JavaOne℠

Sun's 2002 Worldwide Java Developer Conference℠

XML Queries Compiled to Bytecodes

Per Bothner
Consultant
Brainfood Inc.

Xquery

A new very-high-level language standard from W3C for processing XML-like data

Qexo

An implementation that compiles XQuery programs directly to bytecodes for the Java™ platform (“Java bytecodes”)

Agenda

- Introduction to W3C's new XQuery language
- Language introduction and feature overview:
  - Generating XML/HTML
  - Querying XML "databases"
- The Qexo implementation and its features
- Comparison with JavaServer Pages™ (JSP™) technology and XSLT
- Conclusions; demo; questions

My Background

- Spearheaded the Gcc-based GCJ (GNU Compiler for the Java™ platform) project (1996–ongoing)
- Wrote Kawa (1996–), compiling a functional language (Scheme) into the Java™ virtual machine
- Wrote JEmacs (Emacs with Emacs Lisp compiling into bytecodes, using Kawa and the JFC/Swing API)
- Long involvement with GNU/Linux
- Long interest in high-level languages and data types

XQuery: A Language for Processing XML Data

A New Language From W3C

XQuery is a programming language for XML

Designed for processing of XML data

A super-set (mostly) of XPath (used in XSLT)

XSLT and XPath being revised at same time

Designed by World Wide Web Consortium

Standard hopefully to be released this year

XQuery Executive Summary

A very high-level programming language

Designed for querying, merging (joining), and generating XML data sets

Works on tree structures (DOM-like), not text

One goal is “SQL for XML data-bases”

No assignments or side-effects (updates may be added later)

Statically typed (optionally) and “optimizable”

Generating XML and HTML

Most HTML-generating tools generate text

This includes: JSP™ technology, ASP, PHP, CGI

Text is fine for final output, but not further processing

Most tools are statement-oriented:

```
print "<em>now</em>"
```

XSLT generates “result fragments”

XQuery generates node sequences

Expressions

An XQuery “program” is an expression

Can evaluate to a simple value:

`3+4` evaluates to the number 7

An element constructor expression:

```
<title>some text</title>
```

evaluates to a node object, *not* a string

No parsing needed to extract parts of a node

Local Variables

You can bind values to local variables, using a **`let`** expression

Can substitute values in element constructor

```
let $x := <bold>more text</bold>
  return <title> some {$x}</title>
```

Uses static (lexical) scoping (like the Java programming language)

Sequences

Values can be sequences of simple values

Comma operator **`3,4`** appends sequenes

Sequences cannot be nested

The **`children`** function returns a sequence of child nodes of a given node:

```
children(<a><b>X</b><c>Y</c></a>)
```

evaluates to: **`<b>X</b>, <c>Y</c>`**

Superset of XPath

Most XPath expressions are part of Xquery

This includes most *path expressions.* Example:

Get all **`sections`** with a **`title`** attribute of **`"Answers"`**, that are within the first **`chapter`**:

```
./chapter[1]//section[@title="Answers"]
```

This is a compact sub-language for selecting parts from an XML data-set

FLWR Expressions

FLWR expressions have 3 parts:

- One or more for or let clauses
  - A for loop binds a local variable for each element of a sequence
  - let binds a local variable to a value as a whole
- An optional where clause
- A result clause

```
for $ch in $doc/chapter
where $ch/@number <= 10
return <h2>{$ch/title}</h2>
```

'for' is like SQL 'select'

Join of **customers** and **orders**, using SQL:

```
select customers.name
from customers, orders
where customers.cid=order.cid
   and orders.oid="xx"
```

Same join, using XQuery:

```
for $c in customers, $o in orders
where $c.cid=$o.cid and $o.oid="xx"
return $c.name
```

Functions

Parameters and results can be primitive values, nodes, or sequences of either

```
define function descendent-or-self ($x)
{
  x,
  for $x in children($x)
    return descendent-or-self($z)
}
descendent-or-self(<a>X<b>Y</b></a>)
```

Evaluates to:

```
<a>X<b>Y</b></a>, X, <b>Y</b>, Y
```

Types

XQuery is (optionally) statically typed

Has type hierachy, type tests, coercions

Also has less usual types:

- Nodes, elements with specific tags, etc;
- Sequence types;
- Regular expressions over tree types

E.g., sequence of xhtml table rows:

```
(element html:tr)*
```

Typeswitch Expressions

Convenient syntax for testing type of a value

```
typeswitch ($animal)
  case element duck
    return quack($animal)
  case element dog
    return woof($animal)
  default
    return "No sound"
```

An Application: Photo Album

Application: organize digital photos (off-line)

Reads `index.xml` creating linked html pages

Previously used XSLT; re-wrote in XQuery

Substantial extra functionality with similar code size

Much faster than earlier version using Xalan

If curious browse: http://pics.bothner.com/

Implementing XQuery: Qexo

Implementations

Various groups are implementing Xquery

W3C specification is still (as of March 2002) incomplete and even inconsistent

Thus so far only prototypes

Will discuss Qexo (aka Kawa-XQuery), my open-source implementation

- Compiler-based implementation
- Servlet support extension

Qexo: Compiling to Bytecodes

Qexo (aka Kawa-XQuery) compiles XQuery programs and expressions to bytecodes

Uses Kawa compiler toolkit (which has compiled Scheme to JVM bytecodes since 1996)

Qexo takes advantage of Kawa optimizations and utilities

Fast interactive response (eval, load)

Internals: Consumer interface

`Consumer`: An “event” interface similar to SAX **`DocumentHandler`**

Used to transmit structured sequences

Kawa compiler optimizes many operations if “result context” is **`Consumer`**

E.g., this XQuery program does not create a “DOM”:

```
document("Dinosaurs.xml")/book/chapter
```

Internals: TreeList DOM

Kawa uses **`TreeList`** to store nodes

Uses a char buffer + an Object array

More efficient than standard DOM:

Normally just append new nodes to the buffer

A node is an index into the **`TreeList`**'s buffer

Sequences also use **`TreeList`**

Servlets

Qexo can compile XQuery program to servlet

XQuery result becomes servlet's response

Predefined variable **`$request`** is the **`HttpServletRequest`**

`$request/pathInfo` is shorthand for **`invoke($request, "getPathInfo")`**

Easy as JSP™ Technology

Simple JSP code example:

```
<p>Today is <%= new Date() %>.<p/>
```

Kawa-XQuery can do the same:

```
<p>Today is {make("java.util.Date")}.</p>
```

JSP processor generates text

XQuery generates node(s)

Output stage turns that into XML or HTML

HTTP Response Headers

Can set server response headers:

```
response-header("Content-type",
                "text/plain")
```

Same script works for both servlets and CGI

Setting **`Content-type`** optional; default type inferred from following data

`response-header` returns a special datum; does not “set” anything until final output

Debugging XQuery

Compiling to byte-code aids debugging:

Compilation-time errors refer to XQuery source line

Likewise for run-time stack traces

Contrast JSP technology: Most errors refer to generated Java™ file, not original JSP™ page

Qexo vs. Other Implementations

Qexo is open source with liberal license

Useful (though incomplete); available now

Compiles to Java bytecodes

Can interoperate with Java technology

Same program can run under shell, as CGI, or as servlet

Comparisons and Conclusions

JSP Technology vs. XQuery

Like JSP technology, a high-level language is compiled down and run by servlet engine

XQuery is a full programming language

XQuery is a unified language, not a mix of HTML and Java technology

Can analyze XML data, which neither JSP or Java technologies can do as conveniently

XQuery at a higher level than Java technology

XSLT vs. XQuery

XQuery does not have XSLT's convenient template model

XQuery is usually less verbose

Programming in XSLT (even just conditionals) is tedious

XQuery is a powerful programming language

Optimization easier for XQuery

XQuery Conclusion

XQuery is a powerful, optimizable language

Useful for querying and merging XML datasets

Useful for generating XML and HTML

W3C has released working drafts

Some problematic issues remain: static typing, updating, syntax embedding in XML, …

Prototype implementations can perform useful tasks now

Check It Out!

Resources:

XQuery: http://www.w3.org/XML/Query

Qexo: http://www.qexo.org/

Kawa: http://www.gnu.org/software/kawa/

Me: mailto:per@bothner.com

Demo

Q&A

JavaOne℠

Sun's 2002 Worldwide Java Developer Conference™

BEYOND BOUNDARIES

Session 2768