XI Konferencja PLOUG
Kościelisko
Październik 2005

Technologie szablonów dla serwletów Java

Dariusz Aleksandrowicz, Marek Wojciechowski

Politechnika Poznańska, Instytut Informatyki

e–mail: marek@cs.put.poznan.pl

Streszczenie
Technologie szablonów dla serwletów Java stanowią alternatywę dla JSP w zakresie budowy interfejsu użytkownika w aplikacjach internetowych na platformie J2EE. Podobnie jak w przypadku JSP, motywacją dla ich powstania było umożliwienie separacji statycznych fragmentów dokumentów HTML i kodu Java dynamicznie generującego fragmenty zmienne. Celem niniejszego artykułu jest dokonanie przeglądu istniejących technologii szablonów oraz ocena ich możliwości i pozycji na rynku. Omówione i zilustrowane przykładem prostej aplikacji zostaną najpopularniejsze obecnie technologie szablonów: Velocity, FreeMarker i WebMacro.

1. Wprowadzenie

Podstawowym zadaniem aplikacji internetowych jest generacja dokumentów HTML stanowiących interfejs użytkownika, w odpowiedzi na żądanie przesłane przez przeglądarkę WWW. W przypadku aplikacji tworzonych w języku Java, początkowo to zadanie było realizowane przez serwlety [Servlets]. Programista implementujący serwlet musiał zawrzeć w nim instrukcje wysyłające do strumienia wyjściowego kolejne wiersze generowanego dokumentu HTML. Kod serwletu musiał zawierać instrukcje generujące zarówno zmienne fragmenty dokumentu HTML jak i te, które mają charakter statyczny, tzn. nie zmieniają się między kolejnymi odwołaniami do serwletu. Szybko zaobserwowano, że takie podejście utrudnia korzystanie z narzędzi do tworzenia dokumentów HTML i podział pracy między programistów Java i grafików – projektantów wizualnej strony generowanych dokumentów.

Jako odpowiedź na powyższą wadę serwletów, firma Sun opracowała technologię JavaServer Pages (JSP) [JSP]. Technologia JSP oparta została o ideę popularnej technologii ASP, zaproponowanej wcześniej przez Microsoft i cieszącej się dużą popularnością wśród twórców aplikacji internetowych. Idea polegała na zagnieżdżaniu fragmentów kodu Java w dokumentach HTML za pomocą specjalnych znaczników. Dodatkowo, zalecane było umieszczanie w miarę możliwości kodu Java w klasach pomocniczych i odwoływanie się do nich za pomocą specjalnych konstrukcji składniowych JSP. Technologia JSP od początku swojego istnienia została zaakceptowana przez dostawców serwerów J2EE i środowisk projektanckich jako standard do tworzenia interfejsu użytkownika w aplikacjach internetowych w języku Java. Należy podkreślić, że od strony technicznej JSP stanowi nowe wcielenie serwletów, gdyż dokumenty JSP przed pierwszym uruchomieniem są przekształcane na serwerze do odpowiadających im serwletów. Dzięki temu JSP zachowało wiele pozytywnych cech serwletów Java, które przyczyniły się do ich popularności.

Początkowe wersje specyfikacji JSP posiadały jednak szereg istotnych wad [Hun00], przez co spora grupa programistów nie chciała zrezygnować z serwletów. Po pierwsze, JSP początkowo nie pozwalały na całkowite usunięcie kodu Java z dokumentów, gdyż specyfikacja nie przewidywała znaczników do implementacji instrukcji warunkowych czy pętli, których często wymaga logika prezentacji. W takich przypadkach konieczne było wykorzystanie odpowiednich konstrukcji języka Java. Po drugie, konstrukcje składniowe służące do odwołania się do obiektów klas pomocniczych i ich właściwości były stosunkowo rozbudowane i mało czytelne. Do tego doszły prozaiczne, ale bardzo uciążliwe problemy z debugowaniem aplikacji JSP, gdyż sygnalizowane przez środowisko uruchomieniowe błędy odwoływały się do numerów linii wynikowego serwletu, a nie źródłowego dokumentu JSP.

Odpowiedzią na powyższe wady JSP było powstanie technologii szablonów dla serwletów Java. Technologie szablonów dla serwletów Java nie tylko w pełni umożliwiają, ale wręcz narzucają rozdział statycznych części generowanych dokumentów HTML i kodu Java. Koncepcja technologii szablonów polega na umieszczeniu statycznych fragmentów HTML w szablonie, w którym zaszyte są odwołania do obiektów Java udostępnionych przez serwlet za pomocą prostej notacji, wykorzystującej znak dolara. Języki tworzenia szablonów zostały wyposażone w konstrukcje programistyczne takie jak np. pętle, aby umożliwić implementację na poziomie szablonu złożonej logiki prezentacji bez posiłkowania się kodem Java.

Obecnie JSP nie posiada już większości wspomnianych wyżej wad, ze względu na wsparcie dla języka wyrażeń Expression Language (EL) i istnienie standardowej biblioteki znaczników JSTL Core [JSTL], umożliwiających odpowiednio wygodny sposób odwoływania się do obiektów Java i specyfikowanie pętli i warunków za pomocą znaczników JSP. W obowiązującym wzorcu projektowym model-view-controller (MVC) dla aplikacji J2EE, JSP stanowi podstawową technologię tworzenia widoków (ewentualnie w połączeniu z komponentami JavaServer Faces [JSF]), a czysta technologia serwletów jest wykorzystywana do implementacji kontrolera.

Technologia JSP dzięki swej ewolucji, statusowi specyfikacji firmy Sun i wsparciu komercyjnych dostawców oprogramowania, takich jak np. Oracle, skutecznie obroniła się przed technologiami szablonów skazując je na rolę technologii niszowych. Niemniej, technologie szablonów w dalszym ciągu są rozwijane i posiadają swoich sympatyków, którym nie odpowiada składnia JSP. Należy też podkreślić, że technologie szablonów są uwzględniane jako pełnoprawna alternatywa dla JSP w popularnych implementacjach wzorca projektowego MVC: Struts [Struts] i Spring Framework [Spring].

Celem niniejszego artykułu jest omówienie zasady działania technologii szablonów dla serwletów Java i przedstawienie oraz porównanie trzech najpopularniejszych technologii szablonów: Velocity [Vel], FreeMarker [FM] i WebMacro [WM].

2. Charakterystyka i zasada działania technologii szablonów

Generatory szablonów służą do generowania wszelkiego rodzaju dokumentów tekstowych z wcześniej przygotowanych szablonów. Najczęściej stosuje się je do generowania dokumentów HTML, ale można je stosować również do generowania dokumentów XML, RTF, WML, wiadomości e-mail, sms, kodów źródłowych programów lub plików konfiguracyjnych.

Podstawową zaletą stosowania szablonów jest możliwość pełnego odseparowania zawartości dokumentu od warstwy znaczników opisujących jego formę. Projektant serwisu internetowego odwołuje się do metod zdefiniowanych w kodzie języka Java. Programiści Java mogą skupić się na tworzeniu efektywnego i elastycznego kodu. Projektanci mogą w tym czasie skupić się na stronie wizualnej serwisu. Oba zespoły mogą pracować niezależnie i równolegle. Część wizualna może powstawać nawet jeśli treść, która ją wypełni, jeszcze nie istnieje.

Kolejną cechą, jaką zauważą projektanci wykorzystujący technologię szablonów, jest czytelność wprowadzanego kodu. Nie miesza się on z kodem programu, jak ma to miejsce w innych technologiach. Zdecydowanie ułatwia to modyfikację wyglądu serwisu oraz sprawia, że użytkownicy popełniają mniej błędów w czasie tego procesu. Łatwiejsze jest też zarządzanie serwisem, gdyż zmiana koncepcji wizualnej strony nie wymaga angażowania programistów. Analogicznie zmiana treści dokumentów serwisu lub sposobu dostępu do danych nie zmusza do angażowania ludzi odpowiedzialnych za wygląd strony.

Na rynku technologii generatorów szablonów dla serwletów Java istnieje trzech liczących się producentów. Najbardziej znanym jest grupa Apache Jakarta Project (http://jakarta.apache.org/), która w zakresie technologii szablonów tworzy i rozwija technologię **Velocity** (http://jakarta.apache.org/velocity/). Kolejnym producentem, stanowiącym poważną konkurencję dla Velocity, jest hiszpańskie stowarzyszenie Visigoth Software Society, zajmujące się technologią **FreeMarker** (http://freemarker.sourceforge.net/). Ostatnim autorem liczącej się technologii szablonów jest grupa pod przewodnictwem Justina Wellsa, zajmująca się rozwojem technologii **WebMacro** (http://www.webmacro.org/).

Znamienne jest, że wszystkie wspomniane wyżej technologie dostępne są na licencjach opensource. Każda jest na bieżąco rozwijana, a ich twórcy deklarują pomoc, za pośrednictwem poczty elektronicznej, w przypadku kłopotów. Podstawowym źródłem informacji na temat technologii szablonów są serwisy internetowe producentów. Obszerna dokumentacja dostępna w każdym z serwisów jak również forum dyskusyjne dla użytkowników systemu pozwalają na dogłębne zapoznanie się z technologiami oraz wymianę doświadczeń pozwalającą rozwiązać ewentualne problemy w eksploatacji.

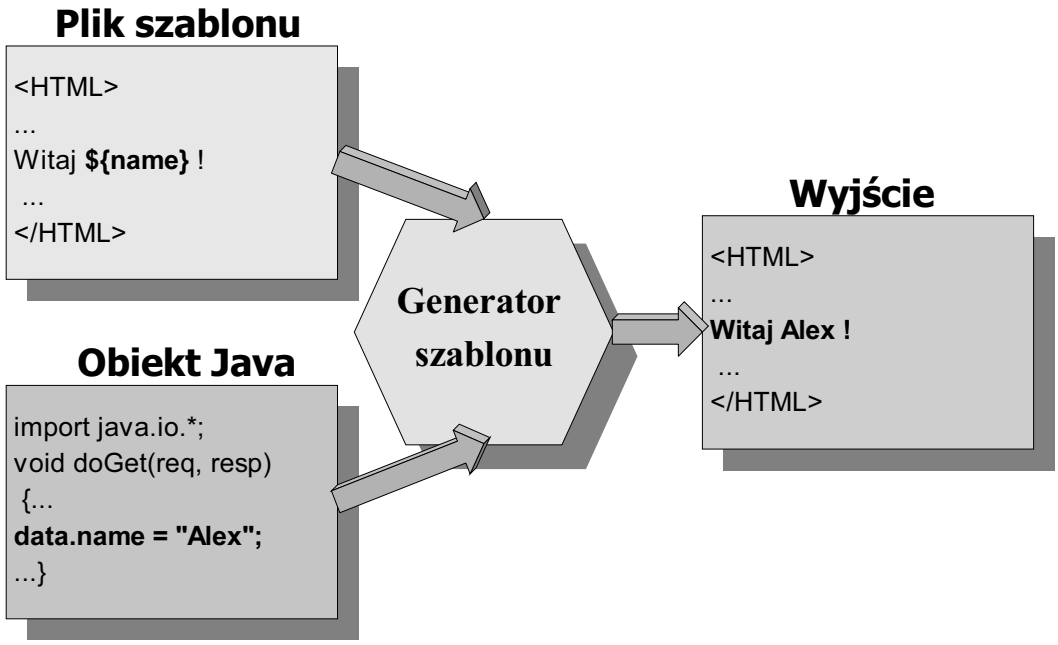
Wszystkie omówione technologie oferują podobną składnię języka tworzenia szablonów i przyjęły podobny sposób podziału elementów konstrukcji składniowych. Zasadniczo każda z nich ma do dyspozycji **zmienne**, które mogą posiadać swoje **własności** a nawet **metody**. Wynika to z faktu, że z punktu widzenia programistów tworzących te zmienne, są one w istocie obiektami

Java, a te mogą posiadać własności i metody. Większość technologii pozwala przechowywać w zmiennych łańcuchy znaków lub wartości liczbowe całkowite.

Drugim elementem składniowym stosowanym przez wszystkie technologie są **dyrektywy**. Ich zadaniem jest dostarczenie podstawowej funkcjonalności, najczęściej stosowanej przy przetwarzaniu danych, jak na przykład pętle, funkcje warunkowe czy importujące zawartość zbiorów zewnętrznych. Każda z technologii pozwala definiować własne dyrektywy zawierające dowolne konstrukcje składniowe języka szablonu. Dyrektywy zdefiniowane przez użytkownika nazywane są **makrowyrażeniami** (ang. *macro*).

Podstawowym zadaniem generatora szablonów jest dokonanie połączenia danych udostępnianych przez kasy Java z szablonem, określającym sposób ich wykorzystania. Szablon powinien posiadać referencje do zmiennych oraz minimalną ilość dyrektyw tak, by projektant mógł skupić się na kodzie HTML celem prezentacji dostarczonych mu danych, a nie przetwarzaniem ich. Jak największa część operacji przetwarzania danych, obliczania wartości końcowych, powinna być po stronie programisty. Projektant powinien otrzymywać gotowe do prezentacji, w pełni przetworzone dane. Dlatego istotne jest stworzenie odpowiedniego modelu danych, realizacja tego modelu w postaci struktur implementowanych w kodzie Java oraz po odpowiednim przetworzeniu uzyskanych danych, umieszczenie ich w tych strukturach celem przekazania do szablonu. Projektantowi pozostaje już tylko umieszczenie dostarczonych danych w odpowiednich miejscach.

W tym kontekście, zadaniem generatora jest połączenie informacji modelu danych i ich miejscami przeznaczenia opisanymi w szablonie. Sposób realizacji tego zadania ilustruje poniższy rysunek:



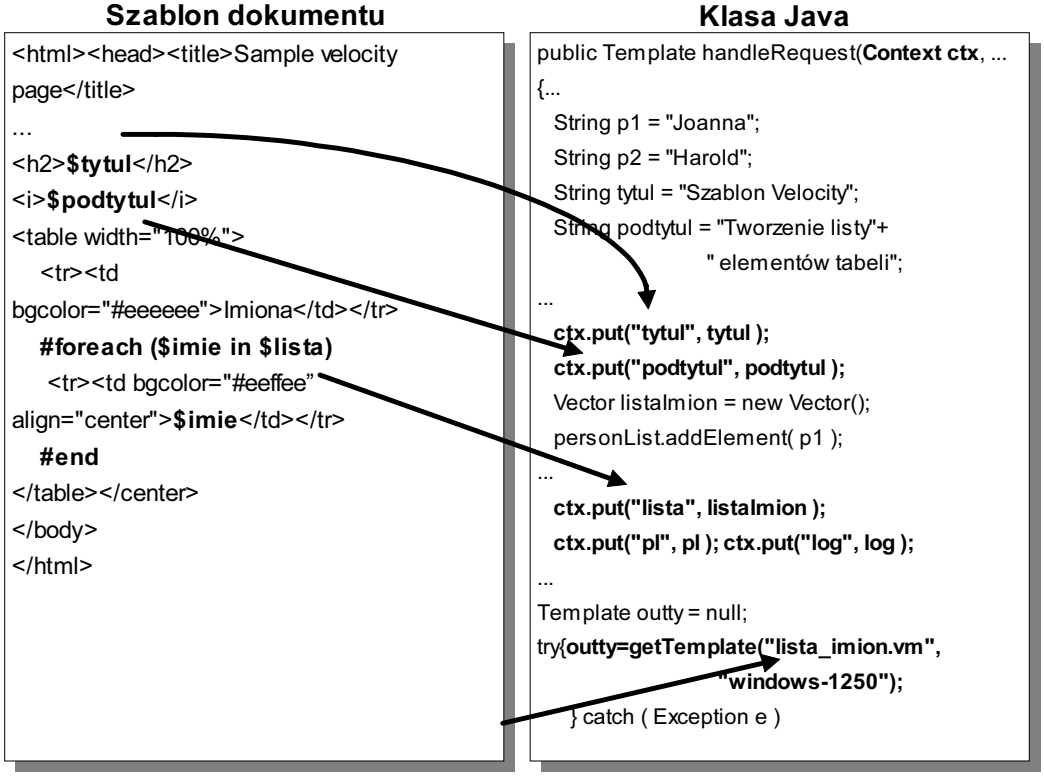
Rys. 1. Schemat łączenia danych i szablonu

Klasa Java przedstawiona na powyższym schemacie posiada obiekt *data*, który posiada własność *name*. Własność ta otrzymuje wartość w postaci łańcucha znaków „Alex”. Plik szablonu natomiast posiada referencję do tej własności. Generator szablonu łączy te dwie struktury poprzez wstawienie odpowiednich danych w miejsce wystąpienia referencji do tych danych. W ten sposób powstaje dokument wyjściowy.

Generatory szablonów są dostępne w postaci bibliotek klas języka Java. W celu ich użycia nie jest wymagana żadna szczególna procedura instalacyjny. Archiwum JAR zawierające klasy danego generatora szablonów należy umieścić na ścieżce CLASSPATH lub skopiować do odpowiedniego podkatalogu w drzewie katalogów serwera J2EE.

Generatory szablonów mogą być w zasadzie w ten sam sposób wykorzystane w różnych typach aplikacji tworzonych w języku Java. Jednakże ze względu na fakt, że najczęściej wykorzystywane są one w serwletach, wszystkie trzy omawiane technologie szablonów oferują wzorcową klasę serwletu znacząco upraszczającą korzystanie z szablonów. W przypadku Velocity jest to klasa *VelocityServlet*, dla FreeMarker *FreeMarkerServlet*, a dla WebMacro *WMServlet*. Programiści aplikacji mogą tworzyć serwlety jako klasy dziedziczące z tych specjalizowanych klas serwletów lub rozszerzyć standardową klasę *HttpServlet* i odwoływać się do generatora szablonów w sposób uniwersalny.

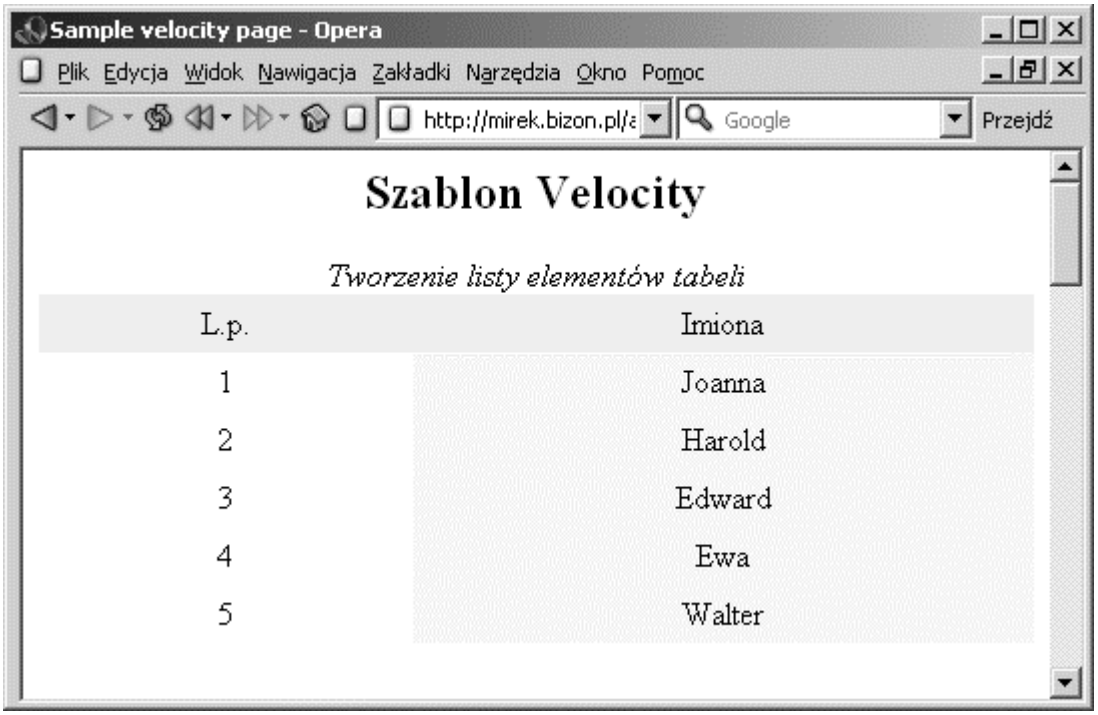
Wykorzystanie specjalizowanych klas serwletów oferowanych przez technologie szablonów sprowadza zadanie programisty do zdefiniowania jednej metody obsługującej żądanie. Przykładowo, w kontekście technologii Velocity i dostarczanej przez nią wzorcowej klasy serwletu *VelocityServlet* jest to metoda *handleRequest*, której jednym z parametrów jest kontekst (obiekt klasy *Context*), w którym definiowane będą obiekty, do których odwołuje się szablon. Obiekty są „umieszczane" w kontekście za pomocą metody *put*. Poniższy rysunek ilustruje powiązanie pomiędzy referencjami w szablonie i metodami w kodzie Java, które udostępniają obiekty odpowiadające tym referencjom, na przykładzie technologii Velocity:



Rys.2. Sposób powiązania referencji i zmiennych

W powyższym przykładzie, do zawartości kontekstu szablonu, dodane są powiązania z obiektami *tutul*, *podtytul*, oraz *listaImion* klasy *Vector*, który zawiera obiekty klasy *String*, zawierające imiona. Dzięki użyciu metody *put* klasy *Context*, wewnątrz szablonu można stosować referencje *$tytul*, *$podtytul*, oraz *$lista*. Referencja *$imie*, jest referencją lokalną, z punktu widzenia szablonu, związaną ze sposobem działania dyrektywy *#foreach*. Gdy generator szablonu połą-

czy zawartość szablonu z wartościami obiektów klas Java, wygeneruje dokument HTML, który wyświetlony na ekranie przeglądarki internetowej będzie wyglądał tak, jak to przedstawiono na Rys.3.



Rys.3. Efekt działania kodu przedstawionego na Rys.2

Generatory szablonów nie wymagają żadnego szczególnego środowiska uruchomieniowego. Oczywiście dla konkretnej wersji Velocity, FreeMarker, czy WebMacro dokumentacja podaje wymaganą wersję J2SE oraz minimalną wersję specyfikacji serwletów, którą musi wspierać wykorzystywany kontener serwletów, co jest istotne w przypadku typowego zastosowania technologii szablonów jakim są serwlety. Ogólnie mówiąc, wszystkie rozwijane aktualnie kontenery serwletów (np. Apache Tomcat) bez problemu współpracują z generatorami szablonów.

3. Velocity

Podstawowym celem, jaki przyświecał autorom Velocity, było udostępnienie tej technologii użytkownikom do dalszej implementacji w różnych zastosowaniach. Velocity służy, na przykład, jako wtyczka (ang. *plug-in*) w edytorze programistycznym jEdit (http://www.jedit.org), pozwalająca definiować szablony standardowych funkcji dowolnych języków programowania lub często wykorzystywanych konstrukcji składniowych. Velocity może też służyć jako rdzeń samodzielnej aplikacji zajmującej się, na przykład, generowaniem korespondencji seryjnej, faktur dla klientów i wielu innych systemów. Znaczącą część zastosowań Velocity stanowi jednak współpraca z technologią serwletów Java.

Język szablonów Velocity (ang. Velocity Template Language – VTL) stanowi zbiór niezbędnych konstrukcji pozwalających wykonywać typowe operacje na danych dostarczonych przez serwlety. Autorzy systemu celowo nie poszerzają zbioru instrukcji dostępnych w ramach języka szablonów. Zachowują tym samym przejrzystość tego języka dając jednocześnie użytkownikowi możliwość implementacji własnych rozwiązań, lepiej dostosowanych do potrzeb jego systemu.

W skład konstrukcji składniowych Velocity wchodzą przede wszystkim dyrektywy. Pozwalają one na konstruowanie klasycznych pętli na elementach list, warunkowe włączanie określonego fragmentu kodu, importowanie danych ze zbiorów zewnętrznych, tworzenie makrowyrażeń powtarzających się fragmentów kodu, a nawet tworzenie własnych zmiennych, w ramach szablonu.

Podstawowym źródłem danych dla projektanta serwisu internetowego tworzącego szablon Velocity, są obiekty udostępniane przez serwlet, przygotowane przez programistę Java. Projektant serwisu odwołuje się do nich poprzez **referencje**. O tym jaki typ danych przechowuje obiekt, decyduje programista Java. Od tego właśnie typu zależy jakiego rodzaju referencji do obiektu może użyć projektant. Velocity przewiduje referencje do **zmiennych**, do **własności** obiektów oraz do **metod** obiektów. Projektant ma możliwość odwoływania się do dowolnych elementów przygotowanych obiektów i wstawiania ich we właściwych miejscach szablonu, pomiędzy znacznikami HTML i tekstem, znajdującymi się w szablonie. Istnieje również możliwość modyfikacji udostępnionych zmiennych i własności obiektów.

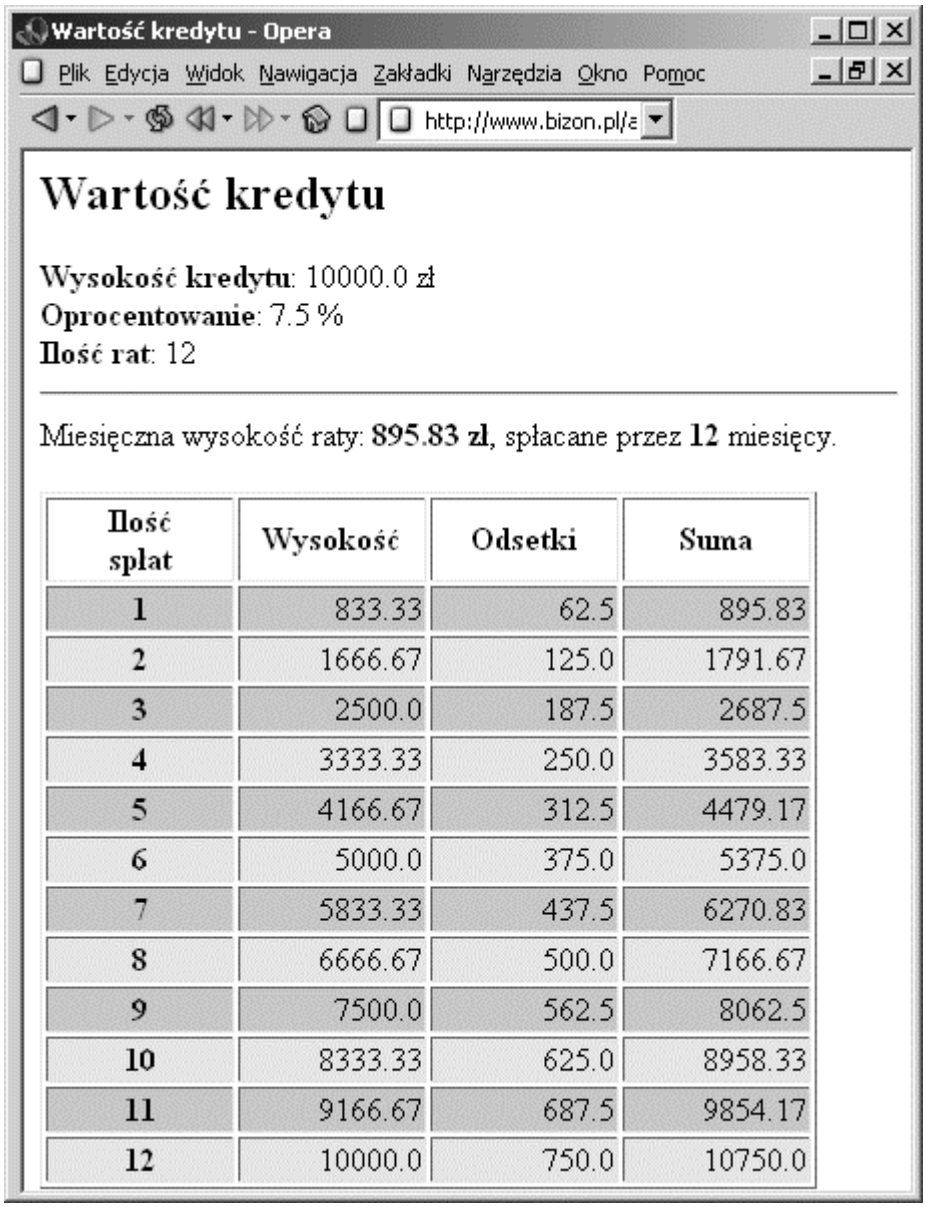
Metody zdefiniowane po stronie serwletu i przekazane do szablonu jako obiekty wraz z własnościami innych obiektów oraz zmiennymi, dają programistom możliwość stworzenia dowolnego modelu danych, który przekazany do szablonu pozwala projektantom serwisu internetowego na stworzenie dynamicznych dokumentów składających się na profesjonalną witrynę internetową.

Poniższy przykład prezentuje zastosowanie składni języka szablonów Velocity. Dla ułatwienia, elementy składni języka Velocity zostały wyróżnione pogrubioną czcionką. Przedstawiony kod szablonu stanowi prostą aplikację, która na podstawie parametrów wywołania serwletu oblicza i prezentuje koszty, jakie poniesie kredytobiorca, po upływie kolejnych okresów spłaty kredytu, wraz z odsetkami. Użytkownik podaje wysokość kredytu, oprocentowanie oraz ilość rat spłaty kredytu. Cały proces obliczania poszczególnych wartości został przeniesiony do serwletu, który przekazuje do szablonu jedynie wyniki obliczeń, gotowe do prezentacji w formie, którą projektant uzna za najbardziej odpowiednią. Nie musi on skupiać się na algorytmie obliczania kosztów spłaty kredytu. Używa jedynie referencji do wyników obliczeń.

```
#set ($tytul = "Wartość kredytu" )
<html><head><title>$tytul</title></head>
<body bgcolor="#FFFFFF" text="#000000">
<h2>$tytul</h2>
<b>Wysokość kredytu</b>: $kredyt zł<br>
<b>Oprocentowanie</b>:   $procent %<br>
<b>Ilość rat</b>:        $ile_rat<br>
<hr align="center" width="100%">
Miesięczna wysokość raty: <b>$raty.get(0) zł</b>,
spłacane przez <b>$ile_rat</b> miesięcy.<br><br>
<table border="1" align="left" bordercolor="#000000">
  <tr align="center" valign="middle">
  #foreach ( $element in $naglowek )
    <th width="90">$element</th>
  #end
  </tr>
  #set ( $jaki_kolor = 1 )
  #foreach ( $element in [1..$ile_rat])
    #if ( $jaki_kolor == 1)
```

```
      #set ( $kolor = "#66FFAA" )
      #set ( $jaki_kolor = 0 )
    #else
      #set ( $kolor = "#AAFFFF" )
      #set ( $jaki_kolor = 1)
    #end
  <tr align="right" valign="middle" bgcolor="$kolor">
    <th align="center" width="90">$element</th>
    <td width="90">$wysokosci.get($velocityCount)</td>
    <td width="90">$odsetki.get($velocityCount)</td>
    <td width="90">$raty.get($velocityCount)</td>
  </tr>
  #end
</table></BODY></HTML>
```

Efekt działania powyższego szablonu (w połączeniu z serwletem dokonującym obliczeń i odwołującym się do szablonu) przedstawia poniższy rysunek.



Rys.4. Efekt działania przykładowego szablonu aplikacji prezentującej koszty kredytu

Dokumentacja technologii Velocity podzielona jest na części, przeznaczone dla różnych członków zespołów zajmujących się prowadzeniem serwisu internetowego opartego na tej technologii. „Przewodnik użytkownika" od podstaw zapoznaje projektantów serwisów z możliwościami wykorzystania systemu, oraz całą filozofią podziału zadań między członkami zespołów. Przedstawione są skrótowo przykładowe zastosowania podstawowych mechanizmów dostępnych w systemie z punktu widzenia projektanta zajmującego się wizualną stroną serwisu internetowego. Osobną część stanowi „przewodnik programisty", przeznaczony dla programistów Java, którzy będą mieli za zadanie tworzenie klas współpracujących z szablonami. Ponadto istnieje podręcznik języka szablonów Velocity, który szczegółowo przedstawia składnię i przykładowe użycie wszystkich elementów składniowych języka. Cała dokumentacja napisana jest w sposób spójny i przystępny. Wprowadzony podział tematyczny pozwala łatwo dotrzeć do potrzebnej informacji w kontekście swoich zadań przy wykorzystaniu technologii.

4. FreeMarker

Autorzy technologii FreeMarker tworząc swój system, przyjęli odwrotną strategię niż twórcy technologii Velocity. Dostarczają oni swoim odbiorcom wiele gotowych rozwiązań mogących ułatwić przetwarzanie danych, typowe dla tworzenia dynamicznych dokumentów hipertekstowych.

FreeMarker posiada w zakresie swojej funkcjonalności bardzo bogatą pulę dyrektyw wraz z wieloma opcjami. Poza podstawowymi dyrektywami są też takie, które pozwalają tworzyć konstrukcje charakterystyczne dla klasycznych języków programowania. FreeMarker pozwala definiować własne funkcje i dyrektywy, deklarować własne zmienne o różnym zakresie widzialności oraz znacząco rozszerza ilość typów danych i obiektów dostępnych w szablonie. Jednocześnie oferuje kilkadziesiąt funkcji wbudowanych, pozwalających zaspokoić praktycznie wszystkie potrzeby związane z przetwarzaniem przeznaczonych do prezentacji danych.

FreeMarker jasno precyzuje typy i struktury danych, jakimi będzie się posługiwał projektant serwisu internetowego tworząc szablon. Wyróżnia **zmienne skalarne**, przeznaczone do przechowywania pojedynczych danych, **hasze**, **sekwencje** oraz **kolekcje**. Trzy ostatnie struktury stanowią swego rodzaju kontenery zawierające inne obiekty. Różnica między nimi polega na sposobie dostępu do tych obiektów, który może być realizowany poprzez podanie łańcucha znaków skojarzonego z danym obiektem bądź poprzez podanie numeru indeksu żądanego obiektu. Każdy z tych typów pozwala na dowolne zagnieżdżanie innych obiektów w podelementach tak, że powstaje swego rodzaju struktura drzewiasta, charakterystyczna dla obiektowych struktur danych.

Projektant, mający dane zawarte we wspomnianych obiektach, umieszcza je w szablonie poprzez tak zwane **interpolacje**, czyli specjalne znaczniki zawierające odwołania do obiektów. Są to tak naprawdę referencje, analogiczne do stosowanych w innych technologiach.

Doświadczony projektant z pewnym doświadczeniem programistycznym z dużym powodzeniem wykorzysta możliwości zawarte w technologii FreeMarker, lecz początkujący użytkownik może poczuć się trochę zagubiony w dużej liczbie opcji oferowanych przez tę technologię. Zawsze jednak może się ograniczyć do niezbędnego minimum funkcjonalności.

Przedstawiony poniżej kod szablonu stanowi tę samą aplikację, która została wykorzystana wcześniej do zilustrowania składni Velocity. Tym razem aplikacja powstała przy użyciu języka szablonów FreeMarker. Zwraca uwagę duże podobieństwo składni FreeMarker i Velocity. Gdyby projektant serwisu, posługujący się już językiem szablonów Velocity, miał zacząć tworzyć szablony w oparciu o technologię FreeMarker, nie miałby z tym dużo problemów.

```
<#assign tytul = "Wartość kredytu" >
<html><head><title>${tytul}</title></head>
<body bgcolor="#FFFFFF" text="#000000">
<h2>${tytul}</h2>
<b>Wysokość kredytu</b>: ${kredyt} zł<br>
<b>Oprocentowanie</b>:   ${procent} %<br>
<b>Ilość rat</b>:        ${ile_rat}<br>
<hr align="center" width="100%">
Miesięczna wysokość raty: <b>${raty.get(0)} zł</b>,
spłacane przez <b>${ile_rat}</b> miesięcy.<br><br>
<table border="1" align="left" bordercolor="#000000">
  <tr align="center" valign="middle">
```

```
  <#list naglowek as element >
    <th width="90">${element}</th>
  </#list>
  </tr>
  <#assign jaki_kolor = 1 >
  <#list 1..ile_rat as element >
    <#if ${jaki_kolor} == 1 >
      <#assign kolor = "#66FFAA" >
      <#assign jaki_kolor = 0 >
    <#else>
      <#assign kolor = "#AAFFFF" >
      <#assign jaki_kolor = 1 >
    </#if>
  <tr align="right" valign="middle" bgcolor="${kolor}">
    <th align="center" width="90">${element}</th>
    <td width="90">${wysokosci.get(${element_index})}</td>
    <td width="90">${odsetki.get(${element_index})}</td>
    <td width="90">${raty.get(${element_index})}</td>
  </tr>
  </#list>
</table></BODY></HTML>
```

Dokumentacja technologii FreeMarker również podzielona jest na części związane z podziałem ról członków zespołu. Zawartość całego podręcznika dostępna jest z jednego menu ułożonego hierarchicznie. Daje to wrażenie uporządkowania oraz możliwość zorientowania się w zakresie danej tematyki już na pierwszy rzut oka. Również i tu dostępny jest podręcznik języka szablonów z opisem wszystkich elementów składniowych i krótkimi przykładami. Na uwagę zasługuje fakt, że w całej dokumentacji technologii FreeMarker autorzy stosują kolorowanie istotnych merytorycznie elementów tekstu, celem ich wyróżnienia. Powoduje to, że zarówno prezentacja technik wykorzystania technologii jak i sam podręcznik języka są bardziej czytelne i zrozumiałe niż dokumentacja Velocity. Z drugiej strony autorzy systemu niekiedy zbyt ogólnie ujmują niektóre zagadnienia, zwłaszcza odnośnie konfiguracji generatora szablonu. Widać iż niektóre aspekty uważają za oczywiste i nie warte wyjaśnienia, co znacząco utrudnia uruchomienie systemu i poznanie podstawowych mechanizmów w nim obowiązujących.

5. WebMacro

Technologia WebMacro posiada konstrukcję odpowiednią do zastosowań związanych z tworzeniem serwisów internetowych. Posiada umiarkowaną ilość dyrektyw, których funkcjonalność dobrze nadaje się do tego celu. Składnia języka szablonów WebMacro jest bardzo podobna do składni Velocity. Dotyczy to również nazewnictwa oraz funkcjonalności dyrektyw dostępnych w tym języku szablonów. Wynika to z faktu, iż Velocity powstało jako alternatywa w stosunku do WebMacro, o mniej restrykcyjnej licencji, zachowująca jego funkcjonalność. Z czasem jednak te dwie technologie rozeszły się i w aktualnych wersjach w niektórych aspektach składnia WebMacro różni się od Velocity. Widać to wyraźnie na przykładzie sposobu oznaczania bloków kodu szablonu, które określane są przy pomocy nawiasów klamrowych, podobnie jak w języku Java.

Projektant tworzący szablon WebMacro, ma do dyspozycji zestaw dyrektyw posiadających funkcjonalność odpowiednią do prezentacji danych w dokumentach hipertekstowych. Niektóre dyrektywy zostały zaprojektowane specjalnie pod kątem współpracy z procedurami dynamicznie tworzącymi tabele w formacie HTML. Inna dyrektywa pozwala kontrolować poprawność typów obiektów przekazywanych przez serwlet do wnętrza szablonu. Jest to więc typowa technologia przeznaczona do tworzenia serwisów internetowych, wymagająca od projektanta pewnego doświadczenia w tym zakresie.

Podobnie jak we wcześniej omawianych technologiach, język szablonów WebMacro pozwala uzyskiwać dostęp do zmiennych, własności obiektów oraz metod obiektów, przekazanych do wnętrza szablonu przez serwlety Java. Istnieje również możliwość tworzenia własnych zmiennych oraz modyfikacji już istniejących. Sposób użycia zmiennych i metod jak również terminologia je opisująca jest w zasadzie tożsama ze stosowaną przy opisie systemu Velocity.

Poniższy przykład jest implementacją aplikacji prezentującej koszty kredytu w języku szablonów WebMacro. Podobieństwo składni do Velocity jest jeszcze większe, niż do FreeMarker. Wynika to z faktu, że Velocity jest następcą technologii WebMacro.

```
#set $tytul = "Wartość kredytu"
<html><head><title>$tytul</title></head>
<body bgcolor="#FFFFFF" text="#000000">
<h2>$tytul</h2>
<b>Wysokość kredytu</b>: $kredyt zł<br>
<b>Oprocentowanie</b>:   $procent %<br>
<b>Ilość rat</b>:        $ile_rat<br>
<hr align="center" width="100%">
Miesięczna wysokość raty: <b>$raty.get(0) zł</b>,
spłacane przez <b>$(ile_rat)</b> miesięcy.<br><br>
<table border="1" align="left" bordercolor="#000000">
  <tr align="center" valign="middle">
  #foreach $element in $naglowek
  {
    <th width="90">$element</th>
  }
  </tr>
  #set $jaki_kolor = 1
  #count $i from 1 to $ile_rat step 1
  {
    #if ( $jaki_kolor == 1) {
      #set $kolor = "#66FFAA"
      #set $jaki_kolor = 0
    } #else {
      #set $kolor = "#AAFFFF"
      #set $jaki_kolor = 1
    }
  <tr align="right" valign="middle" bgcolor="$(kolor)">
```

```
    <th align="center" width="90">$element</th>
    <td width="90">$wysokosci.get($i)</td>
    <td width="90">$odsetki.get($i)</td>
    <td width="90">$raty.get($i)</td>
  </tr>
  }
</table></BODY></HTML>
```

Dokumentacja technologii WebMacro jest najgorzej zorganizowana spośród wszystkich omawianych technologii szablonów. Trudno się w niej dopatrzyć ogólnego schematu czy hierarchii. Nie ma też wyraźnego podziału tematycznego, który pozwoliłby projektantom czy programistom Java odnaleźć interesujące ich zagadnienia. Ponadto wiele jest miejsc zawierających merytoryczne błędy, na przykład brak zgodności definicji elementu składni z jego zastosowaniem w poniższym przykładzie. Niektóre przykłady zawierają opcje, które nie zostały wyjaśnione w podręczniku języka. Generalnie, dokumentacja technologii WebMacro nie zachęca do korzystania z tego systemu.

6. Porównanie technologii szablonów

Ze względu na ten sam cel, jaki mają do spełnienia, przedstawione technologie cechuje duże podobieństwo. Jednym z założeń technologii szablonów jest dostarczenie możliwie prostego języka dla projektantów serwisów internetowych, ponieważ przyjmuje się, że mogą oni nie posiadać doświadczenia programistycznego. Ten fakt dodatkowo ogranicza spektrum sposobów implementacji i rozwiązań stosowanych w trakcie tworzenia języka szablonów. W efekcie dostępne technologie różnią się od siebie w dużej mierze z powodu osobistych preferencji jej twórców.

Inne powody dużych podobieństw opisanych technologii są dosyć prozaiczne. Autorzy technologii Velocity na stronach serwisu internetowego ich produktu wyjaśniają przyczyny, dla których składnia ich języka szablonów jest identyczna ze składnią systemu WebMacro. Ich zdaniem po prostu „nie ma sensu na nowo wynajdywać koła", a poza tym stosując tę samą składnię co konkurencja, w łatwy sposób można dostarczyć użytkownikom narzędzi do konwersji szablonów konkurencji na własny format. Zasada ta jednak działa w obie strony. Autorzy technologii FreeMarker dostarczają narzędzi do konwersji szablonów i klas Velocity na ich format.

Pomimo oczywistych podobieństw, są jednak pewne aspekty technologii szablonów, które różnią je w znaczący sposób. Podstawową cechą systemu Velocity jest jego prostota. Jego twórcy dbają o to, aby zbiór funkcji i dyrektyw dostępnych w języku szablonów nie powiększał się w kolejnych wersjach systemu i zawierał tylko te, które są niezbędne. Powoduje to przejrzystość składni języka oraz łatwość, z jaką mogą się nim posługiwać projektanci. Dyrektywy Velocity nie posiadają całej gamy opcji. Brak jest też funkcji dedykowanych pod konkretne zastosowania, jak ma to miejsce w technologii WebMacro. W tej ostatniej dostępne są na przykład dyrektywy wspierające tworzenie tabel w języku HTML. Velocity pozwala za to na stworzenie własnych funkcji i dyrektyw rozszerzających funkcjonalność, lepiej dostosowanych do potrzeb jego użytkowników. Dlatego technologia ta stanowi bazę do wielu innych zastosowań, wykraczających poza współpracę z serwletami Java.

Metody i klasy, wchodzące w skład wewnętrznej struktury technologii WebMacro, w wielu przypadkach zadeklarowane są jako prywatne lub finalne (ang. *private*, *final*). Ma to na celu uzyskanie lepszej wydajności działania generatora szablonu. Powoduje to jednak brak możliwości rozszerzania oraz dalszej implementacji klas wewnętrznych. Użytkownicy nie mają więc możliwości tak łatwego dostosowywania do swoich potrzeb oraz znajdywania nowych zastosowań sys-

temu WebMacro, jak ma to miejsce w technologii Velocity. Być może ten fakt zdecydował o wyraźnej dominacji tej ostatniej technologii na rynku generatorów szablonów.

Autorzy technologii FreeMarker przyjęli strategię wszechstronności. Ich system posiada bogatą gamę dyrektyw i ich opcji, funkcji oraz dostarcza znacznie więcej typów danych. Velocity oraz WebMacro obsługują, na przykład, wartości numeryczne jedynie typu całkowitego (integer). FreeMarker natomiast przetwarza liczby zmiennoprzecinkowe, jednocześnie dając użytkownikowi możliwość określenia ilości miejsc po przecinku. Dostępny jest również typ daty. Oba typy danych przy wyświetlaniu na ekranie, prezentowane są w formacie zgodnym z formatem stosowanym w kraju użytkownika. Kraj ten określany jest przez system FreeMarker na podstawie deklaracji kodowania znaków narodowych dla przetwarzanego szablonu. Co ciekawe, nawet zmienne w kodzie szablonu mogą zawierać dowolne znaki narodowe. Ta cecha wyróżnia technologię FreeMarker spośród pozostałych systemów.

Technologią, która jest najbardziej tolerancyjna na błędy w składni szablonu jest Velocity. W tym systemie przyjęto zasadę, że jeśli jakaś konstrukcja nie jest zrozumiała dla systemu, to jest ona traktowana jako tekst, który powinien się pojawić na wyjściu po przetworzeniu szablonu. Takie zachowanie może doprowadzić do przeoczenia błędów gdyż niektóre konstrukcje składniowe mogą zginąć w gąszczu słów zwykłego tekstu. Najbardziej restrykcyjny pod względem kontroli poprawności jest FreeMarker. Każdy błąd w jego szablonie jest sygnalizowany w oknie przeglądarki komunikatem błędu.

Technologia FreeMarker, w stosunku do pozostałych technologii, najbardziej przypomina klasyczny język programowania. Dodatkowe opcje przy wykorzystaniu pętli czy możliwości wykonywania obliczeń lub tworzenia wyrażeń nie odbiegają od tych dostępnych programistom serwletów. Definicje makrowyrażeń mogą być umieszczane w dowolnym miejscu szablonu, a nie przed ich pierwszym wywołaniem, analogicznie do definicji funkcji w językach programowania. Velocity, dla porównania, wymaga od projektantów definiowania wszystkich makrowyrażeń przed miejscem ich wywołania.

Nieocenioną cechą systemu FreeMarker są przestrzenie nazw dla zmiennych. Pozwalają one na przykład korzystać z bibliotek zmiennych lub makrowyrażeń bez obawy o jednakowe ich nazwy. Programiści je tworzący nie muszą się umawiać co do nazewnictwa, gdyż każda biblioteka może się znajdować w odrębnej przestrzeni.

Liczba wbudowanych funkcji przeznaczonych do przetwarzania łańcuchów znaków czy bezpośrednio związanych ze wspomaganiem przy tworzeniu dokumentów HTML jest w przypadku technologii FreeMarker większa nawet w stosunku do systemu WebMacro. To bogactwo czyni system FreeMarker najbardziej wszechstronnym, ale jednocześnie najbardziej skomplikowanym. Do jego pełnego wykorzystania potrzebny jest doświadczony programista. Ponieważ jednak podstawowym odbiorcą technologii szablonów miał być z założenia projektant-artysta, który nie posiada żadnego doświadczenia programistycznego, można by się zastanawiać do kogo jest skierowana ta technologia. Oczywiście funkcjonalność systemu FreeMarker można uprościć poprzez stosowanie tylko niezbędnych dyrektyw i zmiennych. Jednak wyraźnie widać, że technologia ta jest raczej skierowana do programistów, którzy próbują swych sił na polu projektowania serwisów internetowych lub dla tych programistów, którzy z powodu oszczędności w firmach, zmuszeni są zajmować się zarówno programowaniem jak i projektowaniem serwisów.

7. Podsumowanie

Technologie szablonów dla serwletów Java umożliwiają i narzucają rozdział statycznych części generowanych dokumentów HTML i kodu Java. Ich powstanie stanowiło odpowiedź na niedoskonałości wczesnych wersji technologii JSP, której geneza powstania była taka sama. Pierwotne wersje specyfikacji JSP oferowały relatywnie złożoną składnię dostępu do obiektów Java, a do tego nie przewidywały konstrukcji typu pętle czy instrukcje warunkowe, przez co w wielu przypadkach konieczne było użycie w dokumencie fragmentów kodu w języku Java.

Aktualna wersja specyfikacji JSP nie posiada już wspomnianych wyżej wad ze względu na wsparcie dla języka wyrażeń Expression Language i istnienie biblioteki znaczników JSTL Core. Ponadto, od początku swojego istnienia JSP jako technologia opracowana przez firmę Sun jest wspierana przez komercyjnych dostawców oprogramowania takich jak np. Oracle. Dlatego też, JSP powinno być traktowane jako domyślny i naturalny wybór w zakresie prezentacji danych w aplikacjach J2EE. Niemniej, technologie szablonów są nadal rozwijane i wykorzystywane, gdyż koncepcja i składnia JSP nie wszystkim projektantom aplikacji J2EE odpowiada. Co więcej, technologie szablonów są uwzględniane jako pełnoprawna alternatywa dla JSP w popularnych implementacjach obowiązującego obecnie wzorca projektowego MVC (model-view-controller) Struts i Spring Framework.

Najpopularniejsze obecnie technologie szablonów to Velocity, FreeMarker i WebMacro. Technologia Velocity jest liderem w zakresie prostoty tworzenia części wizualnej serwisu, natomiast FreeMarker – w zakresie funkcjonalności technologii szablonów. System WebMacro stanowi pewną wypadkową pomiędzy pierwszą i drugą technologią.

Bibliografia

[FM] FreeMarker – The FreeMarker Project, http://freemarker.sourceforge.net/

[Hun00] Hunter J.: The Problems with JSP, http://www.servlets.com/soapbox/problems-jsp.html, January 25, 2000

[JSF] JavaServer Faces Technology, java.sun.com/j2ee/javaserverfaces

[JSTL] Java Server Pages Standard Tag Library (JSTL), http://java.sun.com/products/jsp/jstl/index.jsp

[JSP] JavaServer Pages Technology, http://java.sun.com/products/jsp/

[Servlets] Java servlet technology, http://java.sun.com/products/servlet/

[Spring] Spring Framework, http://www.springframework.org/

[Struts] Struts, http://struts.apache.org/

[Vel] Velocity – The Apache Jakarta Project, http://jakarta.apache.org/velocity/

[WM] WebMacro, http://www.webmacro.org/