

XII Konferencja PLOUG
Zakopane
Październik 2006

Raportowanie w XML dla zwykłych ludzi – Oracle XML Publisher

Tomasz Traczyk

Politechnika Warszawska
e-mail: ttraczyk@ia.pw.edu.pl

Abstrakt

Od dawna wiadomo, że XML może świetnie służyć do tworzenia złożonych raportów z danych. Potrzebne do tego technologie i narzędzia, takie jak XSQL, XSL, XSL-FO, istnieją już od kilku lat, ale dla przeciętnego użytkownika są zbyt skomplikowane. Zaproponowane przez Oracle narzędzie XML Publisher zdaje się zmieniać ten stan rzeczy: wykorzystuje siłę i elastyczność technologii XML-owych, ale jest dostosowane do potrzeb i możliwości „zwykłego człowieka”.

Wprowadzenie

Użytkownicy systemów informacyjnych od lat korzystają z różnorodnych narzędzi do tworzenia raportów z danych. Moduły raportowania – tworzące zestawienia danych w formie w zasadzie przeznaczonej do druku (choć niekoniecznie rzeczywiście drukowanej) – stanowią ważny składnik każdego systemu informacyjnego. Bez raportów nie potrafi się obejść niemal żaden system.

Raporty mogą mieć różnoraki charakter, mogą to być np. wydruki odpowiadające dokumentom finansowym i innym drukom urzędowym, bilanse, zestawienia okresowe, zestawienia analityczne. W skład większości systemów informacyjnych wchodzi gotowe moduły raportów; praktyka pokazuje jednak, że zwykle niezbędna jest także możliwość tworzenia przez samych użytkowników raportów *ad hoc*. Raportowanie jest w zasadzie działaniem typu *off-line*: tworzenie złożonych raportów jest często czaso- i zasobochłonne, powinno więc móc być wykonywane wsadowo.

„Klasyczne” narzędzia do raportowania (np. Oracle Reports) świetnie nadają się do tworzenia dokumentów finansowych, bilansów czy zestawień. Ich działanie zwykle w jakimś stopniu opiera się na koncepcji grup łamiących (*break groups*), czyli podziale wierszy wyniku zapytania na grupy o jednakowych wartościach w wyróżnionych kolumnach i odpowiedniej obsłudze takich grup: wyliczaniu podsumowań, opatrzeniu grup nagłówkami i stopkami itp. Choć użycie tych narzędzi nie jest zwykle zbyt trudne, są one jednak przeznaczone dla osób z odpowiednim przygotowaniem; końcowy użytkownik systemu informacyjnego raczej z tych narzędzi nie korzysta.

Choć te „klasyczne” narzędzia umożliwiają tworzenie prostych zestawień analitycznych, np. tabel z kilkupoziomowymi podsumowaniami lub zestawień krzyżowych (raportów macierzowych), to do bardziej złożonych analiz, zwłaszcza typu analizy wielowymiarowej, nie są wygodne. Dlatego stworzono specjalne narzędzia, wyspecjalizowane do analizy danych (np. Oracle Discoverer); opierają się one najczęściej na koncepcji tabeli przestawnej (*pivot table*). Ponieważ analizy wielowymiarowe często prowadzi się interaktywnie (OLAP), narzędzia te zwykle są do tego dostosowane, są też na tyle łatwe w użyciu, by mogli z nich korzystać użytkownicy nie mający przygotowania informatycznego. Choć narzędzia te działają interaktywnie, niemniej jednak – ze względu na to że jedynie odczytują one dane oraz że tworzą wyniki sformatowane odpowiednio do wydruku – można je uznać za rodzaj narzędzi do raportowania.

Wszystkie te narzędzia nadają się dobrze do tworzenia pojedynczych dokumentów czy zestawień danych. Gdy jednak trzeba przygotować większe opracowanie, zawierające wiele różnych zestawień, połączonych ze zwykłym tekstem, grafiką itp., trzeba posłużyć się edytorem tekstu i wklejeniem zestawień, co nie jest wygodne i sprzyja powstawaniu błędów.

Raportowanie w XML

XML jest uniwersalnym formatem, który pozwala zapisać w przejrzysty i elastyczny sposób wszelkie dane dające się wyrazić tekstowo. W szczególności zapis w XML danych z baz relacyjnych nie nastęrcza trudności.

Jednocześnie XML nadaje się świetnie do zapisu informacji semistrukturalnej (o strukturze nie do końca określonej, zmiennej, częściowo nieistotnej itp.). Nie sprawia także większych problemów łączenie informacji semistrukturalnej, np. „wolnego” tekstu z elementami formatowania (akapity, sekcje, rozdziały itp.), z danymi w pełni ustrukturyzowanymi (np. z baz relacyjnych). Dlatego XML wydaje się idealnym środkiem do tworzenia złożonych opracowań, zawierających liczne zestawienia danych przeplecione tekstem.

Jeśli źródłem danych nie są jedynie bazy danych, lecz chcemy skorzystać ze źródeł o innym charakterze, to XML wydaje się idealnym środkiem do integracji informacji (patrz [Tra05]), jest

bowiem elastyczny i większość danych dość łatwo daje się na XML przekształcić; wiele źródeł (np. *Web Services*) od razu udostępnia dane właśnie w XML.

Wydaje się więc, że jak najbardziej wskazane jest użycie XML do tworzenia raportów i bardziej złożonych opracowań zawierających dane pochodzące z baz danych i innych systemów informacyjnych. Pozostaje jednak kwestia odpowiedniego sformatowania wyniku raportowania. Oprócz – co oczywiste – przejrzystej i estetycznej formy tego wyniku, oczekujemy współcześnie także możliwości wyprowadzania wyniku w różnych postaciach, np. w formatach HTML i PDF, dostępu do wyniku przez WWW i dostępu do „surowych” danych bez formatowania, w celu ich dalszego przetwarzania. Pożądana jest także możliwość przynajmniej częściowej parametryzacji wyniku, np. prezentacji wyników na kilka różnych sposobów (wg różnych „szablonów”), czy zmiany języka opisów.

Od dawna istnieją związane z XML technologie, które pozwalają przekształcić XML-ową reprezentację informacji w odpowiednią końcową formę raportu:

1. Język XSLT [XSLT06] pozwala transformować wejściowy format danych w XML w format wyjściowy, odpowiadający strukturze tworzonego raportu i odpowiedniej formie prezentacji. Formatem wyjściowym może być język HTML (XHTML) dla prezentacji w WWW, obiekty formatujące XSL-FO dla dokumentów przeznaczonych do druku, zaś język SVG [SVG03] dla grafiki.
2. Obiekty formatujące XSL-FO [XSL01], uzupełnione o SVG, mogą być użyte do tworzenia dokumentów przeznaczonych do druku, z jakością profesjonalnego składu; formatem wyjściowym może być np. PDF.

Nie nastęrcza trudności pozyskanie informacji z baz danych w formacie XML; w przypadku baz Oracle służyć do tego mogą różnorodne środki wbudowane w bazę danych, wchodzące w skład XML DB, np. zapytania SQLX czy pakiety wbudowane generujące dane w formacie XML; podobnemu celowi służy także biblioteka XSU, wchodząca w skład XML Developer's Kit, oraz serwlet XSQL, udostępniający dane z bazy w XML przez protokół HTTP.

Połączenie tych technologii (patrz [Tra02]) daje możliwości budowania złożonych, dopracowanych wizualnie i elastycznych raportów, łączących dane z wielu źródeł z tekstem i grafiką. Mimo istnienia licznych dojrzałych i zaawansowanych technologii oraz narzędzi, raportowanie z użyciem XML nie stało się szczególnie popularne. Przyczyną jest stosunkowo duża złożoność tych technologii i znaczne kwalifikacje niezbędne do ich skutecznego użycia. By bowiem zbudować raport, trzeba znać którąś z technologii pozyskiwania danych w XML z bazy danych, język XSLT, obiekty formatujące XSL-FO, język HTML i – jeśli chcemy tworzyć grafikę – język SVG. Trzeba też opanować posługiwanie się narzędziami implementującymi te technologie i zarządzanie serwerem aplikacyjnym, który umożliwi integrację działania narzędzi i odpowiednią dystrybucję wyników. Takie raportowanie w XML, choć potencjalnie daje bardzo duże możliwości, z całą pewnością nie jest zajęciem dla „zwykłych ludzi”.

Czym jest XML Publisher?

XML Publisher (zwany także Oracle BI Publisher, a w skrócie XMLP) jest opartym na technologiach XML-owych narzędziem firmy Oracle, przeznaczonym do tworzenia raportów. Pierwotnie produkt ten był oferowany jako część Oracle E-Business Suite, został też włączony do systemów Peoplesoft i JD Edwards; ostatnio zaś ukazał się także jako niezależne narzędzie do tworzenia raportów w środowisku WWW, adresowane – jak się wydaje – do użytkowników korporacyjnych.

Narzędzie bazuje na wspomnianych wyżej technologiach XSLT i XSL-FO, ale obudowanych przyjaznym dla użytkownika środowiskiem. Choć zaawansowany użytkownik może korzystać wprost z wielu możliwości tych technologii, dla „zwykłego” użytkownika pozostają one ukryte.

Co więcej, implementacje ww. technologii, choć nie spełniające w 100% standardów W3C, są wyspecjalizowane do celów raportowania i zapewniają wysoką wydajność i niezawodność.

Po co XML Publisher?

XML Publisher pozwala tworzyć zaawansowane raporty z użyciem XML jako reprezentacji danych, pozostając narzędziem, którego mogą – w dość znacznym zakresie – używać osoby bez przygotowania informatycznego.

Podstawową ideą działania jest tu separacja trzech „warstw”: danych, układu graficznego i języka narodowego.

- Dane są pozyskiwane bezpośrednio w XML lub transformowane na XML, np. z wyniku zapytania SQL; ta sama reprezentacja danych może służyć do stworzenia raportów o różnych układach i w różnych językach narodowych.
- Układ graficzny jest definiowany przez stworzenie szablonu określającego wygląd, zawierającego znaczniki powiązane z elementami XML-owej reprezentacji danych; dla jednej reprezentacji danych można utworzyć wiele różnych układów.
- Stałe elementy tekstowe mogą być automatycznie wyekstrahowane z definicji raportu, zapisane w standardowym formacie i przedłożone do tłumaczenia na inne języki. Dla każdego ze zdefiniowanych układów można potem wykonać raport w każdym z języków, na które dokonano tłumaczenia.

Oznacza to, że z tej samej reprezentacji danych można uzyskać raporty o wielu różnych układach i w wielu językach. Dobrze odpowiada to potrzebom przedsiębiorstw, gdzie z reguły na podstawie tych samych (lub bardzo zbliżonych) zestawów danych tworzy się różne dokumenty, a konieczność udostępniania dokumentów w kilku językach staje się powszechna.

Innym istotnym pomysłem jest możliwość tworzenia definicji układów raportu za pomocą typowych programów biurowych Microsoft Word i Adobe Acrobat.

- Tworzenie i modyfikowanie definicji układów za pomocą MS Word jest stosunkowo proste i nie wymaga kwalifikacji informatycznych, zwłaszcza gdy używa się dostarczanego dodatku (*add-in*) Template Builder for Word. W szczególności możliwe jest wykorzystanie przygotowanego wcześniej wzorcowego dokumentu w Wordzie i wstawienie do niego znaczników, które spowodują, że ów wzorcowy dokument stanie się szablonem definiującym układ raportu.
- Podobnie wykorzystać można wzorce dokumentów przygotowane w formacie PDF. Za pomocą narzędzia Acrobat można je opatrzyć znacznikami, które przekształcą wzorcowe dokumenty w szablony definiujące układy raportów.
- Raz stworzone XML-owe reprezentacje danych mogą być wykorzystane jako podstawa wielu różnych wydruków, różniących się układem i częściowo zawartością.

Z tego wynika, że gdy informatyk przygotowuje źródło danych w XML, użytkownicy mogą łatwo sami tworzyć raporty lub dostosowywać wygląd raportów do swoich potrzeb; współpraca między użytkownikami a informatykami znacznie się tu upraszcza. Dodać należy, że samo definiowanie źródeł danych też nie jest trudne i nieco bardziej wykwalifikowani użytkownicy powinni sobie z tym zadaniem poradzić.

Dodatkowo XML Publisher pozwala dokonywać interaktywnych analiz danych zawartych w raporcie za pomocą tabeli przestawnej (*pivot table*); analizy te można prowadzić bezpośrednio przez WWW lub też – korzystając z odpowiedniego dodatku (*add-in*) – z użyciem programu Microsoft Excel.

Z punktu widzenia „zwykłego człowieka” XML Publisher jest zatem narzędziem pozwalającym wykorzystać siłę technologii XML-owych bez konieczności nauki tych technologii, za to w środowisku, do którego użytkownik jest przyzwyczajony: przeglądarki WWW, edytora tekstu i arkusza kalkulacyjnego. Pozwalając z tego samego zestawu danych uzyskiwać różne raporty w wielu językach, narzędzie pozwala zaoszczędzić mnóstwo pracy oraz ustrzec się błędów i niezgodności.

Jak to działa?

Podstawową częścią samodzielnej (tzn. niezwiązanej z E-Business Suite) wersji XMLP jest serwer zwany XML Publisher Enterprise. Jest to aplikacja w technologii J2EE, napisana w języku Java, która może działać na dowolnym serwerze aplikacyjnym zgodnym z J2EE, np. OC4J lub Tomcat. Aplikacja ta pełni rolę motoru raportowania, a także udostępnia „portal raportowy” służący do definiowania raportów oraz zarządzania nimi.

Do wspomaganie tworzenia szablonów raportów za pomocą MS Word oraz prowadzenia analiz danych programem MS Excel służą dodatki (*add-ins*) zebrane w pakiecie XML Publisher Desktop.

Składniki raportu

Definicja raportu XMLP składa się z kilku specyfikacji:

- modelu danych (*data model*); zawiera on definicje zestawów danych (*data sets*), które określają z jakich źródeł i w jaki sposób mają być pobrane dane;
- definicji parametrów raportu, uzupełnionych ewentualnie listami wartości dopuszczalnych dla tych parametrów;
- szablonów układu (*layout templates*) raportu; każdy raport może mieć wiele szablonów;
- opcjonalnie specyfikacji wersji językowych raportu, zawierających tłumaczenia napisów.

Jak buduje się raport?

Raport definiuje się za pomocą interfejsu WWW serwera XML Publisher Enterprise. Tworząc nowy raport umieszcza się go w jednym z folderów.

Następnie definiuje się model danych. Jeśli raport ma opierać się na zapytaniu SQL, to uprzednio trzeba zdefiniować źródło danych (*data source*), które określa sposób połączenia z bazą danych oraz nazwę i hasło użytkownika w tej bazie. Zapytanie SQL można wpisać tekstowo, można też użyć do jego tworzenia wbudowanego konstruktora zapytań typu QBE (*Query By Example*). W wyniku zastosowania modelu danych powstaje XML-owa reprezentacja danych wejściowych; można ją wyeksportować do pliku XML.

Kolejnym krokiem jest zdefiniowanie jednego lub kilku szablonów układu raportu. Szablon zwykle tworzy się za pomocą programu MS Word; można tu skorzystać z dodatku Template Builder for Word – przydatny wówczas będzie wyeksportowany plik z danymi wejściowymi w postaci XML. Utworzony szablon zapisuje się w pliku RTF i łąduje się do repozytorium serwera XMLP Enterprise.

Jeśli raport ma mieć wiele wersji językowych, należy je przygotować, eksportując za pomocą XMLP napisy z definicji raportu i z szablonów do plików w standardzie XLIFF (*XML Localization Interchange File Format*) [XLIFF03]. W tych plikach należy umieścić tłumaczenia na odpowiednie języki i załadować je do repozytorium XMLP. Można także utworzyć osobne wersje szablonów, przeznaczone dla poszczególnych języków.

Tak przygotowany raport jest gotowy do użycia; wywołuje się go przez interfejs WWW aplikacji XMLP Enterprise. Na ekranie startowym raportu można wybrać jeden z szablonów oraz format wyniku, wpisuje się tu też wartości parametrów, jeśli raport je ma. Wersja językowa raportu zostaje wybrana automatycznie na podstawie języka określonego w preferencjach użytkownika. Do wyboru jest kilka formatów wyniku, w tym HTML, PDF, RTF i XML (same dane). Można też wywołać narzędzie do interaktywnej analizy danych z raportu lub przejść do programu MS Excel w celu przeprowadzenia takiej analizy.

Przetwarzanie danych w XMLP

Przetwarzanie danych przez XML Publisher jest właściwie podobne do tego, które należałoby zastosować tworząc samodzielnie raport technologiami XML-owymi; użytkownik jednak jest zwolniony z potrzeby biegłego posługiwania się tymi technologiami, a XMLP dostarcza gotowego środowiska implementującego i integrującego niezbędne technologie oraz pozwalającego zarządzać raportami, uprawnieniami itp.

Dane są więc pobierane ze źródeł danych, łączone i konwertowane na format XML. Załadowane na etapie definiowania raportu szablony są przez XMLP przekształcane na skrypty XSLT. Skrypt, odpowiadający wybranemu szablonowi raportu, jest stosowany do transformacji wejściowej XML-owej reprezentacji danych na reprezentację w postaci obiektów formatujących XSL-FO. Następnie dokument jest przekształcany przez procesor XSL-FO, który tworzy postać końcową.

Należy zwrócić uwagę, iż choć do przygotowania szablonów używa się oprogramowania firmy Microsoft lub Adobe, to w czasie wykonania raportów nie występuje uzależnienie od wymagających opłat produktów tych firm.

Możliwości narzędzia

Modele danych

Każdy raport zawiera definicję modelu danych, składającą się z jednego lub kilku tzw. zestawów danych (*data sets*).

Źródła i zestawy danych

Zestaw danych definiuje sposób pobierania danych z jednego źródła. XML Publisher może korzystać z danych o różnym pochodzeniu.

- Dane relacyjne są pobierane przez zapytania SQL, wykonywane przez interfejs JDBC; można więc korzystać z praktycznie wszystkich baz relacyjnych. Parametry połączenia można zdefiniować wprost, można także wyszukać je za pośrednictwem JNDI (*Java Naming and Directory Interface*).

Źródła danych (*data sources*) JDBC i JNDI definiuje się za pomocą interfejsu WWW XMLP i ze źródeł tych korzysta się tworząc modele danych SQL.

- Dane dostępne od razu w XML można pobierać wprost z ich lokalizacji w sieci przez protokół HTTP. Oczywiście dane te nie muszą być statyczne – adres URL może odnosić się do usługi dynamicznie kreującej XML, takiej jak XSQL. Żądanie HTTP może odbywać się metodą GET lub POST i można przekazać w nim parametry.
- Dane można pobierać także z usług *Web Services*, ale tylko z takich, które zwracają jako wynik dobrze sformułowany XML. Podaje się adres URL dokumentu WSDL opisującego usługę i nazwę metody do wywołania oraz ewentualnie jej parametry.

Każde z tych źródeł może być buforowane. Buforowanie pozwala wykonywać raport z różnymi szablonami układu bez konieczności ponawiania pobierania danych z ich źródła; oznacza to oczywiście odciążenie bazy danych.

Wynikiem działania zestawu danych jest dokument XML zawierający pobrane dane; struktura tego dokumentu zależy od rodzaju źródła. W przypadku źródła SQL-owego jest to tzw. postać kanoniczna (ROWSET – ROW – elementy odpowiadające kolumnom), znana z XSL czy XSQL. Jeśli w raporcie zdefiniowano kilka zestawów danych, to wynikowa XML-owa reprezentacja danych może łączyć w sobie dane uzyskane ze wszystkich tych zestawów. Dzięki temu można tworzyć rozbudowane opracowania, łączące w jednym, w pełni automatycznie składanym dokumencie wiele różnorodnych zestawień i wykresów.

Parametry

Raporty mogą mieć parametry; ich wartości przekazuje się do raportu w wywołaniu przez URL lub wypełniając pola na startowej stronie WWW raportu. Parametry mogą mieć określone listy dopuszczalnych wartości; listy te mogą być stałe lub tworzone dynamicznie przez wykonanie zapytania SQL. Wyniki zapytań określających możliwe wartości kolejnych parametrów mogą być uzależnione od wartości już wybranych we wcześniejszych parametrach.

Parametry raportu mogą być wykorzystane jako zmienne związane (*bind variables*) w zapytaniach SQL, parametry żądania HTTP przy odwołaniu do XML-owego źródła danych oraz parametry usługi przy odwołaniu do źródła typu *Web Service*. Do parametrów można się także odwoływać w szablonach układu.

Data Templates

Bardziej złożone zestawy danych można definiować używając tzw. *Data Templates* – specyfikacji zapisanych w XML (patrz przykład w tabeli 3). Określają one pochodzenie danych, definiując zapytania SQL. Między wynikami zapytań można ustanowić związki referencyjne przez użycie zmiennych związanych lub specjalnego znacznika; można też wyspecyfikować parametry i użyć ich w zapytaniach.

Data Templates określają też budowę wynikowej struktury XML-owej, definiując zagnieżdżone grupy danych. Grupy zawierają definicje elementów wynikowej struktury XML; można tu także tworzyć elementy wyliczające agregacje (podsumowania itp.) dla grup.

Używając *Data Templates* można odwoływać się do funkcji PL/SQL: filtrów dla grup danych oraz wyzwalaczy wykonywanych przed raportem lub po nim.

Przykłady

Przykłady zbudowano w oparciu o relacyjną strukturę danych zawierającą tabele PRZEDMIOTY i KLASY_TEMATYCZNE, połączone związkiem *n-1*, zawierające informacje o przedmiotach prowadzonych na pewnej uczelni i ich klasyfikacji. Opisowe informacje o klasach (konspekty) umieszczono w pliku XML.

Przykład pokazany w tabeli 1 prezentuje model danych ze „zwykłym” zapytaniem SQL, wykonującym złączenie tabel; wynik jest tu zatem „płaski”, bez zachowania hierarchii. Dodatkowo pokazano użycie parametru raportu w warunku zapytania.

Tabela 1. Model danych ze zwykłym zapytaniem SQL i parametrem

Zestaw danych	Zapytanie / źródło
Zapytanie SQL	<pre>select id_klasy, k.nazwa as klasa, id_przedmiotu, p.nazwa as przedmiot, liczba_godzin from klasy_tematyczne k join przedmioty p using (id_klasy) where k.nazwa like '%' :parametr '%'</pre>
Wynikowa struktura XML	
<pre><ROWSET> <ROW> <ID_KLASY>TEOR</ID_KLASY> <KLASA>Przedmioty teoretyczne</KLASA> <ID_PRZEDMIOTU>OTW</ID_PRZEDMIOTU> <PRZEDMIOT>Ogólna teoria wszystkiego</PRZEDMIOT> <LICZBA_GODZIN>10</LICZBA_GODZIN> </ROW> ... </ROWSET></pre>	

Przykład pokazany w tabeli 2 przedstawia model danych z zapytaniem SQL zawierającym operator CURSOR, wynik zachowuje tu więc zależność nadrzędny-podrzędny. Pokazano też łączne użycie dwóch źródeł danych: zapytania SQL i dokumentu XML.

Tabela 2. Model danych zawierający zapytanie SQL z operatorem CURSOR i źródło XML-owe

Zestaw danych	Zapytanie / źródło
Zapytanie SQL	<pre>select id_klasy, k.nazwa as klasa, cursor (select id_przedmiotu, nazwa as przedmiot, liczba_godzin from przedmioty p where p.id_klasy=k.id_klasy) as przedmioty from klasy_tematyczne k</pre>
Dane XML	Dokument XML zawierający konspekty
Wynikowa struktura XML	
<pre><DATA> <przedmioty> <ROW> <ID_KLASY>TEOR</ID_KLASY> <KLASA>Przedmioty teoretyczne</KLASA> <PRZEDMIOTY> <PRZEDMIOTY_ROW> <ID_PRZEDMIOTU>OTW</ID_PRZEDMIOTU> <PRZEDMIOT>Ogólna teoria wszystkiego</PRZEDMIOT> <LICZBA_GODZIN>10</LICZBA_GODZIN> </PRZEDMIOTY_ROW> ... </PRZEDMIOTY> </ROW> ... </przedmioty> <konspekty> <KONSPEKTY_KLAS> <KONSPEKT ID_KLASY="TEOR"> <TEKST> Klasa obejmuje ogólną teorię wszystkiego i wnioski z niej płynące. </TEKST> </KONSPEKT> ... </ KONSPEKTY_KLAS> </konspekty> </DATA></pre>	

Przykład w tabeli 3 pokazuje użycie *Data Template*; łączone są tu wyniki dwóch zapytań (z zachowaniem zależności referencyjnej), a wynik zachowuje hierarchię danych. Pokazano też użycie parametru raportu oraz tworzenie agregacji (sumy) na poziomie grupy.

Tabela 3. Model danych z *Data Template*

Zestaw danych	Zapytanie / źródło
<i>Data Template</i>	<pre> <dataTemplate name="przedmioty" description="Klasy i przedmioty" Version="1.0"> <parameters> <parameter name="parametr" dataType="character" /> </parameters> <dataQuery> <sqlStatement name="klasy"> <![CDATA[SELECT id_klasy as klasa, nazwa FROM klasy_tematyczne WHERE id_klasy = nvl(:parametr, id_klasy)]]> </sqlStatement> <sqlStatement name="przedmioty"> <![CDATA[SELECT id_przedmiotu, nazwa, liczba_godzin FROM przedmioty WHERE id_klasy = :klasa]]> </sqlStatement> </dataQuery> <dataStructure> <group name="klasa" source="klasy"> <element name="id" value="KLASA" /> <element name="nazwa" value="NAZWA" /> <group name="przedmiot" source="przedmioty"> <element name="id" value="ID_PRZEDMIOTU"/> <element name="nazwa" value="NAZWA" /> <element name="godziny" value="LICZBA_GODZIN"/> </group> <element name="suma_godziny" value="przedmiot.godziny" function="SUM()" /> </group> </dataStructure> </dataTemplate> </pre>
Wynikowa struktura XML	
<pre> <przedmioty> <parametr>TEOR</parametr> <LIST_KLASA> <KLASA> <ID>TEOR</ID> <NAZWA>Przedmioty teoretyczne</NAZWA> <LIST_PRZEDMIOT> <PRZEDMIOT> <ID>OTW</ID> <NAZWA>Ogólna teoria wszystkiego</NAZWA> <GODZINY>10</GODZINY> </PRZEDMIOT> ... </LIST_PRZEDMIOT> <SUMA_GODZIN>15</SUMA_GODZIN> </KLASA> </LIST_KLASA> </przedmioty> </pre>	

Formatowanie raportów

Raporty XMLP formatuje się definiując szablony układu w formacie RTF lub PDF.

Definiowanie szablonów RTF

Szablony są plikami w formacie RTF, tworzonymi zwykle za pomocą programu Microsoft Word. Zawierają one stałe elementy formatowania: napisy, tabele, elementy żywej paginy (nagłówki, stopki), paginację itd. Wśród nich umieszcza się znaczniki XMLP pobierające dane, wykonujące obliczenia itp.

By pobrać ciągi danych, w szablonie trzeba zdefiniować odpowiednio zagnieżdżone tzw. grupy powtarzane (*repeating groups*); robi się to za pomocą pary znaczników odpowiadających początkowi i końcowi grupy (patrz przykłady w tabelach 4 i 5). Znaczniki do tego służące odpowiadają wybranym instrukcjom XSLT i, podobnie jak w XSLT, odwołania do danych zapisuje się za pomocą wyrażeń XPath [XPATH06].

Znaczniki można umieszczać w szablonie RTF w dwóch postaciach:

- jako zwykłe napisy ujęte w znaki `<? oraz ?>` (nie ma zgodności z XML!); wewnątrz znajduje się nazwa znacznika, a po dwukropku ewentualne parametry odseparowane cudzysłowami;
- jako pola formularzowe (*form fields*); we własności pola, przeznaczonej normalnie na tekst pomocy, umieszcza się odpowiedni znacznik XMLP.

Znaczniki powodujące wstawianie wartości danych (tzw. *placeholders*) mają nazwy zgodne z nazwami elementów XML-owej reprezentacji danych. Formatowanie – np. czcionka – zastosowane do tych znaczników (lub zawierających je pól) będzie użyte także w wynikowym raporcie.

Szablony RTF – jak już wspomniano – można łatwo tworzyć używając dodatku Template Builder for Word. Wynik działania tego narzędzia można oczywiście „ręcznie” udoskonalać, np. poprawiając formatowanie czy wykorzystując zaawansowane możliwości szablonów.

Przykłady

W pierwszym przykładzie pokazano prosty szablon dla modelu danych z tabeli 1. Ponieważ wynik zapytania relacyjnego jest „płaski”, by dokonać grupowania przedmiotów w ramach klas użyto instrukcji XSLT 2.0 `for-each-group`. Pokazano tu zagnieżdżanie grup danych, sortowanie w każdej z tych grup oraz podsumowanie wyliczane na poziomie grupy.

Tabela 4. Szablon i wynik raportowania dla danych z tabeli 1

<?for-each-group:ROW;./ID_KLASY?><?sort:ID_KLASY?> Klasa: <?ID_KLASY?> Nazwa klasy: <?KLASA?>		Id przedmiotu Nazwa przedmiotu Liczba godzin	
<?for-each:current-group()?> <?sort:ID_PRZEDMIOTU?> <?ID_PRZEDMIOTU?> <?PRZEDMIOT?>			<?LICZBA_GODZIN?> <?end for-each?>
RAZEM			<?sum (current-group()/LICZBA_GODZIN)?>
<?end for-each-group?>			
Klasa: PRAK Nazwa klasy: Przedmioty praktyczne		Id przedmiotu Nazwa przedmiotu Liczba godzin	
APOT Aspekty praktyczne ogólnych teorii			3
			RAZEM
			3
Klasa: TEOR Nazwa klasy: Przedmioty teoretyczne		Id przedmiotu Nazwa przedmiotu Liczba godzin	
OTW Ogólna teoria wszystkiego			10
STNR Szczególna teoria niektórych rzeczy			5
			RAZEM
			15

Drugi przykład prezentuje szablon dla modelu danych z tabeli 2. Wynik zapytania relacyjno-obiektowego zawiera od razu odpowiednio zagnieżdżone elementy, tu nie ma więc konieczności wykonywania dodatkowego grupowania. W strukturze raportu, odpowiadającej wynikowi zapytania SQL, umieszczono dane pochodzące z drugiego zestawu danych – dokumentu XML zawierającego konspekty – odpowiednio wybrane za pomocą wyrażenia XPath z warunkiem.

Tabela 5. Szablon i wynik raportowania dla danych z tabeli 2

<?for-each:ROW?><?sort:ID_KLASA?> Klasa: <?ID_KLASA?> Nazwa klasy: <?KLASA?> <?for-each://KONSPEKT[@ID_KLASA=current()/ID_KLASA]?> <?TEKST?> <?end for-each?>		
	Id przedmiotu Nazwa przedmiotu Liczba godzin	
<?for-each:PRZEDMIOTY_ROW?> <?ID_PRZEDMIOTU?> <?PRZEDMIOT?>		<?LICZBA_GODZIN?> <?end for-each?>
RAZEM		<?sum (LICZBA_GODZIN)?>
<?end for-each?>		
Klasa: PRAK Nazwa klasy: Przedmioty praktyczne Klasa obejmuje ogólną teorię wszystkiego i wnioski z niej płynące.		
	Id przedmiotu Nazwa przedmiotu Liczba godzin	
APOT	Aspekty praktyczne ogólnych teorii	3
RAZEM		3
Klasa: TEOR Nazwa klasy: Przedmioty teoretyczne Klasa obejmuje zastosowania ogólnej teorii wszystkiego i teorii pochodnych w praktyce gospodarczej i administracyjnej, ze szczególnym uwzględnieniem problemów integracji europejskiej.		
	Id przedmiotu Nazwa przedmiotu Liczba godzin	
OTW		

Ogólna teoria wszystkiego	10
STNR	
Szczególna teoria niektórych rzeczy	5
	RAZEM
	15

Grafika i wykresy

Wszelkie stałe elementy graficzne (obrazy, wzory, napisy specjalnymi czcionkami, „znaki wodne” itp.), zdefiniowane w szablonie, umieszczane są także w wynikowym raporcie.

Możliwe jest też umieszczanie grafiki sterowane danymi; można w ten sposób sterować położeniem obrazu, jego wielkością, liczbą powtórzeń; można także umieścić na obrazie tekst pobrany z danych.

XML Publisher pozwala także tworzyć różnorodne wykresy. Wykorzystuje do tego narzędzie Oracle BI Beans, które tworzy wykresy na podstawie struktury danych zapisanej w XML [OBIB]. Wykresy takie osadza się w szablonie tworząc obiekt graficzny o odpowiedniej wielkości, a w jego właściwości *Alternate Text* umieszczając instrukcje XSLT, tworzące potrzebny XML na podstawie danych raportu. Proste wykresy można łatwo tworzyć odpowiednim kreatorem wchodzącym w skład narzędzia Template Builder for Word.

Jeśli raport jest prezentowany w HTML, elementy graficzne mogą być generowane w SVG.

Zaawansowane możliwości formatowania

XML Publisher oferuje także liczne zaawansowane możliwości kształtowania raportu.

- Wspierana jest większość możliwości formatowania oferowanych przez MS Word, w tym mechanizmy stronicowania i numeracji stron, justowanie, zagnieżdżone tabele, sekcje wielospaltowe, generowanie spisów treści.
- Za pomocą znaczników XMLP można dodatkowo sterować paginacją, wymuszając zmiany stron w odpowiednich miejscach, np. między grupami danych; można także wpływać na numerację stron. Znaczników XMLP można używać w żywej paginie (nagłówki, stopki), z możliwością zróżnicowania dla różnych części dokumentu oraz dla stron parzystych i nieparzystych.
- Dostępne jest formatowanie warunkowe: część raportu może być prezentowana tylko gdy spełniony jest określony warunek; istnieją konstrukcje sterujące *if*, *if-then-else* oraz *choose*. W podobny sposób można warunkowo sterować formatowaniem fragmentów raportu, np. kolorować liczby przekraczające pewien limit.
- Wykonywać można różnego rodzaju obliczenia, np. podsumowania grup, podsumowania stron, sumy bieżące.
- Można tworzyć tabele z dynamicznie generowanymi kolumnami oraz zestawienia krzyżowe (*cross-tab*).
- Dane numeryczne oraz daty można formatować używając formatów MS Word lub deskryptorów formatu znanych z funkcji konwersji Oracle.

XML Publisher jest dostarczany z zestawem typowych czcionek. Jeśli potrzebne są inne, nietypowe czcionki, to nie wystarczy ich użycie w szablonie; trzeba jeszcze umieścić je na serwerze

oraz zadeklarować w pliku konfiguracyjnym XMLP; serwer akceptuje czcionki TrueType oraz Type1.

Szablony PDF

Szablony układu raportów można także tworzyć w formacie PDF. Za pomocą narzędzia Adobe Acrobat przekształca się dokument PDF w szablon, tworząc na nim pola (*form fields*) odpowiadające elementom danych. Podobnie jak w szablonach RTF, zasięg grup powtarzanych określa się tworząc dodatkowe pola zawierające znaczniki XMLP, oznaczające początek i koniec grupy.

Działający raport można w ten sposób zbudować na podstawie właściwie każdego dokumentu, np. zeskanowanego formularza papierowego.

XSLT, XSL-FO i SQL w szablonach

Znaczniki XMLP odpowiadają wybranym konstrukcjom XSLT. W szablonach można jednak także umieścić wprost znaczniki XSLT oraz XSL-FO (w postaci pól *form fields*). Można tu użyć w zasadzie każdej instrukcji XSLT oraz znacznego podzbioru obiektów formatujących; daje to zaawansowanemu użytkownikowi niemal nieograniczone możliwości.

Dodatkowo stworzono możliwość odwoływania się w szablonach do wyrażeń języka SQL, co ułatwia umieszczanie wyników obliczeń, dokonywanie konwersji itp. Służą do tego specjalne znaczniki `<?xdoxslt:wyrażenieSQL?>`.

Repozytorium XMLP

Pliki definiujące raporty XMLP Enterprise przechowuje w swoim repozytorium. Definicje raportów zapisywane są w postaci plików XML o rozszerzeniu .xdo; szablony układów są zaś przechowywane w swym oryginalnym formacie RTF lub PDF. Repozytorium może być po prostu odpowiednio zorganizowanymi folderami w systemie plików, może też być umieszczone w bazie danych Oracle w XML DB.

Działaniem XMLP steruje także zbiór plików konfiguracyjnych, które definiują źródła danych (bazy relacyjne), użytkowników, role i ich uprawnienia, serwery druku i inne miejsca przeznaczenia raportów, parametry konfiguracyjne serwera (np. wielkości buforów, dostęp do serwera LDAP itp.), własności generowanych plików PDF, czcionki itp.

Interfejsy programistyczne XMLP

Przygotowane raporty można wywoływać interaktywnie, korzystając ze stron WWW serwera XMLP Enterprise; można także wywołać raport wprost, podając jego adres URL i przekazując odpowiednie parametry.

Wiele z usług XMLP można też wywoływać programowo przez interfejsy API (*Application Programming Interfaces*) dla języka Java; w ten sposób udostępniono m.in.:

- *PDF Form Processing Engine* – procesor tworzący raporty na podstawie szablonów PDF i danych w XML;
- *RTF Processor* – procesor przekształcający szablony RTF w skrypty XSL;
- *FO Engine* – procesor tworzący wynikowe dokumenty na podstawie zapisu w XSL-FO;
- *Document Processor Engine* – narzędzie sterujące wsadowym przetwarzaniem i wysyłką raportów;
- *Data Engine* – procesor tworzący XML-owe reprezentacje danych na podstawie specyfikacji *Data Template* lub zapytań SQL; narzędzie może także wygenerować schemat *XML Schema* odpowiadający wynikowemu XML oraz domyślny szablon RTF.

XML Publisher Enterprise jako „portal” raportowy

Serwer Oracle XML Publisher Enterprise zawiera nie tylko motor wykonujący raporty, lecz pełni także rolę „portalu” raportowego, pozwalającego zarządzać całym procesem tworzenia i wykonywania raportów.

Za pomocą tego właśnie „portalu” użytkownik o uprawnieniach projektanta może definiować nowe raporty i modyfikować definicje istniejących, np. dodawać nowe szablony lub wersje językowe. XMLP umożliwia zorganizowanie zbioru zdefiniowanych raportów w strukturę folderów.

Użytkownik końcowy, by wykonać przygotowany raport, musi być do tego uprawniony. System uprawnień oparty jest na rolach, które przypisuje się użytkownikom, a które uprawniają do wykonywania raportów zawartych w określonych folderach. Definicje użytkowników i ról mogą być zapisane we własnym repozytorium XMLP, możliwe jest też korzystanie z serwera LDAP lub autoryzacja z użyciem SSO (*Single Sign On*).

Uprawniony użytkownik korzystając z „portalu” wybiera z odpowiedniego folderu potrzebny raport, podaje wartości parametrów i inicjuje wykonanie raportu. Może także skorzystać z możliwości harmonogramowania raportów, określając kiedy raport ma być wykonany (jednokrotnie lub cyklicznie) i jakie ma być jego przeznaczenie. Gotowy raport może być przekazany do serwera druku, rozesłany pocztą elektroniczną, faksem, udostępniony przez protokół WebDAV lub ftp.

Podsumowanie

XMLP a inne technologie raportowania

Użytkownik tradycyjnego narzędzia do raportowania – jak Oracle Reports – znajdzie w XMLP znane elementy, takie jak definicje modelu danych czy parametrów. Nie zaskoczą go także możliwości zarządzania raportami, ich harmonogramowania czy definiowania uprawnień: w porównaniu z bardzo dojrzałym produktem, jakim jest Oracle Reports Server, XMLP nic nadzwyczajnego tu nie oferuje.

Natomiast istotnym *novum* jest możliwość zdefiniowania raportu z wieloma układami, a wspólnym modelem danych; wydaje się to rozwiązaniem bardzo sensownym i praktycznie przydatnym. Podobnie ułatwienia w tłumaczeniu treści raportów na różne języki i możliwość uruchomienia tego samego raportu w wielu wersjach językowych to bardzo dobre rozwiązania, znacznie lepsze od tworzenia i utrzymywania wielu raportów różniących się tylko językiem.

Użyty w XMLP sposób tworzenia szablonów definiujących układy raportów jest bardzo pomysły, gdyż odwołuje się do narzędzia powszechnie znanego użytkownikom, jakim jest edytor tekstu. Choć co do wyboru akurat formatu RTF można mieć wątpliwości (autor wolałby w tym miejscu raczej XML-owy i otwarty standard OpenDocument), to sam pomysł użycia edytora tekstu jako narzędzia do tworzenia szablonów wydaje się znakomity. W porównaniu z „ramkologią” Oracle Reports jawi się to jako sposób bardzo przejrzysty i przyjazny.

XMLP jest zupełnie bezkonkurencyjny jeśli chodzi o tworzenie rozbudowanych opracowań, łączących w jedną całość wiele różnych raportów z grafiką, „wolnym” tekstem, zaawansowanym formatowaniem, żywą paginą itp. Tradycyjne narzędzia oferowały tu bardzo niewiele, XMLP oferuje zaś właściwie wszystko to, co jest dostępne we współczesnych edytorach tekstu.

Szablony PDF pozwalają z kolei łatwo przekształcać gotowe wzorce formularzy urzędowych w działające raporty. W porównaniu z mozolnym „rzeźbieniem” w tradycyjnym narzędziu, tworzenie takiego raportu jest dużo łatwiejsze, szybsze i – co najważniejsze – zapewnia doskonałą zgodność z wzorcem.

Niewątpliwą zaletą XMLP jest udostępnienie jego funkcjonalności przez interfejsy programistyczne (API). Twórca systemu informacyjnego ma dzięki temu możliwość wkomponowania raportowania za pomocą XMLP w działanie swojego systemu, z zachowaniem pełnej kontroli nad tym procesem.

Do wad XMLP – w porównaniu do Oracle Reports – można zapewne zaliczyć brak integracji z tradycyjnymi narzędziami, jak Oracle Forms, co utrudnia wykorzystanie narzędzia w tradycyjnie zbudowanych aplikacjach, zwłaszcza tych generowanych za pomocą systemu Oracle Designer.

Jak się jednak wydaje, w znakomitej większości zastosowań XML Publisher może zastąpić narzędzia typu Oracle Reports, a do niektórych zastosowań nadaje się znacznie lepiej.

Trudniej porównywać XMLP z narzędziami do raportowania *ad hoc* i analiz interaktywnych, takimi jak Oracle Discoverer, gdyż ich przeznaczenie wyraźnie się różni. Można jednak pokusić się o stwierdzenie, że w niektórych zastosowaniach, szczególnie tam, gdzie wolumeny analizowanych danych nie są wielkie i nie ma potrzeby stosowania zaawansowanych mechanizmów optymalizujących wydajność, XMLP może zastąpić i takie narzędzia. Wbudowany kreator zapytań typu QBE, choć niedoskonały, umożliwia budowanie zapytań *ad hoc*, a kreator raportów, działający jako dodatek do MS Word, pozwala tworzyć raporty użytkownikom nie mającym kwalifikacji informatycznych. Analizator danych umożliwia wykonywanie analiz wielowymiarowych, zaś wbudowane możliwości graficzne pozwalają wizualizować wyniki analiz.

Można więc przypuszczać, że w pewnych zastosowaniach XML Publisher może zastąpić także narzędzia takie jak Oracle Discoverer.

W porównaniu z metodą budowania raportów przy użyciu technologii XML-owych zestawianych samodzielnie przez projektanta, XMLP nie ma przewagi co do możliwości funkcjonalnych. Oferuje jednak większość funkcji, które mogą być potencjalnie potrzebne, w prostym w obsłudze zintegrowanym środowisku. Według zapewnień producenta osiągnięto też bardzo dobrą wydajność i niezawodność, nawet dla bardzo dużych zadań; jest to zapewne przewaga nad rozwiązaniami budowanymi z komponentów typu *open-source*, nie wszystkie one bowiem mają pod tym względem dobrą opinię. Nie do wzgardzenia jest oczywiście wsparcie techniczne producenta. Jedynie pod względem kosztów oprogramowania „domorośle” rozwiązania mają wyraźną przewagę.

Zalety i wady XMLP

XML Publisher wydaje się mieć szereg ważnych zalet. Kluczową jest oczywiście pomysłowy sposób tworzenia szablonów raportów, umożliwiający wykorzystanie wzorcowych dokumentów w formacie MS Word lub PDF. Nie mniej ważną zaletę stanowi możliwość związania wielu różnych szablonów i wielu wersji językowych z jednym modelem danych. Możliwości kształtowania wyglądu raportu są imponujące; ważna jest zwłaszcza możliwość połączenia wielu różnych danych, pochodzących z różnych źródeł, w jedno rozbudowane i porządnie złożone opracowanie. Możliwość integracji narzędzia z własnymi programami przez interfejsy API docenią programiści tworzący systemy o niebanalnej logice.

Do wad narzędzia zaliczyć można mało wyczerpującą dokumentację. Brakuje wsparcia dla stanowiącego alternatywę wobec RTF standardu OpenDocument. Dokuczliwa jest niemożność korzystania z usług *Web Services* zwracających wyniki skalarne. Nie zachwycają „jednokierunkowe” kreatory (w tym kreator zapytań SQL): umieją one utworzyć od nowa odpowiedni fragment definicji raportu, ale nie można ich użyć do zmiany tej definicji. Zalety nie stanowią też niewątpliwie obecne warunki licencjonowania produktu...

XML Publisher narzędziem dla „zwykłych ludzi”?

Wydaje się jednak, że zalety narzędzia przeważają nad wadami. Niewątpliwie podstawowe koncepcje konstrukcji XMLP są bardzo udane. Technologię XML zastosowano tu w pełni zgodnie z jej przeznaczeniem i „ideologią”, starając się utrzymać też zgodność ze standardami. Narzędzie w swym obecnym kształcie jest już dojrzałe i nadaje się do praktycznego zastosowania.

Przy swych dużych możliwościach XML Publisher pozostaje produktem prostym w obsłudze. Choć informatycy znajdują tu wiele zaawansowanych technik, typowe raporty jest w stanie budować i dostosowywać do swych potrzeb osoba nie mająca wysokich kwalifikacji, wystarczy bowiem zrozumienie kilku podstawowych pomysłów i umiejętność posługiwania się edytorem tekstu. Można zatem śmiało powiedzieć, że narzędzie nadaje się dla „zwykłych ludzi”.

Bibliografia

- [XMLP06] Oracle® XML Publisher Enterprise User's Guide. Oracle Corp., June 2006.
- [Ritt06] Rittman M.: Build an Online Reporting Application Using Oracle XML Publisher. June 2006. <http://otn.oracle.com>
- [XSL01] Extensible Stylesheet Language (XSL) Version 1.0. W3C Recommendation, 2001. <http://www.w3.org/TR/xsl/>
- [XSLT06] XSL Transformations (XSLT) Version 2.0. W3C Candidate Recommendation, 2006. <http://www.w3.org/TR/xslt20>
- [XPath06] XML Path Language (XPath) 2.0. W3C Candidate Recommendation, 2006. <http://www.w3.org/TR/xpath20>
- [SVG03] Scalable Vector Graphics (SVG) 1.1 Specification. W3C Recommendation, 2003. <http://www.w3.org/TR/SVG/>
- [XLIFF04] XLIFF 1.1 Specification. OASIS, 2003 <http://www.oasis-open.org/committees/xliff/documents/xliff-specification.htm>
- [Tra00] Traczyk T.: Język XSL. Materiały VI konferencji użytkowników i deweloperów Oracle. Kościelisko, październik 2000.
- [Tra02] Traczyk T.: Obiekty formatujące w języku XSL. Materiały VIII konferencji użytkowników i deweloperów Oracle. Kościelisko, październik 2002. ISSN 1641-2117.
- [Tra05] Traczyk T.: Język XQuery jako narzędzie do integracji danych – Oracle XML Data Synthesis. Materiały IX konferencji użytkowników i deweloperów Oracle. Kościelisko, październik 2005. ISSN 1641-2117.
- [OBIB] Oracle Business Intelligence Beans 10g <http://www.oracle.com/technology/products/bib/index.html> oraz http://www.oracle.com/technology/products/reports/htdocs/getstart/whitepapers/graphdtd/graph_dtd_technote_2.html