Katherine Sanders – Staff Software Engineer, WebSphere Application Server – Web Services QoS Team, IBM Hursley
23 March 2011

WebSphere Application Server V7 Feature Packs for XML and Communications Enabled Applications

Agenda

- WebSphere Application Server Feature Packs Overview
- WebSphere Application Server V7 Feature Pack for XML
  - Existing XML Technologies
  - New XML Technologies
  - IBM XML API
  - Migration
- WebSphere Application Server V7 Feature Pack for Communications Enabled Applications
  - Overview
  - Business Scenarios
  - Interfaces
- Summary and Resources

Agenda

- WebSphere Application Server Feature Packs Overview
- WebSphere Application Server V7 Feature Pack for XML
  - Existing XML Technologies
  - New XML Technologies
  - IBM XML API
  - Migration
- WebSphere Application Server V7 Feature Pack for Communications Enabled Applications
  - Overview
  - Business Scenarios
  - Interfaces
- Summary and Resources

WebSphere Application Server Feature Packs Overview

- Add new capabilities to an existing version of WebSphere Application Server
- Allow access to the latest standards and programming models
- Avoid the disruption of upgrading to the next version
- Free to existing WebSphere Application Server customers (with the corresponding version)

Currently Available Feature Packs

- WebSphere Application Server V6.1:
  - Web Services (integrated into core runtime on V7)
  - EJB 3.0 (integrated into core runtime on V7)
- WebSphere Application Server V6.1 and V7:
  - Dynamic Scripting
- WebSphere Application Server V6.1 and V7, Community Edition v2.0 and V2.1:
  - Web 2.0
- WebSphere Application Server V7:
  - Modern Batch
  - OSGi Applications and Java™ Persistence API (JPA) 2.0
  - Service Component Architecture
  - XML
  - Communications Enabled Applications

Feature Pack Installation

- Install WebSphere Application Server
- Install IBM Installation Manager
- Run Installation Manager
  - Import the existing WebSphere Application Server installation database pointing to your WebSphere Application Server install directory
    - This syncs Installation Manager to previous WebSphere Application Server installation and service history
  - Install the Feature Pack
    - For workstations, use the default network repository
    - For servers (behind firewall), download the zipped repository and add it to the repositories of Installation Manager
- Augment the profile to use the Feature Pack using the Profile Management Tool
  - This step selectively activates the new runtime's OSGi bundle and makes it available to WebSphere Application Server applications within servers under that profile
  - Note that it only augments profiles you select to augment allowing you to experiment with the Feature Pack without impacting other non-augmented profiles

Agenda

- WebSphere Application Server Feature Packs Overview
- WebSphere Application Server V7 Feature Pack for XML
  - Existing XML Technologies
  - New XML Technologies
  - IBM XML API
  - Migration
- WebSphere Application Server V7 Feature Pack for Communications Enabled Applications
  - Overview
  - Business Scenarios
  - Interfaces
- Summary and Resources

eXtensible Markup Language (XML) 1.0

- Set of rules for encoding documents in machine readable form
- Originally designed for electronic publishing
- Now important in structured data interchange scenarios e.g REST, WS-*
- Every WebSphere Application Server application is using XML in some way or another
- Example: Names.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<doc>
  <name first="Jane" last="Doe"/>
  <name first="Jim" last="Smith"/>
  <name first="John" last="Doe"/>
</doc>
```

Existing XML Support in WebSphere Application Server V7

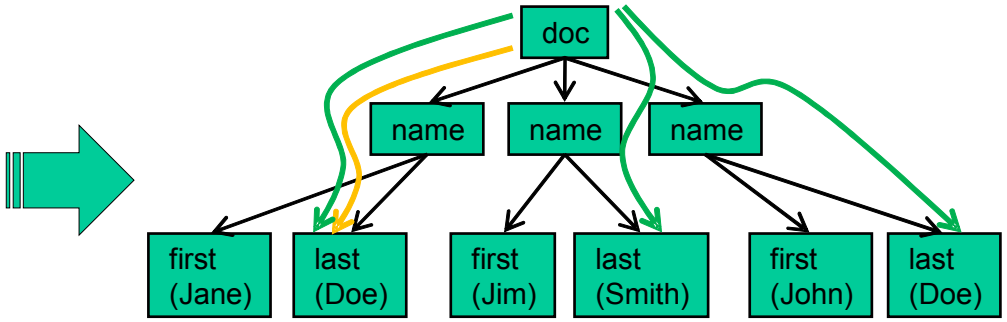
- Java API for XML Processing (JAXP)
  - 3 basic parsing interfaces: Document Object Model (DOM), Simple API for XML (SAX), Streaming API for XML (StAX)
  - XPath 1.0 and XSLT 1.0 for document transformation
- Java Architecture for XML Binding (JAXB) to map classes to and from XML representations
- IBM has provided optimized XML runtimes for these technologies in WebSphere Application Server and on IBM JDK's
- The problem with JAXP moving forward
  - JAXP has not been updated to support the newer XPath 2.0, XSLT 2.0, XQuery 1.0 programming models
  - Updating this API would take significant work as the data model is significantly different
  - JAXP wasn't as thread-safe as needed in a server environment
  - Creates performance and/or complexity issues in user programs
  - High level API's and extension mechanisms between XPath and XSLT are not consistent

XPath 1.0

- From specification (http://www.w3.org/TR/xpath)
- “The primary purpose of XPath is to address parts of an XML document … XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document.”

```
<?xml version="1.0" encoding="UTF-8"?>
<doc>
  <name first="Jane" last="Doe"/>
  <name first="Jim" last="Smith"/>
  <name first="John" last="Doe"/>
</doc>
```

names.xml



/doc/name[1]/@last → Doe

/doc/name/@last → Doe, Smith, Doe

Imperative vs Declarative XML Programming Models

```
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://www.w3.org/2005/Atom")) {
            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("http://www.w3.org/2005/Atom")) {
                    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") && node.getNamespaceURI().equals("http://
                                BlogBean bb = new BlogBean(blogid, node.getTextContent());
                                System.out.println(bb);
                            }
                        }
                    }
                }
            }
        }
    }
}
```

1 2 3 4 5

DOM

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”

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

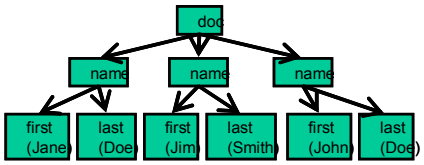
1 2 3 4 5

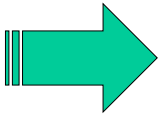
XPath

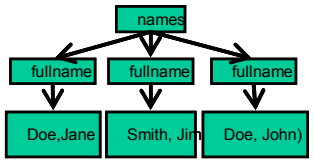
- Note the following differences
  - Imperative – User tells the runtime **how** to navigate data to find data of interest
  - Declarative – User tells the runtime **what** data is of interest
  - The DOM example creates many fine grained objects which can cause performance issues
  - The DOM code is much more complex and difficult to maintain

eXtensible Styleshet Language Transformations (XSLT) 1.0

- From specification (http://www.w3.org/TR/xslt)
  - "A transformation expressed in XSLT describes rules for transforming a source tree into a result tree. The transformation is achieved by associating patterns with templates. A pattern is matched against elements in the source tree. A template is instantiated to create part of the result tree. The result tree is separate from the source tree. The structure of the result tree can be completely different from the structure of the source tree. In constructing the result tree, elements from the source tree can be filtered and reordered, and arbitrary structure can be added."





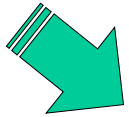


```
<?xml version="1.0" encoding="UTF-8"?>
<doc>
  <name first="Jane" last="Doe"/>
  <name first="Jim" last="Smith"/>
  <name first="John" last="Doe"/>
</doc>
```

names.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

    <xsl:template match="/">
        <names>
        <xsl:for-each select="doc/name">
            <xsl:element name="fullname">
                <xsl:value-of select="@last"/>,<xsl:value-of select="@first"/>
            </xsl:element>
        </xsl:for-each>
        </names>
    </xsl:template>
</xsl:stylesheet>
```

basic10.xsl



```
<?xml version="1.0" encoding="UTF-8"?>
<names>
    <fullname>Doe,Jane</fullname>
    <fullname>Smith,Jim</fullname>
    <fullname>Doe,John</fullname>
</names>
```

basic10Out.xml

WebSphere Application Server V7 Feature Pack for XML

- Adds support for:
  - XPath 2.0
  - XSLT 2.0
  - XQuery 1.0
- Includes backwards-compatibility modes for XPath 1.0 and XSLT 1.0
- Provides the IBM XML Application Programming Interface (IBM XML API)
  - Similar interface and interoperability between XPath 2.0, XSLT 2.0 and XQuery 1.0
  - Allows cascading one language output to next language input
  - Designed for use in a multi-threaded environment
- Includes the IBM Thin Client for XML with WebSphere Application Server
- Contains 49 samples for the API and features of the new standards
  - End to end demonstrations showing typical web applications as well as integration with XML databases such as DB2 pureXML

XPath 2.0

- Based on sequences – an ordered collection of zero or more items. An item is either a node or an atomic value
- Extensive Library of Functions and Operators
  - Date/time Handling – Includes current date functions to enable timestamping of output
  - New and Enhanced String Functions
  - Regular expression syntax for pattern matching
  - New and Extended Numeric Functions – floor, ceiling, round, abs, …
  - New Sequence Manipulation Functions – reverse, subsequence, remove, index-of, …
  - New Operators – intersect, except…
- Tests for more kinds of nodes and typed tests
  - element() – refers to any element node
  - element(po:shipto) – refers to an element node with the name po:shipto
  - //element(*, tns:AddressType)/postalCode returns the postalCode of any elements with a type of AddressType
    - If the document had both BillingAddress and MailingAddress, both would be returned
    - With XPath 1.0, you would have to look for both //tns:BillingAddress and //tns:MailingAddress
- Conditional, Iteration and Quantified Expressions – if, for, some and every keywords

XSLT 2.0 - Grouping

- Supports grouping with xsl:for-each-group, current-group() and current-grouping-key()

names.xml

```
<xsl:template match="/doc">
    <names>
        <xsl:for-each-group select="name" group-by="@last">
            <lastName>
                <xsl:attribute name="lastName">
                    <xsl:value-of select="current-grouping-key()"/>
                </xsl:attribute>

                <xsl:for-each select="current-group()">
                    <xsl:value-of select="@first"/><xsl:text> </xsl:text>
                </xsl:for-each>
            </lastName>
        </xsl:for-each-group>
    </names>
</xsl:template>
```

for-each-group.xsl

```
<names>
    <lastName lastName="Doe">Jane John </lastName>
    <lastName lastName="Smith">Jim </lastName>
</names>
```

for-each-groupOut.xml

XSLT 2.0 – Multiple Result Trees

- XSLT 1.0 limited output to a single tree
- In XSLT 2.0, you can declare what result-document is to be constructed for all output
  - This means you can output to two or more output results

names.xml

```
<xsl:template match="/doc">
   <xsl:result-document href="doc1">
      <out>
         <xsl:copy-of select="name[@last='Doe']"/>
      </out>
   </xsl:result-document>
   <xsl:result-document href="doc2">
      <out>
         <xsl:copy-of select="name[@first='Jim']"/>
      </out>
   </xsl:result-document>
</xsl:template>
```

multipleresults.xsl

```
<out>
   <name first="Jane" last="Doe"/>
   <name first="John" last="Doe"/>
</out>
```

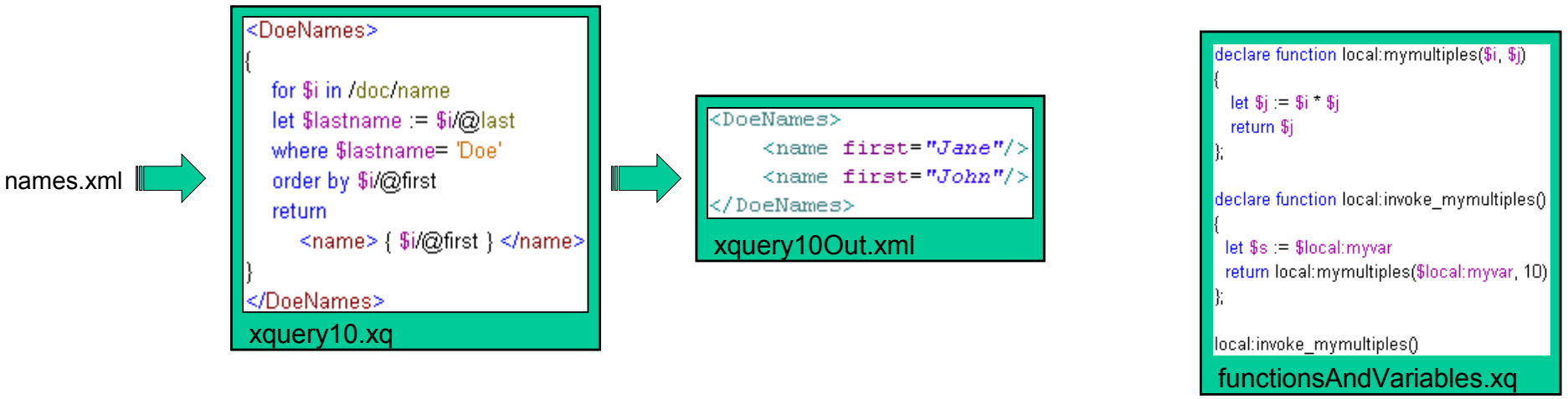
multipleresultsOut1.xml

```
<out>
   <name first="Jim" last="Smith"/>
</out>
```

multipleresultsOut2.xml

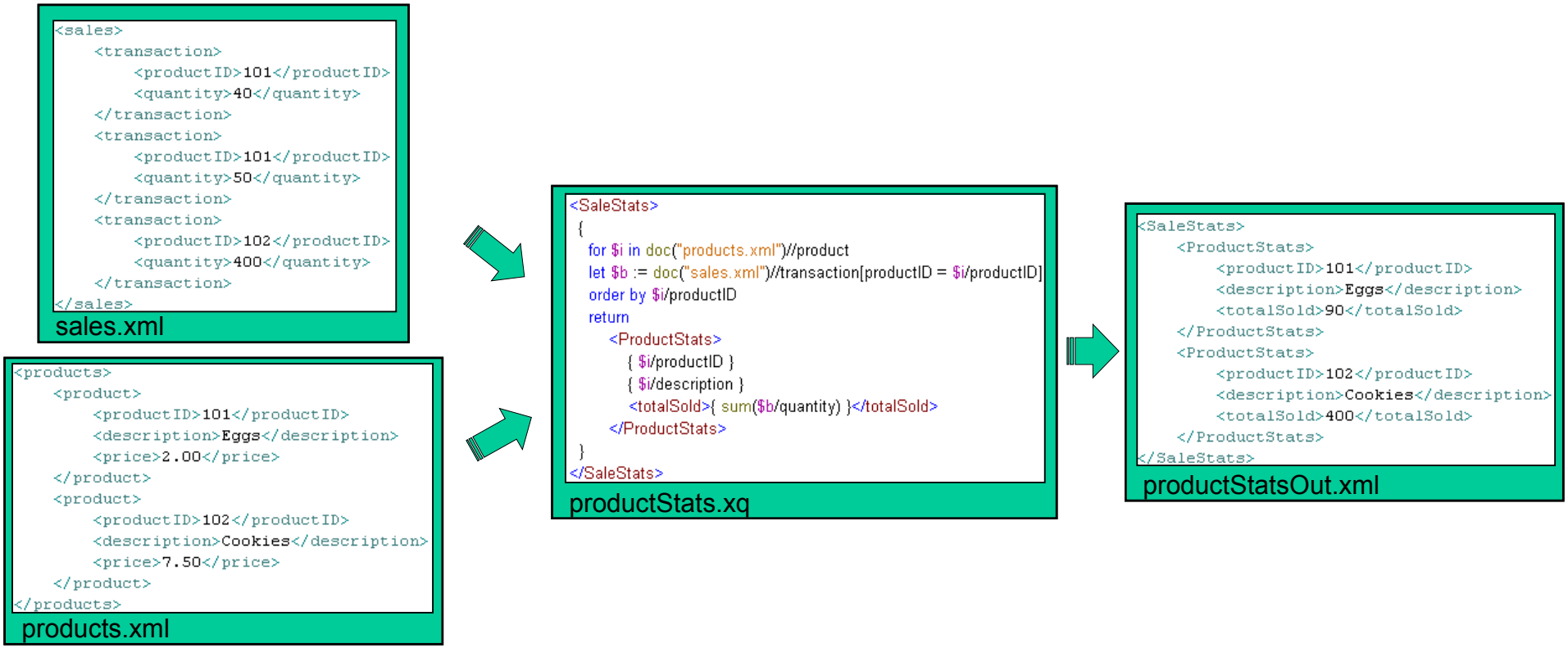
XQuery 1.0

- 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
  - Can be nested within any other XQuery expression, including as arguments to functions



XQuery 1.0 - Joins

- One of the strongest values of XQuery is its ability to join data across XML trees
  - Easy to understand language
  - All the typical joins (outer join, left outer join, full outer join) are possible




```
<sales>
    <transaction>
        <productID>101</productID>
        <quantity>40</quantity>
    </transaction>
    <transaction>
        <productID>101</productID>
        <quantity>50</quantity>
    </transaction>
    <transaction>
        <productID>102</productID>
        <quantity>400</quantity>
    </transaction>
</sales>
```

```
<products>
    <product>
        <productID>101</productID>
        <description>Eggs</description>
        <price>2.00</price>
    </product>
    <product>
        <productID>102</productID>
        <description>Cookies</description>
        <price>7.50</price>
    </product>
</products>
```

```
<SaleStats>
 {
  for $i in doc("products.xml")//product
  let $b := doc("sales.xml")//transaction[productID = $i/productID]
  order by $i/productID
  return
     <ProductStats>
        { $i/productID }
        { $i/description }
        <totalSold>{ sum($b/quantity) }</totalSold>
     </ProductStats>
 }
</SaleStats>
```

```
<SaleStats>
    <ProductStats>
        <productID>101</productID>
        <description>Eggs</description>
        <totalSold>90</totalSold>
    </ProductStats>
    <ProductStats>
        <productID>102</productID>
        <description>Cookies</description>
        <totalSold>400</totalSold>
    </ProductStats>
</SaleStats>
```

IBM XML API – Factories

- Used to create top level entities in API
- XFactory
  - All usages of the API will start with XFactory
  - Prepare executables
  - Get instances of other factories
  - Get instances of static/dynamic contexts
  - Perform schema loading, set validation defaults
- XItemFactory
  - Create atomic and complex items
  - Create sequence from items
- XSequenceTypeFactory
  - Create sequence types to represent types and cardinality of sequence items
- XCompilationFactory
  - Compile XML artifacts and load (from pre-compiled classes) compiled artifacts
  - Used with XCompilationParameters

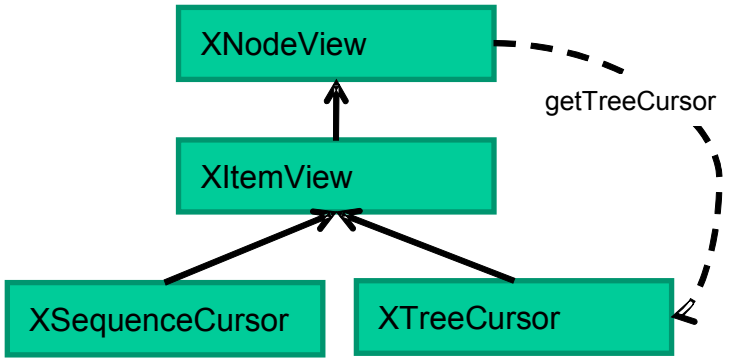
IBM XML API – Executables

- Used to execute XML expressions and programs
- XExecutable
  - Base class of all other Executables
  - execute returning XSequenceCursor
  - executeToList returning List<XItemView>
  - Thread-safe (re-usable across threads)
- XPathExecutable
  - XPath 2.0 (or 1.0 Backwards Compatibility Mode) Expression
- XSLTExecutable
  - XSLT 2.0 (or 1.0 Backwards Compatibility Mode) Stylesheet
- XQueryExecutable
  - XQuery 1.0 Expression

IBM XML API – Contexts

- Used to associate statically and dynamically known contexts with executable execution
  - These are concepts defined by W3C specifications
- Static context defines items that don't change across invocations of an executable
- Dynamic context defines items that are unique to each invocation of an executable
- XStaticContext
  - Declare user defined functions, namespaces, variables
  - Set compilation mode (compiler or interpreter)
  - Set 2.0 mode or 1.0 backwards compatibility mode
  - Setting of user implemented ErrorHandlers, SourceResolvers (for import/include)
  - Set input base URI, math modes, etc.
- XDynamicContext
  - Bind user defined function and variables
  - Setting of user implemented ErrorHandlers, SourceResolvers, ResultResolvers
  - Set output base URI, timezone, features, etc.
  - Set XSLT initial mode and template

IBM XML API – Data Views and Cursors



- Used for navigating (cursors) and viewing (views) data
- Views
  - XNodeView – view of a node or an item with complex type
  - XItemView – view of an item in a sequence with access methods for atomic types
- Cursors – pulls data only as requested by application over multiple invocations
  - XSequenceCursor – cursor for navigating returned sequences, returning current item
  - XTreeCursor – cursor for navigating returned nodes, attributes, namespaces, children, siblings, parent, the root
- Non-cursor ways to get data – pulls all data needed at time of single invocation
  - Simplifies programming experience, but incurs possible performance penalties
  - XExecutable.executeToList (instead of XSequenceCursor)
    - Performance penalties when result sequences are lengthy and/or deeply complex
  - XNodeView.getDOMNode (instead of XTreeCursor)
    - Performance penalties when backing source isn't DOM

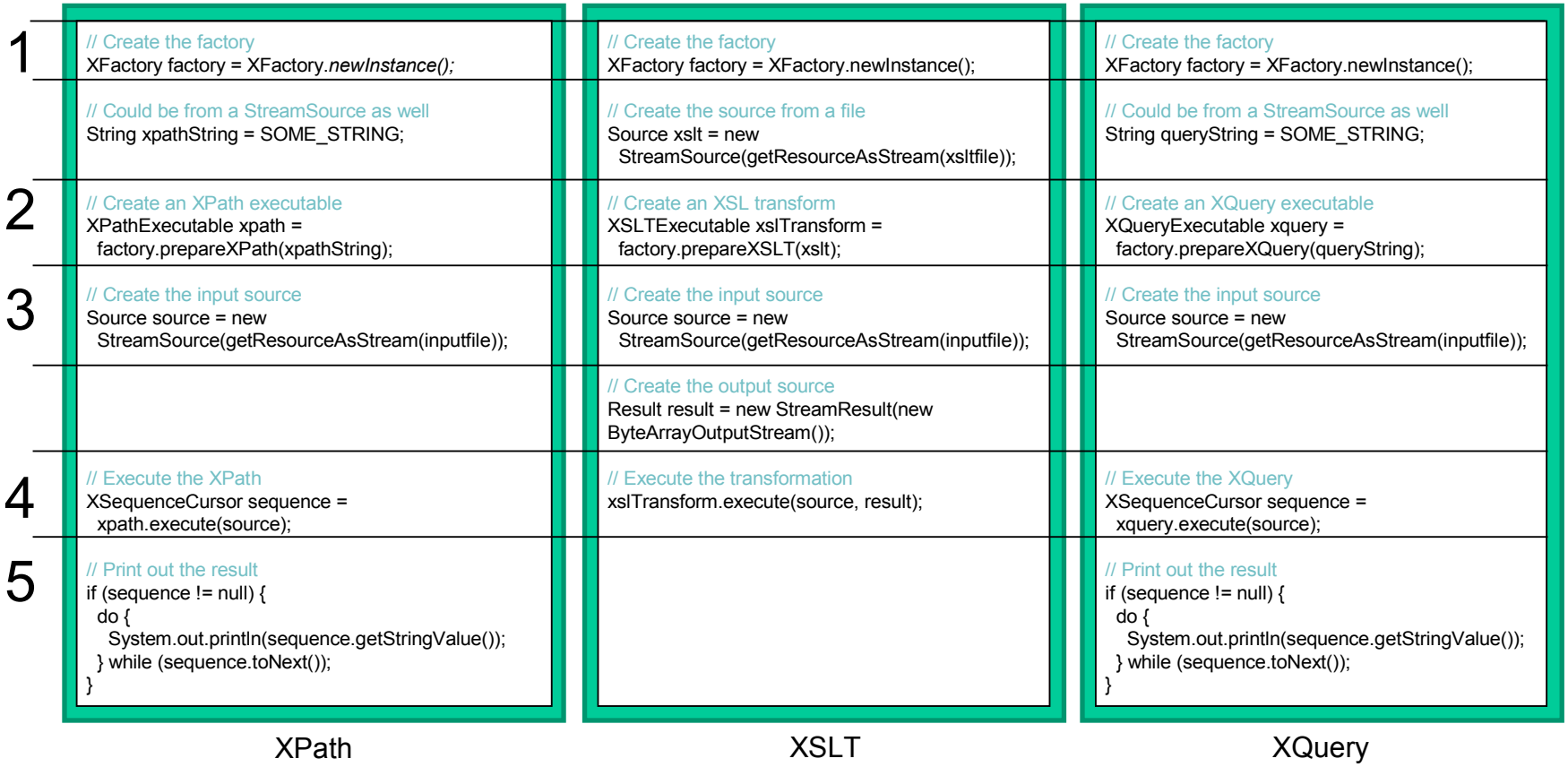
IBM XML API – Resolvers

- Used to resolve input stylesheets, input documents and output documents
- Default loading uses relative URIs – use resolvers to override default behavior
- XSourceResolver
  - Under static context – used to resolve imported and included stylesheets
  - Under dynamic context – used to resolve dynamically resolved input documents
- XResultResolver
  - Under dynamic context – used to resolve dynamically resolved output documents
- XCollectionResolver
  - Used at runtime by collection() function to retrieve a collection of nodes from arbitrary sources
- XSchemaResolver
  - Used at runtime to resolve user provided schema documents
- XUnparsedTextResolver
  - Used at runtime to resolve user provided textual resources

IBM XML API – Errors and Exceptions

- Used to handle errors and exceptions
- XErrorHandler
  - Recommend that users create and set in Static and Dynamic Context
  - If not created and set, exception will propagate to calling code with less information
- XSourceLocation
  - Reported as part of the error by XErrorHandler
  - Gives public id, columns and lines pointing to error in expression or stylesheet
- Common exceptions
  - XProcessException
    - Fatal error when preparing an expression or stylesheet due to errors in the input
    - Use the error to understand what to correct in your expression or stylesheet
  - XViewException
    - Fatal error when converting an item to a non-convertible type (e.g. Trying to get a date from a boolean)
    - Can be avoided by calling getValueType first

Basic XML Feature Pack API Usage



- Consistent steps across XPath/XSLT/XQuery
  1)Create a factory
  2)Create an executable (all have common base class)
  3)Create a source for the executable
  4)Execute the executable
  5)Navigate and print the results (XSLT stores results in Result object instead)

Cascading Example

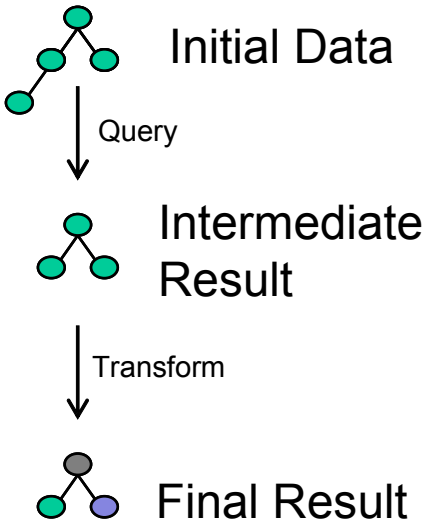
- An example of cascading XQuery output as XSLT input
  1)Start with initial data
  2)Execute the query on the input document, producing new data result
  3)Pass the result of the query as the input to the transformation, producing final data

```
// Create the factory
XFactory factory = XFactory.newInstance();

// Create the Query string
String queryString = SOME_STRING;

// Create the XQuery executable
XQueryExecutable query = factory.prepareXQuery(queryString);

// Create the stylesheet source from a file
Source xsltSource = new StreamSource(getResourceAsStream(xsltFile));

// Create the XSLT executable
XSLTExecutable transform = factory.prepareXSLT(xsltSource);

// Create the input source
Source source = new StreamSource(getResourceAsStream(inputFile));

// Execute the query
XSequenceCursor queryResult = query.execute(so

// Create the output source
Result result = new StreamResult(new ByteArrayOutputStream());

// Execute the transformation using the query result as input
transform.execute(queryResult, result);
```

1

2

3



Sample Applications

XPath 2.0
- Sample 1: Simple XPath invocation
- Sample 2: Invoking XPath 1.0 in backwards compatibility mode
- Sample 3: Invoking schema-aware XPath 2.0 expressions
- Sample 4: XPath 2.0 - document function
- Sample 5: XPath in compiled mode
- Sample 6: Path in pre-compiled mode
- Sample 7: XPath collation support

XQuery 1.0
- Sample 1: Simple XQuery invocation
- Sample 2: XQuery FLWOR support - doc function and cross document joins
- Sample 3: XQuery declare functions and variables
- Sample 4: XQuery type declation
- Sample 5: XQuery in compiled mode
- Sample 6: XQuery in pre-compiled mode
- Sample 7: XQuery type operations (typeswitch/cast as)
- Samples 8-11: XQuery validation of input/output
- Sample 12: XQuery schema aware processing

XSLT 2.0
- Sample 1: Simple XSLT invocation
- Sample 2: Invoking XSLT 1.0 in BC mode
- Sample 3: XSLT 2.0 updated for-each support
- Sample 4: XSLT 2.0 grouping support
- Sample 5: XSLT 2.0 regular expression support
- Sample 6: XSLT 2.0 date formatting
- Sample 7: XSLT 2.0 multiple results
- Sample 8: XSLT 2.0 tunnel parameters
- Sample 9: XSLT 2.0 stylesheet functions
- Sample 10: XSLT 2.0 initial template
- Sample 11: XSLT 2.0 template with multiple modes
- Sample 12-13: XSLT 2.0 XHTML support
- Sample 14: XSLT 2.0 character maps
- Sample 15: XSLT 2.0 as attribute
- Sample 16: XSLT 2.0 embedded stylesheets
- Sample 17: XSLT 2.0 in compiled mode
- Sample 18: XSLT 2.0 in pre-compiled mode
- Sample 19: XSLT 2.0 undeclare prefixes
- Sample 20: XSLT 2.0 next-match
- Sample 21: XSLT 2.0, XPath 2.0 collection function
- Samples 22-27: XSLT 2.0 validation of input/output/temp trees
- Sample 28: XSLT 2.0 schema-aware stylesheets
- Sample 29: XSLT 2.0 use-when
- Sample 30: XSLT 2.0 collation support

- 49 demonstrations of the API and features of XPath 2.0, XSLT 2.0, and XQuery 1.0 standards
- End to End Web Application Samples based on the “Blog Comment Checker”
  - Show how you can search Web 2.0 atom feeds for data and represent the results using transformation to XHTML
  - Demonstration of how natural and easy it is to select data, transform and query from XML
  - Additionally, shows how to merge and query data from an XML database

Migration

- Existing JAXP XPath 1.0, XSLT 1.0 applications will continue to run on the existing v7 XML runtime without any changes since it is a different codebase
- These applications can be converted to the new XML runtime by
  - Converting the invocation API to the IBM XML API (ease of conversion dependent on depth of JAXP usage)
  - Setting the backwards compatibility flag in the API (for XPath 1.0) or the version attribute in the stylesheet (for XSLT 1.0)
    - Backwards compatibility, as defined by the specification, handles most incompatibilities between 1.0 and 2.0
- It is recommended that long term, these XPath and XSLT applications be moved to the 2.0 version and runtime in order to take advantage of functional enhancements and reduced complexity

Migrating JAXP to the IBM XML API

	JAXP	XML Feature Pack API
1	// Create the factory TransformerFactory factory = TransformerFactory.newInstance();	// Create the factory XFactory factory = XFactory.newInstance();
2	// Create the source from a file Source xslt = new StreamSource(getResourceAsStream(xsltfile));	// Create the source from a file Source xslt = new StreamSource(getResourceAsStream(xsltfile));

3	// Create an XSL transform
	Transformer xslTransform =
	factory.newTransformer(xslt);
		// Create an XSL transform
		XSLTExecutable xslTransform =
		factory.prepareXSLT(xslt);

4	// Create the input source Source source = new StreamSource(getResourceAsStream(inputfile));	// Create the input source Source source = new StreamSource(getResourceAsStream(inputfile));

5	// Set the parameter value
	xslTransform.setParameter(“sortType”, “pricecode”);
		// Set the parameter value
		XDynamicContext dc = factory.newDynamicContext();
		dc.bind(new QName(“sortType”), “pricecode”);

6	// Create the result Result result = new StreamResult(new ByteArrayOutputStream());	// Create the result Result result = new StreamResult(new ByteArrayOutputStream());

7	// Execute the transformation
	xslTransform.transform(source, result);
		// Execute the transformation
		xslTransform.execute(source, dc, result);

1)Use an XFactory instead of a TransformerFactory
2)Source can be used to load the stylesheet in both API’s
3)Create a XSLTExecutable object instead of a Transfomer
4)Source can be used to load the input data in both API’s
5)Create a dynamic context and bind variables to this context
- The dynamic context is valuable to ensure thread safety of Executables

6)Result can be used to store the output in both API’s
7)Execute the transformation

Agenda

- WebSphere Application Server Feature Packs Overview
- WebSphere Application Server V7 Feature Pack for XML
  - Existing XML Technologies
  - New XML Technologies
  - IBM XML API
  - Migration
- WebSphere Application Server V7 Feature Pack for Communications Enabled Applications
  - Overview
  - Business Scenarios
  - Interfaces
- Summary and Resources

WebSphere Application Server V7 Feature Pack for Communications Enabled Applications (CEA)

"A **communications enabled application (CEA)** is a set of information technology (IT) components and communication technology components that are integrated using a particular service-oriented architecture (SOA) to increase the productivity of an organization and/or improve the quality of users' experiences." - Wikipedia

- Add communications capabilities to new and existing applications
- Improves customer support experience through new methods of interaction
- Lowers costs through reuse of existing skills, applications and telephony infrastructure
- Developers do not need to understand underlying communications protocols
- No client-side installation required

CEA feature pack version 1.0

- Enables communications capabilities such as:
  - **Click-to-call** – customer requests a call from a company by entering their phone number on the company's website
  - **Co-browsing** – two Web users to share the same browsing session. One user controls the session; the other user has no control, but can view the activity of the other user.
  - **Two-way synchronised forms** – HTML forms in which two people can collaboratively edit and validate fields. Both parties can see the same form. The fields in the form change in response to input provided by either person. Individual form fields, such as credit card number, can be masked to only display a subset of the field data entered to the reviewing party.
- Provides access to standards-based telephony infrastructure
  - Can make / receive phone calls and receive call notifications
  - Supports the latest Session Initiation Protocol (SIP) Servlet 1.1 standard (JSR 289)
- Utilises diverse developer skills with Web services and REST services-based APIs and Web 2.0 widgets
- Includes a unit test environment to prototype and test applications without the need to access the corporate telephony network

New Features in CEA Feature Pack Fixpacks

- 1.0.0.1
  - Clustering and failover support on distributed platforms
  - More secure Web collaboration URIs using a nonce to prevent session snooping
  - iWidget support for CEA widgets
  - Widgets support Dojo level 1.3.2 (previously supported 1.3.1)
- 1.0.0.3 – Clustering and failover support on z/OS
- 1.0.0.5
  - New Web 2.0 mobile widgets:
    - Click to call, Call Notifications, Contact Center Co-browsing, Peer-to-peer Co-browsing and Two Way Synchronized Forms optimized for iPhone and Android
    - Helps to build applications with native mobile web application look and feel
    - Support for mobile specific interactions such as touch and gesture support for selecting, scrolling and zooming on mobile browsers
    - Enables mobile users to Co-browse with desktop or mobile users
  - SIP servlet requests can include proprietary header fields
- 1.0.0.7 – Administration using the Admin Agent
- 1.0.0.9 – Widgets support Dojo level 1.5

Business scenario 1: Click-to-call with co-browsing

Customer browsing a commercial website has a question for a customer service representative (CSR).

Customer types their phone number into the Web page to request a call be initiated with the CSR (click-to-call).

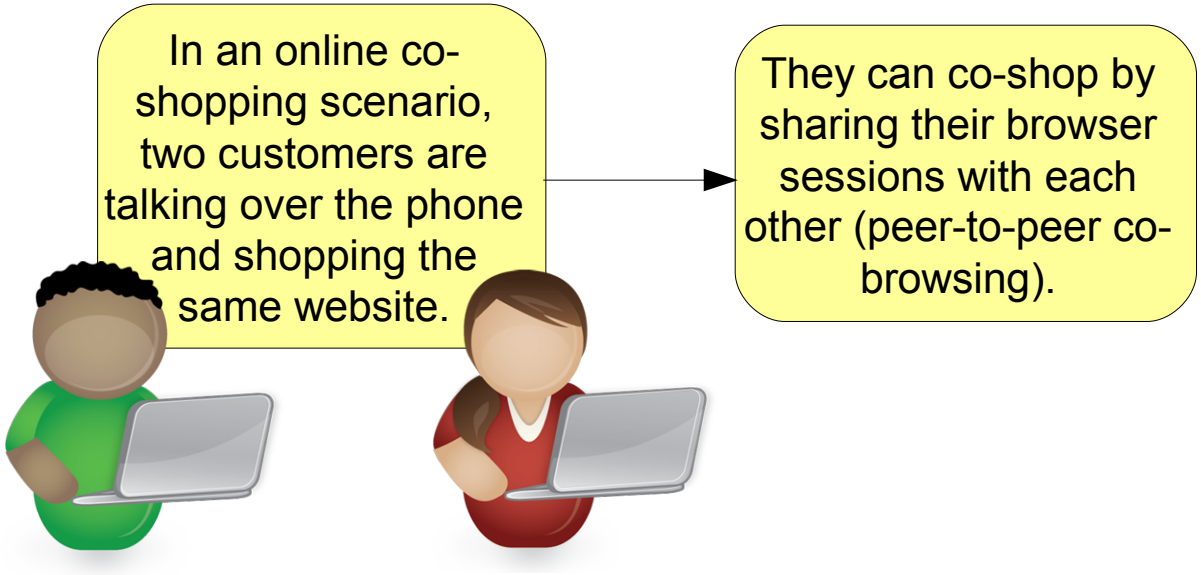
A CSR is logged in and waiting for call notifications at the company.

When the customer requests a call, the CSR is connected to the customer through voice over IP (VOIP).

The CSR can also start a co-browsing session with the customer. This allows the customer to view the pages the CSR is highlighting, including items on the page, such as price or model number.

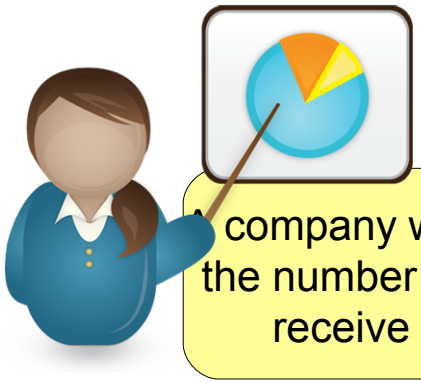
- Registered customers, with preferences, might have additional features available.
  - E.g. The CSR could check the inventory of their preferred store location for the item
  - E.g. The customer can view the page in a different language than the CSR.

Business scenario 2: Shopping with a friend



- Two-way collaboration is enabled, allowing each person to show pages to the other and highlighting items of interest.
- Each user, however, is able to maintain a separate session with their own shopping cart and preferences.

Business scenario 3: Tracking and reporting call statistics



A company wants to track the number of calls they receive in a day.

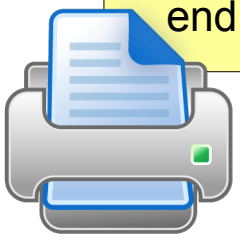
They have a business application that uses a Web service client to interact with the CEA Web service.

The application registers to be notified of incoming calls using the Web service client.

When calls are received, the telephone system notifies the CEA Web service, which sends a notification to the business application.

The business application receives the notification and increments a counter.

The data that the application gathers in its counter is logged and stored, to be used in a report that is generated at the end of the business day.



REST Interface

- REpresentational State Transfer (REST) is an architectural style that allows you to manipulate resources defined at URIs with pre-defined methods
- Calls are similar to standard HTTP requests, except that they have a specific format that is recognised by the REST interface
  - The CEA Web service has a servlet that is constantly listening for requests.
  - The servlet parses each incoming request and returns a response to it using JSON or XML
- The REST APIs allow you to:
  - Access telephony services
    - Make/End a call
    - Get call status
    - Register/Unregister for call notification
    - Get call notification information
  - Share data across two sessions:
    - Enable collaboration
    - Get collaboration status
    - Start/End a collaboration session with a peer
    - Send data to the collaboration peer
    - Retrieve event data (call status, collaboration status, collaboration data)

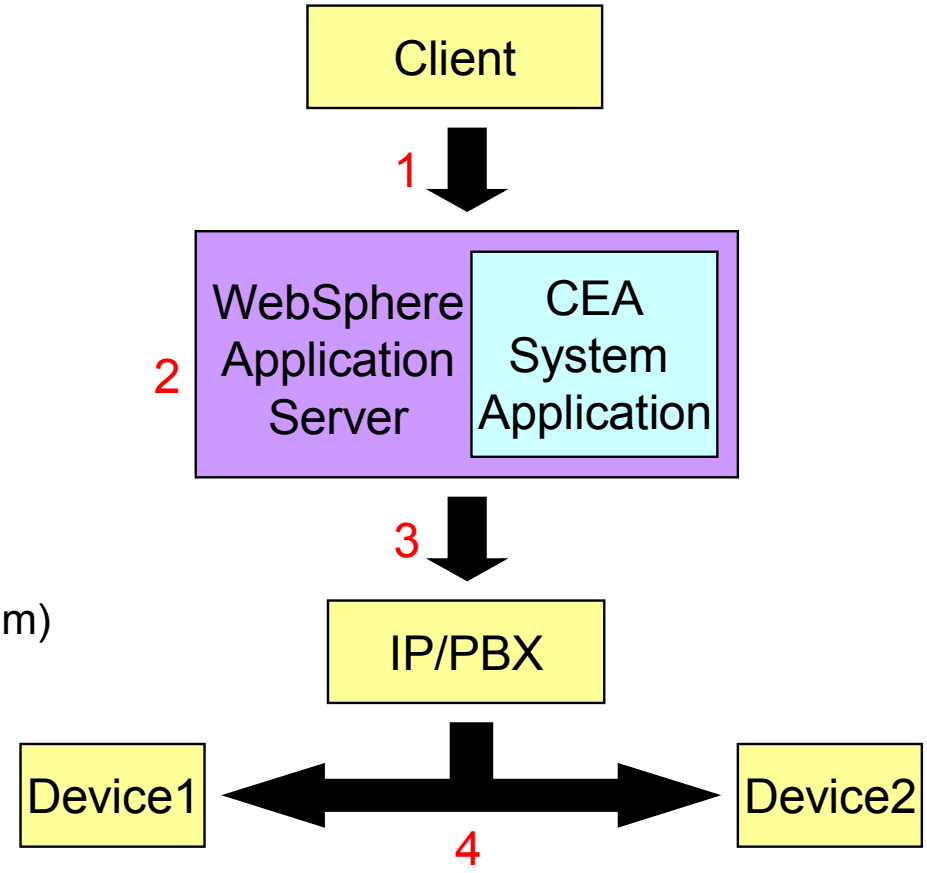
REST Telephony Request Flow

1)The client sends an HTTP REST request

2)The WebSphere Application Server Web container routes the request to the CEA system application
- The CEA servlet interprets the REST request

3)The WebSphere Application Server Web container routes the request to the CEA system application

4)The Internet Protocol (IP) Private Branch Exchange (PBX) (business telephone system) establishes a call between the devices



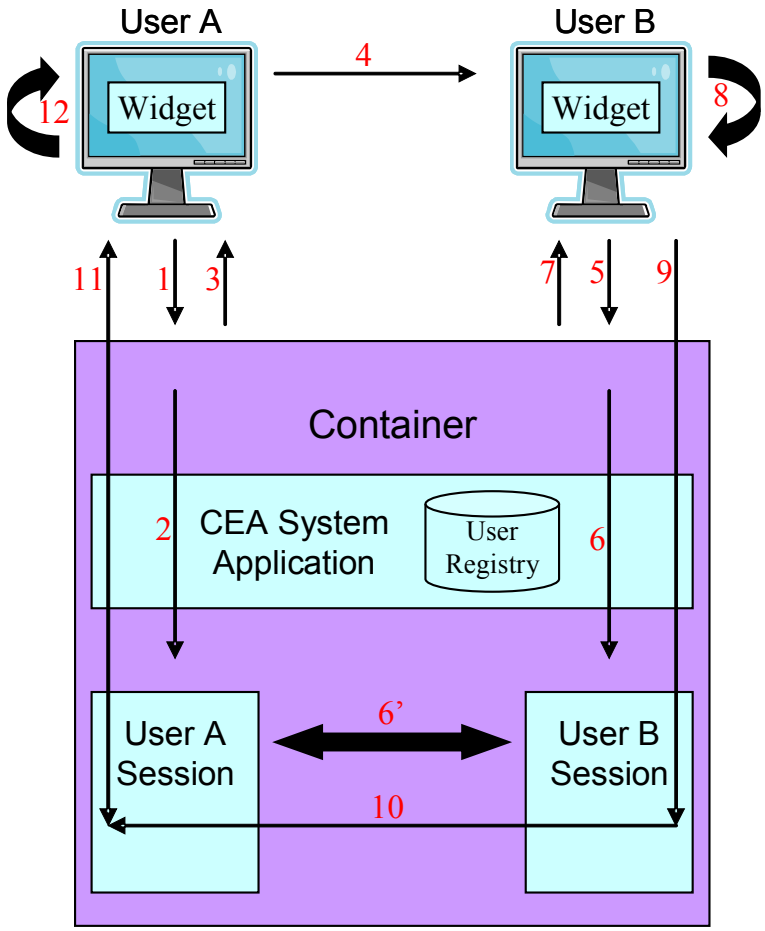
REST Web Collaboration Request Flow

1)User A sends a REST request to enable collaboration
2)The WebSphere Application Server Web container calls the CEA system application
- User A session is established and session location is placed the user registry

3)REST Response is sent to User A – includes URI for peers to start collaboration
4)User A sends User B the “for peer URI”
5)User B sends a REST request with that URI
6)The WebSphere Application Server Web container calls the CEA system application
- User B session is established and session location is placed in user registry
- User A is found in user registry and a “collaboration link” is established between them

7)REST Response is sent to User B
- Includes data exchange URI (to send/fetch data)
- Modal windows activate in each widget

8)User B highlights text / scrolls / fills form
9)Widget on User B sends events via REST
10)CEA system application sends data to “linked” User A session
11)User A widget polls for events via REST
12)User B events are captured in User A’s widget



REST Click To Call Example (1)

- Full code example is available in the information center:
  - http://publib.boulder.ibm.com/infocenter/wasinfo/fep/index.jsp?topic=/com.ibm.websphere.ceafep.multiplatform.doc/info/ae/ae/tcea_manage_calls_rest_step5.html
- Create a simple HTML form where the user can enter two SIP URIs that represent two phones:

```
<form name="myform">
  Caller Uri: <input type="text" name="caller_uri" value=""></input>
  Callee Uri: <input type="text" name="callee_uri" value=""></input>
  <input type="submit" onclick="sendRequest()" value="Send JSON">
</form>
```

- The form also has a “Send JSON” button that calls the sendRequest javascript method when it is clicked

REST Click To Call Example (2)

```
<script type="text/javascript"
        language="javascript">
function createJSON(addressOfRecord, peerAddressOfRecord) {
  var json = "{";
  json += "addressOfRecord:" + addressOfRecord;
  json += ",peerAddressOfRecord:" + peerAddressOfRecord;
  Json += ",enableCollaboration:false";
  json += "}";
  return json;
}
function sendRequest() {
  try {
    var request = new XMLHttpRequest();
    var caller = document.myform.caller_uri.value;
    var callee = document.myform.callee_uri.value;
    request.open("PUT", "CommServlet/call?JSON=true", false);
    request.send(createJSON(caller, callee));
    alert(request.responseText);
  } catch (err) {
    alert(err.description);
  }
}
</script>
```

Generates HTTP PUT request with body:
{"addressOfRecord":"sip:phone1@192.168.1.100",
"peerAddressOfRecord":"sip:phone2@192.168.1.100"
enableCollaboration:"false"}
To URL: http://<host>:<port>/commsvc.rest/CommServlet/call?JSON=true

REST Click To Call Example (3)

- In the JSON response you can see that a call has been attempted between the caller and callee, and a collaboration ID has been generated to identify the collaboration.

```
{
 "returnCode":200,
 "infoMsg":"Call attempted between sip:phone1@192.168.1.100 and
sip:phone2@192.168.1.100.",
 "callerAddressOfRecord":"sip:phone1@192.168.1.100",
 "calleeAddressOfRecord":"sip:phone2@192.168.1.100",
 "callServiceUri":"CommServlet/call;ibmappid=local.1242140626552_42",
 "callNotifyUri":"CommServlet/callerNotification;
ibmappid=local.1242140626552_42",
 "collaborationStatus":"NOT_ESTABLISHED",
 "eventUri":"CommServlet/event;ibmappid=local.1242140626552_42"
}
```

- The service can then be polled to see if the status of the call has changed to make sure the call was established.
- A full list of the REST calls available and more examples are available in the information center: http://publib.boulder.ibm.com/infocenter/wasinfo/fep/index.jsp?topic=/com.ibm.websphere.ceafep.multiplatform.doc/info/ae/ae/rcea_rest_overview.html

Widgets Interface

- Dojo widgets are pre-packaged components of JavaScript and HTML code that add interactive features that work across platforms and browsers.
- The CEA feature pack comes with three ready-to-integrate widgets, and two extendable widgets that developers can customise to handle more advanced tasks.
- These widgets provide the following capabilities in Web applications:
  - Request phone calls (click-to-call)
  - Receive call notifications
  - Collaborate and Co-browse
  - Create and configure two-way forms
- The widgets can be embedded into any Web page by adding a JavaScript reference to the CEA Dojo toolkit and adding the tag definitions to the HTML files.
- The widgets communicate with the back-end service using the REST APIs.
- The Plants By WebSphere sample application has been extended to include the click-to-call and call notification widgets for the feature pack as an example

Widget Click To Call Example

- Just 2 imports and 1 line of HTML to add click-to-call on any Web page

1)Import fully customizable CSS

```
<style type="text/css">
  @import "<contextRoot>/ceadojo/dijit/themes/tundra/tundra.css";
  @import "<contextRoot>/ceadojo/cea/widget/ClickToCall/ClickToCall.css";
  @import "<contextRoot>/ceadojo/cea/widget/CollaborationDialog/
CollaborationDialog.css";
</style>
```

2)Import extensible JavaScript

```
<script type="text/javascript" src="<contextRoot>/ceadojo/dojo/dojo.js"
  djconfig="parseOnLoad: true, isDebug: false"></script>
```

3)Add one line of HTML

```
<div ceadojoType="cea.widget.ClickToCall" widgetNumber="xxx-xxx-xxxx"
 enableCollaboration="false" canControlCollaboration="false"
 defaultCollaborationUri="<contextRoot>/cobrowseWelcome.html">
```

JSR 289 Interface

- SIP is an application-layer protocol that allows for the creation and management of multimedia communication sessions between devices.
- SIP provides the technology to support the click-to-call function in this feature pack, and is also available for use in developing SIP-based applications.
- WebSphere Application Server V7 supports SIP Servlet 1.0 (JSR 116).
- The CEA feature pack supports SIP Servlet 1.1 standard (JSR 289), which adds additional features, including:
  - **Application routing** – on initial requests the container calls the application router to determine which application to invoke based on the type of request
  - **Annotation-based programming** – a fast way to develop applications by embedding metadata directly in the code
  - **Converged applications** – contain SIP servlet components and other Java EE components, like HTTP servlets and enterprise beans
  - **Back-to-back user agent (B2BUA) APIs** – mediate signaling between two endpoints without breaking the functionality by using a new B2BUA helper class that automatically maintains links between sessions on both sides of the B2BUA.

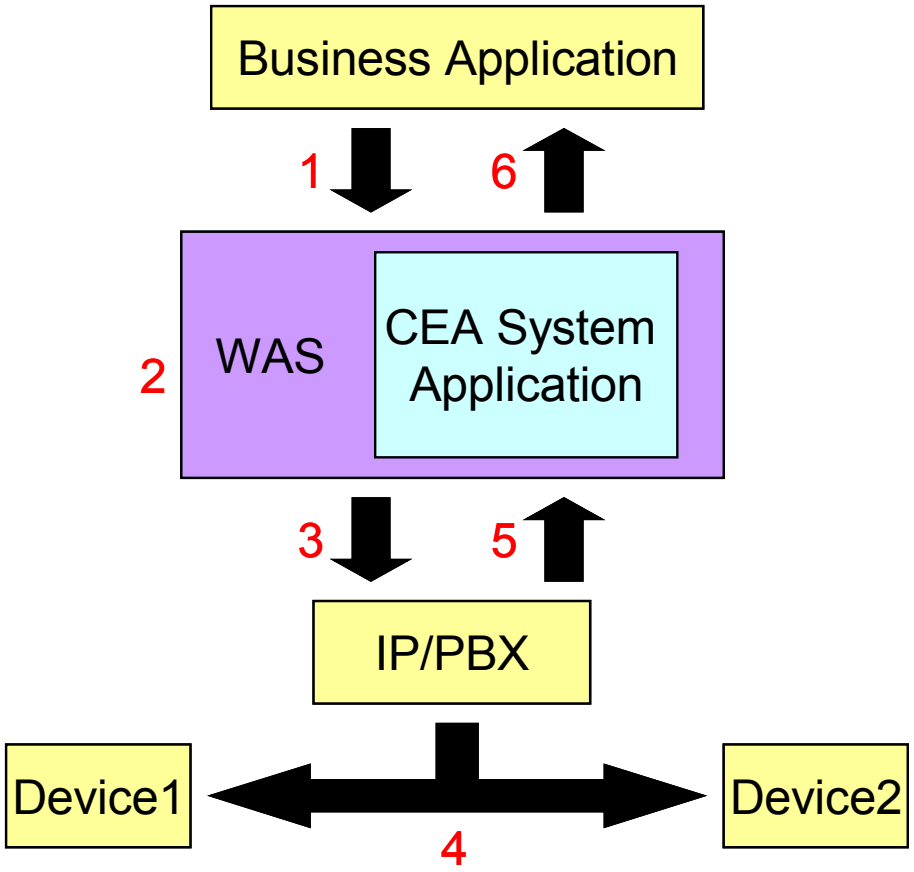
JSR 289 Samples

- IBM Rational® Application Developer provides three examples that you can use as a learning exercise:
  - The Call Blocking sample checks a list to determine if the caller is valid. If the caller is not valid, the call is blocked. If the caller is valid, the call is forwarded.
  - The Call Forwarding sample checks to determine if the caller is in a forwarding list and forwards the call.
  - The Third Party Call Control sample demonstrates how to use the Converged capability by implementing a controller that sets up and manages a communications relationship between two parties.

Web services Interface

- Typically Web service applications:
  - Communicate using XML messages that follow the SOAP standard over HTTP
  - Publish a machine-readable description of the operations offered by the service written in the Web Services Description Language (WSDL)
- The CEA Feature Pack provides a ControllerService that can be invoked by Web Service clients to manage telephone calls
- Web service calls can be made by business applications to access the communication system and:
  - Open a telephony session
  - Make a call
  - End a call
  - Close a telephony session
  - Get asynchronous call status updates using WS-Notification (a standard for publish and subscribe messaging for Web services)

Web Services Request Flow



1)The business application sends a call request to the CEA Web service

2)The WebSphere Application Server Web container routes the request to the CEA system application

3)The CEA system application sends the request to the IP PBX in a SIP message

4)The IP PBX establishes a call between the devices

5)The IP PBX sends device events to the CEA system application

6)The CEA system application sends device events to the business application using WS-Notification

Web Services Sample Application

- A sample Web services application is supplied with the CEA Feature Pack
- You can regenerate this application using Rational Application Developer 7.5:

1)Obtain the WSDL and schema files for the CEA service from the application server by saving the files at these URLs:
   - http://host:port/commsvc.rest/ControllerService?wsdl
   - http://host:port/commsvc.rest/CeaNotificationConsumer?wsdl
   - http://host:port/commsvc.rest/ControllerService/WEB-INF/wsdl/ControllerService_schema1.xsd

2)Generate the Web services client proxy code and the WS-Notification consumer service
   - Create a dynamic web project, import the WSDL and schema files
   - Right-click the ControllerService.wsdl file and select Web Services > Generate Client
   - Right-click the CeaNotificationConsumer.wsdl file and select Web Services > Generate Java bean skeleton

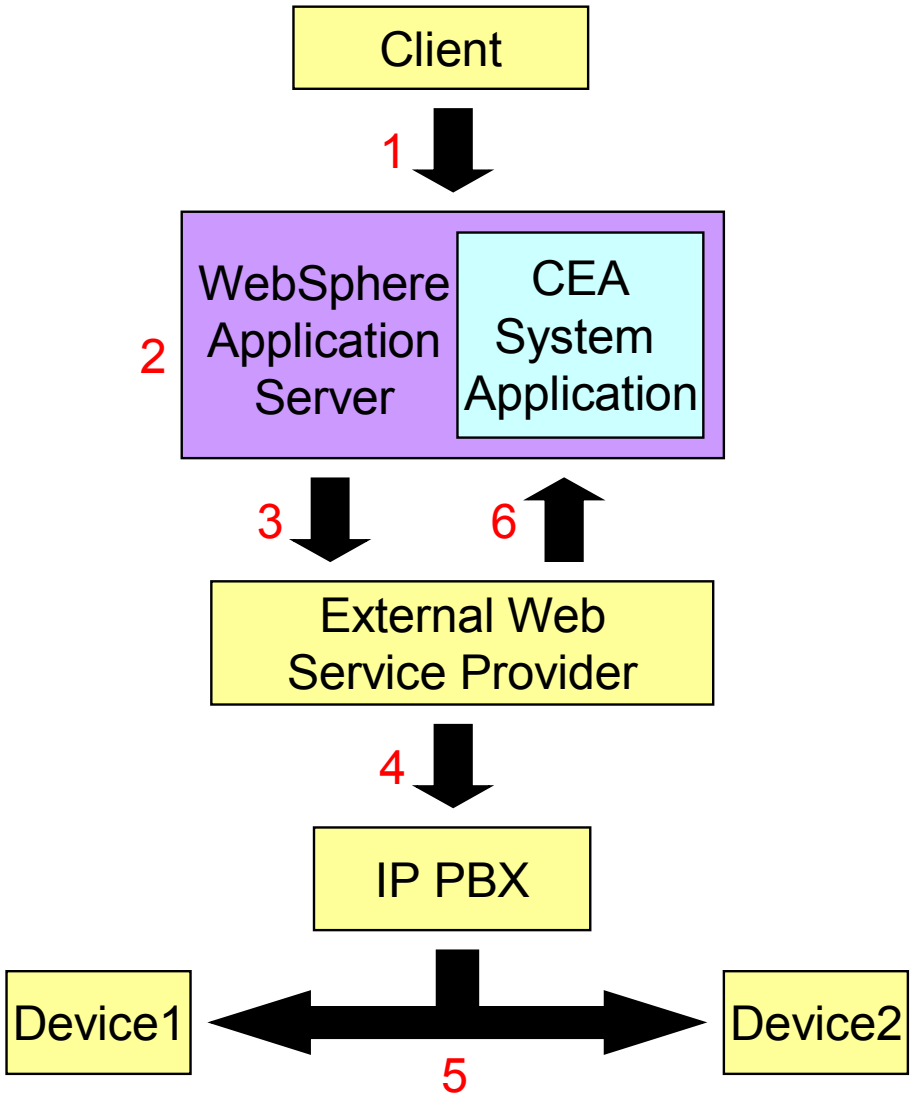
3)Create a front-end servlet that calls the client proxy methods to manage telephone calls

4)Implement the notify() method on the WS-Notification consumer service to set the call status that is displayed by the servlet code

External Web Services Support

- When the CEA Web service is invoked, it interacts with an IP PBX to monitor and control phones.
- If an external provider creates a Web service that implements the same WSDL as the CEA Web service, then CEA can be configured to use that provider instead.
- This allows vendors to customize interactions with their IP PBX.
- This configuration disables the existing CEA Web service, but the REST interface is still available.
- As REST requests are received, CEA uses a Web services client to communicate with the external Web service provider.
- The external Web service provider is responsible for all communication with the IP PBX to provide third party call control.

External Web Services Call Flow



1)The client sends an HTTP REST request.

2)The WebSphere Application Server Web container calls the CEA system application.
- The CEA servlet interprets the REST request.
- The CEA servlet uses a Web services client.

3)The CEA system application sends the Web service request to the external Web service provider.

4)The external Web service interacts with the IP PBX. Interaction can be proprietary.

5)The IP PBX establishes a call between devices.

6)The external Web service sends device events to CEA using WS-Notification.

Which interface should I use?

- Depends on type of application and existing technologies / skills
- REST
  - Have an existing user interface so do not need the widgets
- Widgets
  - Provide a customisable user interface that can be embedded into any Web application to provide CEA-enabled features
- Web services
  - Typically used by other business applications that can embed a Web service client.
- JSR 289
  - For telephony applications that prefer to use the telephony APIs to build communication applications.

Agenda

- WebSphere Application Server Feature Packs Overview
- WebSphere Application Server V7 Feature Pack for XML
  - Existing XML Technologies
  - New XML Technologies
  - IBM XML API
  - Migration
- WebSphere Application Server V7 Feature Pack for Communications Enabled Applications
  - Overview
  - Business Scenarios
  - Interfaces
- Summary and Resources

Summary

- The XML Feature Pack:
  - Adds support for XPath 2.0, XSLT 2.0 and XQuery 1.0
  - Includes backwards-compatibility modes for XPath 1.0 and XSLT 1.0
  - Provides the IBM XML Application Programming Interface (IBM XML API)
  - Includes the IBM Thin Client for XML with WebSphere Application Server
  - Contains 49 samples for the API and features of the new standards
- The CEA Feature Pack:
  - Simplifies the addition of communications capabilities to new and existing applications
  - Improves customer support experience through new methods of interaction such as click-to-call, co-browsing and synchronised two-way forms
  - Lowers costs through reuse of existing Java skills, applications and telephony infrastructure
  - Provides REST, Dojo widgets, JSR 289 and Web services interfaces
- Both feature packs are free to existing WebSphere Application Server V7.0 customers

Resources

- Specifications:
  - XPath 2.0: http://www.w3.org/TR/xpath20/
  - XSLT 2.0: http://www.w3.org/TR/xslt20/
  - XQuery 1.0: http://www.w3.org/TR/xquery/
- Product sites:
  - Feature Packs: http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/
  - Installation Manager: http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg24023498
  - CEA fixpacks: http://www-01.ibm.com/support/docview.wss?uid=swg27017328
  - XML fixpacks: http://www-01.ibm.com/support/docview.wss?uid=swg24027719
- Documentation:
  - developerWorks XML Zone: http://www.ibm.com/developerworks/xml/
  - XML Redbook: http://www.redbooks.ibm.com/redpapers/abstracts/redp4654.html
  - CEA YouTube Channel: http://www.youtube.com/user/IBMcea
  - CEA Blog: http://ibmcea.blogspot.com/
  - CEA Redbook: http://www.redbooks.ibm.com/redpieces/abstracts/redp4613.html
- My email – katherine_sanders@uk.ibm.com

Any Questions?