward
    - e.g. serialized relational data into serialized relational data
  - the target XML data format is under your control
  - performance is important
- When to use XSLT:
  - the target XML data format is not completely under your control
    - e.g. your internal XML data into an external partner's XML data
  - the target data format is for display e.g. HTML or WML
  - performance is less important

Question: which data transport protocol?

- XML data transport protocols
  - synchronous: HTTP / HTTPS
    - HTTP client (ActiveX or Java)
    - Web server (with ASP or Java servlet)
  - asynchronous: MOM (**M**essage **O**riented **M**iddleware)
    - MQSeries, MSMQ, EntireX, …
- Synchronous data transport protocols
  - for fast, reliable networks
  - for point-to-point system interactions
- Asynchronous data transport protocols
  - for scalability
  - for slow, unreliable networks
  - for geographically widely distributed systems

Asynchronous data transport protocol



- Message oriented middleware
  - guaranteed, once-only exchange of messages between applications
  - here: messages with XML payload

- IBM *MQSeries*
  - MOM for (almost) any platform
  - *MQSeries Integrator*
    - message routing and content transformation & formatting for message brokering
  - MQSeries messages in XML will be able to start MQSeries Integrator activities (1Q2000)
- Microsoft *MSMQ*
  - MOM for the Windows NT platform
  - tight integration with MTS and IIS
  - version 3.0 will also support HTTP message delivery

Synchronous data transport protocol

- Microsoft: Microsoft *MSXML* parser (IE 5.01 version)
  - provides support for HTTP-based XML data transport services
    - client: send XML request document through HTTP POST, receive the XML response document, and have the Microsoft XMLDOM parse that response
    - server: receive XML request document / send XML response document

client (only)

```
<script language="JScript">
  function PostOrder(xmldoc)
  { var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    xmlhttp.Open("POST", "http://www.abc.com/processorder.asp", false);
    xmlhttp.Send(xmldoc);
    return xmlhttp.responseXML; }
</script>
```

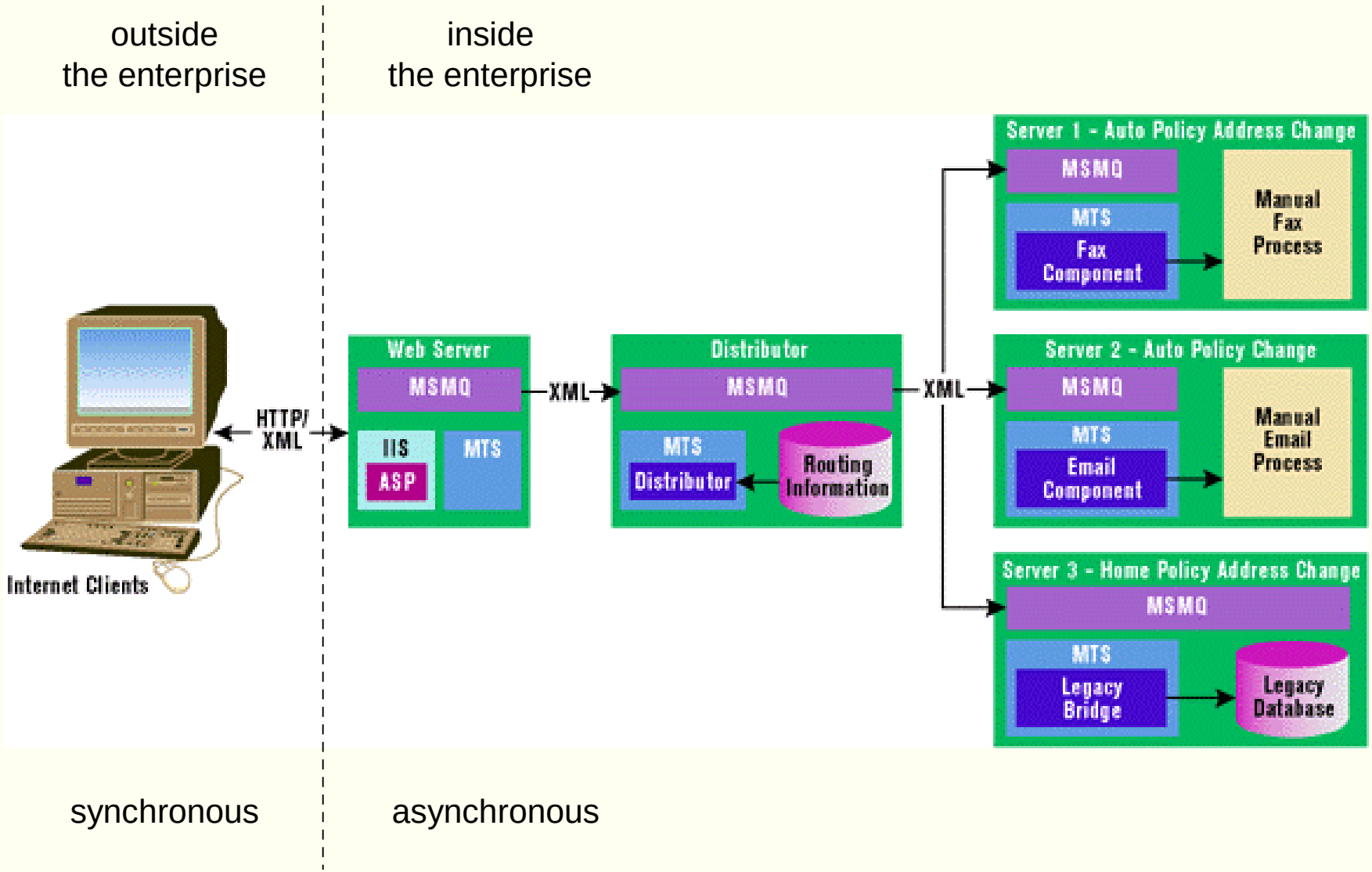
server

```
<%@ language="JScript" %>
<%
    // Load the posted XML document
    var doc = Server.CreateObject("Microsoft.XMLDOM");
    doc.load(Request);
    var result = Server.CreateObject("Microsoft.XMLDOM");
    // now process the order and build the result document
    ...
    // Send the result document back
    Response.ContentType = "text/xml";
    result.save(Response);
%>
```

Synchronous data transport protocol

- Sun: Java Project X Release 2
  - provides support for HTTP/HTTPS-based XML data transport services
    - client: send XML request document through HTTP/HTTPS POST
    - server: receive XML request document / send XML response document
  - *XmlRpcClient* and *XmlRpcServlet* classes
    - allows you to add application-specific behaviours
    - to handle specific kinds of request / response documents
- Note: does not provide a complete data transport protocol
  - WDDX (**W**eb **D**istributed **D**ata E**x**change)
    - **http://www.wddx.org/**
  - SOAP (**S**imple **O**bject **A**ccess **P**rotocol)
    - **http://msdn.microsoft.com/xml/general/soapspec-v1.asp**
  - XML-RPC
    - **http://www.xml-rpc.com/**

Use both data transport protocols



Your XML data engineering toolkit

- Creating XML data
  - XML schema designer
    - to define XML data schemas
    - to visualize XML data schemas
  - XML editor
    - to create sample XML data
    - to understand XML data format
  - XML parser
    - to perform stand-alone validation
    - to use in your own application

XML schema designer

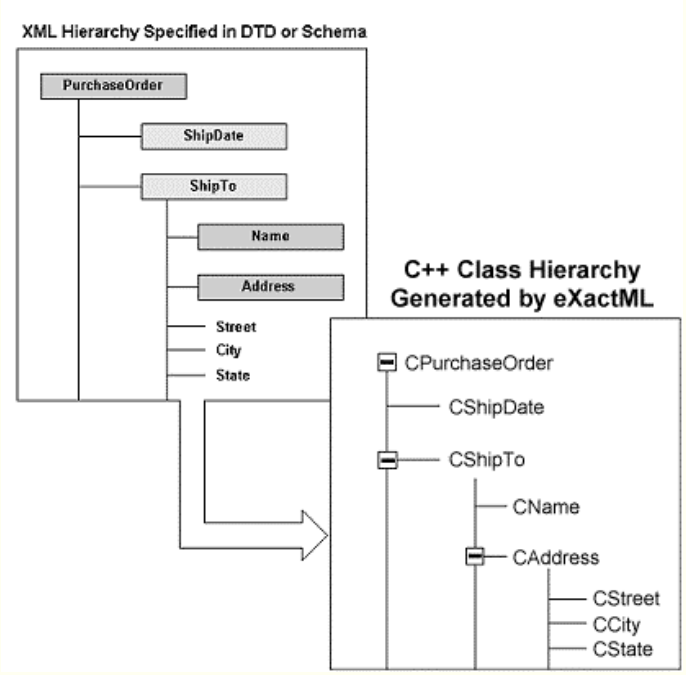
- For DTD schemas:
  - Microstar *Near & Far Designer* http://www.microstar.com/
    - easy to use graphical interface for creating valid DTDs
    - extensive reporting functionalities
      - DTD structure maps for analysis and design reviews
      - element and attribute reports for SAX/DOM programming
- For XDR schemas:
  - Extensibility *XML Authority* http://www.extensibility.com/
    - harder to use graphical interface, but good for data format design
    - generates different XML Schemas:
      - DTD, XDR, DCD, SOX, DDML, …
      - import and export BizTalk compatible XDR schemas
    - extensive documentation on XML data modeling
      - set of guidelines and best practices
      - XMLSchema site: schema validation/conversion http://apps.xmlschema.com/

XML editor

- For XML data formats based on DTD schemas:
  - Icon Information-Systems *XML Spy* http://www.xmlspy.com/
    - has an interface that is well suited for XML data editing
    - helps you write an XML file which conforms to a DTD schema
    - has a “database view” for displaying sequences of repeating elements
    - can create (and edit afterwards) a DTD schema from well-formed XML data
- For XML data formats based on XDR schemas:
  - Stilo *XMLDeveloper* http://www.stilo.com/
    - helps you write an XML file which conforms to an XDR schema
    - can extend an XDR schema while you add data elements
    - can create an XDR schema from well-formed XML data
    - can check an XDR schema for syntactical correctness

Validating XML parser

- Java:
  - IBM *XML4J*: fastest, Sun *Java Project X*: most compliant, …
- Other languages:
  - IBM *XML4C* (C++), *XML::Parser* (Perl), …
- Parser generator:
  - turn DTD or XDR schema into Visual Basic, Java or C++ classes
  - check well-formedness and validity
  - use classes directly instead of SAX/DOM
    - e.g. *eXactML* `http://www.bristol.com/`



Validating XML parser

- Windows:
  - Microsoft *MSXML* (IE 5.01 version) to be used in ASP, VB, …
    - warning: supports pre-standard version of XSL (“MS-XSL”)
  - Microsoft *MSXML Version 3.0* (October 2000)
    - completely supports standard XSLT and XPath
    - user-defined extension functions in VBScript or JScript
    - optimizations for improved document throughput (2/3x faster)
      - e.g. server-side XSL stylesheet caching
    - can be installed alongside or as a replacement of the original *MSXML*
    - COM object accessible from C++, Visual Basic and scripting environments
- **http://www.netcrucible.com/xslt/msxml-faq.htm**

Your XML data engineering toolkit

- Transforming XML data
  - XSL editor
    - to create XSL stylesheets
    - to understand XSL transformations
  - XSL processor
    - to perform stand-alone transformations
    - to use in your own application

XSL editor

- Windows:
  - eXcelon Corp. *Stylus* http://www.exceloncorp.com/stylus/
    - tree view of the structure of the XML data
    - syntax-aware stylesheet editing and debugging
    - can use both built-in XSL processor and IE 5 XSL processor
- Java:
  - IBM *XSL Editor* http://www.alphaworks.ibm.com/
    - tree view of the structure of the XML data
    - automatically generates XPath syntax from sample XML data
    - *XSL Trace*
      - steps through XSL stylesheet and shows active transformation rules
    - *Visual XML Transformation*
      - input the source DTD and visually define the target DTD
      - generates XSL stylesheet that transforms source into target XML data

XSL processor

- Windows:
  - James Clark *XT* http://www.jclark.com/xml/xt.html
    - to be used from the command line
    - adheres very closely to the official XSL standard
    - has extensions for calling Java code and handling multiple documents
  - Infoteria *iXSLT* http://www.infoteria.com/
    - (expensive) XSL processor, available as a EXE / DLL / COM
- Java:
  - IBM *LotusXSL* http://www.alphaworks.ibm.com/
    - to be used from the command line, or wrapped in an applet or servlet
    - is most widely used for XML to (X)HTML transformation
    - *XML Enabler*
      - sniffs incoming browser type
      - selects appropriate XSL stylesheet

Your XML data engineering toolkit

- Storing XML data
  - XML data stores:
    - Bluestone *XML Suite* http://www.bluestone.com/xml/
    - Software AG *Tamino* http://www.softwareag.com/tamino/
    - Object Design *eXcelon* http://www.odi.com/excelon/
  - relational databases:
    - Microsoft *SQL Server* http://msdn.microsoft.com/sqlserver/
      - XML Technology Preview
    - IBM *DB2 Universal DB* http://www-4.ibm.com/software/data/
      - DB2 XML Extender
    - Sybase *Adaptive Server* http://www.sybase.com/
      - Java in SQL Extensions
    - Informix *Internet Foundation.2000* http://www.informix.com/xml/
      - Web DataBlade
    - *Oracle 8i* http://www.oracle.com/xml/

Your XML data engineering toolkit

- Accessing XML data
  - from XML files:
    - Merant *DataDirect for XML* http://www.merant.com/datadirect/
  - from relational databases:
    - Rogue Wave *XML-DB Link* http://www.roguewave.com/xml/
    - Stonebroom *ASP2XML* http://www.stonebroom.demon.co.uk/
    - IBM *XLE* http://www.alphaworks.ibm.com/tech/xle/

Conclusions

- Don’t use XML for data …

  unless you have a good reason to do so

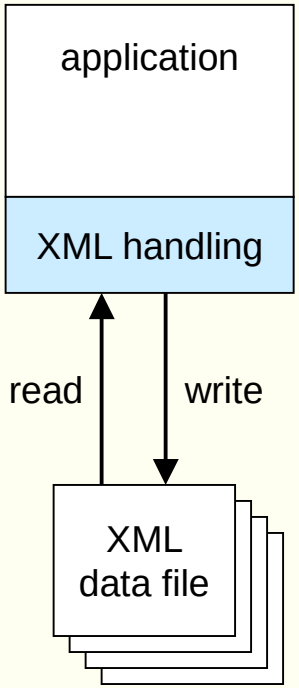
- So don't use XML for data
  - in real-time transaction environments
  - when you are in full control of the data flow and the applications at both ends of the data flow

- But do use XML for data
  - when you need to exchange data on the Web
  - when you want to open up your applications
  - when you don't know what client/server you will have to interact with in the future

XML for data scenario's

- Format for data storage
  - for a single application
- Format for data exchange
  - between two applications
  - between an application and a database
  - between two different types of databases
- Format for Web data exchange
  - between a Web back-end and a Web browser
  - between a Web back-end and a Web application
  - between two different types of Web back-ends

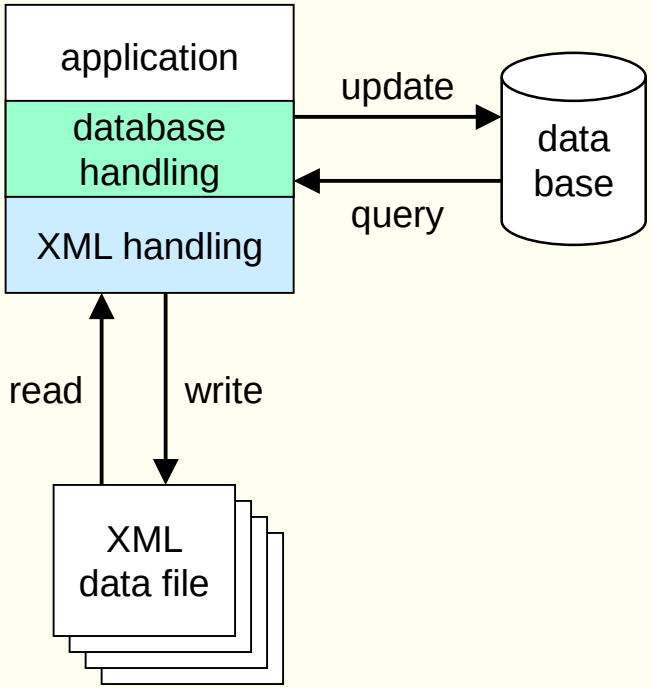
Format for data storage



for a single application

- use:
  - application initialization files
  - configuration & setup files
  - data storage files
- how:
  - XML handling
    - write yourself
    - buy off-the-shelf

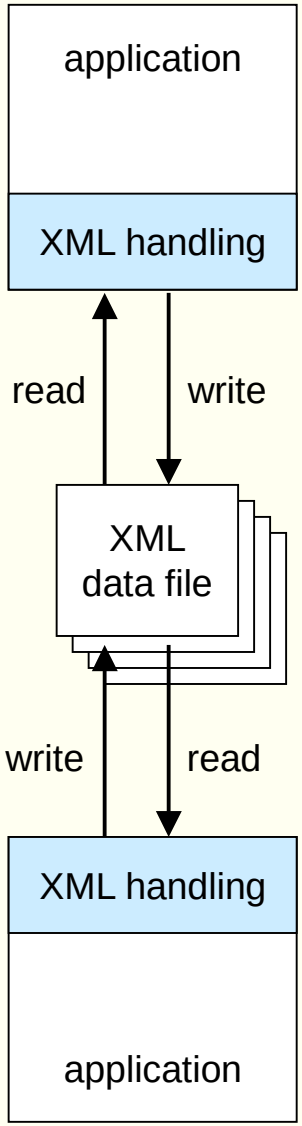
Format for data storage



for a single application

- use:
  - application initialization files
  - configuration & setup files
  - data storage files
- how:
  - XML handling
    - write yourself
    - buy off-the-shelf
  - database handling
    - write yourself
      - ODBC
      - JDBC
      - ...
    - buy of-the-shelf

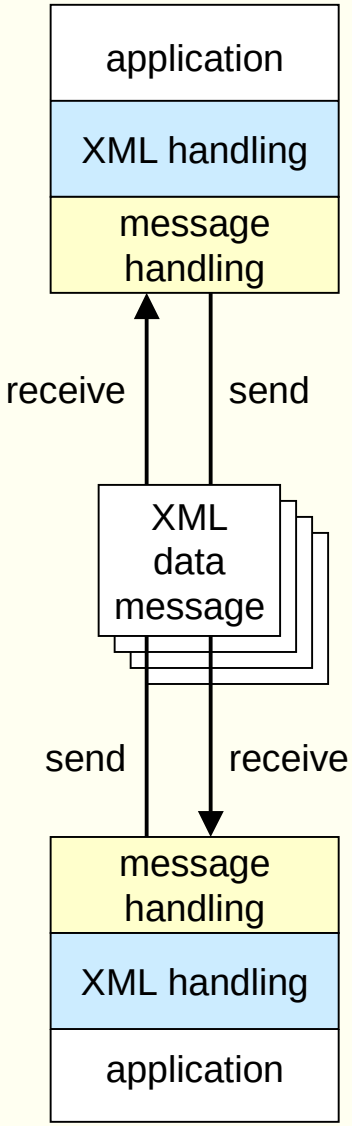
Format for data exchange



between two applications

- use:
  - exchange common data
  - import/export data
- how:
  - XML handling
  - exchange handling
    - file-based (NFS, FTP)

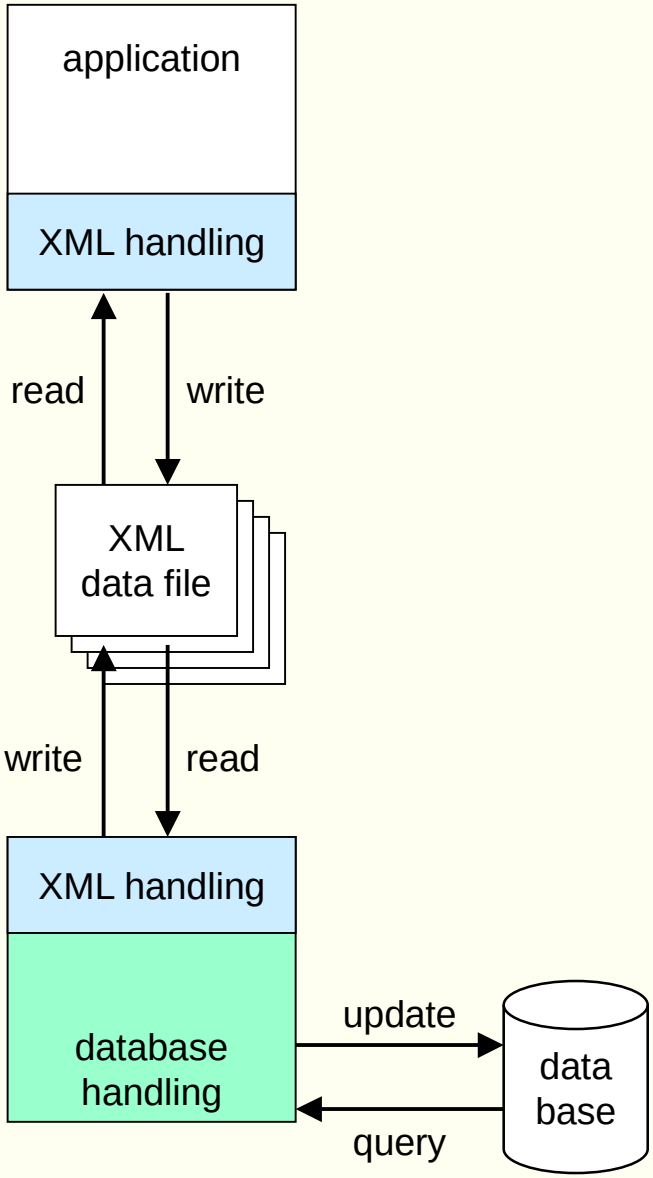
Format for data exchange



between two applications

- use:
  - exchange common data
  - import/export data
- how:
  - XML handling
  - exchange handling
    - file-based (NFS, FTP)
    - message-based
      - MQSeries
      - MSMQ
      - Oracle AQ
      - EntireX
      - …

Format for data exchange

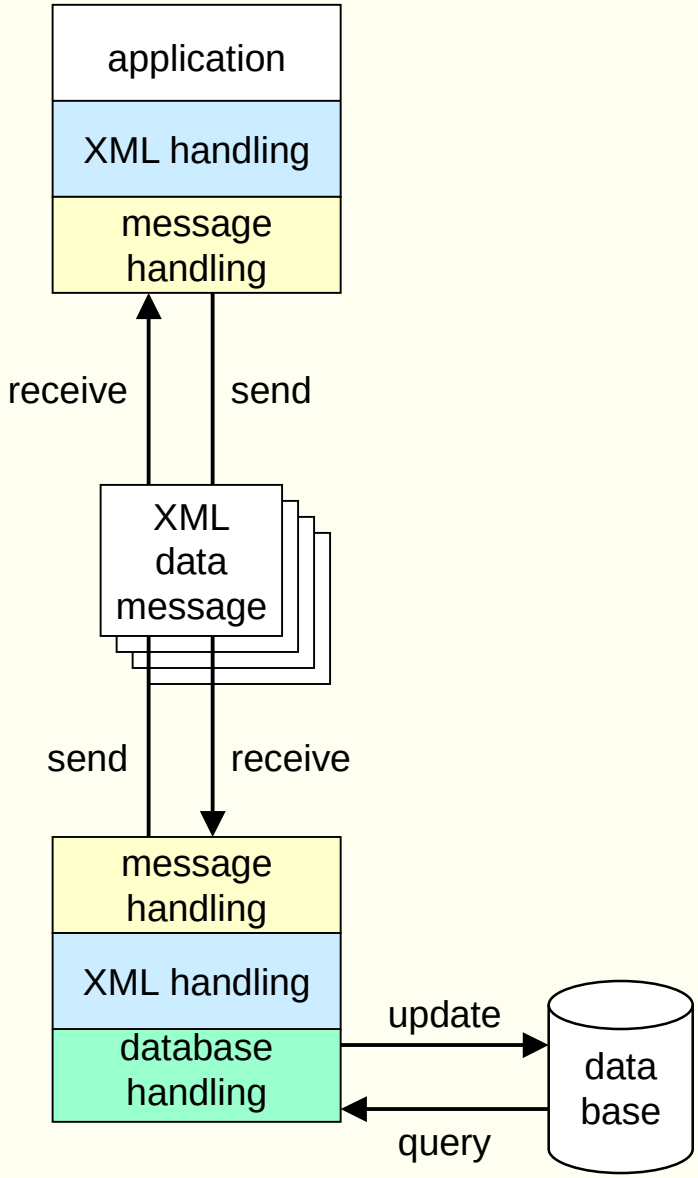


between an application

and a database

- use:
  - import/export data
  - copy of (part of) database
- how:
  - XML handling
  - database handling
    - write yourself
    - use database's support for XML handling

Format for data exchange

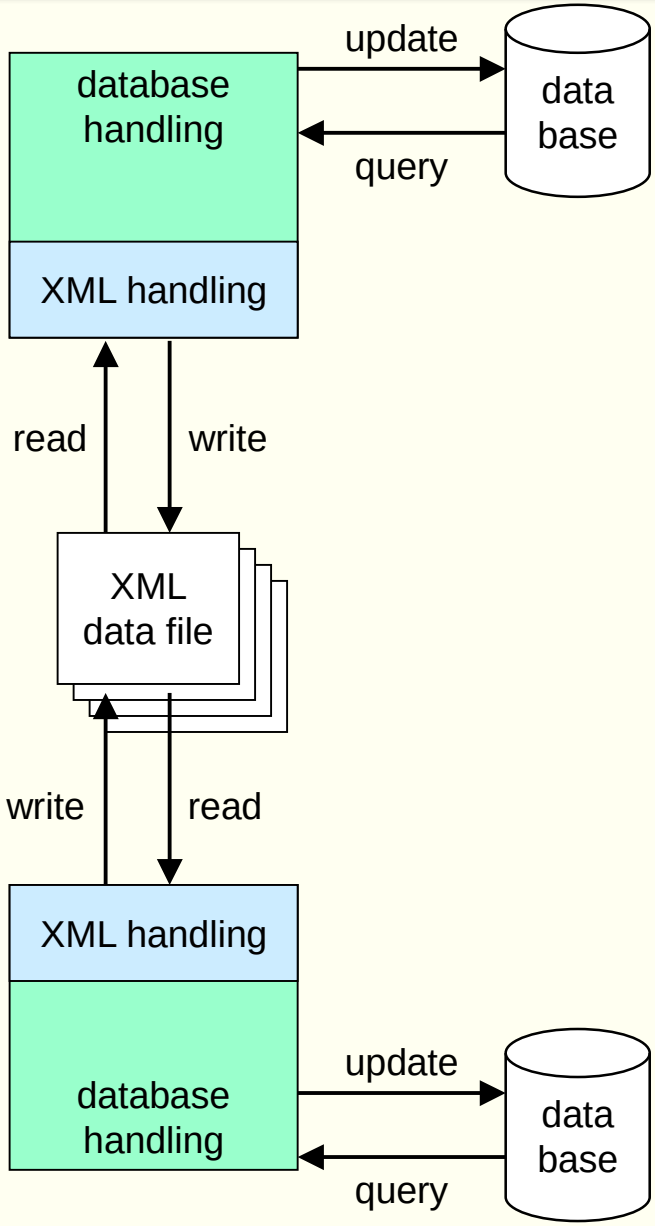


between an application

and a database

- use:
  - import/export data
  - copy of (part of) database
- how:
  - XML handling
  - database handling
    - write yourself
    - use database's support for XML handling
  - exchange handling

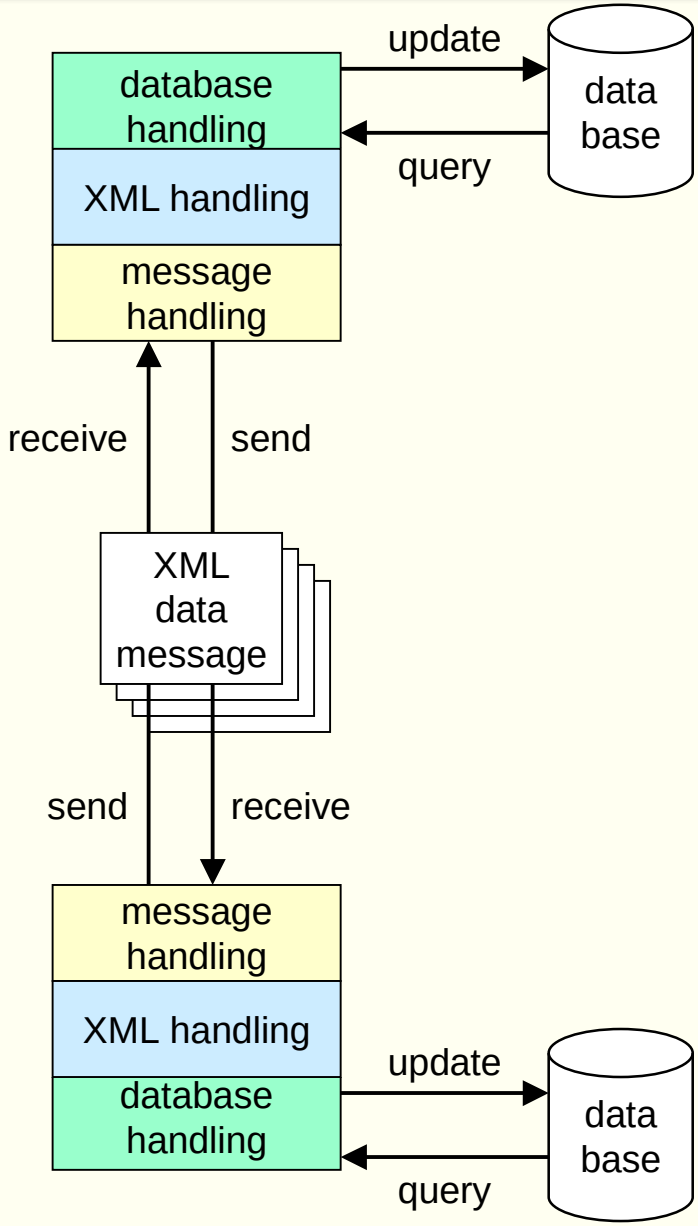
Format for data exchange



between two different types of databases

- use:
  - import/export data
  - share (part of) database
- how:
  - XML handling
  - database handling
    - write yourself
    - use database's support for XML handling

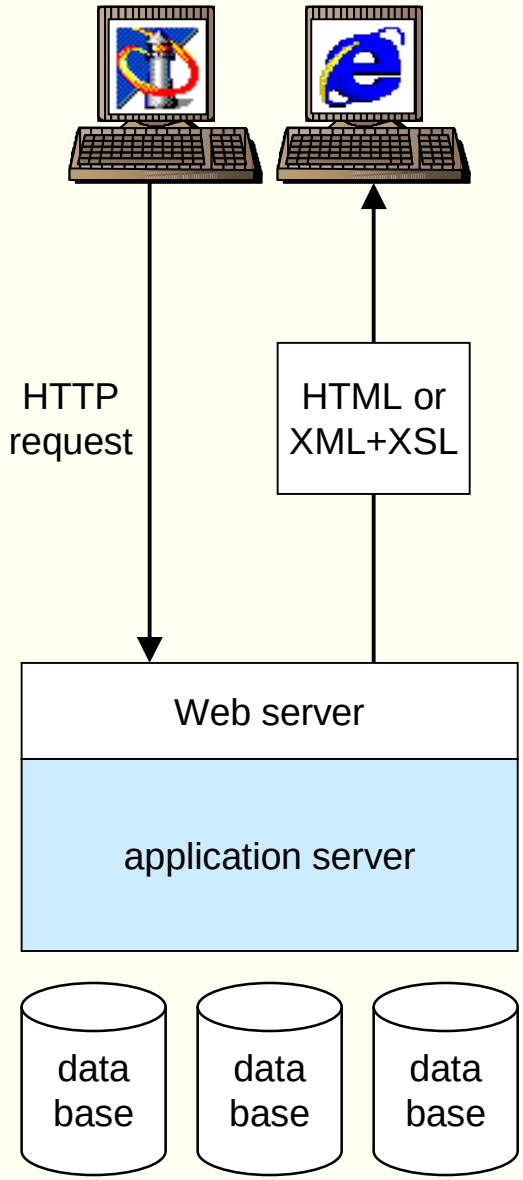
Format for data exchange



between two different types of databases

- use:
  - import/export data
  - share (part of) database
- how:
  - XML handling
  - database handling
    - write yourself
    - use database's support for XML handling
  - exchange handling

Format for Web data exchange

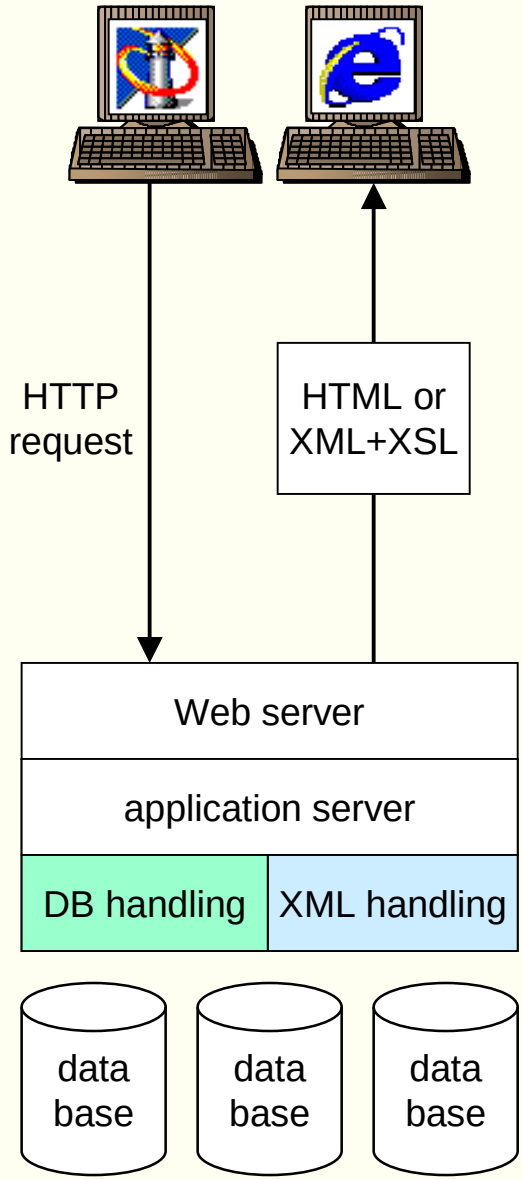


between a Web back-end

and a Web browser

- use:
  - browser as thin client
  - hide data sources from user
- how:
  - XML handling
    - client-side
    - server-side
  - database handling
    - write yourself
      using an application server

Format for Web data exchange

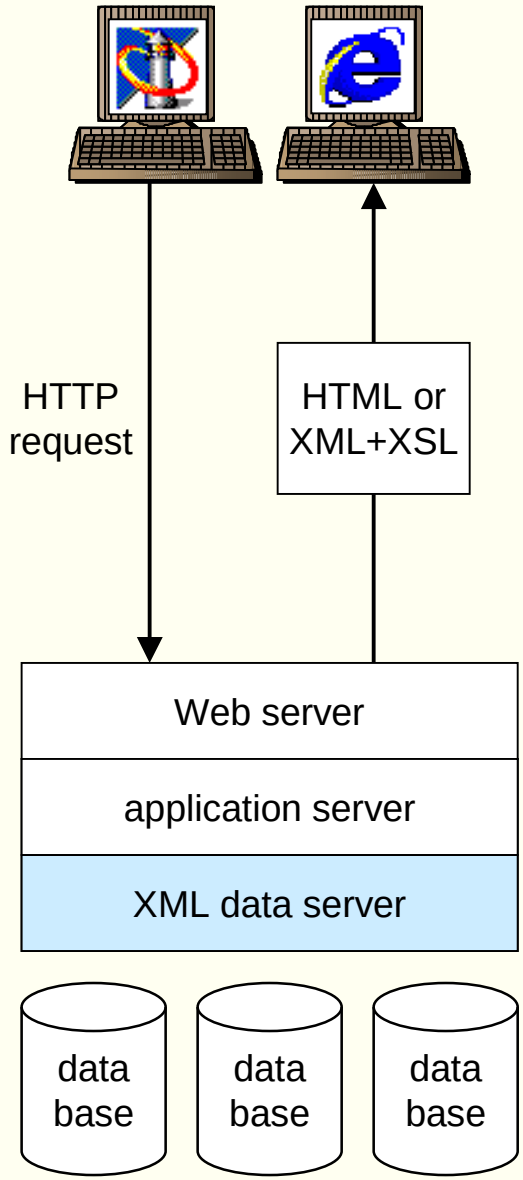


between a Web back-end

and a Web browser

- use:
  - browser as thin client
  - hide data sources from user
- how:
  - XML handling
    - client-side
    - server-side
  - database handling
    - buy of-the-shelf

Format for Web data exchange

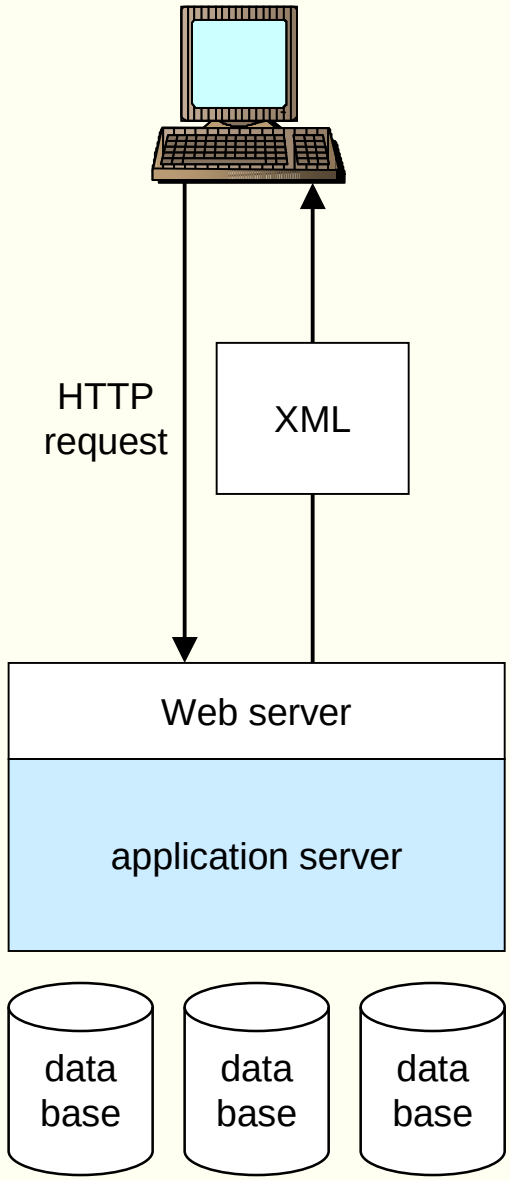


between a Web back-end

and a Web browser

- use:
  - browser as thin client
  - hide data sources from user
- how:
  - XML handling
    - client-side
    - server-side
  - database handling
    - use services of
      an XML data server

Format for Web data exchange

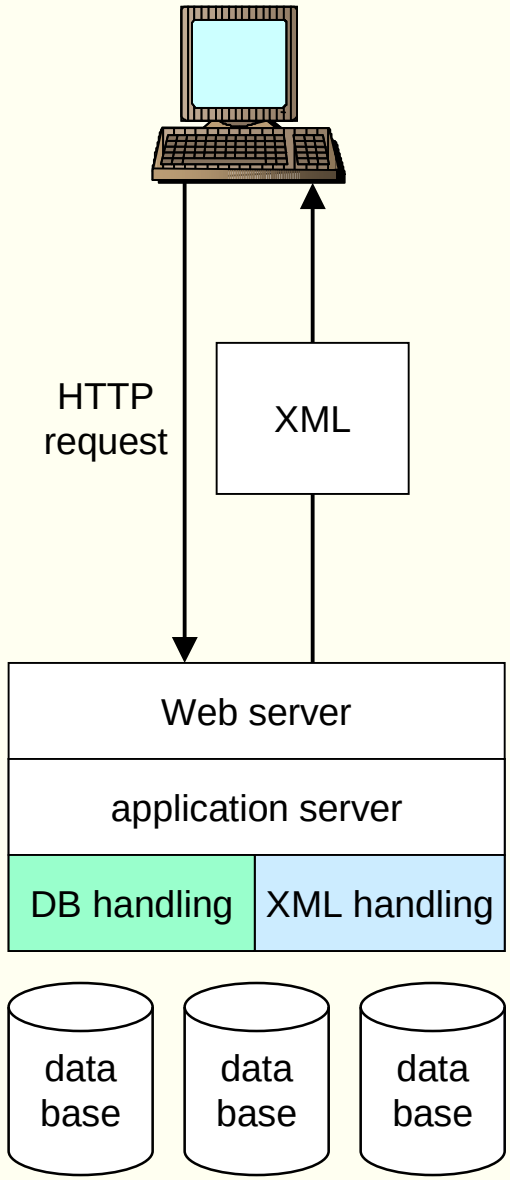


between a Web back-end

and a Web application

- use:
  - Web-based client/server
  - hide data sources from application
- how:
  - XML handling
    - client-side
      - DOM
      - SAX
  - database handling
    - write yourself
      using an application server

Format for Web data exchange

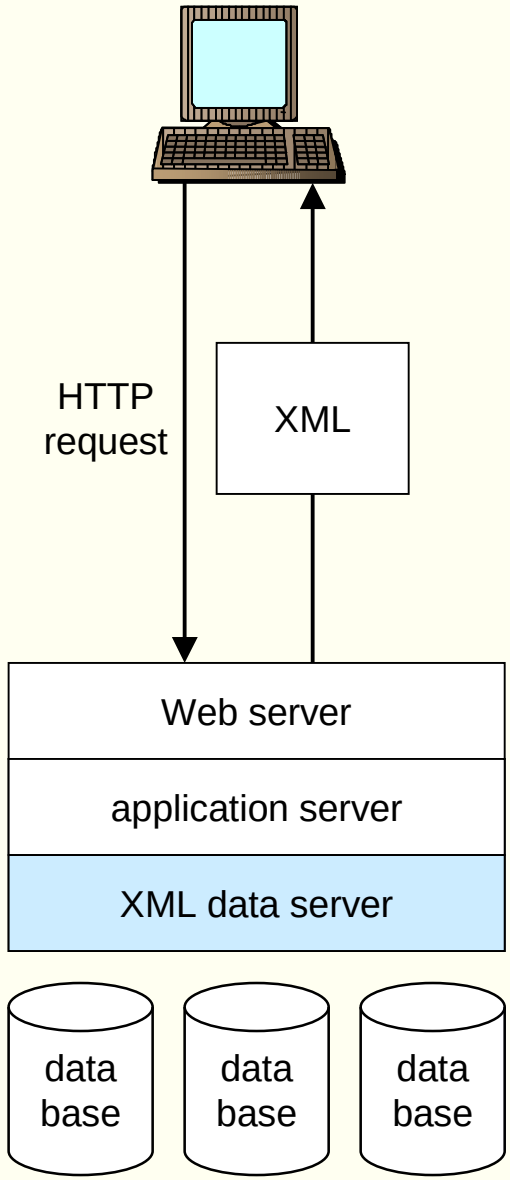


between a Web back-end

and a Web application

- use:
  - Web-based client/server
  - hide data sources from application
- how:
  - XML handling
    - client-side
      - DOM
      - SAX
  - database handling
    - buy of-the-shelf

Format for Web data exchange

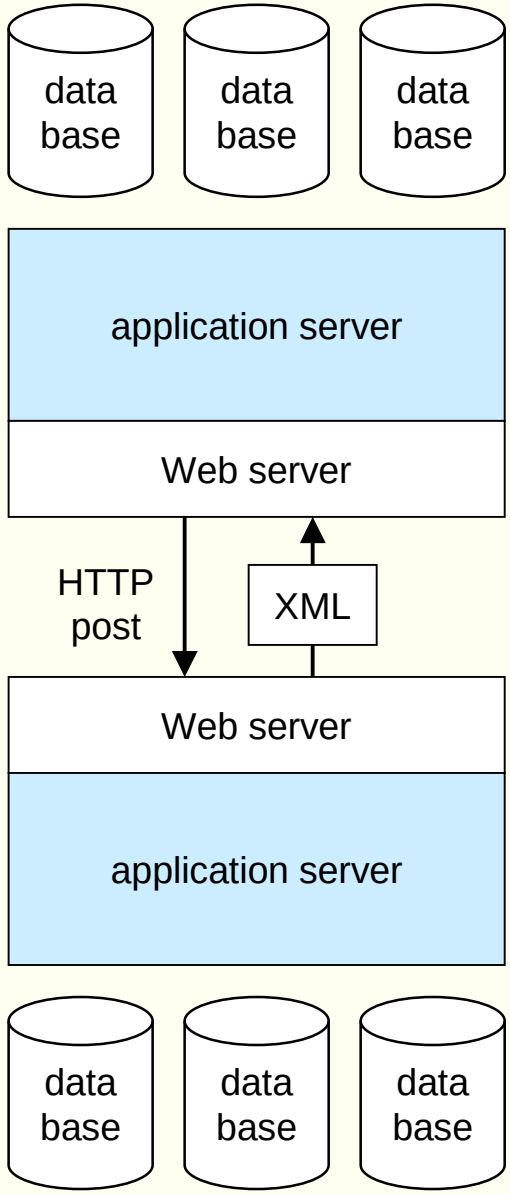


between a Web back-end

and a Web application

- use:
  - Web-based client/server
  - hide data sources from application
- how:
  - XML handling
    - client-side
      - DOM
      - SAX
  - database handling
    - use services of
      an XML data server

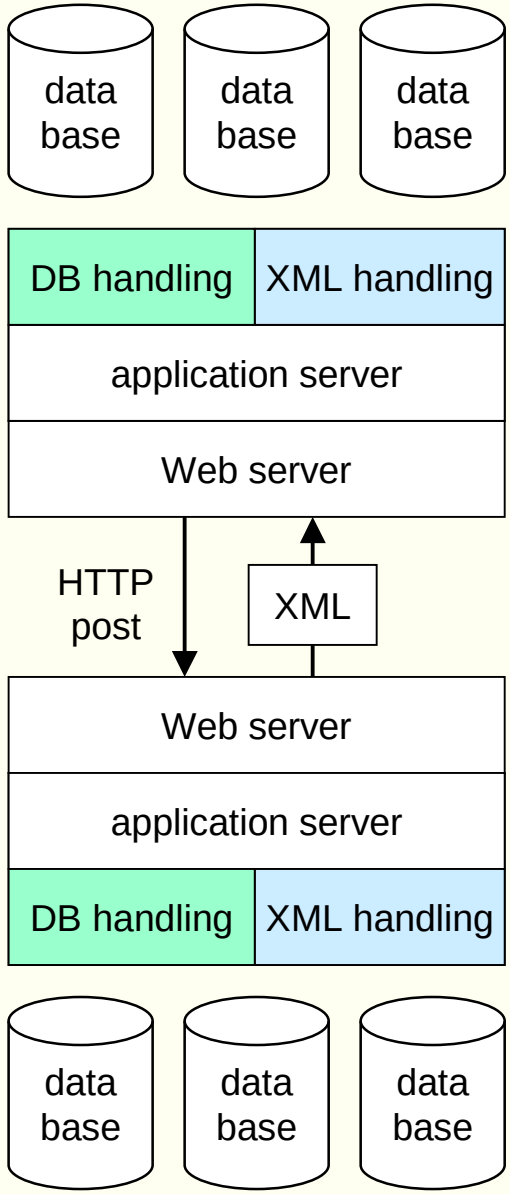
Format for Web data exchange



between two different types of Web back-ends

- use:
  - exchange common data
  - import/export data
- how:
  - XML handling
    - server-side
      - DOM
      - SAX
  - database handling
    - write yourself using an application server

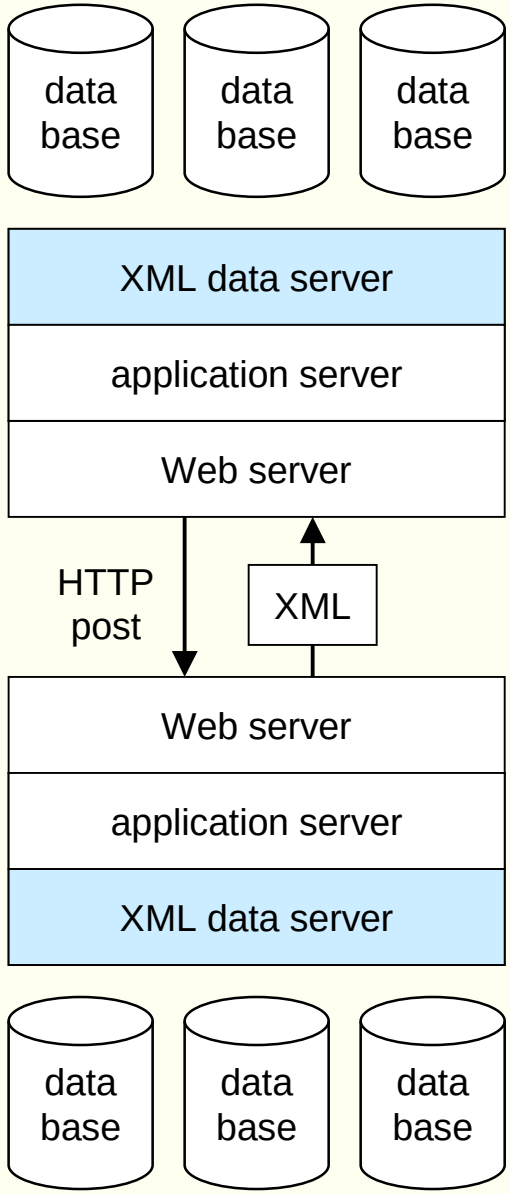
Format for Web data exchange



between two different types of Web back-ends

- use:
  - exchange common data
  - import/export data
- how:
  - XML handling
    - server-side
      - DOM
      - SAX
  - database handling
    - buy of-the-shelf

Format for Web data exchange



between two different types of Web back-ends

- use:
  - exchange common data
  - import/export data
- how:
  - XML handling
    - server-side
      - DOM
      - SAX
  - database handling
    - use services of an XML data server

General XML sites

- The **I.T. Works XML** page: **http://www.itworks.be/XML/**
- The XML home page: **http://www.xml.com/**
- Industry leaders
  - **http://java.sun.com/xml/**
  - **http://www.oracle.com/xml/**
  - **http://msdn.microsoft.com/xml/**
  - **http://www.ibm.com/developer/xml/**
  - **http://developer.netscape.com/xml/**
- Belgian-Luxembourgian SGML/XML Users Group
  - **http://www.sgmlbelux.be/**
- XML newsgroups
  - **news:comp.text.xml** **news:microsoft.public.xml**

XML developer sites

- The *XML Zone* at *DevX*
  - **http://www.xml-zone.com/**
- The *XMLTree* XML directory
  - **http://www.xmltree.com/**
- ZDNet Developer's *XML section*
  - **http://www.zdnet.com/devhead/filters/xml/**
- Web Developer's Virtual Library *XML section*
  - **http://wdvl.internet.com/Authoring/Languages/XML/**
- *VBXML* for Visual Basic and ASP XML developers
  - **http://www.vbxml.com/**

XML developer reading material

- Microsoft's *XML Developer's Guide*
  - **http://msdn.microsoft.com/xml/xmlguide/**
- Microsoft's *XSL Developer's Guide*
  - **http://msdn.microsoft.com/xml/xslguide/**
- Microsoft's *XML & XSL Samples and Demos*
  - **http://msdn.microsoft.com/xml/demos/**
- Sun's Java/XML Tutorial *Working with XML*
  - **http://java.sun.com/xml/tutorial_intro.html**
- Real world applications using XML and Java2
  - **http://developerlife.com/**