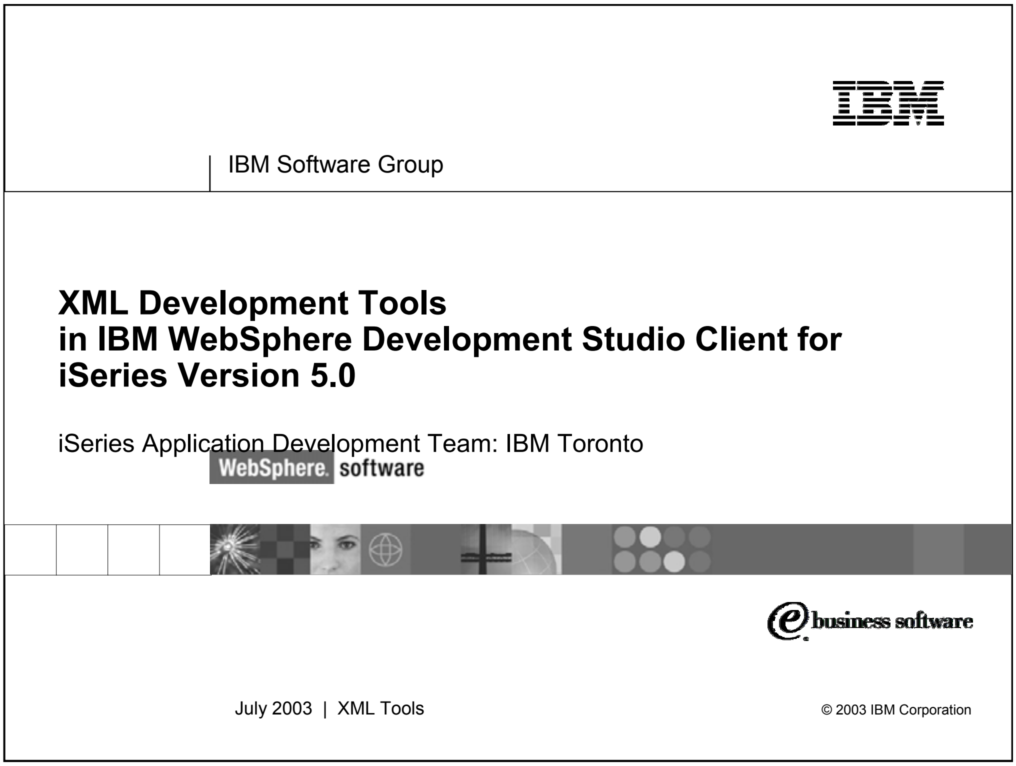


IBM

IBM Software Group

# XML Development Tools in IBM WebSphere Development Studio Client for iSeries Version 5.0

iSeries Application Development Team: IBM Toronto

WebSphere. software

e business software

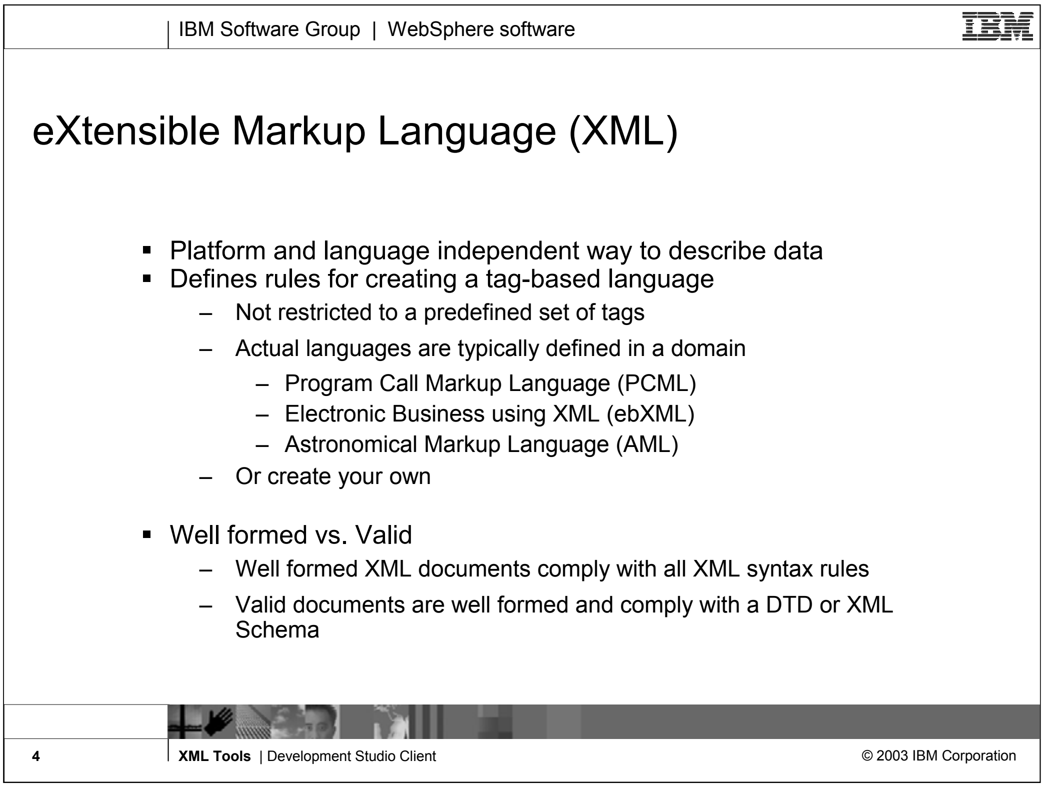
July 2003 | XML Tools

This presentation reviews XML and XML Development Tools included with WebSphere Development Studio Client for iSeries.

# Agenda

- XML Introduction
  - XML, DTD, XML Schema and XSLT
- Getting Started With the Tools
  - Creating an XML document
- XML Tools in Development Studio Client
  - The Editors - XML, DTD and XML Schema
  - XML to XML Mappings
  - XML JavaBeans
  - XML From SQL Query Wizard
  - Relational DB to XML Mapping
- Summary and References

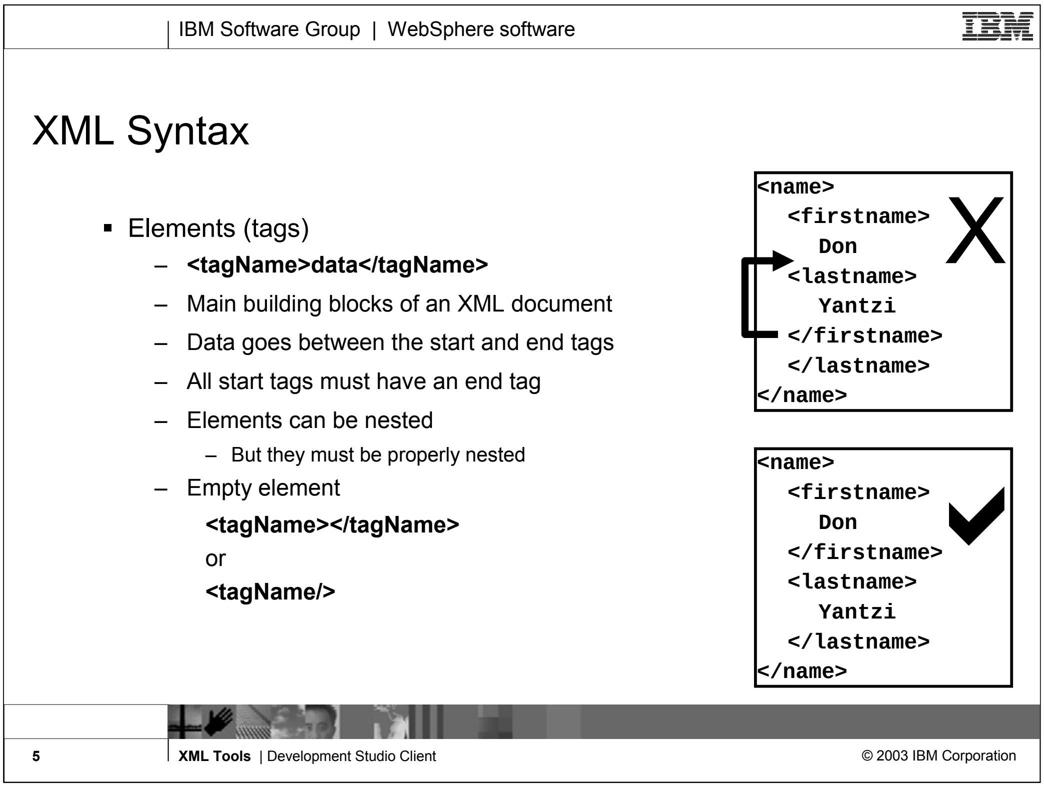
# XML Introduction



# eXtensible Markup Language (XML)

- Platform and language independent way to describe data
- Defines rules for creating a tag-based language
  - Not restricted to a predefined set of tags
  - Actual languages are typically defined in a domain
    - Program Call Markup Language (PCML)
    - Electronic Business using XML (ebXML)
    - Astronomical Markup Language (AML)
  - Or create your own
- Well formed vs. Valid
  - Well formed XML documents comply with all XML syntax rules
  - Valid documents are well formed and comply with a DTD or XML Schema

The XML specification outlines rules that an XML document must follow in order to be well formed. It does not specify any specific tag (element) names or attribute names. It specifies what a tag is and that it can have attributes and the syntax for specifying all of this.So, XML itself is not a language, but describes a class of languages. Each language will be domain specific (or even application specific) such as the examples given on this slide. The actaul tag and attribute names are specified in language specific DTD or XML Schema. If the tags, attributes and structure of an XML document comply with its associated DTD or Schema then the document is also valid.



# XML Syntax

- Elements (tags)
  - **<tagName>data</tagName>**
  - Main building blocks of an XML document
  - Data goes between the start and end tags
  - All start tags must have an end tag
  - Elements can be nested
    - But they must be properly nested
  - Empty element

    **<tagName></tagName>**

    or

    **<tagName/>**

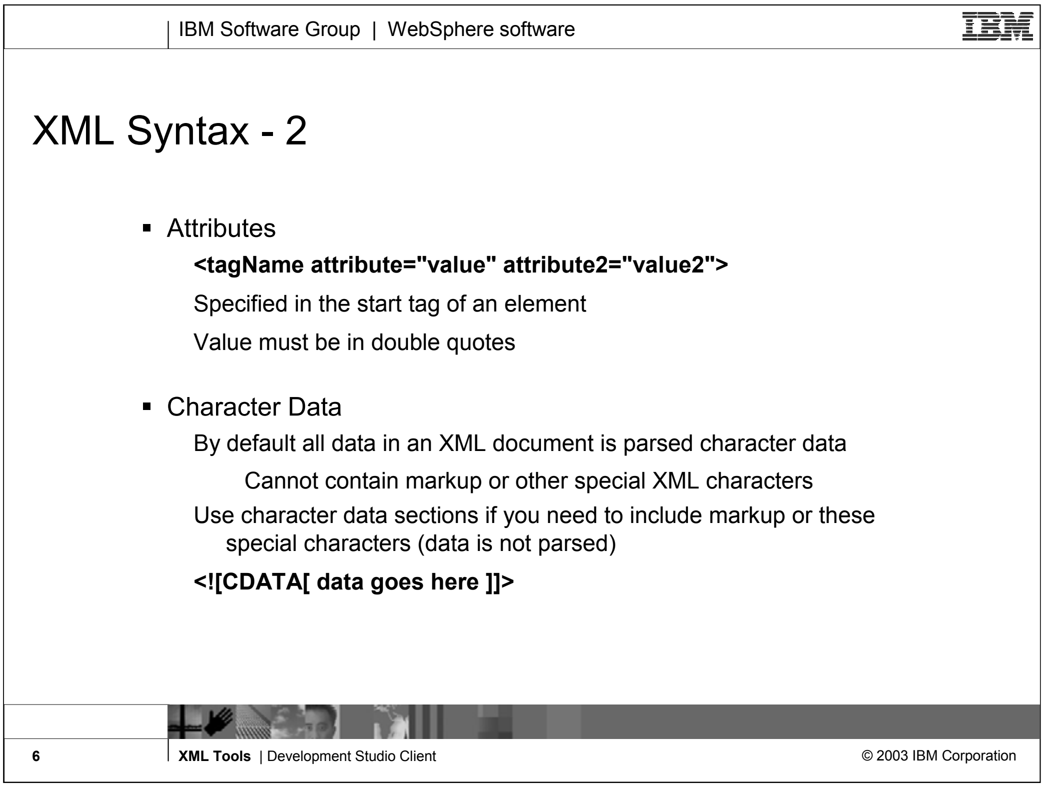
```
<name>
  <firstname>
    Don
  <lastname>
    Yantzi
  </firstname>
  </lastname>
</name>
```

X

```
<name>
  <firstname>
    Don
  </firstname>
  <lastname>
    Yantzi
  </lastname>
</name>
```

✓

Some of the basic rules of XML



# XML Syntax - 2

- Attributes

  **<tagName attribute="value" attribute2="value2">**

  Specified in the start tag of an element

  Value must be in double quotes

- Character Data

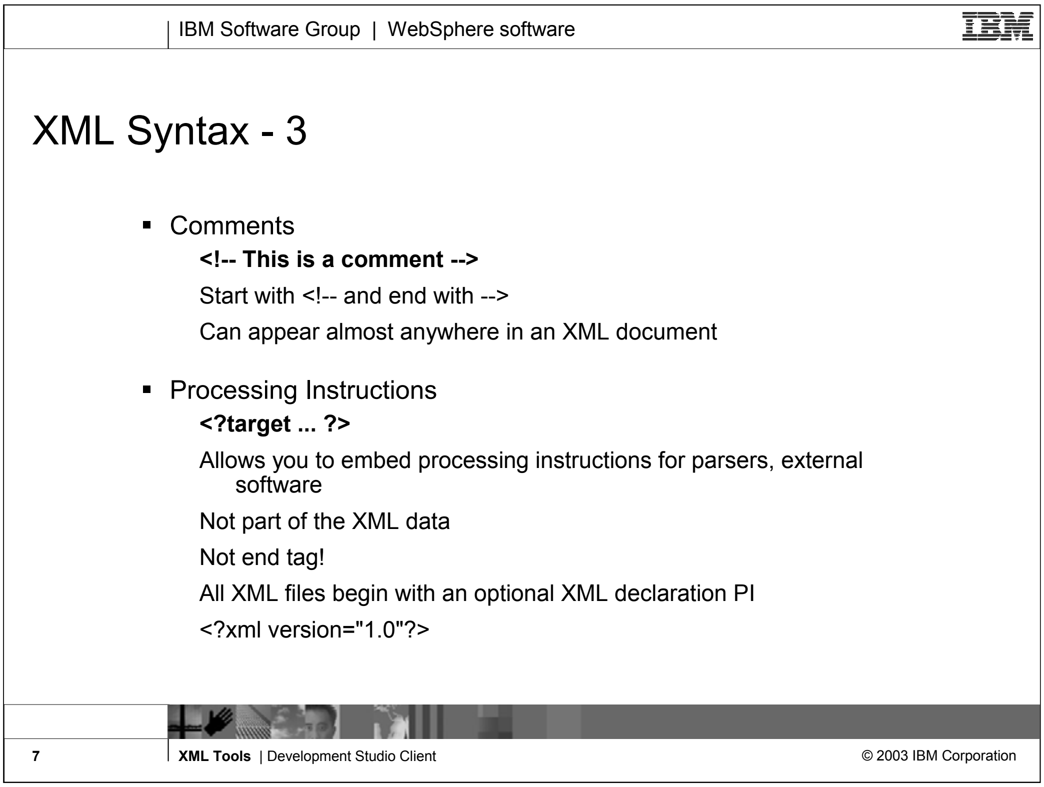
  By default all data in an XML document is parsed character data

  Cannot contain markup or other special XML characters

  Use character data sections if you need to include markup or these special characters (data is not parsed)

  **<![CDATA[ data goes here ]]>**

The syntax of attributes and comments. All data in XML is assumed to be parsed character data. This means that it cannot contain any markup or else the XML parser will try to intrepret the markup and likely cause the XML validation to fail. If you need to include markup or other special characters in your XML documents you can use a CDATA section which is not paresed by the XML parsers. It is important to know what the terms are, not so much the specific syntax because the tools can help with that!



# XML Syntax - 3

- Comments

  **<!-- This is a comment -->**

  Start with <!-- and end with -->

  Can appear almost anywhere in an XML document

- Processing Instructions

  **<?target ... ?>**

  Allows you to embed processing instructions for parsers, external software

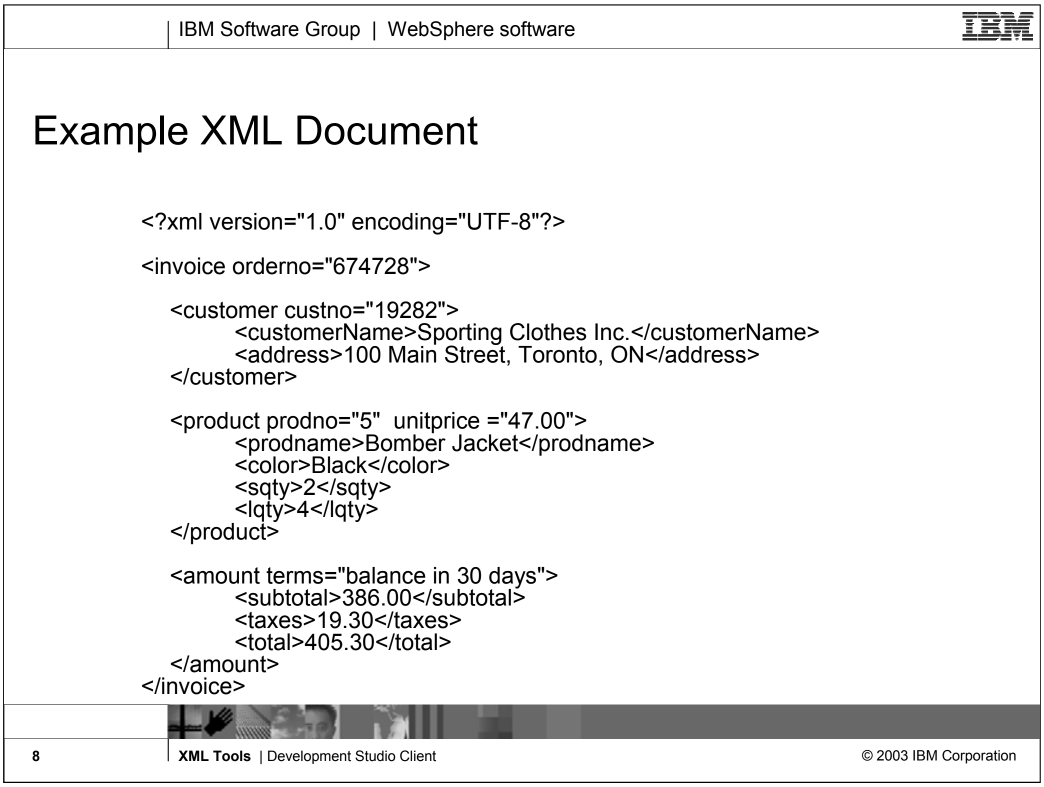
  Not part of the XML data

  Not end tag!

  All XML files begin with an optional XML declaration PI

  <?xml version="1.0"?>

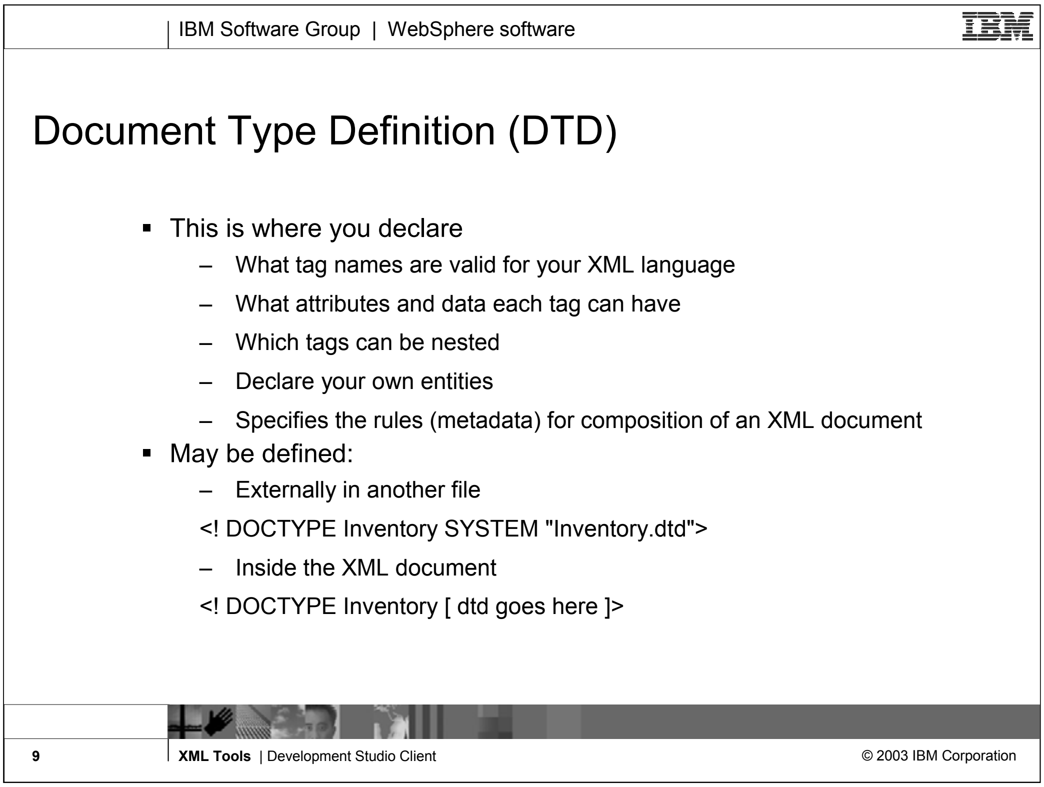
Some more XML language constructs...



# Example XML Document

```
<?xml version="1.0" encoding="UTF-8"?>

<invoice orderno="674728">

   <customer custno="19282">
         <customerName>Sporting Clothes Inc.</customerName>
         <address>100 Main Street, Toronto, ON</address>
   </customer>

   <product prodno="5"  unitprice ="47.00">
         <prodname>Bomber Jacket</prodname>
         <color>Black</color>
         <sqty>2</sqty>
         <lqty>4</lqty>
   </product>

   <amount terms="balance in 30 days">
         <subtotal>386.00</subtotal>
         <taxes>19.30</taxes>
         <total>405.30</total>
   </amount>
</invoice>
```

Here is an example XML document showing the XML declaration (very first line), some tags and attributes. One question that frequently comes up is when to use elements versus attributes.  It is really upto the designer of the XML language.  In the example shown the prodno attribute for product could just as well have been a prodno element within the product element.



# Document Type Definition (DTD)

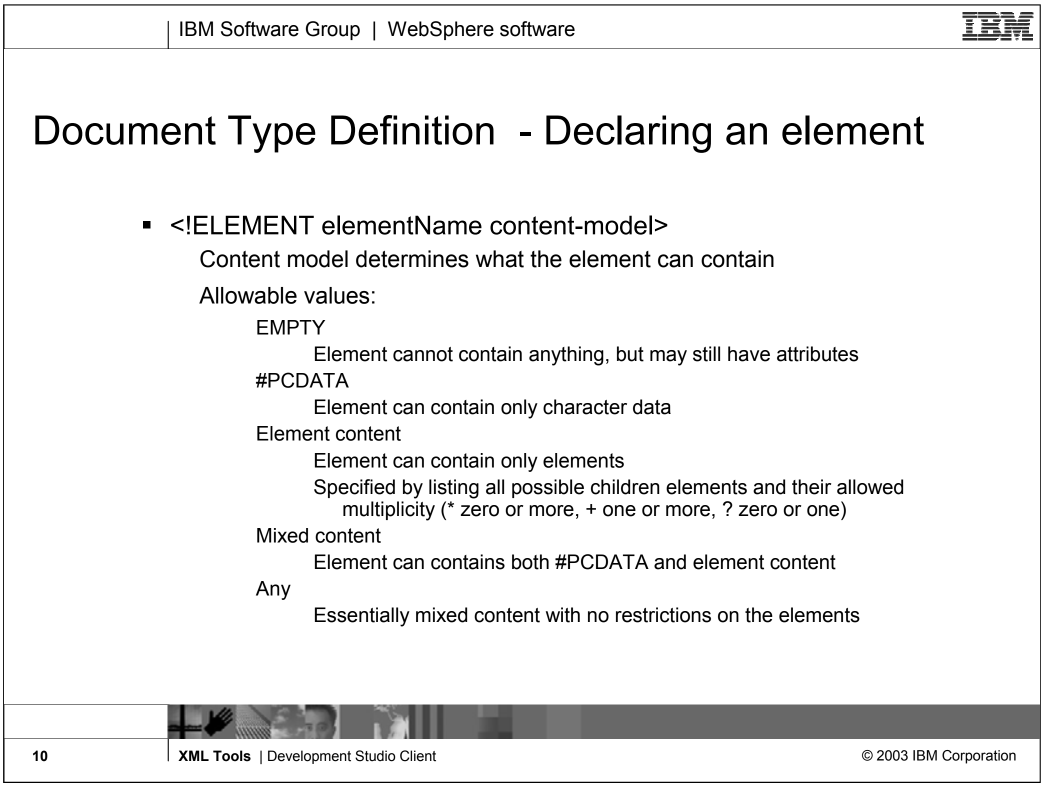
- This is where you declare
  - What tag names are valid for your XML language
  - What attributes and data each tag can have
  - Which tags can be nested
  - Declare your own entities
  - Specifies the rules (metadata) for composition of an XML document
- May be defined:
  - Externally in another file

  <! DOCTYPE Inventory SYSTEM "Inventory.dtd">

  - Inside the XML document

  <! DOCTYPE Inventory [ dtd goes here ]>

Document Type Definitions are used to specify a specific XML language. So for each of the examples shown on the first XML slide (PCML, ebXML, AML) will have a corresponding DTD which describes the language. So DTD's do not contain any data, they describe the tags and attributes that are then used to markup the actual data in an XML document. A DTD can be linked to an XML document either externally using the first DOCTYPE declaration shown or can be included within the XML document itself using the second declaration shown.



# Document Type Definition - Declaring an element

- <!ELEMENT elementName content-model>

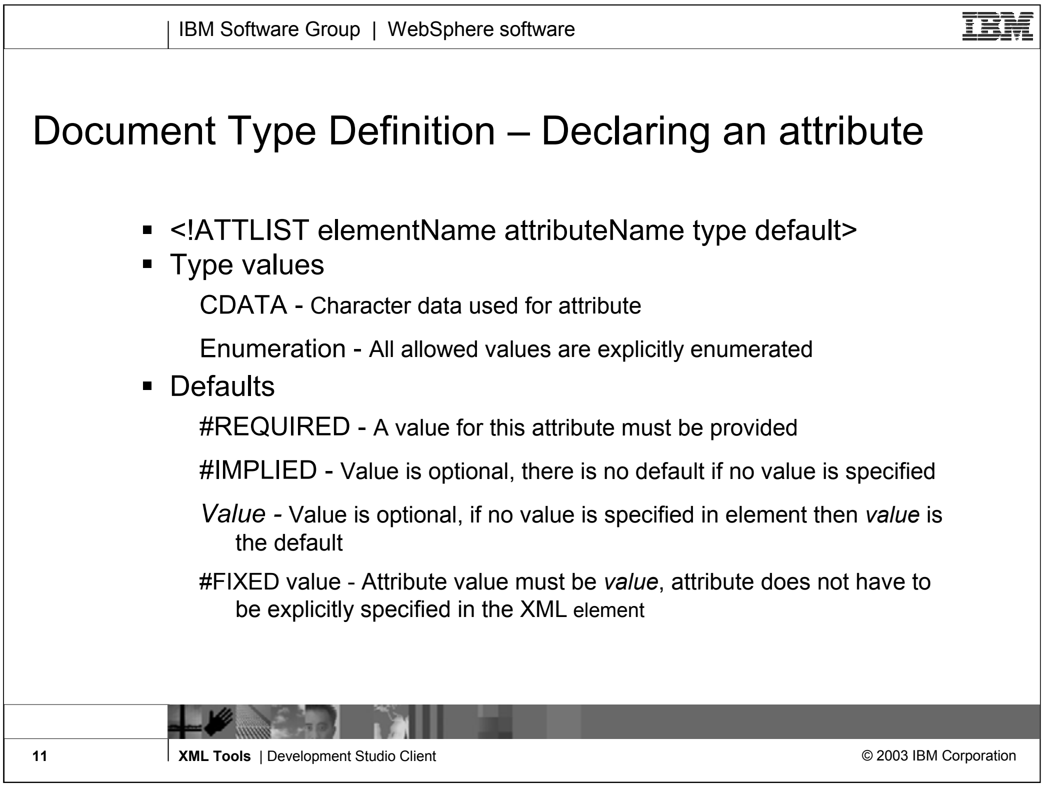
  Content model determines what the element can contain

  Allowable values:

  - EMPTY
    - Element cannot contain anything, but may still have attributes
  - #PCDATA
    - Element can contain only character data
  - Element content
    - Element can contain only elements
    - Specified by listing all possible children elements and their allowed multiplicity (* zero or more, + one or more, ? zero or one)
  - Mixed content
    - Element can contains both #PCDATA and element content
  - Any
    - Essentially mixed content with no restrictions on the elements

This slide shows the DTD syntax for declaring an element. The elementName is simply the name of the tag that will be used in the corresponding XML documents.

For each element you specify the what the element can contain (between the start and end tags). This is known as the content model as the possible values are listed on this slide.



## Document Type Definition – Declaring an attribute

- <!ATTLIST elementName attributeName type default>
- Type values

  CDATA - Character data used for attribute

  Enumeration - All allowed values are explicitly enumerated
- Defaults

  #REQUIRED - A value for this attribute must be provided

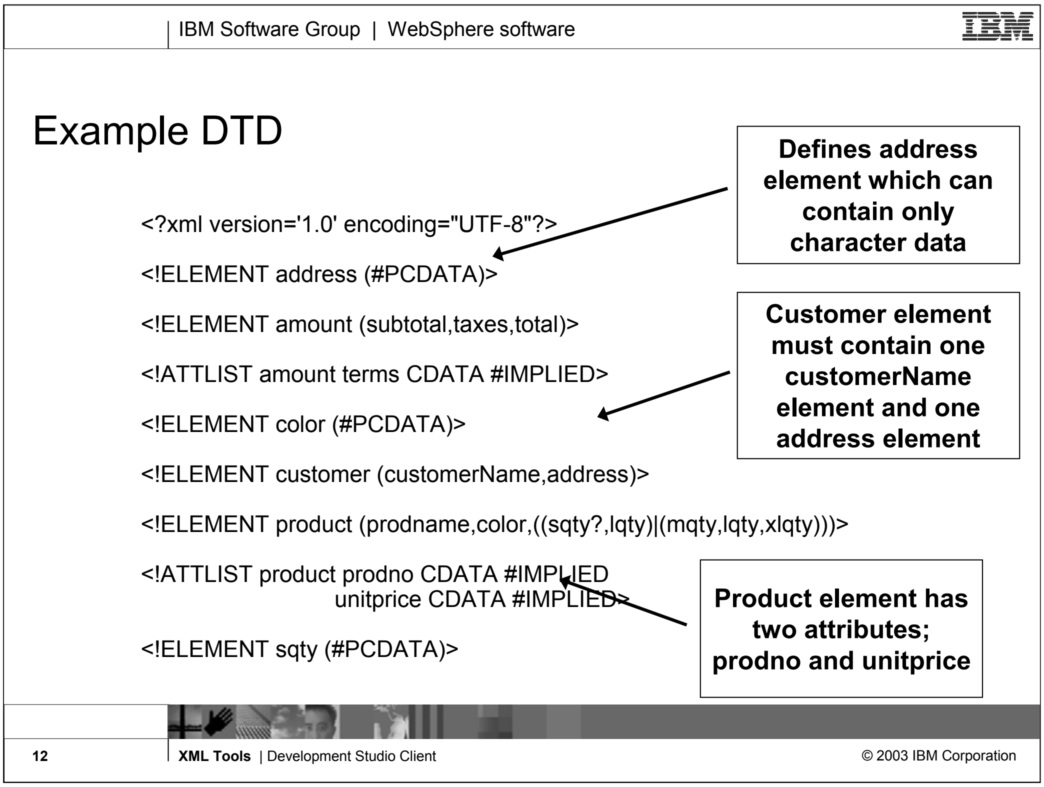
  #IMPLIED - Value is optional, there is no default if no value is specified

  *Value* - Value is optional, if no value is specified in element then *value* is the default

  #FIXED value - Attribute value must be *value*, attribute does not have to be explicitly specified in the XML element

This slide shows the syntax for declaring an attribute. The delcaration includes the element name the attribute is associate with and the attributes name.

You also specify a type for an attribute (character or an enumeration) as well as its default value. DTDs really only support character data types. This is one of the reasons that XML Schema is a better alternative.



## Example DTD

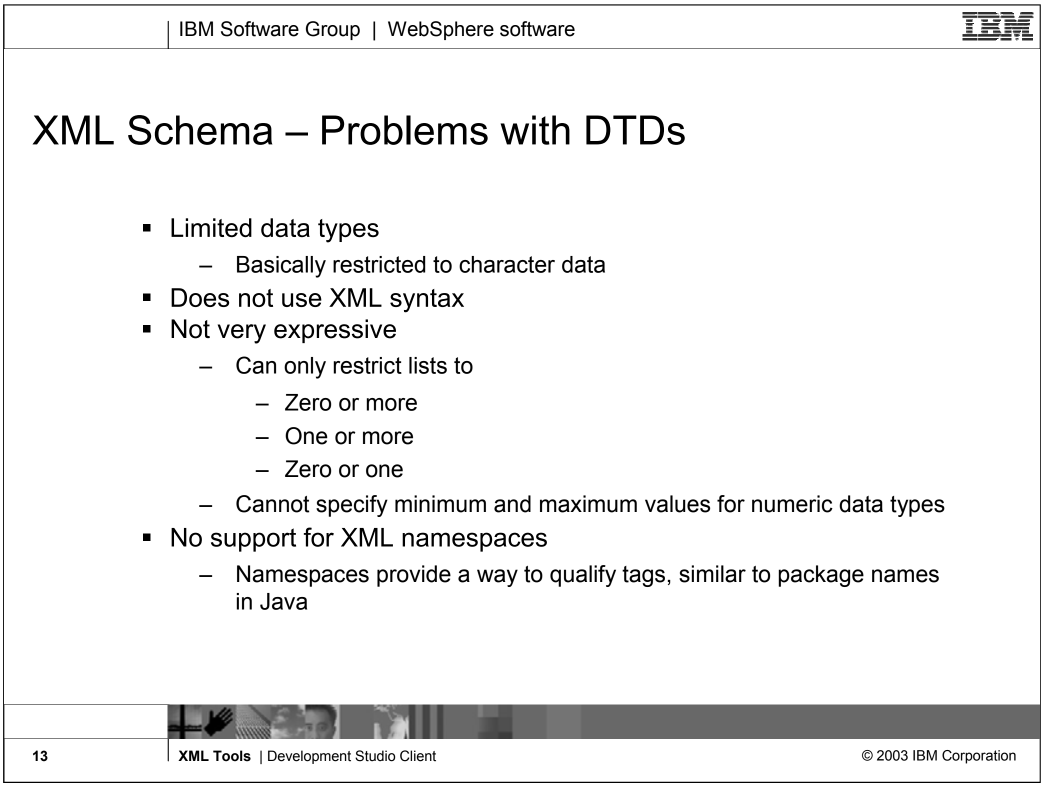
```
<?xml version='1.0' encoding="UTF-8"?>
<!ELEMENT address (#PCDATA)>
<!ELEMENT amount (subtotal,taxes,total)>
<!ATTLIST amount terms CDATA #IMPLIED>
<!ELEMENT color (#PCDATA)>
<!ELEMENT customer (customerName,address)>
<!ELEMENT product (prodname,color,((sqty?,lqty)|(mqty,lqty,xlqty)))>
<!ATTLIST product prodno CDATA #IMPLIED
                  unitprice CDATA #IMPLIED>
<!ELEMENT sqty (#PCDATA)>
```

**Defines address element which can contain only character data**

**Customer element must contain one customerName element and one address element**

**Product element has two attributes; prodno and unitprice**

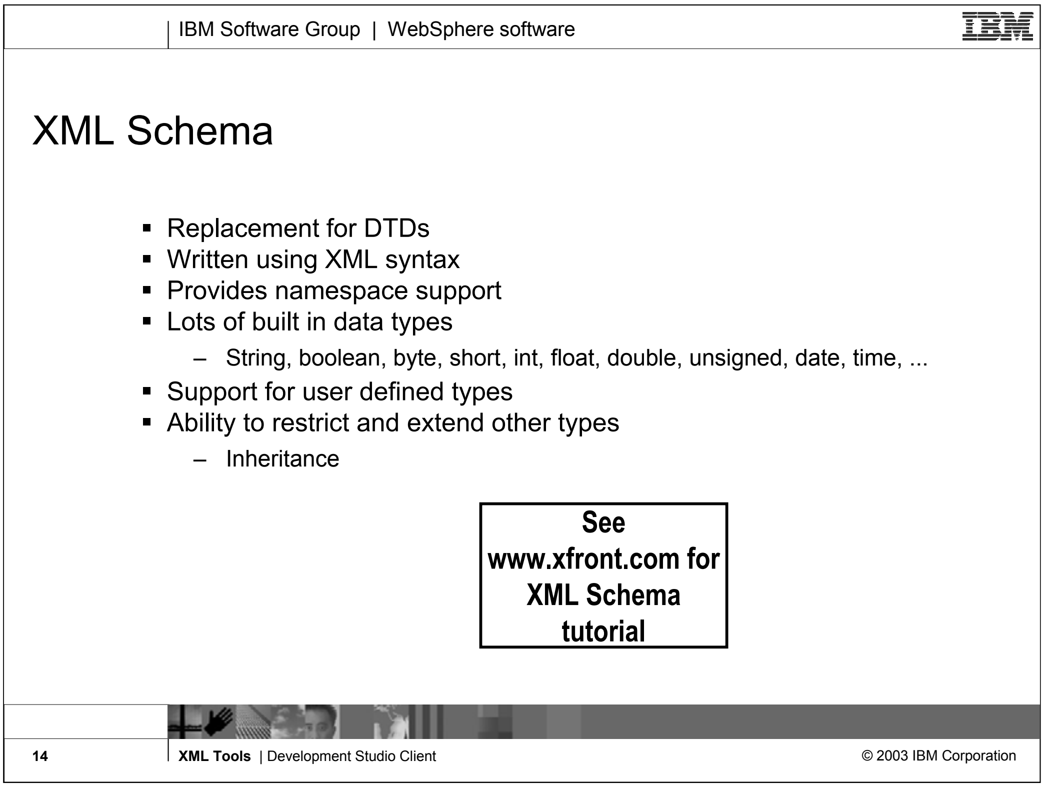
An example DTD. The thing to point out here is that DTDs do not use XLM syntax and therefore require a different set of skills (therefore increasing the learning curve.) The tooling can definitally help with this, but XML Schemas are a better alternative.



## XML Schema – Problems with DTDs

- Limited data types
  - Basically restricted to character data
- Does not use XML syntax
- Not very expressive
  - Can only restrict lists to
    - Zero or more
    - One or more
    - Zero or one
  - Cannot specify minimum and maximum values for numeric data types
- No support for XML namespaces
  - Namespaces provide a way to qualify tags, similar to package names in Java

Some problems with DTDs. Basically they only support character data type and are not able to express anything but very simple constraints.

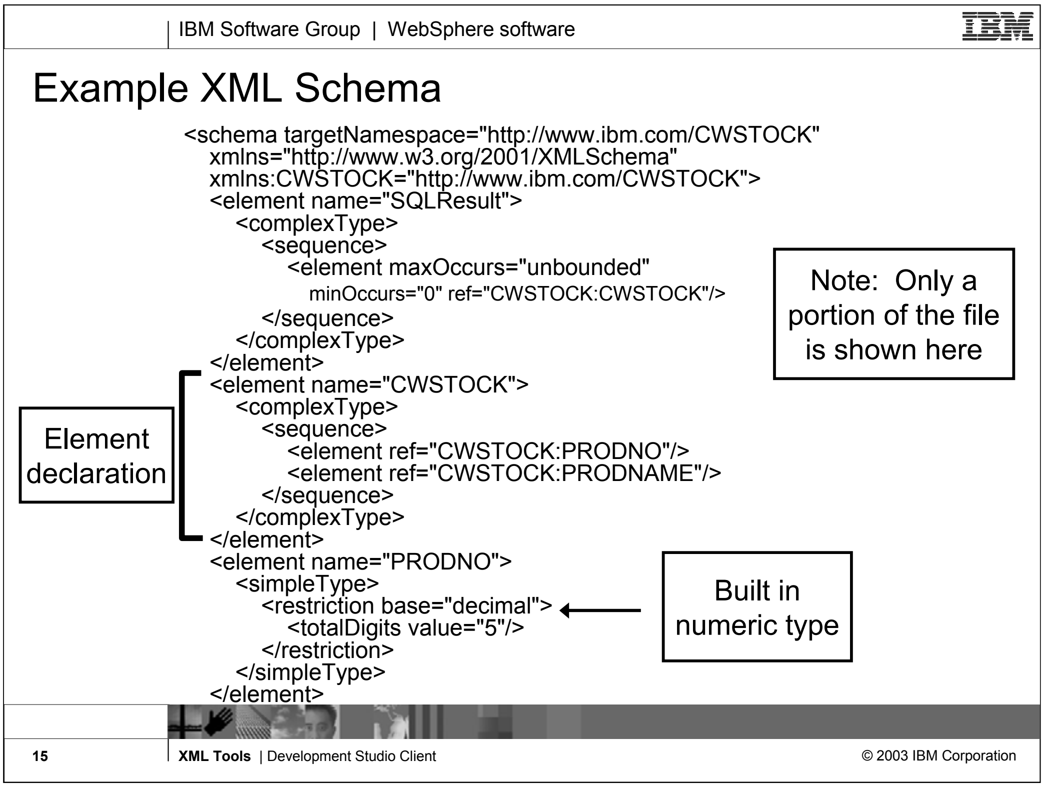


# XML Schema

- Replacement for DTDs
- Written using XML syntax
- Provides namespace support
- Lots of built in data types
  - String, boolean, byte, short, int, float, double, unsigned, date, time, ...
- Support for user defined types
- Ability to restrict and extend other types
  - Inheritance

**See www.xfront.com for XML Schema tutorial**

XML Schema came out a couple of years after DTDs and addresses most (if not all) of the shortcomings of DTDs.



# Example XML Schema

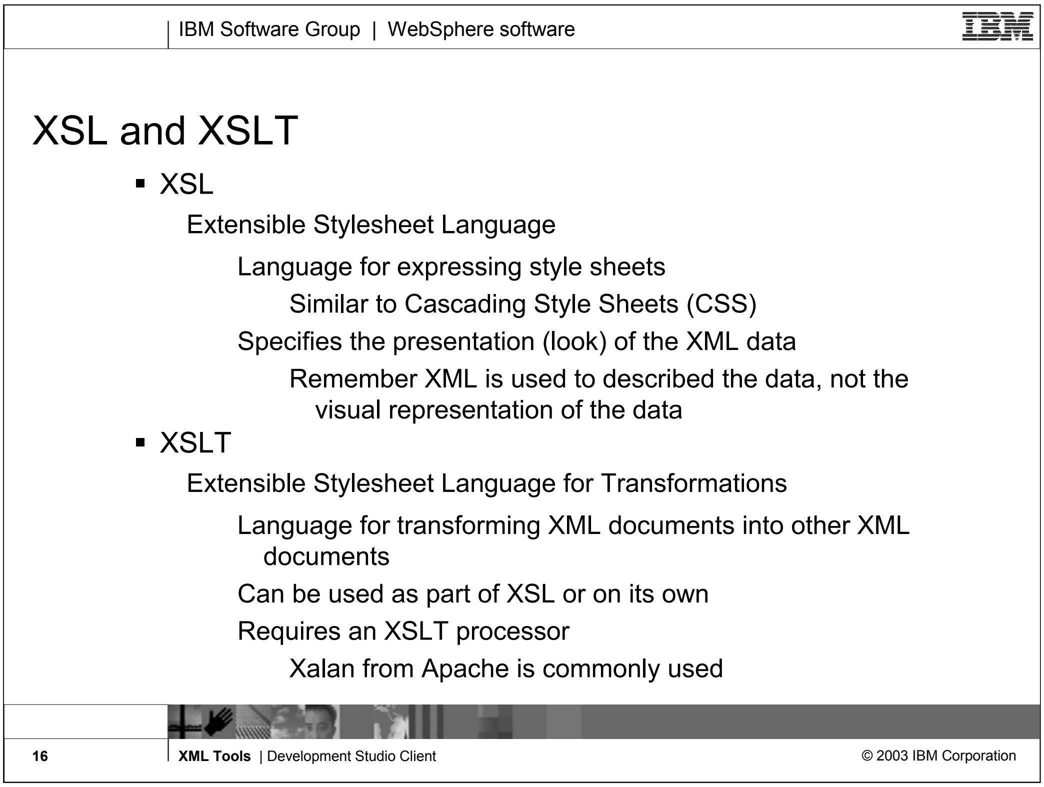
```
<schema targetNamespace="http://www.ibm.com/CWSTOCK"
   xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:CWSTOCK="http://www.ibm.com/CWSTOCK">
   <element name="SQLResult">
      <complexType>
         <sequence>
            <element maxOccurs="unbounded"
               minOccurs="0" ref="CWSTOCK:CWSTOCK"/>
         </sequence>
      </complexType>
   </element>
   <element name="CWSTOCK">
      <complexType>
         <sequence>
            <element ref="CWSTOCK:PRODNO"/>
            <element ref="CWSTOCK:PRODNAME"/>
         </sequence>
      </complexType>
   </element>
   <element name="PRODNO">
      <simpleType>
         <restriction base="decimal">
            <totalDigits value="5"/>
         </restriction>
      </simpleType>
   </element>
```

Note: Only a portion of the file is shown here

Element declaration

Built in numeric type

With the extra functionality provided by XML Schema comes additional complexity in the language. We won't get into the details of XML Schema here because of this complexity. Again this is where the tooling can really help out. The user only needs to understand the concepts and the tools will generate the XML code.



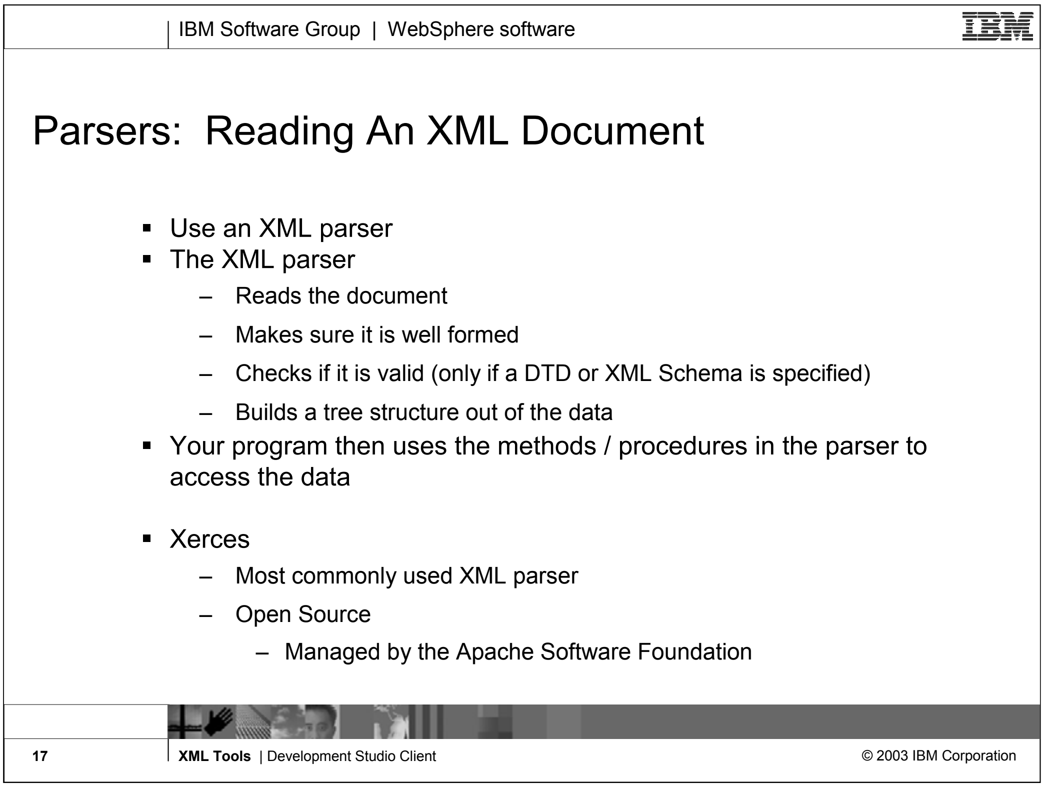
# XSL and XSLT

- XSL
  - Extensible Stylesheet Language
    - Language for expressing style sheets
      - Similar to Cascading Style Sheets (CSS)
    - Specifies the presentation (look) of the XML data
      - Remember XML is used to described the data, not the visual representation of the data
- XSLT
  - Extensible Stylesheet Language for Transformations
    - Language for transforming XML documents into other XML documents
    - Can be used as part of XSL or on its own
    - Requires an XSLT processor
      - Xalan from Apache is commonly used

XSL and XSLT provide the fourth key XML technology (the first three are the XML language itself, DTDs and XML Schema.) Remember that XML is used to describe the data not how the data should be rendered in a user interface. This is the job of XSL. It is similar to CSS and is used to specify the presentation of an XML document. Although part of XSL, XSLT can be used independently to perform any XML to XML transformation. This is very useful if two applications are communicating via XML but each uses a slightly different XML syntax. XSLT can be used to transform one syntax into another. For example:

•To change element or attribute names

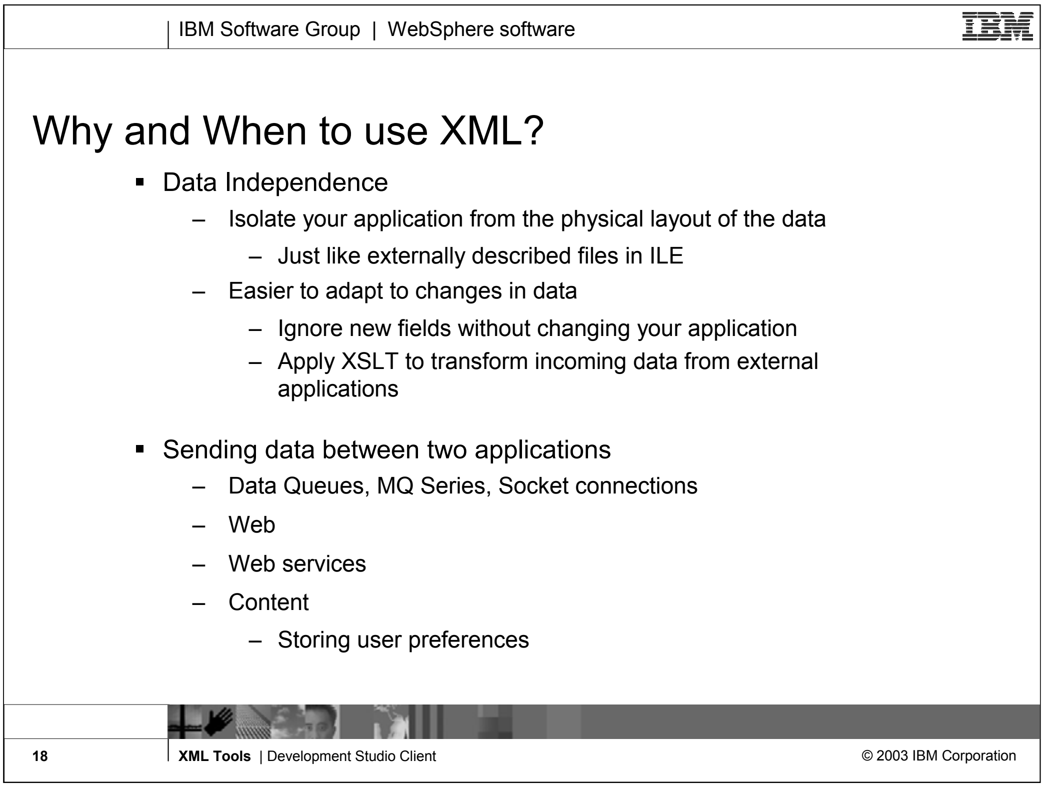
•To changes attributes into elements and vice versa



# Parsers: Reading An XML Document

- Use an XML parser
- The XML parser
  - Reads the document
  - Makes sure it is well formed
  - Checks if it is valid (only if a DTD or XML Schema is specified)
  - Builds a tree structure out of the data
- Your program then uses the methods / procedures in the parser to access the data
- Xerces
  - Most commonly used XML parser
  - Open Source
    - Managed by the Apache Software Foundation

When you use XML with your application you do not need to write the code to read the XML document and parse out the actual data from the elements and attributes. This is handled for you by an XML parser. The parser will check that an XML document is well formed and valid and will build a tree strucutre out of the data contained in the document. Your application then uses the parser APIs to retrieve the data.

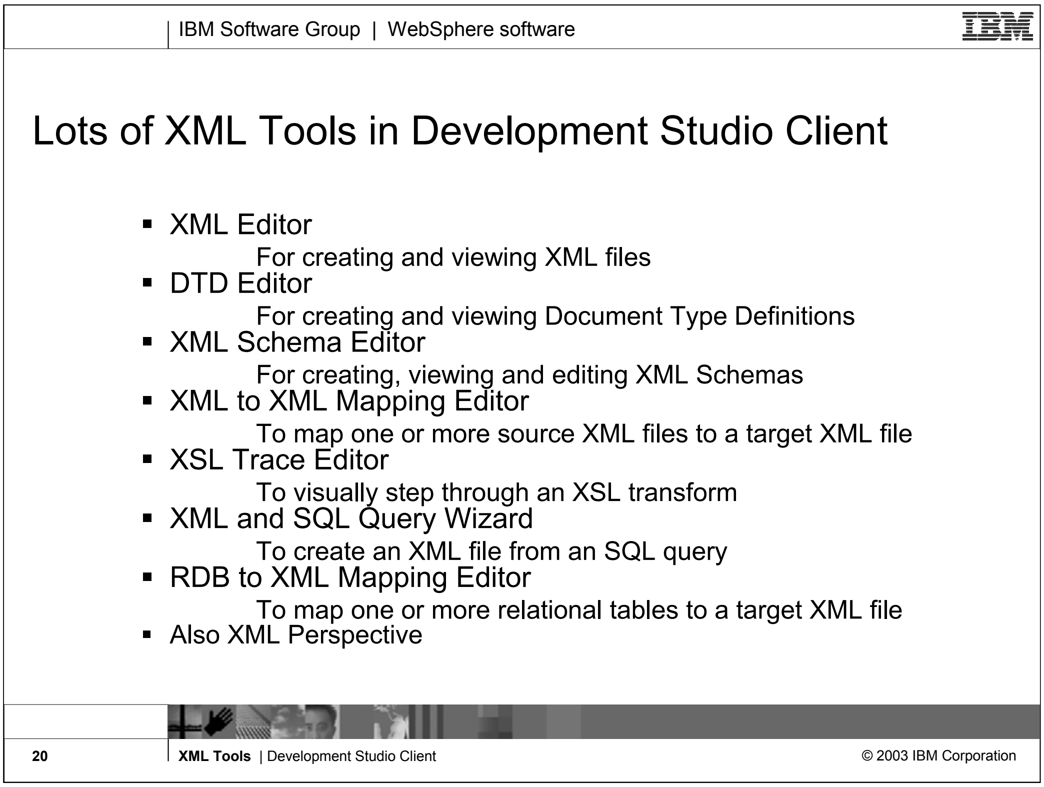


## Why and When to use XML?

- Data Independence
  - Isolate your application from the physical layout of the data
    - Just like externally described files in ILE
  - Easier to adapt to changes in data
    - Ignore new fields without changing your application
    - Apply XSLT to transform incoming data from external applications
- Sending data between two applications
  - Data Queues, MQ Series, Socket connections
  - Web
  - Web services
  - Content
    - Storing user preferences

Here are some justifications for using XML and some examples of where XML could be used.

# XML Development Tools



# Lots of XML Tools in Development Studio Client

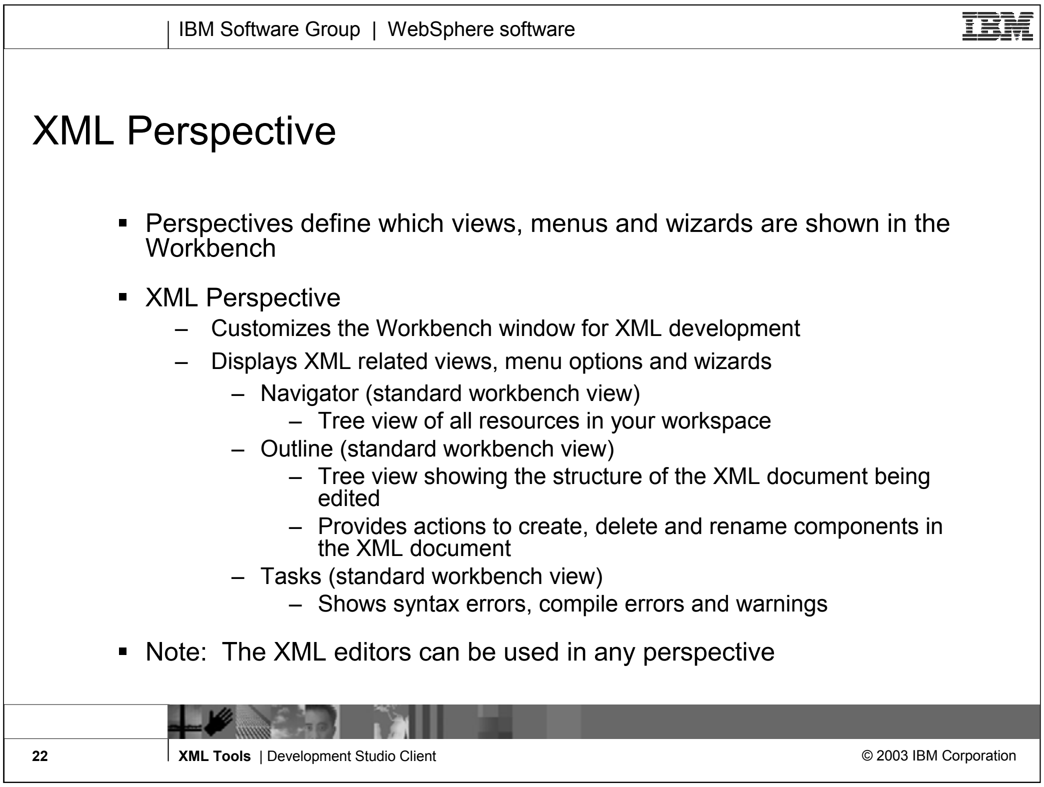
- XML Editor
  For creating and viewing XML files
- DTD Editor
  For creating and viewing Document Type Definitions
- XML Schema Editor
  For creating, viewing and editing XML Schemas
- XML to XML Mapping Editor
  To map one or more source XML files to a target XML file
- XSL Trace Editor
  To visually step through an XSL transform
- XML and SQL Query Wizard
  To create an XML file from an SQL query
- RDB to XML Mapping Editor
  To map one or more relational tables to a target XML file
- Also XML Perspective

Here is a list of all the XML tools contained in WebSphere Studio Site Developer Advanced. The rest of this presentation will look at each tool individually.

# XML Development Tools: Getting Started

WebSphere. software

e business software



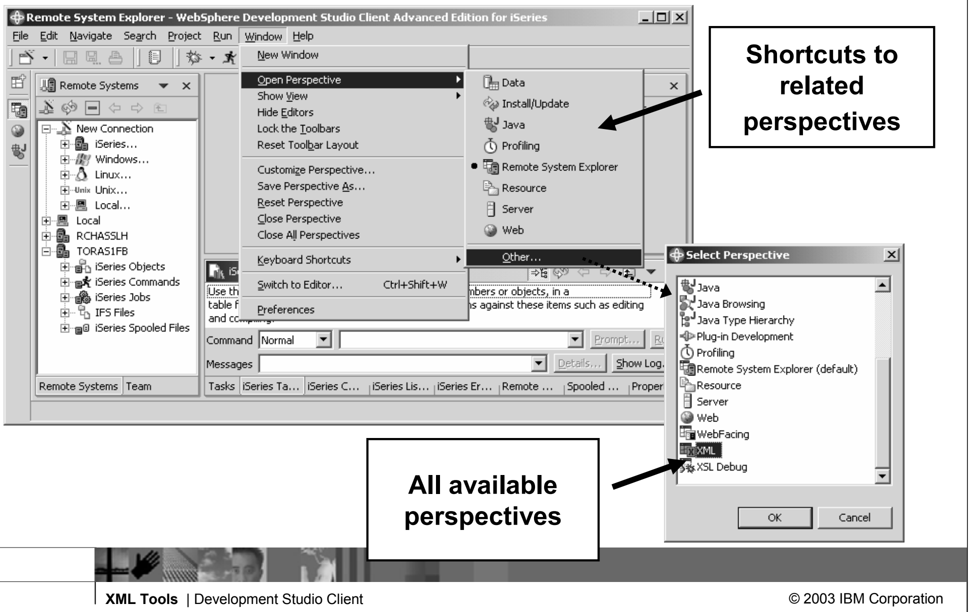
# XML Perspective

- Perspectives define which views, menus and wizards are shown in the Workbench
- XML Perspective
  - Customizes the Workbench window for XML development
  - Displays XML related views, menu options and wizards
    - Navigator (standard workbench view)
      - Tree view of all resources in your workspace
    - Outline (standard workbench view)
      - Tree view showing the structure of the XML document being edited
      - Provides actions to create, delete and rename components in the XML document
    - Tasks (standard workbench view)
      - Shows syntax errors, compile errors and warnings
- Note: The XML editors can be used in any perspective

Perspectives are used in the Workbench to provide a coherent selection and layout of views related to a specific type of development (Java, XML, WebFacing, ...)

By default the XML perspective shows the Navigator, outline and tasks views. The navigator view shows all resources in the user's workspace. The outline view is a standard Workbench view for showing the outline of the resource currently opened in the editor. It works for XML and other resource types like Java and SQL scripts. The tasks view is another standard workbench view and shows all errors, warnings and tasks in the workspace.

# XML Perspective

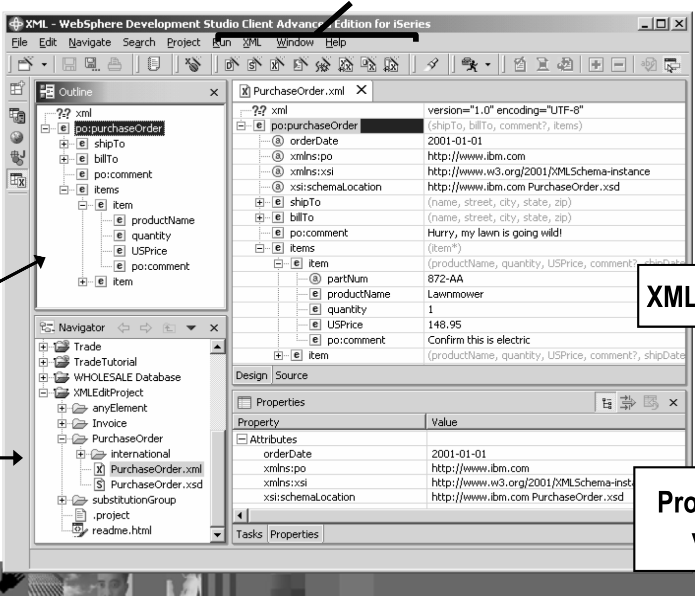


**Shortcuts to related perspectives**

**All available perspectives**

To open any perspective in the workbench select the Perspective -> Open -> Other menu option. This displays a list of all available perspectives. Select the desired perspective and click OK. Depending on the user preferences the perspective will either open in the same workbench window or a new workbench window.

# XML Perspective



| | |
|---|---|
| ?? xml | version="1.0" encoding="UTF-8" |
| po:purchaseOrder | (shipTo, billTo, comment?, items) |
| orderDate | 2001-01-01 |
| xmlns:po | http://www.ibm.com |
| xmlns:xsi | http://www.w3.org/2001/XMLSchema-instance |
| xsi:schemaLocation | http://www.ibm.com PurchaseOrder.xsd |
| shipTo | (name, street, city, state, zip) |
| billTo | (name, street, city, state, zip) |
| po:comment | Hurry, my lawn is going wild! |
| items | (item*) |
| item | (productName, quantity, USPrice, comment?, shipDate |
| partNum | 872-AA |
| productName | Lawnmower |
| quantity | 1 |
| USPrice | 148.95 |
| po:comment | Confirm this is electric |
| item | (productName, quantity, USPrice, comment?, shipDate |

| Property | Value |
|---|---|
| Attributes | |
| orderDate | 2001-01-01 |
| xmlns:po | http://www.ibm.com |
| xmlns:xsi | http://www.w3.org/2001/XMLSchema-inst |
| xsi:schemaLocation | http://www.ibm.com PurchaseOrder.xsd |

XML specific wizards

Open perspectives

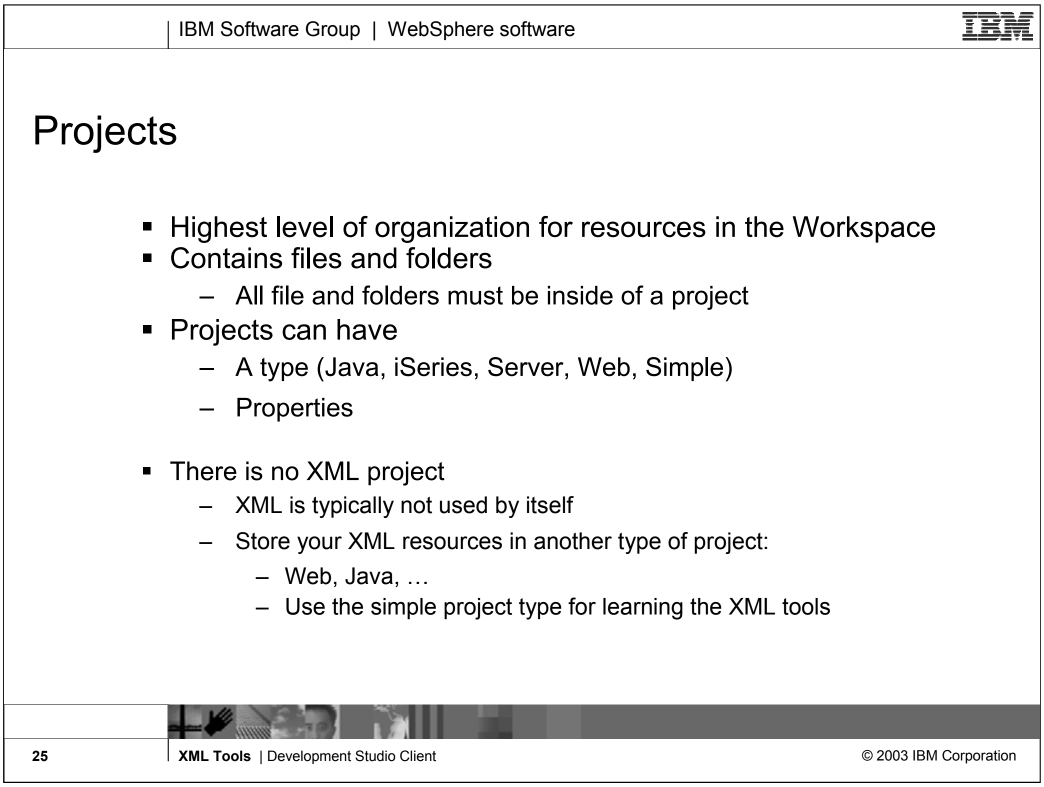
Outline view

Navigator view

XML editor

Properties view

This slide shows a screen shot of the XML perspective (default layout.) All open perspectives are shown in the left hand column of the workbench window. Simply click the perspectives icon to switch to the open perspective (this applies to all perspectives not just the XML perspective.) Each perspective typically contributes shortcuts to the workbench toolbar. The XML perspective adds icons for creating a new XML file, DTD, XML Schema and for launching the other XML wizards.

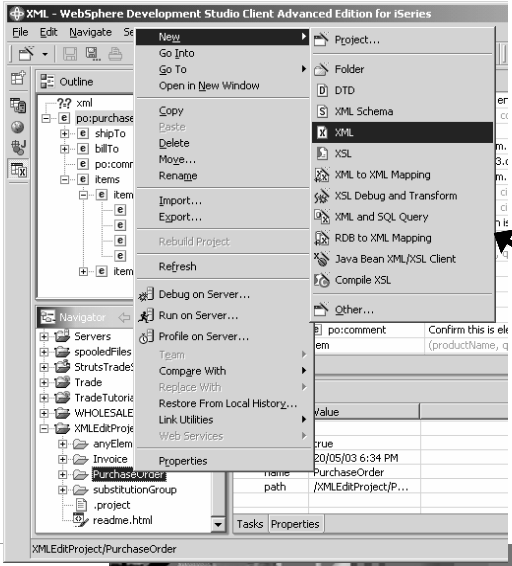


## Projects

- Highest level of organization for resources in the Workspace
- Contains files and folders
  - All file and folders must be inside of a project
- Projects can have
  - A type (Java, iSeries, Server, Web, Simple)
  - Properties
- There is no XML project
  - XML is typically not used by itself
  - Store your XML resources in another type of project:
    - Web, Java, …
    - Use the simple project type for learning the XML tools

Projects provide the highest level of grouping resources (source files, graphics, executables, ...) in the workspace. Projects always have a type such as Java, WebFacing, Web. Those projects that are not linked to a specific type of development should use the "Simple" type. By default projects map to a subdirectory of the workspace directory on the local file system with the same name as the project. However, there is no XML project because XML resources are usually used in the context of some bigger application should as a Java or Web application.

# Creating an XML File (and Other XML Wizards)



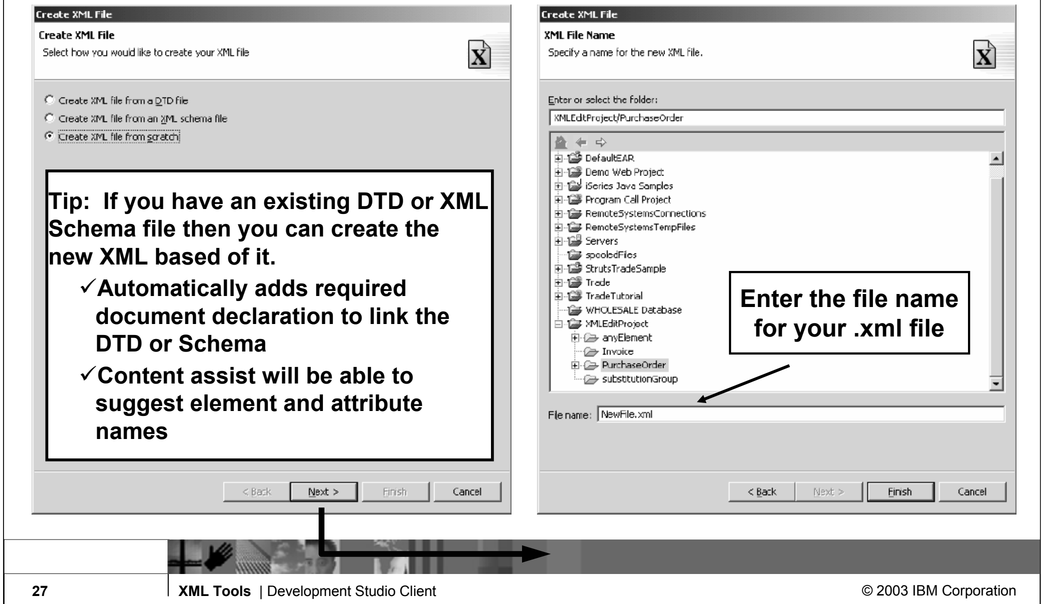
1. **Select the project where you want to create the XML file**
2. **Right click and select New -> XML File**

- **Note: These menu items only appear when you are in the XML perspective.**

- **If you are in another perspective use the File -> New -> Other menu option.**

The first thing we are going to look at is generating an XML file. Typically XML files are generated by applications not manually, however if you are creating a DTD or XML Schema it may be easier to first create a sample XML document the way it should look and then generate the DTD or Schema from the XML document. To create a new XML document first select the project (or folder) where the document will be created, right click and select new -> XML File

# Create XML File Wizard



Create XML File

Select how you would like to create your XML file

○ Create XML file from a DTD file
○ Create XML file from an XML schema file
● Create XML file from scratch

Tip: If you have an existing DTD or XML Schema file then you can create the new XML based of it.

- ✓Automatically adds required document declaration to link the DTD or Schema
- ✓Content assist will be able to suggest element and attribute names

XML File Name

Specify a name for the new XML file.

Enter or select the folder: XMLEditProject/PurchaseOrder

File name: NewFile.xml

Enter the file name for your .xml file

The first page of the new XML File wizard asks if you are creating this file from a DTD, Schema or from scratch. If you have an existing DTD or Schem it is a good idea to specify that here. This will:

•Allow the wizard to generate the XML code to link the XML file to the DTD / Schema

•Allow the content assist in the XML editor to provide you will a better list of options

The second page of the wizard just asks for the XML file name. The target project /folder is already selected based on what was selected when the wizard was launched.

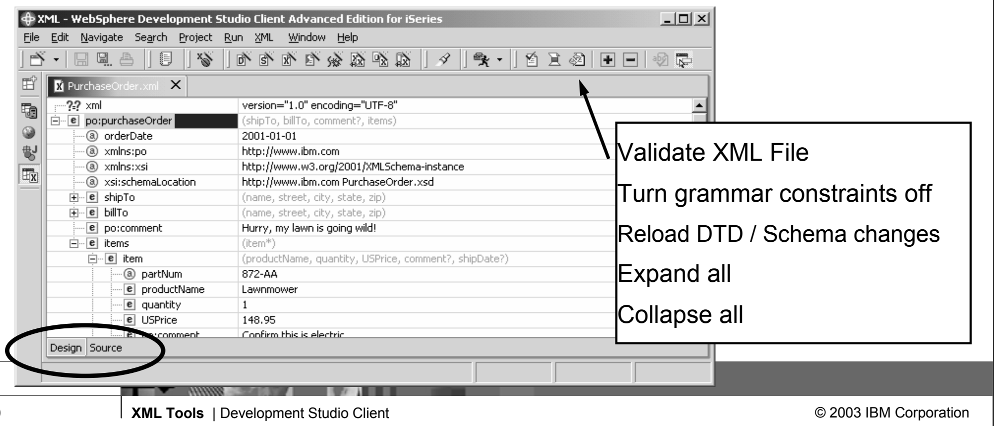
# XML Development Tools: XML Editor

WebSphere. software

e business software

# XML Editor

- Edit and validate XML files
- Import existing XML files for structured viewing
- Associate XML files with DTDs or XML schemas
- Two views - the Design view and the Source view
  - –Views are kept synchronized – easily switch between them while editing!


XML - WebSphere Development Studio Client Advanced Edition for iSeries
File Edit Navigate Search Project Run XML Window Help
PurchaseOrder.xml
Design
Source

| | |
|---|---|
| ?? xml | version="1.0" encoding="UTF-8" |
| po:purchaseOrder | (shipTo, billTo, comment?, items) |
| orderDate | 2001-01-01 |
| xmlns:po | http://www.ibm.com |
| xmlns:xsi | http://www.w3.org/2001/XMLSchema-instance |
| xsi:schemaLocation | http://www.ibm.com PurchaseOrder.xsd |
| shipTo | (name, street, city, state, zip) |
| billTo | (name, street, city, state, zip) |
| po:comment | Hurry, my lawn is going wild! |
| items | (item*) |
| item | (productName, quantity, USPrice, comment?, shipDate?) |
| partNum | 872-AA |
| productName | Lawnmower |
| quantity | 1 |
| USPrice | 148.95 |
| po:comment | Confirm this is electric |

Validate XML File

Turn grammar constraints off

Reload DTD / Schema changes

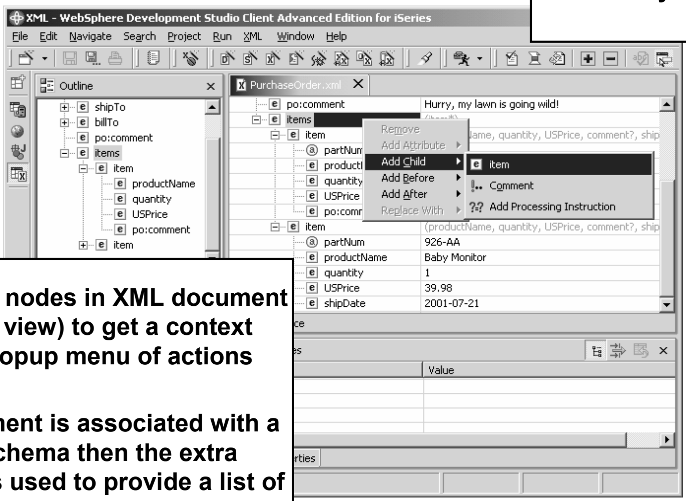
Expand all

Collapse all

The XML Editor has many uses. The first thing to note about the XML editor is that is has two different views; design and source. These views are kept synchronized so you can switch between them while editing without having to save the file first. The XML editor adds some actions to the workbench toolbar. If performance becomes an issue when editing XML documents you can turn off the grammar contstraint checking. This will prevent the editor from performing live grammar checking. To check grammar you can then either save the file or use the validate XML file action.

# XML Editor: Design View

Select a cell in the table to directly edit values



| | |
|---|---|
| po:comment | Hurry, my lawn is going wild! |
| items | (item*) |
| item | ...ame, quantity, USPrice, comment?, ship |
| partNum | |
| productName | |
| quantity | |
| USPrice | |
| po:comment | |
| item | (productName, quantity, USPrice, comment?, ship |
| partNum | 926-AA |
| productName | Baby Monitor |
| quantity | 1 |
| USPrice | 39.98 |
| shipDate | 2001-07-21 |

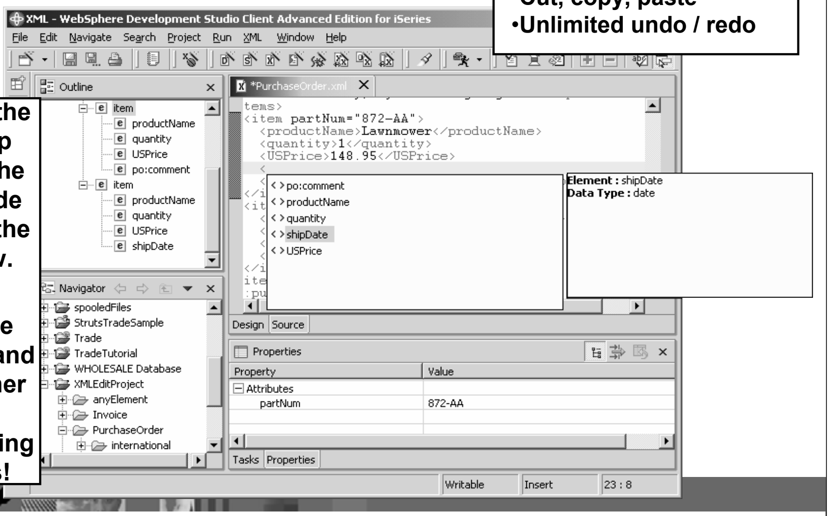
Right click on nodes in XML document (or Outline view) to get a context sensitive popup menu of actions

If XML document is associated with a DTD or Schema then the extra information is used to provide a list of allowed elements and attributes

The XML editor's design view gives you a tabular tree view of the XML document. This view allows you to edit an XML document without having to worry about the underlying XML syntax. Each element and attribute is represented by a single row in the table. You can expand and collapse elements to see their contents. To change the value of an element / attribute click the cell in the right and column of the table and simply enter the new value. To create / delete elements and attributes click on the target node and select the appropriate action from the popup menu.

# XML Editor: Source View

- •Token highlighting
- •Content assist
- •Automatic formatting
- •Spell checking
- •Cut, copy, paste
- •Unlimited undo / redo



```
tems>
<item partNum="872-AA">
  <productName>Lawnmower</productName>
  <quantity>1</quantity>
  <USPrice>148.95</USPrice>
  <
```

**Element :** shipDate
**Data Type :** date

| Property | Value |
|---|---|
| ⊟ Attributes | |
| partNum | 872-AA |

**You can get the same popup menu from the previous slide on nodes in the outline view.**

**Tip: Use the outline view and editor together for some powerful editing capabilities!**

The source view of the XML editor lets you directly edit the XML source code. Just like the Java editor in WDSc, the XML editor provides content assist to help you with element and attribute names. Use the popup menus in the content view to quickly insert / delete elements and attributes (just like the popup menus in the design view.)

# XML Development Tools: DTD and XML Schema Editors

WebSphere. software

e business software

# DTD and XML Schema Editors

- Provide many of the same capabilities as the XML editor, they all have
  - Design and source views
  - Context sensitive popup menus for adding new items
  - Display a high level structure of the file in the Outline view
  - Use the properties view for modifying attributes of the selected element
- DTDs and XML Schemas can be created from
  - Scratch
  - Existing XML documents
    - Tool creates the DTD or Schema by examining the structure of the existing XML document

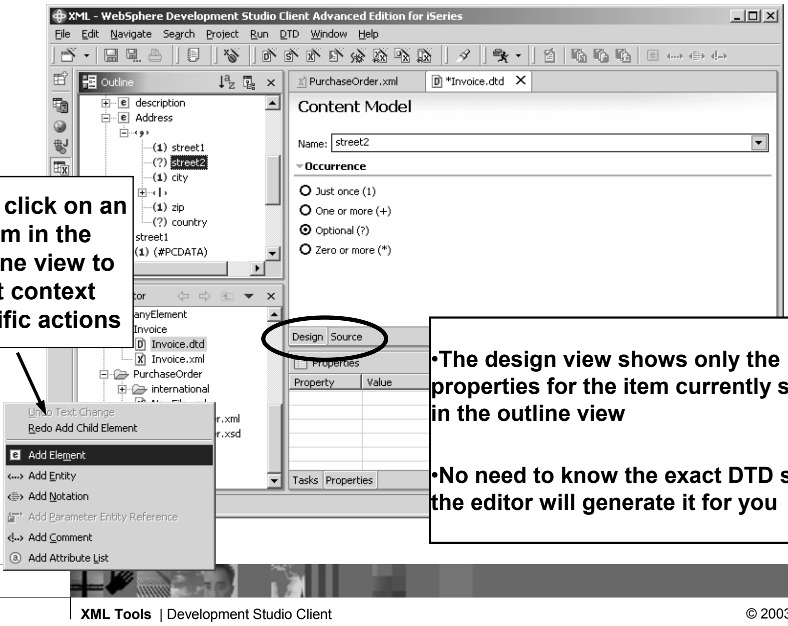
# DTD Editor

- Create new DTDs from scratch or from existing XML documents
- Validate DTDs
- Create, delete and edit DTDs
  - elements, attributes, entities, notations and comments
- Import existing DTDs for structured viewing.
- Generate:
  - XML schema
    - Migrate to using XML Schema from using DTDs
  - Java Beans to read and write XML documents for this DTD
  - HTML form
  - XML file

# XML Schema Editor

- Create, edit and delete XML schema components such as:
  - Complex and simple types, elements, attributes, attribute groups, and groups
- Validate XML schemas
- Import existing XML schemas for structured viewing
- Generate:
  - Document Type Definition (DTD)
  - Java Beans for reading and writing XML documents for the Schema
  - XML files
  - Relational table files
    - Generates DDL

# DTD Editor - Design View



Content Model

Name: street2

Occurrence

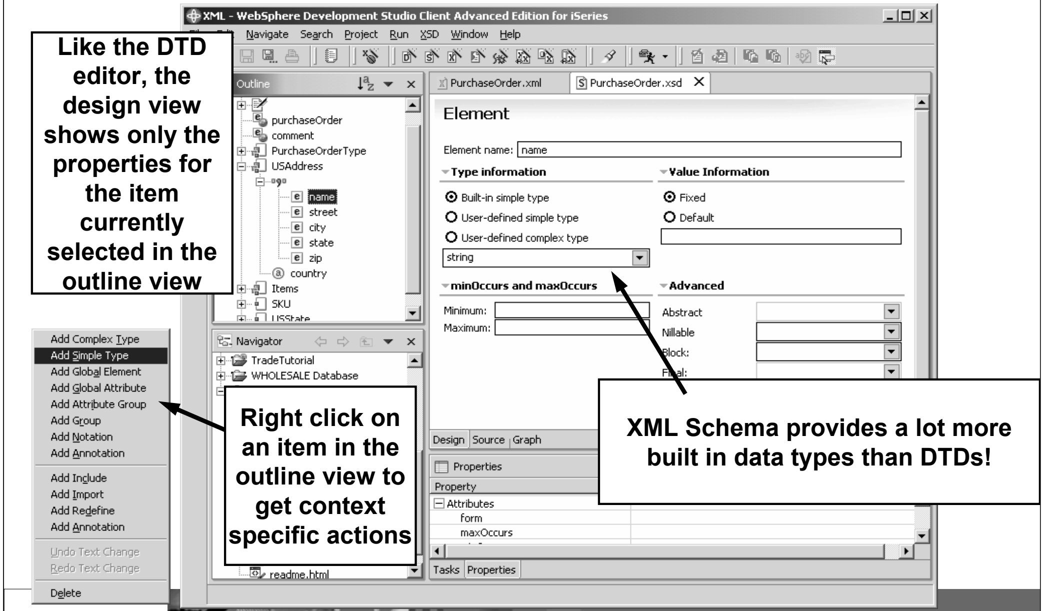
○ Just once (1)
○ One or more (+)
● Optional (?)
○ Zero or more (*)

**Right click on an item in the outline view to get context specific actions**

**•The design view shows only the properties for the item currently selected in the outline view**

**•No need to know the exact DTD syntax, the editor will generate it for you**

# XML Schema Editor: Design View



Element name: name

● Built-in simple type

string

● Fixed

**Like the DTD editor, the design view shows only the properties for the item currently selected in the outline view**

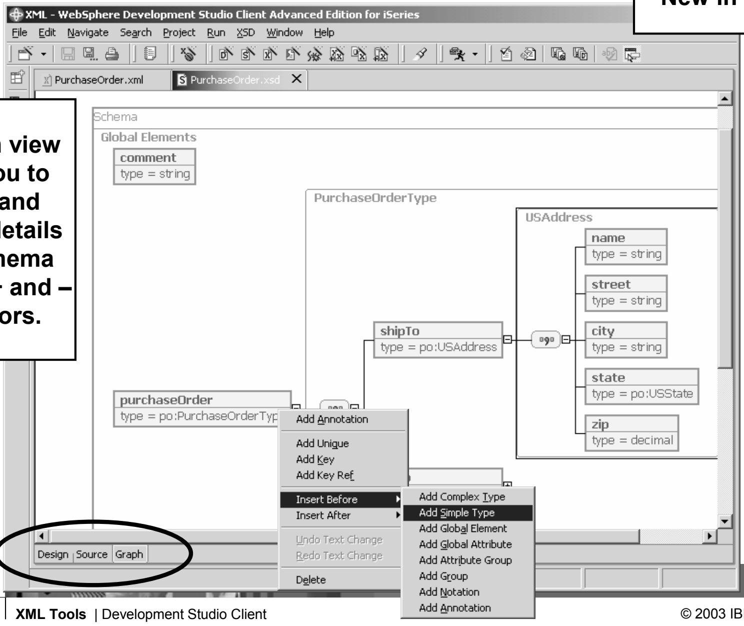
**Right click on an item in the outline view to get context specific actions**

**XML Schema provides a lot more built in data types than DTDs!**

Here we see the XML Schema editor. The outline of the schema is shown on the left, and the popup menu contains actions for editing the selected nodes. The right is the Schema editor that allows editing of the selected schema node, either in design mode (shown) or source mode (shown in next slide).

# XML Schema Editor:  Graph View

**New in V5.0**



**The graph view allows you to expand and collapse details of the schema using the + and – connectors.**

# XML Development Tools: XML to XML Mappings

WebSphere. software

e business software

# XML to XML Mapping Editor

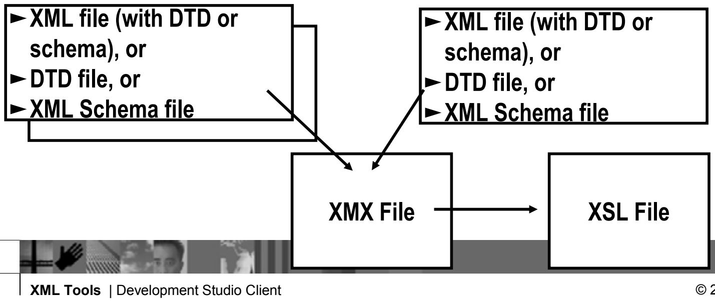
- Input files

  One or more *source* DTDs or XML Schemas

  A *target* DTD or XML Schema
- Input information (using the XMI to XML mapping editor)

  What ***source*** elements map to what ***target*** elements
- Output

  An **XSLT** file that captures the mapping



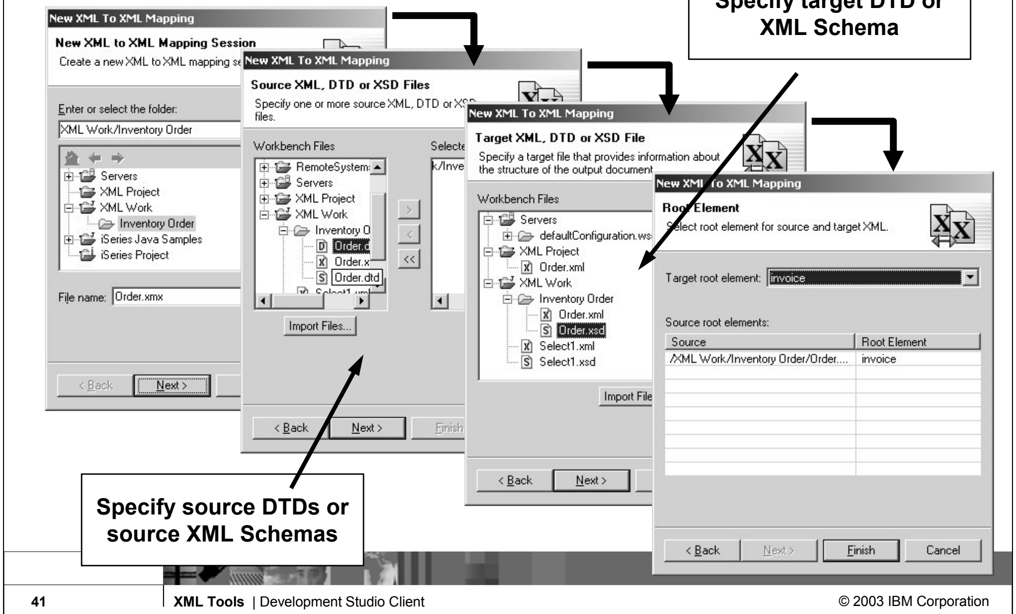
►XML file (with DTD or schema), or
►DTD file, or
►XML Schema file

►XML file (with DTD or schema), or
►DTD file, or
►XML Schema file

XMX File

XSL File

# XML To XML Mapping Wizard



New XML To XML Mapping

New XML to XML Mapping Session

Create a new XML to XML mapping se

Enter or select the folder: XML Work/Inventory Order

File name: Order.xmx

New XML To XML Mapping

Source XML, DTD or XSD Files

Specify one or more source XML, DTD or XSD files.

New XML To XML Mapping

Target XML, DTD or XSD File

Specify a target file that provides information about the structure of the output document

New XML To XML Mapping

Root Element

Select root element for source and target XML.

Target root element: invoice

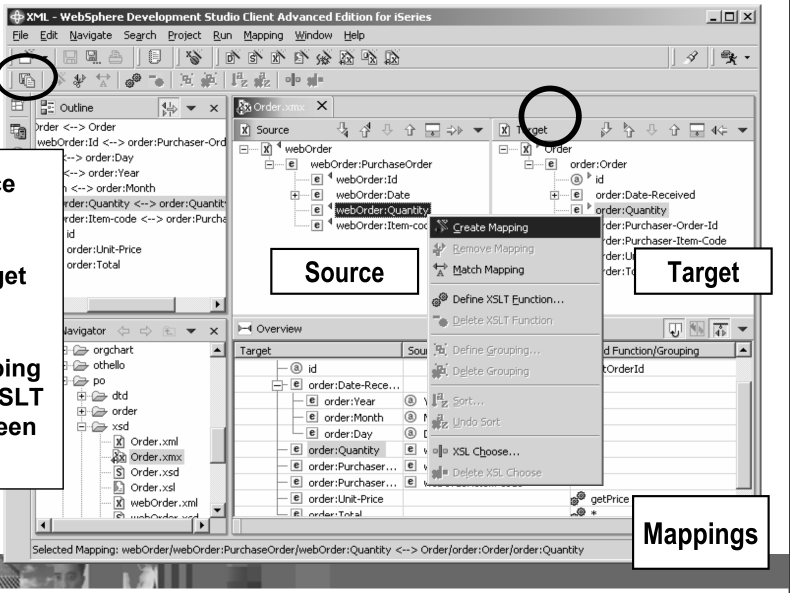
Source root elements:

| Source | Root Element |
|---|---|
| /XML Work/Inventory Order/Order.... | invoice |

Specify target DTD or XML Schema

Specify source DTDs or source XML Schemas

# XML-to-XML Editor



Selected Mapping: webOrder/webOrder:PurchaseOrder/webOrder:Quantity <--> Order/order:Order/order:Quantity

Generate XSLT script

1. **Select a source element or attribute**
2. **Select the target element or attribute**
3. **Create a mapping or define an XSLT function between the two.**

**Source**

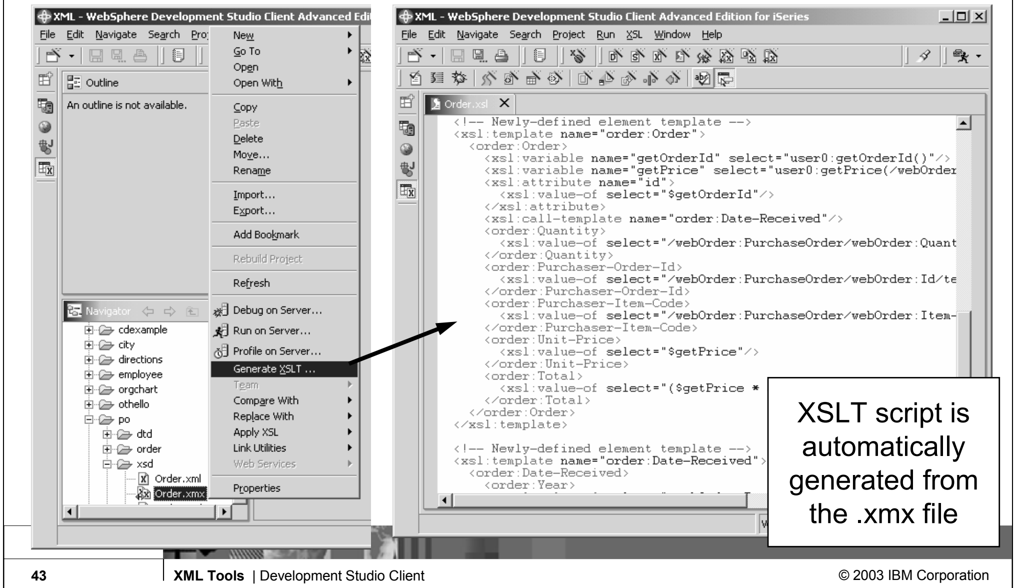
**Target**

**Mappings**

Here we see the XML-to-XML mapping editor. In the upper right we see an outline of the mappings. In the middle we see the input xml on the left, and the output xml on the right. To map two tags or attributes, select one on the left, and one on the right, and select the map button from the toolbar in the view at the bottom.

That bottom view shows the mappings so far. Once the nodes have been mapped, right click and select "Generate XSLT" to generate the XSLT that captures the mapping data.

# Generating XSLT From XMX File



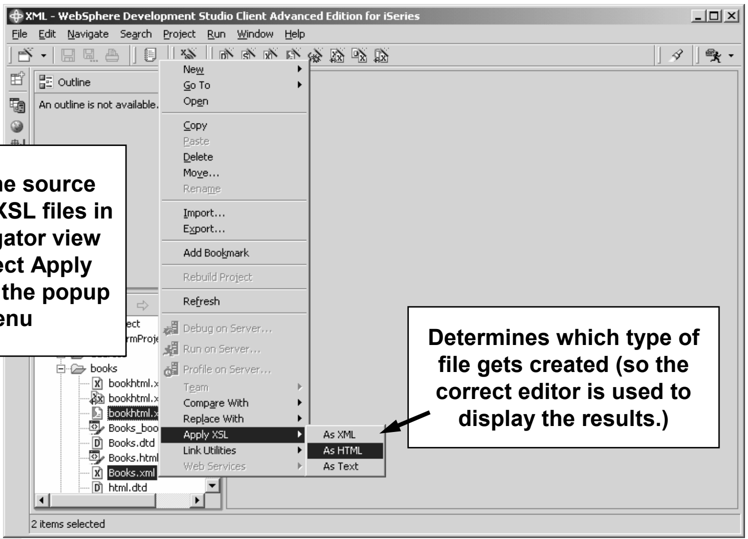
```
<!-- Newly-defined element template -->
<xsl:template name="order:Order">
  <order:Order>
    <xsl:variable name="getOrderId" select="user0:getOrderId()"/>
    <xsl:variable name="getPrice" select="user0:getPrice(/webOrder
    <xsl:attribute name="id">
      <xsl:value-of select="$getOrderId"/>
    </xsl:attribute>
    <xsl:call-template name="order:Date-Received"/>
    <order:Quantity>
      <xsl:value-of select="/webOrder:PurchaseOrder/webOrder:Quant
    </order:Quantity>
    <order:Purchaser-Order-Id>
      <xsl:value-of select="/webOrder:PurchaseOrder/webOrder:Id/te
    </order:Purchaser-Order-Id>
    <order:Purchaser-Item-Code>
      <xsl:value-of select="/webOrder:PurchaseOrder/webOrder:Item-
    </order:Purchaser-Item-Code>
    <order:Unit-Price>
      <xsl:value-of select="$getPrice"/>
    </order:Unit-Price>
    <order:Total>
      <xsl:value-of select="($getPrice *
    </order:Total>
  </order:Order>
</xsl:template>

<!-- Newly-defined element template -->
<xsl:template name="order:Date-Received">
  <order:Date-Received>
    <order:Year>
```

XSLT script is automatically generated from the .xmx file

# XSL Debug

- Playback tool designed to work with XML-to-XML mapping editor
  - To test the generated XSLT transformation file
    - Apply the XSLT to an input XML file
    - Verify the output XML or HTML file is valid
    - Records the transformation made by the XALAN XSLT processor
  - Steps through the transformation
    - Highlights the transformation rules as they are fired
    - View the output XML or HTML in "real time"

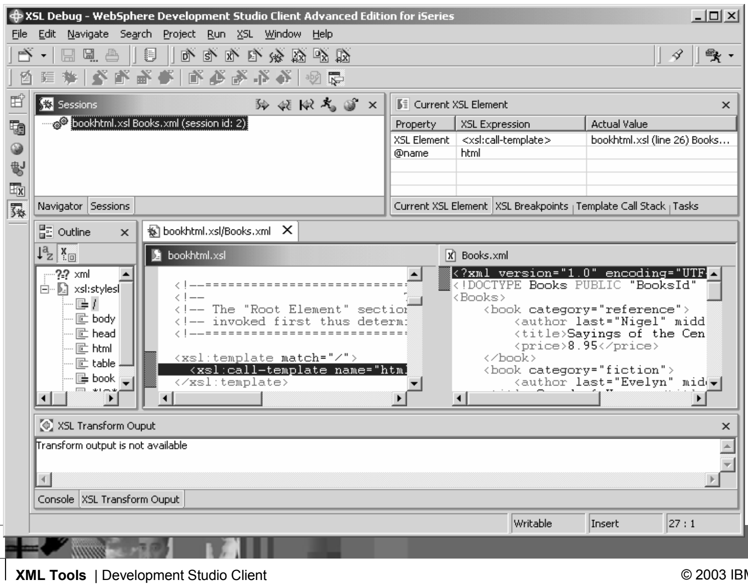
# XSL Debug



**Select the source XML and XSL files in the navigator view and select Apply XSL from the popup menu**

**Determines which type of file gets created (so the correct editor is used to display the results.)**

# XSL Debug



bookhtml.xsl Books.xml (session id: 2)

Current XSL Element

| Property | XSL Expression | Actual Value |
|---|---|---|
| XSL Element | <xsl:call-template> | bookhtml.xsl (line 26) Books… |
| @name | html | |

```
  <!--=============================
  <!--
  <!-- The "Root Element" sectio
  <!-- invoked first thus determ
  <!--=============================

  <xsl:template match="/">
    <xsl:call-template name="htm
  </xsl:template>
```

```
<?xml version="1.0" encoding="UTF
<!DOCTYPE Books PUBLIC "BooksId"
<Books>
    <book category="reference">
        <author last="Nigel" midd
        <title>Sayings of the Cen
        <price>8.95</price>
    </book>
    <book category="fiction">
        <author last="Evelyn" mid
```

Transform output is not available

# XML Development Tools: Generating Java Beans

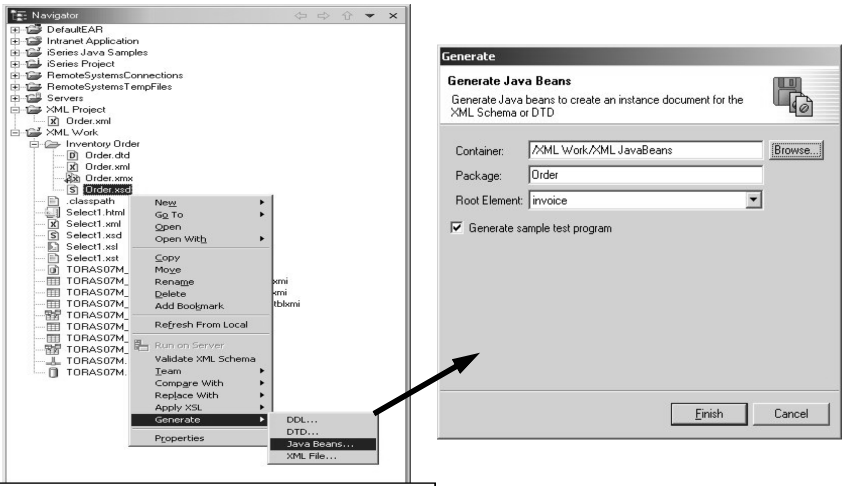
WebSphere. software

e business software

# Generating JavaBeans from DTD or XML Schema

- Generated JavaBeans handle the XML parsing for you!
  - You don't have to learn how to code XML parser API calls!
  - More natural way of working with XML documents
- Use the JavaBeans in your Java applications, Servlets or EJBs to:
  - Read existing XML documents and retrieve the information
    - The XML file is read from an input stream (disk, socket, …), parsed and returned as instances of the JavaBeans
  - Create a new XML file
    - Use the Factory class to create a new XML document by creating instances of the JavaBeans

# Generating JavaBeans



Generate

**Generate Java Beans**

Generate Java beans to create an instance document for the XML Schema or DTD

Container: /XML Work/XML JavaBeans

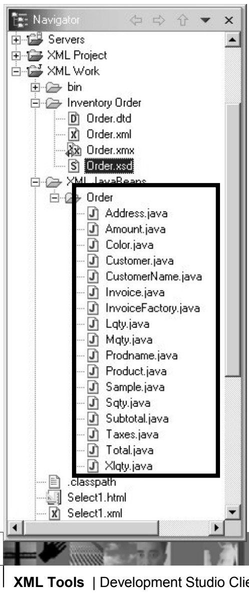
Package: Order

Root Element: invoice

☑ Generate sample test program

JavaBeans can be generated from either a DTD or XML Schema

# Generated JavaBeans



- Generated JavaBeans
  - One class for each XML Element
  - *RootElement*Factory
    - Generated for the root element
    - Methods
      - loadDocument(String xmlFileName)
      - save(String xmlFileName)
      - create*ElementName*(...)
  - Sample
    - Sample Java program with code that creates / saves and loads XML file using generated classes

# XML Development Tools: XML from SQL Wizard

WebSphere. software

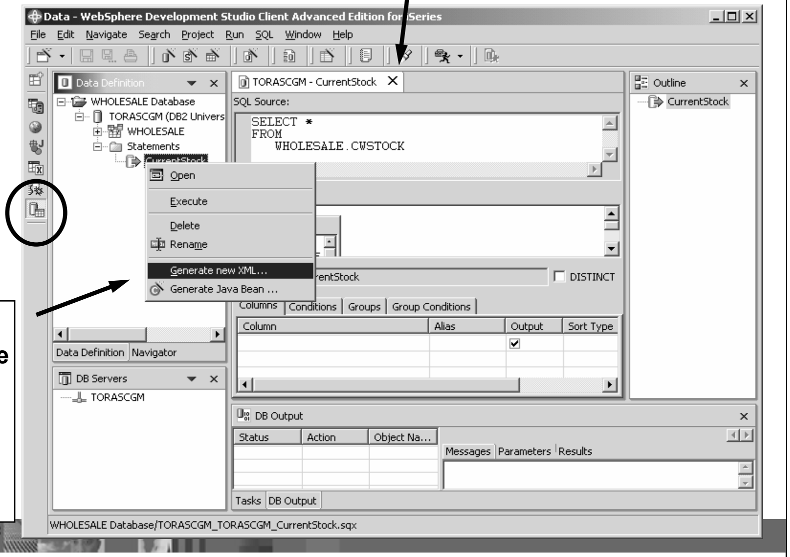
# XML and SQL Query Wizard

- Generate the following from an SQL query:
  - XML file and HTML file - Both contain the query results
  - DTD or XML Schema - Describes format of the resulting XML file
  - XSLT - For transforming resulting XML document to HTML
  - XST
    - XML query template file
    - Use XST file in your application to generate the XML file from the SQL query at runtime
    - Requires the following runtimes (all shipped with WDSc)
      - Xerces XML parser and Xalan XSL processor (Apache)
      - SQLToXML (IBM)
- Can also be used for writing XML documents to a relational database
- First you need to create an SQL query either manually or using the Data perspective

# Creating An SQL Query

**SQL Statement editor is part of the Database tools included in the Workbench**

**Data Perspective**



```
SELECT *
FROM
   WHOLESALE.CWSTOCK
```

WHOLESALE Database/TORASCGM_TORASCGM_CurrentStock.sqx

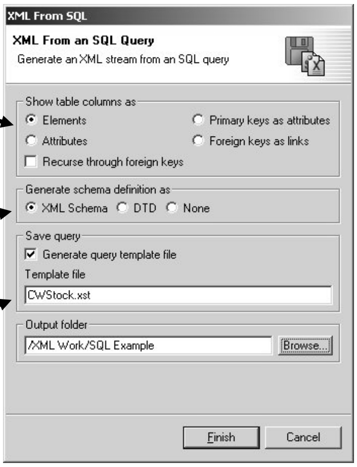
**Right click on the SQL statement and select *Generate New XML…***

# XML From SQL Query Wizard

In the resulting XML file each column in the database table will be stored as either a separate element or as an attribute (with one element per row)

Optionally generate a DTD or XML Schema for that describes the results of the query

Generate the query template file if you will be rerunning the query at runtime in your Java application



XML From SQL

**XML From an SQL Query**

Generate an XML stream from an SQL query

Show table columns as

● Elements ○ Primary keys as attributes

○ Attributes ○ Foreign keys as links

□ Recurse through foreign keys

Generate schema definition as

● XML Schema ○ DTD ○ None

Save query

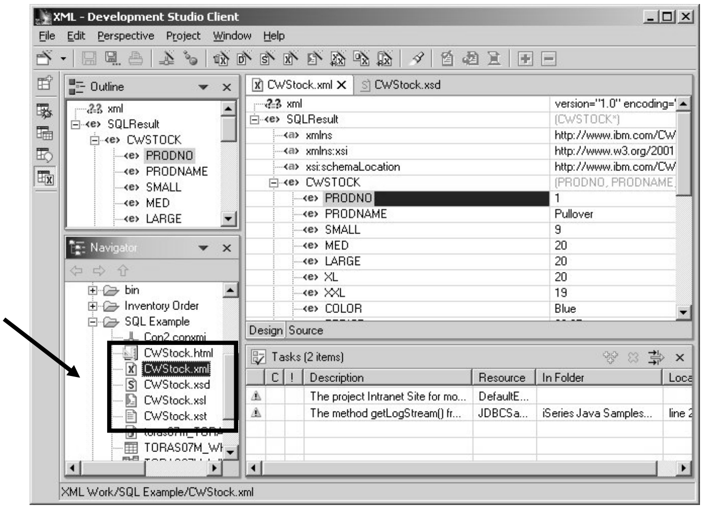
☑ Generate query template file

Template file: CWStock.xst

Output folder: /XML Work/SQL Example

# Result

- ►HTML and XML files with results of running the SQL query
- ►XML Schema file which describes format of results
- ►XSL file for transforming XML result to HTML
- ►XST query template file for executing the query at runtime

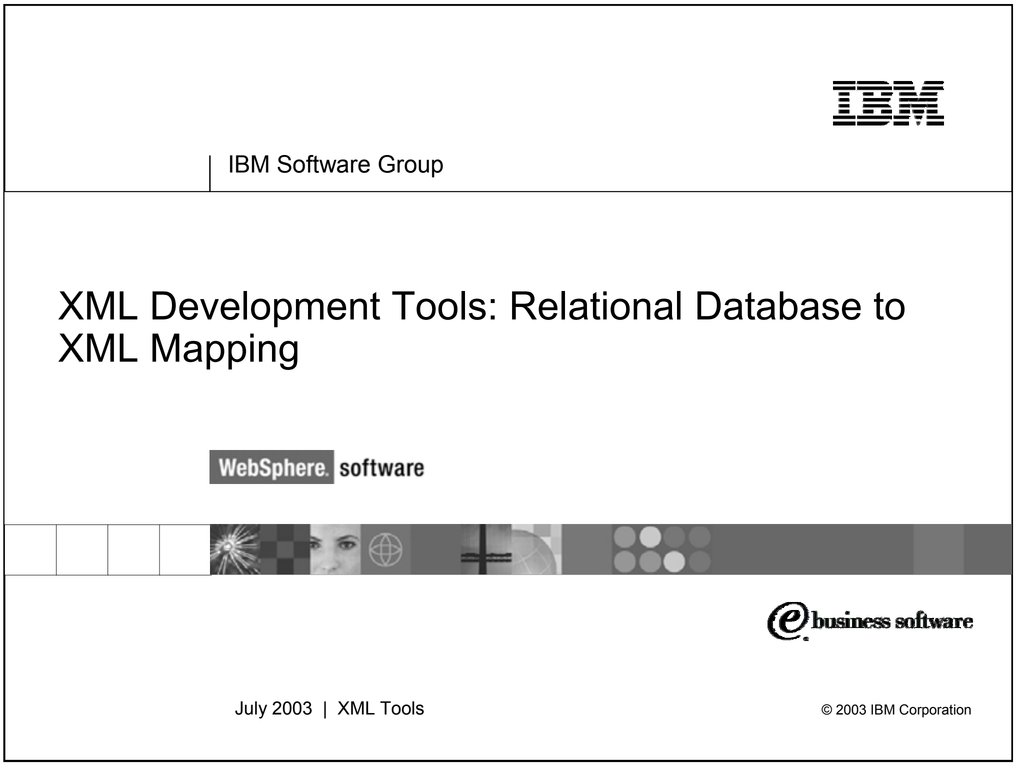


| | |
|---|---|
| xml | version="1.0" encoding=' |
| SQLResult | (CWSTOCK*) |
| xmlns | http://www.ibm.com/CW |
| xmlns:xsi | http://www.w3.org/2001 |
| xsi:schemaLocation | http://www.ibm.com/CW |
| CWSTOCK | (PRODNO, PRODNAME, |
| PRODNO | 1 |
| PRODNAME | Pullover |
| SMALL | 9 |
| MED | 20 |
| LARGE | 20 |
| XL | 20 |
| XXL | 19 |
| COLOR | Blue |

| | C | ! | Description | Resource | In Folder | Loca |
|---|---|---|---|---|---|---|
| ⚠ | | | The project Intranet Site for mo... | DefaultE... | | |
| ⚠ | | | The method getLogStream() fr... | JDBCSa... | iSeries Java Samples... | line 2 |

XML Work/SQL Example/CWStock.xml

These files will not show up in the in Data or DBExplorer views, you must switch to the Navigator view



# XML Development Tools: Relational Database to XML Mapping

WebSphere. software
e business software

# Relational DB to XML Mapping

- Map XML data to a relational data
  - And relation data to XML data
- Generate a Document Access Definition (DAD) file
  - Use DAD file with DB2 XML Extender to
    - Retrieve relational data into XML files
    - Store XML files to a relational database
  - Also generates a test harness
- How is this different from XML-SQL Wizard?
  - More flexible
  - More work
    - Mappings must be specified in the mapping editor
  - Must be used in conjunction with DB2 XML Extenders

# XML Development Tools: Summary

# Summary

- Lots of great XML Tools
  - XML, DTD, XML Schema and XSLT wizards and editors
  - Generate JavaBeans to easily read, write and create XML documents in Java
  - Tools for creating XML documents from and writing XML documents to relational databases

- Plus, as an integrated tool in the Workbench the XML Tooling inherits:
  - Searching
  - Team development (CVS, Rational ClearCase, iSeries SCM vendors)
  - Integration with other tooling
    - Java development tools
    - Web development tools
    - WebSphere test environment
    - iSeries development tools
  - Integrated, online help
  - Easy to use import and export wizards

# XML Development Tools: Additional Resources

WebSphere. software

e business software

# XML and the iSeries

- How do I access XML on the iSeries
- XML Toolkit for iSeries
  - licensed program 5733-XT1
  - Contains two XML parsers
    - XML4C 4.0
      - Use with C++
  - XML4PR 4.0 (PR -> Procedure Languages)
    - Use with ILE C, ILE RPG and ILE COBOL
- XML Parser for Java
  - Shipped as part of OS/400
    - /QIBM/ProdData/Os400/xml/lib
    - Includes Xerces parser and Xalan processor

# Lots of Information on the Web

- w3c.org

  **XML Standards**
- ibm.com/developer/XML

  **Links to online articles**
- xml.org

  **Links to online articles**
- xml.apache.org

  **XML Xerces parser**

  **XSLT Xalan processor**
- xfront.com

  **Tutorial on XML Schema**

# Trademarks & Disclaimers

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

| | | | |
|---|---|---|---|
| AS/400 | IBM(logo) | | |
| AS/400e | iSeries | | |
| e (logo) business | OS/400 | | |
| IBM | | | |

Lotus, Freelance Graphics, and Word Pro are registered trademarks of Lotus Development Corporation and/or IBM Corporation.
Domino is a trademark of Lotus Development Corporation and/or IBM Corporation.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.
Other company, product and service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

# Disclaimer

- **Acknowledgment:**

  This presentation is a collaborative effort of the IBM Toronto AS/400 Application Development presentation team, including work done by:

  Don Yantzi, Phil Coulthard, George Farr, Claus Weiss, David Slater, Alison Butteril, Linda Cole

- **Disclaimer:**

  The information contained in this document has not been submitted to any formal IBM test and is distributed on an as is basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customers' ability to evaluate and integrate them into the customers' operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

- **Reproduction:**

  The base presentation is the property of IBM Corporation. Permission must be obtained PRIOR to making copies of this material for any reason.

IBM Software Group

# XML Development Tools in IBM WebSphere Development Studio Client for iSeries Version 5.0

WebSphere. software

e business software