Berkenalan dengan XSP Cocoon 2.x

Internet sudah tidak dipungkiri menjadi sebuah media yang strategis untuk pengembangan *software*. Apalagi setelah banyak aplikasi bisnis mem-*porting* aplikasinya menjadi *web-enabled*, maka secara tidak langsung pengembangan aplikasi web menjadi sangat penting.

Ide pengembangan aplikasi web dengan hasil akhir berbentuk HTML dipelopori oleh ASP (*Active Server Pages*) yang dibuat oleh Microsoft, bersamaan dengan launching ADO (*Active Data Object*) 1.0, dengan IDE-nya Microsoft Visual Interdev. Sebelumnya, kita menggunakan editor teks biasa untuk pengembangan aplikasi atau dengan FrontPage yang saat itu merupakan desainer HTML terbaik.

Tidak lama berselang, popularitas ASP dilawan oleh PHP buatan **Rasmus** (*http://www.php.net*) yang bekerja secara *native* dan bisa berjalan di banyak *platform* mulai dari Windows, Linux sampai Solaris. PHP bekerja sangat cepat. Sampai artikel ini ditulis, sepertinya belum ada *parsing* teknologi yang dapat mengalahkannya. Saat artikel ini dibuat, PHP versi 5.0 beta 1 baru dikeluarkan.

Ternyata ada satu teknologi yang ketinggalan, yaitu Java. Saat itu posisinya sebagai aplikasi web hanya seputar *applet* dan *cgi development* dengan Servlet. Sun sebagai pencipta Java memperkenalkan extension Servlet, yaitu JSP, sebuah teknologi yang memungkinkan programer Java membuat aplikasi HTML. JSP merupakan *extended version* dari Servlet.

JSP/Servlet ini bekerja berbeda dengan ASP maupun PHP yang ter-*embed* dalam http server, baik itu Apache Httpd ataupun IIS. Teknologi JSP/Servlet lebih memilih pendekatan pengembangan aplikasi server tersendiri yang terkoneksi secara *proxy* dengan *web server*-nya, walaupun ada juga yang mengembangkan versi ISAPI-nya, tetapi sepertinya kurang populer.

JSP bekerja mirip sekali dengan ASP dan PHP. Memang Sun mencontek habis cara kerja ASP maupun PHP. Yang membedakannya adalah kemampuan OOP-nya Java teradopsi ke dalam JSP. Ini hal yang sulit sekali dilakukan oleh ASP maupun PHP.

Para programer Java di seluruh dunia tidak tinggal diam, dan keluarlah sebuah *alternative programming* dari tangan dingin mereka yang dipelopori **Stefano**, yaitu XSP (*Extensible Server Pages*). XSP adalah suatu teknik pemograman yang memungkinkan kita meng-embed code program ke dalam XML, baik itu PHP, JavaScript (Rhino), Rexx, ataupun Java itu sendiri.

Berkenalan dengan Cocoon

XSP adalah fitur dari sebuah *project open source* dari Apache.org, yaitu Cocoon. Apache.org adalah sebuah komunitas Java Open Source populer, yang juga merupakan penyedia project web server terpopuler Apache.

Sebelum berkenalan dengan XSP, kita harus mengenal dulu Cocoon, sebuah web *publishing framework* yang dicetuskan idenya oleh **Stefanno Mazzochi** saat menjadi mahasiswa di Italia. Maklum, XSP berjalan di atas Cocoon, jadi kalau Cocoon *engine* tidak terinstal dan ter-*setting* dengan baik XSP tidak bisa berjalan. Saat artikel ini dibuat, Cocoon 2.1 versi M1. Sedangkan semua *script* pada artikel ini dibuat menggunakan Cocoon 2.0.4.

Cocoon adalah salah satu project dari Apache XML terpopuler, yang membuat Cocoon di-*promote* ke level sub-*domain*. Sehingga Cocoon menjadi setingkat dengan HTTPd (Apache Web Server) dan Apache Jakarta ataupun Apache XML itu sendiri, tempat Cocoon berasal.

Untuk menginstal Cocoon sebenarnya tidak sulit, *download* saja *binary distribution*-nya dari *http://cocoon.apache.org*. Untuk menjalankan Cocoon, pertama-tama harus melakukan instalasi Tomcat. Harap diperhatikan JSDK yang dipakai, kalau JSDK 1.3 dapat menggunakan Tomcat yang *full edition*, tetapi JSDK 1.4.x diharapkan pakai yang Tomcat LE (*Limited Edition*). Hal ini dikarenakan semenjak versio 1.4, JSDK memasukkan Crimson (XML Parser) dan Xalan (XML Transformation) ke dalam distribusinya.

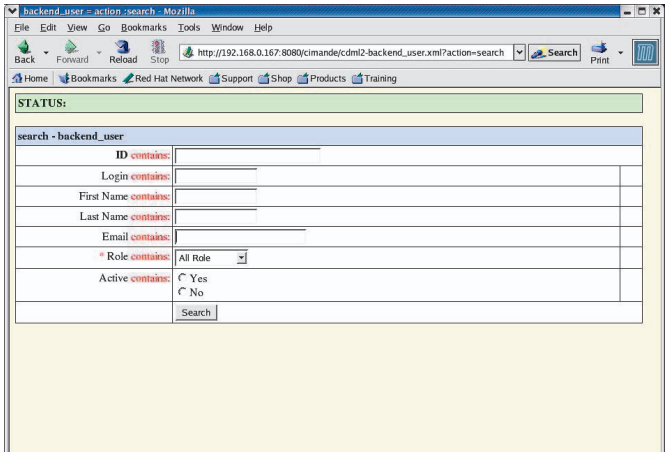
Setelah Tomcat berjalan dengan baik, baru lah Cocoon dimasukkan ke dalam container JSP/Servelt-nya Tomcat. Cara termudahnya adalah dengan memindahkan cocoon.war ke dalam webapps dan melakukan *restart* Tomcat.

Bila telah berhasil akan keluar pesan di console Tomcat atau di *log*-nya seperti berikut:

```
Opening database: C:\jakarta-tomcat-4.1.24-LE-jdk14\webapps\cocoon\WEB-INF\db\cocoondb
HSQLDB server 1.7.1 is running
Use SHUTDOWN to close normally. Use [Ctrl]+[C] to abort abruptly
Sun Jul 13 01:16:08 GMT+07:00 2003
Listening for connections ...
```

Cocoon 2.x secara default mengaktifkan HSQLDB sebagai server. HSQLDB adalah 100% pure Java RDBMS. *Nah*, kalau Cocoon sudah jalan, tinggal coba di *browser* dengan mengetik *http://localhost:8080/cocoon.* Bila keluar Cocoon pages dengan tulisannya *"The Apache Software Foundation is proud present Cocoon",* berarti instalasi telah selesai dan kita siap belajar apa itu XSP.

Sebelum mendalami XSP, ada baiknya pembaca membaca dokumentasi yang disertakan, terutama mengenai *formatting sitemap*. Karena semenjak 2.x, arstitektur Cocoon berubah secara mendasar.



Contoh halaman web xml

Cocoon diakses dari Mozilla

XSP 1.0 diperkenalkan oleh Cocoon 1.x. Yang terkenal adalah Cocoon 1.8.2 yang dimasukkan dalam paket Borland JBuilder 7.0. Untuk Cocoon 1.8.2 ada tutorial yang cukup terkenal yang dibuat oleh **Brett McLaughlin** dalam bukunya *Java XML* keluaran ORielly. Brett adalah sang pencipta JDOM bersama-sama dengan **Jason Hunter** (pengarang buku *Servlet Programming*). Untuk project Cocoon 1.8.2 yang lebih *advanced* bisa mencoba *script* yang penulis buat, yaitu BlueOxygen Cimande yang dapat di-download di *http://www.sourceforge.net/projects/cimande.* Cimande adalah sebuah *Workspace Framework* untuk bisnis.

XSP pada Cocoon 2.0 dan Cocoon 1.0 itu secara *syntax* tidak jauh berbeda, tetapi ada beberapa *tags* yang dihilangkan. Hal ini dikarenakan Cocoon 1.x dan Cocoon 2.x menggunakan teknologi *caching* yang berbasis sitemap.xmap. Jadi kalau bermain-main dengan Cimande, pasti pusing dengan konsep Cocoon 2.0. Ini juga yang masalah yang tidak pernah ditulis bahwa seperti seringnya terjadi ketidakkompatibelan antar-project open source yang berbeda versi.

Berikut ini adalah sebuah script dari XSP dengan bahasa yang digunakan Java.

```
<xsp:page language="java">
  <xsp:logic>
    <![CDATA[
    for (int i=0; i<3; i++)
    {]]><![CDATA[
    ]]>
      <li>
        Item <xsp:expr>i</xsp:expr>
      </li>
    <![CDATA[
    }]]><![CDATA[
    ]]>
  </xsp:logic>
</xsp:page>
```

Script di atas kalau dieksekusi akan melakukan *looping* seperti layaknya perintah *for* pada Java. Selebihnya adalah perintah XML standard, seperti CDATA. Untuk mempermudah pemahaman XSP, pembaca harus mengerti juga syntax transformasi XML. Penulis mengacu pada bukunya **Michael Kay** untuk *XSLT Reference*, jadi bisa membedakan *syntax standard* XSLT dan XSP.

Kelebihan XSP

XSP bekerja berbeda dengan bahasa pemograman ASP, JSP, ataupun PHP. XSP memungkinkan memasukan *code* ke dalam metadata maupun transformationnya, dan juga memungkinkan terjadi manipulasi hasil akhirnya. Ini adalah sebuah metode yang tidak mungkin terjadi pada ASP, JSP, PHP, ataupun bahasa pemograman lain. Karena kemampuan ini, maka XSP memungkinkan menghasilkan sebuah program *publishing* yang sangat efektif dan cepat.

Malah saat ini Cocoon dengan XSP-nya diposisikan sebagai *backend model* untuk pemograman berbasis MVC, walaupun masih ditemukan beberapa masalah saat melakukan integrasi dengan Struts.

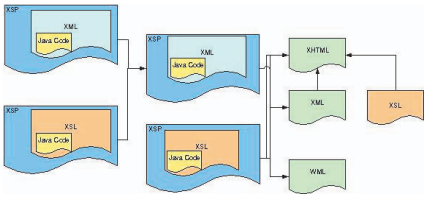
Cocoon juga dapat diintegrasikan dengan Velocity yang memungkinkan terjadi pengembangan berbasis *template*. Maklum, Velocity memungkinkan terjadi pemisahan code program antara *business logic* dan *presentation layer*. Akibatnya, aplikasi dengan Velocity bisa bekerja 4x lebih cepat dibandingkan dengan JSP.

Untuk mempermudah pemahaman cara kerja XSP, lihatlah ilustrasi transformasi XML. XSP dapat dikatakan merupakan sebuah extension dari transformasi XML / XSLT. XSP dapat di-embed dalam XML maupun XSLT, malah *output* transformasi akan tereksekusi secara *server side* bila output-nya berformat XSP. Tentu saja semua ini tergantung pada setting-nya.

Transformasi XML bisa dilakukan secara terus menerus sampai didapat output yang dikehendaki. Hal yang memerlukan tenaga ekstra bila melakukan pemograman dengan parser XML lainnya seperti Xerces ataupun Crimson, walaupun itu JDOM sekalipun. Malah integrasi dengan *persistant object*, atau XML binding lainnya akan membuat XSP lebih asyik. Maklum, XSP adalah sebuah teknologi yang membuat Java memiliki nilai tambah untuk pemograman web. Ini yang tidak dimiliki oleh bahasa apapun. Kita bisa membuat aplikasi Python di dalam XSP dengan tambahan component Jython.

Bekerja dengan transformasi XML

Sebelum dapat memulai kode XSP yang pertama, pembaca harus mengerti cara bagaimana mentransformasikan XML menjadi XHTML dengan XSL. Karena umumnya programer XSP bekerja mengacu dari sana. XSP dapat dikatakan sebagai cara membuat transformasi XML menjadi sangat dinamis.



Transformasi-XML

Kasus di bawah akan menjelaskan bagaimana sebuah transformasi greeting.xml menjadi greeting.html berdasarkan *stylesheet greeting.xsl*. Untuk membuat *script greeting*.html, kita dapat menggunakan editor teks/HTML biasa dan printahnya sederhana sekali. Kalau kita membukanya dengan browser, hasilnya adalah sebuah text bertulisan "Hello, world!" dengan heading H1.

```
<html>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

Nah, untuk membuat HTML di atas dengan metode transformasi XML, maka harus dibuat satu metadata (XML) dan transformasinya (XSL) yaitu greeting1.xml:

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

dan sebuah file greeting1.xsl:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://
www.w3.org/1999/XSL/Transform"
version="1.0">

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

</xsl:stylesheet>
```

File greeting.xsl akan mencari *tag root* dari XML. Bila ditemukan, akan menghasilkan keluaran, tag xsl:value-of akan diganti dari nilai value dari tag <greeting>

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

Untuk kasus di atas, apa yang terjadi bila kita hendak membuat sebuah sumber data (greeting.xml) yang dinamis, sehingga value greeting bukan hanya Hello, world! tetapi (misalnya) nama pengakses, IP address pengakses, ataupun berdasarkan session login dari pengakses? Untuk pemogramanan web services, greeting.xml bisa saja berbentuk format SOAP, UDDI, WSDL, ebXML ataupun XPDL. Untuk project BlueOxygen Cimande, file greeting.xml adalah CDML.

Kemudian apa yang terjadi bila output-nya bukan XHTML seperti yang dijelaskan di atas, tetapi bisa juga sebuah XML lainnya, SOAP, ebXML, atau WML? Apa yang terjadi bila template XSL-nya ada 100? Apakah hal ini bisa dilakukan dengan *programming embed* HTML yang lain seperti ASP, PHP, ataupun JSP/Servlet?

Dalam pemograman konvensional non-XSP, maka kita harus membuat file transformernya satu-satu, tetapi dengan XSP kita hanya perlu 1 file XML yang dinamis dan 1 file XSL yang dinamis, serta *setting sitemap*. Efisien bukan, tetapi tentu saja kalau ada *template* XSL 100 buah, akan semakin kompleks XSP-nya. Penulis sudah mencoba menggunakan 3 XSL saja, sudah susah di-*debug*. Maklum saat ini belum ada XSP Debugger.

Dari hasil analisis penulis, sebuah XSP yang dinamis di kedua belah pihak baik XML maupun XSL akan bereksponensial tajam, bila terjadi transformasi lebih dari 2 kali transformasi. Malah XML yang 15 tag itu bisa ekivalen dengan 1000 baris code Java yang sudah di-macro-kan. Di-macro-kan artinya kita telah membuat code-code atau object sehingga ke 1000 baris code tersebut tinggal panggil *component*-nya. Ekstrim sekali, bukan?

Bekerja dengan XSP

Untuk mendinamiskan file greeting.xml, caranya mudah sekali. Tambahkan tag <xsp:page>, maka secara otomatis file xml tersebut akan menjadi file xsp dan bila dimasukkan ke dalam *container* Cocoon, dia akan dieksekusi otomatis (dengan menambahkan type-nya serverpages di sitemap).

Di bawah ini adalah contoh file myxsp.xsp yang dibuat berdasarkan greeting1.xml.

```
<?xml version="1.0"?>

<xsp:page xmlns:xsp="http://apache.org/
xsp" xmlns:greeting="http://
blueoxygen.org/tutorial/greeting">
<page>
  <xsp:logic>
    // this could be arbitrarily complex Java
code, JDBC queries, etc.
    String msg = "Hello, myXSP.xsp!";
  </xsp:logic>

  <greeting>
    <xsp:expr>msg</xsp:expr>
    <greeting:hello-world/>
  </greeting>
</page>
</xsp:page>
```

File myxsp.xsp akan mengubah *string* msg dan memasukkannya ke dalam tag greeting. Nah, bisa dilihat *script* di atas adalah Java code. Dalam tag <xsp:logic> kita memungkinkan untuk mengganti atau menambah Java code-nya, seperti membaca database, membaca session, atau memanggil component lainnya.

Apa yang terjadi kalau ternyata hasil transformasi di atas itu masih kurang? Dengan XSP, memungkinkan sebuah output melakukan kompilasi lagi yaitu dengan memasukan code XSP pada XSL-nya, contohnya di bawah ini (xsp2mainxsp.xsl):

```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/
XSL/Transform"
  xmlns:xsp="http://apache.org/xsp"
  version="1.0">

<xsl:template match="/">
<xsp:page>
 <xsl:copy>
  <xsl:apply-templates/>
 </xsl:copy>
</xsp:page>
</xsl:template>

<xsl:template match="greeting:hello-
```

```
world">
  <!– more complex XSLT is possible here as well –>
<xsl:attribute name="name">frans</xsl:attribute>
  <xsp:logic>
    // this could be arbitrarily complex Java code, JDBC queries, etc.
    String msg = "Hello, XSL!";
  </xsp:logic>
  <xsp:expr>msg</xsp:expr>
</xsl:template>

<!– This template simply copies stuff that doesn't match other –>
<!– templates and applies templates to any children.  –>
<xsl:template match="@*|node()" priority="-1">
 <xsl:copy>
  <xsl:apply-templates select="@*|node()"/>
 </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

Hasil output myxsp.xsp tersebut dapat ditransformasikan kembali menggunakan xsp2mainxsp.xsl. Hasil output transfromasi myxsp.xsp dengan xsp2mainxsp.xsl adalah sebuah XSP.

Setting Sitemap

Sitemap dalam Cocoon 2.x adalah darahnya, sitemap yang disimpan dalam 1 context Cocoon akan mengacu pada sitemap.xmap. Setiap sitemap.xmap akan mem-*parsing* semua file yang ada di dalam direktori tempat sitemap.xmap berada dan juga direktori di bawahnya.

Ada beberapa trik yang harus dilakukan agar Cocoon mau menganggap sebuah file itu adalah file XSP, yaitu menambahkan perintah type="serverpages" pada sitemap.xmap dalam salah satu *pipeline*-nya.

Untuk file di atas, cobalah masukkan baris berikut pada map:pipeline:

```
<map:match pattern="main.html">
  <map:generate src="cocoon:/mainxsp2"/>
  <map:transform src="workspace/greeting.xsl"/>
  <map:serialize type="html"/>
</map:match>

<map:match pattern="mainxsp2">
  <map:generate type="serverpages" src="cocoon:/mainxsp"/>
  <map:transform src="workspace/greeting.xsl"/>
  <map:serialize type="html"/>
</map:match>

<map:match pattern="mainxsp">
  <map:generate type="serverpages" src="workspace/myxsp.xsp"/>
  <map:transform src="workspace/xsp2mainxsp.xsl"/>
  <map:serialize type="xml"/>
</map:match>
```

Atau untuk lebih mudahnya lihatlah contoh sitemap.xmap pada cimande-0.2a.war. *Nah*, tag sitemap di atas ini berarti perintah untuk melakukan transformasi 2x, berupa output transformasi pertama antara mainxsp (myxsp.xsp dengan xsp2mainxsp.xsl) dengan greeting.xsl. Output yang dikeluarkan adalah XHTML yang sama dengan greeting.html.

Untuk mengaksesnya, dapat mengetik http://localhost:8080/cocoon/mainxsp untuk melihat hasil output transformasi pertama, atau http://localhost:8080/cocoon/main.html atau http://localhost:8080/cocoon/mainxsp2.

Sebenarnya, main.html dan mainxsp2 itu sama saja, tetapi karena patternnya main.html, mungkin pengakses tidak dapat membedakan antara file dinamis XSP dengan HTML biasa. Ini adalah trik untuk orang yang mengakses, bahwa file yang diakses statis, padahal dinamis.

Untuk lebih mudahnya, coba instalasi cimande-0.2a.war yang dapat di-download dari *http://www.blueoxygen.org/download.* Untuk menjalankan code di atas, juga disertakan script yang lebih kompleks yang memungkinkan terjadi transformasi berdasarkan *session*. Kalau cimande-0.1 telah berhasil diinstal, cimande-0.2a adalah versi berikutnya yang belum selesai. Jika ada di antara pembaca yang mau melakukan *upgrade* cimande 0.1 yang ada di *sourceforge* sehingga mendukung Cocoon 2.x, dapat bekerja sama dengan mengirim e-mail ke penulis.

Frans Thamura *(frans@blueoxygen.org)*