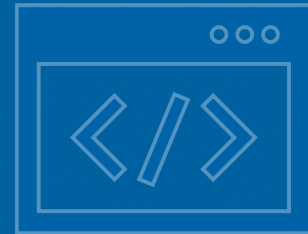


CSF205 – Data Representation, Markup Languages, & Web Services

Lecture 11: More XML Programming and XML Security



Learning Objectives

1. Recap of JDOM
2. Namespaces
3. Schema Validation using JDOM
4. Transformation using JAXP
5. XML Security

JDOM

API of concrete classes.

JDOM isn't DOM (we do not use `w3c.dom.*` packages in JDOM).

Does not have a parser of its own; depends on SAX Parser and DOM Parser inputs.

Open source, tree based, pure Java API for XML processing.

JDOM

Represents an XML document as a tree composed of elements, attributes, comments, processing instructions etc.

Entire tree available at all time.

Unlike DOM, JDOM allows you to access any part of the tree at anytime.

No generic **Node** interface like in DOM.

NAMESPACES

Namespaces in JDOM

The basic JDOM rule about namespaces:

When an element or attribute is in a namespace, rather than specifying its full qualified name, you give its local name, its prefix, and its URI, in that order.

If the element is in the default namespace, omit the prefix.

You do not need to add attributes for the namespace declarations.

The outputter will figure out reasonable places to put them when the document is serialized.

Namespaces - Prefixed in JDOM

```
Element root = new Element("math", "mathml",  
    "http://www.w3.org/1998/Math/MathML");  
//Prefix Namespace  
Element mrow = new Element("mrow", "mathml",  
    "http://www.w3.org/1998/Math/MathML");  
Element mi = new Element("mi", "mathml",  
    "http://www.w3.org/1998/Math/MathML");
```

Results in

```
<mathml:math xmlns:mathml="http://www.w3.org/1998/Math/MathML">  
  <mathml:mrow>  
    <mathml:mi>f(1)</mathml:mi>
```

Namespaces - UnPrefixed in JDOM

```
Element root = new Element("math", "mathml",  
    "http://www.w3.org/1998/Math/MathML");  
//Prefix Namespace  
Element mrow = new Element("mrow",  
    "http://www.w3.org/1998/Math/MathML");  
Element mi = new Element("mi",  
    "http://www.w3.org/1998/Math/MathML");
```

Results in

```
<mathml:math xmlns:mathml="http://www.w3.org/1998/Math/MathML">  
  <mrow xmlns="http://www.w3.org/1998/Math/MathML">  
    <mi>f(1)</mi>
```


Namespace class – Prefixed in JDOM

Namespace ns =

```
    Namespace.getNamespace("mathml",  
    "http://www.w3.org/1998/Math/MathML");  
Element root = new Element("math", ns);  
Element mrow = new Element("mrow", ns);  
Element mi = new Element("mi", ns);
```

Returns

`<mathml:math`

`xmlns:mathml="http://www.w3.org/1998/Math/MathML">`

`<mathml:mrow>`

`<mathml:mi>f(1)</mathml:mi>`

Namespace class - UnPrefixed in JDOM

```
Namespace ns =  
    Namespace.getNamespace("http://www.w3.org/19  
    98/Math/MathML");  
Element root = new Element("math", ns);  
Element mrow = new Element("mrow", ns);  
Element mi = new Element("mi", ns);
```

Results

```
<math xmlns="http://www.w3.org/1998/Math/MathML">  
  <mrow>  
    <mi>f(1)</mi>
```

Namespace Comparison in JDOM

Namespace objects are compared based on the URI's

Two Namespace objects are equal if their URI's are the equal
(regardless of their prefix)

```
Namespace ns =  
    Namespace.getNamespace("mathmla",  
        "http://www.w3.org/1998/Math/MathML");  
Namespace ns2 =  
    Namespace.getNamespace("mathmlb",  
        "http://www.w3.org/1998/Math/MathML");  
System.out.println(ns.equals(ns2));  
Returns True!
```

SCHEMA VALIDATION

Validation using JDOM

Internal Schema Validation:

- Use the schema specified in the XML specified using *noNamespaceSchemaLocation* or *schemaLocation*.
- Let the input framework know to validate the XML using the internal Schema.
- In case of a Schema error, if the XML does not adhere to the XSD, a JDOMParseException is thrown.
- E.g.

```
saxBuilder = new  
SAXBuilder(XMLReaders.XSDVALIDATING);
```

Validation using JDOM

External Schema Validation:

- Used when the XML doesn't have a Schema specified and you want to validate it against one.
- Specify a schema in the Java Code for validation against the XML.
- Ensure that both the Schema and XML file are in the classpath.
- E.g.

```
XMLReaderJDOMFactory schemafac = new  
XMLReaderXSDFactory("<XSD file>");  
saxBuilder = new SAXBuilder(schemafac);
```

XPATH

XPath Support

XPathFactory & XPathExpression: Two classes used to evaluate an XPath using string lookup in JDOM.

//Get an instance of the XPathFactory

```
XPathFactory xpf = XPathFactory.instance();
```

```
XPathExpression<Element> expr = xpf.compile("XPath",  
      <Filters>, <variables>,<namespaces>);
```

```
List<Element> books = expr.evaluate(xmlDoc);
```

XPathFactor is thread-safe.

XPathExpression isn't thread-safe.

XPath Support

XPathHelper class can be used to find the absolute or relative path to a content (element) or an attribute.

E.g.

`XPathHelper.getAbsolutePath(<Element reference of a Title>)`

returns

```
/*[local-name() = 'schemeOfWork' and namespace-uri() =  
  'http://www.example.org/sow']/*[local-name() =  
  'booksRecommended' and namespace-uri() =  
  'http://www.example.org/sow']/book/title
```

Can also use **XPathBuilder** to build the expression.

Beware of Namespaces in XPath.

XSLT USING TRAX

Transform XML using JAXP

Package used **java.xml.transform** and its subpackages.

Steps:

- Obtain a **TransformerFactory** object.
- Present the **stylesheet** as a **Source**.
- Use the factory object to obtain a **Transformer** object for the given stylesheet.
- Present the **input document** as a **Source**.
- Provide a **Result** object to accept the **output document**.
- Use the **Transformer** object to process the **input document**.

Transform API's for XML

```
TransformerFactory factory =  
    TransformerFactory.newInstance();  
StreamSource stylesheet = new  
    StreamSource(new File(xslFile));  
Transformer transformer =  
    factory.newTransformer(stylesheet);  
StreamSource source = new StreamSource(new  
    File(schemaXMLFile));  
StreamResult result = new StreamResult(new  
    File(targetHTMLFile));  
transformer.transform(source, result);
```

TrAX

TrAX provides tremendous flexibility in moving from various input types to various output types, and in using XSL stylesheets in a variety of formats, such as files, in-memory DOM trees, SAX readers, and so on.

Templates is used when a set of output properties is desired across multiple transformations, or when a set of transformation instructions can be used repeatedly.

XML SECURITY

XML Security

W3C states “Manipulating data with XML sometimes requires integrity, authentication and privacy”.

XML Security needed to build a “web of trust”.

XML Signatures: used to provide integrity & message authentication.

XML Encryption: used to provide privacy by encryption of messages.

XML Digital Signatures

Recco by the W3C on XML Digital Signature in February 2002.

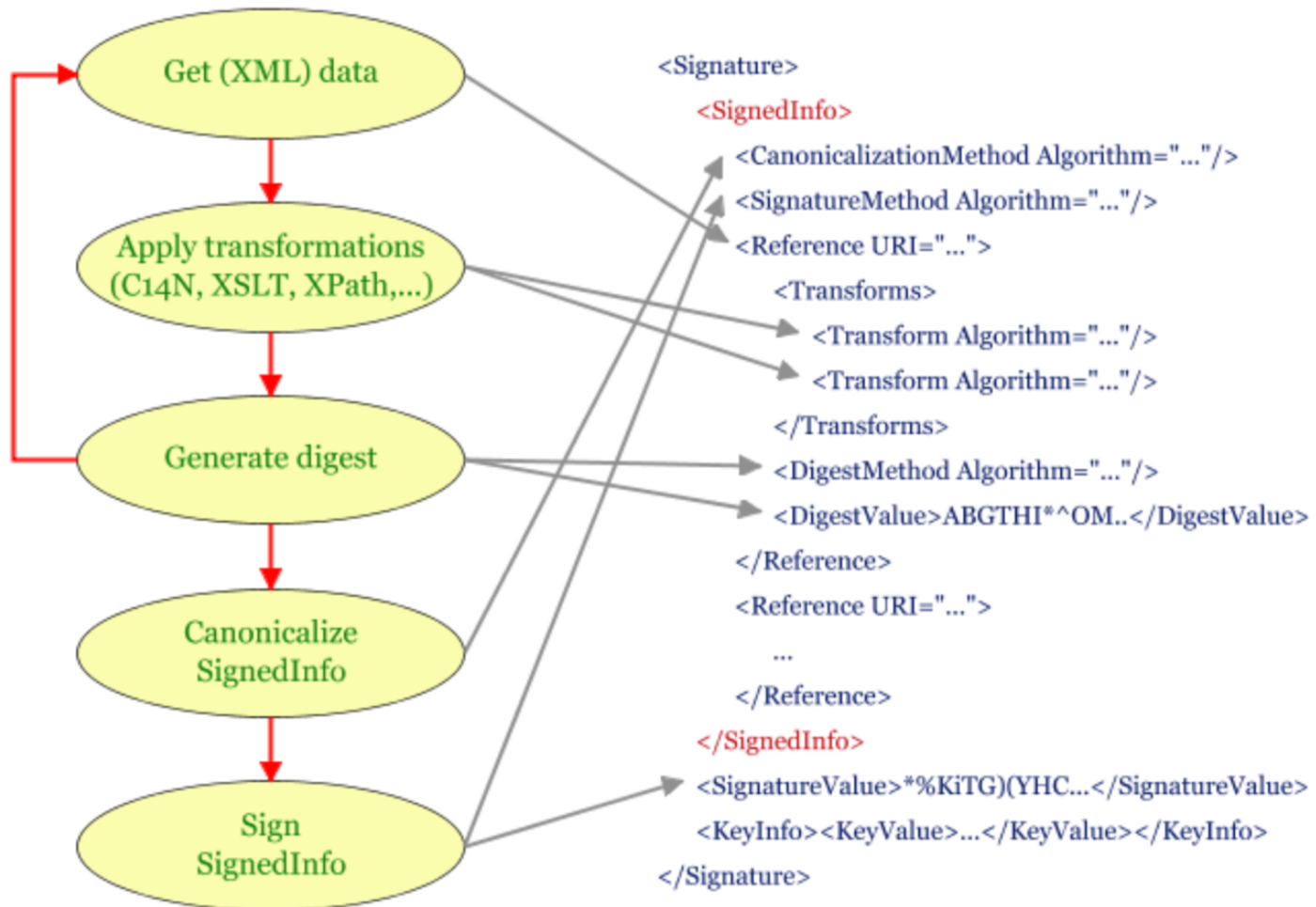
XML Digital Signature provides applications with authentication, data integrity and non-repudiation abilities.

XML Digital Signature can be applied to one or more items of digital content.

An XML Signature can sign:

- (part of the) same XML document which contains the signature (“enveloped”).
- another (XML) document, referenced through an URI (“detached”).

XML Signature - Example



Courtesy: W3C

XML Encryption

The XML Encryption specification satisfies confidentiality requirements in XML messages and followed XML Signatures.

XML Signatures points to what is being signed but XML Encryption specifies which elements are being encrypted.

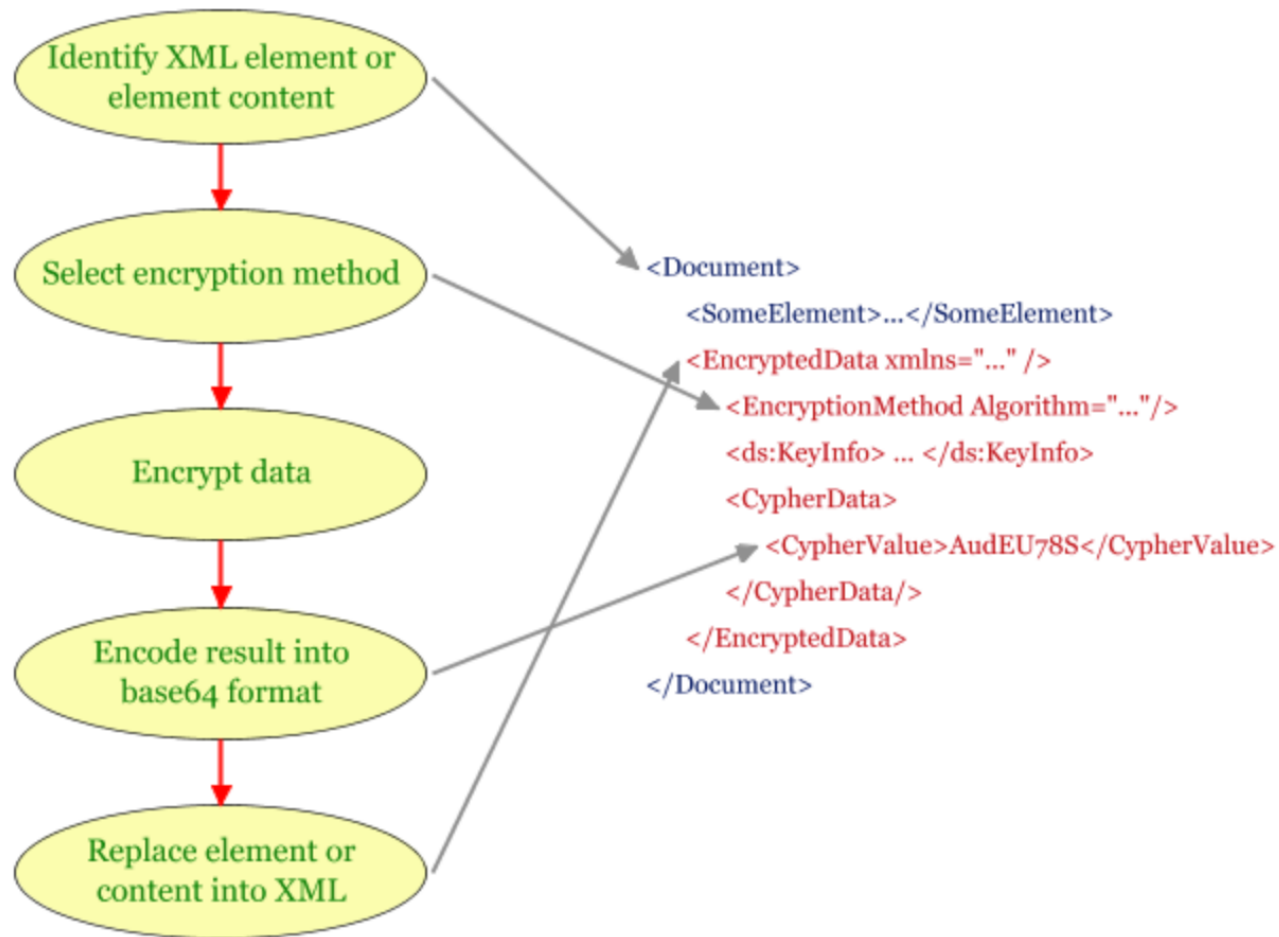
XML Encryption shares the Key information element with XML Signature.

Goal is to encrypt:

- a full XML tree rooted at an element;
- the content of an element; or
- only character data of an element.

The encrypted data *replaces* the original content.

XML Encryption - Example



Courtesy: W3C

XML Encryption Example

```
<PaymentInfo xmlns='http://example.org/paymentv2'>
```

```
  <Name>John Smith</Name>
```

```
  <CreditCard Limit='5,000' Currency='USD'>
```

```
    <Number>4019 2445 0277 5567</Number>
```

```
    <Issuer>Example Bank</Issuer>
```

```
    <Expiration>04/02</Expiration>
```

```
  </CreditCard>
```

```
</PaymentInfo>
```

```
<PaymentInfo xmlns='http://example.org/paymentv2'>
```

```
  <Name>John Smith</Name>
```

```
  <EncryptedData
```

```
    Type='http://www.w3.org/2001/04/xmlenc#Element'
```

```
    xmlns='http://www.w3.org/2001/04/xmlenc#'>
```

```
    <CipherData>
```

```
      <CipherValue>A23B45C56</CipherValue>
```

```
    </CipherData>
```

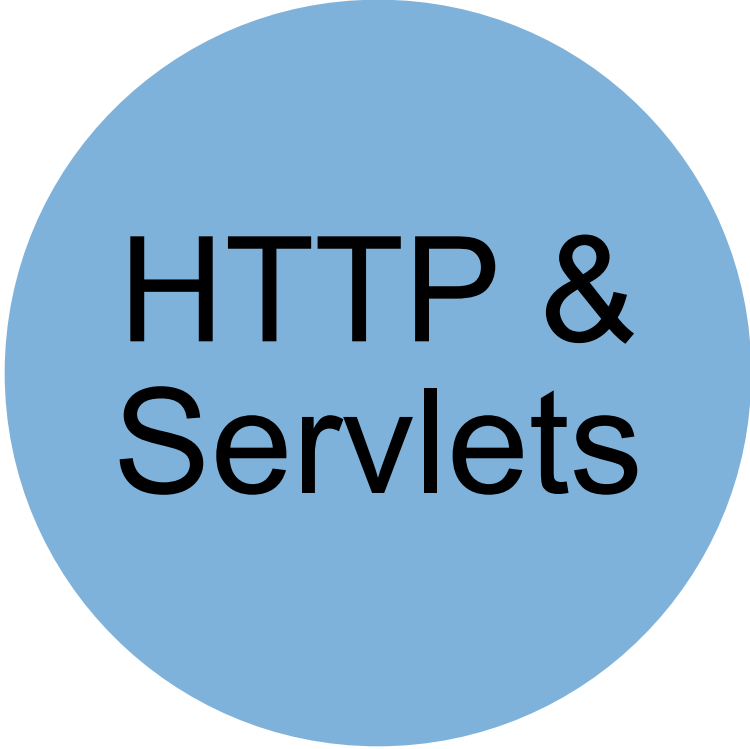
```
  </EncryptedData>
```

```
</PaymentInfo>
```

Summary

- JDOM provides API's to validate XML against an internal schema and an external schema
- Classes are available to associate elements with namespaces
- JDOM supports XPath evaluation
- JAXP has API's to perform XSL transformation on XML instances in 6 steps
- XML Security via XML Encryption and XML digital signatures is possible

Coming up...



HTTP &
Servlets