eclipseCON 2006

XML Binding with the Eclipse Modeling Framework

http://eclipse.org/emf/docs/presentations/EclipseCon/

Ed Merks
IBM Rational Software
Toronto, Canada
EMF Project

Agenda

- Introduction
- Binding XML with W3C's DOM
- Binding XML with EMF's AnyType
- Raising the level of abstraction with grammar
- Generating Java™ interfaces
- Reflecting on the landscape
- Conclusions
- Questions and answers

What's the problem?

- XML is widely accepted as a language-, platform-, vendor-neutral data persistence and data exchange mechanism
- Manipulating it efficiently remains an important problem
- There are many approaches
  - DOM, JAXB 1.0/2.0, XML Beans, Castor, SDO, EMF
- We'll focus on DOM and EMF because they are representative of the two ends of the spectrum

How does EMF solve the problem?

- EMF provides Ecore
  - A model for representing models, i.e., a meta model
  - Analogous to W3C XML Schema, and Java reflection
- EMF provides EObject
  - A model for representing instances
  - Analogous to W3C DOM Element, and Java Object

- EMF Ecore describes models
- W3C XML Schema describes valid XML structure
- Java reflection describes implementations

Let's be concrete

- We'll consider how to manipulate this instance document

```
<?xml version="1.0" encoding="UTF-8"?>
<tree:rootNode xmlns:tree="http://www.eclipse.org/emf/example/dom/Tree" label="root">
  <tree:childNode label="text">text</tree:childNode>
  <tree:childNode label="comment"><!--comment--></tree:childNode>
  <tree:childNode label="cdata"><![CDATA[<cdata>]]></tree:childNode>
</tree:rootNode>
```

Creating a DOM Document

- Set up the environment to create a Document

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
documentBuilderFactory.setNamespaceAware(true);
DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
Document document = documentBuilder.newDocument();
```

- Define some constants

```
final String NAMESPACE_URI = "http://www.eclipse.org/emf/example/dom/Tree";
final String NAMESPACE_PREFIX = "tree";
```

Creating a DOM root Element

- Use the Document to create and configure the root Element

```
Element rootTreeNode =
  document.createElementNS(NAMESPACE_URI, NAMESPACE_PREFIX + ":rootNode");
document.appendChild(rootTreeNode);
rootTreeNode.setAttributeNS
  ("http://www.w3.org/2000/xmlns/", "xmlns:" + NAMESPACE_PREFIX, NAMESPACE_URI);
rootTreeNode.setAttributeNS(null, "label", "root");
```

Creating DOM child Elements

- Use the Document to create each child and add it with formatting

```
Element textChildTreeNode =
  document.createElementNS(NAMESPACE_URI, NAMESPACE_PREFIX + ":childNode");
rootTreeNode.appendChild(document.createTextNode("\n  "));
rootTreeNode.appendChild(textChildTreeNode);
textChildTreeNode.setAttributeNS(null, "label", "text");
textChildTreeNode.appendChild(document.createTextNode("text"));
// ...
commentChildTreeNode.appendChild(document.createComment("comment"));
// ...
cdataChildTreeNode.appendChild(document.createCDATASection("<cdata>"));
//...
rootTreeNode.appendChild(document.createTextNode("\n"));
```

Saving a DOM Document

- Transform the Document into text

```
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
transformer.transform(new DOMSource(document), new StreamResult(System.out));
```

- Produces the desired result

```
<?xml version="1.0" encoding="UTF-8"?>
<tree:rootNode xmlns:tree="http://www.eclipse.org/emf/example/dom/Tree" label="root">
  <tree:childNode label="text">text</tree:childNode>
  <tree:childNode label="comment"><!--comment--></tree:childNode>
  <tree:childNode label="cdata"><![CDATA[<cdata>]]></tree:childNode>
</tree:rootNode>
```

Loading a DOM Document

- Set up the environment to parse a Document

```
String DATA_FOLDER="c:/data/";
  …
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
documentBuilderFactory.setNamespaceAware(true);
DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
Document document =
  documentBuilder.parse(new File(DATA_FOLDER + "DOMTreeNode.xml"));
```

Traversing the Elements of a DOM Document

- Recursively visit each Element starting with the Document's root

```
new Object()
{
  public void traverse(String indent, Element element)
  {
    System.out.println
      (indent + "{" + element.getNamespaceURI() + "}" + element.getLocalName());
    System.out.println(indent + "  label=" + element.getAttributeNS(null, "label"));
    for (Node child = element.getFirstChild(); child != null; child = child.getNextSibling())
      // Consider each type of child.
  }
}.traverse("",  document.getDocumentElement());
```

Handling the DOM Node types

- Switch on the type of node and visit each appropriately

```
switch (child.getNodeType())
{
  case Node.TEXT_NODE:
    System.out.println(indent + "  '" + child.getNodeValue().replaceAll("\n", "\\\\n") + "'");
    break;
  case Node.COMMENT_NODE:
    System.out.println(indent + "  <!--" + child.getNodeValue() + "-->");
    break;
  case Node.CDATA_SECTION_NODE:
    System.out.println(indent + "  <![CDATA[" + child.getNodeValue() + "]]>");
    break;
  case Node.ELEMENT_NODE:
    traverse(indent + "  ", (Element)child);
    break;
}
```

The resulting output

```
{http://www.eclipse.org/emf/example/dom/Tree}rootNode
    label=root
    '\n  '
    {http://www.eclipse.org/emf/example/dom/Tree}childNode
      label=text
      'text'
    '\n  '
    {http://www.eclipse.org/emf/example/dom/Tree}childNode
      label=comment
      <!--comment-->
    '\n  '
    {http://www.eclipse.org/emf/example/dom/Tree}childNode
      label=cdata
      <![CDATA[<cdata>]]>
    '\n'
```

Creating an EMF ResourceSet

- Set up the environment to create meta and instance data

```
ResourceSet resourceSet = new ResourceSetImpl();
ExtendedMetaData extendedMetaData =
  new BasicExtendedMetaData(resourceSet.getPackageRegistry());
resourceSet.getLoadOptions().put
  (XMLResource.OPTION_EXTENDED_META_DATA, extendedMetaData);
```

- Define some constants

```
String NAMESPACE_URI = "http://www.eclipse.org/emf/example/dom/Tree";
String NAMESPACE_PREFIX = "tree";
```

Creating an EMF document root

- Demand create the model and use it to create an instance

```
EStructuralFeature rootNodeFeature =
  extendedMetaData.demandFeature(NAMESPACE_URI, "rootNode", true);
EClass documentRootClass = rootNodeFeature.getEContainingClass();
EObject documentRoot = EcoreUtil.create(documentRootClass);
```

- Reflectively get the map of XMLNS declarations to update it

```
EMap xmlnsPrefixMap =
  (EMap)documentRoot.eGet
    (extendedMetaData.getXMLNSPrefixMapFeature(documentRootClass));
xmlnsPrefixMap.put(NAMESPACE_PREFIX, NAMESPACE_URI);
```

Creating the root EMF AnyType instance

- Create an instance of the type corresponding to XML Schema's anyType and add it to the document root

```
AnyType rootTreeNode = XMLTypeFactory.eINSTANCE.createAnyType();
documentRoot.eSet(rootNodeFeature, rootTreeNode);
```

- Demand create the attribute feature and use it to set the label

```
EStructuralFeature labelAttribute = extendedMetaData.demandFeature(null, "label", false);
rootTreeNode.eSet(labelAttribute, "root");
```

Creating child AnyType instances

- Use the root's mixed feature to add children and formatting

```
FeatureMap rootMixed = rootTreeNode.getMixed();
EStructuralFeature childNodeFeature =
  extendedMetaData.demandFeature(NAMESPACE_URI, "childNode", true);
AnyType textChildTreeNode = XMLTypeFactory.eINSTANCE.createAnyType();
FeatureMapUtil.addText(rootMixed, "\n  ");
rootMixed.add(childNodeFeature, textChildTreeNode);
textChildTreeNode.eSet(labelAttribute, "text");
FeatureMapUtil.addText(textChildTreeNode.getMixed(), "text");
// …
 FeatureMapUtil.addComment(commentChildTreeNode.getMixed(), "comment");
// …
FeatureMapUtil.addCDATA(cdataChildTreeNode.getMixed(), "<cdata>");
FeatureMapUtil.addText(rootMixed, "\n");
```

Saving with an EMF Resource

- Register a default resource factory

```
resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap().put
  (Resource.Factory.Registry.DEFAULT_EXTENSION,
   new GenericXMLResourceFactoryImpl());
```

- Create a resource and add to it the document root

```
Resource resource =
  resourceSet.createResource
    (URI.createFileURI(DATA_FOLDER + "EMFDOMTreeNode.xml"));
resource.getContents().add(documentRoot);
```

- Save the resource with default options

```
resource.save(System.out, null);
```

Saving a Resource as a DOM Document

- Use the specialized XML resource to save as a DOM document

```
Document document = ((XMLResource)resource).save(null, null, null);
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
transformer.transform(new DOMSource(document), new StreamResult(System.out));
```

- Either way produces the desired result

```
<?xml version="1.0" encoding="UTF-8"?>
<tree:rootNode xmlns:tree="http://www.eclipse.org/emf/example/dom/Tree" label="root">
  <tree:childNode label="text">text</tree:childNode>
  <tree:childNode label="comment"><!--comment--></tree:childNode>
  <tree:childNode label="cdata"><![CDATA[<cdata>]]></tree:childNode>
</tree:rootNode>
```

Loading a Resource

- Using the same environment, demand load the Resource to fetch the document root

```
Resource resource =
  resourceSet.getResource
    (URI.createFileURI(DATA_FOLDER + "EMFDOMTreeNode.xml"), true);
EObject documentRoot = (EObject)resource.getContents().get(0);
AnyType rootTreeNode = (AnyType)documentRoot.eContents().get(0);
```

Traversing an EMF AnyType instance

- Recursively visit each AnyType instance starting with the root

```
new Object()
{
  public void traverse(String indent, AnyType anyType)
  {
    System.out.println
      (indent + "{" + extendedMetaData.getNamespace(anyType.eContainmentFeature())
                + "}" + extendedMetaData.getName(anyType.eContainmentFeature()));
    System.out.println(indent + "  label=" + anyType.eGet(labelAttribute));
    FeatureMap featureMap = anyType.getMixed();
    for (int i = 0, size = featureMap.size(); i < size; ++i)
      // Consider each entry
  }
}.traverse("", rootTreeNode);
```

Handling EMF FeatureMap content for mixed data

- Switch on the type of feature and visit each value appropriately

```
EStructuralFeature feature = featureMap.getEStructuralFeature(i);
if (FeatureMapUtil.isText(feature))
  System.out.println
    (indent + "  '" + featureMap.getValue(i).toString().replaceAll("\n", "\\\\n") + "'");
else if (FeatureMapUtil.isComment(feature))
  System.out.println(indent + "  <!--" +  featureMap.getValue(i) + "-->");
else if (FeatureMapUtil.isCDATA(feature))
  System.out.println(indent + "  <![CDATA[" + featureMap.getValue(i) + "]]>");
else if (feature instanceof EReference)
  traverse(indent + "  ", (AnyType)featureMap.getValue(i));
```

The resulting output just

```
{http://www.eclipse.org/emf/example/dom/Tree}rootNode
    label=root
    '\n  '
    {http://www.eclipse.org/emf/example/dom/Tree}childNode
      label=text
      'text'
    '\n  '
    {http://www.eclipse.org/emf/example/dom/Tree}childNode
      label=comment
      <!--comment-->
    '\n  '
    {http://www.eclipse.org/emf/example/dom/Tree}childNode
      label=cdata
      <![CDATA[<cdata>]]>
    '\n'
```

Why is neither approach very satisfying?

- Both focus on representing the totality of the XML infoset which is complex
- Different realizations of the same underlying model will tend to look similar and have similar complexity
- The model seems to provide little additional value to justify its complexity
- EMF doesn't appear to be a better DOM

What do you expect?

- The problem is how we've defined the problem
- The instance is being processed as if its model were specified by this schema

```
<xsd:schema  xmlns:tree="http://www.eclipse.org/emf/example/dom/Tree"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.eclipse.org/emf/example/dom/Tree">
   <xsd:element name="rootNode" type="xsd:anyType"/>
  </xsd:schema>
```

The anyType allows anything

- W3C XML Schema 1.0 defines the anyType as follows

```
<xsd:complexType name="anyType" mixed="true">
  <xsd:sequence>
    <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xsd:sequence>
  <xsd:anyAttribute processContents="lax"/>
</xsd:complexType>
```

- EMF maps that model as follows

```
public interface AnyType extends EObject
{
    FeatureMap getMixed();
    FeatureMap getAny();
    FeatureMap getAnyAttribute();
}
```

We're expecting something more specific

- The following is a much better model of what's expected

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema  xmlns:tree="http://www.eclipse.org/emf/example/dom/Tree"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.eclipse.org/emf/example/dom/Tree">

 <xsd:complexType mixed="true" name="TreeNode">
  <xsd:sequence>
   <xsd:element form="qualified" maxOccurs="unbounded"
       name="childNode" type="tree:TreeNode"/>
  </xsd:sequence>
  <xsd:attribute name="label" type="xsd:ID"/>
 </xsd:complexType>

 <xsd:element name="rootNode" type="tree:TreeNode"/>

</xsd:schema>
```

EMF supports typed references

- Define an attribute to hold href-like cross references

```
<?xml version="1.0" encoding="UTF-8"?>
  <xsd:schema xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
      xmlns:tree="http://www.eclipse.org/emf/example/dom/Tree"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.eclipse.org/emf/example/dom/Tree">
    <xsd:complexType mixed="true" name="TreeNode">
      <xsd:sequence>
        <xsd:element ecore:name="childNodes" form="qualified" maxOccurs="unbounded"
            name="childNode" type="tree:TreeNode"/>
      </xsd:sequence>
      <xsd:attribute name="label" type="xsd:ID"/>
      <xsd:attribute ecore:reference="tree:TreeNode" name="references">
        <xsd:simpleType>
          <xsd:list itemType="xsd:anyURI"/>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
    <xsd:element name="rootNode" type="tree:TreeNode"/>
  </xsd:schema>
```

The value of modeling is design reuse

- EMF provides XSD model to represent instances of XML Schemas
- Creating an XML Schema instance is just like creating an AnyType

```
XSDSchema xsdSchema = XSDFactory.eINSTANCE.createXSDSchema();
xsdSchema.setTargetNamespace(NAMESPACE_URI);
```

- Load a schema instance and convert it to Ecore

```
XSDEcoreBuilder xsdEcoreBuilder = new XSDEcoreBuilder(extendedMetaData);
URI schemaLocationURI =
  URI.createFileURI
    (new File(MODEL_FOLDER + "DOMEMFTreeNode.xsd").getAbsolutePath());
xsdEcoreBuilder.generate(schemaLocationURI);
EPackage ePackage = extendedMetaData.getPackage(NAMESPACE_URI);
```

What's in an Ecore EPackage?

```
System.out.println("package " + ePackage.getNsURI());
for (Iterator i = ePackage.getEClassifiers().iterator(); i.hasNext(); )
{
  EClassifier eClassifier = (EClassifier)i.next();
  if (eClassifier instanceof EClass)
  {
    EClass eClass = (EClass)eClassifier;
    System.out.println("  class " + eClass.getName());
    for (Iterator j = eClass.getEStructuralFeatures().iterator(); j.hasNext(); )
      // Handle each feature
  }
  else
  {
    EDataType eDataType = (EDataType)eClassifier;
    System.out.println("  data type " + eDataType.getName());
  }
}
```

What's in an Ecore EClass?

```
EStructuralFeature eStructuralFeature = (EStructuralFeature)j.next();
if (eStructuralFeature instanceof EReference)
{
  EReference eReference = (EReference)eStructuralFeature;
  System.out.println
    ("    reference " + eReference.getName() + ":" +
                        eReference.getEReferenceType().getName());
}
else
{
  EAttribute eAttribute = (EAttribute)eStructuralFeature;
  System.out.println
    ("    attribute " + eAttribute.getName() + ":" + eAttribute.getEAttributeType().getName());
}
```

What's in the Ecore Tree Node model?

package **http://www.eclipse.org/emf/example/dom/Tree**
- class **DocumentRoot**
  - attribute **mixed**:EFeatureMapEntry
  - reference **xMLNSPrefixMap**:EStringToStringMapEntry
  - reference **xSISchemaLocation**:EStringToStringMapEntry
  - reference **rootNode**:TreeNode
- class **TreeNode**
  - attribute **mixed**:EFeatureMapEntry
  - reference **childNodes**:TreeNode
  - attribute **label**:ID
  - reference **references**:TreeNode

Loading using the strongly typed Tree Node model

- Load as before and verify that the root is of the expected type

```
if (rootTreeNode.eClass() != extendedMetaData.getType(NAMESPACE_URI, "TreeNode"))
{
  throw new Exception("Bad meta data");
}
```

Cache the tree node's features for use while traversing the instance

```
EStructuralFeature mixedFeature =
  extendedMetaData.getMixedFeature(rootTreeNode.eClass());
EStructuralFeature labelAttribute =
  extendedMetaData.getAttribute(rootTreeNode.eClass(), null, "label");
EStructuralFeature referencesAttribute =
  extendedMetaData.getAttribute(rootTreeNode.eClass(), null, "references");
```

Traversing tree nodes using EObject reflection

- Traverse as before but using EObject reflection

```
System.out.println(indent + "  label=" + eObject.eGet(labelAttribute));
FeatureMap featureMap = (FeatureMap)eObject.eGet(mixedFeature);
```

- Set each tree node to reference the root node

```
((List)eObject.eGet(referencesAttribute)).add(rootTreeNode);
```

- Save the result to produce

```
<?xml version="1.0" encoding="UTF-8"?>
  <tree:rootNode xmlns:tree="http://www.eclipse.org/emf/example/dom/Tree" label="root"
      references="#root">
    <tree:childNode label="text" references="#root">text</tree:childNode>
    <tree:childNode label="comment" references="#root"><!--comment--></tree:childNode>
    <tree:childNode label="cdata" references="#root"><![CDATA[<cdata>]]></tree:childNode>
  </tree:rootNode>
```

Binding Ecore to Java

- Ecore instances have a simple mapping onto Java

```
public interface TreeNode extends EObject
{
    FeatureMap getMixed();
    EList getChildNodes();
    String getLabel();
    void setLabel(String value);
    EList getReferences();
}

public interface DocumentRoot extends EObject
{
    FeatureMap getMixed();
    EMap getXMLNSPrefixMap();
    EMap getXSISchemaLocation();
    TreeNode getRootNode();
    void setRootNode(TreeNode value);
}
```

Using the generated API

- Load the XML as an instance of the generated Java model

```
resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap().put
  (Resource.Factory.Registry.DEFAULT_EXTENSION,
   new TreeResourceFactoryImpl());
resourceSet.getPackageRegistry().put(TreePackage.eNS_URI, TreePackage.eINSTANCE);
Resource resource =
  resourceSet.getResource
   (URI.createFileURI(DATA_FOLDER + "EMFDOMTreeNodeWithReferences.xml"), true);
DocumentRoot documentRoot = (DocumentRoot)resource.getContents().get(0);
TreeNode rootTreeNode = documentRoot.getRootNode();
```

Doing cool things

- Use EMF's reflective copier to copy the document root

```
EcoreUtil.Copier copier = new EcoreUtil.Copier();
DocumentRoot documentRootCopy = (DocumentRoot)copier.copy(documentRoot);
copier.copyReferences();
```

- Traverse recursively as before and make the copy reference the original

```
TreeNode treeNodeCopy = (TreeNode)copier.get(treeNode);
treeNodeCopy.getReferences().add(treeNode);
```

EMF supports cross document references

- Cross document references are represented as proxies that are loaded on demand

```
<?xml version="1.0" encoding="UTF-8"?>
<tree:rootNode xmlns:tree="http://www.eclipse.org/emf/example/dom/Tree" label="root"
    references="#root TreeNodeWithReferences.xml#root">
  <tree:childNode label="text"
      references="#root TreeNodeWithReferences.xml#text">text</tree:childNode>
  <tree:childNode label="comment"
      references="#root TreeNodeWithReferences.xml#comment"><!--comment--></tree:childNode>
  <tree:childNode label="cdata"
      references="#root TreeNodeWithReferences.xml#cdata"><![CDATA[<cdata>]]></tree:childNode>
  </tree:rootNode>
```

Reflecting on the landscape

- Let’s step way back and look at where we’ve been

How many ways are there to bind XML Schema?

- Alternative solutions to the same problems tend to look the same
- For the example schema (using IDREF not anyURI), JAXB 2.0 generates

```
public class TreeNode
{
  public List<Serializable> getContent() {}
  public String getLabel() {}
  public void setLabel(String value) {}
  public List<Object> getReferences() {}
}
```

- When mixed content is removed from the picture, EMF and JAXB 2.0 yield effectively identical APIs

Only the complex things are done differently

- XML and XML Schema's subtle complexities tend to yield suboptimal bindings in essentially all approaches
  - JAXB 2.0's JAXBElement and EMF's FeatureMap and are both a result of the need to handle mixed content, wildcards, and substitution groups
  - Unlike for EMF, for JAXB 2.0, the childNode element does not result a generated accessor, so getContent() and JAXBElement must be used; EMF supports both getMixed() and getChildNodes()
  - Mismatches with Java's type system

Is all the complexity necessary?

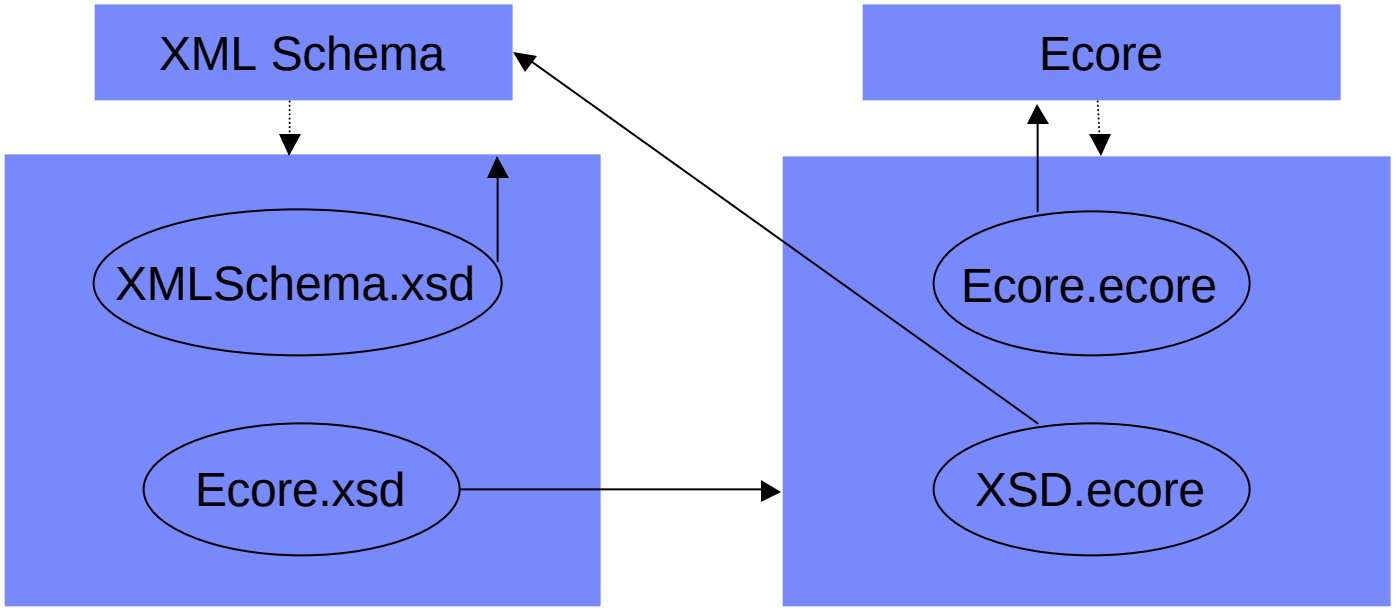
- Not necessarily; simple things can be done simply
- EMF provides an implicit mapping between EObject and XML as well as between Ecore and XML Schema, so one can start with the simple abstract models and generate the complex create things as necessary
- Beautiful concrete syntax is only skin deep, but ugly abstract syntax goes right to the bone

Assimilation: A model is a model is a model

- EMF's application to both pure XML and XML Schema binding has arisen as a natural evolutionary consequence of a powerful representation's ability to assimilate other models and other data
- Ecore is simpler than XML Schema and yet more powerful
  - typed references
  - multiple inheritance support
- EObject is simpler than DOM and yet more powerful
  - type-safe efficient reflective access
- It should come as no surprise that the XML binding problem can be subsumed as a modeling problem

The model of a model is at the core

- In the universe of all models, the simplest self describing model plays a singularly unique role as the one model that binds all other models



Questions?

- Eclipse EMF Help
  - overviews, tutorials, API reference
- EMF/XSD Project Web Site
  - http://www.eclipse.org/emf/
  - http://www.eclipse.org/xsd/
  - documentation, newsgroup, mailing list, Bugzilla
- Eclipse Modeling Framework by Frank Budinsky et al.
  - Addison-Wesley; 1st edition (August 13, 2003)
  - ISBN: 0131425420