XML -- table of contents

Development

0.3.2: What are application server configuration files?

Application server configuration files define the administrative configuration of your WebSphere Application Server product installation. That is, applicationconfiguration files describe the available application servers, their configurations, andtheir contents.

WebSphere Application Server Advanced Edition Version 3.5 stored administrative data in a database. Version 4.0 (non-Single Server Edition) also stores administrative datain a database.

Version 4.0 (Single Server Edition) does not use a database. Instead, it uses application server configuration files, which are flattened, XML versions of the administrative database. When you start the product, the XML is parsedto determine the contents of the application server being started.

Location of the application server configuration files

One or more configuration files are provided for you to copy and modify. The followingfile contains a configuration for a default application server and several other applications and resources that you can use as examples or defaults:

product_installation_root`/config/server-cfg.xml`

It also contains the administrative application, thin-admin.

A second file contains a very basic configuration on which you can build:

product_installation_root`/config/template-server-cfg.xml`

A third file contains a stripped down version of the default server-cfg.xml file.It contains just the configuration for the administration application. The admin-server-cfg.xml is provided in case you want to run the administrative console application in a separate application server instance.

product_installation_root`/config/admin-server-cfg.xml`

Use the console to edit files

Use the administrative console to modify configurationfiles, rather than modifying them directly.

How can I tell which one I am editing?

To determine which configuration file that you are currently using theadministrative console to edit, start the administrative console. Look for the server configuration file name,under the banner, towards the top left corner of the screen.

Note, a different configuration file could be in use by the runningapplication server.

4.2.3: Incorporating XML

IBM WebSphere Application Server provides XML Document Structure Services, which consist of a document parser, a document validator, and a document generator for server-side XML processing.

See article 4.1.1.2 for all of the details about XML supportin the product.If you are just becoming familiar with XML, start with article 0.33, a primer on XML concepts, vocabulary, and uses.

Other related information provides guidance on the following topics:

- Structure -- defining and obeying the syntax for an XML tag set
- Content -- determining the mechanism for filling XML tags with data
- Presentation -- determining the mechanism for formatting and displaying XML content

In addition, some special topics are covered, including DOM objects andmanipulation of Channel Definition Format (CDF) files as illustrated bythe SiteOutliner example.

When you install IBM WebSphere Application Server, the core XML APIs are automatically added to the appropriate class path, enabling you to serve static XML documents as soon as the product is installed.

To serve XML documents that are dynamically generated, use the core APIs to develop servlets or Web applications that generate XML documents (for example, the applications might read the document content from a database) and then deploy those components on your application server.

4.2.3.2: Specifying XML document structure

The structure of an XML document is governed by syntax rules for its tag set. Those tags are defined formally in an XML-based grammar, such as a Document Type Definition (DTD). At the time of this publication, DTD is the most widely-implemented grammar. Therefore, this article discusses options for using DTDs.

Options for XML document structure include:

Do not use a DTD. Not using a DTD enables maximum flexibility in evolving XML document structure, but this flexibility limits the ability to share the documents among users and applications. An XML document can be parsed without a DTD. If the parser does not find an inline DTD or a reference to an external DTD, the parser proceeds using the actual structure of the tags within the document as an implied DTD. The processor evaluates the document to determine whether it meets the rules for well-formedness.

Use a public DTD. Various industry and other interest groups are developing DTDs for categories of documents, such as chemical data and archival documents. Many of these DTDs are in the public domain and are available over the Internet. Using an industry standard DTD maximizes sharing documents among applications that act on the grammar. If the standard DTD does not accomodate the schema the applications need, flexibility is limited.

Several industry and interest groups have developed and proposed DTD grammars for the types of documents they produce and exchange. To make it easier for you to use those grammars, local copies are installed with the product. Use the grammars as examples in developing your own grammars as well as for creating and validating XML documents of those types. The library is located atproduct_installation_root\web\xml\grammar\

Develop a DTD. If none of the public DTDs meet an enterprise's needs and enforcing document validity is a requirement, the XML implementers can develop a DTD. Developing a DTD requires careful analysis of the information (data) that the documents will contain.

For DTD updates,visit the XML Industry Portal.For details about the DTD specifications and sample DTDs, refer toIBM's developerWorkssite for education and other DTD resources.

4.2.3.3: Providing XML document content

The content of an XML document is the actual data that appears within the document tags. XML implementers must determine the source and the mechanism for putting the data into the document tags. The options include:

Static content. XML document content is created and stored on the Web server as static files. The XML document author composes the document to include valid XML tags and data in a manner similar to how HTML authors compose static HTML files. This approach works well for data that is not expected to change or that will change infrequently. Examples are journal articles, glossaries, and literature.

Dynamically generated content. XML document content can be dynamically generated from a database and user input. In this scenario, XML-capable servlets, Java beans, and even inline Java code within a JavaServer Page (JSP) file can be used to generate the XML document content.

A hybrid of static and dynamically generated content. This scenario involves a prudent combination of static and dynamically generated content.

You can also use XSL to add to or remove information from existing XML content.For details, see the Related information.

4.2.3.4: Rendering XML documents

Options for presenting XML documents include:

Present the XML document in an XML-enabled browser. An XML-enabled browser can parse a document, apply its XSL stylesheet, and present the document to the user. Searching and enabling users to modify an XML document are other possible functions of XML-enabled browsers.

Present the XML document to a browser that converts XML to HTML. Until XML-enabled browsers are readily available, presenting XML documents to users will involve converting the XML document to HTML. That conversion can be handled by conversion-capable browsers. Another option is to use JavaScript or ActiveX controls embedded within the XML document. Microsoft Internet Explorer Version 5 is an XML-to-HTML converter. HTML is not the only format to which XML documents can be converted. It's just the easiest to implement given the commerically available browsers and user agents.

Send an HTML file to the browser. If the users do not have XML-capable browsers, the XML document must be converted at the server before being transmitted to the browser. The server-side XML application that handles the conversion could also determine the capability of the browser before converting the document to HTML, to avoid unnecessary processing if the browser is XML-capable. The XSL processor included with this product supports such server-side functions.

Using XSL to convert XML documents to other formats

IBM WebSphere Application Server includes the Lotus XSL processor and its open-sourceversion, Xalan, for formatting and converting XML documents. Processing can be done at the server or at the browser, to HTML or to other XML-compliant markup languages.For sample code, see the Xalan documentation.

Converting XML documents at the server

One option for presenting an XML document is for the server to convert the XML document to HTML and return the HTML document to the client. On the server side, this typically requires the creation of a servletto handle the processing of one data stream (the XML document)with another (the XSL document).The output of that process is then forwarded back to the browser.

Server-side processing often requires the passing in of parameters through theXSL processor to customize the output.For an example, see the Xalan documentation.

4.2.3.6: Using DOM to incorporate XML documents into applications

The Document Object Model (DOM) is an API for representing XML and HTML documentsas objects that can be accessed by object-oriented programs, such as business logic, for the purposes of creating, navigating, manipulating, and modifying the documents.

Article 0.33.3 introduces DOM concepts and vocabulary. Article 4.1.1.2 tells youwhere to find the DOM specification and org.w3c.dom package.

Article 4.2.3.6.1 providesa quick reference so that you can jump right into DOM development, referring to thepackage and specification as needed.

4.2.3.6.1: Quick reference to DOM object interfaces

This section highlights a few of the object interfaces. Refer to theDOM Specification for details (see article 4.1.1.2).

Node methods

Node methods include:

Method	Description
hasChildNodes	Returns a boolean to indicate whether a node has children
appendNode	Appends a new child node to the end of the list of children for a parent node
insertBefore	Inserts a child node before the existing child node
removeChild	Removes the specified child node from the node list and returns the node
replaceChild	Replaces the specified child node with the specified new node and returns the new node

Document methods

The Document object represents the entire XML document. Document methods include:

Method	Description
createElement	Creates and returns an Element (tag) of the type specified. If the document will be validated against a DTD, that DTD must contain an Element declaration for the created element.
createTextNode	Creates a Text node that contains the specified string
createComment	Creates a Comment node with the specified content (enclosed within <! - - and - -> tags)
createAttribute	Creates an Attribute node of the specified name. Use the setAttribute method of Element to set the value of the Attribute. If the document will be validated against a DTD, that DTD must contain an Attribute declaration for the created attribute.
createProcessingInstruction	Creates a Processing Instruction with the specified name and data (enclosed within <? and ?> tags). A processing instruction is an instruction to the application (such as an XML document formatter) that receives the XML document.

Element methods

Element node methods include:

Method	Description
getAttribute	Returns the value of the specified attribute or empty string
setAttribute	Adds a new attribute-value pair to the element

removeAttribute	Removes the specified attribute from the element
getElementsByTagName	Returns a list of the element descendants that have the specified tag name

A Text node can be a child of an Element or Attribute node and contains the textual content (character data) for the parent node. If the content does not include markup, all of the content is placed within a single Text node. If the content includes markup, that markup is placed in one or more Text nodes that are siblings of the Text node that contains the non-markup content.

The Text node extends the CharacterData interface, which has methods for setting, getting, replacing, inserting, and making other modifications to a Text node. In addition to those methods, the Text node adds a method:

Method	**Description**
splitText	Splits the Text node at the specified offset. Returns a new Text node, which contains the original content starting at the offset. The original Text node contains the content from the beginning to the offset.

4.2.3.7: SiteOutliner sample

The SiteOutliner servlet illustrates how to use the XML Document Structure Services to generate and view a Channel Definition Format (CDF) file for a target directory on the servlet's Web server. Use Lotus Notes 5 (the Headlines page), Microsoft Internet Explorer 4 Channel Bar, PointCast, Netscape Navigator 4.06, orother CDF-capable viewers to view and manipulate the CDF file.

SiteOutliner is part of the WebSphere Samples Gallery. When you open the gallery,follow the links to SiteOutliner to run it on your local machine.