transform.xq for dummies

Eric van der Vlist

vdv@dyomedea.com
@evlist

http://vdv.re/xmlondon2015
Wish me good luck

- New computer
- New OS version (Ubuntu 15.04)
- New oXygen version (17)
- New presentation format (George Bina's DITA CSS)
- New topics (XQuery, higher-order functions)
Data-Driven Programming in XQuery

Introduction
Introduction

A loooong time ago

Data-Driven Classes in Ruby

Michael Granger and David McCorkhill

O’Reilly Open Source Convention
July 7 - 11, 2003

granger_mccorkhill.pdf
Introduction

Eureka

- Data-driven techniques minimize the impact of new or changing data structures and APIs
- Expressing functionality with data allows for run-time changes
- Allows code to be managed in parceled chunks

granger_mccorkhill.pdf
Introduction

A source of inspiration

- [XML Driven Classes in Python](#OSCON 2004)
- [TreeBind](#XTech 2006) and [XML 2006]
Introduction

What are “data-driven” classes?

- All classes are data-driven in some sense.
- Data-driven classes are classes defined by dynamic information.
  - Classes which modify or create their behavior based on state (data).
  - Classes not (completely) defined by source or object code

/granger_mccorkhill.pdf
Introduction

What is Data-Driven programming?
Introduction

What is Data-Driven programming?

Wikipedia

“In computer programming, data-driven programming is a programming paradigm in which the program statements describe the data to be matched and the processing required rather than defining a sequence of steps to be taken.[1] Standard examples of data-driven languages are the text-processing languages sed and AWK,[1] where the data is a sequence of lines in an input stream – these are thus also known as line-oriented languages – and pattern matching is primarily done via regular expressions or line numbers.”

-- Wikipedia
Introduction

What is Data-Driven programming?

c2.com

“Data Driven Programs are programs which process data files whose contents cause the program to do something different. The extreme case is an interpreter and the interpretable program files.”

-- c2.com
Introduction

What is Data-Driven programming?

Data-oriented Design

“If the ultimate result of an application is data, and all input can be represented by data, and it is recognised that all data transforms are not performed in a vacuum, then a software development methodology can be founded on these principles, the principles of understanding the data, and how to transform it given some knowledge of how a machine will do what it needs to do with data of this quantity, frequency, and it's statistical qualities. Given this basis, we can build up a set of founding statements about what makes a methodology data-oriented.”

-- # Data-oriented Design
Introduction

Examples
Introduction

Examples

Hello world

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

- This hello world is 0% data-driven.
- The JVM that runs this program is 100% data-driven.
Introduction

Examples

A backup utility

```xml
<backups>
  <task name="apache-norm">
    <postgres name="test" user="apache" dump="dbtestdump.sql"/>
    <mysql name="wikixmlfr" user="wikixmlfr" password="XXXX" dump="/home/apache/wikixmlfrdump.sql"/>
    <user name="apache" host="gwnormandy.dyomedea.com">
      <exclude>xmlfr</exclude>
      <exclude>vdv/albums</exclude>
    </user>
  </task>

  <!-- ... -->

</backups>
```

```
# backup.xml
```
Introduction

Examples

A mailing list manager

- Used by the XML Guild
- Powered by TreeBind
- XML elements are mapped to classes and methods that wrap IMAP and SMTP actions.
- See »mailing-list.xml«.
<?xml version="1.0" encoding="UTF-8"?>
<listManager>
  <server>localhost</server>
  <storeType>imap</storeType>
  <user>...</user>
  <password>...</password>
  <port>143</port>
  <folderManager>
    <folder>INBOX</folder>
    <messageHandler>
      <ifEither>
        <ifIsRecipient>
          info@xmlguild.org
        </ifIsRecipient>
        <ifIsRecipient>
          info@thexmlguild.org
        </ifIsRecipient>
        <ifIsRecipient>
          info@xmlguild.info
        </ifIsRecipient>
        <ifIsRecipient>
          info@xml-guild.org
        </ifIsRecipient>
        <ifIsRecipient>
          info@xml-guild.com
        </ifIsRecipient>
      </ifEither>
      <sendToList>
        <subjectPrefix>[the XML Guild]</subjectPrefix>
        <footer><![CDATA[
          The XML Guild
          where you find established XML experts...
          http://xmlguild.org/
          info@xmlguild.org
        ]]>"
      </footer>
    </messageHandler>
    <recipient>vdv@dyomedea.com</recipient>
    <recipient>...</recipient>
    <!-- other recipients removed -->
    <envelopeFrom>info-bounce@xmlguild.org</envelopeFrom>
    <header name="Precedence">List</header>
    <header name="List-Iid">&lt;Info.xmlguild.org&gt;</header>
    <header name="List-Id">&lt;Info.xmlguild.org&gt;</header>
    <server>localhost</server>
    <user>...</user>
    <archive>archive</archive>
    <sendToList>
      <moveTo>done</moveTo>
    </messageHandler>
    <moveTo>unparsed</moveTo>
  </folderManager>
</listManager>
Introduction

XSLT

- Like sed and awk mentioned by [wikipedia](http://wikipedia.org), XSLT is data-driven by design.
- `<xsl:template match=""">` natively binds data to XSLT instructions
Introduction

Functional programming

- Data-Driven programming in functional worlds have not been widely explored yet.
- Functional programming considers functions as data, does that help?
- How can we bind data to functions?
- That's the topic of this talk!
Introduction

Conclusion

• It's a paradigm
• Programming languages features and libraries can facilitate data-driven programming
• Data binding libraries are helpful to do data-driven programming in object oriented worlds
• XSLT is data-driven by design
• What about functional programming?
Functional XQuery
Functional XQuery

Hangman

- Rules borrowed from Wikipedia.
- State persisted in XML (hangman.xml).

```xml
<hangman status="in-progress" misses="Z">
  <word>
    <letter guessed="true">H</letter>
    <letter guessed="true">A</letter>
    <letter guessed="true">N</letter>
    <letter guessed="false">G</letter>
    <letter guessed="true">M</letter>
    <letter guessed="true">A</letter>
    <letter guessed="true">N</letter>
  </word>
  <display>
    <head shown="true"/>
    <body shown="true"/>
    <right_arm shown="true"/>
    <left_arm shown="true"/>
    <right_leg shown="false"/>
    <left_leg shown="false"/>
  </display>
</hangman>
```
**Functional XQuery**

In Python

*Reference implementation* in Python using the XML-Driven classes presented at OSCON 2004.

```python
# Class Letter: individual letters to be guessed
class Letter(XmlObject.XmlObjectElement):

    def addGuess(self, guess):
        if self._value() == guess:
            self.guessed._set('true')
            return True
        else:
            return False

    def isGuessed(self):
        return self.guessed._value() == 'true'

XmlObject.XmlObjectElement.letter = Letter
```
Functional XQuery

In XSLT

```xml
<xsl:template match="letter[@=guess]/@guessed">
  <xsl:attribute name="guessed">true</xsl:attribute>
</xsl:template>
```
Functional XQuery

transform.xq
Functional XQuery

XQuery with transform.xq

```xquery
''transform.xq'' using ''transform.xq''.

(: Guesses :) tfm:rule('letter/@guessed', function($node, $node) {
  attribute(xs:QName(name($node))) {
    if ($node/.. = $guess)
      then 'true'
    else $node
  }
}),

Doesn't work on Saxon for whatever reason."
```
Functional XQuery

XQuery with transform.xq and annotations

```xquery
declare %tm:rule("default","letter/@guessed",2)
function f:guessed-att($mode, $node)
    f:copy($node,
        if ($node/../ = $guess)
        then 'true'
        else $node
    );

'Relies on a MarkLogic extension function.'
```
Functional XQuery

What transform.xq brings us

- A parser for a subset of XSLT match patterns (converted to XQuery functions).
- A way to bind XQuery functions (playing the role of XSLT templates) to match patterns.
- *What could we do without this outstanding library?*
Functional XQuery

Classical XQuery

```xquery

else if ($node instance of attribute(guessed)) then
  f:copy($node,
    if ($node/.. = $guess)
      then 'true'
      else $node
  )

...}

Could be much worse!
```
Functional XQuery

With higher-order functions

`hangman-hof.xq`

```xml
... (: @guessed attributes :) f:rule(
    function($node as node()) as xs:boolean {
        $node instance of attribute(guessed)
    }, function($node as node(), $transform as function(•)) as node() {
        f:copy($node, if ($node/.. = $guess)
            then 'true'
            else $node)
    }
);
...

Similar to transform.xq without match patterns.
```
Functional XQuery

Missing maps in XQuery 3.0

Dirty hack used by "hangman-hof.xq" to simulate maps:

```xml
(: borrowed from transform.xq :)  
declare function fn:rule(  
    $predicate as (function(node()) as xs:boolean),
    $action as (function(node()), function(*) as node())
  ) as function(*) {

    function($k as xs:string) {
      switch($k)  
      case 'predicate' return $predicate
      case 'action' return $action
      default return ()
    }
}
```
Functional XQuery

The magic of higher-order functions

The `f:transform()` function in `#hangman-hof.xq`:

```xml
(: Some high order functions magic :) 
declare function f:transform($rules as function(*)& as function(*)&)
  as function(*)&
  ($node as node(), $transform as function(*)&) as node()? {
    if (head($rules)('predicate')($node))
      then head($rules)('action')($node, $transform)
    else if (exists(tail($rules)))
      then f:transform(tail($rules))($node, $transform)
    else ()
  }
```


Functional XQuery

Conclusion

- Higher-order functions do facilitate data-driven programming in XQuery.
- Maps are missed in XQuery 3.0 but can be simulated.
- transform.xq gives you all you need but you can also easily write your own mechanism.
- Good old if/then/else and recursion isn't that awful after all!
Functional XQuery

Acknowledgments

Thanks to:

- Tony Graham, Patrick Durusau and my three anonymous reviewers for their reviews, comments, corrections, questions and suggestions.
- John Snelson for his very inspiring transform.xq.
- George Bina for pushing me into dita.
- XML London for this gathering.
Functional XQuery

Questions

- vdv@dyomedea.com
- http://vdv.re/xml london 2015
- @evlist