ApacheCon Europe 2000

Managing a complex Web site with Cocoon

Doug Tidwell
Senior Programmer, IBM
dtidwell@us.ibm.com
ApacheCon Europe 2000

ApacheCon Europe 2000

Agenda

- Goals for a successful Web site
- Transforming content with XML
- The mighty `document()` function
- Serving XML documents with Cocoon
- Tools setup
- We'll discuss various technologies along the way.

ApacheCon Europe 2000

Goals for a successful Web site

Delivering
a million custom pages
to a million customers a day

ApacheCon Europe 2000

Web site goals

- A modern Web site must:
  - Streamline content creation & delivery
  - Support multiple device types easily
  - Generate custom pages on a per-browser **and** per-device basis
  - Scale

ApacheCon Europe 2000

User scenario

- A user comes to our Web site.
- Based on the user's profile, certain articles (XML documents) are retrieved.
- Based on the user's device type and browser, the articles are transformed and delivered.
- The user gets a Web page **tailored** to their interests, **and formatted** for their device.

ApacheCon Europe 2000

What's the meta with your data?

- To manage this content, it's crucial that we have metadata.
- We'll look at some XML case study reports to see how this works.

A case study

```
<case_study geography="Europe" country="Germany">
  <metadata>
    <submitter . . . />
    <modifier . . . />
    <zone name="Java"/>
  </metadata>
  <subject>Banking Online with Java</subject>
  <industry name="Banking"/>
  <company>ARZ</company>
  <abstract>ARZ, a major European. . .</abstract>
</case_study>
```


The power of metadata

- Our content creation process ensures that we have all the metadata we need for each article.
- We can find all the articles that apply to XML, or to Europe, or to Germany, or to Linux, or were updated in the last week....

Intelligent caching

- The only way this scales is through **intelligent caching**.
  - If you've retrieved something before, don't retrieve it again.
  - If you've transformed something before, don't transform it again.
  - For pages composed of smaller pieces (JSPs, XSPs, etc.), we need intelligent caching for the pieces, also.

Transforming content with XML

Document wrangling for fun and profit

Xalan

- An open source implementation of the XSL-T and XPath recommendations
- Can be embedded in your code, can be run from the command line, or can be invoked by Cocoon.
- In our examples here, we'll let Cocoon invoke Xalan for us.

XSL-T and XPath

- **XSL-T** is the Extensible Stylesheet Language for Transformations.
- **XPath** is a language for describing locations in a document.
- They're both official recommendations of the W3C:
  - **www.w3.org/TR/xslt**
  - **www.w3.org/TR/xpath**

Using XSL-T and XPath together

- When we refer to a document, we talk about **Node**s and **NodeSet**s.
- Use an **XPath expression** to describe the part of the document you want to transform.
- Use **XSL-T elements** to describe the transformation itself.
- In general, XPath defines the data, and XSL-T defines the operations on it.

The "other" XSL

- The original Extensible Stylesheet Language was split into two parts: XSL-T and the **formatting objects** spec, XSL-FO.
- XSL-FO is a vocabulary for rendering XML, see **www.w3.org/TR/xsl** for the spec.
- I've tried to avoid using just "XSL."

The compulsory Hello World example

- Here's an XML document:

```
<?xml version="1.0"?>
<greeting>
  Hello, World!
</greeting>
```

- We want to transform this to HTML so we can render it in a browser.

The Hello World stylesheet

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL
  /Transform" version="1.0">
<xsl:output method="html"/>
<xsl:strip-space elements="*"/>
<xsl:template match="/">
  <xsl:apply-templates
       select="greeting"/>
</xsl:template>
```


The Hello Woild stylesheet

```
<xsl:template match="greeting">
 <html>
  <body>
   <h1>
    <xsl:value-of select="."/>
   </h1>
  </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```


Stylesheet results

- **<html><body><h1>Hello, World!</h1></body></html>**
- Our stylesheet produces a complete, legal HTML document, viewable in any browser.

ApacheCon Europe 2000

Reviewing the stylesheet

- The XSL-T elements at the top of the stylesheet are boilerplate; most of your stylesheets will use these same elements.
- You can use other XSL-T elements to control the type of the output, specify the DTD of the output document, etc.

ApacheCon Europe 2000

Reviewing the stylesheet

- The first template in the stylesheet is the **root template**:

```
<xsl:template match="/">
  <xsl:apply-templates
       select="greeting"/>
</xsl:template>
```

- Xalan invokes this template first. Inside our template, we ask that all **<greeting>** elements be processed.

ApacheCon Europe 2000

Reviewing the stylesheet

- The second template processes any **<greeting>** elements in the current context:

```
<xsl:template match="greeting">
 <html>
  <body>
   <h1>
    <xsl:value-of select="."/>
```


ApacheCon Europe 2000

Reviewing the stylesheet

- Because a stylesheet is an XML document, we have to close all of the open tags inside this template:

```
   </h1>
  </body>
 </html>
</xsl:template>
```


ApacheCon Europe 2000

Another approach

- We could have written the root template like this:

```
<xsl:template match="/">
 <html>
  <body>
  <xsl:apply-templates
       select="greeting">
  </body>
 </html>
</xsl:template>
```


ApacheCon Europe 2000

Invoking Xalan

- There are several different ways to use Xalan; for now, we'll use the command line to transform the document.

```
java org.apache.xalan.xslt.Process -in
  myfile.xml -xsl mystyle.xsl -out
  results.html
```

  - where **myfile.xml** is the XML source, **mystyle.xsl** is the stylesheet, and **results.html** holds the results.

Note

- A stylesheet doesn't have to transform an XML document to HTML, or any markup language at all.
- We'll demo a PDF generator in a few minutes.

The mighty document() function

So powerful, so flexible,

so unknown

The document() function

- The **document()** function allows you to read in another document.
- In your stylesheet, the **document()** function returns a set of nodes that represent the parsed document.
- This extremely powerful function allows you to combine a number of documents in a manageable way.

A sample application

- We have a number of case studies. Each case study is a small XML document.
- Keeping the documents small and simple makes them easier to create and manage.
- Being able to combine them with the **document()** function gives us a tremendous amount of flexibility and power.

A case study

- Here's a small XML document, **arz.xml**:

```
<?xml version="1.0"?>
<!DOCTYPE case_study SYSTEM "cs.dtd">
<case_study geography="Europe"
  country="Germany" story_url="…">
 <metadata>…</metadata>
 <subject>Banking online…</subject>
 <industry name="Banking"/>
```


A case study

```
 <company>ARZ</company>
 <abstract>
  ARZ, a major European financial
  services centre…
 </abstract>
</case_study>
```

Creating a master document

- Here's a master document that references a number of case studies:

```
<?xml version="1.0">
<case_studies>
 <docref url="arz.xml"/>
 <docref url="bdb.xml"/>
 <docref url="bt.xml"/>
 <docref url="schwab.xml"/>
</case_studies>
```


Processing a master document

- We'll invoke the **document()** function against all of the url attributes on all of the **<case_study>** elements in our master document.
- We can create a bunch of master documents and a bunch of stylesheets for those master documents. This lets us generate lots of views of the same data, without changing **any** of our source docs.

Processing a master document

- Here's how we invoke the **document()** function for all of the **<docref>** elements:

```
<xsl:template match="case_studies">
 <xsl:for-each
  select="document(//docref/@url)
  //case_study/@geography">
   <xsl:sort select="."/>
   <xsl:apply-templates
  select="ancestor::case_study/>
```


Processing a master document

- This example reads all of the documents referenced in our master document.
- Notice that the result of the **document()** function is a node set, from which we can use XPath expressions to look for other things.
- In this example, we're reading a bunch of case studies, then sorting them by geography.

Processing a master document

- This extremely powerful technique allows you to create any number of views of a set of documents.
- You can sort on any criteria you wish, and filter the data in the documents any way you want.
- Best of all, we don't have to change anything in our source documents.

Serving XML documents with Cocoon

Delivering custom pages for all sorts of devices

ApacheCon Europe 2000

Cocoon

- An open source publishing framework
- Allows you to deliver XML-tagged content to browsers that don't support XML

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 37

ApacheCon Europe 2000

How it works

- A request (`*.xml`) comes in to Apache
- Cocoon takes control, checks the `User-Agent` field
- Cocoon looks at the processing instruction (PI) at the top of the XML file
- Based on the `User-Agent` and the PI, Cocoon invokes a formatter on the XML file and sends the results to the client.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 38

ApacheCon Europe 2000

Web development philosophy

- Cocoon is architected on the idea that there are three separate tasks in Web development:
  - XML creation
  - XML processing
  - XSL rendering
- The files involved in each of these steps are separate from the others.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 39

ApacheCon Europe 2000

XML creation

- Content creators should focus on their DTDs or schemas.
- Rendering should never enter a content creator's mind, ideally....

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 40

ApacheCon Europe 2000

XML processing

- The XML processing step involves generating any dynamic components of the XML document.
- This includes timestamps, database queries, LDAP queries, personalized information, etc.
- The logic required to do this is completely separate from the XML document.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 41

ApacheCon Europe 2000

XSL rendering

- The rendering step is accomplished by creating a stylesheet.
- The stylesheet author (programmer?) looks at the XML content as delivered by the XML processor, then builds a stylesheet to create the desired output.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 42

ApacheCon Europe 2000

Cocoon configuration

- There are a number of things you can configure in Cocoon:
  - Browsers
  - Producers
  - Processors
  - Transformers
  - Formatters
- We'll discuss these on the next few slides.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 43

ApacheCon Europe 2000

Cocoon browsers

- You can define browsers, based on string values in the **`User-Agent`** field of the HTTP header.
- Default browsers: IE, Pocket IE, HandWeb, AvantGo, DoCoMo, Opera, Lynx, Java, Nokia, UP, and Mozilla.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 44

ApacheCon Europe 2000

Cocoon producers

- Producers respond to the **`HttpServletRequest`** object and generate some XML content.
- The default producer simply reads an XML file; you can create your own if you need to.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 45

ApacheCon Europe 2000

Cocoon processors

- Processors handle the XML generated or retrieved by the producers.
- Default processors: XSLT, SQL, LDAP, and XSP (Extensible Server Pages, an enhancement to JSPs).

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 46

ApacheCon Europe 2000

Cocoon transformers

- The default transformer is the Xalan XSL-T engine.
- You can create your own, but that's not usually necessary.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 47

ApacheCon Europe 2000

Cocoon formatters

- Formatters are the most visible components of Cocoon.
- A formatter takes an XML file as input, then uses XSL-T to convert it into something.
- Supplied formatters: XML, text, HTML, XSL-FO, WML, and VRML.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 48

ApacheCon Europe 2000

Demos

Taking XML documents and serving them up

ApacheCon Europe 2000

Cocoon samples

- Cocoon console
- XML to HTML
- Browser-specific stylesheets
- XML to PDF
- XML to WML

ApacheCon Europe 2000

Tools setup

How to get up and running

ApacheCon Europe 2000

Tools setup

- The tools I'm using are from the Apache XML project (**xml.apache.org**):
  - **Xerces** - an XML parser
  - **Xalan** - an XSL-T processor
  - **FOP** - converts XML formatting objects into PDF files
  - **Cocoon** - a Java Servlet-based XML publishing framework

ApacheCon Europe 2000

Cocoon packaging

- Although they're actually four separate tools, Cocoon includes Xerces, Xalan, and FOP.
- The Cocoon distribution contains versions of Xerces, Xalan, and FOP tested to work with Cocoon. When Cocoon is set up, all the other tools are installed as well.

ApacheCon Europe 2000

Cocoon setup

- All of today's demos and examples are running on Apache 1.3.12 with Version 3.1 of the Tomcat servlet engine.
- I'm using Cocoon 1.7.4, which includes Xerces 1.0.3, Xalan 1.0.1, and FOP 0.12.1.

ApacheCon Europe 2000

Cocoon setup

- We'll review some brief setup instructions for running Cocoon under Apache and Tomcat.
- I've successfully run Cocoon under other Web servers as well; any server that supports Version 2.2 of the Servlet spec should work just fine.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 55

ApacheCon Europe 2000

Cocoon setup

- To further simplify installation, we'll use the **jar** files included with the Cocoon package.
- If you want to build the tools yourself, you'll need to use **Ant**, an XML-based build/make tool.
  - Ant's **jar** file is included with Cocoon, but you'll want to get the full package from **jakarta.apache.org**.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 56

ApacheCon Europe 2000

Cocoon setup

- First of all, we'll make changes to Tomcat's initialization script.
  - On Windows systems, the file is **%TOMCAT_HOME%\bin\tomcat.bat**.
  - On Unix & Linux systems, it's **%TOMCAT_HOME%/bin/tomcat.sh**.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 57

ApacheCon Europe 2000

Setting your CLASSPATH

- First of all, you need to set your **CLASSPATH**. Add these files from Cocoon's home directory:
  - **bin/cocoon.jar**
  - **lib/xerces_1_0_3.jar**
  - **lib/xalan_1_0_1.jar**
  - **lib/fop_0_12_1.jar**
- You also need to add **%JAVA_HOME%/lib/tools.jar**.

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 58

ApacheCon Europe 2000

Setting your CLASSPATH

- As you add the new **jar** files, make sure that the Apache XML files come first.
- No matter what, make sure they come before **xml.jar**, which ships as part of the Tomcat package.
  - **The xml.jar** file contains Sun's parser, which you don't want to use....

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 59

ApacheCon Europe 2000

Defining the Cocoon servlet

- Add a **<servlet>** element to **web.xml**:

```
<servlet>
  <servlet-name>
    org.apache.cocoon.Cocoon
  </servlet-name>
  <servlet-class>
    org.apache.cocoon.Cocoon
  </servlet-class>
```

Managing a Web site with Cocoon xml.apache.org ibm.com/developer/xml 60

Defining the Cocoon servlet

```
  <init-param>
    <param-name>
      properties
    </param-name>
    <param-value>
      /usr/dougt/cocoon.properties
    </param-value>
  </init-param>
</servlet>
```


Set up the servlet mapping

- Now add a **<servlet-mapping>**:

```
<servlet-mapping>
  <servlet-name>
    org.apache.cocoon.Cocoon
  </servlet-name>
  <url-pattern>
    *.xml
  </url-pattern>
</servlet-mapping>
```


Testing Cocoon

- To make sure you've installed Cocoon correctly, check this URL:
  - **http://localhost/Cocoon.xml**

    (Cocoon must be capitalized, BTW.)
- You should see a page that summarizes the settings of Cocoon.

Wrapup

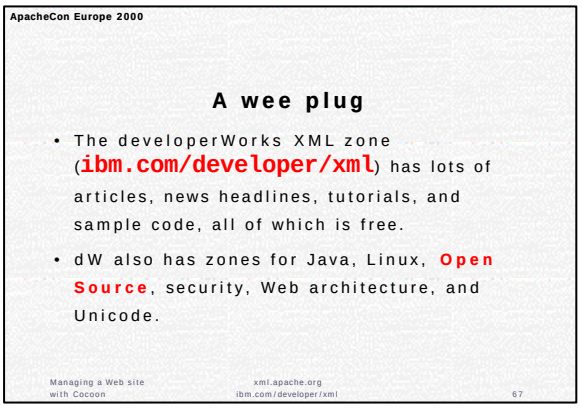
Why this stuff matters

Wrapup

- Our goals were to create a Web site that:
  - Streamlines content creation & delivery
  - Supports multiple device types easily
  - Generates custom pages on a per-user, per-browser, **and** per-device basis
  - Scales
- XML is an important part of all of these requirements.

Wrapup

- All of the tools and interfaces we've covered today are:
  - Based on open standards
  - Available on any Java-enabled platform
  - Open source
  - Available at no cost to you, the home viewer
- XML is the foundation of tomorrow's Web.


ApacheCon Europe 2000
A wee plug
• The developerWorks XML zone (ibm.com/developer/xml) has lots of articles, news headlines, tutorials, and sample code, all of which is free.
• dW also has zones for Java, Linux, Open Source, security, Web architecture, and Unicode.
Managing a Web site with Cocoon
xml.apache.org
ibm.com/developer/xml
67


ApacheCon Europe 2000
Thanks for coming!
Doug Tidwell
dtidwell@us.ibm.com
ibm.com/developer/xml
Managing a Web site with Cocoon
xml.apache.org
ibm.com/developer/xml
68