Integrating XML into the Java™ Programming Language

Mark Reinhold

Java SE Chief Engineer
Sun Microsystems

TS-3441

P8Y7M3D

P8Y7M3D

http://www.w3.org/TR/1998/REC-xml-19980210

SAX
W3C DOM
JDOM
Castor
Zeus
dom4j
JAXB
XMLPULL
XMLBeans
StAX
XOM

Writing XML applications is still painful.

"Huh? What about data-binding tools?"

E.g., the Java™ Architecture for XML Binding (JAXB), JBind, Castor, XMLBeans, Zeus, *etc.*

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="Review">
    <xsd:sequence>
      <xsd:element name="who" type="xsd:string"/>
      <xsd:element name="when" type="xsd:dateTime"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Feature">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:integer"/>
      <xsd:element name="rfe" type="xsd:integer"/>
      <xsd:element name="votes" type="xsd:integer" minOccurs="0"/>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="engineer" type="xsd:string"/>
      <xsd:element name="release" type="xsd:string"/>
      <xsd:element name="state" type="xsd:string"/>
      <xsd:element name="reviewed" type="Review" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="feature" type="Feature"/>

</xsd:schema>
```

(This page intentionally left blank, except for this annoying little self-referential sentence.)

ad hoc /ad-'häk, -'hōc/ *adj*. [L, ‘for this’] For just this particular end or purpose.

Writing *ad-hoc* XML applications is still painful.

Writing *ad-hoc* XML applications is still painful.

Integrating XML directly into the language can help.

Where is the pain?

Construction

Conversion

Navigation

Streaming

Where is the pain?

Construction

Conversion

Navigation

Streaming

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
</feature>
```

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07T13:44:25.000-08:00</when>
  </reviewed>
</feature>
```

```
void addReviewer(Element feature,
                 String reviewer, String time)
{
   ...
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.w3c.dom.*;       // W3C DOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  Document doc = feature.getOwnerDocument();
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(user);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07T13:44:25.000-08:00</when>
  </reviewed>
</feature>
```

```
Element newFeature(Document doc,
                   String name, String engineer,
                   String reviewer, String time)
{
   ...
}
```

```
import org.w3c.dom.*;       // W3C DOM

Element newFeature(Document doc,
                   String name, String engineer,
                   String reviewer, String time)
{
  Element feature = doc.createElement("feature");
  Element id = doc.createElement("id");
  id.setTextContent(Integer.toString(++nFeatures));
  feature.appendChild(id);
  Element n = doc.createElement("name");
  n.setTextContent(name);
  feature.appendChild(n);
  Element e = doc.createElement("engineer");
  e.setTextContent(engineer);
  feature.appendChild(e);
  Element s = doc.createElement("state");
  s.setTextContent("submitted");
  feature.appendChild(s);
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(reviewer);
  review.appendChild(who);
  Element when = doc.createElement("when");
```

```
import org.w3c.dom.*;                    // W3C DOM

Element newFeature(Document doc,
                   String name, String engineer,
                   String reviewer, String time)
{
  Element feature = doc.createElement("feature");
  Element id = doc.createElement("id");
  id.setTextContent(Integer.toString(++nFeatures));
  feature.appendChild(id);
  Element n = doc.createElement("name");
  n.setTextContent(name);
  feature.appendChild(n);
  Element e = doc.createElement("engineer");
  e.setTextContent(engineer);
  feature.appendChild(e);
  Element s = doc.createElement("state");
  s.setTextContent("submitted");
  feature.appendChild(s);
  Element review = doc.createElement("reviewed");
  Element who = doc.createElement("who");
  who.setTextContent(reviewer);
  review.appendChild(who);
  Element when = doc.createElement("when");
  when.setTextContent(time);
  review.appendChild(when);
  feature.appendChild(review);
  return feature;
}
```

```
import org.w3c.dom.*;                    // W3C DOM

Element newTextElement(Document doc,
                       String tag, String text)
{
  Element e = doc.createElement(tag);
  e.setTextContent(text);
  return e;
}

Element newFeature(Document doc,
                   String name, String engineer,
                   String reviewer, String time)
{
  Element f = doc.createElement("feature");
  int id = ++nFeatures;
  f.appendChild(newTextElement(doc, "id",
                               Integer.toString(id)));
  f.appendChild(newTextElement(doc, "name", name));
  f.appendChild(newTextElement(doc, "engineer", engineer));
  f.appendChild(newTextElement(doc, "state", "submitted"));
  Element review = doc.createElement("reviewed");
  review.appendChild(newTextElement(doc, "who", reviewer));
  review.appendChild(newTextElement(doc, "when", time));
  f.appendChild(review);
  return f;
}
```

```
import org.w3c.dom.*;                    // W3C DOM

Element newTextElement(Document doc,
                       String tag, String text)
{
  Element e = doc.createElement(tag);
  e.setTextContent(text);
  return e;
}

Element newFeature(Document doc,
                   String name, String engineer,
                   String reviewer, String time)
{
  Element f = doc.createElement("feature");
  int id = ++nFeatures;
  f.appendChild(newTextElement(doc, "id",
                               Integer.toString(id)));
  f.appendChild(newTextElement(doc, "name", name));
  f.appendChild(newTextElement(doc, "engineer", engineer));
  f.appendChild(newTextElement(doc, "state", "submitted"));
  Element review = doc.createElement("reviewed");
  review.appendChild(newTextElement(doc, "who", reviewer));
  review.appendChild(newTextElement(doc, "when", time));
  f.appendChild(review);
  return f;
}
```


DOM

```
import org.jdom.*;                          // JDOM

Element newFeature(String name, String engineer,
                   String reviewer, String time)
{
  int id = ++nFeatures;
  return new Element("feature")
    .addContent(new Element("id")
                .addContent(Integer.toString(id)))
    .addContent(new Element("name")
                .addContent(name))
    .addContent(new Element("engineer")
                .addContent(engineer))
    .addContent(new Element("state")
                .addContent("submitted"))
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import org.jdom.*;              // JDOM

Element newFeature(String name, String engineer,
                   String reviewer, String time)
{
  int id = ++nFeatures;
  return new Element("feature")
    .addContent(new Element("id")
                .addContent(Integer.toString(id)))
    .addContent(new Element("name")
                .addContent(name))
    .addContent(new Element("engineer")
                .addContent(engineer))
    .addContent(new Element("state")
                .addContent("submitted"))
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07T13:44:25.000-08:00</when>
  </reviewed>
</feature>
```

Where is the pain?

Construction

Conversion

Navigation

Streaming

```
void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
import java.sql.Timestamp;

void addReviewer(Element feature,
                 String reviewer, Timestamp time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(??)));
}
```

```
import java.sql.Timestamp;

void addReviewer(Element feature,
                 String reviewer, Timestamp time)
{
  String ts = time.toString();
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(ts)));
}
```

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07 13:44:25.0</when>
  </reviewed>
</feature>
```

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07 13:44:25.0</when>
    <when>2004-11-07T13:44:25.000-08:00</when>
  </reviewed>
</feature>
```

```
import java.sql.Timestamp;
import javax.xml.datatype.*;

void addReviewer(Element feature,
                 String reviewer, Timestamp time)
{
  GregorianCalendar gc = new GregorianCalendar();
  gc.setTimeInMillis(time.getTime());
  DatatypeFactory df
    = DatatypeFactory.newInstance();
  XMLGregorianCalendar xc
    = df.newXMLGregorianCalendar(gc);
  String ts = xc.toXMLFormat();
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(ts)));
}
```

```
import java.sql.Timestamp;
import javax.xml.datatype.*;

void addReviewer(Element feature,
                 String reviewer, Timestamp time)
  throws DatatypeConfigurationException
{
  GregorianCalendar gc = new GregorianCalendar();
  gc.setTimeInMillis(time.getTime());
  DatatypeFactory df
    = DatatypeFactory.newInstance();
  XMLGregorianCalendar xc
    = df.newXMLGregorianCalendar(gc);
  String ts = xc.toXMLFormat();
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(ts)));
}
```

```
import java.sql.Timestamp;
import javax.xml.datatype.*;

Timestamp getReviewTime(Element feature)
  throws DatatypeConfigurationException
{
  Element r = feature.getChild("reviewed");
  if (r == null)
    return null;
  String ts = r.getChildText("when");
  DatatypeFactory df
    = DatatypeFactory.newInstance();
  XMLGregorianCalendar xc
    = df.newXMLGregorianCalendar(ts);
  return new Timestamp(xc.toGregorianCalendar()
                         .getTimeInMillis());
}
```

Where is the pain?

Construction

Conversion

Navigation

Streaming

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>42</id>
    <name>Zombie references</name>
    <engineer>mark.reinhold</engineer>
    <state>submitted</state>
  </feature>
</feature-list>
```

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>42</id>
    <name>Zombie references</name>
    <engineer>mark.reinhold</engineer>
    <state>rejected</state>
  </feature>
</feature-list>
```

```
void rejectOpenFeatures(Element featureList) {
   ...
}
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
% javac Features.java
Note: Features.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
%
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
% javac -Xlint:unchecked Features.java
Features.java:198: warning: unchecked conversion
found   : java.util.List
required: java.util.List<org.jdom.Element>
  List<Element> fs = featureList.getChildren("feature");
                                             ^
1 warning
%
```

```
@SuppressWarnings("unchecked")
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Element f : fs) {
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(Element featureList) {
  List<Element> fs = featureList.getChildren("feature");
  for (Object fo : fs) {
    Element f = (Element)fo;
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(Element featureList) {
  List fs = featureList.getChildren("feature");
  for (Object fo : fs) {
    Element f = (Element)fo;
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
import org.jdom.xpath.*;

void rejectOpenFeatures(Element featureList) {
  List fs
    = XPath.selectNodes(featureList,
                        "feature[state!='approved']");
  for (Object fo : fs) {
    Element f = (Element)fo;
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
    f.getChild("state").setText("rejected");
  }
}
```

```
import org.jdom.xpath.*;

void rejectOpenFeatures(Element featureList) {
  List fs
    = XPath.selectNodes(featureList,
                        "feature[state!='approved']");
  for (Object fo : fs) {
    Element f = (Element)fo;
    f.getChild("state").setText("rejected");
  }
}
```

```
import org.jdom.xpath.*;

void rejectOpenFeatures(Element featureList)
  throws JDOMException
{
  List fs
    = XPath.selectNodes(featureList,
                        "feature[state!='approved']");
  for (Object fo : fs) {
    Element f = (Element)fo;
    f.getChild("state").setText("rejected");
  }
}
```

Where is the pain?

Construction

Conversion

Navigation

Streaming

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>42</id>
    <name>Zombie references</name>
    <engineer>mark.reinhold</engineer>
    <state>submitted</state>
  </feature>
</feature-list>
```

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature> ...
```

```
Current Mustang feature list
 29: Method to find free disk space
201: Improve painting (fix gray boxes)
```

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature> ...
```

```
Current Mustang feature list
 29: Method to find free disk space
201: Improve painting (fix gray boxes)
```

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature> ...
```

```
Current Mustang feature list
 29: Method to find free disk space
201: Improve painting (fix gray boxes)
```

```
import javax.xml.stream.*;    // StAX (JSR 173)

void summarize(InputStream in, Formatter out)
  throws XMLStreamException
{
  XMLInputFactory xif = XMLInputFactory.newInstance();
  XMLStreamReader xr = xif.createXMLStreamReader(in);
  xr.nextTag();
  xr.require(START_ELEMENT, null, "feature-list");
  xr.nextTag();
  xr.require(START_ELEMENT, null, "release");
  out.format("Current %s feature list%n",
             xr.getElementText());
  xr.require(END_ELEMENT, null, "release");
  while (xr.nextTag() == START_ELEMENT) {
    xr.require(START_ELEMENT, null, "feature");
    String i = null;
    String s = null;
    String n = null;
    while (xr.nextTag() == START_ELEMENT) {
      String ln = xr.getLocalName();
      String t = xr.getElementText();
      if (ln.equals("id")) i = t;
      else if (ln.equals("state")) s = t;
      else if (ln.equals("name")) n = t;
```

```
import javax.xml.stream.*;                    // StAX (JSR 173)

void summarize(InputStream in, Formatter out)
  throws XMLStreamException
{
  XMLInputFactory xif = XMLInputFactory.newInstance();
  XMLStreamReader xr = xif.createXMLStreamReader(in);
  xr.nextTag();
  xr.require(START_ELEMENT, null, "feature-list");
  xr.nextTag();
  xr.require(START_ELEMENT, null, "release");
  out.format("Current %s feature list%n",
             xr.getElementText());
  xr.require(END_ELEMENT, null, "release");
  while (xr.nextTag() == START_ELEMENT) {
    xr.require(START_ELEMENT, null, "feature");
    String i = null;
    String s = null;
    String n = null;
    while (xr.nextTag() == START_ELEMENT) {
      String ln = xr.getLocalName();
      String t = xr.getElementText();
      if (ln.equals("id")) i = t;
      else if (ln.equals("state")) s = t;
      else if (ln.equals("name")) n = t;
      xr.require(END_ELEMENT, null, ln);
    }
    xr.require(END_ELEMENT, null, "feature");
    if (s.equals("approved")) {
      out.format("%3s: %s%n", i, n);
    }
  }
  xr.require(END_ELEMENT, null, "feature-list");
  xr.close();
}
```

```
<html>
  <body>
    <table>
      <caption>Current Mustang feature list</caption>
      <tr>
        <td>29</td>
        <td>Method to find free disk space</td>
      </tr>
      <tr>
        <td>201</td>
        <td>Improve painting (fix gray boxes)</td>
      </tr>
    </table>
  </body>
</html>
```

```
import javax.xml.stream.*;    // StAX (JSR 173)

void summarizeToXHTML(InputStream in, OutputStream out)
  throws XMLStreamException
{
  XMLInputFactory xif = XMLInputFactory.newInstance();
  XMLStreamReader xr = xif.createXMLStreamReader(in);
  XMLOutputFactory xof = XMLOutputFactory.newInstance();
  XMLStreamWriter xw = xof.createXMLStreamWriter(out, "UTF-8");
  xw.writeStartDocument("UTF-8", "1.0");
  xw.writeStartElement("html");
  xw.writeStartElement("body");
  xw.writeStartElement("table");
  xr.nextTag();
  xr.require(START_ELEMENT, null, "feature-list");
  xr.nextTag();
  xr.require(START_ELEMENT, null, "release");
  xw.writeStartElement("caption");
  xw.writeCharacters("Current " + xr.getElementText()
                     + " feature list");
  xw.writeEndElement();
  xr.require(END_ELEMENT, null, "release");
  while (xr.nextTag() == START_ELEMENT) {
    xr.require(START_ELEMENT, null, "feature");
    String i = null;
    String n = null;
    String s = null;
    while (xr.nextTag() == START_ELEMENT) {
      String ln = xr.getLocalName();
      String t = xr.getElementText();
      if (ln.equals("id")) i = t;
      else if (ln.equals("name")) n = t;
      else if (ln.equals("state")) s = t;
      xr.require(END_ELEMENT, null, ln);
    }
    xr.require(END_ELEMENT, null, "feature");
    if (s.equals("approved")) {
      xw.writeStartElement("tr");
      xw.writeStartElement("td");
      xw.writeCharacters(i);
      xw.writeEndElement();
      xw.writeStartElement("td");
      xw.writeCharacters(n);
      xw.writeEndElement();
      xw.writeEndElement();
    }
  }
  xr.require(END_ELEMENT, null, "feature-list");
  xr.close();
  xw.writeEndElement();
  xw.writeEndElement();
  xw.writeEndElement();
  xw.writeEndDocument();
  xw.flush();
}
```

Where is the pain?

Construction

Conversion

Navigation

Streaming

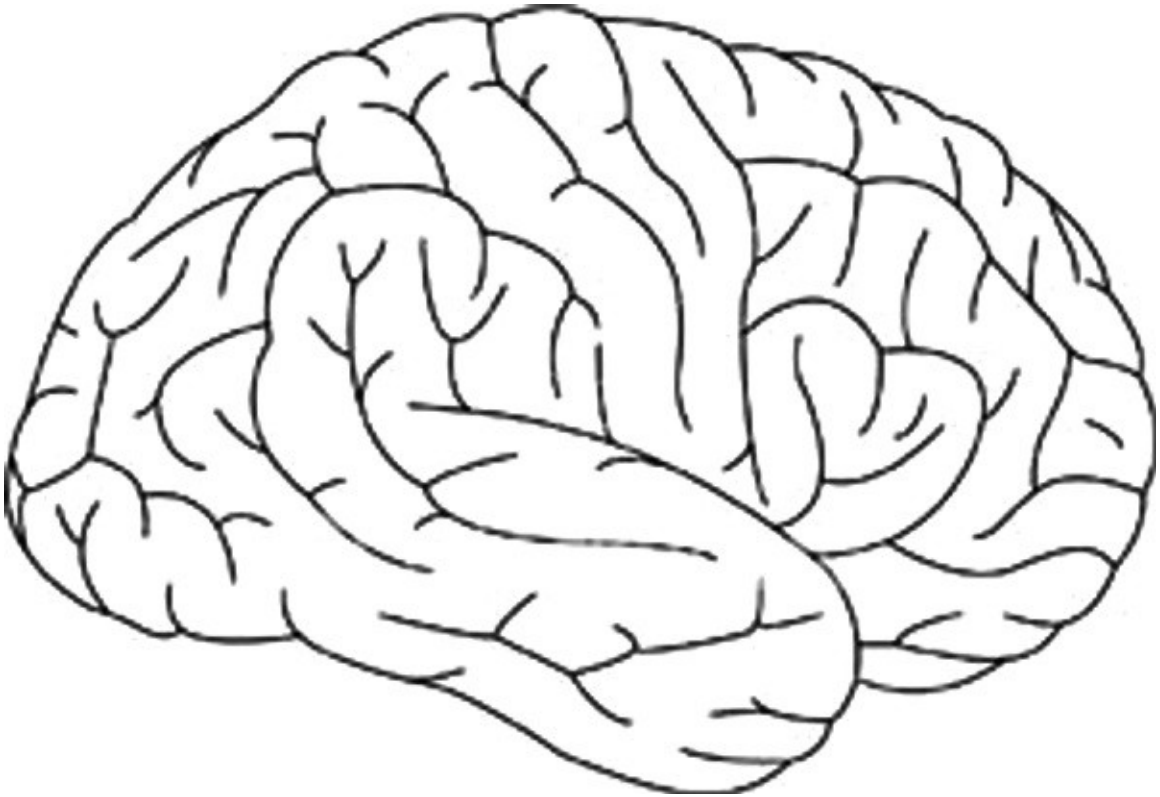
“What about dom4j, or XOM?”

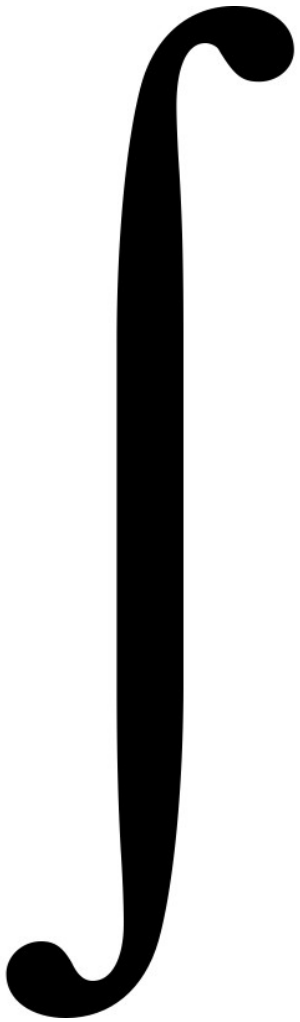
	DOM	JDOM	dom4j	XOM
Well-formed only	✗	✗	✗	✓
Content, not syntax	✗	✗	✗	✓
Streaming	✗	✗	✓	✓
Collections	✗	✓	✓	✗
Generics	✗	✗	✗	✗

Writing *ad-hoc* XML applications is still painful.

Writing *ad-hoc* XML applications is still painful.

Integrating XML directly into the language can help.





boolean
byte
short
char
int
long
float
double

java.lang.String

java.lang.XML

```
java.lang.String  "foo"

    java.lang.XML  <foo/>
```

```
java.lang.String  "foo"

    java.lang.XML  <foo/>
    java.xml.*
```

```
java.lang.String   "foo"

   java.lang.XML   <foo/>
   java.xml.*
```

New Document Object Model

New Syntax

java.lang.XML

Construction	*XML Literals*
Conversion	*Datatype Coders*
Navigation	*Collections* + *Generics* + *Paths*
Streaming	*Hybrid Event/Tree API*

java.lang.XML

Construction	***XML Literals***
Conversion	*Datatype Coders*
Navigation	*Collections* *+ Generics* *+ Paths*
Streaming	*Hybrid Event/Tree API*

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
</feature>
```

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07T13:44:25.000-08:00</when>
  </reviewed>
</feature>
```

```
import org.jdom.*;              // JDOM

void addReviewer(Element feature,
                 String reviewer, String time)
{
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
void addReviewer(XML feature,
                 String reviewer, String time)
{
  feature.add(<reviewed>
                <who>{ user }</who>
                <when>{ time }</when>
              </reviewed>)
}
```

```
<feature>
  <id>29</id>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <state>approved</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07T13:44:25.000-08:00</when>
  </reviewed>
</feature>
```

```
import org.jdom.*;          // JDOM

Element newFeature(String name, String engineer,
                   String reviewer, String time)
{
  int id = ++nFeatures;
  return new Element("feature")
    .addContent(new Element("id")
                .addContent(Integer.toString(id)))
    .addContent(new Element("name")
                .addContent(name))
    .addContent(new Element("engineer")
                .addContent(engineer))
    .addContent(new Element("state")
                .addContent("submitted"))
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(time)));
}
```

```
XML newFeature(String name, String engineer,
               String reviewer, String time)
{
  return <feature>
           <id>{ ++nFeatures }</id>
           <name>{ name }</name>
           <engineer>{ engineer }</engineer>
           <state>submitted</state>
           <reviewed>
             <who>{ reviewer }</who>
             <when>{ time }</when>
           </reviewed>
         </feature>;
}
```

```
XML newFeature(String name, String engineer,
               String reviewer, String time)
{
  return #feature {
           #id { ++nFeatures },
           #name { name },
           #engineer { engineer },
           #state { "submitted" },
           #reviewed {
             #who { reviewer },
             #when { time }
           }
         };
}
```

```
XML newFeature(String name, String engineer,
               String reviewer, String time)
{
  return #feature {
           #id { ++nFeatures },
           #name { name },
           #engineer { engineer },
           #state { "submitted" },
           #reviewed {
             #who { reviewer },
             #when { time }
           }
         };
}
```

```
void addReviewer(XML feature,
                 String reviewer, String time)
{
  feature.add(<reviewed>
                <who>{ user }</who>
                <when>{ time }</when>
              </reviewed>)
}
```

```
void addReviewer(XML feature,
                 String reviewer, String time)
{
  feature.add(#reviewed {
                #user { user },
                #when { time }
              });
}
```

java.lang.XML

Construction	*XML Literals*
Conversion	***Datatype Coders***
Navigation	*Collections* + *Generics* + *Paths*
Streaming	*Hybrid Event/Tree API*

```
import java.sql.Timestamp;
import javax.xml.datatype.*;

void addReviewer(Element feature,
                 String reviewer, Timestamp time)
  throws DatatypeConfigurationException
{
  GregorianCalendar gc = new GregorianCalendar();
  gc.setTimeInMillis(time.getTime());
  DatatypeFactory df
    = DatatypeFactory.newInstance();
  XMLGregorianCalendar xc
    = df.newXMLGregorianCalendar(gc);
  String ts = xc.toXMLFormat();
  feature
    .addContent(new Element("reviewed")
                .addContent(new Element("who")
                            .addContent(reviewer))
                .addContent(new Element("when")
                            .addContent(ts)));
}
```

```
import java.sql.Timestamp;
import javax.xml.datatype.*;

void addReviewer(XML feature,
                 String reviewer, Timestamp time)
  throws DatatypeConfigurationException
{
  GregorianCalendar gc = new GregorianCalendar();
  gc.setTimeInMillis(time.getTime());
  DatatypeFactory df
    = DatatypeFactory.newInstance();
  XMLGregorianCalendar xc
    = df.newXMLGregorianCalendar(gc);
  String ts = xc.toXMLFormat();
  feature.add(<reviewed>
                <who>{ user }</who>
                <when>{ ts }</when>
              </reviewed>);
}
```

```
package java.xml;

abstract class DataCoder {

   ...

}
```

```
package java.xml;

abstract class DataCoder {

  String encode(Object ob);

  <T> decode(String data, Class<T> c);

  boolean supports(Class<?> c);

  ...

}
```

```
package java.xml;

abstract class DataCoder {

  String encode(Object ob);
  String encode(Object ob, String type);

  <T> decode(String data, Class<T> c);
  <T> decode(String data, Class<T> c, String type);

  boolean supports(Class<?> c);
  boolean supports(Class<?> c, String type);

  ...

}
```

```
package java.xml;

abstract class DataCoder {

  String encode(Object ob);
  String encode(Object ob, String type);

  <T> decode(String data, Class<T> c);
  <T> decode(String data, Class<T> c, String type);

  boolean supports(Class<?> c);
  boolean supports(Class<?> c, String type);

  // JLS ch 3 syntax for numeric and boolean types
  public static final DataCoder JAVA;

  // http://www.w3.org/2001/XMLSchema-datatypes
  public static final DataCoder XSD;

}
```

```
import java.sql.Timestamp;
import java.xml.DataCoder;

void addReviewer(XML feature,
                 String reviewer, Timestamp time)
{
  DataCoder dc = DataCoder.XSD;
  feature.add(<reviewed>
                <who>{ user }</who>
                <when>{ dc.encode(time) }</when>
              </reviewed>);
}
```

```
import java.sql.Timestamp;
import javax.xml.datatype.*;

Timestamp getReviewTime(Element feature)
  throws DatatypeConfigurationException
{
  Element r = feature.getChild("reviewed");
  if (r == null)
    return null;
  String ts = r.getChildText("when");
  DatatypeFactory df
    = DatatypeFactory.newInstance();
  XMLGregorianCalendar xc
    = df.newXMLGregorianCalendar(ts);
  return new Timestamp(xc.toGregorianCalendar()
                         .getTimeInMillis());
}
```

```
import java.sql.Timestamp;
import java.xml.DataCoder;

Timestamp getReviewTime(XML feature) {
  XML r = feature.get("reviewed");
  if (r == null)
    return null;
  return r.get("when").decode(DataCoder.XSD,
                              Timestamp.class);
}
```

java.lang.XML

Construction	*XML Literals*
Conversion	*Datatype Coders*
Navigation	***Collections*** ***+ Generics*** ***+ Paths***
Streaming	*Hybrid Event/Tree API*

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>42</id>
    <name>Zombie references</name>
    <engineer>mark.reinhold</engineer>
    <state>submitted</state>
  </feature>
</feature-list>
```

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>42</id>
    <name>Zombie references</name>
    <engineer>mark.reinhold</engineer>
    <state>rejected</state>
  </feature>
</feature-list>
```

```
void rejectOpenFeatures(Element featureList) {
  List fs = featureList.getChildren("feature");
  for (Object fo : fs) {
    Element f = (Element)fo;
    Element s = f.getChild("state");
    if (!s.getText().equals("approved"))
      s.setText("rejected");
  }
}
```

```
void rejectOpenFeatures(XML featureList) {
  List<XML> fs = featureList.findAll("feature");
  for (XML f : fs) {
    XML s = f.get("state");
    if (!s.text().equals("approved"))
      s.set("rejected");
  }
}
```

```
void rejectOpenFeatures(XML featureList) {
  List<XML> fs
    = featureList.findAll("feature[state!='approved']");
  for (XML f : fs) {
    f.get("state").set("rejected");
  }
}
```

java.lang.XML

Construction	*XML Literals*
Conversion	*Datatype Coders*
Navigation	*Collections* + *Generics* + *Paths*
Streaming	***Hybrid Event/Tree API***

```
<feature-list>
  <release>Mustang</release>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature> ...
```

```
Current Mustang feature list
 29: Method to find free disk space
201: Improve painting (fix gray boxes)
```

```
import javax.xml.stream.*;                    // StAX (JSR 173)

void summarize(InputStream in, Formatter out)
  throws XMLStreamException
{
  XMLInputFactory xif = XMLInputFactory.newInstance();
  XMLStreamReader xr = xif.createXMLStreamReader(in);
  xr.nextTag();
  xr.require(START_ELEMENT, null, "feature-list");
  xr.nextTag();
  xr.require(START_ELEMENT, null, "release");
  out.format("Current %s feature list%n",
             xr.getElementText());
  xr.require(END_ELEMENT, null, "release");
  while (xr.nextTag() == START_ELEMENT) {
    xr.require(START_ELEMENT, null, "feature");
    String i = null;
    String s = null;
    String n = null;
    while (xr.nextTag() == START_ELEMENT) {
      String ln = xr.getLocalName();
      String t = xr.getElementText();
      if (ln.equals("id")) i = t;
      else if (ln.equals("state")) s = t;
      else if (ln.equals("name")) n = t;
      xr.require(END_ELEMENT, null, ln);
    }
    xr.require(END_ELEMENT, null, "feature");
    if (s.equals("approved")) {
      out.format("%3s: %s%n", i, n);
    }
  }
  xr.require(END_ELEMENT, null, "feature-list");
  xr.close();
}
```

```
void summarize(XML featureList, Formatter out) {
   ...
}
```

```
void summarize(XML featureList, Formatter out) {
  out.format("Current %s feature list%n",
             featureList.get("release").text());
  List<XML> fs
    = featureList.findAll("feature[state='approved']")
  for (XML f : fs) {
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
  }
}
```

```
void summarize(XML featureList, Formatter out) {
  out.format("Current %s feature list%n",
             featureList.get("release").text());
  List<XML> fs
    = featureList.findAll("feature[state='approved']")
  for (XML f : fs) {
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
  }
}
```

```
void summarize(XML featureList, Formatter out) {
  out.format("Current %s feature list%n",
             featureList.get("release").text());
  List<XML> fs
    = featureList.findAll("feature[state='approved']")
  for (XML f : fs) {
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
  }
}
```

```
void summarize(XML featureList, Formatter out) {
  out.format("Current %s feature list%n",
             featureList.get("release").text());
  List<XML> fs
    = featureList.findAll("feature[state='approved']")
  for (XML f : fs) {
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
  }
}
```

```
void summarize(XML featureList, Formatter out) {
  out.format("Current %s feature list%n",
             featureList.get("release").text());
  List<XML> fs
    = featureList.findAll("feature[state='approved']")
  for (XML f : fs) {
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
  }
}
```

```
package java.xml;

class XMLSource {

   ...

}
```

```
package java.xml;

class XMLSource {

  XMLSource(InputStream in);

  // Is there another element?
  boolean hasNext();

  // If so, does it match the given path?
  boolean matches(String xpath);

  // Parse and return the current element
  XML match();

}
```

```
package java.xml;

class XMLSource {

  XMLSource(InputStream in);

  // Is there another element?
  boolean hasNext();

  // If so, does it match the given path?
  boolean matches(String xpath);

  // Parse and return the current element
  XML match();

}
```

```
package java.xml;

class XMLSource {

  XMLSource(InputStream in);

  // Is there another element?
  boolean hasNext();

  // If so, does it match the given path?
  boolean matches(String xpath);

  // Parse and return the current element
  XML match();

}
```

```
package java.xml;

class XMLSource {

  XMLSource(InputStream in);

  // Is there another element?
  boolean hasNext();

  // If so, does it match the given path?
  boolean matches(String xpath);

  // Parse and return the current element
  XML match();

}
```

```
package java.xml;

class XMLSource {

  XMLSource(InputStream in);

  // Is there another element?
  boolean hasNext();

  // If so, does it match the given path?
  boolean matches(String xpath);

  // Parse and return the current element
  XML match();

}
```

```
package java.xml;

class XMLSource {

  XMLSource(InputStream in);

  // Is there another element?
  boolean hasNext();

  // If so, does it match the given path?
  boolean matches(String xpath);

  // Parse and return the current element
  XML match();

}
```

```
import java.xml.XMLSource;

void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
import java.xml.XMLSource;

void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
import java.xml.XMLSource;

void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
import java.xml.XMLSource;

void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
import java.xml.XMLSource;

void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
import java.xml.XMLSource;

void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
import java.xml.XMLSource;

void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
import java.xml.XMLSource;

void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
<html>
  <body>
    <table>
      <caption>Current Mustang feature list</caption>
      <tr>
        <td>29</td>
        <td>Method to find free disk space</td>
      </tr>
      <tr>
        <td>201</td>
        <td>Improve painting (fix gray boxes)</td>
      </tr>
    </table>
  </body>
</html>
```

```
void summarize(InputStream in, Formatter out) {
  XMLSource xin = new XMLSource(in);
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      out.format("Current %s feature list%n",
                 xin.match().text());
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      out.format("%3s: %s%n",
                 f.get("id").text(),
                 f.get("name").text());
      continue;
    }
  }
}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);
  XML tb = <table/>;
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  return <html><body>{ tb }</body></html>;
}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);
  XML tb = <table/>;
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  return <html><body>{ tb }</body></html>;
}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);
  XML tb = <table/>;
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  return <html><body>{ tb }</body></html>;
}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);
  XML tb = <table/>;
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  return <html><body>{ tb }</body></html>;
}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);
  XML tb = <table/>;
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  return <html><body>{ tb }</body></html>;
}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);
  XML tb = <table/>;
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  return <html><body>{ tb }</body></html>;
}
```

```
package java.xml;

class XMLSink {

   ...

}
```

```
package java.xml;

class XMLSink {

  XMLSink(OutputStream out);

  // Writes x to the output stream, returns this
  XMLSink add(XML x);

  // Creates a sub-sink for a new sub-element
  // with the given name
  XMLSink sink(String name);

  // Closes this sink and any open sub-sinks
  void close();

}
```

```
package java.xml;

class XMLSink {

  XMLSink(OutputStream out);

  // Writes x to the output stream, returns this
  XMLSink add(XML x);

  // Creates a sub-sink for a new sub-element
  // with the given name
  XMLSink sink(String name);

  // Closes this sink and any open sub-sinks
  void close();

}
```

```
package java.xml;

class XMLSink {

  XMLSink(OutputStream out);

  // Writes x to the output stream, returns this
  XMLSink add(XML x);

  // Creates a sub-sink for a new sub-element
  // with the given name
  XMLSink sink(String name);

  // Closes this sink and any open sub-sinks
  void close();

}
```

```
package java.xml;

class XMLSink {

  XMLSink(OutputStream out);

  // Writes x to the output stream, returns this
  XMLSink add(XML x);

  // Creates a sub-sink for a new sub-element
  // with the given name
  XMLSink sink(String name);

  // Closes this sink and any open sub-sinks
  void close();

}
```

```
package java.xml;

class XMLSink {

  XMLSink(OutputStream out);

  // Writes x to the output stream, returns this
  XMLSink add(XML x);

  // Creates a sub-sink for a new sub-element
  // with the given name
  XMLSink sink(String name);

  // Closes this sink and any open sub-sinks
  void close();

}
```

```
package java.xml;

class XMLSink {

  XMLSink(OutputStream out);

  // Writes x to the output stream, returns this
  XMLSink add(XML x);

  // Creates a sub-sink for a new sub-element
  // with the given name
  XMLSink sink(String name);

  // Closes this sink and any open sub-sinks
  void close();

}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);
  XML tb = <table/>;
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  return <html><body>{ tb }</body></html>;
}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);
  XML tb = <table/>;


  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  return <html><body>{ tb }</body></html>;
}
```

```
XML summarizeToXHTML(InputStream in) {
  XMLSource xin = new XMLSource(in);



  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }

}
```

```
void summarizeToXHTML(InputStream in, OutputStream out) {
  XMLSource xin = new XMLSource(in);


  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }

}
```

```
void summarizeToXHTML(InputStream in, OutputStream out) {
  XMLSource xin = new XMLSource(in);
  XMLSink xout = new XMLSink(out);
  XMLSink tb
    = xout.sink("html").sink("body").sink("table");
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  xout.close();
}
```

```
void summarizeToXHTML(InputStream in, OutputStream out) {
  XMLSource xin = new XMLSource(in);
  XMLSink xout = new XMLSink(out);
  XMLSink tb
    = xout.sink("html").sink("body").sink("table");
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  xout.close();
}
```

```
void summarizeToXHTML(InputStream in, OutputStream out) {
  XMLSource xin = new XMLSource(in);
  XMLSink xout = new XMLSink(out);
  XMLSink tb
    = xout.sink("html").sink("body").sink("table");
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  xout.close();
}
```

```
void summarizeToXHTML(InputStream in, OutputStream out) {
  XMLSource xin = new XMLSource(in);
  XMLSink xout = new XMLSink(out);
  XMLSink tb
    = xout.sink("html").sink("body").sink("table");
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  xout.close();
}
```

```
void summarizeToXHTML(InputStream in, OutputStream out) {
  XMLSource xin = new XMLSource(in);
  XMLSink xout = new XMLSink(out);
  XMLSink tb
    = xout.sink("html").sink("body").sink("table");
  while (xin.hasNext()) {
    if (xin.matches("release")) {
      tb.add(<caption>Current {
                          xin.match().text()
                        } feature list</caption>);
      continue;
    }
    if (xin.matches("feature[state='approved']")) {
      XML f = xin.match();
      tb.add(<tr>
               <td>{ f.get("id").text() }</td>
               <td>{ f.get("name").text() }</td>
             </tr>);
      continue;
    }
  }
  xout.close();
}
```

```
import javax.xml.stream.*;    // StAX (JSR 173)

void summarizeToXHTML(InputStream in, OutputStream out)
  throws XMLStreamException
{
  XMLInputFactory xif = XMLInputFactory.newInstance();
  XMLStreamReader xr = xif.createXMLStreamReader(in);
  XMLOutputFactory xof = XMLOutputFactory.newInstance();
  XMLStreamWriter xw = xof.createXMLStreamWriter(out, "UTF-8");
  xw.writeStartDocument("UTF-8", "1.0");
  xw.writeStartElement("html");
  xw.writeStartElement("body");
  xw.writeStartElement("table");
  xr.nextTag();
  xr.require(START_ELEMENT, null, "feature-list");
  xr.nextTag();
  xr.require(START_ELEMENT, null, "release");
  xw.writeStartElement("caption");
  xw.writeCharacters("Current " + xr.getElementText()
                     + " feature list");
  xw.writeEndElement();
  xr.require(END_ELEMENT, null, "release");
  while (xr.nextTag() == START_ELEMENT) {
    xr.require(START_ELEMENT, null, "feature");
    String i = null;
    String n = null;
    String s = null;
    while (xr.nextTag() == START_ELEMENT) {
      String ln = xr.getLocalName();
      String t = xr.getElementText();
      if (ln.equals("id")) i = t;
      else if (ln.equals("name")) n = t;
      else if (ln.equals("state")) s = t;
      xr.require(END_ELEMENT, null, ln);
    }
    xr.require(END_ELEMENT, null, "feature");
    if (s.equals("approved")) {
      xw.writeStartElement("tr");
      xw.writeStartElement("td");
      xw.writeCharacters(i);
      xw.writeEndElement();
      xw.writeStartElement("td");
      xw.writeCharacters(n);
      xw.writeEndElement();
      xw.writeEndElement();
    }
  }
  xr.require(END_ELEMENT, null, "feature-list");
  xr.close();
  xw.writeEndElement();
  xw.writeEndElement();
  xw.writeEndElement();
  xw.writeEndDocument();
  xw.flush();
}
```

java.lang.XML

Construction	*XML Literals*
Conversion	*Datatype Coders*
Navigation	*Collections* + *Generics* + *Paths*
Streaming	*Hybrid Event/Tree API*

```
java.lang.XML   // Elements

java.xml

  // New Document Object Model
  Attribute
  Text
  Content         // Elements | Text
  Node            // Elements | Text | Attributes
  Document

  // Streaming input and output
  XMLSource
  XMLSink

  // More to come ...
```

Syntax
Mutability
Validation
Namespaces
Typechecking
Document types
Processing instructions

Coming soon to a JSR near you!

Writing *ad-hoc* XML applications is still painful.

Integrating XML directly into the language can help.

Q&A

More questions, comments, or ideas?
BOF 0443 **tonight** at 8:30pm
Moscone North Meeting Room (121)

Integrating XML into the Java™ Programming Language

Mark Reinhold
Java SE Chief Engineer
Sun Microsystems

TS-3441