Lech Madeyski

Wydziałowy Zakład Informatyki
Wydział Informatyki i Zarządzania
Politechnika Wrocławska
e-mail: madeyski@ci.pwr.wroc.pl

Paweł Mazur

Wydziałowy Zakład Informatyki
Wydział Informatyki i Zarządzania
Politechnika Wrocławska
e-mail: pmazur@pwr.wroc.pl

Nowoczesne aplikacje internetowe w praktyce

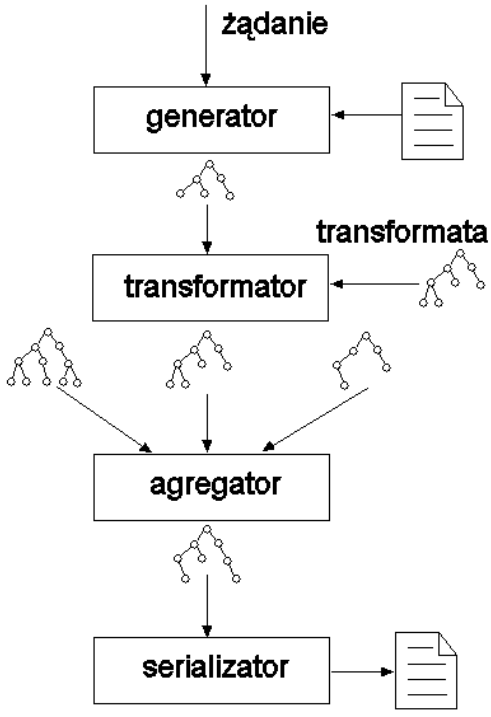
Utworzenie serwisu internetowego to, według wielu, zadanie ambitne i skomplikowane. To prawda. Ale wybierając odpowiednie środowisko i technologie można to zadanie znacznie uprościć. Służą temu najnowocześniejsze środowiska i rozwiązania szkieletowe (ang. *web application frameworks*), takie jak Apache Cocoon czy Apache Struts. Stały się one bardzo cennymi narzędziami w arsenale zespołów tworzących nowoczesne, zaawansowane technologicznie aplikacje internetowe. Niniejszy artykuł jest wynikiem doświadczeń w zakresie użycia najnowszych technologii zebranych podczas pracy nad projektem serwisu e-informatyka.pl. Stanowi jednocześnie praktyczne uzupełnienie artykułu Tomasza Knyziaka „Apache Cocoon", który ukazał się na łamach Telenetforum we wrześniu 2002 roku.

Apache Cocoon jest to środowisko publikacji przeznaczone do tworzenia serwisów i portali internetowych cechujących się dużą ilością dynamicznie

generowanych stron WWW. Twórcy powstającego właśnie elektronicznego czasopisma e-Informatyka.pl postanowili zrealizować serwis bazując na Cocoonie. Jest to prawdopodobnie pierwsze w Polsce wykorzystanie tej technologii do tak poważnych zastosowań. Jakie zatem cechy i mechanizmy Cocoona spowodowały, że wybrano właśnie to rozwiązanie?

XML

Podstawą Cocoona jest XML. Format ten dotyczy wszystkich plików konfiguracyjnych. Także generowanie wynikowej strony oparte jest na potoku przetwarzania (ang. *processing pipeline*) dokumentów XML. Zaletą takiego rozwiązania jest łatwa obsługa generowania wielu formatów wyjściowych (m.in. HTML, PostScript, PDF, WAP, XML) lub kilku wersji dla wybranego formatu (np. strona podstawowa, wersja do druku, wersja lekka posiadająca minimalną ilość grafiki) z tych samych danych wejściowych. Najczęściej realizuje się to używając innego potoku przetwarzania w zależności od postaci żądanego URL. Potoki przetwarzania definiuje się w tzw. mapach serwisu (ang. *site maps*) będących również dokumentami XML. Potok w najprostszej postaci składa się z generatora, który pobiera dane do potoku,



Rys. 1. Potok przetwarzania Cocoona
źródło: http://xml.apache.org

transformatora, który dokonuje obróbki danych (w najprostszym wypadku jest to transformata XSLT, ale są także bardziej zaawansowane) oraz serializatora, który wytwarza dokument wyjściowy. Schematycznie potok przetwarzania przedstawiony jest na Rys. 1.

```
<map:pipelines>
   <map:pipeline>
      <map:match pattern="przyklad_html">
         <map:generate src="wez_dane"/>
         <map:transform src="xsl/to_html.xsl"/>
         <map:serialize type="html"/>
      </map:match>
      <map:match pattern="przyklad_html_druk">
         <map:generate src="wez_dane"/>
         <map:transform src="xsl/to_html_druk.xsl"/>
         <map:serialize type="html"/>
      </map:match>
      <map:match pattern="wez_dane">
         <map:generate src="xml/dane.xml"/>
         <map:transform src="xsl/opcjonalna.xsl"/>
         <map:serialize type="xml"/>
      </map:match>
   </map:pipeline>
</map:pipelines>
```

Przykład zdefiniowania potoku w sitemapie

Widać, że tak działający mechanizm potoków jest odpowiedni dla serwisów typu czasopisma elektroniczne: wielość wynikowych formatów oraz możliwość pobierania danych z wielu źródeł (przy pomocy mechanizmów agregacji, o czym będzie mowa w dalszej części).

Java = komponenty

Cocoon może być uruchamiany poprzez interfejs linii komend (Command Line Interface) jako samodzielny program lub (co jest znacznie bardziej popularne) jako serwlet Javy działający w kontenerze serwletów. Zdecydowanie ułatwia to instalację, która polega na skopiowaniu archiwum WAR i restarcie serwera spełniającego rolę

kontenera serwletów (np. Apache Tomcat lub Jetty). Cocoon oferuje możliwość skorzystania ze standardowych jak i własnych komponentów, co wpływa na jego dużą elastyczność. Komponenty można pogrupować według następujących kategorii:

- generatory (ang. *generators*)
- transformatory (ang. *tranformers*)
- serializatory (ang. *serializers*)
- komparatory (ang. *matchers*)
- selektory (ang. *selectors*)
- akcje (ang. *actions*)
- czytniki (ang. *readers*)

```
<map:components>
    <map:generators/>
    <map:transformers/>
    <map:serializers/>
    <map:readers/>
    <map:selectors/>
    <map:matchers/>
    <map:actions/>
 </map:components>
```

Komponenty w sitemapie

W głównej sitemapie serwisu należy zdefiniować wszystkie używane komponenty podając ich nazwy, odpowiadające im klasy Javy implementujące dany komponent oraz ewentualne parametry. Dokumentacja Cocoona zawiera wszystkie potrzebne informacje o każdym komponencie dostarczanym w dystrybucji. Niektóre komponenty mają wbudowany mechanizm wykorzystujący pamięć podręczną (ang. *cache*). Sposób jego działania zależy od komponentu. Na przykład FileGenerator sprawdza datę ostatniej zmiany pliku zanim zadecyduje o jego odczycie.

Generatory mogą wystąpić tylko na początku potoku przetwarzania, gdyż ich celem jest pobranie danych z określonego źródła (nie koniecznie z pliku na dysku, może to być także między innymi wynik działania innego potoku, strony XSP lub JSP), skonwertowanie ich do XML (w istocie zdarzeń SAX) i przekazanie do dalszych elementów potoku. Nie jest jednak wymagane, by każdy potok rozpoczynał się od generatora. Czytniki można wykorzystać w przypadku takich zasobów, które nie wymagają przetwarzania, są już w pożądanym formacie (np. filmy, prezentacje czy obrazki) i powinny być efektywnie dostarczone do strumienia wyjściowego. Jeżeli potok zawiera generator, wówczas musi być zakończony serializatorem. Jeżeli zachodzi potrzeba obróbki danych, które pochodzą z niestandardowego źródła (np. wybrane elementy plików tekstowych, takie jak komentarze z kodów źródłowych) można wówczas napisać własny generator. Jest to zadanie bardzo proste, polegające na napisaniu klasy Javy implementującej jeden z interfejsów generatorów i nadpisaniu odpowiednich metod (np. w przypadku interfejsu AbstractGenerator wystarczy nadpisać metodę generate()). Dokładny opis tworzenia własnego generatora dostępny jest w dokumentacji Cocoona.

Tab.1. Generatory dostępne wraz z dystrybucją Cocoona

Nazwa generatora (nazwa klasy)	Generuje zdarzenia SAX...
File Generator [FileGenerator]	odczytując dane z dowolnego pliku lokalnego systemu plików lub dowolnego URL
HTML Generator [HTMLGenerator]	odczytując plik HTML i zamieniając go na XHTML
Directory Generator [DirectoryGenerator]	tworząc listing katalogu w postaci dokumentu XML
Image Directory Generator [ImageDirectoryGenerator]	działając tak jak Directory Generator, ale dla obrazków dodaje atrybut z opisem rozmiaru obrazka
Fragment Extractor Generator [FragmentExtractorGenerator]	Połączenie generatora z transformatorem; umożliwia usuwanie pewnych zdarzeń SAX ze strumienia i umieszczenie ich w osobnym potoku; zwykle stosowany do pobrania fragmentu obrazu w formacie SVG, przeniesienie go do innego potoku i zserializowania do formatu np. JPG
JSP Generator [JspGenerator]	na podstawie wyniku strony JSP, określonej w żądaniu HTTP (strona musi zwracać poprawny dokument XML)
Script Generator [ScriptGenerator]	wykonując skrypty z użyciem Bean Scripting Framework
Server Pages Generator [ServerPagesGenerator]	na podstawie strony XSP
Velocity Generator [VelocityGenerator]	parsując wynik uzyskany z motoru Velocity
Request Generator [RequestGenerator]	używając bieżącego żądania jako źródła danych
Status Generator [StatusGenerator]	tworząc dane xml w oparciu o aktualny stan (status) Cocoona
Stream Generator [StreamGenerator]	na podstawie dokumentu XML będącego żądaniem POST protokołu HTTP
Profile Generator [ProfilerGenerator]	zawierające statystyki związane z działaniem Cocoona
Error Generator [ErrorNotifier] [1]	opisujące przyczynę błędu, który zaistniał podczas przetwarzania żądania w potoku.
Search Generator [SearchGenerator]	wykonując zapytanie do wyszukiwarki (aktualnie Lucene) - uzyskana odpowiedź używana jest do utworzenia dokumentu XML
LinkStatus Generator [LinkStatusGenerator]	tworząc listę dostępnych linków (komponent dostępny dopiero w Cocoon 2.1)
Php Generator [2] [PhpGenerator]	na podstawie wyniku strony PHP, określonej w żądaniu HTTP (strona musi zwracać poprawny dokument XML)
Test Parser Generator [TextParserGenerator]	parsując plik tekstowy (używając gramatyki podanej jako parametr wywołania)

Klasy należą do pakietu org.apache.cocoon.generation

---

[1] należy do pakietu org.apache.cocoon.sitemap
[2] generator opcjonalny, tzn. jego użycie wymaga dodatkowej instalacji

źródło: http://xml.apache.org/cocoon/userdocs/generators/generators.html

Transformatory są istotą potoków. Pobierają zdarzenia SAX (ang. *Simple API for XML*) pochodzące od generatora (zdarzenia te odpowiadają strukturze dokumentu XML), przetwarzają je i przesyłają nowe zdarzenia SAX dalej do potoku. Najpopularniejszym transformatorem jest transformata XSLT, ale jest także wiele innych. Transformator nie koniecznie musi działać tylko na przetwarzanym dokumencie (tak jak XSLT), może także na podstawie otrzymanych zdarzeń SAX

wykonać odpowiednie działania. Przykładem takiego transformatora jest SQLTransformer, który dostając odpowiednie znaczniki XML wykonuje zapytanie do bazy danych i zwraca wyniki jako dokument XML.

Tab. 2. Transformatory dostępne wraz z dystrybucją Cocoona

Nazwa transformatora (nazwa klasy)	**Opis**
XSLT Transformer [TraxTransformer]	Wczytuje podany arkusz XSLT i przetwarza nim dokument XML (zdarzenia SAX) znajdujący się w potoku
Fragment Extractor Transformer [FragmentExtractorTransformer]	Zastępuje wybraną treść linkiem
I18n Transformer [I18nTransformer]	Służy do tłumaczenia słów zgodnie z podanym w XMLu słownikiem, dzięki czemu można wykonać wiele wersji językowych portalu
Log Transformer [LogTransformer]	Służy jako debugger poprzez zapisanie do pliku zdarzeń SAX znajdujących się w strumieniu.
Sql Transformer [SQLTransformer]	Umożliwia wykonywanie zapytań SQL do bazy danych.
Filter Transformer [FilterTransformer]	Wybiera na podstawie wartości podanych parametrów tylko niektóre elementy z potoku XML; przykładowym zastosowaniem jest porcjonowanie wyników uzyskanych przy pomocy SQLTranformatora
Read DOM Session Transformer [ReadDOMSessionTransformer]	Wykonuje konwersję drzewa DOM do potoku zdarzeń SAX
Write DOM Session Transformer [WriteDOMSessionTransformer]	Wykonuje konwersję potoku zdarzeń SAX do drzewa DOM
XInclude Transformer [XIncludeTransformer]	Dołącza zewnętrzne dokumenty lub ich fragmenty korzystając ze specyfikacji Xinclude (www.w3.org/TR/xinclude) i Xpointer (www.w3.org/TR/xptr)
CInclude Transformer [CIncludeTransformer]	Funkcjonalność podobna do xinclude bazująca na specyfikacji Cocoon Include
EncodeURL Transformer [EncodeURLTransformer]	Stosuje metodę response.encodeURL() do URLi; najczęściej używany jest jako ostatni z transformatorów w potoku
Augment Transformer [AugmentTransformer]	Rozbudowują wartości atrybutów href do pełnego URL
XT Transformer [1] [XTTransformer]	Alternatywny transformator XSLT
LDAP Transformer [1] [LDAPTransformer]	Traktuje zawartość potoku jako zapytanie do usługi katalogowej LDAP, a odpowiedź zostaje wstawiona jako wynikowy dokument XML
Text Parser Transformer [TextFragmentParserTransformer]	Parsuje zawartość potoku według gramatyki podanej jako parametr transformatora

Transformatory dostępne wraz z dystrybucją Cocoona – klasy należą do pakietu org.apache.cocoon.transformation

---

[1] generator opcjonalny, tzn. jego użycie wymaga dodatkowej instalacji

źródło: http://xml.apache.org/cocoon/userdocs/transformers/transformers.html

Serializatory zawsze kończą potok rozpoczęty od generatora, zamieniają zdarzenia SAX, na których bazuje przetwarzanie wewnątrz potoku, na strumień binarny lub znakowy. Standardowo w Cocoonie dostępnych jest całkiem spora liczba serializatorów, które umożliwiają uzyskanie takich formatów wyjściowych jak: HTML,

XHTML, XML, zwykły tekst, XLS, PDF, PostScript, PCL, WML, JPEG, PNG, TIFF, SVG, VRML.

Komparatory służą do sterowania przebiegiem obsługi żądania klienta, tzn. decydują czy dany potok przetwarzania jest odpowiedni dla określonego żądania. Wszystkie dopasowania żądania do potoku wpisuje się w sitemapie, przy czym należy pamiętać, że dopasowanie odbywa się od początku sitemapy i wybierany jest pierwszy pasujący wzorzec (ang. *pattern*). Dlatego bardziej szczegółowe dopasowania należy umieszczać przed ogólniejszymi. Dopasowanie może być dokładne (np. być wybierane tylko dla żądania przyklad_html) lub odbywać się według szablonu. Dopuszczalne jest używanie nazw wieloznacznych zawierających jokery (ang. *wildcards*):

‘*’ dopasowuje zero lub więcej znaków aż do znaku ‘/’ lub końca URI

‘**’ dopasowuje zero lub więcej znaków, ale dopasowany ciąg może zawierać znak ‘/’

‘\’ podobnie jak w przypadku wyrażeń regularnych umożliwia dopasowanie określonego znaku, czyli ‘*’ dopasuje znak gwiazdki występujący w żądaniu.

Inną, bardzo użyteczną cechą komparatorów, jest możliwość zapisania dopasowanej części żądania do zmiennej (ang. *tokenization*) i odwołanie się do niej we wszystkich elementach występujących wewnątrz komparatora, np. w generatorach i transformatorach. Przykład wykorzystania podany jest poniżej.

Żądanie przyklad_html zostanie przetworzony transformatą to_html.xsl, a np. żądanie przykład_html_druk transformatą to_html_druk.xsl.

```
<map:pipelines>
   <map:pipeline>
      <map:match pattern="przyklad_*">
         <map:generate src="wez_dane"/>
         <map:transform src="xsl/to_{1}.xsl"/>
         <map:serialize type="html"/>
      </map:match>

            . . .

   </map:pipeline>
</map:pipelines>
```

Przykład dopasowania żądania do potoku w sitemapie

Selektory są dość podobne do komparatorów pod względem celu zastosowania, ale używa się ich nieco inaczej. Wybór, co użyć, należy do programisty i jest zależny od jego poczucia logicznej struktury definiowanej przez siebie mapy serwisu. Podstawowa różnica polega na tym, że komparator zwraca wynik działania fragmentu mapy serwisu, która zostanie wybrana, a selektor zwraca wartość true lub false i na podstawie tej wartości można sterować przepływem potoku.

```
<map:pipelines>
   <map:pipeline>
      <map:match pattern="przyklad">
         <map:select type="parameter">
            <map:parametr name="parameter-selector-test" value="{ROLE}"/>
            <map:when test="1">
              <map:redirect-to uri="zwykly_uzytkownik"/>
            </map:when>
            <map:when test="2">
              <map:transform src="recenzent"/>
            </map:when>
            <map:when test="3">
              <map:transform src="redaktor"/>
            </map:when>
            <map:otherwise>
              <map:transform src="gosc"
            </map:otherwise>
         </map:select>
      </map:match>
   </map:pipeline>
</map:pipelines>
```

Przykład użycia selectora

Akcje są komponentami odpowiedzialnymi za przetwarzanie danych, stanowią logikę aplikacji. Obecnie zaleca się korzystanie z akcji zamiast tworzyć kod w XSP, który to mechanizm jest niezbyt czytelny, trudny w debugowaniu i miesza (przeplata) treść z logiką. Akcja nie wytwarza danych, nie wyświetla ich, nie jest elementem występującym w potoku przetwarzającym dokumenty XML. Natomiast akcje świetnie nadają się do autoryzacji użytkowników w portalu, komunikacji z bazami danych, sprawdzania dostępności wskazanych zasobów, walidacji poprawności danych w wypełnionych przez użytkowników formularzach, mają też dostęp do parametrów żądania i mogą przekazywać pewne informacje do innych komponentów występujących w mapie serwisu. Użycie akcji ma tę przewagę nad XSP, że wprowadza pewien porządek i czytelność w kodzie, co wpływa na łatwiejsze utrzymanie portalu w przyszłości, zmniejszenie liczby błędów popełnianych przez

programistów i łatwy podział zadań pomiędzy osoby odpowiedzialne za poszczególne fragmenty portalu (o czym będzie jeszcze mowa w dalszej części).

Czytniki tworzą strumień wyjściowy z zasobów, takich jak obrazki lub filmy, które nie wymagają przetwarzania, ponieważ są już w żądanym formacie wyjściowym.

Tab. 3. Czytniki dostępne wraz z dystrybucją Cocoona

Nazwa	Opis
Resource	Służy do pobierania danych binarnych potoku przetwarzania
Database	Umożliwia pobieranie danych z baz danych
JSP reader	Pobiera wynik wywołania strony JSP, określonej w parametrach czytnika

źródło: http://xml.apache.org/cocoon/userdocs/readers/readers.html

Cocoon oferuje naprawdę róg obfitości komponentów, których użyteczność docenić można bardzo szybko, jeżeli w ciągu kilku chwil osiąga się cel, którego realizacja w innym środowisku tworzenia serwisów internetowych, opartych na dynamicznie generowanej treści, wymaga znacznie większych nakładów czasu i pracy. Możliwość w dość prosty sposób (zakładając posiadanie umiejętności programowania w Javie) utworzenia własnych komponentów, daje twórcom aplikacji internetowych niespodziewanie duże pole manewru przy relatywnie niskim poziomie trudności wykonania swojego zadania.

Agregacja danych

Agregatory to elementy, które z kilku dokumentów XML tworzą jeden, obejmując całość w nowy znacznik.

```
<map:pipelines>
   <map:pipeline>
      <map:match pattern="calosc">
          <map:aggregate element="caly_dokument">
               <map:part src="czesc1"/>
               <map:part src="czesc2"/>
          </map:aggregate>
          <map:serialize type="xml"/>
      </map:match>
   </map:pipeline>
</map:pipelines>
```

Przykład agregacji

W prezentowanym przykładzie wynik potoku realizującego czesc1 oraz czesc2 zostanie umieszczony wewnątrz znacznika caly_dokument. Jednym z wielu możliwych zastosowań tego mechanizmu jest łączenie danych składających się na interfejs wyświetlanej strony WWW oraz danych stanowiących treść tej strony.

Bardziej zaawansowane możliwości łączenia danych daje transformator Xinclude oraz Cinclude.

SoC, czyli Separation of Concerns

U podstaw tworzenia Cocoona leży kilka fundamentalnych założeń. Najważniejszym z nich jest rozdzielenie treści, prezentacji i logiki. Wsparcie realizacji tego zadania w Cocoonie jest znaczące. Jest to bardzo ważne, gdyż tworząc średni lub nawet duży portal można prace podzielić nie tylko na niezależne projekty, ale również wydzielić zakresy odpowiedzialności poszczególnych osób. Zalety takiej separacji odczuwalne są nie tylko na etapie tworzenia portalu, ale także w czasie jego eksploatacji, konserwacji i modyfikacji. Zmiana wyglądu portalu nie pociąga za sobą konieczności edycji kodu odpowiedzialnego za przetwarzanie danych czy wręcz

samych danych. Zdecydowanie utrzymanie (ang. *maintenance*) serwisu jest prostsze przy jasnym podziale zakresu praw i obowiązków, a co więcej, zmniejsza się prawdopodobieństwo, że zmiana jednej rzeczy wprowadzi przypadkowo błąd bądź niepożądany efekt uboczny. Dodatkowo możliwe jest wykonywanie wielu prac równocześnie, gdyż są w pewnym sensie niezależne od siebie.

Prostota użycia

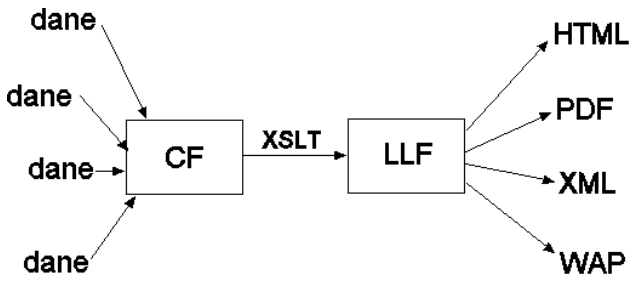
Czy Apache Cocoon jest prosty w użyciu? Właściwie tak, jeżeli posiada się znajomość XML-a oraz umiejętność programowania w Javie. Najtrudniejszym zadaniem wydaje się być konfiguracja Cocoona polegająca na edycji kilku, dość dużych, plików XML. Napisanie sitemapy jest już zadaniem dużo prostszym. Dla początkujących osób nieco odstraszająca może być rozbudowana, trochę skomplikowana i nieczytelna przy pierwszym czytaniu struktura dokumentacji Cocoona (która notabene wykonana jest pod Apache Cocoon). Całkiem pomocne są grupy dyskusyjne, wśród których można znaleźć wiele odpowiedzi na pytania dręczące zarówno początkującego jak i zaawansowanego użytkownika Cocoona. Apache Cocoon jest projektem typu OpenSource, w związku z czym dostępne są jego kody źródłowe, a ich analiza jest także źródłem wielu przydatnych informacji. Tworzenie nieco bardziej zaawansowanego serwisu wymagać zapewne będzie napisania własnych komponentów, a przede wszystkim akcji. Po zapoznaniu się ze sposobem tworzenia akcji nie jest to jednak zadanie bardzo skomplikowane.

Wrażenia z e-Informatyki.pl

Prawie rok temu powstał pomysł stworzenia nowego serwisu internetowego, który stanowiłby centrum publikacyjne i pełnił funkcje elektronicznego czasopisma. Projekt realizowała grupa ponad 50 studentów Wydziału Informatyki i Zarządzania Politechniki Wrocławskiej. Praca została podzielona na podprojekty, które zostały przydzielone poszczególnym zespołom. Nad całością czuwała specjalna grupa (ang. *coreteam*), z której każdy członek wchodził w skład zespołu realizującego wybrany podprojekt. Z podziałem całości na podzadania wiązało się odpowiednie zaplanowanie struktury serwisu, a co za tym idzie, struktura mapy serwisu.

Większość osób było w stanie bardzo szybko zapoznać się z Cocoonem, dzięki czemu praca nad podprojektami rozpoczęła się w stosunkowo krótkim czasie. Nie było też problemów ze zrozumiem koncepcji tworzenia serwisów internetowym pod Cocoonem. Doświadczenie to pozwoliło wyciągnąć wniosek, że założenia przyjęte przez twórców Cocoona sprawdzają się w praktyce.

Potoki przetwarzania zorganizowano zgodnie z pomysłem przedstawionym na Rys. 2. Dane pobierane z kilku źródeł ostatecznie zbierane są w dokumencie XML o określonej przez zespół projektowy strukturze, nazwanej CF (ang. *content format*). Następnie przy pomocy transformatora dodawane są pewne elementy,



Rys. 2. Organizacja potoków przetwarzania w e-Informatyce.pl

związane z zawartością ekranu bądź strony, ale nie mówiące nic o formacie prezentacji. W wyniku otrzymujemy dokument w formacie LLF (ang. *logical layout format*). Zawiera on informację o tym, co ma być zaprezentowane użytkownikowi, czyli oprócz samych danych pochodzących z pliku CF, mogą to być pewne o informacje o paskach menu, o dodatkowych kolumnach z treścią, odsyłacze do innych stron itd. W ostatnim kroku następuje wygenerowanie ostatecznego formatu prezentacji danych.

W trakcie prac ujawniły się także pewne słabości niektórych elementów należących do Cocoona. Wymienić tu można np. transformator SQLTranformer. Przy odpowiednio zaprojektowanej bazie danych i stosunkowo prostych zapytaniach jest to rzeczywiście bardzo wygodny komponent umożliwiający pobieranie danych z baz danych. Jednak przy bardzo skomplikowanych zapytaniach, w których jedno z nich korzysta z wyników zwracanych przez kilka innych zapytań SQL, generowany dokument XML jest nieefektywny i zawiera znaczną redundancję danych.

Ocena końcowa

Apache Cocoon jest bardzo atrakcyjną propozycją dla zespołów, które zamierzają wykonać portal internetowy, zawierający mechanizmy autoryzacji, kilka wersji językowych czy też różnych formatów prezentowania danych na potencjalnie różne urządzenia po stronie klienta. Oparty jest na wielu ciekawych i nowatorskich koncepcjach, oferując w ten sposób duże możliwości, przy umiarkowanym nakładzie pracy i zachowaniu bardzo przejrzystej struktury portalu.

Już sam pomysł potoków przetwarzania, możliwość wykonania wielu transformacji pod rząd zdecydowanie wpływa na ułatwienie osiągnięcia celu, jakim jest utworzenie portalu, a możliwość dołączenia własnych komponentów dodatkowo poszerza i tak już szeroki wachlarz możliwości.

Literatura:

[1] Dokumentacja projektu Apache Cocoon, http://xml.apache.org/cocoon/userdocs/index.html.
[2] Knyziak T., „Apache Cocoon", Telenetforum nr 09/2002, s. 20-22.
[3] Langham M., Ziegler C., Cocoon: Building XML Applications, New Riders 2002.
[4] Madeyski L., "Nowe koncepcje tworzenia aplikacji internetowych na przykładzie portalu E-INFORMATYKA.PL", Materiały konferencji Nowoczesne Technologie Informacyjne w Zarządzaniu 2002 Świeradów Zdrój, Polska, prace naukowe Akademii Ekonomicznej we Wrocławiu Nr 955 s.425-437.

Słowa kluczowe: XML, Apache Cocoon, e-Informatyka