Budowa nowoczesnych aplikacji internetowych z wykorzystaniem języka Java, technologii WebMacro i Topic Maps

Jakub Gałęski, Karol Foremski, Arkadiusz Bigos, Krzysztof Kawa, Zbyszko Królikowski
Instytut Informatyki Politechniki Poznańskiej
Piotrowo 3A, 60-965 Poznań
Zbyszko.Krolikowski@cs.put.poznan.pl

Streszczenie. W pracy zostaną przedstawione zastosowania języka Java i technologii z nim związanych jako uniwersalnych narzędzi do budowania aplikacji internetowych w kontekście klasycznej trójwarstwowej architektury dla tego typu aplikacji. Omówione zostaną następujące technologie: JDBC jako uniwersalne łącze do baz danych, Java Servlets w połączeniu z technologią WebMacro jako warstwa środkowa (tzw. *business application logic*), XML jako forma przechowywania dokumentów w bazie danych oraz TopicMaps jako metoda definiowania powiązań między danymi. W artykule zamieszczona zostanie próba porównania tych technologii z innymi dostępnymi rozwiązaniami. Wymienione technologie omówione zostaną w oparciu o doświadczenia zdobyte przez autorów podczas realizacji projektu portalu internetowego w Niemczech..

1. Wprowadzenie

Sieć Internet wczoraj to sieć projektowana i budowana dla naukowców. Nowe czasy to również nowe wymagania. Spowodowały one, że również Internet musiał się zmienić. Dzisiejszy Internet to sieć zorientowana na zastosowania naukowe, ale także na biznes. Firmy internetowe stymulują obecnie rozwój gospodarczy wielu krajów i należy się z nimi liczyć. Takie upowszechnienie się Internetu wymusiło powstanie wielu nowych technologii, które ułatwiają (przyspieszają) budowę aplikacji internetowych. Powstały także liczne technologie i narzędzia do integracji istniejących już systemów z nowymi. Niniejsza praca jest poświęcona przeglądowi tych technologii. W pracy postaramy się przedstawić znane i powszechnie używane technologie, a także technologie nowatorskie. Dokonamy próby ich porównania, pokażemy ich zalety i wady, szanse rozwoju oraz jak można je wykorzystać przy budowie nowoczesnych aplikacji (portali) internetowych.

2. Przegląd dostępnych dotychczas technologii budowy aplikacji internetowych

Początki Internetu to wielość protokołów (Veronica, Gopher, X.500...). Ich rozwój został w znacznym stopniu zahamowany przez pojawienie się WWW (World Wide Web). Początki WWW to HTML i strony statyczne. Kolejny krok to graficzne przeglądarki WWW (Netscape, Mosaic). Jednak i to szybko stało się nie wystarczające. Naturalnym, a zarazem najprostszym sposobem usunięcia sygnalizowanych niedogodności stało się rozwiązanie polegające na połączeniu technologii baz danych i stron WWW. Informacje są przechowywane w bazie danych systemu informacyjnego, a następnie przez mechanizmy pośredniczące przetwarzane na kod stron języka HTML.

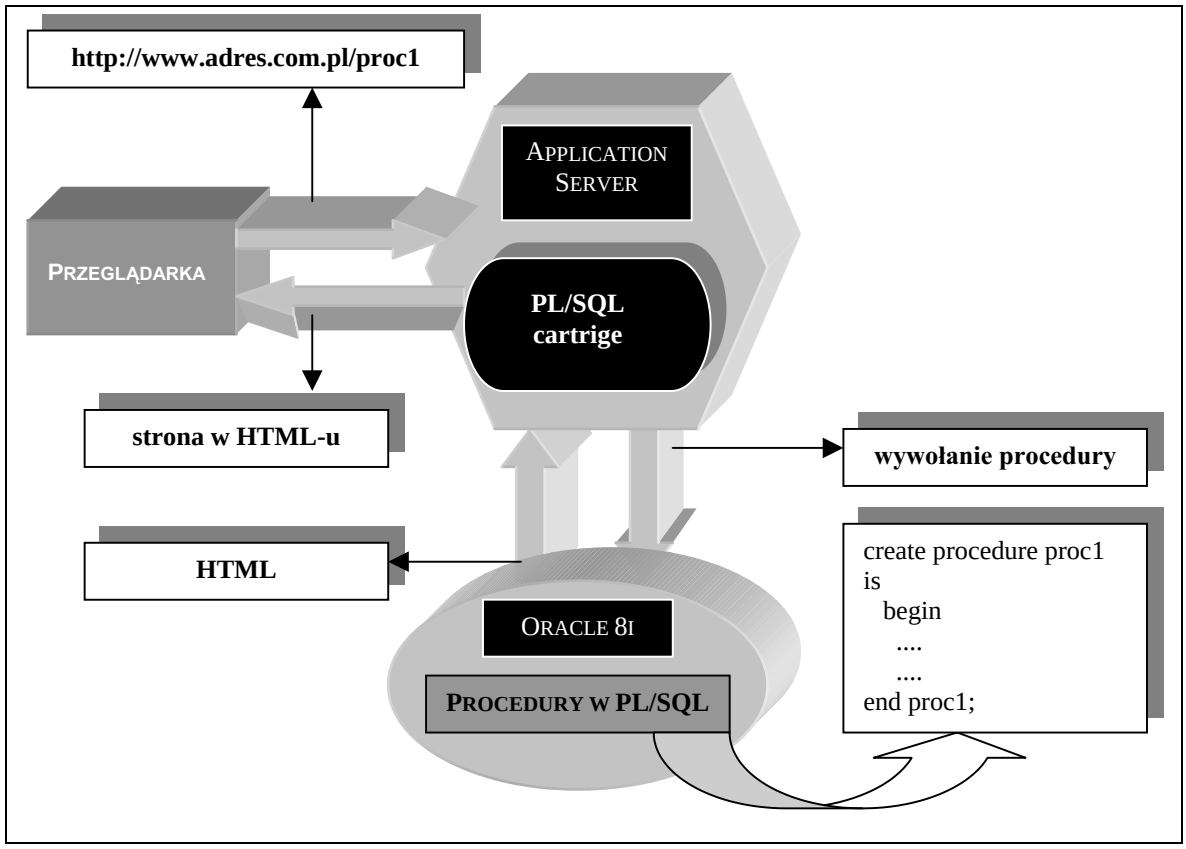
Przyjrzyjmy się najpierw technologii **CGI** (ang. *Common Gateway Interface*). Polega ona na uruchamianiu osobnych procesów w środowisku uruchomieniowym serwera WWW. Program może być napisany w dowolnym języku, którego kompilator lub interpreter jest dostępny na serwerze. Może być to zarówno interpretowany język skryptowy, na przykład Perl, jak i język kompilowany C lub Pascal. Program w CGI może komunikować się, na przykład z bazą danych, za pomocą standardu **ODBC** (ang. *Open Database Connectivity*), a rezultaty w postaci sformatowanego HTML-a są kierowane bezpośrednio do przeglądarki. Jednak CGI ma sporo wad. Po pierwsze,

każde żądanie powoduje ponowne uruchomienie programu CGI. Drugą wadą jest trudność przenoszenia programów na inne platformy sprzętowo-programowe.

Inną technologią jest **PHP**. PHP jest specjalnym językiem skryptowym, który jest umieszczony w kodzie stron HTML-owych na serwerze WWW. Podczas pobierania takich stron skrypt jest wykonywany generując odpowiedni rezultat. Zaletami PHP jest łatwa integracja z bazami danych, dostępność wielu funkcji i prostota. PHP jest obecnie szeroko używaną technologią nawet przy tworzeniu dużych portali.

Również duże firmy takie jak: Netscape czy Oracle przedstawiły swoje rozwiązania do tworzenia portali internetowych. Netscape oferuje bezpośredni interfejs do swoich serwerów Web-owych – **NSAPI** (ang. *Netscape Server API*). NSAPI jest zbiorem funkcji rozszerzających funkcjonalność serwera Web-u. NSAPI jest rozwiązaniem o wiele szybszym od CGI – oferuje wiele funkcji zarówno do połączeń bazo-danowych i autoryzacji użytkowników. Szybkość działania wynika z lepszej integracji z serwerem. Aplikacje NSAPI mogą być uruchamiane jako proces serwera, mogą współdzielić dane i zasoby z serwerem.

Kolejną platformę zaoferowała firma Oracle. Jest to **Oracle Web Application Server** (**OWAS**) + **PL/SQL Cartridge**. OWAS jest serwerem WWW rozszerzonym o możliwość generowania stron HTML kierowanych do przeglądarki. Funkcjonalność serwera rozszerzona jest o PL/SQL Cartridge, który pozwala tworzyć i przechowywać na serwerze procedury napisane w języku PL/SQL, służące do generowania stron w HTML-u. Zyskujemy dzięki temu doskonałą dostępność do danych (poziom PL/SQL-a) za cenę umieszczania kodu języka PL/SQL i HTML razem. Z naszych doświadczeń wynika, że takie podejście przy dużych projektach zmniejsza przejrzystość kodu, utrudnia zmiany i zarządzanie projektem. Wadą tej technologii jest także stosunkowo wysoka cena.



Rys. 1. Architektura Oracle Web Application Server i PL/SQL Cartridge

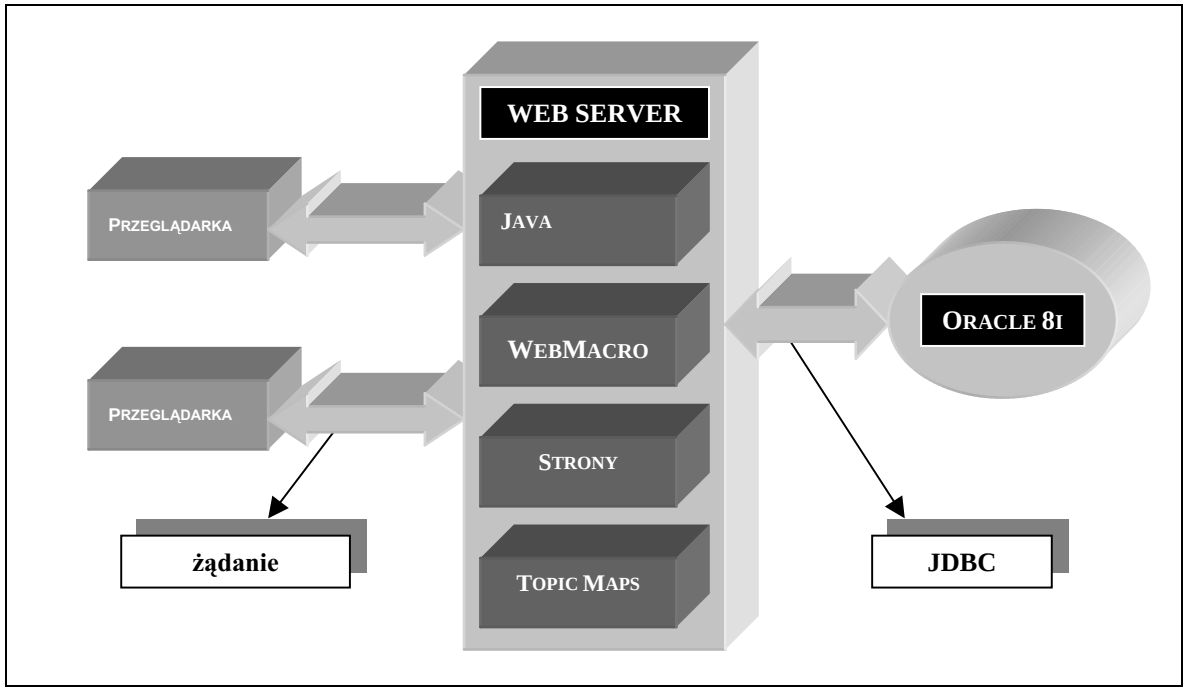
Wszystkie sygnalizowane powyżej technologie mają swoje zalety i wady. Wykorzystywanie **CGI** oznacza małą elastyczność (uzależnienie się od języka i platformy systemowej), stosunkowo

niską szybkość działania (każde żądanie powoduje uruchomienie nowej instancji programu). **PHP** pomimo wielu swoich zalet (łatwy dostęp do baz danych, szybkość, dobre wsparcie techniczne) ma także wady. Na przykład, w dużych projektach umieszczanie skryptów napisanych w PHP naprzemiennie z HTML-em zmniejsza jego czytelność i utrudnia dokonywanie zmian. Stosując **NSAPI** zyskujemy dużą szybkość działania za cenę całkowitego uzależnienia się od platformy firmy Netscape. Rozwiązanie firmy Oracle jest wydajne i otwarte na rozwój, ma jednak pewne mankamenty [1]. Zostało to przez autorów niniejszego artykułu zaobserwowane podczas realizacji projektu internetowego informatora turystycznego dla gminy Krokowa [1, 2]. Część stron informatora były to strony statyczne, natomiast reszta była generowana dynamicznie z bazy danych przy pomocy **Oracle PL/SQL Cartridge**. Składowane na serwerze procedury, napisane w języku PL/SQL generowały na podstawie zapytań do bazy danych kod w HTML-u. Rozwiązanie miało jedną wadę, podobnie jak w PHP kod PL/SQL-a przeplatał się z HTML-em. Utrudniało to w znacznym stopniu zarządzanie logiką jak i samym wyglądem strony. Wpierw należało wygenerować strony przy użyciu generatorów HTML-a, aby następnie zamieniać je na ciąg instrukcji PL/SQL-a wyświetlających skonstruowane strony. Tym samym wykonywana była ta sama praca dwa razy. Architekturę takiego rozwiązania przedstawiono na rysunku 1.

3. Język Java i technologie z nim związane jako uniwersalne narzędzia budowy aplikacji internetowych

3.1. Charakterystyka ogólna technologii języka Java

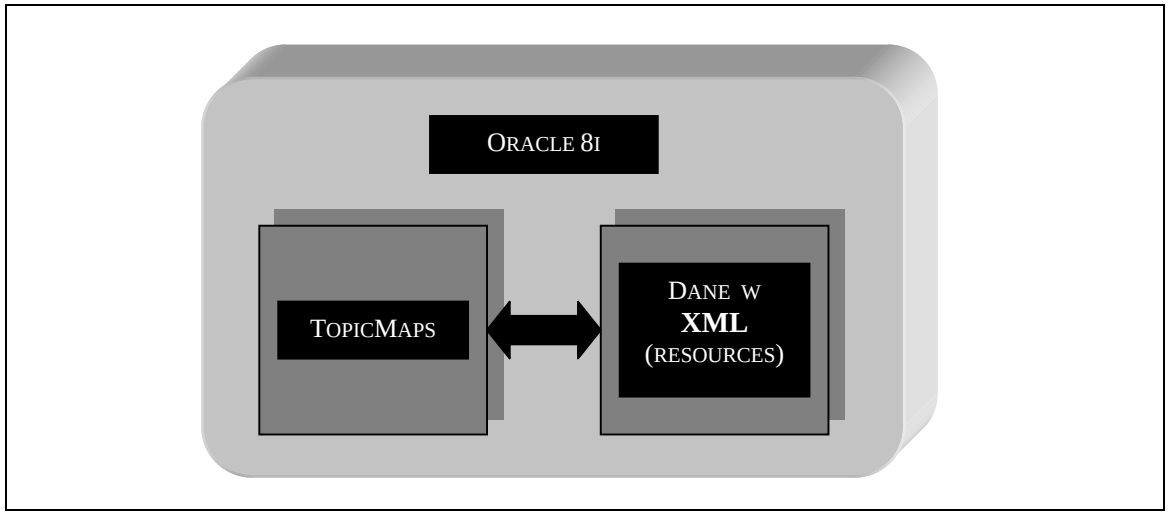
Ten powstały kilka lat temu język programowania w znaczący sposób zmienił pojęcie projektowania i budowania ośrodków webowych. Java zawiera elementy programowania zarówno strukturalnego jak i obiektowego, zdarzeniowego i współbieżnego. Java jest zarazem bardzo bezpiecznym środowiskiem pisania programów. Wokół Javy rozwinęło się bardzo wiele technologii, takich jak: **Java Servlets**, **JDBC** (ang. *Java Database Connectivity'*), **JSP** (ang. *Java Server Pages*), **RMI** (ang. *Remote Method Invocation*), **WebMacro**. Klasy Javy standardowo dostarczają funkcje kryptograficzne niezbędne podczas tworzenia aplikacji e-biznesowych. Java wspomaga także proces parsingu języka XML.



Rys. 2. Architektura nowoczesnego portalu internetowego oparta na technologii języka Java

Przedstawimy obecnie jak przy zastosowaniu języka Java i technologii z nim związanych oraz baz danych można zbudować nowoczesny portal internetowy, charakteryzujący się dużą wydajnością pracy, elastycznością i otwartością rozwiązań. Rysunek 2 przedstawia ogólną architekturę takiego systemu.

Najniższą warstwę (rys. 3) stanowi serwer bazy danych, np. **Oracle 8i**. Na nim przechowywane są dane w postaci XML-a. Przechowywane dokumenty są w postaci tak zwanych **Topic Maps**. Topic Maps są zbiorem dokumentów w SGML-u lub jak w naszym przypadku w XML-u. Topic Maps wykorzystywane są do definiowania zależności i związków pomiędzy tak zwanymi *topikami*. Topic Maps zostaną bliżej przedstawione w dalszej części artykułu.



Rys. 3. Najniższa warstwa architektury systemu wykorzystującego technologię Java

Warstwę środkową stanowi serwer WWW, na przykład Netscape Enterprise Server 4.0. Na serwerze znajduje się zbiór stron statycznych oraz zbiór tak zwanych *servletów* napisanych w języku Java wykorzystujących technologię WebMacro oraz interfejs JDBC do łączenia się z bazą danych.

Servlety są programami wykonywanymi po stronie serwera WWW jako wątki wirtualnej maszyny Javy - JVM (ang. *Java Virtual Machine*). Servlet może zostać uruchomiony wraz z przekazanymi doń parametrami, na przykład z formularza. Ponieważ servlet jest programem napisanym w Javie, można w nim korzystać z całego dostępnego interfejsu Java API - w tym z mechanizmów dostępu do baz danych (JDBC), zdalnych wywołań metod (RMI) oraz interfejsu CORBA. Parametry pobrane ze strony można przekazywać do następnego servletu, tworząc w ten sposób kaskadę, w której każdy servlet odpowiedzialny jest za fragment witryny. Na wyjściu servlet generuje stronę w HTML-u. Do uruchomienia servletów konieczne jest funkcjonowanie serwera WWW z zaimplementowanym modułem Javy. Najpopularniejszym z nich jest obecnie Apache z dodatkowym modułem jServe.

Jednak takie podejście niewiele różniłoby się od CGI, czy Oracle Web Application Server z PL/SQL Cartridge. Zyskujemy nowoczesny język programowania, łatwy dostęp do baz daych, programowanie rozproszone za cenę umieszczania kodu HTML wewnątrz programu w Javie. Aby temu zapobiec można zastosować nowatorską technologię **WebMacro**. Czym jest WebMacro ? WebMacro to zbiór klas Javy rozszerzający funkcjonalność servletów o możliwość odizolowania logiki servletów od projektowania wyglądu stron przez nie generowanych. Takie podejście w dużym stopniu ułatwia zarządzania pracą programistów i projektantów stron podczas realizacji dużego projektu. Technologia WebMacro zostanie omówiona dokładnie w dalszej części pracy. Servlety napisane w Javie do łączenia się z bazą danych wykorzystują interfejs JDBC.

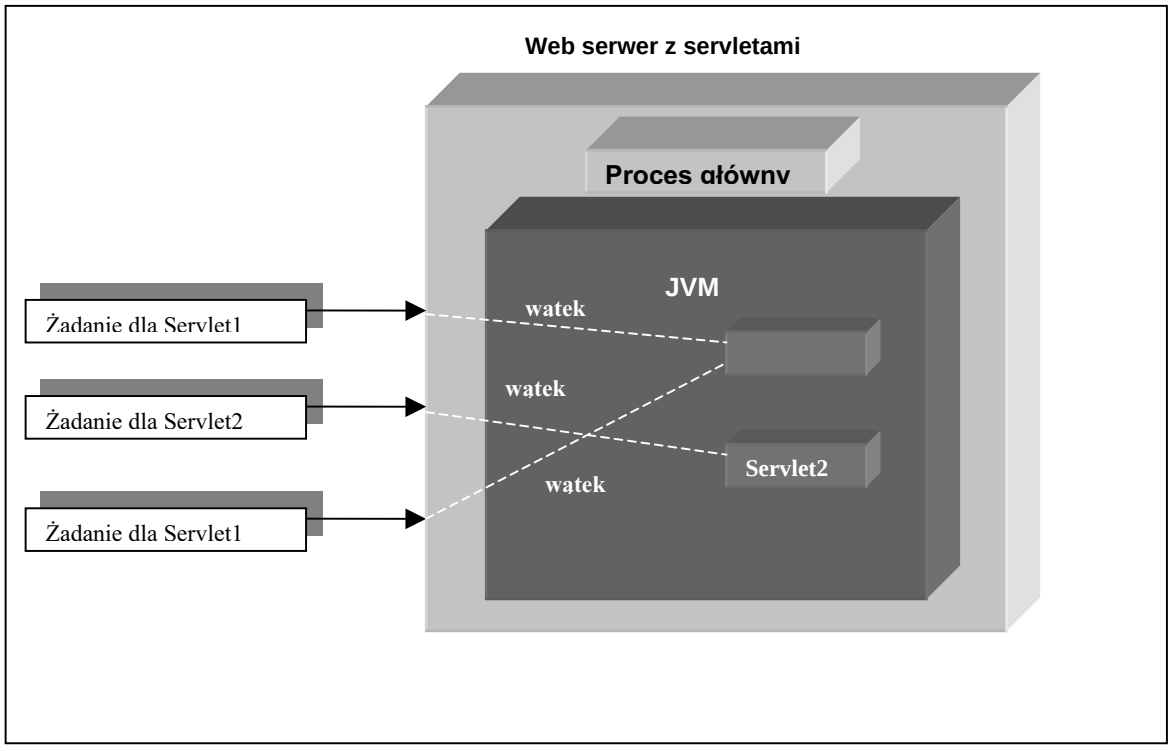
Trzecią wartstwę stanowi przeglądarka WWW. Tutaj wykonywane są skrypty napisane w języku JavaScript, oraz aplety napisane w Javie. Przy pomocy przeglądarki zachodzi również interakcja z portalem.

W dalszej części przedstawimy bliżej wspomniane technologie tj. WebMacro, Topic Maps, XML, JDBC wykorzystywane podczas budowy dużych portali internetowych.

3.2. Servlety

Servlety są rozszerzeniami serwerów webowych – klasa Javy może zostać załadowana dynamicznie w celu rozszerzenia funkcjonalności serwera. Servlety są podobne do firmowych rozszerzeń serwera poza tym, że działają w środowisku Java Virtual Machine, przez co są bezpieczne i przenaszalne.

W przeciwieństwie do CGI i FastCGI, które używają wielu procesów w celu obsłużenia osobnych żądań, servlety są w całości wykonywane w odseparowanych wątkach procesu głównego serwera. Kolejną zaletą servletów jest to, że są one przenaszalne w dwóch aspektach: w stosunku do systemu operacyjnego (JVM) oraz serwera WWW. Servlety ze względu na swoją naturę pozostają w silnym związku z serwerem, na przykład servlet może użyć serwera w celu translacji ścieżek dostępu do plików, wykonywać logowanie, przeprowadzać proces autoryzacji, wykonywać mapowanie typów MIME (ang. *Multipurpose Internet Mail Extensions*). Servlety cechują się także bezpieczeństwem wynikającym wprost z języka Java (np. ścisła kontrola typów, obsługa błędów poprzez wyjątki, możliwość użycia *Java Security Manager*). Na rysunku 4 przedstawiono ogólny schemat życia servletu.



Rys. 4. Cykl życia servletu

3.3. WebMacro

WebMacro to zbiór klas rozszerzających funkcjonalność standardowych servletów o możliwość korzystania z szablonów. Tym samym umożliwiają one rozdzielenie i zrównoleglenie pracy projektantów stron i pracy programistów. WebMacro jest rozpowszechniane z licencją GPL, łącznie

ze źródłami. Jak na razie dostępna jest tylko wersja *pre-release,* jest ona jednak bardzo dobrze przetestowana i nie sprawia najmniejszych problemów.

Programista pisząc servlet, korzystając z WebMacro tworzy klasę dziedziczącą z klasy WMServlet, w której musi oprogramować metodę *handle().* To właśnie ta metoda – analogicznie do metod *doGet()* i *doPost()* w klasycznych servletach - jest wykonywana aby obsłużyć nadchodzące żądanie HTTP. Następnie musi tylko wyznaczyć wartości zmiennych jakie mają być dostępne podczas przetwarzania szablonu (ang. *template*) oraz oczywiście określić, którego szablonu WebMacro ma użyć do wygenerowania dokumentu. Wszystkie zmienne, które mają być widoczne podczas przetwarzania szablonu programista umieszcza w specjalnym obiekcie zwanym kontekstem. W kontekście można umieszczać dowolnego typu obiekty a także kolekcje obiektów (np. tablice). Zadaniem projektanta stron jest stworzenia potrzebnych szablonów, które niewiele różnią się od standardowych dokumentów HTML. Zawierają one tylko kilka dodatkowych dyrektyw umożliwiających dostęp do danych udostępnianych przez programistę. Szablon jest poddawany procesowi parsingu przez WebMacro i przetwarzany do wewnętrznej reprezentacji (dla większej efektywności). Szablon analizowany jest raz, natomiast wykonywany wiele razy (za każdym razem z inną zawartością kontekstu).

3.3.1. Język używany w szablonach

Język dostępny w szablonach nie należy traktować jako język skryptowy. Założeniem twórców WebMacro było rozdzielenie logiki aplikacji od jej wyglądu i dlatego jest to język bardzo prosty i w dużym stopniu intuicyjny. Poniżej przedstawimy jak w tym języku można uzyskać dostęp do zmiennych, jak nawigować po kolekcjach obiektów, nadawać wartości zmiennym oraz korzystać z instrukcji warunkowych.

Zmienne

Zmienne to najprostszy sposób na dostęp do informacji dostępnych w kontekście. Zmienne w szablonach poprzedzone są znakiem $ (podobnie jak w języku Perl), na przykład:

```
$Title
$Customer.Name
$Customer.Products.findProduct("window").PartNo
```

Każda ze zmiennych wskazuje na jakiś obiekt w kontekście. Zadaniem WebMacro jest znalezienie poprawnej wartości w kontekście, natomiast programista musi zadbać oto by ta wartość się tam znalazła. Obiekt *Customer* ma własność (pole lub odpowiednią metodę) *Name.* Do wydobycia poszczególnych własności służy operator 'kropka'. Operatora tego można używać wielokrotnie aby dostać się do żądanej wartości. Trzeci przykład zawiera wywołanie metody *findProduct()* z parametrem. Oczywiście w większości przypadków wystarczy proste odwoływanie się do pól w obiektach, ale czasami może być konieczne wywołanie metody na rzecz jakiegoś obiektu w kontekście – co WebMacro także umożliwia.

Stałe

Stałe w WebMacro mogą być jednym z sześciu typów: słowo, liczba, wartość logiczna, kolekcja, wskaźnik na zmienną, łańcuch znaków. *Słowo* to ciąg liter i cyfr, na przykład: *hello1*. Są dwa specjalne słowa: *true* i *false*, które reprezentują wartości logiczne. *Liczba* to ciąg cyfr, na przykład: `#set $a = 10` jest równoważne umieszczeniu w kontekście obiektu *Integer(10).* Możemy wymusić na WebMacro potraktowanie liczby jako typu *Long* przez dodanie sufiksu **l** lub **L** na przykład: 10L. Jak na razie WebMacro nie wspiera liczb typu *Float* i *Double.* W razie potrzeby WebMacro może przekształcić obiekt typu *Integer (Long)* na wartość prostą *int (long)*, można więc tych stałych używać jako parametry wywołania funkcji, które oczekują wartości typu *int* lub *long.*

Wartość logiczna w szablonie reprezentowana jest przez słowa: *true* i *false.* Są one odpowiednikami stałych języka JAVA: *Boolean.TRUE* i *Boolean.FALSE*. Można ich także używać jako parametr wywołania funkcji, które oczekują wartości logicznej.

Kolekcja to uporządkowana lista stałych na przykład: `#set myList = [ one, two, 3, $variable]`. Możliwe jest dowolne zagnieżdżanie kolekcji na przykład:

```
#set myList = [ [ one, one ], [two, two, 2], [three] ]
```

Łańcuch znaków to dowolny ciąg znaków umieszczony w cudzysłowie. Ponieważ ciąg znaków może zawierać w sobie wskaźnik na zmienną, aby wprowadzić do znak '$' trzeba go poprzedzić znakiem '\' (ta sama zasada odnosi się do inych znaków specjalnych), na przykład:

```
#set $b = "to jest ciag znakow: \$a, a to wskaznik na zmienna $a"
```

Wskaźnik na zmienną – zapisujemy go identycznie jak zmienną i możemy wykorzystywać jako parametr wywołania funkcji, na przykład: `$order.findProduct($Product1).Prize`

Dyrektywy

Dyrektywy poprzedzone są znakiem # i muszą wystąpić w nowej linii (mogą być poprzedzone znakami spacji). Wiele dyrektyw operuje na bloku. Każdy blok rozpoczyna się znakiem '{' i kończy znakiem '}'. Bloki, a tym samym dyrektywy, można zagnieżdżać.

komentarz

Dwa lub więcej znaków # tworzy komentarz – reszta linii zostanie zignorowana.

#foreach $element in $kolekcja {...}

Dyrektywa pozwala przejrzeć wszystkie obiekty w kolekcji. Blok zostanie wykonany tyle razy ile jest obiektów w kolekcji, za każdym razem zmienna *$element* będzie przyjmować wartość kolejnego obiektu, na przykład:

```
<ul>
#foreach $customer in $list {
  <li>$customer.name lives at $customer.address
}
</ul>
```

#if(warunek) {...} #else {...}

Jest to standardowa dyrektywa warunkowa spotykana w większości języków programowania. Druga część dyrektywy, to znaczy „#else {}" jest opcjonalna. W warunku dyrektywy #if można używać następujących operatorów:

&& - and (binarny),
|| - or (binarny),
! - negacja (unarny),
== - równy (binarny),
!= - różny (binarny).

W przypadku dwóch ostatnich operatorów należy mieć na uwadze, że bazują one tylko na metodzie *Object.equals()*. Zmienna traktowana jest jako *false* jeżeli nie została znaleziona w kontekście, jest *null*, bądź jest typu *Boolean* i ma wartość *Boolean.FALSE*. W przeciwnym wypadku traktowana jest jako *true*. Wyrażenia składające się na warunek można dowolnie grupować przy użyciu nawiasów okrągłych, na przykład:

```
#if ( $Customer.owesMoney() && ($Customer.Name != "Fred") ) {
     Pay up, or else!
} #else {
     Thank you for your patronage.
}
```

#include plik – powoduje wczytanie zewnętrznego pliku – WebMacro nie będzie wykonywać operacji analizy (*parsingu)*.

#parse plik – powoduje wczytanie zewnętrznego pliku – WebMacro potraktuje go jako część szablonu i go przeanalizuje.

#set $zmienna = wartość

Do zmiennych w kontekście można się odwoływać, ale także podstawiać pod nie wartości, na przykład:

```
#set $name="Jan Kowalski"
Nazywam się $name
```

WebMacro potrafi także dokonywać zmian na bardziej skomplikowanych obiektach wykorzystując mechanizm introspekcji.

```
#set $customer.Name="Jan Kowalski"
#set $order.findProduct($Product1).Prize = $newPrize
```

#use 'parser' until znacznik-konca

Powyższa dyrektywa nakazuje WebMacro używanie innego parsera dla wskazanego bloku tekstu. Obecnie dostępny jest tylko parser *„text”*, który nie robi nic. Jest on jednak bardzo potrzebny ponieważ znaki {}, które mogą występować w JavaScript bardzo często przeszkadzają w poprawnej interpretacji szablonu, na przykład:

```
#use 'text' until '-end-'
ten tekst nie będzie analizowany przez parser WebMacro więc
znaki {{{}}} nie będą nam przeszkadzać
-end-
A ten tekst będzie analizowany.
```

Ponad to w kontekście umieszczane są zawsze następujące obiekty:

- *$CGI* umożliwiający dostęp do zmiennych jakie spotykane są w CGI,
- *$Form* umożliwiający dostęp do zmiennych z formy,
- *$Config* umożliwiający dostęp do zmiennych zdefiniowanych w pliku *WebMacro.properties*,
- *$Cookie* umożliwiający odczytywanie i zmienianie *Cookies*,
- *$Session* umożliwiający dostęp do API sesji z JSDK,
- *$Request* to obiekt *HTTP request* (*HttpServletRequest*).
- *$Response* to obiekt *HTTP response* (*HttpServletResponse*)

3.3.2. Introspekcja – wyznaczanie wartości zmiennych

W WebMacro zastosowano introspekcję do wyznaczania wartości zmiennych. Dzięki temu programista może umieszczać w kontekście obiekty dowolnego typu. Introspekcja, w skrócie polega na tym, że WebMacro aby wyznaczyć wartość zmiennej będzie interpretować zapis w szablonie na kilka sposobów – poszukując pól i metod w obiekcie o danej nazwie. Najlepiej rozpatrzyć to na przykładzie. Załóżmy, że w szablonie znajduje się zapis:

$Order.Customer.Fred

WebMacro starając się wyznaczyć wartość tej zmiennej podejmie następujące próby (dokładnie w takiej kolejności):

1. sprawdzi czy obiekt *Order* ma publiczne pole *Customer*,
2. sprawdzi czy obiekt *Order* ma publiczną metodę *getCustomer()*,
3. sprawdzi czy obiekt *Order* ma publiczną metodę *getCustomer(”Fred”)*,
4. sprawdzi czy obiekt *Order* ma publiczną metodę *get(”Customer”)*.

Gdy zajdzie przypadek trzeci to dla tego przykładu zakończy to wyznaczanie wartości. W pozostałych wypadkach na rzecz wyznaczonego obiektu, WebMacro zastosuje identyczną procedurę by wyznaczyć własność *Fred*.

W przypadku nadawania wartości (przy korzystaniu z dyrektywy #set) WebMacro zastosuje analogiczną procedurę, przy czym zamiast poszukiwać odpowiednich metod *get* będzie poszukiwać metod *set*.

Kolekcje

Dla WebMacro obiekt będzie kolekcją, a tym samym może być przeglądany za pomocą dyrektywy *#foreach*, jeżeli:

- obiekt jest tablicą,
- obiekt implementuje interfejs Iterator,
- obiekt implementuje interfejs Enumeration,
- ma publiczną metodę "Iterator iterator()",
- ma publiczną metodę "Enumeration elements()".

3.3.3. Pisanie servletów z wykorzystaniem WebMacro

Najprostszym sposobem napisania servletu korzystającego z WebMacro jest stworzenie klasy dziedziczącej z WMServlet. W klasie tej musimy zdefiniować metodę *handle()*, która posiada jeden parametr typu *WebContext* (jest to wspomniany już kontekst) i zwraca obiekt typu *Template*. Poprzez kontekst programista ma dostęp do wszystkich obiektów dostępnych standardowo w servletach (tj. *Cookies, Session API, HttpServletRequest, HttpServletResponse*). Dostęp do podstawowych operacji jak: pobieranie danych z formy i operacje na *Cookies* został uproszczony poprzez dodanie kilku metod dostępnych bezpośrednio w obiekcie *WebContext*. Głównym zadaniem programisty jest wyznaczenie wszystkich potrzebnych zmiennych i umieszczenie ich w kontekście. Zmienne w kontekście są przechowywane za pomocą tablicy haszowej. Programista umieszcza więc w kontekście obiekty lub kolekcję obiektów pod odpowiednim kluczem (łańcuchem znaków, który będzie stanowił pierwszy człon w nazwie zmiennej w szablonie). Rozważmy następujący przykład:

```
public Template handle(WebContext context){
  context.put(”counter”, new Integer(10));
  Vector v = new Vector();
  v.addElement(”Ala”);
  v.addElement(”Ola”);
  v.addElement(”Ewa”);
  context.put(”vector”,v);
}
```

W szablonie możemy odwołać się do tych zmiennych przez

```
$counter<br>
#foreach $i in $vector {
  $i<br>
}
```

Dokument wygenerowany przez ten fragment kodu będzie wyglądał następująco:

```
10<br>
Ala<br>
Ola<br>
```

```
Ewa<br>
```

Ponieważ mamy do czynienia z tablicą haszową ponowne użycie tego samego klucza spowoduje nadpisanie poprzedniego obiektu. Uwaga ta także dotyczy dyrektywy *#set* bowiem operuje ona na tym samym kontekście *(#set $a = 10* w szablonie jest równoważne z wykonaniem *context.put* (*"a",new Integer(10))* w kodzie servletu). Trzeba także uważać wybierając nazwę dla zmiennej *$element* w dyrektywie „*#foreach $element in $kolekcja*", gdyż ona także jest przechowywana w kontekście.

Metoda *handle()* musi zwrócić obiekt typu *Template*. Za dostarczanie tego obiektu w WebMacro odpowiedzialny jest specjalny moduł tzw. *TemplateProvider*. Z modułu tego korzysta metoda „*Template getTemplate(String)*", którą programista może wykorzystać do określenia szablonu jaki ma być użyty do wygenerowania strony. Standardowy *TemplateProvider* wyszukuje szablony w katalogach (wskazanych w pliku konfiguracyjnym) na dysku (każdy szablon jest plikiem). Dostępne są także inne wersje tego modułu umożliwiające pobieranie szablonów przez protokół HTTP, bądź z bazy danych.

WebMacro ze względów efektywnościowych ma zaimplementowany mechanizm buforowania szablonów (czas po jakim szablony muszą być ponownie pobrane można ustawić w pliku konfiguracyjnym). Dodatkowo programista może zaimplementować metody *start()* i *stop(),* które są wykonywane dokładnie raz, odpowiednio przy starcie servletu i przy jego zatrzymywaniu.

3.3.4. Konfiguracja

Aby móc korzystać z WebMacro należy wykonać następujące kroki:

1. dodać plik *webmacro.jar* do CLASSPATH w konfiguracji serwera WWW,
2. skopiować plik *WebMacro.properties* do katalogu, który jest wymieniony w CLASSPATH,
3. w pliku *WebMacro.properties* zmienić wartość parametru *TemplatePath* (parametr ten określa katalogi, w których będą wyszukiwane szablony),
4. dodatkowo można w pliku *WebMacro.properties* zmienić wartości innych pożytecznych parametrów zgodnie z załączonymi w tym pliku instrukcjami (np.: *TemplateExpireTime* – czas buforowania szablonów, *LogFile* – nazwa pliku, do którego będą trafiać informacje przydatne przy debugowaniu servletów).

3.3.5. Dlaczego warto używać WebMacro?

WebMacro jest potężnym i wydajnym narzędziem, które doskonale wspiera tworzenie aplikacji internetowych. Rozdzielenie pracy programisty i projektanta stron jest bardzo ważne przy tworzeniu dużych i skomplikowanych systemów. Dzięki temu zyskuje się na przejrzystości kodu aplikacji, która nie jest „zaśmiecana" przez kod HTML i vice versa. Przejrzystość kodu wpływa na zmniejszenie ilości błędów w aplikacji. Możliwe jest zrównoleglenie pracy programistów i projektantów stron. Proste poprawki edytorskie sprowadzają się tylko do poprawienia szablonu – nie ma potrzeby ponownej rekompilacji aplikacji. Ponieważ korzysta się z szablonów nic nie stoi na przeszkodzie by tą samą aplikację przystosować do innych technologii, na przykład WAP – wystarczy tylko użyć innych szablonów.

3.4. JDBC

JDBC jest interfejsem API języka Java pozwalającym na wykonywanie zapytań SQL (zbiór klas oraz interfejsów). Wykorzystanie tego standardu pozwala na napisanie uniwersalnego programu który będzie można użyć wielokrotnie do łączenia się z różnymi bazami danych. W skrócie JDBC pozwala na:

- nawiązanie połączenia z bazą danych,
- wysyłanie zapytań SQL,
- przetwarzanie wyników.

Poniższy przykład stanowi ilustrację tych możliwości.

```
   Connection con = DriverManager.getConnection (
                    "jdbc:oracle:thin:@host:port:sid",
                    "login",
                    "password");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM
tabelka1");
        while (rs.next()) {
          int x = rs.getInt("a");
          String s = rs.getString("b");
          float f = rs.getFloat("c");
          }
```

Aby móc skorzystać z tego standardu należy się zaopatrzyć w sterownik, który jest implementacją interfejsów – odpowiedni dla danego producenta bazy danych. JDBC nie nakłada żadnych restrykcji na polecenia SQL, gwarantuje dużą elastyczność, pozwalając na użycie specyficznej dla danego producenta składni SQL. JDBC wymaga aby sterownik dostarczył przynajmniej możliwości ANSI SQL-2. Oznacza to, że użytkownik może liczyć na podstawowy poziom funkcjonalności.

W JDBC wyróżniamy następujące interfejsy podstawowe: *Connection, DriverManager, Statement, ResultSet, PreparedStatement, CallableStatement.*

- *Connection*

Obiekt Connection reprezentuje połączenie z bazą danych. Sesja połączenia zawiera polecenia SQL, które są wykonywane a wyniki zwracane przez połączenie. Pojedyncza aplikacja może mieć jedno lub wiele połączeń z daną bazą danych lub połączenia z różnymi bazami. Pracując z bazami często implementuje się pulę połączeń, która znacznie przyspiesza pracę ze względu na fakt, że czas tworzenia nowego połączenia jest stosunkowo duży. Fakt ten uwzględniono w JDBC 2.0, który dostarcza odpowiednich interfejsów.

- *DriverManager*

Klasa *DriverManager* jest warstwą zarządzającą, pracującą między użytkownikiem a sterownikiem. Klasa ta steruje procesem nawiązywania połączenia między bazą danych a odpowiednim sterownikiem.

- *Statement*

Klasa *Statement* jest tworzona przez wywołanie metody *createStatement()* na rzecz obiektu *Connection*. Używana jest w celu prostego wysłania zapytania do bazy danych. Obiekty tej klasy mogą zawierać zapytania SQL, które mogą używać *SQL escape syntax*. *Escape syntax* sygnalizuje, że kod zawarty między znacznikami powinien być różnie obsługiwany. Sterownik JDBC wyszukuje w pytaniu *escape syntax* i zamienia go na kod zrozumiały dla danej bazy danych. Pozwala to na zadawanie zapytań niezależnych od implementacji różnych systemów zarządzania bazą danych, na przykład:

```
stmt.executeQuery("SELECT * FROM pracownicy
     WHERE pracuje_od ={ts `2000-01-01 12:0:0.0 '};
```

- *Resultset*

ResultSet zawiera wszystkie krotki relacji spełniające warunki zapytania oraz umożliwia dostęp do tych krotek poprzez określony zbiór metod *getXXX()*. Obiekt *ResultSet* pozostawia w bazie kursor, który jest otwarty do momentu zamknięcia obiektu.

- *PreparedStatement*

Obiekt *PreparedStatement* używany jest do wykonywania prekompilowanych poleceń SQL zawierających jedno lub więcej pól parametrów (oznaczanych znakiem "?" – są tak zwane parametry IN). Parametrom wejściowym należy następnie przypisać pożądane wartości poprzez użycie odpowiednich metod *setXXX()*. Ponieważ obiekty *PreparedStatement* są prekompilowane ich wykonanie może być szybsze niż obiektów *Statement*. Zapytania SQL wykonywane wielokrotnie (w pętli) są tworzone ze względów wydajnościowych z wykorzystaniem *PreparedStatement*.

```
PreparedStatement pstmt = con.prepareStatement(
                  "UPDATE table4 SET m = ? WHERE x = ?");
pstmt.setLong  (1, 123456789);
pstmt.setString(2, "qwerty");
int rowCount = pstmt.executeUpdate();
```

- *CallableStatement*

Obiekt *CallableStatement* jest wykorzystywany do stworzenia odwołania do przechowywanych w bazie danych procedur i budowany z wykorzystaniem metody *prepareCall* na rzecz obiektu *Connection*, na przykład:

```
CallableStatement cstmt = con.prepareCall( "{call getTestData(?,
?)}");
```

Parametry oznaczone „?” mogą być wejściowe, wyjściowe lub wejściowo-wyjściowe np.:

```
CallableStatement cstmt = conn.prepareCall("{? = CALL balance(?)}");
cstmt.registerOutParameter(1, Types.FLOAT);
cstmt.setInt(2, 13);
cstmt.executeUpdate();
float acctBal = cstmt.getFloat(1);
```

Dla potrzeb pracy z bazą danych za pomocą języka SQL istotne jest odwzorowanie typów danych Java->JDBC->SQL->DBMS. JDBC 2.0 API jest zgodny z SQL-2, a co więcej udostępnia typy danych zgodne z propozycją standardu SQL-3 (na przykład, typ danych BLOB). Teoretycznie konstruktor oprogramowania bazy danych powinien traktować typy danych z baz danych, tak jakby to były typy danych SQL.

3.5. TOPIC MAPS

3.5.1. XML

XML jest idealnym formatem do przechowywania strukturalizowanych informacji, umożliwiającym przy tym wielokrotne używanie danego zbioru informacji przy użyciu różnych aplikacji. XML jako taki nie jest jednak wystarczający, gdy ilość informacji, które otrzymuje użytkownik końcowy jest rzędu megabajtów pomimo, że są one ściśle strukturalizowane. Potrzebny jest mechanizm umożliwiający otrzymywanie żądanych wyników szybko, precyzyjnie i zgodnie z aktualnym kontekstem. Jednym z rozwiązań spełniających te wymagania są TOPIC MAPS®.

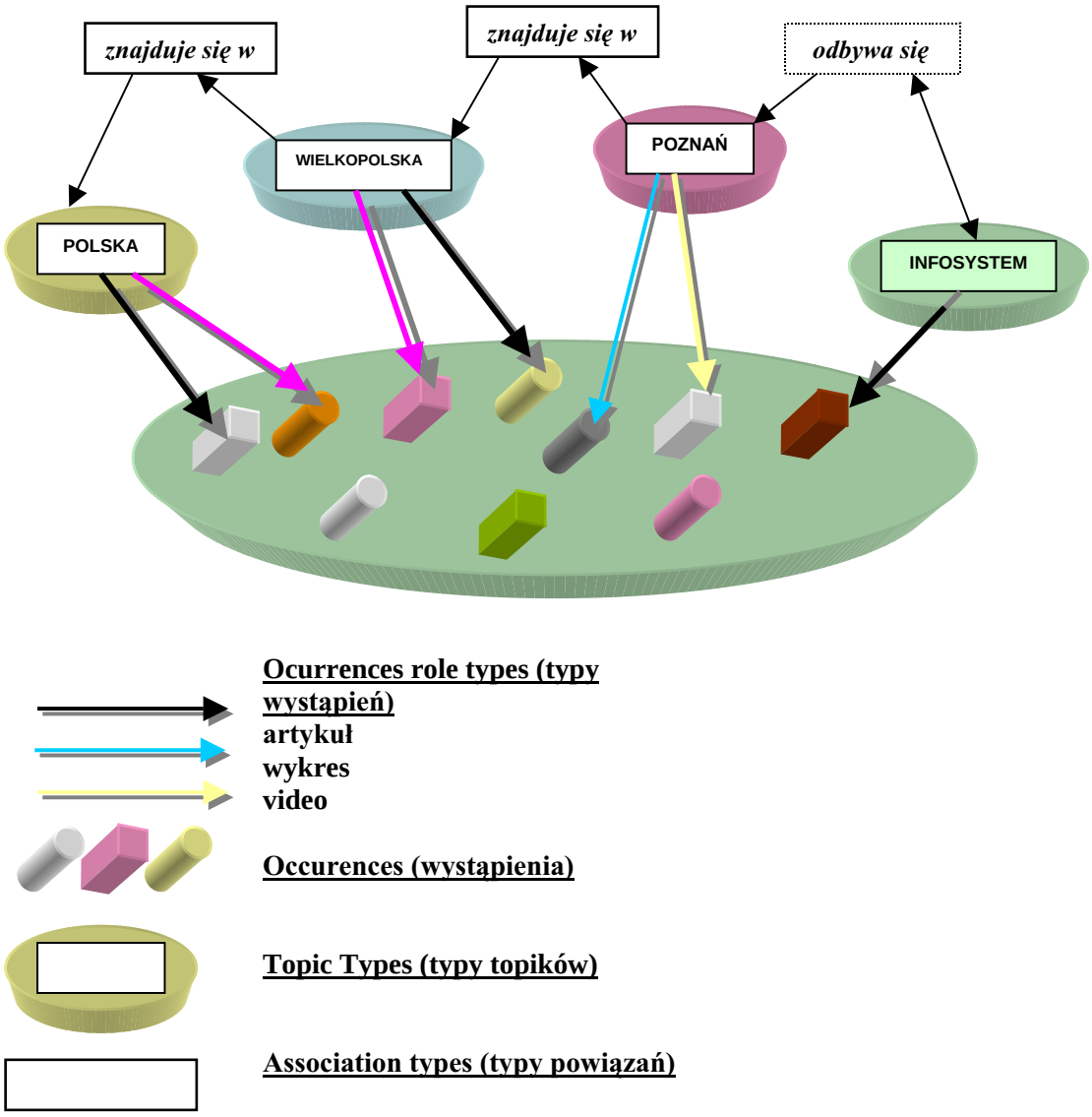
3.5.2. Czym są TOPIC MAPS ?

TOPIC MAPS jest usystematyzowaną notacją używaną do reprezentacji wiedzy. Pozwala ona na definiowanie pojęć – nazywanych tutaj *topikami* i zależności między nimi. Fizycznie jest to zbiór dokumentów (zawierających informacje) połączonych zależnościami, które używają notacji zgodnej ze standardem (*ISO/IEC FCD 13250*). TOPIC MAPS strukturalizują zbiór informacji, poprzez budowanie sieci semantycznej „nad” zasobami. Sieć ta umożliwia łatwą nawigację i selekcję otrzymywanych informacji.

3.5.3. Model TOPIC MAPS

Podstawowym językiem, który umożliwia definiowanie TOPIC MAPS jest język XML. TOPIC MAPS to dokument XML-wy w którym znajdują się elementy różnych typów wywiedzione ze zbioru podstawowych form architektury TOPIC MAPS. Używane one są do definiowania: topików, wystąpień topików (ang. *occurences of topics*), i powiązań między topikami (ang. *associations between topics*). Podstawowymi elementami modelu są:

- *Topic* – reprezentuje dowolne pojęcie: osobę, jednostkę, koncepcję, ideę bez względu czy ona istnieje i jaką ma charakterystykę. Każdy z topików może mieć jeden lub wiele typów (ang. *topic types*). Zależność między topikiem a jego typem jest tym samym co zależność klasa – instancja klasy. Na przykład, topiki: Polska, Wielkopolska, Poznań, Infosystem 2000 oraz typy topików: kraj, województwo, miasto, konferencja.
- *Topic name* – składa się z trzech części: *base name* (nazwa podstawowa), *display name* (nazwa wyświetlana), *sort name* (nazwa dla sortowania). Na przykład: *(base/display/sort):* Poznan/Poznań/poznan.
- *Topic occurence* (wystąpienie topiku) i *occurence role type* (rola wystąpienia) – wystąpienie jest referencją do konkretnego zasobu związanego z danym topikiem. Każde wystąpienie ma określony typ, wyrażony jako typ wystąpienia (*occurence role type*) i zakres wystąpienia (*occurence scope*). Typ wystąpienia jest też sam w sobie topikiem. Na przykład, *wystąpienie (occurence):* wykres_1.xls, wojna.doc, killer.mpg, hey_Joe.mp3 oraz *typ wystąpienia (occurence role types):* wykres, artykuł, video, plik muzyczny.
- *Topic association* (powiązanie topiku) i *association type* (typ powiązania) – powiązanie określa relację między dwoma lub większą ilością topików. Każde powiązanie jest określonego typu. Każdy powiązany topik pełni określoną rolę w związku (ang. *association role type*) Zarówno typ powiązania topików jak i rola jaką pełni ten typ powiązania są także same w sobie topikami. Na przykład, *powiązanie*: Wielkopolska znajduje się w Polsce, Poznań znajduje się w Wielkopolsce, Infosystem odbywa się w Poznaniu, natomiast *typ powiązania*: znajduje się w, odbywa się w oraz *rola powiązania*: województwo/państwo, miasto/województwo, konferencja/miasto.
- *Scope* (zakres): dowolny przydział (nazw, wystąpień, powiązań) dla określonego topiku jest uważany jako poprawny (sensowny) w określonych granicach, które mogą ale nie muszą być wyraźnie wyspecyfikowane. Ta granica ważności danego przydziału jest nazywana zakresem (scope). Na przykład, *zakresy (scopes):* aby zachować rozróżnienie między: miastem „Paris" we Francji, miastem „Paris" w Texasie i bohaterem mitologicznym o imieniu „Paris" musimy powiązać poszczególne topiki z zakresami ważności dla każdego z nich. I tak dla tych przypadków zakresami mogą być : „Francja", „USA", „Mitologia Grecka".



Rys. 5. Przykładowa mapa topików

3.5.4. Zastosowania

TOPIC MAPS mogą znaleźć zastosowanie w:

- kwalifikowaniu zawartości obiektów przechowujących informacje oraz aby ułatwić nawigację przy użyciu odpowiednich metod, takich jak: indeksy, systemy słownikowe, systemy przeszukiwania cytatów, systemów tworzenia dokumentacji technicznych dla przedsiębiorstw,
- filtrowaniu zbiorów informacji dla ograniczenia zakresu danych widocznych dla określonego użytkownika systemu lub konkretnego zastosowania. Filtrowanie może być pomocne w zarządzaniu wielojęzycznymi dokumentami, zarządzaniu trybami dostępu w zależności do polityki ochrony informacji danej instytucji, dostarczaniu częściowego dostępu do informacji w zależności od profilu użytkownika, dziedziny wiedzy itd.,

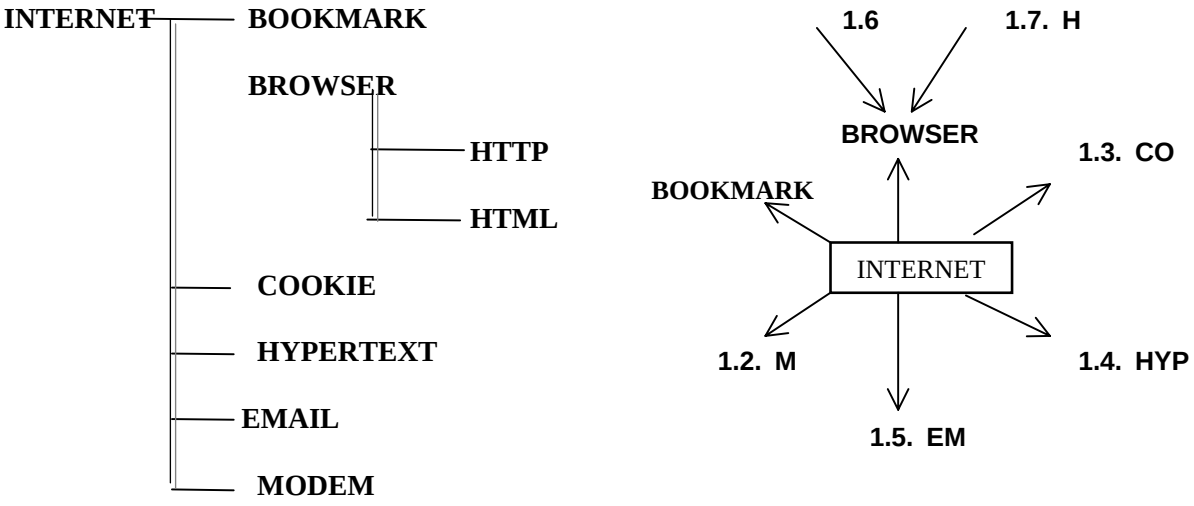
- strukturalizacji niestrukturalizowanej informacji, dla ułatwienia tworzenia interfejsów użytkownika jako map topików umożliwiających łatwe i szybkie nawigowanie wśród informacji.

TOPIC MAPS mogą być rozumiane jako „rozszerzony mechanizm znaczników" dlatego, że użycie ich polega na nałożeniu określonej struktury na konkretny zbiór informacji bez wprowadzania zmian w samych danych i ich formie.

3.5.5. Nawigacja w strukturze TOPIC MAPS

Istnieją dwa sposoby użycia TOPIC MAPS: nawigacja przy użyciu „linków" oraz odpytywanie węzłów (rys. 6).

- Nawigacja przy użyciu linków może zostać zaimplementowana jako graf lub jako struktura oparta na drzewie.



Rys. 6. Drzewo nawigacji i graf nawigacji

Mapa topików może zawierać miliony węzłów i linków dlatego odpowiednio jasna reprezentacja graficzna tej struktury ma ogromne znaczenie dla użytkownika.

- Odpytywanie węzłów wymaga zastosowania języka zapytań zgodnego z koncepcją TOPIC MAPS. Użytkownik definiujący zapytania potrzebuje w tym przypadku dodatkowego wsparcia ze strony aplikacji, na przykład rozwijalne listy, menu wyświetlające informacje dostępne na mapie.

3.5.6. Projektowanie, tworzenie i utrzymywanie map topików (procedury, narzędzia, algorytm, doradztwo)

Projektowanie mapy topików to proces przyrostowy. Definiowanie typów, ról ich nadklas i podklas musi odbywać się pod kontrolą narzędzi wspomagających projektowanie. Bardzo ważne jest zapewnienie spójności mapy.

Pierwszym etapem projektowania jest wygenerowanie wszystkich topików, powiązań i wystąpień. W dużych aplikacjach mogą ich być tysiące, a nawet miliony. Narzędzie do projektowania powinno umożliwiać łatwy dostęp do wszystkich tych obiektów oraz mieć możliwość sprawdzania spójności powstającej mapy.

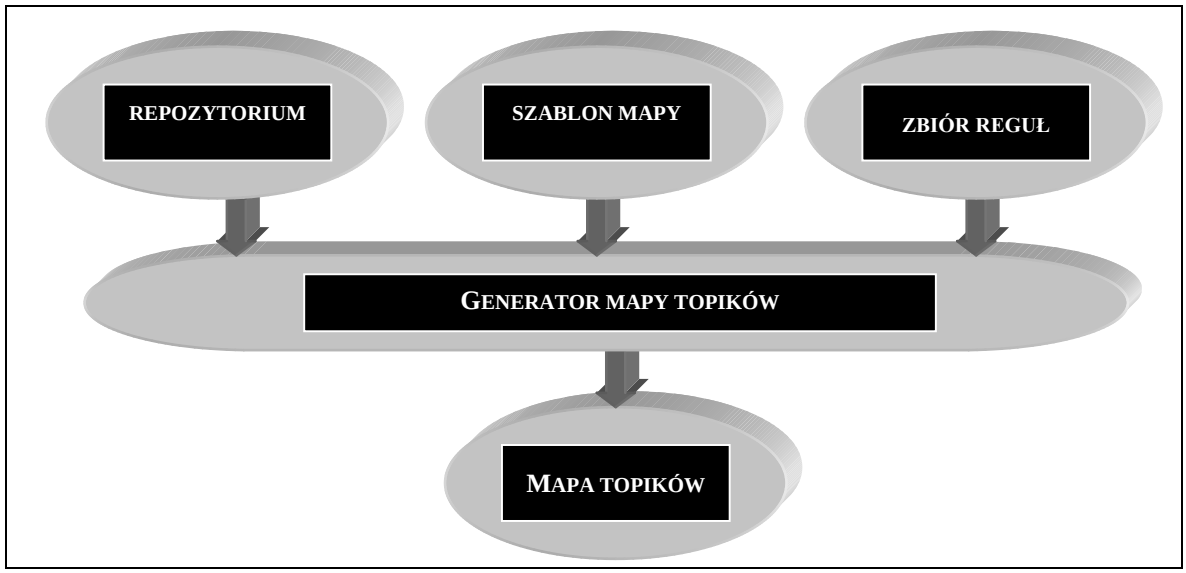
Granica między projektowaniem a tworzeniem mapy topików jest płynna. Tylko w początkowej fazie można rozróżnić etap definiowania typów i ról z jednej strony i definiowania topików, wystąpień, powiązań z drugiej. Podczas rozwijania mapy obie te czynności mogą być wykonywane równolegle przez różne grupy specjalistów.

Automatyczna generacja map topików

Dostępne informacje do automatycznego generowania map są następujące:

- wzorzec mapy topików (ang. *topic map template*) ze zdefiniowanymi typami topików, typami wystąpień, typami asocjacji,
- adresy zasobów informacyjnych w systemie plików, bazie danych, Internecie,
- metadane o zasobach informacyjnych (nazwa, format, klasyfikacja),
- struktura zasobów informacji dostępna jako instancje XML.

Automatyczna generacja jest kontrolowana przez zbiór reguł zgromadzonych w skrypcie. Program przetwarzający interpretuje skrypt, pobiera dostępne informacje jako parametry wejściowe. Wynikiem jest mapa topików zgodna ze standardem.



Rys. 7. Schemat automatycznego generowania map topików

Identyfikacja topików i wystąpień

Pierwszą fazą automatycznej generacji jest identyfikacja wśród zasobów informacji kandydatów na topiki określonego typu. Jeżeli zasoby zostaną odnalezione tworzony jest topik i używany jest algorytm wyszukujący nazwę dla topiku z zasobów lub metainformacji. Potem tworzone jest powiązanie topiku z zasobem fizycznym (ang. *occurence*).

Przykładowa reguła wyszukująca topiki w zbiorze zasobów może być następująca:

```
if resource fulfills metadata <condition> and/or
contains structure <element> in <context> containing <content>
then create topic of <type> with name derived from metadata
<field> or
   name derived from <element> in <context> and
   create occurence to resource with <role>
```

Identyfikacja powiązań

Drugą fazą automatycznej generacji jest tworzenie powiązań pomiędzy topikami stworzonymi w pierwszej fazie. Jest to bardzo złożone zadanie, ponieważ uzyskiwanie struktur wiedzy z istniejących danych odbywa się przy użyciu algorytmów sztucznej inteligencji.

3.5.7. Dostępne aplikacje tworzące mapy topików

Norma ISO 13250 została opublikowana w styczniu 2000. Aplikacje służące do tworzenia map topików są jeszcze w trakcie rozwoju. Przodującą firmą w zakresie tworzenia oprogramowania wspomagającego TOPIC MAPS jest firma InfoLoom. Dostarcza ona oprogramowanie Topic Map Loom® rozwijane od 5 lat.

3.5.8. Dlaczego pojawienie się TOPIC MAPS jest ważnym wydarzeniem ?

- TOPIC MAPS daje możliwość uniwersalnego łączenia wiedzy w formie sieci informacji tworzących bazę wiedzy,
- TOPIC MAPS mają ogromne możliwości przy zachowaniu prostoty i elastyczności,
- TOPIC MAPS są krokiem w rozwoju idei XML, ukierunkowując systemy przechowujące informacje na Internet,
- TOPIC MAPS mogą być tworzone nad różnymi źródłami informacji (HTML, SGML, XML, ASCII, MSWORD, PDF, EMAIL, a nawet nad bazami danych).

Bibliografia

1. Z. Królikowski, M. Zakrzewicz, Środowisko rozwoju aplikacji internetowych Oracle Web Application Server + Pl/SQL Cartridge, w: PC Kurier, Nr 4/2000, str. 80-86, 2000.
2. Z. Królikowski, M. Zakrzewicz, Charakterystyka platform rozwoju aplikacji internetowych, Informatyka, Nr.5, str. 23-30, 2000 r.
3. http://php.zone.pl/.
4. www.webmacro.org.
5. www.netscape.com.
6. http://technet.oracle.com/.
7. http://java.sun.com/.
8. Jason Hunter, William Crawford, "JAVA Servlet Programming", O'REILLY, 1998.
9. Seth White, Maydane Fisher, Rick Cattel, Graham Hamilton, Mark Hapner, "JDBCTM API Tutorial and Reeference, Second Edition" , Sun Microsystems, 1999.
10. Dokument ISO/IEC-FCD 13250:1999-Topic Maps,
11. Hans Holger Rath, "Technical Issues on Topic Maps".
12. Hans Holger Rath, Steve Pepper, "Topic Maps at work".
13. www.infoloom.com.
14. www.step.pl.
15. www.cogsci.princeton.edu/~wn/ (implementacja Topic Maps).
16. www.quid.fr (implementacja Topic Maps).
17. www.wissen.de (implementacja Topic Maps).