Cocoon – środowisko publikacyjne oparte na XML-u

Szymon Zioło
empolis Polska sp. z o.o.
ul. Płocka 5a
02-776 Warszawa
szz@empolis.pl

Streszczenie

Cocoon jest darmowym, rozwijanym przez środowisko open source, narzędziem do tworzenia witryn internetowych i aplikacji webowych. Dzięki wykorzystaniu XML-a i sprytnego mechanizmu przekształceń, pozwala twórcom witryny na oddzielenie kompetencji autorów tekstów od autorów układu graficznego i od zarządzania strukturą witryny.

Wprowadzenie

W przygotowanie i aktualizowanie witryny internetowej – poza trywialnymi przypadkami np. stron osobistych – zaangażowanych jest zwykle co najmniej kilka osób, pełniących różne role. Autorzy przygotowują treść publikowanych tekstów. Graficy opracowują układ graficzny stron oraz styl graficzny samych artykułów. Wreszcie redaktor planuje logiczną strukturę witryny oraz powiązania pomiędzy poszczególnymi działami, stronami i tekstami.

Gdy witryna tworzona jest w tradycyjny sposób, oddzielenie tych ról od siebie sprawia trudności. Członkowie zespołu pełniący różne role mogą łatwo wchodzić w zakres kompetencji innych ról. Jeśli np. przyjmiemy konwencję, zgodnie z którą autorzy tworzą teksty bezpośrednio w HTML-u, to będą oni mogli w niekontrolowany sposób wpływać na ich styl graficzny. Użycie przez autorów różnych konwencji formatowania spowoduje utratę spójności graficznej witryny. Jednak jeśli pozwolimy grafikom modyfikować zaproponowane przez autorów formatowanie, lub wręcz przyjmiemy, że artykuły powstają w postaci tekstu niesformatowanego, a całość formatowania nakłada grafik, to pojawia się niebezpieczeństwo, że tekst zostanie sformatowany niewłaściwie, niezgodnie z intencjami autora. Podobna relacja może wystąpić pomiędzy redaktorem a grafikiem. Grafik bowiem, z racji tego, że przygotowuje układ graficzny stron, może wpływać na logiczną strukturę witryny.

Na kłopoty – XML

Problem ten – tam, gdzie rzeczywiście jest dokuczliwy - zwykło się rozwiązywać, decydując się na kodowanie tekstów w XML-u lub (w czasach przed-XML-owych) w SGML-u. Kluczową zasadą obu technik jest oddzielenie znaczenia tekstu od sposobu jego prezentacji. Wiąże się to z oznakowaniem tekstu w sposób abstrakcyjny, przy pomocy znaczników opisujących jego znaczenie. Autor dokumentu może więc korzystając z takiego repertuaru znaczników precyzyjnie wyrazić, co *oznacza* dany fragment tekstu. Grafik zaś przy pomocy arkusza stylów zdecyduje, w jaki sposób tekst o określonym znaczeniu będzie sformatowany (np. nazwy zmiennych programistycznych sformatuje czcionką o stałej szerokości znaków). Wszystkie teksty, nawet pochodzące od różnych autorów, będą dzięki temu sformatowane w sposób jednolity, bez ryzyka, że np. grafik pomyli nazwę zmiennej z nazwą pliku i sformatuje ją kursywą. Potrzeba do tego jedynie narzędzia, które zaaplikuje arkusz stylów do dokumentów XML. Najczęściej bywa nim procesor XSL, wykonujący odpowiednie przekształcenie do docelowego formatu.

W ten sposób rozwiązujemy jednak tylko część problemu. Oddzielamy kompetencje autora dokumentu oraz grafika na poziomie pojedynczego dokumentu. Tymczasem jeśli mówimy o witrynie internetowej, mamy do czynienia z kolekcją powiązanych ze sobą dokumentów, które w dodatku są prezentowane na stronach o określonym układzie graficznym, z ustandaryzowanymi nagłówkami, stopkami, elementami menu, itp. Mamy także do czynienia z trzecią wspomnianą rolą - redaktorem.

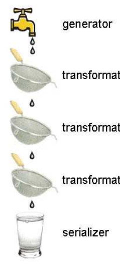
Nic nie stoi na przeszkodzie, aby rozszerzyć XML-owy pomysł oddzielenia znaczenia tekstu od jego formatowania także na strukturę witryny. Tyle że w tym wypadku potrzebujemy bardziej zaawansowanego narzędzia, potrafiącego nie tylko dokonać przekształceń, ale też złożyć w ustalony sposób zawartość stron z treści artykułów, nagłówka, stopki, wygenerować menu na podstawie informacji o strukturze witryny, itp.

Cocoon – separacja kompetencji

Jednym z ciekawszych narzędzi pozwalających tego dokonać, jest platforma Cocoon, napisana w Javie i rozwijana jako inicjatywa *open-source* w ramach projektu Apache. Ideą przyświecającą jej twórcom jest separacja kompetencji (*separation of concerns*). Zgodnie z nią, podczas tworzenia publikacji webowej mamy do czynienia z czterema obszarami kompetencji – wcześniej wspomnianymi logiką, zawartością i stylem, oraz obszarem zarządzania, definiującym tzw. kontrakty (punkty styku) pomiędzy pozostałymi rolami.

Od strony technicznej, w bardzo dużym uproszczeniu, Cocoon można sobie wyobrazić jako narzędzie wykonujące po stronie serwera transformacje źródłowych dokumentów XML oraz serwujące wyniki tych transformacji. To zresztą jest najczęstszy sposób jego wykorzystania. Ale taka transformacja jest jedynie trywialnym przykładem możliwości, jakie daje Cocoon. Przekształcenia można bowiem łączyć w tzw. łańcuchy przetwarzania (*pipelines*), w których wynik jednego przekształcenia jest przekazywany na wejście do kolejnego. Dostępny repertuar przekształceń nie ogranicza się jedynie do XSLT, co pozwala na wykonywanie nietrywialnych zadań w ramach odpowiednio skonfigurowanego łańcucha. To właśnie dzięki umiejętnemu łączeniu przekształceń możliwa jest separacja kompetencji. Oprócz prostych transformacji dokumentów XML do ich postaci wizualnej (np. HTML), możemy bowiem skorzystać np. z przekształceń konstruujących zawartość wynikowej strony z treści artykułu (być może już wstępnie przekształconej), nagłówka i stopki strony oraz elementów nawigacyjnych, wygenerowanych na podstawie pliku definiującego strukturę witryny.

Model przetwarzania w Cocoonie jest oparty na przekazywaniu pomiędzy kolejnymi składnikami (komponentami) łańcucha przetwarzania tzw. ciągu zdarzeń SAX. SAX (Simple API for XML) jest standardem przetwarzania dokumentów XML, zgodnie z którym zawartość dokumentu traktuje się jako ciąg zdarzeń, takich jak np. pojawienie się znacznika początkowego, końcowego czy tekstu. Cocoon wykorzystuje ten mechanizm w przewrotny sposób, ponieważ posługuje się ciągami zdarzeń nie mającymi odzwierciedlenia w konkretnych, fizycznych dokumentach. Wprawdzie te ciągi zdarzeń reprezentują pewne abstrakcyjne dokumenty, lecz dokumenty te nie są nigdzie fizycznie zapisywane – istnieją wyłącznie w postaci ulotnych ciągów zdarzeń, przekazywanych bezpośrednio pomiędzy kolejnymi elementami łańcucha przetwarzania.



Rysunek 1: Model przetwarzania platformy Cocoon

Rysunek 1 przedstawia ogólny model przetwarzania oferowany przez Cocoon. W procesie tworzenia wyjściowego dokumentu w silniku Cocoon biorą udział co najmniej następujące komponenty:

- generator,
- transformator,
- serializer.

Zadaniem generatora jest wygenerowanie – na podstawie pewnego źródła danych – ciągu zdarzeń SAX. Tym źródłem danych zwykle jest po prostu plik, ale może nim być także np. baza danych. Dzięki tej możliwości nie trzeba zapisywać wyników eksportu z bazy danych do postaci fizycznego pliku – można bezpośrednio z bazy wygenerować odpowiedni ciąg zdarzeń. Zdarzenia otrzymane od generatora są przekazywane transformatorowi, który wykonuje pewne przekształcenia wejściowego ciągu zdarzeń i generuje inny ciąg wyjściowy. W ten sposób można połączyć ze sobą kilka transformatorów, z których kolejny przyjmuje na wejście zdarzenia wygenerowane przez poprzedni. Na końcu całego łańcucha przekształceń znajduje się serializer. Jego zadaniem jest wygenerowanie na podstawie ciągu zdarzeń dokumentu, który zostanie przesłany do klienta.

Pełny model przetwarzania Cocoon-a jest bardziej skomplikowany, niż przedstawiony powyżej, i dopuszcza użycie także innego rodzaju komponentów (np. agregatorów – łączących kilka niezależnych ciągów zdarzeń w jeden). Zwykle jednak w zupełności wystarcza podstawowy model przetwarzania. Co więcej, najczęściej nie trzeba implementować poszczególnych elementów ciągu przetwarzania, gdyż implementacja Cocoon-a zawiera najczęściej używane generatory, transformatory oraz serializery, m. in:

- generator z pliku,
- generator z katalogu w systemie plików,
- transformator XSLT (dla którego trzeba oczywiście przygotować odpowiednią transformację XSLT),

- serializery do plików XML, HTML, SVG, PDF.

Przykład: książka kucharska

Możliwości Cocoona najlepiej będzie zilustrować na konkretnym przykładzie. Wyobraźmy sobie, że gromadzimy przepisy kulinarne w postaci dokumentów XML i chcemy je publikować w sieci. Przepisy mają jednolitą, ustaloną strukturę (są zgodne z pewnym schematem). Oto przykład takiego przepisu:

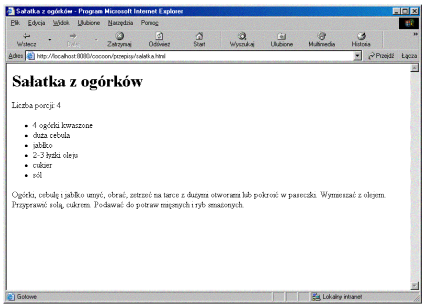
```
<?xml version="1.0" encoding="UTF-8"?>
<przepis ilosc-osob="4">
  <tytul>Sałatka z ogórków</tytul>
  <skladniki>
    <skladnik>4 ogórki kwaszone</skladnik>
    <skladnik>duża cebula</skladnik>
    <skladnik>jabłko</skladnik>
    <skladnik>2-3 łyżki oleju</skladnik>
    <skladnik>cukier</skladnik>
    <skladnik>sól</skladnik>
  </skladniki>
    <wykonanie>Ogórki, cebulę i jabłko umyć,
obrać, zetrzeć na tarce z dużymi otworami
lub pokroić w paseczki. Wymieszać z olejem.
Przyprawić solą, cukrem. Podawać do potraw
mięsnych i ryb smażonych.</wykonanie>
</przepis>
```

Dla uproszczenia przyjmijmy, że nasze przepisy będą składowane jako pliki w wybranym katalogu na dysku. W realnym zastosowaniu pochodziłyby one zapewne z systemu zarządzania treścią, gdzie podlegałyby procesowi redakcji, korekty i akceptacji.

Wyświetlenie konkretnego przepisu w postaci sformatowanej jest bardzo proste. Wymaga jedynie stworzenia odpowiedniego przekształcenia XSLT oraz skonfigurowania prostego łańcucha przetwarzania, składającego się z generatora z pliku, transformatora XSLT oraz serializera do pliku HTML. Konfiguracja łańcuchów przetwarzania zapisana jest w pliku konfiguracyjnym Cocoona `sitemap.xmap`, który w tym celu musimy zmodyfikować, dodając do niego deklarację łańcucha przetwarzania:

```
<map:match pattern="przepisy/*.html">
  <map:generate src="przepisy/{1}.xml"/>
  <map:transform
    src="przepisy/xsl/przepis2html.xsl"/>
  <map:serialize/>
</map:match>
```

Deklaracja ta będzie używana wówczas, gdy żądany URL uda się dopasować do wzorca podanego w atrybucie `pattern` elementu `match`. Do poprawnego działania tak zadeklarowanego łańcucha niezbędne jest oczywiście istnienie przekształcenia `przepis2html.xsl`. Jest to jednak dość proste przekształcenie, więc pominiemy w tym miejscu jego kod. Efekt jego działania może być np. taki jak na rys. 2.



Rysunek 2: Przykładowa wizualizacja przepisu kulinarnego

W ten sposób udało nam się w miarę bezboleśnie oddzielić kompetencje autora przepisu i grafika odpowiedzialnego za jego wizualizację. Zajmijmy się teraz organizacją witryny. W naszym przykładzie będzie ona miała postać listy przepisów dostępnych w książce kucharskiej.

Jeśli pliki z przepisami są składowane w wybranym, konkretnym katalogu, to łańcuch przetwarzania generujący listę przepisów powinien zaczynać się od generatora z katalogu, który odczytuje listę plików we wskazanym katalogu i generuje ciąg zdarzeń opisujący kolejne pliki. Dokument odpowiadający takiemu ciągowi zdarzeń wygląda np. tak:

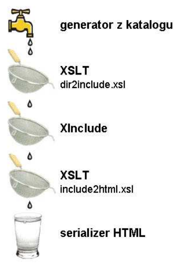
```
<?xml version="1.0" encoding="UTF-8"?>
<dir:directory name="przepisy"
 lastModified="1058127098000"
 date="03-07-13 22:11" requested="true"
 xmlns:dir="http://apache.org/cocoon/directory/2.0">
  <dir:file name="salatka.xml"
   lastModified="1057959740000"
   date="03-07-11 23:42"/>
  <dir:file
    name="Pierogi_z_baranina_i_szynka.xml"
    lastModified="1058207994000"
   date="03-07-14 20:39"/>
  <dir:file name="Bliny_na_piwie.xml"
   lastModified="1058206830000"
   date="03-07-14 20:20"/>
</dir:directory>
```

Widać więc, że informacja w nim zawarta nie wystarczy nam do wyświetlenie czytelnej listy przepisów, ponieważ np. nie zawiera pełnych tytułów przepisów, a jedynie nazwy plików. Tytuły są zaś zawarte w treści samych plików.

I tu przyjdzie nam z pomocą związany z XML-em standard XInclude, opisujący m.in. sposób włączania do treści dokumentu XML fragmentów innych dokumentów. Tak się dobrze składa, że wśród standardowych transformatorów Cocoona jest transformator zgodny ze standardem XInclude. Musimy jedynie odpowiednio przygotować dla niego ciąg zdarzeń o zawartości katalogu – umieszczając w nim specjalne odsyłacze XInclude, tak, aby transformator wiedział, w którym miejscu i co wstawić. Odsyłacz taki wygląda na przykład tak:

```
<xi:include
  href="dok.xml#xpointer(/przepis/tytul)"/>
```

gdzie `xi:` jest przedrostkiem przestrzeni nazw XInclude. Podczas działania procesora XInclude, odsyłacz ten zostanie zastąpiony węzłem wskazywanym przez ścieżkę `/przepis/tytul`, a więc elementem tytul, pobranym z pliku `dok.xml`. Wygenerowanie strony HTML z czytelnym spisem treści książki kucharskiej musi być zatem obsługiwane przez nieco bardziej złożony niż poprzednio ciąg transformacji (por. rysunek 3).



Rysunek 3: Ciąg przetwarzania generujący listę przepisów

W pierwszej transformacji XSLT – `dir2include.xsl` – wstawiamy odpowiednie odsyłacze XInclude oraz usuwamy zbędne informacje (np. daty ostatniej modyfikacji):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dir="http://apache.org/cocoon/directory/2.0"
 xmlns:xi="http://www.w3.org/2001/XInclude">
  <xsl:template match="dir:directory">
    <directory><xsl:apply-templates/></directory>
  </xsl:template>

  <xsl:template match="dir:file">
    <file name="{@name}">
      <xi:include
href="przepisy/{@name}#xpointer(/przepis/tytul)"/>
    </file>
  </xsl:template>
</xsl:stylesheet>
```

Na wynikach tego przekształcenia uruchamiany jest transformator XInclude. Dokument opisujący zawartość katalogu po przetworzeniu przez ten transformator – a więc już zawierający komplet niezbędnych nam informacji – wygląda na przykład tak:

```
<?xml version="1.0" encoding="UTF-8"?>
<directory
  xmlns:xi="http://www.w3.org/2001/XInclude"
  xmlns:dir="http://apache.org/cocoon/directory/2.0">
  <file name="salatka.xml">
    <tytul>Sałatka z ogórków</tytul>
  </file>
  <file name="Pierogi_z_baranina_i_szynka.xml ">
    <tytul>Pierogi z baraniną i szynką
   w cieście drożdżowym</tytul></file>
  <file name="Bliny_na_piwie.xml">
    <tytul>Bliny na piwie</tytul>
  </file>
</directory>
```

Ostatnia transformacja XSLT – `include2html.xsl` – tworzy na podstawie tych informacji docelowy dokument HTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="directory">
    <html>
      <head>
        <title>Internetowa książka kucharska</title>
      </head>
      <body>
        <h1>Internetowa książka kucharska</h1>
        <p><xsl:apply-templates/></p>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="file">
    <a href="{substring(@name, 1,
     string-length(@name)-4)}.html">
    <xsl:value-of select="tytul"/></a>
    <br/>
  </xsl:template>
</xsl:stylesheet>
```

W transformacji tej musimy użyć dość skomplikowanego wyrażenia, aby rozszerzenie nazw plików `.xml`, występujące w nazwach w wejściowym ciągu zdarzeń, zamienić na `.html`, rozpoznawane przez skonfigurowany wcześniej ciąg zdarzeń dla pojedynczego przepisu. W tym celu obcinamy ostatnie cztery znaki nazwy pliku (są to zawsze znaki `.xml`), po czym dodajemy prawidłowe rozszerzenie.

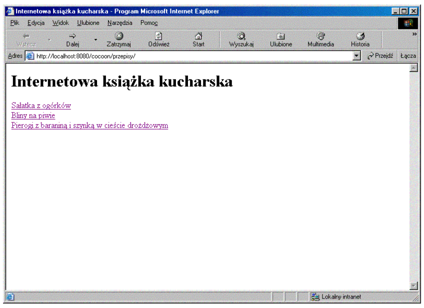
Mając gotowe wszystkie transformacje, możemy skonfigurować w pliku `sitemap.xmap` pełny łańcuch przetwarzania:

```
<map:match pattern="przepisy/">
  <map:generate type="directory"
   src="przepisy">
    <map:parameter
     name="include"
```

```
    value=".*\.xml"/>
  </map:generate>
  <map:transform
   src="przepisy/xsl/dir2include.xsl"/>
  <map:transform type="xinclude"/>
  <map:transform
   src="przepisy/xsl/include2html.xsl"/>
  <map:serialize/>
</map:match>
```

Będzie on wykonywany po wprowadzeniu URL-a `przepisy/`. Generator z katalogu – dzięki parametrowi include - wybierze dla nas tylko pliki z rozszerzeniem `.xml`. Kolejne trzy transformacje nie wymagają komentarza. Wygenerowanie dokumentu HTML jest domyślnym sposobem serializacji w Cocoonie, więc nie ma potrzeby podawania nazwy serializera.

W tym momencie możemy już przetestować pełną funkcjonalność aplikacji, wyświetlając listę przepisów (por. rys. 4) oraz nawigując z niej do treści poszczególnych przepisów. Organizacja naszej prostej witryny została zatem odseparowana od roli grafika (odpowiedzialnego za przekształcenie `include2html.xml`) i polega obecnie na umieszczaniu odpowiednich plików z przepisami w ustalonym katalogu.



Rysunek 4: Główna strona aplikacji – lista przepisów

Co dalej?

Referat przedstawia zaledwie wierzchołek góry lodowej możliwości Cocoon-a, który oddaje do dyspozycji kilkadziesiąt gotowych generatorów, transformatorów, serializerów i innych ciekawych komponentów. A jeśli nawet ten zestaw klocków nam nie wystarczy, zawsze możemy napisać własne.

Publikowanie przy pomocy platformy Cocoon jest poza tym zwykle jedynie końcowym etapem większego procesu, na który składa się planowanie zawartości witryny, zlecanie przygotowania treści, korekta i akceptacja artykułów, ewentualnie ich późniejsze aktualizacje. Proces ten jest wspierany przez specjalizowane systemy zarządzania dokumentami. My uprościliśmy go do absolutnego minimum, zakładając, że dokumenty do publikacji po prostu pojawiają się w określonym katalogu. Tymczasem w profesjonalnym środowisku wydawniczym Cocoon byłby jedynie odbiorcą dokumentów tworzonych w systemie zarządzania treścią i eksportowanych z niego do publikacji. Nie umniejsza to oczywiście randze tego narzędzia, ani nie pozbawia elegancji zastosowanemu w nim pomysłowi separacji kompetencji przy pomocy łańcuchów przekształceń.

Cocoon – technikalia

Z technicznego punktu widzenia, Cocoon jest serwletem. Do jego uruchomienia potrzebny jest więc kontener serwletów. Do celów testowych wystarczy darmowy silnik Tomcat, będący referencyjną implementacją technologii serwletów.

Trzecim elementem niezbędnym do uruchomienia witryny internetowej jest oczywiście serwer internetowy. Na szczęście Tomcat zawiera prosty serwer webowy, w zupełności wystarczający do zastosowań testowych. Nie jest więc potrzebne instalowanie i konfigurowanie osobnego serwera.

Konfiguracja zastosowana przez autora do uruchomienia przykładu: Java 2 SDK, Standard Edition 1.3.1_08, Apache Tomcat 3.3.1a, Apache Cocoon 2.0.4.

Literatura

[1] T. Knyziak, Apache Cocoon, Tele.Net Forum, wrzesień 2002, `http://www.telenetforum.pl/index_2.php?show=archiwum&art_id=98`

[2] The Apache Cocoon Project, `http://cocoon.apache.org`

[3] Wirtualna Kuchnia Polska, `http://kuchnia.verusinter.net`

[4] Zioło, S., *Cocoon i XML – książka kucharska*, Software 2.0 nr 9/2003, Wydawnictwo Software