Taming XML: Objects First, Then Markup

Matt Bone, Peter F. Nabicht, Konstantin Läufer,
and George K. Thiruvathukal

Emerging Technologies Laboratory
Department of Computer Science
Loyola University Chicago

2008 IEEE International Conference
on Electro/Information Technology

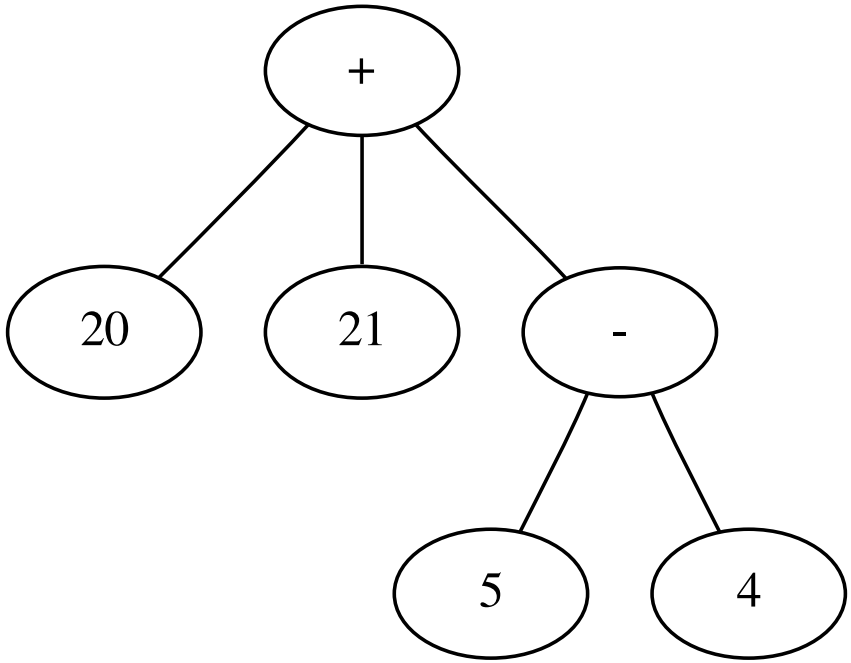
Motivation

The BetterXML framework aims to:

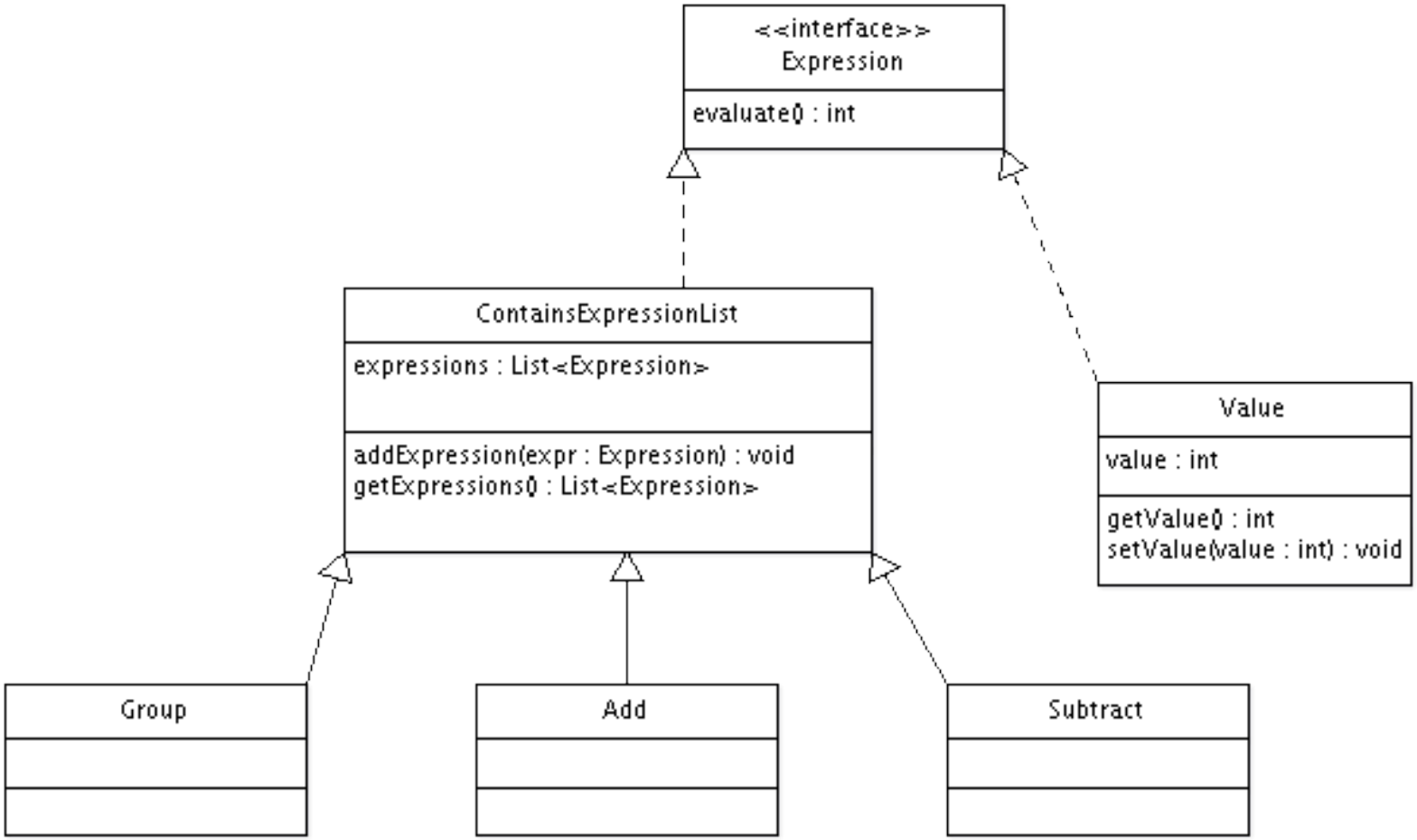
- Minimize the accidental complexity of handling XML
- Focus on the domain objects and bind to XML later

A Guiding Example

Thinking of a simple calculator, we can represent a calculation like $21 + 20 + (5 - 4)$ as an expression tree:



Expression Tree In an Object Oriented Language



Expression Tree as Markup

```
<group>
  <add>
    <value value="21"/>
    <value value="20"/>
    <subtract>
      <value value="5"/>
      <value value="4"/>
    </subtract>
  </add>
</group>
```

Mapping the Expression Tree to Markup with XElement

```
public class Add extends XElement implements Expression{

  public int evaluate() {
    int result=0;

    for (XElement element: this.getChildrenElements()) {
      if (element instanceof Expression)
      result += ((Expression) element).evaluate();
    }

    return result;
  }
}
```

Mapping the Expression Tree to Markup with XElement

Using XElement is quite similar except that individual classes may be bound to individual elements:

```
ToXElementContentHandler handler =
    new ToXElementContentHandler();
handler.registerElementClass(Add.class, "add");
handler.registerElementClass(Subtract.class, "subtract");
handler.registerElementClass(Group.class, "group");
handler.registerElementClass(Value.class, "value");
```

Mapping the Expression Tree to Markup with NaturalXML

```
@Element("add")
public class Add extends ContainsExpressionList {

  public int evaluate() {
    int sum = 0;

    for(Expression expression: expressions) {
      sum += expression.evaluate();
    }

    return sum;
  }
}
```

Mapping the Expression Tree to Markup with NaturalXML

```
public abstract class ContainsExpressionList
    implements Expression {

  @Children({Group.class, Add.class,
    Subtract.class, Value.class})
  protected List<Expression> expressions =
    new ArrayList<Expression>();

  public List<Expression> getExpressions() {
    return expressions;
  }

  public void addExpression(Expression expression) {
    this.expressions.add(expression);
  }
}
```

Mapping the Expression Tree to Markup with NaturalXML

```
@Element("value")
public class Value implements Expression {

  @Attribute("value")
  private String value;

  public String getValue() { return value; }
  public void setValue(String value) { this.value = value; }

  public int evaluate() {
    return Integer.valueOf(value);
  }
}
```

NaturalXML Annotations

NaturalXML allows the binding of elements to classes with class metadata:

- @Element
- @Attribute
- @Children
- @CData
- @Singleton
- @Namespace

XML Intermediate Representation (XIR)

- A record oriented, lossless representation of XML
- Character data encoded as Base64
- Easy to parse

Some Simple XML...

```
<value value="5">some cdata</value>
```

...translated to XIR

```
xir.type:verbatim=element
ns:verbatim=
xir.subtype:verbatim=begin
qname:verbatim=value
name:verbatim=value
attributes:verbatim=1

xir.type:verbatim=attribute
ns:verbatim=
xir.subtype:verbatim=none
qname:verbatim=value
name:verbatim=value
value:verbatim=5

xir.type:verbatim=characters
cdata:base64=c29tZSBjZGF0YQ==
xir.subtype:verbatim=none
length:verbatim=10
```

Future Directions

- XElement expressed as a subset of NaturalXML
- Various Input/Output formats: YAML, s-expressions
- RelaxNG Schema to BetterXML transformer
- Interesting applications: BetterPipes

Any questions?

Availability

The BetterXML framework can be downloaded at:
`http://betterxml.googlecode.com`