IBM Software Group

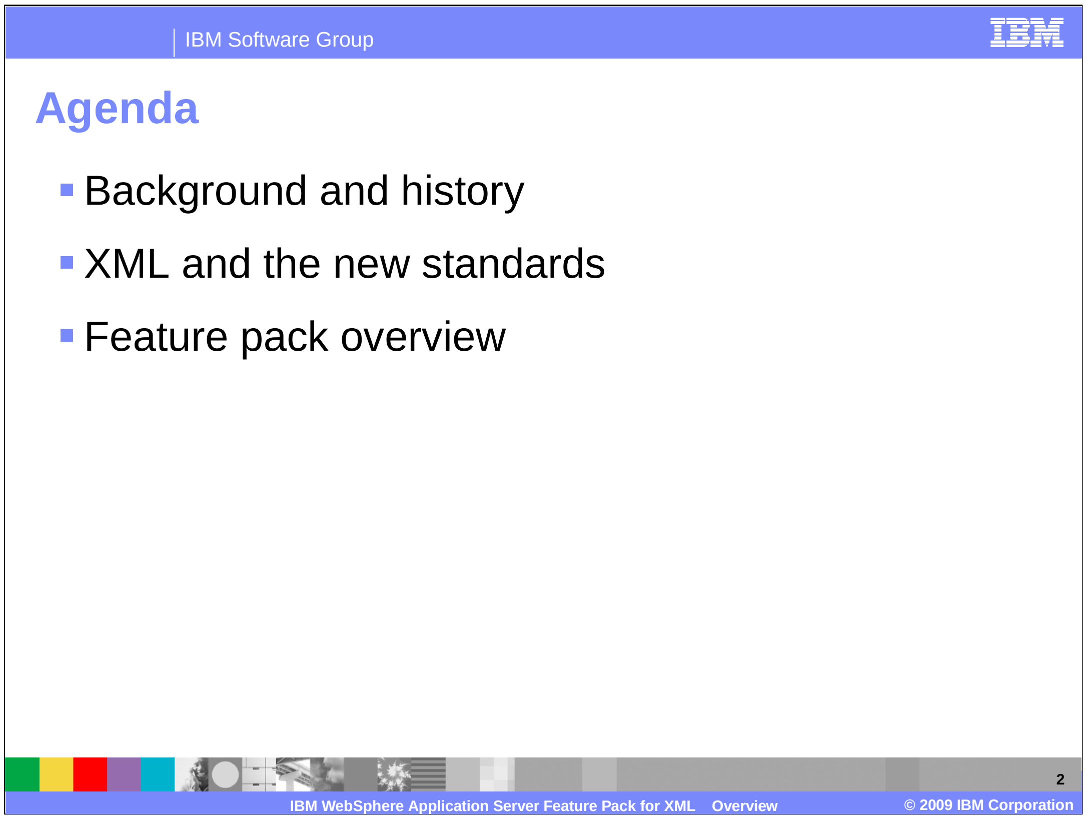
IBM WebSphere Application Server Feature Pack for XML

Overview

business on demand.

Updated November 18, 2009

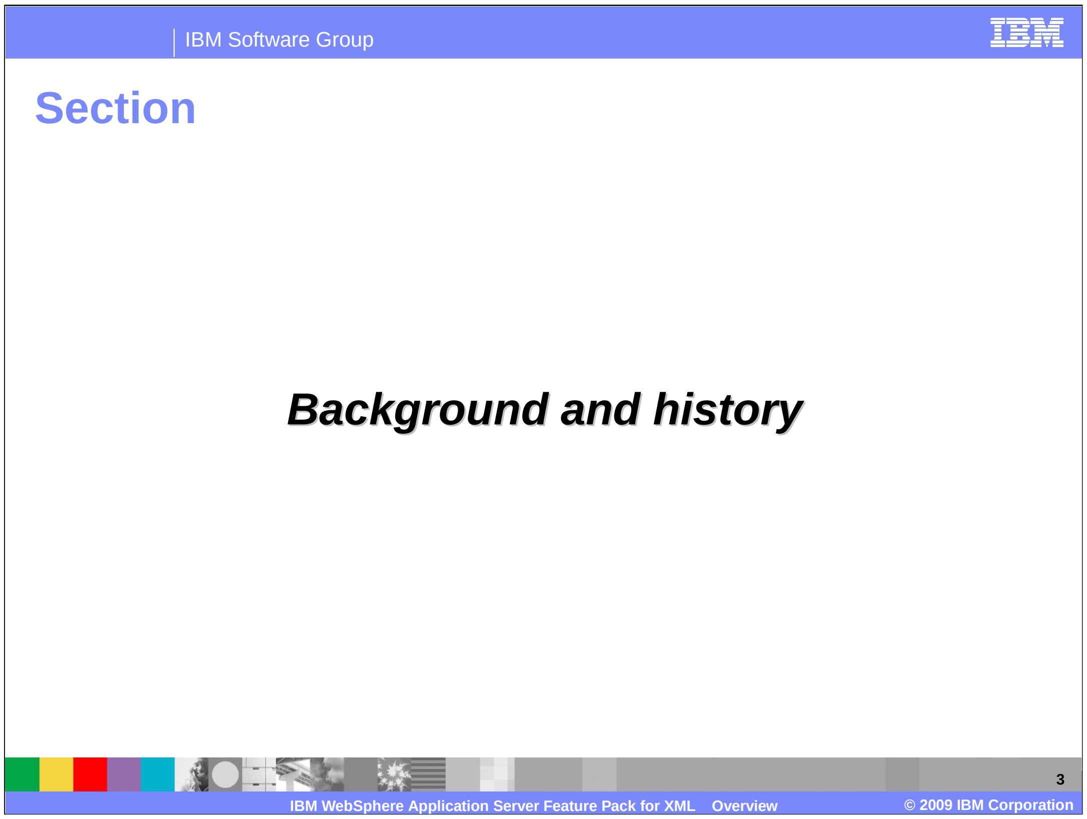
This presentation is an overview of the IBM WebSphere® Application Server Feature Pack for XML.



Agenda

- Background and history
- XML and the new standards
- Feature pack overview

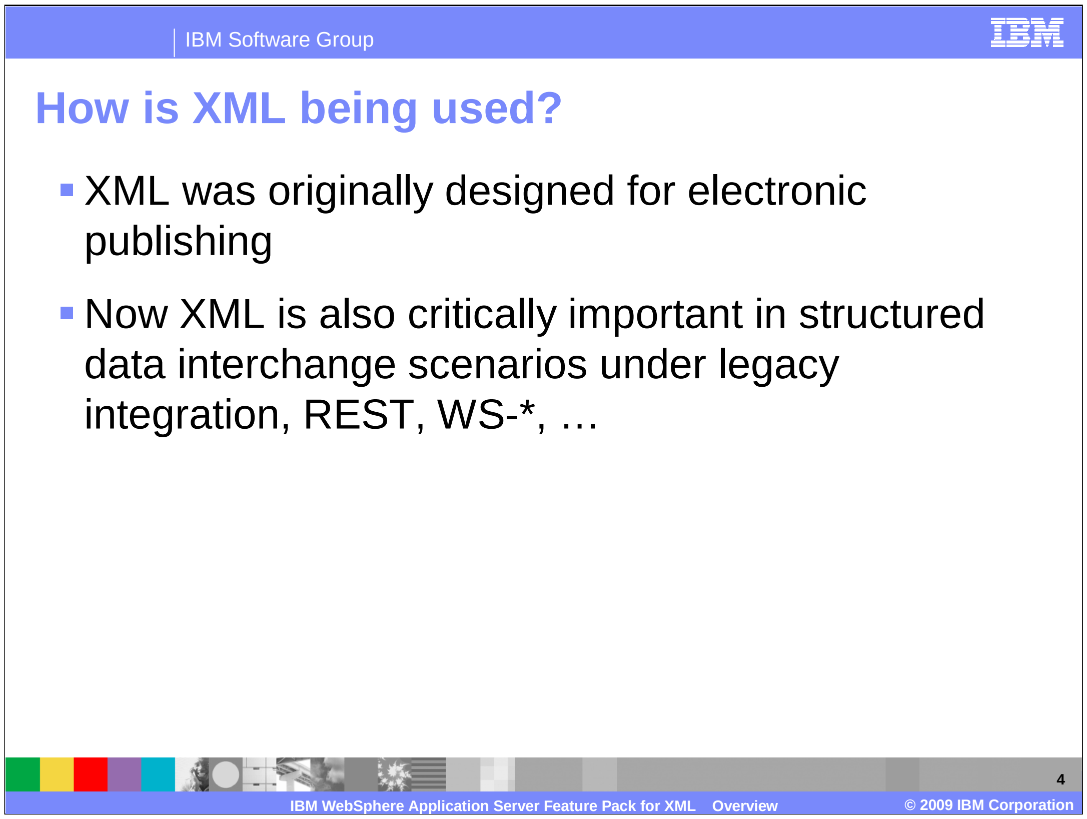
This presentation covers some basic XML background and history and XML's usage in WebSphere Application Server. It will also go through XML and the new standards, and a basic overview of the feature pack.



Section

Background and history

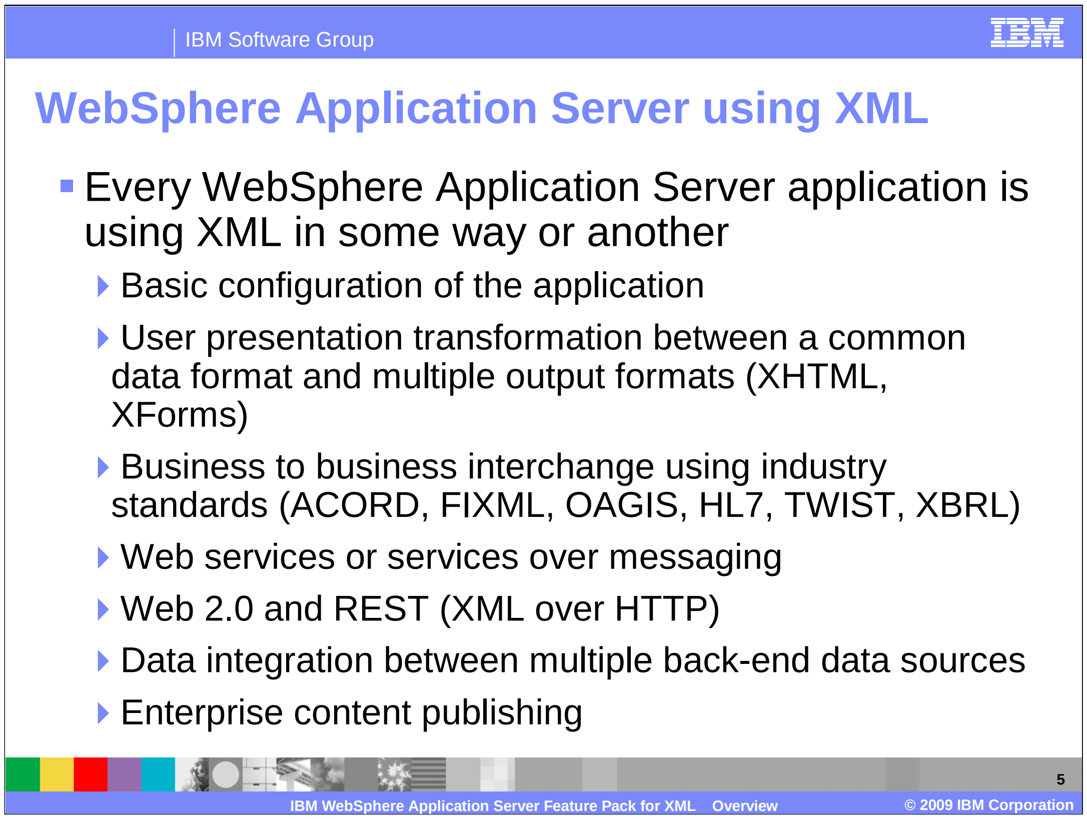
The next few slides will demonstrate the XML usage in Java™ based middleware and particularly in WebSphere Application Server. These slides will show the typical programming models used for the common application server XML scenarios.



How is XML being used?

- XML was originally designed for electronic publishing
- Now XML is also critically important in structured data interchange scenarios under legacy integration, REST, WS-*, …

XML was originally designed for electronic publishing, document style. Now XML is also critically important in structured data interchange scenarios, data style, under legacy integration, REST, WS-*, and so on. Every WebSphere Application Server application is using XML in some way or another.


IBM Software Group

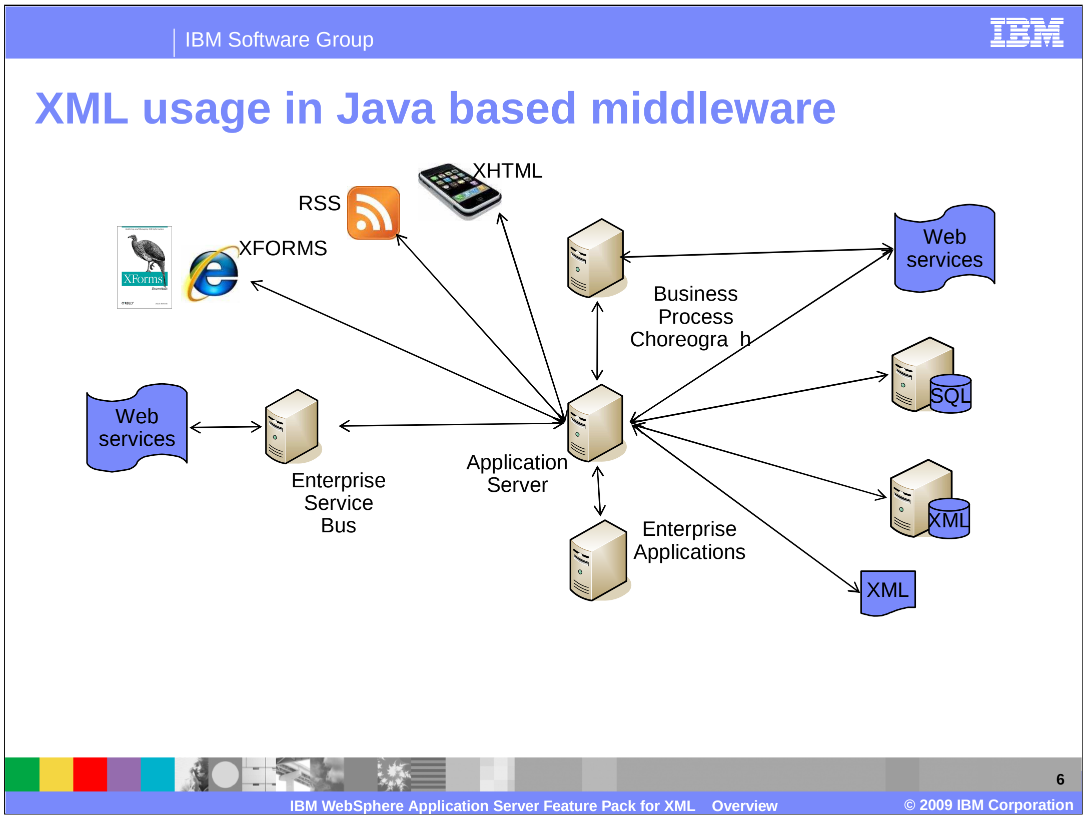
WebSphere Application Server using XML

- Every WebSphere Application Server application is using XML in some way or another
  - Basic configuration of the application
  - User presentation transformation between a common data format and multiple output formats (XHTML, XForms)
  - Business to business interchange using industry standards (ACORD, FIXML, OAGIS, HL7, TWIST, XBRL)
  - Web services or services over messaging
  - Web 2.0 and REST (XML over HTTP)
  - Data integration between multiple back-end data sources
  - Enterprise content publishing

5

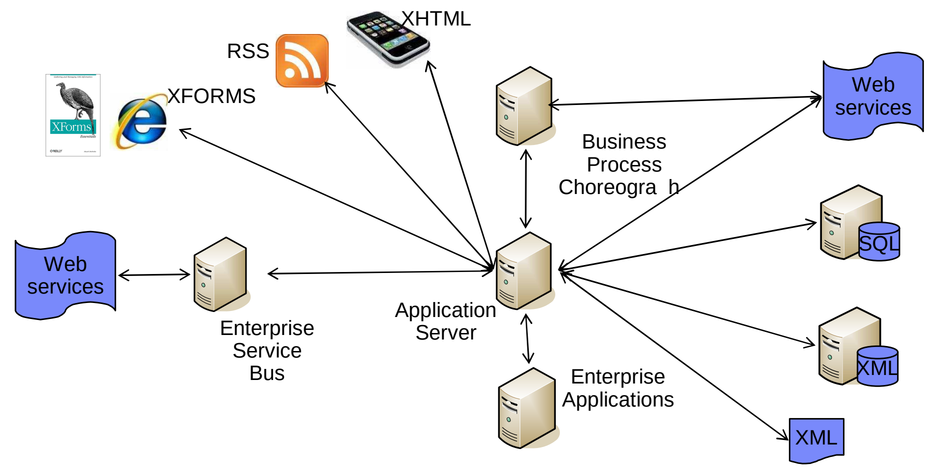
IBM WebSphere Application Server Feature Pack for XML Overview © 2009 IBM Corporation

XML is being used in the basic configuration of the application and user presentation transformation with outputs such as XHTML and XForms. XML is also used in business-to-business interchange, Web services, Web 2.0 and REST, data integration and enterprise content publishing.



IBM Software Group

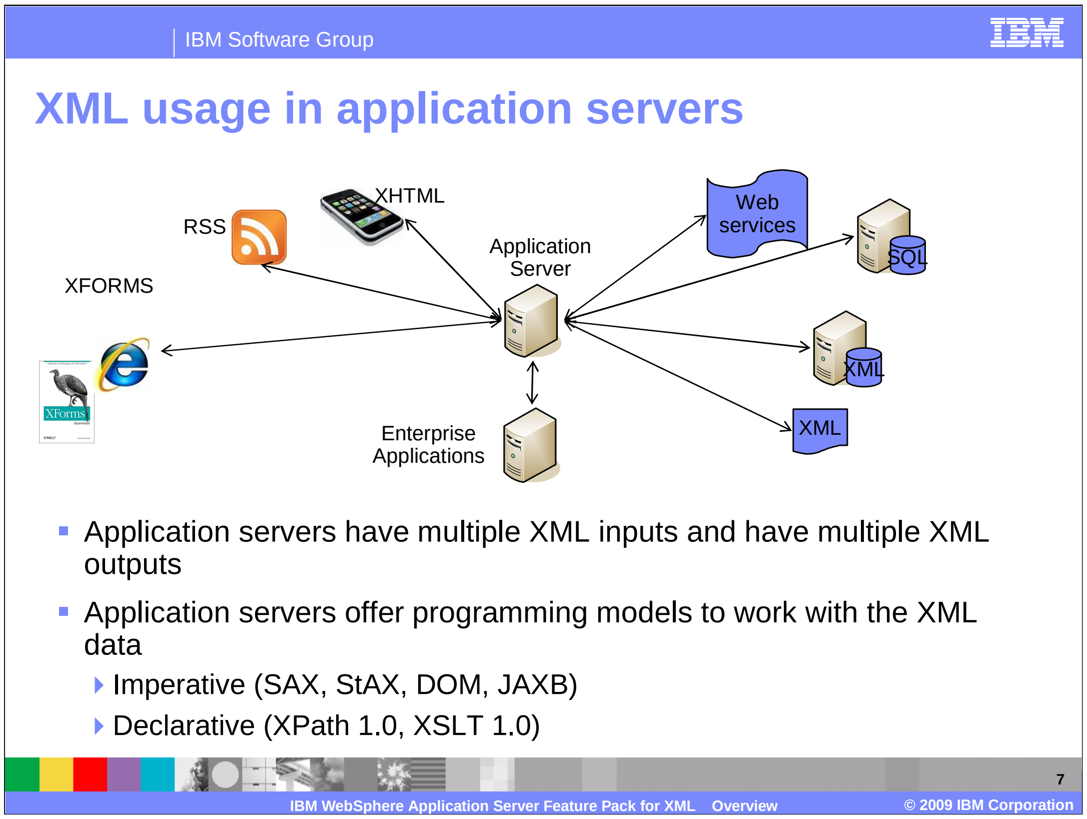
XML usage in Java based middleware



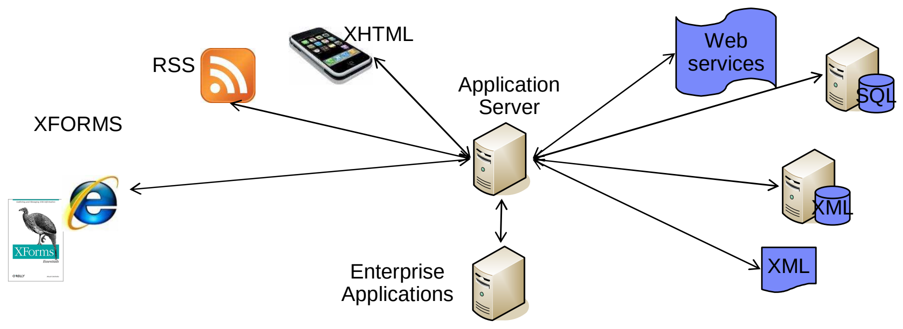
6

IBM WebSphere Application Server Feature Pack for XML Overview © 2009 IBM Corporation

XML is used in Java-based middleware, computer software that helps connects components or applications through services based on Java. XML and Java technology are natural partners in helping developers exchange data and programs across the Internet. That is because XML has emerged as the standard for exchanging data across disparate systems, and Java technology provides a platform for building portable applications. This partnership is particularly important for Web services, which promise users and application developers program functionality on demand from anywhere to anywhere on the Web. XML and Java technology are recognized as ideal building blocks for developing Web services and applications that access Web services.



XML usage in application servers



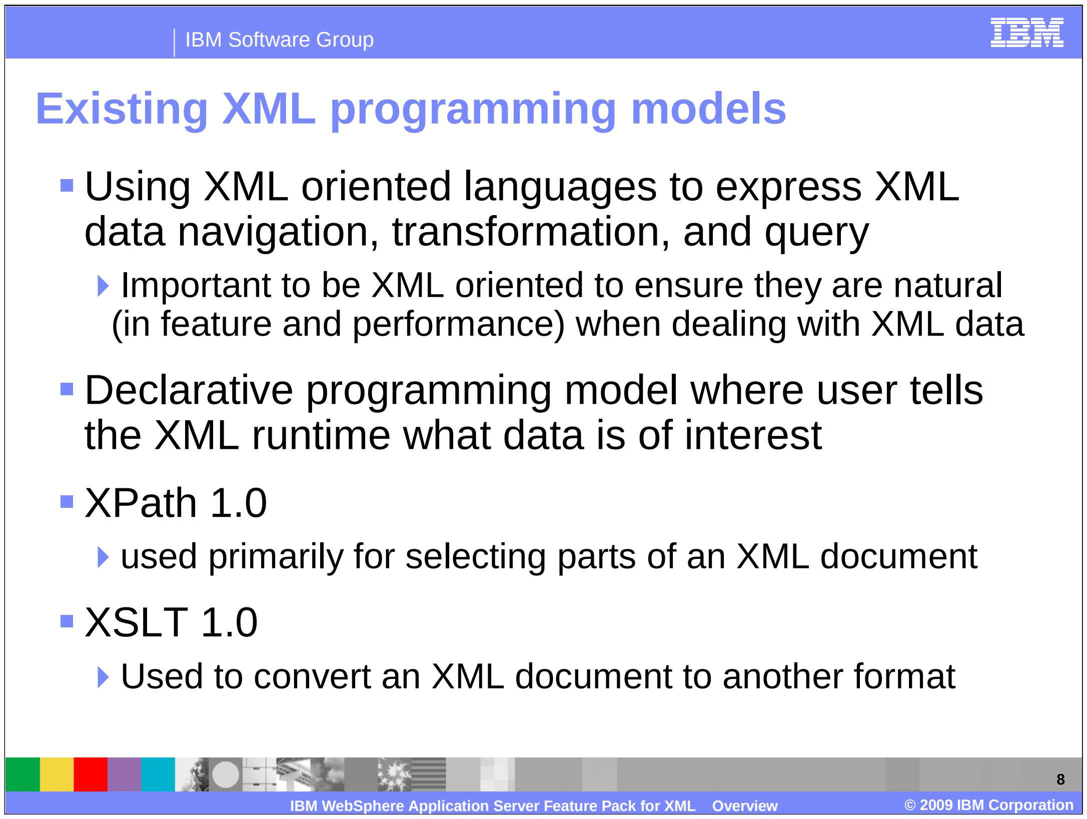
- Application servers have multiple XML inputs and have multiple XML outputs
- Application servers offer programming models to work with the XML data
  - Imperative (SAX, StAX, DOM, JAXB)
  - Declarative (XPath 1.0, XSLT 1.0)

Application servers have multiple XML inputs such as Web services, XML databases, relational databases, enterprise applications exported as XML, and XML data stores. Application servers have multiple XML outputs such as XHTML presentation, RSS feeds, Web services, and XForms presentation. So the question that comes to mind is: How do you access and use an XML document through the Java programming language?

Application servers offer programming models to work with the XML data. A programming model is **imperative** if you specifically tell the Java runtime how to navigate data, as compared to **declarative** if you tell the XML runtime what data is of interest.

Application servers offer imperative programming models to work with the XML. One way to do this, perhaps the most typical way, is through parsers that conform to the Simple API for XML - SAX- or the Document Object Model - DOM. Both of these parsers are provided by Java API for XML Processing, JAXP. Java developers can invoke a SAX or DOM parser in an application through the JAXP API to parse an XML document - that is, scan the document and logically break it up into discrete pieces. The parsed content is then made available to the application. There is also the Java Architecture for XML Binding, called JAXB, which provides a way to process XML content using Java objects by binding its XML schema to Java representation.

Application servers can also offer declarative programming models to work with the XML such as the XML Path Language, XPath 1.0 and XSL Transformations, XSLT 1.0.


IBM Software Group

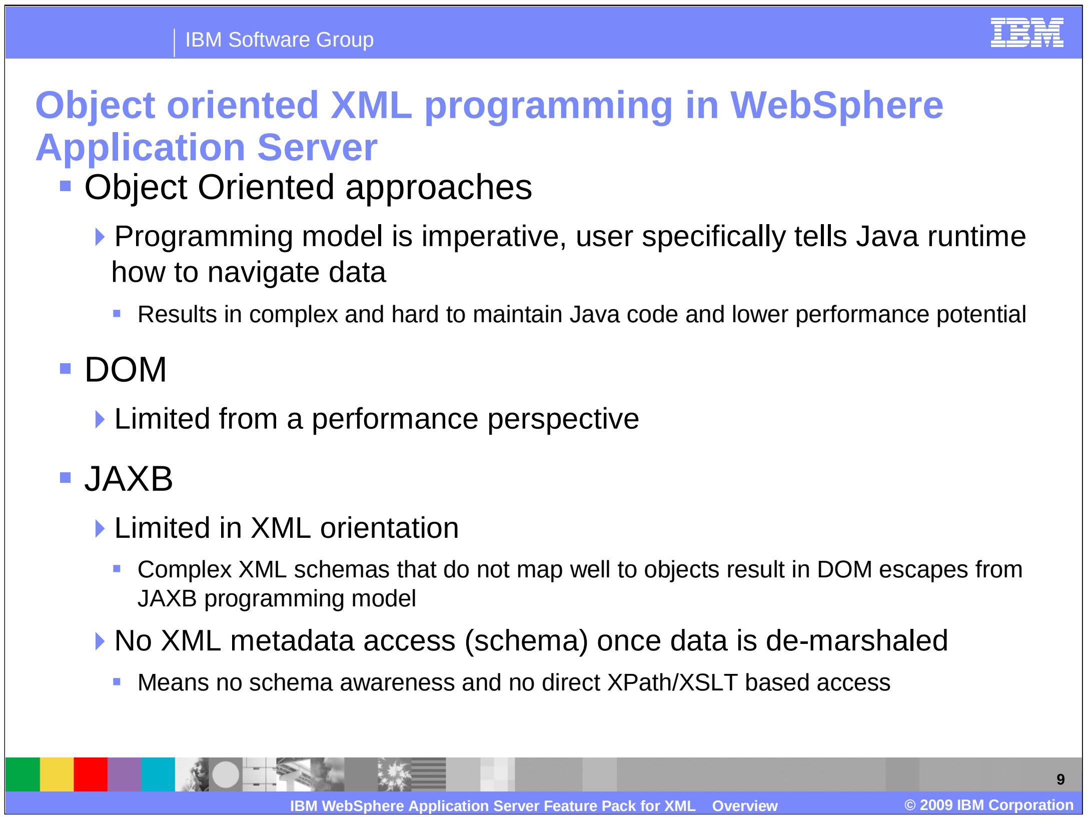
Existing XML programming models

- Using XML oriented languages to express XML data navigation, transformation, and query
  - Important to be XML oriented to ensure they are natural (in feature and performance) when dealing with XML data
- Declarative programming model where user tells the XML runtime what data is of interest
- XPath 1.0
  - used primarily for selecting parts of an XML document
- XSLT 1.0
  - Used to convert an XML document to another format

8

IBM WebSphere Application Server Feature Pack for XML Overview © 2009 IBM Corporation

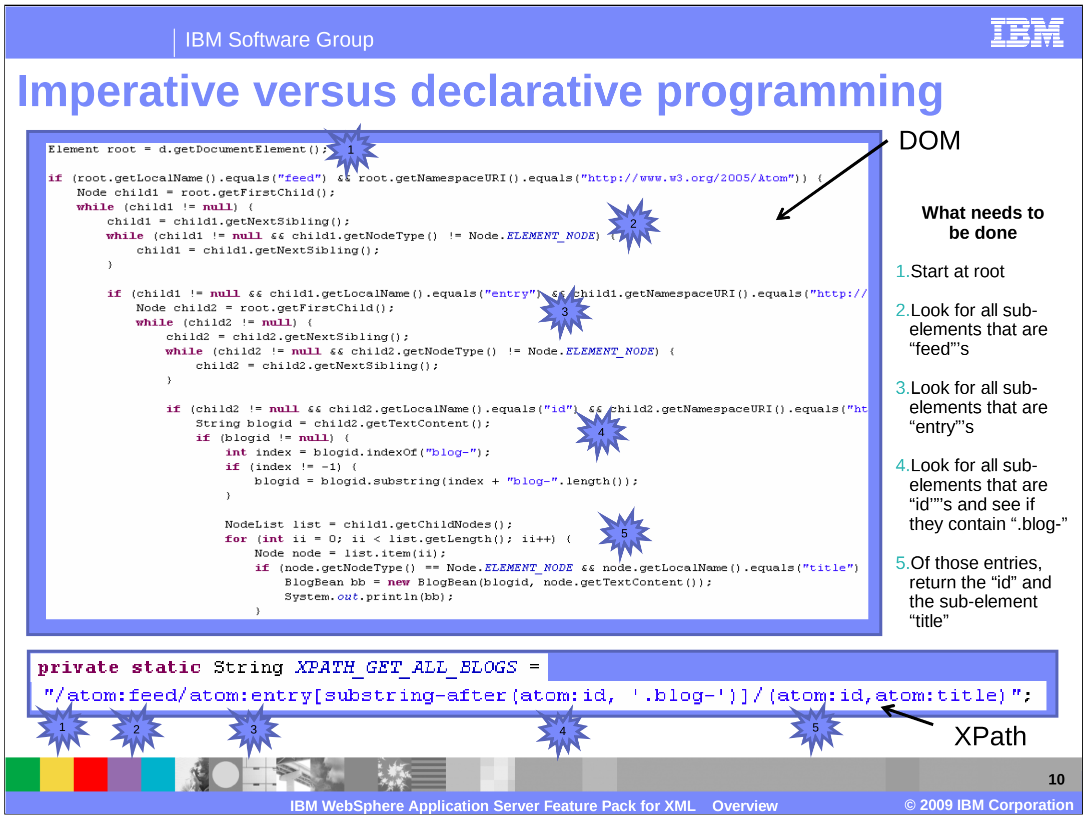
The existing XML programming models are XPath 1.0 and XSLT 1.0. These are declarative programming models where you tell the XML runtime what data is of interest. These models use XML oriented languages to express XML data navigation, transformation, and queries. It is important to be XML oriented to ensure they are natural in feature and performance when dealing with XML data. XPath 1.0 is used primarily for selecting parts of an XML document. It allows nodes to be selected by means of a hierarchic navigation path through the document tree. An example of XPath is that it allows you to navigate all sales orders that are greater than $100. XSLT 1.0 is used to convert an XML document to another format. For example, convert all existing sales orders to a bill of materials.



Object oriented XML programming in WebSphere Application Server

- Object Oriented approaches
  - Programming model is imperative, user specifically tells Java runtime how to navigate data
    - Results in complex and hard to maintain Java code and lower performance potential
- DOM
  - Limited from a performance perspective
- JAXB
  - Limited in XML orientation
    - Complex XML schemas that do not map well to objects result in DOM escapes from JAXB programming model
  - No XML metadata access (schema) once data is de-marshaled
    - Means no schema awareness and no direct XPath/XSLT based access

Currently there are some object oriented XML programming approaches used in the WebSphere Application Server. This programming model is imperative meaning that you specifically tell the Java runtime how to navigate data. This results in complex and hard to maintain Java code, and results in lower performance potential. The DOM approach is limited from a performance perspective. JAXB provides a way to process XML content using Java objects by binding its XML schema to a Java representation. JAXB is limited in XML orientation. Complex XML schemas that do not map well to objects result in DOM escapes from JAXB programming model. Also, there is no XML metadata access (schema) once data is de-marshaled. This means no schema awareness and no direct XPath and XSLT based access.



Imperative versus declarative programming

```
Element root = d.getDocumentElement();

if (root.getLocalName().equals("feed") && root.getNamespaceURI().equals("http://www.w3.org/2005/Atom")) {
    Node child1 = root.getFirstChild();
    while (child1 != null) {
        child1 = child1.getNextSibling();
        while (child1 != null && child1.getNodeType() != Node.ELEMENT_NODE) {
            child1 = child1.getNextSibling();
        }

        if (child1 != null && child1.getLocalName().equals("entry") && child1.getNamespaceURI().equals("http://
            Node child2 = root.getFirstChild();
            while (child2 != null) {
                child2 = child2.getNextSibling();
                while (child2 != null && child2.getNodeType() != Node.ELEMENT_NODE) {
                    child2 = child2.getNextSibling();
                }

                if (child2 != null && child2.getLocalName().equals("id") && child2.getNamespaceURI().equals("ht
                    String blogid = child2.getTextContent();
                    if (blogid != null) {
                        int index = blogid.indexOf("blog-");
                        if (index != -1) {
                            blogid = blogid.substring(index + "blog-".length());
                        }

                        NodeList list = child1.getChildNodes();
                        for (int ii = 0; ii < list.getLength(); ii++) {
                            Node node = list.item(ii);
                            if (node.getNodeType() == Node.ELEMENT_NODE && node.getLocalName().equals("title")
                                BlogBean bb = new BlogBean(blogid, node.getTextContent());
                                System.out.println(bb);
                            }
```

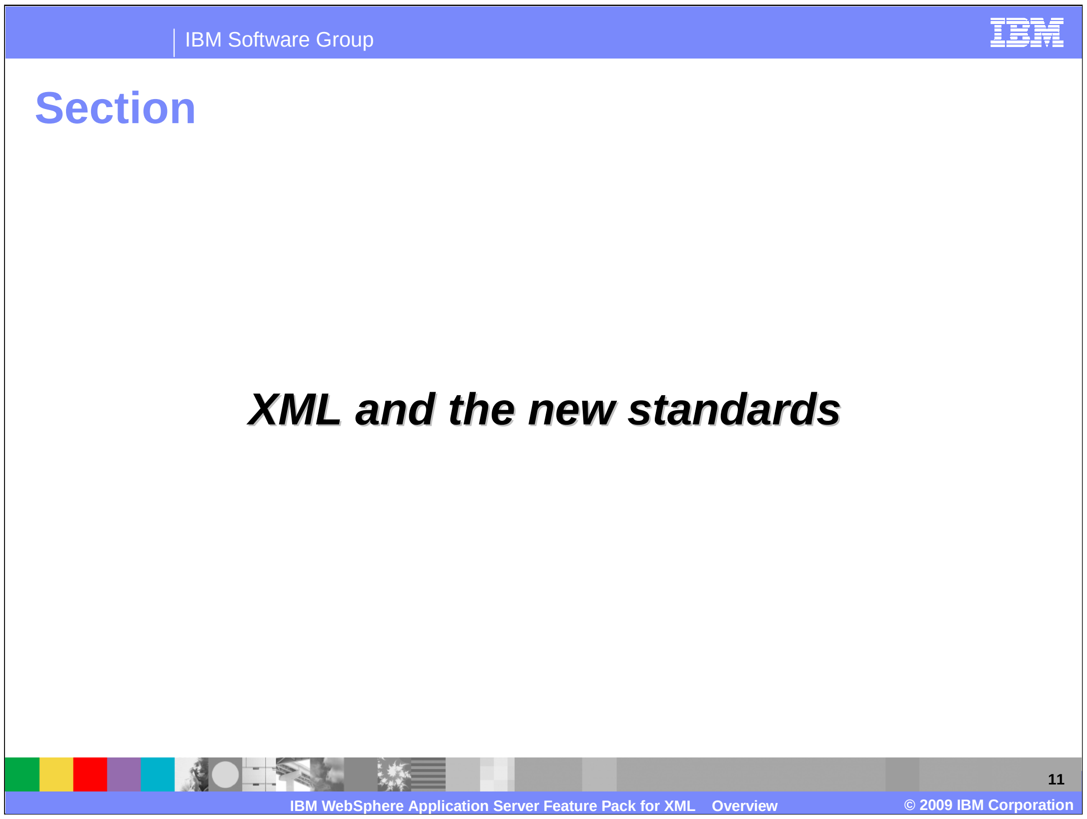
```
private static String XPATH_GET_ALL_BLOGS =
 "/atom:feed/atom:entry[substring-after(atom:id, '.blog-')]/(atom:id,atom:title)";
```

What needs to be done

1. Start at root
2. Look for all sub-elements that are "feed"'s
3. Look for all sub-elements that are "entry"'s
4. Look for all sub-elements that are "id"'s and see if they contain ".blog-"
5. Of those entries, return the "id" and the sub-element "title"

This slide compares the imperative versus declarative programming. The code goes through an atom feed looking for blog entries that have an ID that contains .blog-. It then returns the full ID and sub-element title. The steps are to start at the root, looking for all sub elements that are feeds and then the sub elements that are entries. Then look for all the sub-elements that are IDs and see if those IDs contain .blog-. Of those entries that do contain the .blog-, return the full ID and the sub-element title. The approach shown here is the DOM approach, the parser creates a tree of objects that represents the content and organization of data in the document. In this case, the tree exists in memory. The application code can then navigate through the tree to access the data it needs, and if appropriate, manipulate it. The declarative approach to do the same thing as the DOM approach is also shown on this slide. It is at the bottom of the slide. The same steps are taking in the imperative programming mode, where you first start at the root, indicated here with a 1. Then at 2, look for all sub-elements that are feeds. At 3, look for all the sub-elements that are entries, and then at 4 look for all the sub-elements that are IDs and see if they contain .blog-. Finally it returns the ID and the sub-element title.

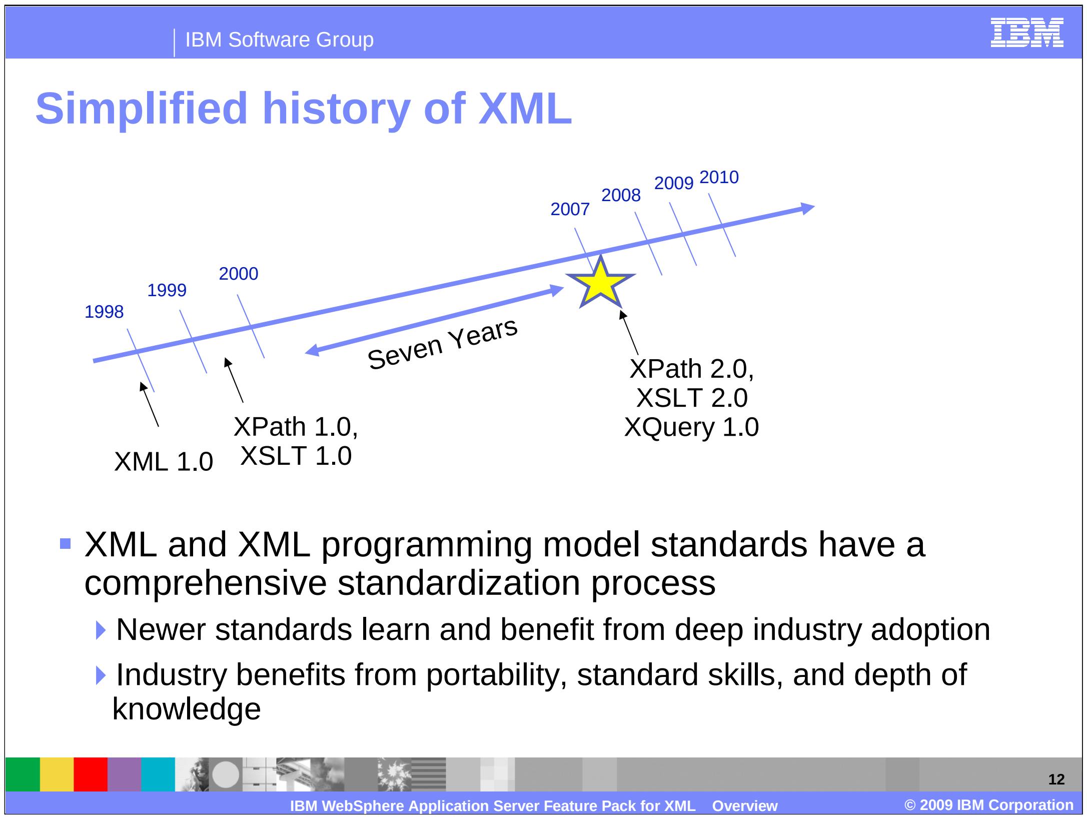
Note the differences between the Imperative and Declarative approach. The object oriented, or imperative, approach tells the runtime how to retrieve the data, while the XPath or declarative approach tells the runtime what data is needed. The DOM example creates many fine grained objects; this can cause performance issues. A more complex example can show more simplification by using XPath instead of DOM. Transformation of data as compared to using XSLT is even more complex due to DOM construction.



Section

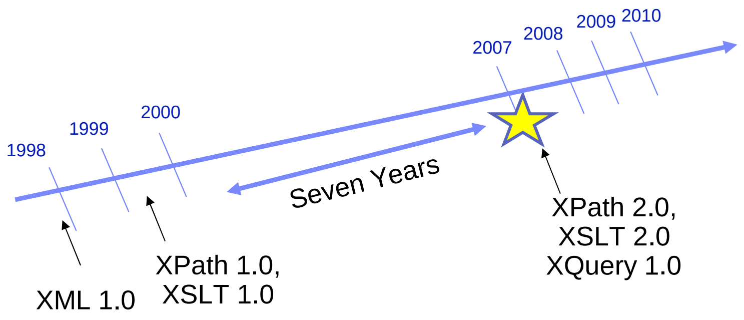
XML and the new standards

The next few slides will go over XML and the new standards.



IBM Software Group

Simplified history of XML

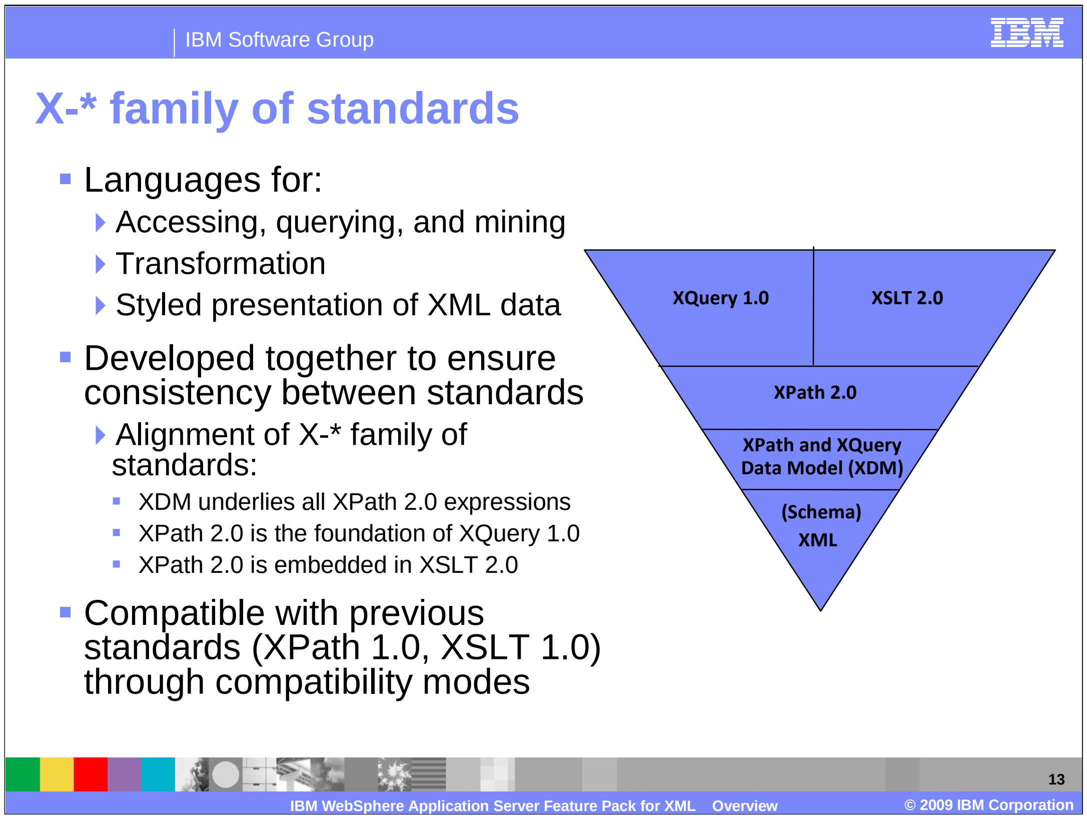


- XML and XML programming model standards have a comprehensive standardization process
  - Newer standards learn and benefit from deep industry adoption
  - Industry benefits from portability, standard skills, and depth of knowledge

12

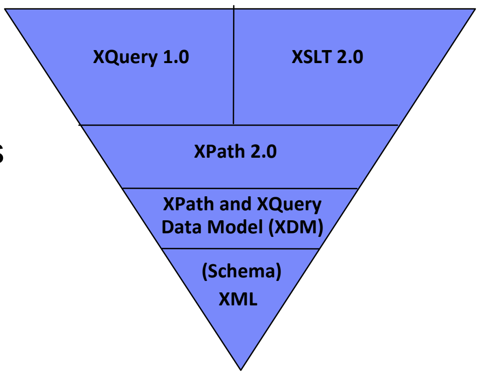
IBM WebSphere Application Server Feature Pack for XML Overview © 2009 IBM Corporation

XML and XML programming model standards have a comprehensive standardization process. Newer standards learn and benefit from deep industry adoption, and industry benefits from portability, standard skills, and depth of knowledge. These standards however have had a seven year gap. XPath 2.0, XSLT 2.0, and XQuery 1.0 were introduced in 2007.

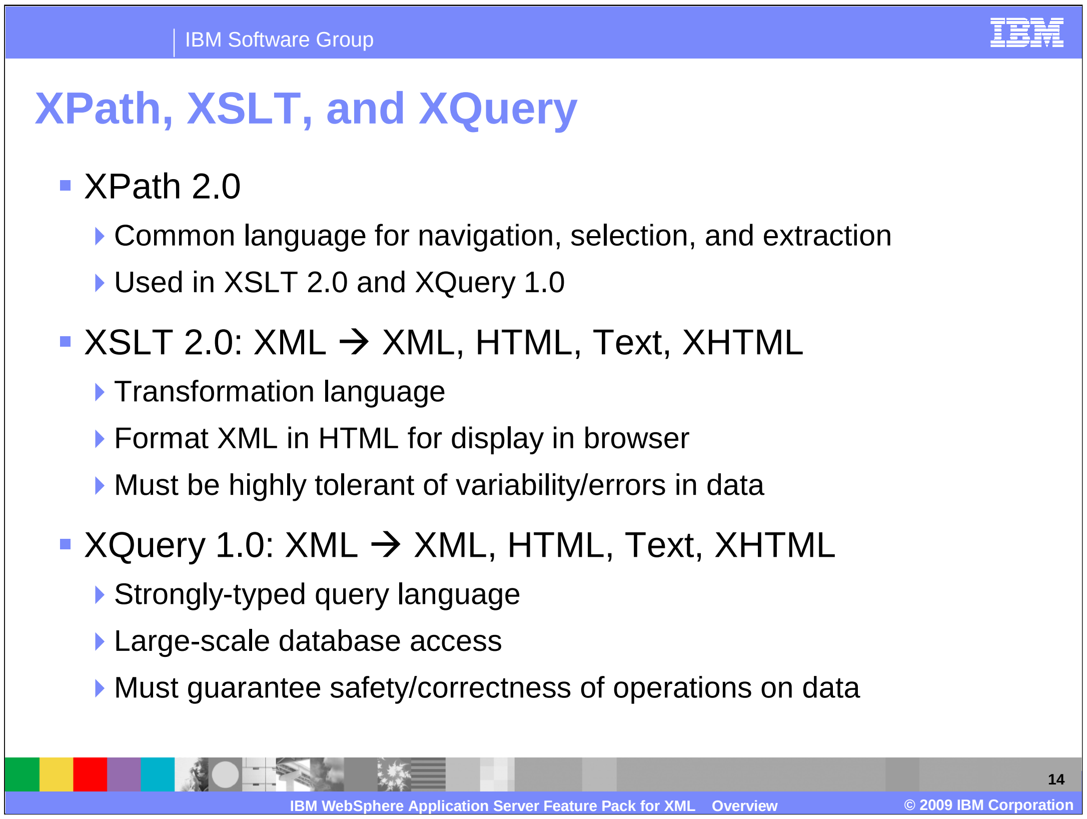


X-* family of standards

- Languages for:
  - Accessing, querying, and mining
  - Transformation
  - Styled presentation of XML data
- Developed together to ensure consistency between standards
  - Alignment of X-* family of standards:
    - XDM underlies all XPath 2.0 expressions
    - XPath 2.0 is the foundation of XQuery 1.0
    - XPath 2.0 is embedded in XSLT 2.0
- Compatible with previous standards (XPath 1.0, XSLT 1.0) through compatibility modes



This slide shows the alignment of XML and the different XML programming model standards. X* commonly refers to the different standards in the XML family. There are languages for accessing, querying, and mining; a language for transformation; and a language for styled presentation of XML data. The X-* standards were developed together to ensure consistency between the standards. The alignment of X-* family of standards starts with XML at the bottom of the pyramid. The XPath 2.0 and XQuery 1.0 Data Model referred to as XDM underlies all XPath 2.0 expressions allowing XPath 2.0, XSLT 2.0, and XQuery 1.0 to have a single common data model. XPath 2.0 is the foundation of XQuery 1.0, and XPath 2.0 is embedded in XSLT 2.0. These standards were developed with several years of experience of XPath 1.0 and XSLT 1.0 usage. They were developed with the idea of compatibility with the previous standards of XPath 1.0 and XSLT 1.0, and developed with other upcoming standards in mind.



XPath, XSLT, and XQuery

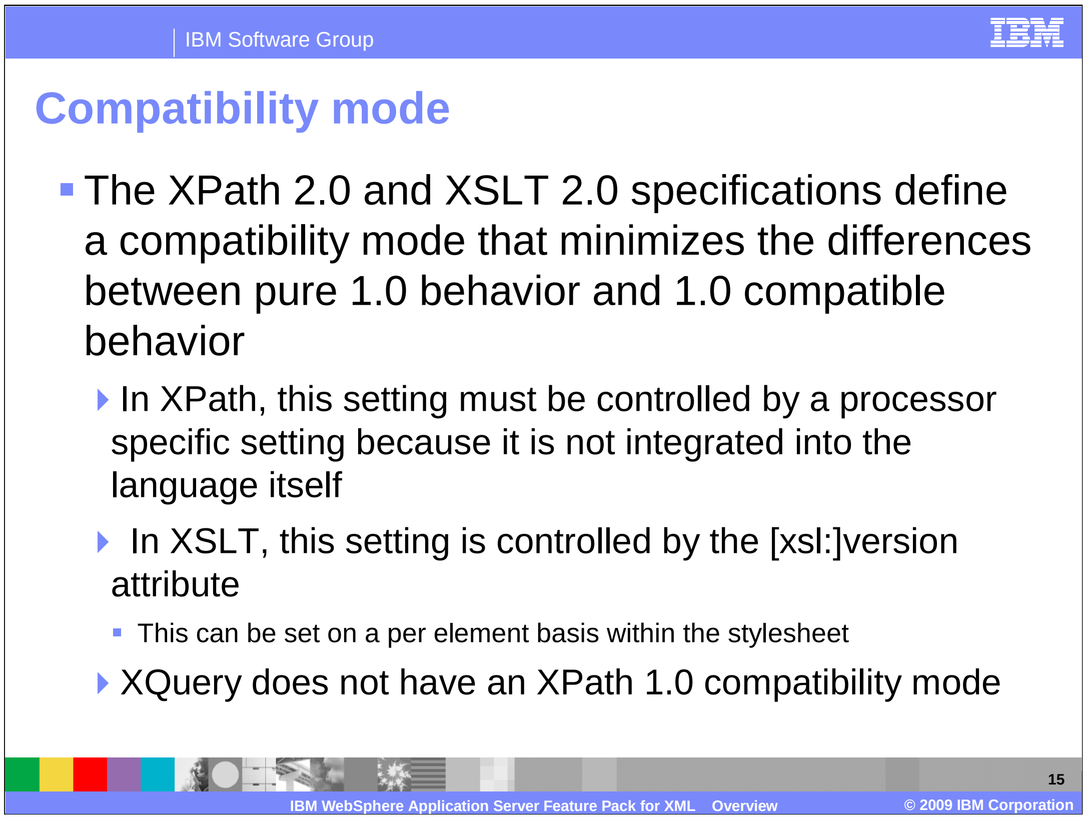
- XPath 2.0
  - Common language for navigation, selection, and extraction
  - Used in XSLT 2.0 and XQuery 1.0
- XSLT 2.0: XML → XML, HTML, Text, XHTML
  - Transformation language
  - Format XML in HTML for display in browser
  - Must be highly tolerant of variability/errors in data
- XQuery 1.0: XML → XML, HTML, Text, XHTML
  - Strongly-typed query language
  - Large-scale database access
  - Must guarantee safety/correctness of operations on data

There are three standard query languages for XML: XPath, XSLT, and the new XQuery. XSLT 2.0, XPath 2.0, and XQuery 1.0 are W3C recommendations.

XPath 2.0 is an expression language that allows the processing of values conforming to XQuery and XPath Data Model, abbreviated XDM. XPath is used primarily for selecting parts of an XML document and allows nodes to be selected by means of a hierarchic navigation path through the document tree. The result of an XPath expression can be a selection of nodes, value, or any sequence allowed by the data model. XPath 2.0 is a common language for navigation, selection, and projection that is used in XSLT, XQuery and several other languages.

An XSLT stylesheet contains instructions for transforming the contents of an XML document into another format. XSLT uses XPath to define parts of the source document that should match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document. XSLT is a loosely typed scripting language whose primary purpose is to transform XML documents into HTML for display in a browser. Like other browser-oriented technologies, XSLT is designed to be highly tolerant of variability and errors in input data. XQuery, however, is a strongly typed query language whose primary purpose is to support queries over large-scale XML databases. XQuery provides the means to extract and manipulate data from XML documents or any data source that can be viewed as XML. It uses XPath expression syntax to address specific parts of an XML document. Like other database query languages, XQuery must guarantee the safety and correctness of operations on data. For example, it is undesirable to process 2GB of data before discovering that an expected integer value is actually a date.

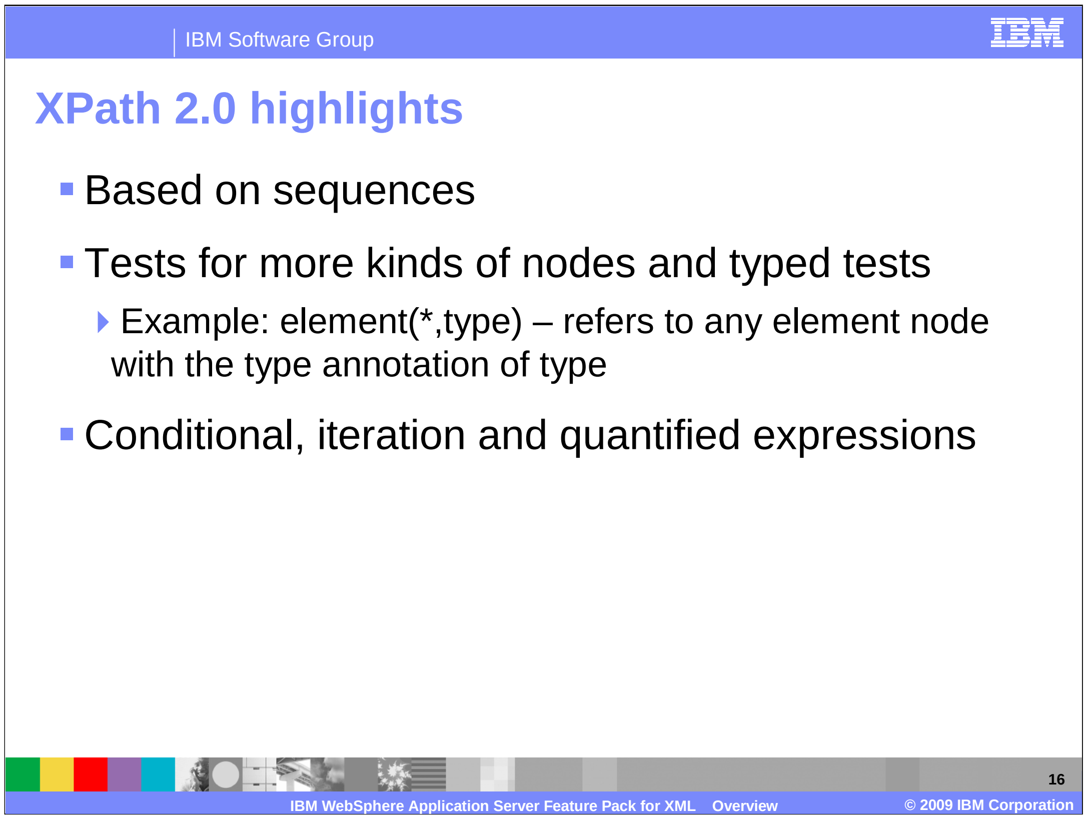
The differences between XSLT and XQuery, however, are primarily historical and somewhat artificial. Over time, both languages can serve the needs of many application domains. As a common sublanguage of both XSLT and XQuery, XPath 2.0 was designed to have a consistent syntax and semantics so that programmers can use it in both host languages.



Compatibility mode

- The XPath 2.0 and XSLT 2.0 specifications define a compatibility mode that minimizes the differences between pure 1.0 behavior and 1.0 compatible behavior
  - In XPath, this setting must be controlled by a processor specific setting because it is not integrated into the language itself
  - In XSLT, this setting is controlled by the [xsl:]version attribute
    - This can be set on a per element basis within the stylesheet
  - XQuery does not have an XPath 1.0 compatibility mode

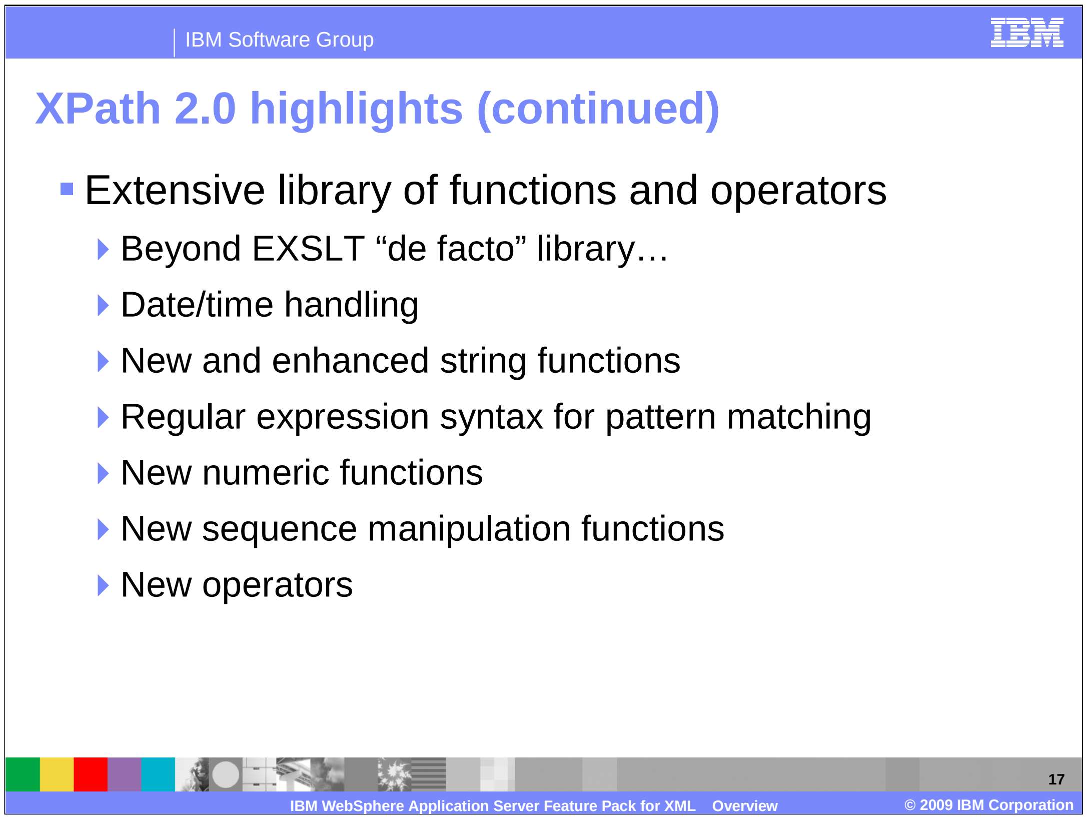
It is great to have new functions and features, but what about existing XPath 1.0 and XSLT 1.0 applications? The XPath 2.0 and XSLT 2.0 specifications define a compatibility mode that minimizes the differences between pure 1.0 behavior and 1.0 compatible behavior. In XPath, this setting must be controlled by a processor specific setting because it is not integrated into the language itself. In XSLT, this setting is controlled by the xsl:version attribute, which can be set on a per-element basis within the style sheet. The xsl:use-when attribute can be used to have different behavior on different versions. Some of these items defined in compatibility mode can be fixed in your 1.0 code to avoid any difference; some incur more risk (for example documents that have been schema-validated). XQuery does not have an XPath 1.0 compatibility mode.



XPath 2.0 highlights

- Based on sequences
- Tests for more kinds of nodes and typed tests
  - Example: element(*,type) – refers to any element node with the type annotation of type
- Conditional, iteration and quantified expressions

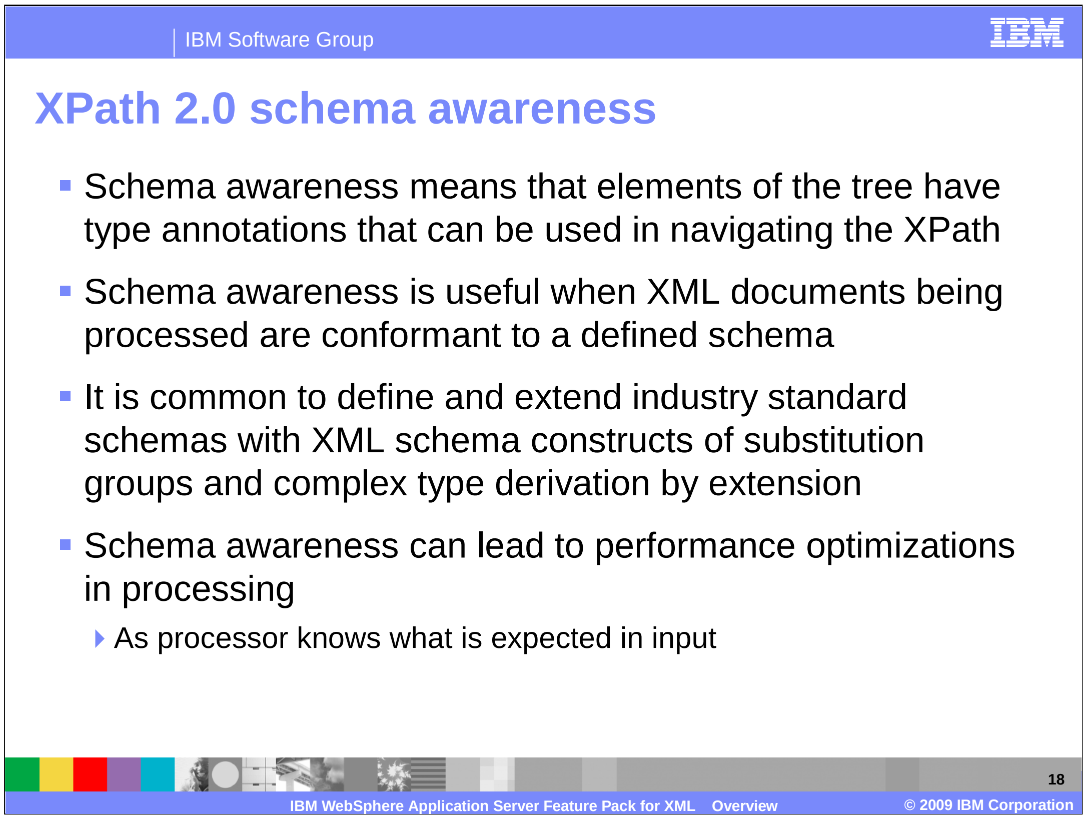
The next two slides briefly touch upon the XPath 2.0 highlights. To start, XPath 2.0 is based on sequences. A sequence is an ordered collection of zero or more items. An item is either a node or an atomic value. XPath 2.0 contains more tests for more kinds of nodes and typed tests, for example element() – refers to any element node. element(*, addressType) – refers to any element node with the type annotation addressType. element(shipto, addressType) – refers to an element node with the name shipto and with type annotation addressType. XPath 2.0 also contains conditional, iteration and quantified expressions such as if, for, some and every keywords.



XPath 2.0 highlights (continued)

- Extensive library of functions and operators
  - Beyond EXSLT “de facto” library…
  - Date/time handling
  - New and enhanced string functions
  - Regular expression syntax for pattern matching
  - New numeric functions
  - New sequence manipulation functions
  - New operators

XPath 2.0 has an extensive library of functions and operators. There are 24 new functions and 44 new operators that deal with date and time handling, plus constructor functions. These functions include current date, time, and datetime functions to enable timestamping of output. There are some new and enhanced string functions and new collation argument on many string manipulation functions. Added in XPath 2.0 is regular expression syntax for pattern matching, and new numeric functions such as floor, ceiling, round, and abs. XPath 2.0 also contains new sequence manipulation functions such as reverse, subsequence, remove, and index-of and new operators such as intersect and except.



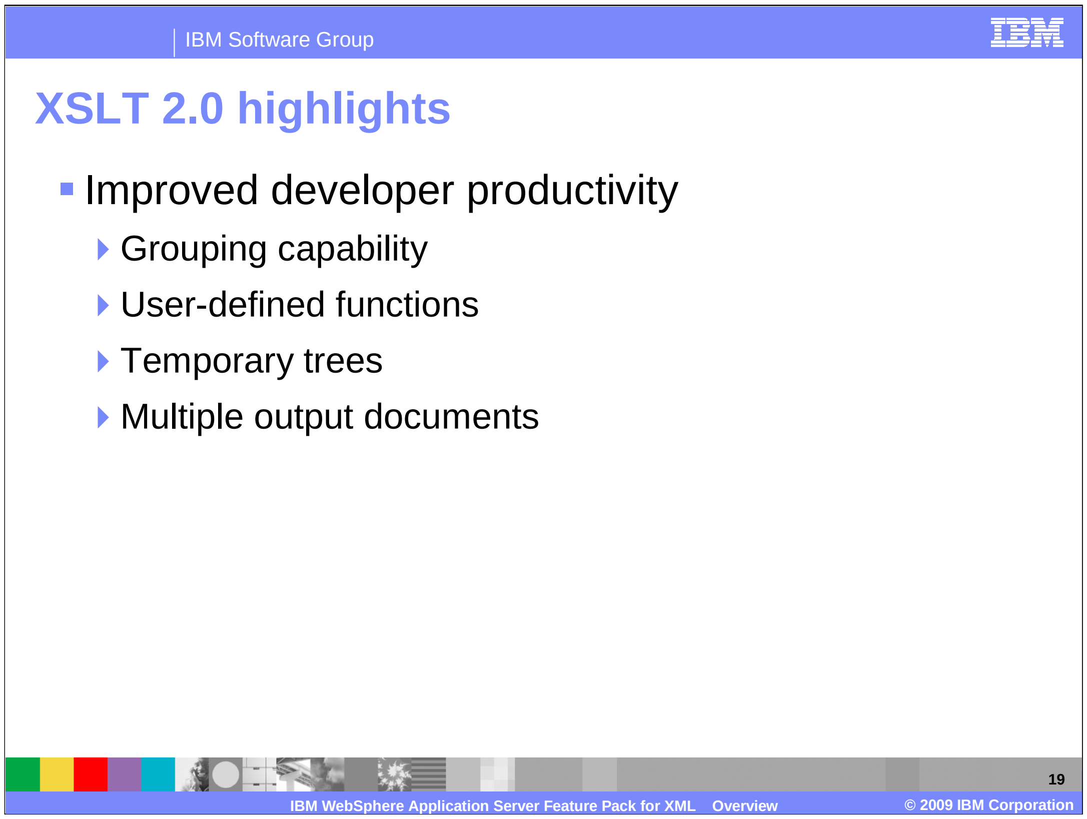
XPath 2.0 schema awareness

- Schema awareness means that elements of the tree have type annotations that can be used in navigating the XPath
- Schema awareness is useful when XML documents being processed are conformant to a defined schema
- It is common to define and extend industry standard schemas with XML schema constructs of substitution groups and complex type derivation by extension
- Schema awareness can lead to performance optimizations in processing
  - As processor knows what is expected in input

Schema awareness means that elements of the tree have type annotations that can be used in navigating the XPath. Schema awareness is useful when XML documents being processed are conformant to a defined schema, which is common in many vertical industries. Schema awareness for built-in types, such as Instance of, cast as, castable as, treat as, and constructor functions is required. Schema awareness for user-defined types (through XML schema definition) is optional. Full validation of the input is an optional addition that is not required by the specification.

It is common to define and extend industry standard schemas with XML schema constructs of substitution groups and complex type derivation by extension. In XPath 1.0, you have to search for every possible substitution in the group and every possible complex type derivation type. This is impossible if you are writing an application that processes data on the group and base types without knowledge of how eventual users might extend the groups and types. In XPath 2.0, due to its schema awareness, you can search for the substitution group head and complex type base type. This will work even if users later add to the substitution group and extend the base complex type.

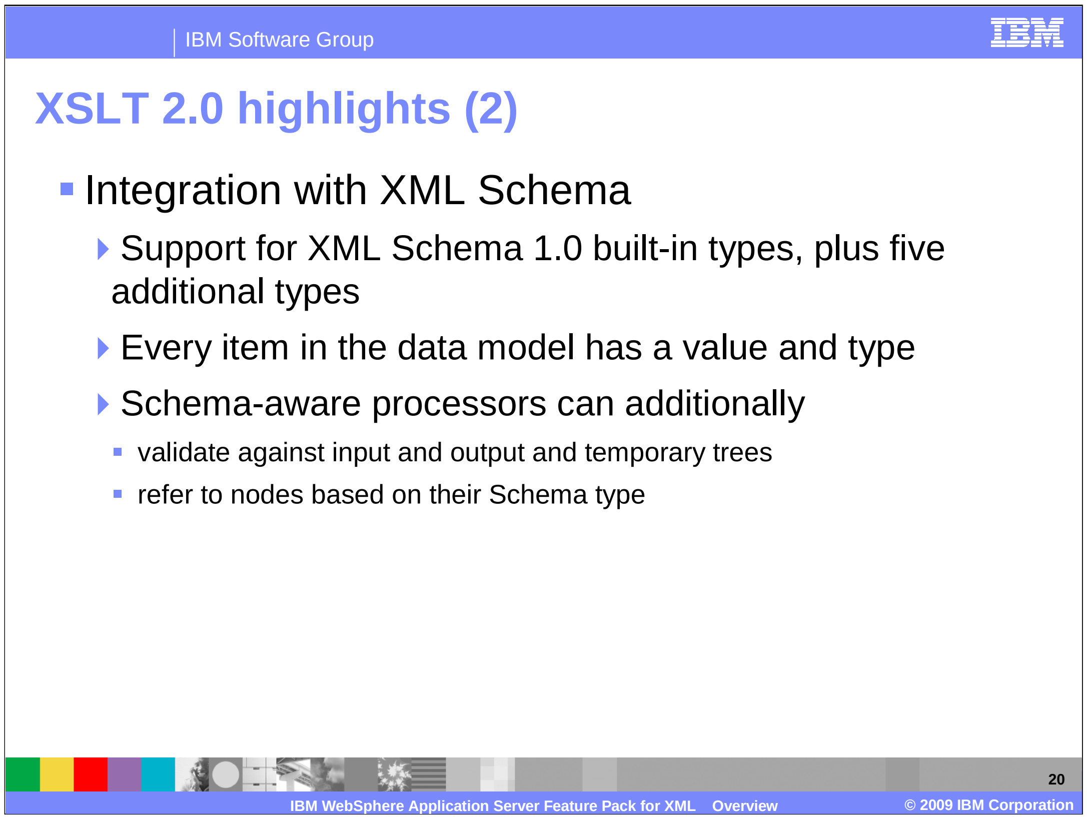
Schema awareness can lead to performance optimizations in processing as the processor knows what is expected in input.



XSLT 2.0 highlights

- Improved developer productivity
  - Grouping capability
  - User-defined functions
  - Temporary trees
  - Multiple output documents

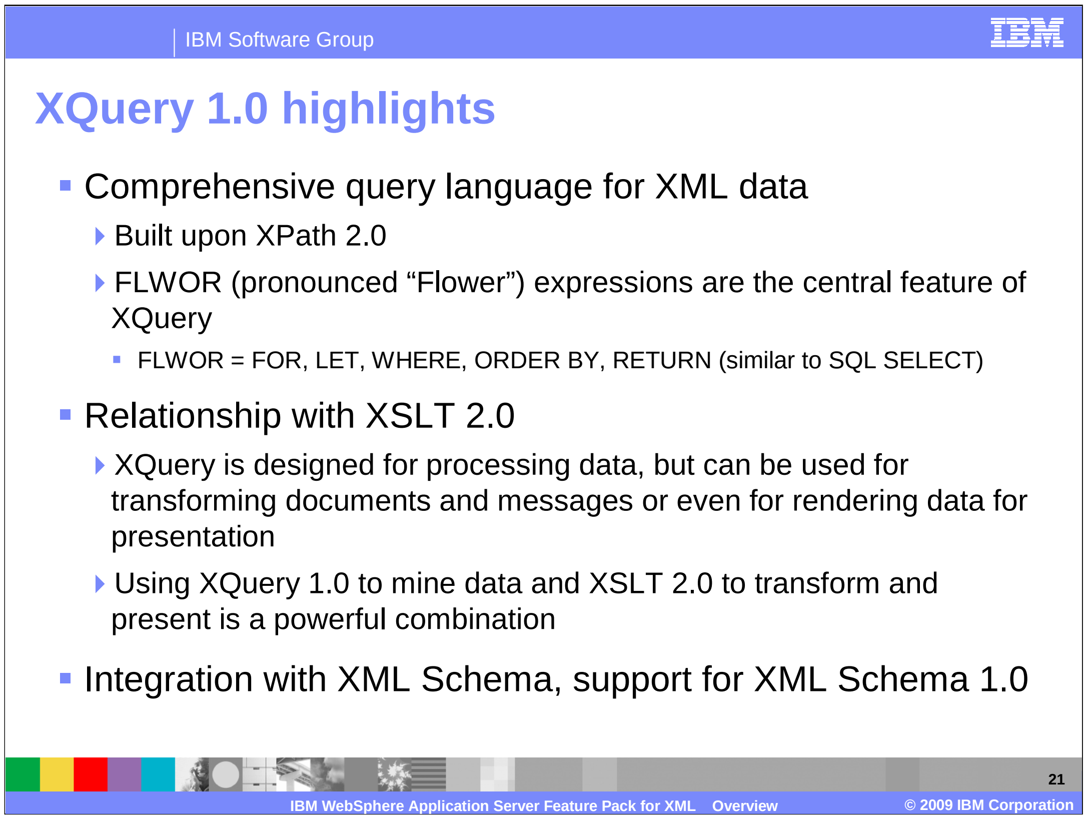
The next two slides briefly touch upon the XSLT 2.0 highlights. XSLT 2.0 includes improved developer productivity with new capabilities. It contains a grouping capability with the introduction of xsl:for-each-group and supporting functions. This vastly simplifies style sheets and enables processor performance optimizations. XSLT 2.0 also allows user-defined functions allowing you to write extension functions in XSLT rather than using named templates or resorting to another language. It includes the capability of temporary trees providing standard access to intermediate results. There is also support for multiple output documents where one style sheet can serialize to multiple destinations plus many more capabilities to improve developer productivity.



XSLT 2.0 highlights (2)

- Integration with XML Schema
  - Support for XML Schema 1.0 built-in types, plus five additional types
  - Every item in the data model has a value and type
  - Schema-aware processors can additionally
    - validate against input and output and temporary trees
    - refer to nodes based on their Schema type

XSLT 2.0 is integrated with XML Schema, it has support for XML Schema 1.0 built-in types, plus five additional types. Every item in the data model has a value and type. The XSLT 2.0 specification also talks about an optional feature of schema awareness. Schema-aware processors can additionally validate against input and output and temporary trees and refer to nodes based on their Schema type.


IBM Software Group

XQuery 1.0 highlights

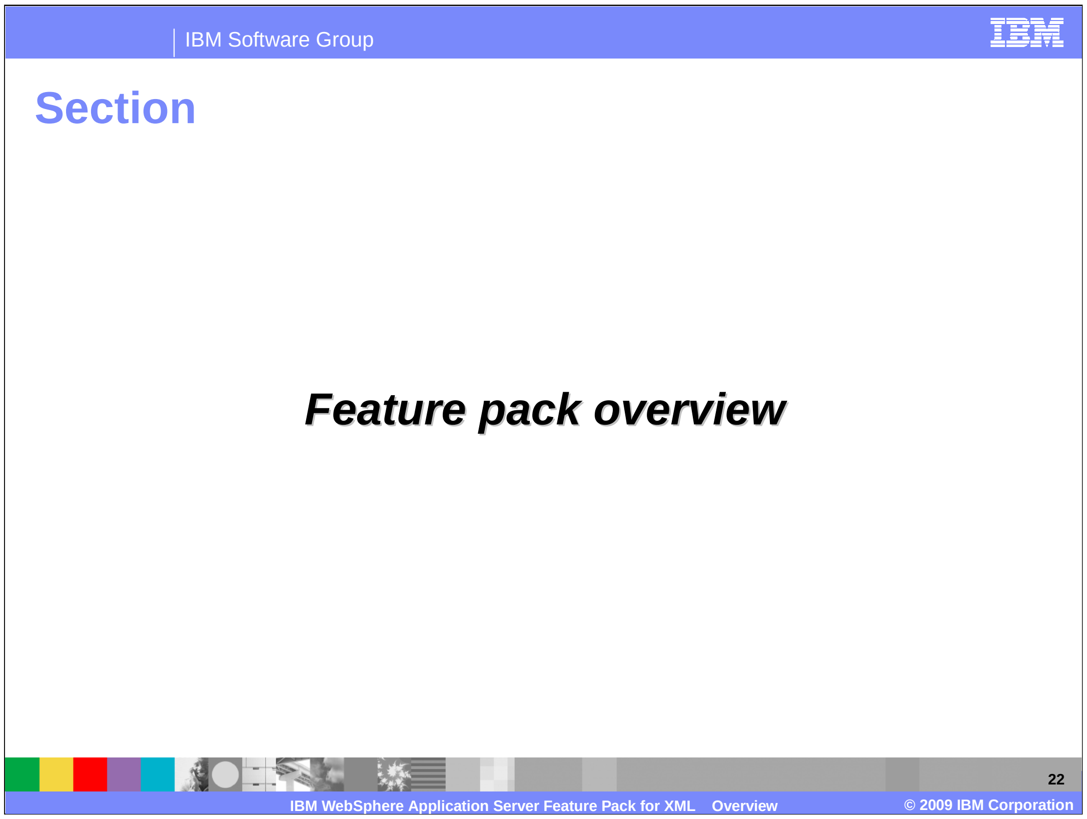
- Comprehensive query language for XML data
  - Built upon XPath 2.0
  - FLWOR (pronounced "Flower") expressions are the central feature of XQuery
    - FLWOR = FOR, LET, WHERE, ORDER BY, RETURN (similar to SQL SELECT)
- Relationship with XSLT 2.0
  - XQuery is designed for processing data, but can be used for transforming documents and messages or even for rendering data for presentation
  - Using XQuery 1.0 to mine data and XSLT 2.0 to transform and present is a powerful combination
- Integration with XML Schema, support for XML Schema 1.0

21

IBM WebSphere Application Server Feature Pack for XML Overview © 2009 IBM Corporation

Here are some XQuery 2.0 highlights. XQuery provides the means to extract and manipulate data from XML documents or any data source that can be viewed as XML. It uses XPath expression syntax to address specific parts of an XML document. XQuery contains an SQL-like "FLWOR expression“. FLWOR expressions are the central feature of XQuery. FLWOR stands for FOR, LET, WHERE, ORDER BY, RETURN. The FLOWR expressions can be nested within any other XQuery expression, including as arguments to functions.

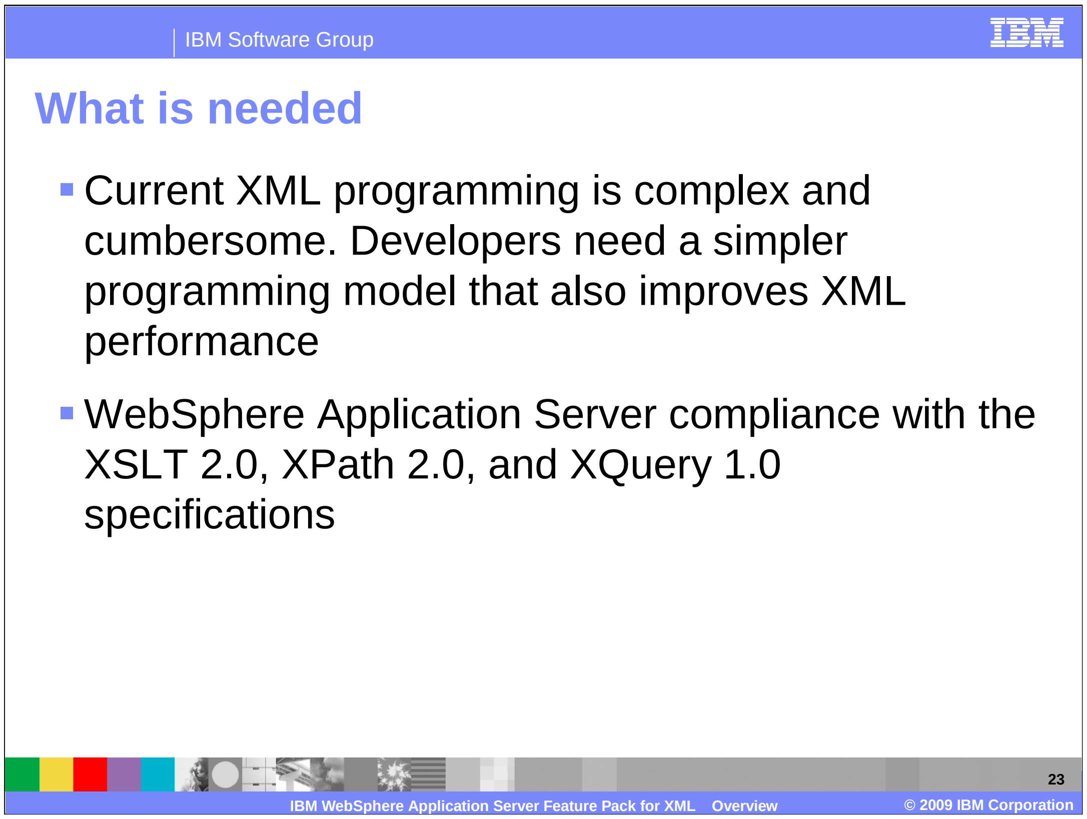
XQuery is designed for processing data, but can be used for transforming documents and messages or even for rendering data for presentation. A powerful combination is to use XQuery 1.0 to mine data and XSLT 2.0 to transform and present. XQuery includes integration with XML schema; it has support for XML Schema 1.0.



Section

Feature pack overview

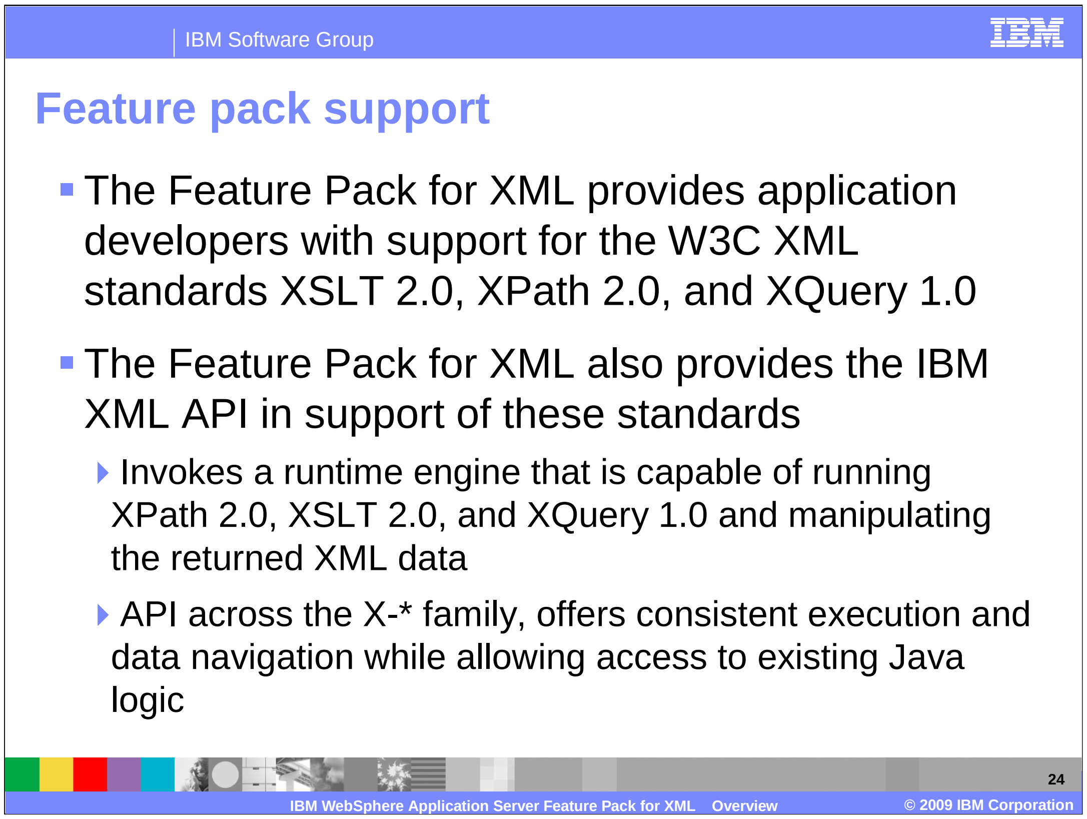
This section covers the reasons for, and features of, the IBM WebSphere Application Server Feature Pack for XML.



What is needed

- Current XML programming is complex and cumbersome. Developers need a simpler programming model that also improves XML performance
- WebSphere Application Server compliance with the XSLT 2.0, XPath 2.0, and XQuery 1.0 specifications

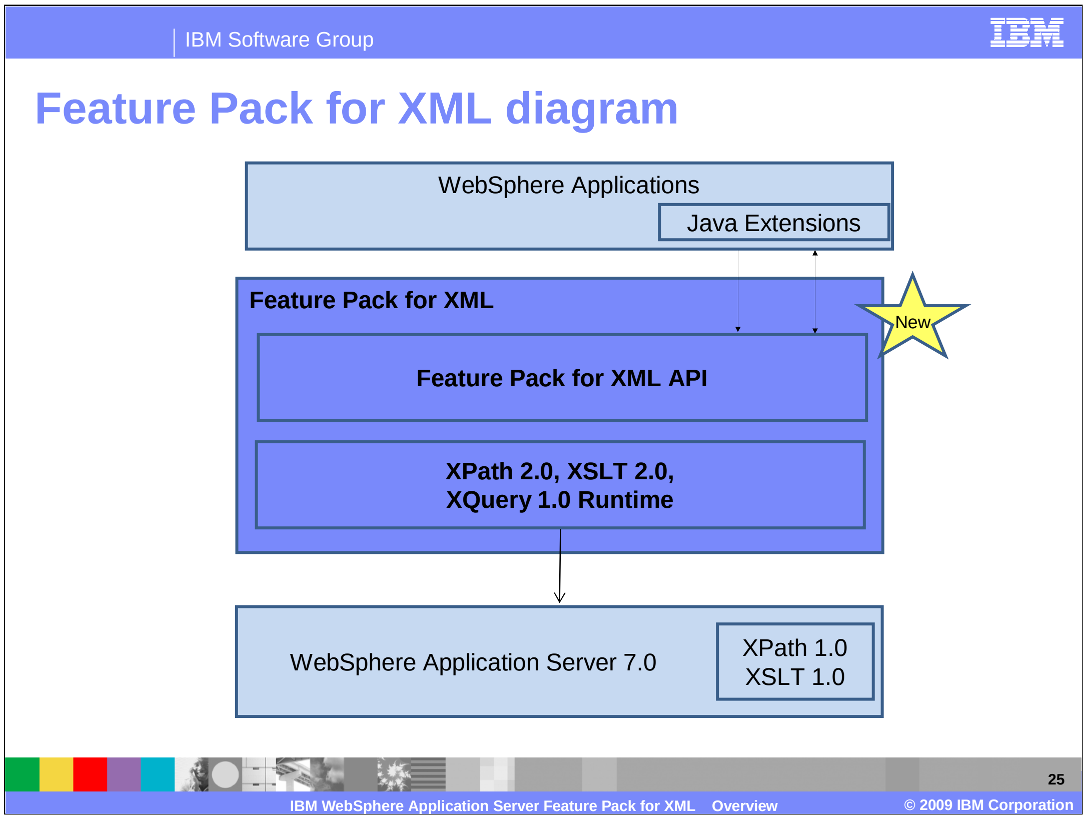
The current XML programming is complex and cumbersome and developers are in need of a simpler programming model that also improves XML performance. What is needed is WebSphere Application Server compliance with the XSLT 2.0, XPath 2.0, and XQuery 1.0 specifications. This leads to the IBM WebSphere Application Server Feature Pack for XML.



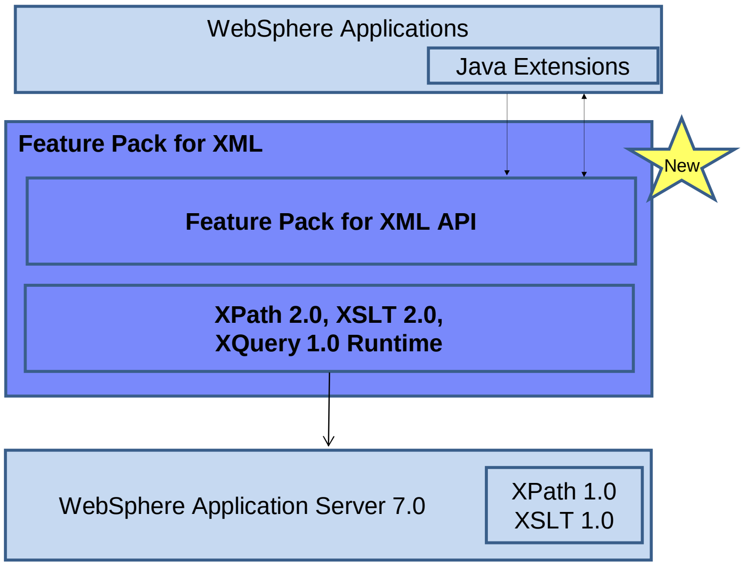
Feature pack support

- The Feature Pack for XML provides application developers with support for the W3C XML standards XSLT 2.0, XPath 2.0, and XQuery 1.0
- The Feature Pack for XML also provides the IBM XML API in support of these standards
  - Invokes a runtime engine that is capable of running XPath 2.0, XSLT 2.0, and XQuery 1.0 and manipulating the returned XML data
  - API across the X-* family, offers consistent execution and data navigation while allowing access to existing Java logic

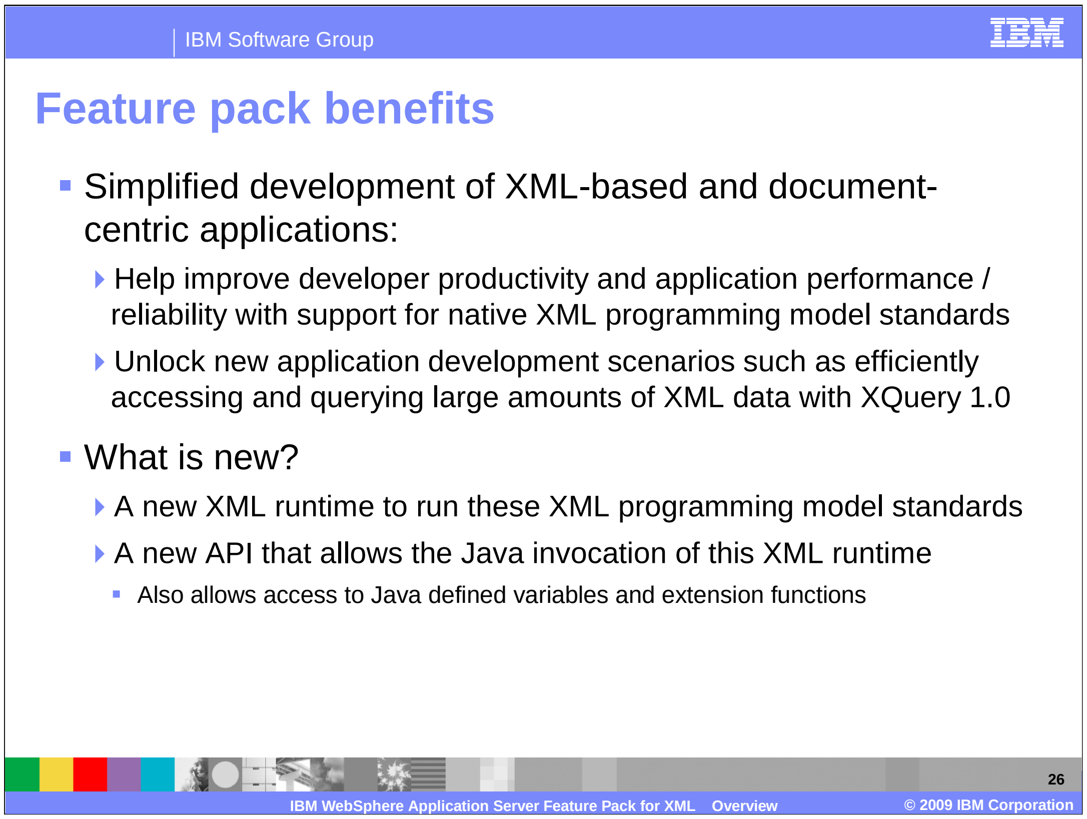
The feature pack for XML delivers critical technology that enables adoption of key XML standards and principles. XML-structured data has become the predominant format for data interchange. XML data is navigated, queried, or transformed in almost every existing WebSphere application. Since first being standardized in 1999, XML usage in application-development environments has grown significantly to include many scenarios. WebSphere Application Server is a leading platform for the latest application development standards that matter, including XML. The feature pack for XML provides application developers with support for the W3C XML standards XSLT 2.0, XPath 2.0, and XQuery 1.0. The feature pack for XML also provides the IBM XML API in support of these standards. The API invokes a runtime engine that is capable of running XPath 2.0, XSLT 2.0, and XQuery 1.0 and manipulating the returned XML data. The XML API across the X-* family offers consistent execution and data navigation while allowing access to existing Java logic.



Feature Pack for XML diagram



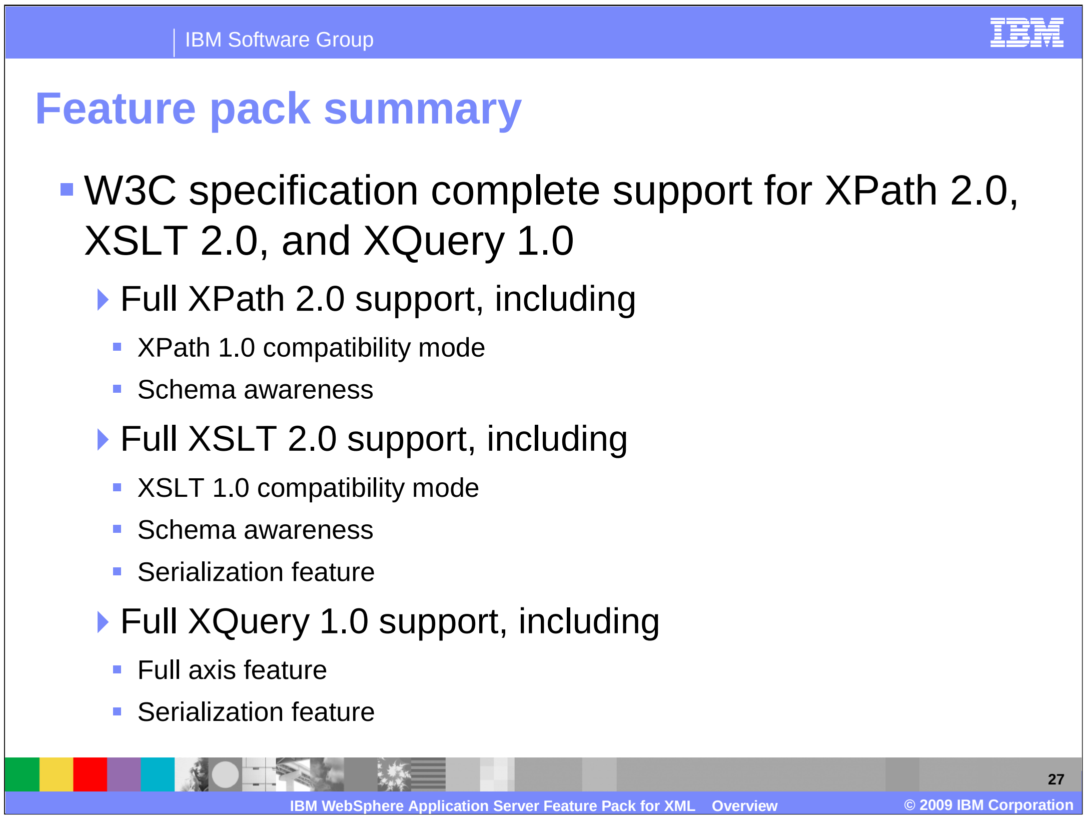
The slide shows the layout of the feature pack for XML and how the XML API and runtime lie between the WebSphere Application Server V .0 and WebSphere applications.



Feature pack benefits

- Simplified development of XML-based and document-centric applications:
  - Help improve developer productivity and application performance / reliability with support for native XML programming model standards
  - Unlock new application development scenarios such as efficiently accessing and querying large amounts of XML data with XQuery 1.0
- What is new?
  - A new XML runtime to run these XML programming model standards
  - A new API that allows the Java invocation of this XML runtime
    - Also allows access to Java defined variables and extension functions

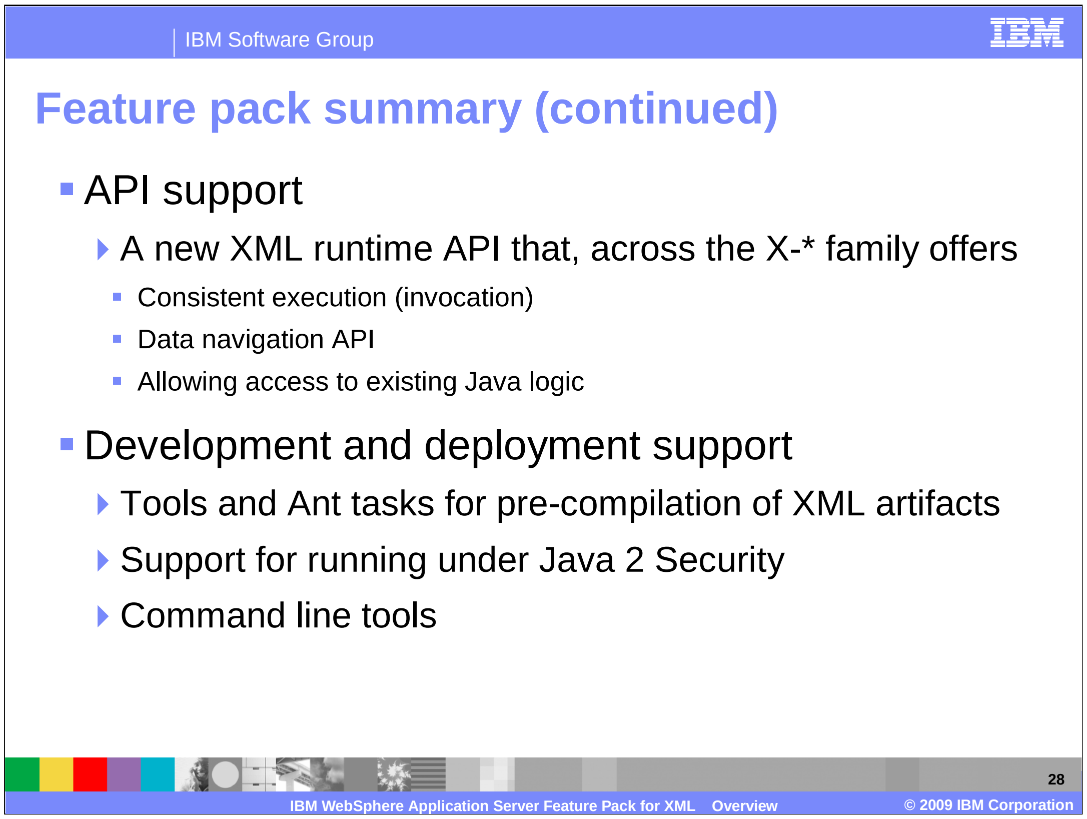
The feature pack allows WebSphere Application Server compliance with the XSLT 2.0, XPath 2.0, and XQuery 1.0 specifications. For WebSphere applications that currently work with XML through object binding programming models such as DOM and JAXB, the feature pack will improve performance, fidelity, and ease of use while leveraging standards-based skill sets. For WebSphere applications that work with XML through existing XML programming models XPath 1.0 and XSLT 1.0, the feature pack simplifies existing expressions and style sheets, and achieves scenarios not possible before by use of new functionality. There is also the ability to query large amounts of data stored in XML outside of a database with XQuery 1.0 and new features introduced in the W3C specifications to address previous shortcomings. The feature pack provides a new XML runtime to run these XML programming model standards and a new API that allows the Java invocation of this XML runtime. The API also allows access to Java defined variables and extension functions.



Feature pack summary

- W3C specification complete support for XPath 2.0, XSLT 2.0, and XQuery 1.0
  - Full XPath 2.0 support, including
    - XPath 1.0 compatibility mode
    - Schema awareness
  - Full XSLT 2.0 support, including
    - XSLT 1.0 compatibility mode
    - Schema awareness
    - Serialization feature
  - Full XQuery 1.0 support, including
    - Full axis feature
    - Serialization feature

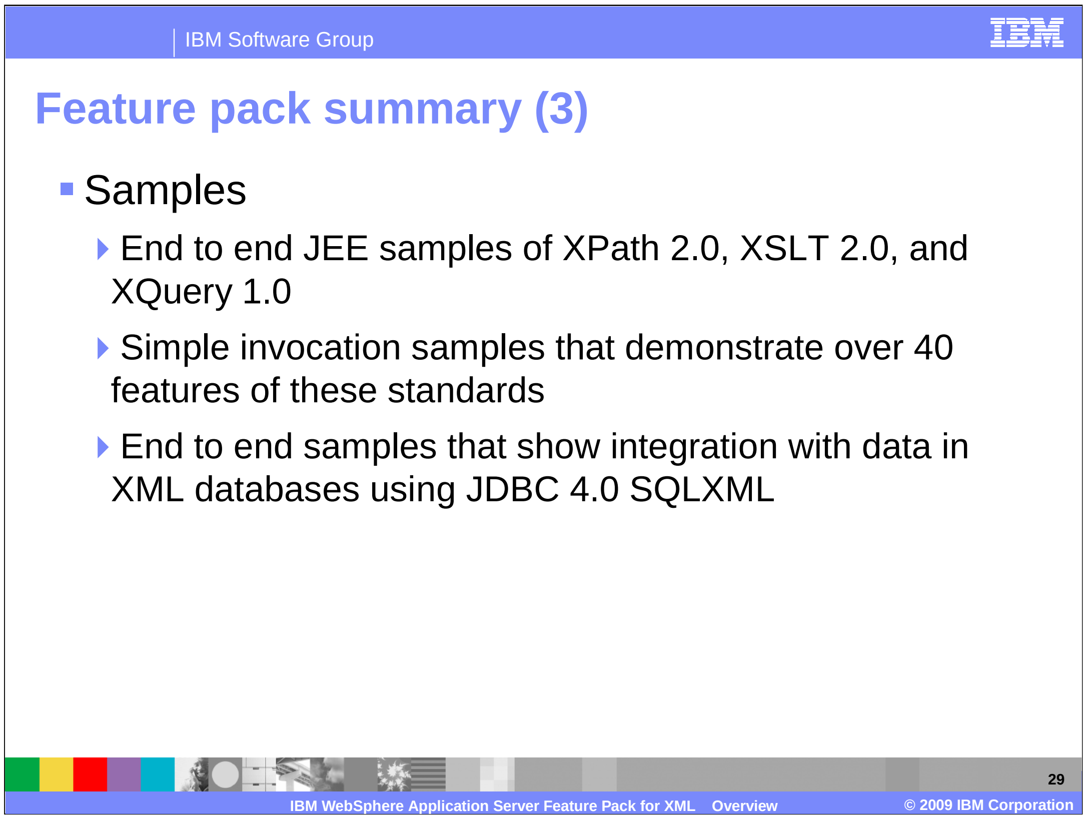
The feature pack includes W3C specification support for XPath 2.0, XSLT 2.0, and XQuery 1.0. The feature pack has full XPath 2.0 support, including XPath 1.0 compatibility mode and schema awareness. The feature pack has full XSLT 2.0 support, including XSLT 1.0 compatibility mode, schema awareness, and a serialization feature. The feature pack has full XQuery 1.0 support, including full axis and serialization features.



Feature pack summary (continued)

- API support
  - A new XML runtime API that, across the X-* family offers
    - Consistent execution (invocation)
    - Data navigation API
    - Allowing access to existing Java logic
- Development and deployment support
  - Tools and Ant tasks for pre-compilation of XML artifacts
  - Support for running under Java 2 Security
  - Command line tools

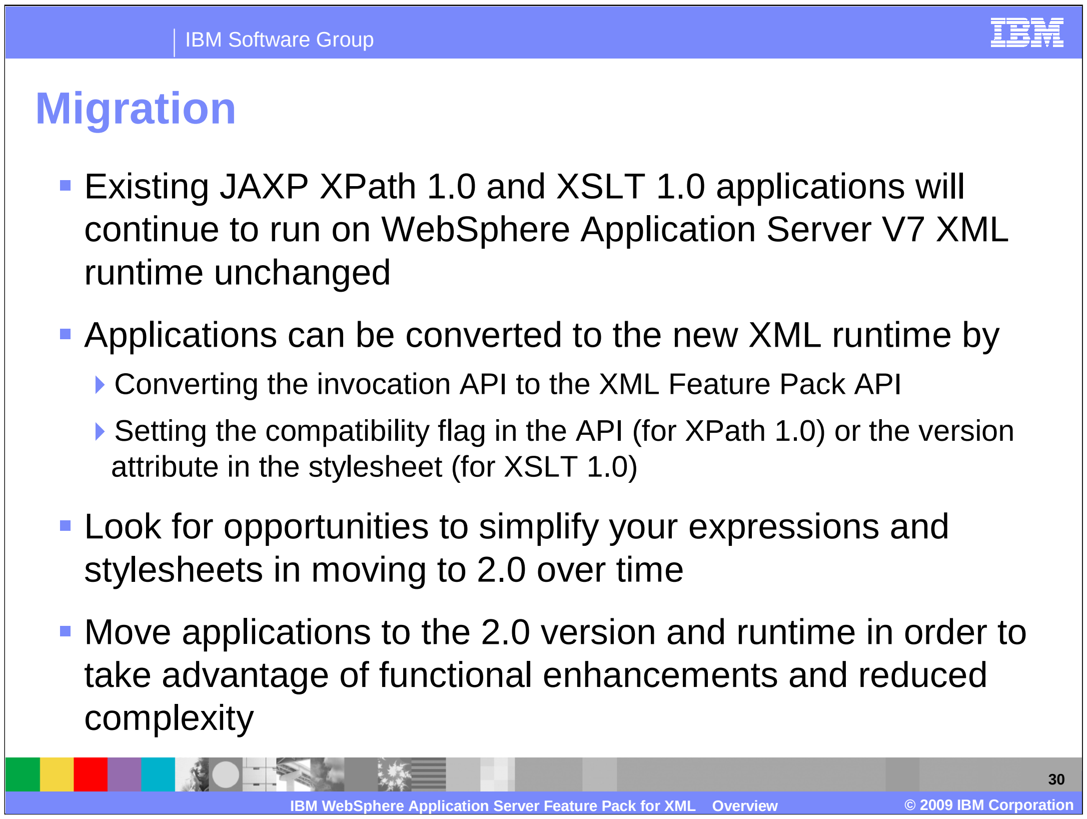
The feature pack for XML provides the IBM XML API in support of XML standards of XPath 2.0, XSLT 2.0, and XPath 1.0. The API offers a consistent invocation and data navigation API while allowing access to existing Java logic. The feature pack includes a runtime engine that is capable of running XPath 2.0, XSLT 2.0, and XQuery 1.0 and manipulating the returned XML data. The feature pack includes development and deployment support including tools and Ant tasks for pre-compilation of XML artifacts and support for running under Java 2 Security. There are also command line tools.



Feature pack summary (3)

- Samples
  - End to end JEE samples of XPath 2.0, XSLT 2.0, and XQuery 1.0
  - Simple invocation samples that demonstrate over 40 features of these standards
  - End to end samples that show integration with data in XML databases using JDBC 4.0 SQLXML

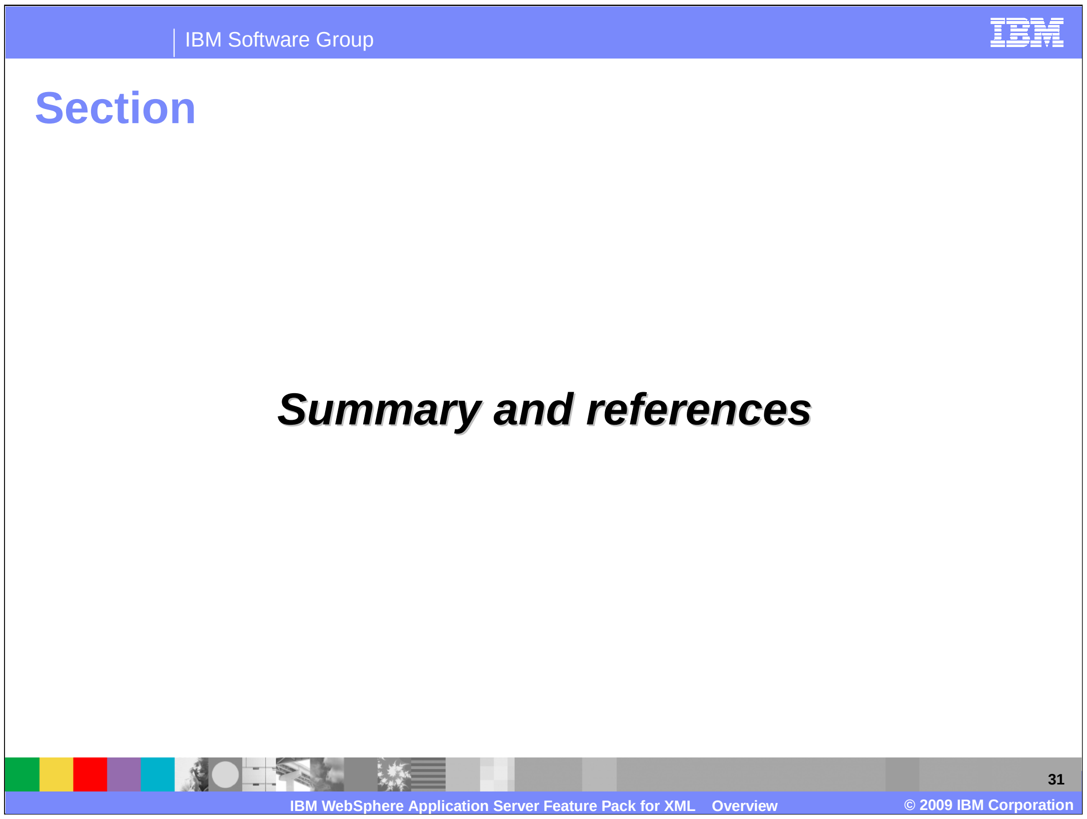
The feature pack also includes lots of samples including end-to-end JEE samples of XPath 2.0, XSLT 2.0, and XQuery 1.0 and simple invocation samples that demonstrate over 40 features of these standards. There are also end-to-end samples that show integration with data in XML databases using JDBC 4.0 SQLXML.



Migration

- Existing JAXP XPath 1.0 and XSLT 1.0 applications will continue to run on WebSphere Application Server V7 XML runtime unchanged
- Applications can be converted to the new XML runtime by
  - Converting the invocation API to the XML Feature Pack API
  - Setting the compatibility flag in the API (for XPath 1.0) or the version attribute in the stylesheet (for XSLT 1.0)
- Look for opportunities to simplify your expressions and stylesheets in moving to 2.0 over time
- Move applications to the 2.0 version and runtime in order to take advantage of functional enhancements and reduced complexity

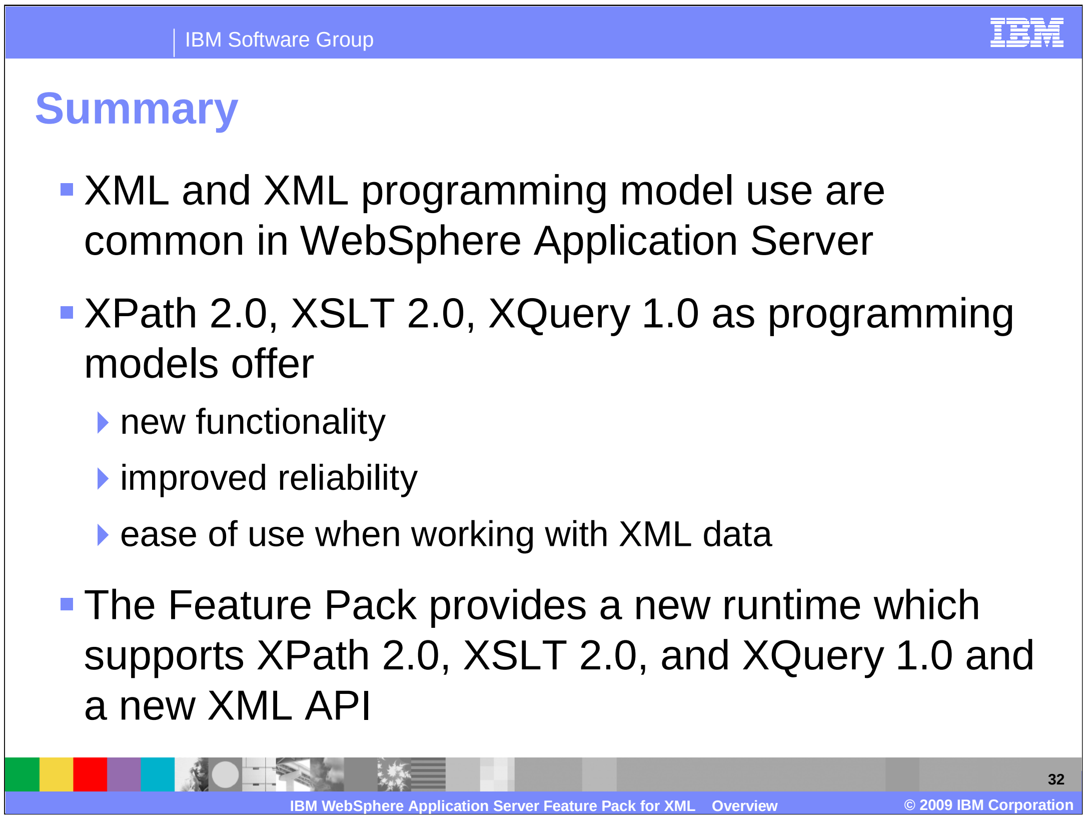
Existing JAXP XPath 1.0 and XSLT 1.0 applications will continue to run on WebSphere Application Server V7 XML runtime unchanged. Applications can be converted to the new XML runtime by first converting the invocation API to the XML feature pack API. The ease of conversion is dependent on depth of JAXP usage. To convert the application you will also use the compatibility mode that minimizes the differences between pure 1.0 behavior and 1.0 compatible behavior. Compatibility mode, as defined by the specification, handles most incompatibilities between 1.0 and 2.0. When migrating applications from older standards to newer standards, change from running on 1.0 processors to running in 1.0 compatible mode on 2.0 processors and fix any issues. Over time, look for opportunities to simplify your expressions and style sheets in moving to 2.0 as you will want to take advantage of functional enhancements and reduce complexity. For new application, start with the newer 2.0 standards.



Section

Summary and references

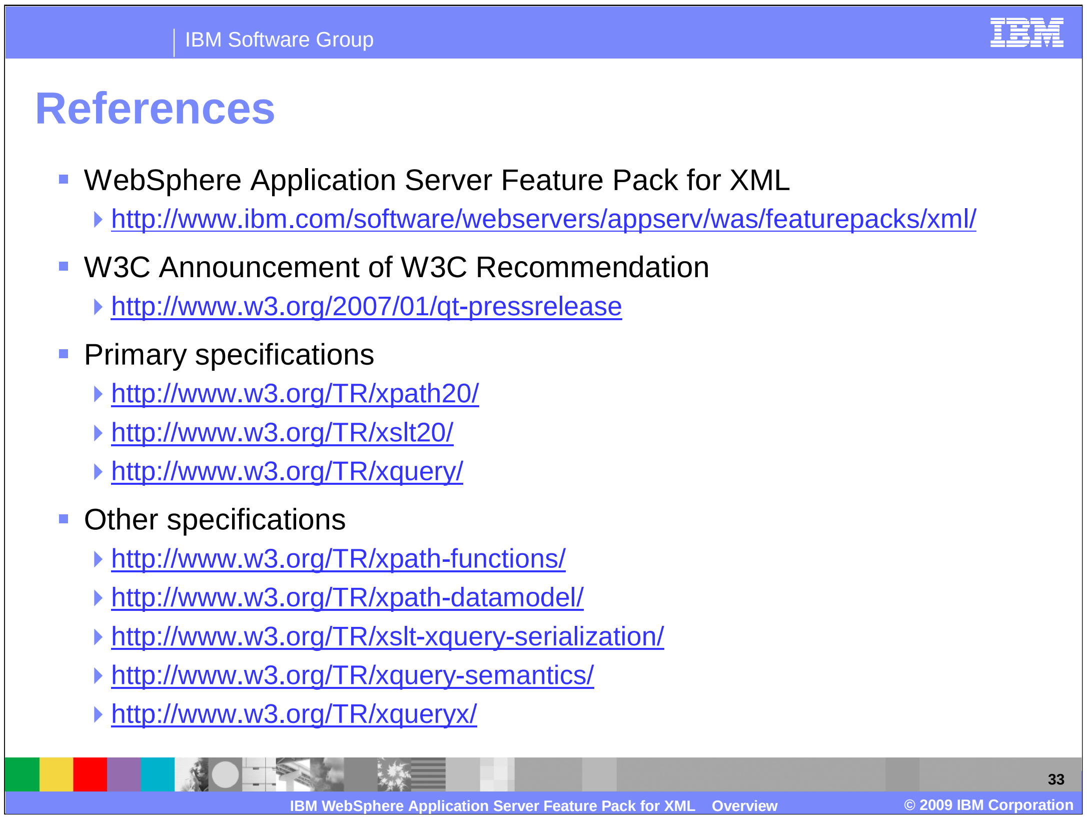
The next slides cover the summary and references.



Summary

- XML and XML programming model use are common in WebSphere Application Server
- XPath 2.0, XSLT 2.0, XQuery 1.0 as programming models offer
  - new functionality
  - improved reliability
  - ease of use when working with XML data
- The Feature Pack provides a new runtime which supports XPath 2.0, XSLT 2.0, and XQuery 1.0 and a new XML API

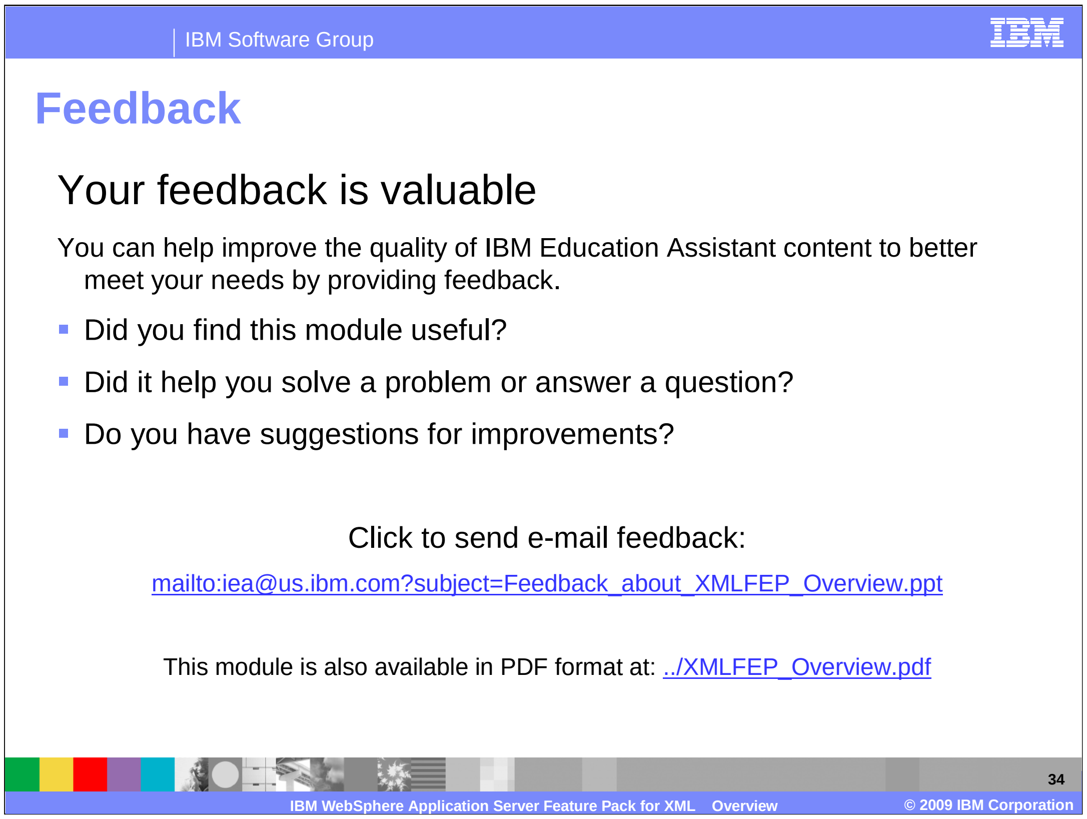
XML and XML programming model use are common in WebSphere Application Server. XPath 2.0, XSLT 2.0, and XQuery 1.0 as programming models offer new functionality, improved reliability, and ease of use when working with XML data. The feature pack provides a new runtime that supports XPath 2.0, XSLT 2.0, and XQuery 1.0 and a new XML API.



References

- WebSphere Application Server Feature Pack for XML
  - http://www.ibm.com/software/webservers/appserv/was/featurepacks/xml/
- W3C Announcement of W3C Recommendation
  - http://www.w3.org/2007/01/qt-pressrelease
- Primary specifications
  - http://www.w3.org/TR/xpath20/
  - http://www.w3.org/TR/xslt20/
  - http://www.w3.org/TR/xquery/
- Other specifications
  - http://www.w3.org/TR/xpath-functions/
  - http://www.w3.org/TR/xpath-datamodel/
  - http://www.w3.org/TR/xslt-xquery-serialization/
  - http://www.w3.org/TR/xquery-semantics/
  - http://www.w3.org/TR/xqueryx/

Here are some useful reference links.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_XMLFEP_Overview.ppt

This module is also available in PDF format at: ../XMLFEP_Overview.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers