Extracting Typed Values from XML Data

Fabio Simeoni      David Lievens      Paolo Manghi
Steve Neely      Richard Connor
XML and Language Values

Language values are increasingly generated and manipulated outside the language jurisdiction, and occur instead as fragments of XML documents.

- Semistructured Documents:
  - irregular structure
  - unstable structure

- Portable Documents:
  - regular structure
  - self-describing
XML and Language Values

Semistructured Document

Portable Document
### XML and Language Values

```
<staff>
  <member code = "123517">
    <name>Richard Connor</name>
  </member>
  <member code = "123345">
    <name>Steve Neely</name>
  </member>
  <member code = "175417">
    <name>Fabio Simeoni</name>
  </member>
</staff>
```

---

**Class Member**

```java
private String name;
private int code;
String getName() {...}
void setName (String n) {...}
int getCode() {...}
void setCode (int c) {...}
```

**Class Staff**

```java
Private Member[] members;
Member[] getMembers() {...}
void setMembers(Member[] m) {...}
```
A Computational Requirement

Programming over data should be as:

- simple
- safe
- efficient

as programming over staff with existing programming languages.
Programming Models over XML

- Dedicated QLs (XQL, XML-QL, Lorel, etc.)
  - computationally incomplete
  - not well suited to complex tasks
  - essentially untyped

- Research Models (XDuce, Tequila, etc.)
  - graph types vs. language types
  - steep learning curve/lack of tool support

- Language Bindings (DOM, SAX) ...
SAX & DOM

**Data Structure:**
- SAX: a string served as a temporal sequence of tokens
- DOM: a tree

**Programming Model:**
- SAX: events and call-backs
- DOM: tree navigation
SAX & DOM Limitations

Data structures do not support **data semantics**
(staff members are neither strings nor tree nodes)

SAX supports the **syntactic structure** of the data.

Think as a Parser!

DOM supports the **logical structure** of the data.

Think as a Botanist! 😊

Good for syntactic (**SAX**) and structural (**DOM**) manipulations, not domain-specific tasks.
Class SaxTask extends DefaultHandler {
    private String name, code;
    private boolean inProject;

    private CharArrayWriter buffer = new CharArrayWriter();

    public void characters (char[] ch, int start, int length) {buffer.write(ch,start,length);}

    public void startElement (String uri, String name, String qName, Attributes atts) {
        if (name.equals("member")) code = atts.getValue("code");
            if (name.equals("project")) inProject = true;
            buffer.reset();
    }

    public void endElement (String uri, String name, String qName) {
        if (name.equals("project")) inProject = false;
            if (name.equals("name") && !inProject)
                name = buffer.toString().trim()
                ... do something with name and code...}}
DOM Programming

String name=null;
int code;

Element staff = d.getDocumentElement();
NodeList members = staff.getElementsByTagName("member");
int membCount = members.getLength();

for (int i=0;i<membCount;i++) {
    Element member = (Element) members.item(i);
    code = Integer.parseInt(member.getAttribute("code"));
    NodeList children = member.getChildNodes();
    int length = children.getLength();
    for (int j=0;j<length;j++) {
        Node child = children.item(j);
        if (child.getNodeType()==Node.ELEMENT_NODE) {
            String tagName = ((Element) child).getTagName();
            if (tagName.equals("name")) name=
                ((character data) child.getFirstChild()).getData();
        }
        ...
do something with name and code...
    }
}
SAX & DOM Programming

How easy is to write, maintain, and evolve the code?

✓ Programming is tedious, redundant and error-prone
✓ Code is convoluted even for simple tasks

How safe is the code?

Nodelist members = staff.getElementsByTagName("mebmer");
✓ Safety is responsibility of the programmer
✓ Programmatic checks worsen code readability
✓ Programmatic checks are simply not enough...
**SAX & DOM Programming**

Compare with:

```java
Member[] members = staff.getMembers();
for (int i=0; i<members.length; i++) {
    code = members[i].code;
    name = members[i].name;
    {...do something with name and code...}
}
```

✅ Simple because the programming algebra is domain-specific

✅ Safe and efficient because static knowledge is domain-specific

Think Straight!
High-Level Bindings

Conclusion:

✓ DOM & SAX are low-level bindings

✓ We need high-level bindings that automate the conversion of XML fragments into their counterpart within the language
A High-Level Binding

Extraction Mechanism

Types Staff and Member

staff

Programming Language

Binding as type projection

d

d'