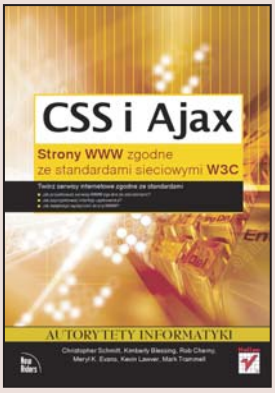
CSS i Ajax. Strony WWW zgodne ze standardami sieciowymi W3C



Autor: Christopher Schmitt, Kimberly Blessing, Rob Cherny, Meryl K. Evans, Kevin Lawver, Mark Tramme
Tłumaczenie: Robert Górczyński
ISBN: 978-83-246-1755-5
Tytuł oryginału: Adapting to Web Standards: CSS and Ajax for Big Sites (Voices That Matter)
Format: 168x237, stron: 312

Twórz serwisy internetowe zgodne z standardami!

- Jak projektować serwisy WWW zgodne z standardami?
- Jak zaprojektować interfejs użytkownika?
- Jak zwiększyć wydajność strony WWW?

Współczesne strony internetowe różnią się, a przynajmniej powinny, od tych sprzed kilku lat. Jeżeli różnic nie widać od strony graficznej, to z pewnością można je znaleźć w kodzie strony. Nowoczesne witryny WWW charakteryzują się zgodnością ze standardami, wykorzystaniem kaskadowych arkuszy stylów oraz udogodnieniami dla osób niepełnosprawnych. A dzięki zastosowaniu technologii AJAX współczesne witryny kuszą interaktywnością i dynamizmem.

Autorzy niniejszej książki wprowadzają Czytelnika w tajniki projektowania serwisów zgodnych z zasadami ustalonymi przez organizację W3C. Dzięki tej książce poznasz rodzaje i zastosowanie standardów sieciowych. Dowiesz się, w jaki sposób wybrać język znaczników, oraz poznasz zalety i wady języków HTML i XHTML. Nauczysz się korzystać z kaskadowych arkuszy stylów, a następnie odkryjesz sposób wykorzystania technologii AJAX w celu tworzenia interaktywnych stron WWW. Co najważniejsze, dostosujesz swoją stronę do współczesnych standardów, dzięki czemu zyska na dostępności i interaktywności, przyciągając rzesze użytkowników!

- Zalety standardów sieciowych
- Przygotowanie interfejsu użytkownika
- Wady i zalety języków HTML oraz XHTML
- Wpływ deklaracji DOCTYPE na zachowanie przeglądarki
- Korzyści płynące z zastosowania kaskadowych arkuszy stylów
- Sposoby dołączania arkuszy stylów do strony
- Zasady wykorzystania technologii AJAX
- Współpraca języka JavaScript z CSS
- Systemy zarządzania treścią
- Programowanie aplikacji sieciowych
- Sposoby na zwiększenie wydajności witryn internetowych

Dostosuj się do standardów – zwiększ dostępność i popularność swojej strony internetowej!

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

4

Programowanie aplikacji sieciowych

Profesjonalne zespoły programistów budujące witryny internetowe nie ograniczają się do tworzenia treści statycznych, ale oferują także aplikacje dynamiczne i witryny wykorzystujące systemy zarządzania treścią (CMS). W takim przypadku używany jest kod warstwy interfejsu użytkownika (UI), podzielony dalej na kod języka znaczników, arkusze stylów CSS oraz skrypty JavaScript. Znaczenie UI powoduje, że gotowa witryna internetowa jest większa, a ponadto wydłużają się okresy czasu między znaczącymi modernizacjami witryny. Powstaje więc pytanie, dlaczego nie pokusić się o ograniczenie zależności od programistów tworzących kod wewnętrzny bądź ekspertów z zakresu systemów CMS, skoro programiści UI mogliby samodzielnie zarządzać zmianami przeprowadzanymi w UI?

Jeżeli dodać do tego wyzwania wynikające z konieczności integracji z elementami opracowanymi przez firmy trzecie oraz starszymi i nowszymi elementami sieciowymi organizacji, to wyraźnie widać nieskończone pokłady korzyści płynących ze stosowania standardów sieciowych. W rzeczywistości wszystko, co jest tworzone w ramach kodu interfejsu użytkownika końcowego, powinno zostać zaimplementowane z użyciem technik zgodnych ze standardami sieciowymi.

Aplikacje sieciowe obciążone jarzmem przeszłości

Wielu programistów aplikacji sieciowych nauczyło się języka HTML w latach dziewięćdziesiątych ubiegłego stulecia i od tamtego okresu nie uaktualnili stosowanych technik programowania w celu wdrożenia nowych technologii. Przedstawienie nowych poziomów separacji i ścisłego używania standardów lub specyfikacji aplikacji programiście, który kształcił się w okresie powszechnego stosowania „zupy znaczników", może być naprawdę ciekawym doświadczeniem. Nowoczesne podejście bazujące na standardach sieciowych jest znacznie czystsze, efektywniejsze i logiczniejsze. Zwykle zmierza również do postawienia programisty sieciowego przed sytuacją „albo to działa, albo nie działa" — często wskrzeszając w nim zainteresowanie programowaniem interfejsu.

Z definicji witryny internetowe opierające się na standardach sieciowych dążą do ścisłego przestrzegania standardów, zawsze gdy tylko jest to możliwe. Kiedy oprogramowanie sieciowe umieszcza lub nawet tworzy kod warstwy UI, należy oczekiwać takiego samego poziomu dyscypliny. W dłuższej perspektywie czasu takie podejście przyniesie korzyści witrynie oraz aplikacji poprzez ograniczenie ryzyka, które nieuchronnie pojawia się wraz z kolejnymi modyfikacjami. Podejście wykorzystujące standardy sieciowe zwiększa poziom zgodności i dostępności treści witryny oraz redukuje poziom używanych zasobów i wysiłku, który trzeba włożyć w zmianę interfejsu użytkownika aplikacji lub statycznych fragmentów witryny.

Problem polega jednak na tym, że wciąż istnieje ogromna ilość archaicznego, niespójnego lub po prostu przestarzałego kodu, który został zaimplementowany w oprogramowaniu sieciowym. Programiści muszą bardzo dokładnie przyjrzeć się zarówno własnym, jak i komercyjnym implementacjom oprogramowania w celu określenia jak i kiedy oprogramowanie witryny może zyskać na zastosowaniu standardów sieciowych.

Jakość oprogramowania i analiza jego elementów

W celu właściwego określenia natury sytuacji, absolutnie każdy element oprogramowania sieciowego w projekcie powinien zostać poddany ewidencji, odpowiednio skatalogowany i określony pod względem zgodności (bądź jej braku) ze standardami sieciowymi.

Problemy związane z tworzeniem aplikacji sieciowej

Z perspektywy standardów sieciowych, problemy zwykle pojawiają się tam, gdzie programiści aplikacji muszą utworzyć warstwę UI — w obszarach związanych z obsługą, przeprojektowywaniem, delikatnymi zmianami w projekcie oraz oczywiście w kwestiach dotyczących dostępności treści i możliwości przeglądarek internetowych.

Zazwyczaj programista sieciowy albo wrzuca kod do szablonów przygotowanych przez zespół projektantów, albo próbuje dopasować wygląd i działanie do przedstawionego w statycznej reprezentacji aplikacji. Istnieją również pakiety oprogramowania, które generują kod UI, pozwalając na różne poziomy dostosowania go do własnych potrzeb. Ponadto, ponieważ programiści sieciowi tworzą również własny kod interfejsu, skutkuje to otrzymaniem różnego stopnia zgodności ze standardami — czasami całkiem świetnego i wystarczającego, a w innych przypadkach wręcz przeciwnie.

Oczywiście jest to dość złożony problem o wielu aspektach, ale kluczowy jest fakt, że struktury aplikacji, materiały edukacyjne, wygenerowany kod oraz narzędzia poważnie cierpią w wyniku braku demonstracji najlepszych praktyk i rozwiązań. Wielu programistów posługuje się różnymi formami zintegrowanego środowiska programowania (ang. *Integrated Development Environment* — IDE), bibliotekami kodu lub kopiuje kod, który może, ale nie musi opierać się na najlepszych rozwiązaniach.

W wyniku takiej sytuacji kod wewnętrzny jest tworzony z użyciem niewłaściwych znaczników wymieszanych wraz z warstwą prezentacyjną, w projektach stosuje się układy bazujące na tabelach, a ponadto dodatkowe i zupełnie zbędne znaczniki. Interfejsy często zawierają skrypty błędnie napisane lub utworzone pod kątem określonej przeglądarki internetowej, lub skrypty innych interfejsów w ogóle nie są brane pod uwagę. W takim przypadku zmiana projektu staje się prawdziwym problemem — a wcale nie musi nim być.

Warstwa UI jest albo utworzona w wyniku pracy człowieka, albo może być wygenerowana poprzez oprogramowanie (przy minimalnym poziomie wysiłku) i ślepo zaakceptowana przez użytkownika.

Powstaje pytanie: jak dużo wysiłku trzeba włożyć w jej opracowanie? Ponieważ współcześni inżynierowie aplikacji, platformy serwerowe i narzędzia zwykle domyślnie nie tworzą kodu interfejsu zgodnego ze standardami sieciowymi, to zadanie zapewnienia odpowiedniej jakości spadło na zespół. Osiągnięcie zgodności ze standardami sieciowymi interfejsu aplikacji i systemu zarządzania treścią wymaga włożenia dużego wysiłku. Ustanowienie zbioru najlepszych praktyk dotyczących kodu UI w oprogramowaniu działającym po stronie serwera może być punktem zwrotnym, od którego rozpocznie się usprawnianie oprogramowania.

Trzeba zdawać sobie sprawę z istnienia prawdziwego kompromisu między rozwiązaniem, które pozwoli na uniknięcie problemów w przyszłości, a po prostu naciskiem na jak najszybsze wydanie produktu. Z perspektywy dłuższego okresu czasu dążenie do uzyskania UI zgodnego ze standardami sieciowymi powoduje ograniczenie ryzyka związanego z brakiem spójności warstw logiki aplikacji, prezentacji i UI. Jednakże implementacja standardów może czasami zabrać dużo czasu.

Kryteria wartościowania

W przypadku oceny każdego dokumentu aplikacji należy wziąć pod uwagę kilka czynników:

- Skąd pochodzi dana warstwa UI?
- Jaka jest struktura kodu interfejsu?
- Czy kod to poprawny HTML lub XHTML?
- Czy w dokumencie znajduje się deklaracja DOCTYPE, a dokument jest generowany w „trybie standardów" czy „dziwactw"?
- W jaki sposób zostały zdefiniowane aspekty prezentacyjne?
- Czy programiści aplikacji zrozumieli kod UI?
- Czy programiści UI zrozumieli kod wewnętrzny?
- Ilu członków zespołu zrozumiało sposób integracji UI oraz kodu wewnętrznego?
- W jaki sposób jest generowany kod znaczników?
- Czy oprogramowanie odwołuje się do jakiegokolwiek bieżącego kodu prezentacyjnego, takiego jak pliki arkuszy stylów CSS? Jeżeli tak, to do jakich plików? Czy te pliki zmieniają się w projekcie?
- Jak blisko kod interfejsu jest powiązany z wewnętrzną logiką biznesową (o ile w ogóle jest powiązany)?
- Czy istnieje jakikolwiek sposób abstrakcji kodu UI od kodu aplikacji?
- Jeśli aplikacja generuje kod interfejsu, czy istnieje jakikolwiek sposób kontroli tego procesu?
- Jeżeli użytkownik ma możliwość generowania treści lub kodu, to czy istnieje jakikolwiek sposób zablokowania tego, co użytkownik może zrobić?
- Jak duży wysiłek trzeba będzie włożyć w modyfikacje warstwy UI?
- Jak duże jest ryzyko związane z modyfikacją warstwy UI aplikacji?
- Jakie są potencjalne korzyści i wady wynikające z zaniechania przeprojektowania aplikacji?
- Jakie są ograniczenia związane z tym, że oprogramowanie firm trzecich może wymagać dostosowania do własnych potrzeb?

Szczerze mówiąc, to jest ogromna ilość informacji, które trzeba zebrać, przeanalizować, a następnie rozważyć.

Czy aplikacja może zostać uaktualniona?

Na koniec trzeba wspomnieć, że w przypadku części oprogramowania być może niewiele będzie można zdziałać. Taka sytuacja może wynikać z faktu, że po prostu nie będzie możliwości uaktualnienia kodu interfejsu do postaci zgodnej ze standardami sieciowymi. Powodem mogą być albo kwestie związane z architekturą aplikacji,

albo nieakceptowanym poziomem ryzyka związanego z uaktualnieniem. W takich przypadkach wysiłki stają się ćwiczeniem w zakresie kompromisów. Albo potrzeby staną się powodem utworzenia długoterminowego planem nastawionego na rozwiązanie problemów, albo zarówno użytkownicy, jak i biznes będą cierpieli z powodu bolesnego procesu uaktualnienia.

Między krótkoterminowym wysiłkiem i ryzykiem oraz długoterminowym wysiłkiem i ryzykiem występują wyraźne kompromisy. Niektóre rozwiązania częściowe zakładają albo implementację części stylów (nawet wyodrębnionych z określonych plików CSS), albo powierzchowne ponowne opracowanie wyglądu i działania danej aplikacji. Jednak z perspektywy długiego okresu czasu organizacja będzie z tego powodu cierpiała, ponieważ coraz trudniejsze stanie się wprowadzanie zmian w aplikacji. Te zmiany staną się przypadkami szczególnymi i za każdym razem będą wymagały szczególnego podejścia. Punkt krytyczny będzie inny dla każdej organizacji.

Wskazówki, reguły oraz standardy sieciowe

Aplikacje sieciowe opracowane przez organizację mogą osiągać korzyści wynikające ze stosowania standardów sieciowych, wytycznych programowania, punktów wzajemnego oddziaływania oraz wskazówek dotyczących sposobu pisania oraz stylu kodu UI.

Reguły programowania

Istnieją standardy dotyczące jakości i spójności programowania, podobne do często stosowanych przez programistów w kodzie wewnętrznym aplikacji. Jednak w chwili obecnej zostały rozszerzone na kod interfejsów, co może być zupełną nowością.

- Każdy kod UI powinien spełniać podstawowe, najlepsze praktyki dotyczące standardów sieciowych, czyli między innymi stosować styl POSH, arkusze stylów CSS oraz dyskretny kod JavaScript (jak to opisano w rozdziałach — odpowiednio — 1., 2. i 3.).
- Kluczowe znaczenie ma tworzenie kodu niezależnego od rodzaju przeglądarek internetowych, zapewniającego dostępność oraz umożliwiającego (w razie potrzeby) elegancką degradację funkcji.
- Programy warstwy UI powinny odwoływać się do klas CSS i atrybutów `ID` poprzez wyciągnięcie ich z głębi logiki aplikacji do innych, łatwo dostępnych miejsc kodu. W ten sposób przeprowadzanie modyfikacji informacji prezentacyjnych niesie ze sobą minimalne ryzyko dla aplikacji. Wymienione klasy CSS i atrybuty `ID` mogą być powierzchownymi właściwościami klas obiektu, przechowywane w plikach konfiguracyjnych lub zmiennych o zasięgu aplikacji, co umożliwia ich łatwą zmianę w przyszłości.

- Należy za wszelką cenę unikać stosowania osadzonych w kodzie stylów bądź innych elementów prezentacyjnych.
- W kwestii aplikacji JavaScript warto współpracować z programistami interfejsu, tak aby maksymalnie współdzielić skrypty i unikać związanych z nimi konfliktów.
- Opracować najprostszy z możliwych i semantyczny kod znaczników aplikacji.
- Utworzyć proste, standardowe reguły CSS formularzy, tak aby po dodaniu nowych formularzy zostały w nich zastosowane style CSS bez potrzeby włożenia większego wysiłku.

Dzięki stosowaniu powyższych wskazówek oraz tworzeniu czystego, rozdzielonego kodu zgodnego ze standardami sieciowymi, programista zagwarantuje, że aplikacja i oprogramowanie o znaczeniu krytycznym dla biznesu nie będzie wymagało dużych bądź ryzykownych modyfikacji podczas kolejnego przeprojektowania.

Niestety, wymienione powyżej wskazówki często odbiegają od tego, co znajduje się w pakietach oprogramowania, narzędziach oraz kodzie generowanym przez IDE lub edytory typu WYSIWYG. Te narzędzia zwykle pozwalają programistom jedynie na usunięcie z kodu ustawień prezentacyjnych, umieszczenie warstwy prezentacyjnej w klasach CSS, itp. Analiza alternatywnych ustawień lub eksperymenty z funkcjami narzędzia pozwalające na osiągnięcie pożądanych wyników mogą wymagać większej ilości czasu i wysiłku. Jednak bardzo często wystarczy powziąć proste kroki pozwalające na oddzielenie logiki aplikacji od kodu wewnętrznego.

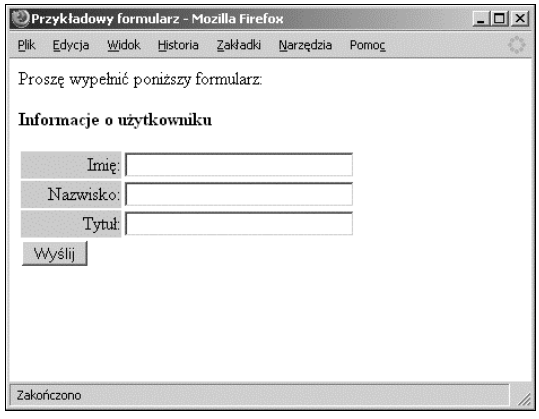
Lepsze formularze tworzone za pomocą nowoczesnego kodu znaczników

Większość aplikacji sieciowych zawiera w sobie pewne rodzaje formularzy. Aby zapewnić tym formularzom elegancki układ, programiści najczęściej używają tabel HTML.

Chociaż stosowanie tej techniki właściwie nie jest błędne, to trzeba przypomnieć, że tabele zostały opracowane z myślą o umieszczaniu w nich danych tabelarycznych. Ponieważ formularz nie zawiera tego rodzaju danych, to nie ma najmniejszego powodu do stosowania tabeli. Ponadto tabele często zawierają atrybuty prezentacyjne, takie jak `bgcolor`, `align` i `width`, oraz nie oferują takich korzyści w kwestii dostępności treści, jakie można uzyskać stosując podejście zgodne ze standardami sieciowymi. Formularze bardzo często używają także kodu po stronie serwera, którego zadaniem jest umieszczenie pewnych wartości w formularzu (więcej na ten temat zostanie przedstawione w dalszej części rozdziału).

Przykładowo: poniżej przedstawiono kod wyświetlający prosty formularz, który został umieszczony w tabeli (zobacz rysunek 4.1):

```
<p>Proszę wypełnić poniższy formularz:</p>
<p><b>Informacje o użytkowniku</b></p>
<form action="submit.php" method="post">
```

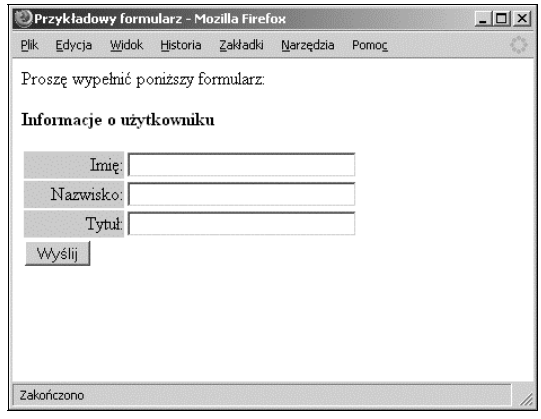


RYSUNEK 4.1. Formularze są najczęściej umieszczane w tabelach HTML i używają znaczników prezentacyjnych

```
<table width="300" border="0">
<tr>
   <td bgcolor="#cccccc" width="30%" align="right">Imię:</td>
   <td><input type="text" name="txtFName" size="30" /></td>
</tr>
<tr>
   <td bgcolor="#cccccc" align="right">Nazwisko:</td>
   <td><input type="text" name="txtLName" size="30" /></td>
</tr>
<tr>
   <td bgcolor="#cccccc" align="right">Tytuł:</td>
   <td><input type="text" name="txtTitle" size="30" /></td>
</tr>
<tr>
   <td colspan="2"><input type="submit" value="Wyślij" /></td>
</tr>
</table>
</form>
```

Dla porównania, poniżej znajduje się nowa wersja kodu powodującego wyświetlenie tego samego formularza (zobacz rysunek 4.2):

```
<p>Proszę wypełnić poniższy formularz:</p>
<form action="submit.php" method="post">
<div id="formBlock">
   <fieldset>
   <legend>Informacje o użytkowniku</legend>
   <p>
      <label for="txtFName">Imię:</label>
      <input type="text" id="txtFName" tabindex="1" />
   </p>
   <p>
```



RYSUNEK 4.2. Stosowanie nowoczesnego kodu znaczników powoduje, że tworzenie prostych formularzy jest zdumiewająco łatwe

```
      <label for="txtLName">Nazwisko:</label>
      <input type="text" id="txtLName" tabindex="2" />
   </p>
   <p>
      <label for="txtFName">Tytuł:</label>
      <input type="text" id="txtTitle" tabindex="3" />
   </p>
   <p><input type="submit" value="Wyślij" tabindex="4" /></p>
   </fieldset>
</div>
</form>
```

Przedstawiony powyżej kod formularza może zostać połączony z poniższym arkuszem stylów CSS, który powinien być stosowany względem każdego formularza na witrynie, co zapewni im spójny wygląd. Wprawdzie takie rozwiązanie wymaga użycia dodatkowego kodu, ale zastosowanie spójnego sposobu tworzenia formularzy i nadawania im stylu będzie ogromną zaletą dla aplikacji.

```
h1 {
   margin: 0;
   font-weight: normal;
   font-size: 15px;
}
#formBlock {
   width: 300px;
}
#formBlock p {
   margin: 0 0 3px;
}
p {
   width: 100%;
}
```

```
label {
   background-color: #ccc;
   display: block;
   float: left;
   width: 28%;
   text-align: right;
   margin-right: 2px;
   padding: 2px 0;
}
input {
   float: left;
   width: 65%;
   margin-bottom: 4px;
   margin-top: 1px;
   padding: 1px;
}
input[type=submit] {
   width: 10%;
}
fieldset {
   border: none;
   padding: 0;
   margin: 0;
}
legend {
   font-weight: bold;
   margin-bottom: 12px;
}
```

Analiza nowego kodu XHTML napisanego z uwzględnieniem standardów sieciowych pokazuje zasadnicze usprawnienia w stosunku do formularzy utworzonych bez zastosowania podejścia semantycznego:

- W kodzie nie ma już niepotrzebnej tabeli HTML służącej do nadania wyglądu formularzowi.
- Formularz zawiera elementy `<label>`, które przypisują etykiety faktycznym kontrolkom formularza — doskonałe rozwiązanie z punktu widzenia dostępności, funkcja opracowana z myślą o użytkowniku i obsługiwana przez większość przeglądarek internetowych.
- Zastosowany został atrybut `tabindex` ułatwiający ustalenie kolejności przechodzenia między polami oraz zwiększający dostępność strony za pomocą klawiatury.
- Formularz został zgrupowany w znaczniku `<fieldset>` oraz oznaczony etykietą za pomocą znacznika `<legend>`, który grupuje i opisuje formularz w celu zwiększenia jego dostępności.

- Ponieważ kod formularza jest czystszy i czytelniejszy, to po dołączeniu do aplikacji programiści zajmujący się jej obsługują będą mogli łatwiej odczytywać i modyfikować ten formularz.

Struktury działające po stronie serwera oraz narzędzia szablonów

Jeżeli chodzi o technologie skryptowe w Internecie, to dostępnych jest ich kilka, między innymi PHP (PHP: Hypertext Preprocessor), klasyczne ASP (ang. *Active Server Pages*), ColdFusion (opracowana przez firmę Macromedia, która kilka lat temu została przejęta przez Adobe). Wymienione platformy językowe służące do wykonywania zadań po stronie serwera posiadają modele, które umieszczają w tych samych plikach kod interfejsu oraz kod wewnętrzny aplikacji. W jednym wierszu kodu programiści natykają się na kod wewnętrzny umieszczony wewnątrz zdefiniowanych ograniczników (na przykład takich jak <%...%>), a następnie znajdują się standardowe znaczniki HTML. Ponadto kod działający po stronie serwera będzie używał poleceń `print` w celu wygenerowania kodu HTML, często wymieszanego wraz z danymi prezentacyjnymi.

Dostępnych jest kilka platform oraz technik programowania — na przykład ColdFusion Fusebox (*http://fusebox.org*) oraz system szablonów PHP Smarty (*http://smarty.php.net*) — które zapewniają modularyzację warstw oraz próbują rozdzielić logikę biznesową od kodu interfejsu. W tym celu tworzą pliki szablonów zawierające kod interfejsu, podczas gdy kod wewnętrzny zostaje umieszczony w oddzielnych plikach. W rzeczywistości osiągnięte wyniki są z reguły bardzo zróżnicowane, ponieważ kod działający po stronie serwera nadal musi „wyprodukować" interfejs użytkownika.

Pytanie, które można zadać pod koniec omawiania tego tematu, brzmi: jaka jest jakość kodu znaczników tworzącego UI nawet po jego oddzieleniu od logiki biznesowej? Wszelkie dostępne na świecie warstwy oprogramowania aplikacji nie zdadzą się na nic, jeżeli podstawowy kod HTML będzie łamał opracowane najlepsze rozwiązania lub zostanie umieszczony w plikach, których programiści tworzący UI nie będą mogli kontrolować lub przez które nie będą mogli przebrnąć.

Proste kroki prowadzące do tworzenia lepszych skryptów działających po stronie serwera

W przypadku wszystkich platform lub systemów wykorzystujących szablony można powziąć proste kroki, które mogą ograniczyć potencjalne szkody, nawet w przypadku podstawowego PHP lub podobnych środowiskach. W najbardziej podstawowym sensie wszelkie dane wyjściowe aplikacji lub generowany kod UI będą przedmiotem stosowania tych samych reguł, które odnoszą się do kodu UI nieużywającego bazy danych. Jedyna różnica polega na tym, że tego rodzaju kod jest po prostu generowany, a nie tworzony ręcznie przez programistów.

Warto zwrócić uwagę, że wyzwania związane z uzyskaniem czystego rozdziału logiki biznesowej po stronie serwera od warstwy prezentacyjnej nie są jedynie domeną języka PHP. Większość języków programowania działających po stronie serwera wykorzystuje dobre praktyki programowania i dyscyplinę. Cechą charakterystyczną, szczególnie w przypadku PHP, klasycznego ASP i ColdFusion, jest bardzo często osadzanie logiki aplikacji w tych samych plikach, które zawierają kod interfejsu, na przykład:

```
<?php
// Wyświetlenie wyników.
echo "<table border=\"1\" width=\"400\">";
while ($row = mysql_fetch_assoc($result)) {
   echo "<tr valign=\"top\">";
   echo "<td bgcolor=\"#ffffcc\"><b>$row['username']</b></td>";
   echo "<td>$row['firstname']</td>";
   echo "<td>$row['lastname']</td>";
   echo "<td><font color=\"grey\">$row['notes']</font></td>";
   echo "</tr>";
}
echo "</table>";
?>
```

Przedstawiony powyżej kod wykonuje w PHP proste zadanie — kolejno przechodzi przez wiersze zbioru wynikowego zwróconego przez bazę danych. Większość języków skryptowych, w celu wyświetlenia danych wyjściowych pobranych z bazy danych stosuje podobne techniki do przedstawionej powyżej, opierając się na strukturze pętli. Oczywiście, przedstawiony przykład jest mały i bardzo prosty, ale może on być umieszczony głęboko wewnątrz innej struktury pętli bądź skomplikowanej struktury pętli.

Zaprezentowany powyżej kod (oraz inne, podobne kody działające po stronie serwera) ma kilka wad:

- Elementy prezentacyjne zostały osadzone w kodzie pętli.
- Zmiana projektu wizualnego będzie wymagała zmian w aplikacji. W tym celu trzeba będzie skorzystać z usług programisty, zamiast po prostu wprowadzić zmiany w zewnętrznym (z punktu widzenia aplikacji) pliku arkuszy stylów CSS.
- Za każdym razem, gdy zajdzie potrzeba zmiany wyglądu tabeli, programista będzie musiał odszukać kod prezentacyjny i przeprowadzić modyfikacje bezpośrednio w kodzie logiki aplikacji.
- W atrybutach prezentacyjnych zastosowano znaki sterujące, ponieważ atrybuty w języku HTML używają znaków cudzysłowów. Ten krok powoduje, że kod staje się mniej czytelny i trudniejszy w zarządzaniu.
- W każdej tabeli znajdującej się na witrynie trzeba będzie ręcznie wprowadzić dane prezentacyjne.

Mimo że powyższy kod jest prosty, to demonstruje pomieszanie kodu wewnętrznego z warstwą prezentacyjną. Łatwo sobie wyobrazić, jak mógłby wyglądać kod wraz z zagnieżdżonymi tabelami służącymi do utworzenia krawędzi i menu oraz stosujący inne atrybuty prezentacyjne dla każdej z kolumn… Kod stałby się zupełnie nieczytelny. Elegancką alternatywą jest następujący kod:

```
<style type="text/css">
#results {
   border: 1px solid #000;
   width: 400px;
}
#results td {
   border: 1px inset #000;
   vertical-align: top;
}
td.username {
   background-color: #ffc;
}
td.notes {
   color: gray;
}
</style>
<?php
echo "<table id=\"results\">";
while ($row = mysql_fetch_assoc($result)) {
   echo "<tr>";
   echo "<td class=\"username\">$row['username']</td>";
   echo "<td>$row['firstname']</td>";
   echo "<td>$row['lastname']</td>";
   echo "<td class=\"notes\">$row['notes']</td>";
echo "</tr>";
}
echo "</table>";
?>
```

W celu zachowania zwięzłości, na powyższym fragmencie kodu przedstawiono tylko blok `<style>` zawierający definicję arkuszy stylów CSS. Można jednak dostrzec, że w celu zmiany wyglądu tabeli, czcionek lub sposobu wyrównania treści w komórce tabeli nie zachodzi potrzeba modyfikacji żadnego wiersza kodu PHP. Przykład jest dość szczątkowy, ale należy pamiętać o jednym — im bardziej skomplikowana logika biznesowa, tym większe korzyści płyną z wyrzucenia informacji prezentacyjnych z logiki biznesowej.

UWAGA

W przedstawionej powyżej tabeli idealnym rozwiązaniem byłoby również użycie wiersza `<th>` w celu utworzenia nagłówków opisujących kolumny. W ten sposób można zwiększyć zarówno poziom dostępności treści strony, jak i semantyczność kodu. Jeżeli projekt graficzny nie zakłada wyświetlania nagłówków tabeli to zawsze można je ukryć za pomocą arkusza stylów CSS. Pozostaną one jednak dostępne dla urządzeń wspomagających.

Problem

Zaletą stosowania technologii, takich jak PHP lub nawet klasyczne ASP, jest fakt, że programista sieciowy zachowuje pełną kontrolę nad sposobem generowania kodu UI — zaleta, której nie należy lekceważyć. Koszty związane z utrzymaniem i obsługą interfejsu użytkownika aplikacji mogą stać się problemem w zespole, gdy zachodzi prawdopodobieństwo, że kod UI nie został napisany przez programistów tego zespołu. W takich przypadkach bardzo ważna jest przede wszystkim komunikacja oraz ponowna ocena. Oczywiście, są to przecież procedury biznesowe, a nie techniki programowania, ale bardzo często te podstawowe procesy po prostu nie zachodzą.

Problem polega na tym, że we współczesnym świecie coraz więcej nowych produktów i technologii zyskuje szeroką akceptację, a więc proces oceny oraz programowania UI stał się znacznie trudniejszy niż wcześniej. W przypadku złożonych aplikacji biznesowych języki skryptowe — na przykład te omówione dotąd w książce — wypadły z obiegu, ponieważ sama platforma nie narzuca architektury warstwom aplikacji. Ciężarem spoczywającym na programistach jest stosowanie strukturalnych wzorców projektowych, które wymagają używania dobrych praktyk w zakresie programowania. Starsze narzędzia, takie jak PHP i ColdFusion, były często znacznie bardziej dostępne dla programistów lub projektantów UI niż niektóre nowsze technologie.

Zasadniczy problem pozostał, a w przypadku większości nowych środowisk programowania działających po stronie serwera stał się nawet jeszcze poważniejszy. Problem dotyczy zwłaszcza sytuacji, gdy osadzony skrypt działający po stronie serwera, na przykład PHP lub klasyczne ASP (obecnie ASP.NET) stosujący podejście warstwowe, próbuje oddzielić logikę biznesową od kodu wewnętrznego. Problem może również wystąpić, gdy język XSLT jest używany do generowania kodu XHTML lub HTML z XML.

Jaki jest więc ten zasadniczy problem? Kiedy inżynierowie oprogramowania budowali coraz dojrzalsze i działające na warstwach środowiska interfejsów, takie jak Java, ASP.NET lub XML/XSLT, to programiści interfejsów nie poznali elementów odpowiedzialnych za interfejs w tychże środowiskach programistycznych — *i najczęściej nigdy nie poznają*.

Zwykle elementy odpowiedzialne za interfejs w tych platformach oprogramowania generują dane wyjściowe, w postaci takiego samego, przestarzałego i niespełniającego standardów kod interfejsu, który już wcześniej był stosowany. Prawdziwym wyzwaniem jest więc implementacja standardów sieciowych w interfejsie tego rodzaju aplikacji.

Ostrzeżenie przed kodem generowanym po stronie serwera

Część nowoczesnych środowisk aplikacji posiada funkcje zaprojektowane w celu pomocy podczas usuwania „zbędnego" generowania kodu HTML lub innego kodu UI z aplikacji. Koncepcja polega na tym, aby kod HTML (lub XHTML) mógł zostać w łatwy sposób wyizolowany, a następnie dynamicznie wygenerowany za pomocą poleceń przekazanych do języka programowania pakietu narzędziowego. Zazwyczaj stosowane są właściwości przypisywane zbiorom danych zwracanych przez zapytania do bazy danych lub inne podobne struktury kontrolujące dynamiczne tworzenie danych wyjściowych.

Problem związany z tego rodzaju podejściem to fakt, że zarówno projekt, jak i ostatecznie generowany kod opierają się w dużym stopniu na decyzjach programistów tworzących oprogramowanie używane do abstrahowania i generowania kodu. Dlatego też sukces jest uzależniony również od wysiłków programisty mających na celu poznanie funkcji UI dostępnych w używanym oprogramowaniu. Różne platformy oferują zróżnicowane poziomy jakości. Programiści muszą więc „poznać kod źródłowy" i faktycznie obserwować strukturę kodu znaczników po jego wygenerowaniu, zamiast po prostu przyglądać mu się, gdyż kod znaczników jest generowany dynamicznie.

W większości przypadków nic nie jest w stanie zastąpić czynnika ludzkiego. Przejrzenie rzeczywistych działaniach realizowanych przez kod i zorientowanie się w nich wymaga wykonania kilku dodatkowych kroków. Czasami trzeba będzie wprowadzić kilka poprawek, a innym razem nie będzie to konieczne. Skoro Czytelnik dotarł aż do tego miejsca książki, to powinno być dla niego oczywiste, że utworzenie interfejsu zgodnego ze standardami sieciowymi może przynieść wymierne korzyści.

Platforma Microsoft ASP.NET

Jedną z platform, która została szeroko zastosowana zwłaszcza w dużych organizacjach, jest dość krytycznie przyjęta technologia firmy Microsoft o nazwie ASP.NET (następca „klasycznego" ASP). Klasy podstawowe .NET, obiekty oraz API mają ogromne możliwości.

Technologia ASP.NET wymusza poprzez stosowany model używanie warstw kodu, które oddzielają do pewnego stopnia logikę programu oraz bazę danych od plików „interfejsu użytkownika" posiadających rozszerzenie *.aspx*. Wymienione pliki zawierają kod HTML oraz własne znaczniki ASP.NET, do których powrócimy w dalszej części rozdziału. Teoretycznie programiści UI uczą się wymienionych znaczników w celu uzyskania kontroli nad interfejsem i utworzenia punktów zaczepienia dla programistów kodu wewnętrznego. To jest faktycznie możliwe, ale w rzeczywistości rzadko osiągane.

Zespół programistów mógłby podjąć nieśmiałą decyzję dotyczącą edukacji zespołu programistów UI, a następnie współpracować z nimi podczas nauki, natomiast zespół UI mógłby wykazać motywację do nauki. Takie rzeczy się zdarzają.

Zazwyczaj, aplikacje sieciowe ASP.NET są tworzone w środowisku IDE o nazwie Microsoft Visual Studio.NET. Wymienione środowisko IDE oferuje okno działające w trybie WYSIWYG uzupełniające widok kodu źródłowego oraz pozwalające programistom na szybkie przeciąganie obiektów na stronę, ustawianie właściwości, także tych dotyczących atrybutów prezentacyjnych bez konieczności napisania ani jednego wiersza kodu. Te ustawienia generują kod HTML.

TO NIE TAK, ŻE CZEPIAMY SIĘ PLATFORMY ASP.NET

W rzeczywistości jest wręcz na odwrót. Omówione w dalszej części rozdziału przykłady zaprezentują potężne możliwości i wszechstronność ASP.NET oraz podobnych platform w rękach osoby, która doskonale wie, w jaki sposób posługiwać się oprogramowaniem w celu uzyskania kodu zgodnego ze standardami sieciowymi. Dostępnych jest wiele podobnych platform, a większość z nich pozwala na modyfikację sposobu generowania danych wyjściowych. Wymaga to jednak włożenia wysiłku w celu odszukania i poznania funkcji, które pomagają w wygenerowaniu wysokiej jakości kodu interfejsu.

Między poszczególnymi platformami koncepcje są niemal takie same. Konieczne jest włożenie pewnego wysiłku i poświęcenie czasu na zrozumienie, jakie dane zostaną wygenerowane, zamiast zwykłego stwierdzenia „to z grubsza wygląda dobrze". W ten sposób zyskują wszyscy — użytkownicy witryny, firma i cała reszta. Podczas pracy nad aplikacją programiści bywają całkiem mocno spięci i często nie przejmują się ostatnimi krokami. Oczywiście, istnieją wyjątki, ale ta sekcja nie jest przeznaczona dla takich indywidualności.

Z punktu widzenia standardów sieciowych istnieje kilka kwestii, na które trzeba zwrócić uwagę, używając Visual Studio.NET, zwłaszcza w wersji wcześniejszej niż 2005, oraz kodu ASP.NET generowanego przez serwer:

- Większość przykładów i materiałów edukacyjnych ASP.NET zawiera nieprawidłowy kod.
- Visual Studio.NET 2003 oraz wcześniejsze wersje maja złą sławę z powodu przepisywania i zmiany formatowania kodu HTML strony (kiedy programista przejdzie do trybu graficznego), nawet jeśli została utworzona i starannie dopracowana w innej aplikacji bądź zwykłym edytorze tekstowym przez programistę UI.
- Formularze i kontrolki danych mogą być „znacznikami" ASP.NET, które bardzo często mają osadzone w nich atrybuty prezentacyjne. Wygenerowany kod źródłowy najczęściej nie jest zgodny z XHTML lub HTML, choć w wersji 2005 zanotowano

na tym polu ogromną poprawę. Jeżeli programista nie włoży dodatkowego wysiłku w nadzór generowanych danych wyjściowych, to wymienione atrybuty znaczników będą miały osadzoną olbrzymią ilość danych prezentacyjnych.

- W wersjach 2003 i wcześniejszych, kiedy programista używał wbudowanych narzędzi do tworzenia strony, tak utworzona strona HTML była wyświetlana w „trybie dziwactw", ponieważ deklaracja DOCTYPE dołączona do dokumentu nie zawierała URI.
- ASP.NET zawiera funkcję „inteligentnego generowania", która oznacza, że serwer podejmuje decyzje dotyczące rodzaju wysyłanej do przeglądarki internetowej warstwy kodu UI na podstawie ustawień dokonanych po stronie serwera. Serwer „zgaduje" poziom obsługi danej technologii przez przeglądarkę internetową.
- Kod ASP.NET działający po stronie serwera używa dużych ilości atrybutów `ID` w kontrolkach osadzonych na stronie. Najczęściej, w zależności od kontekstu, wymienione atrybuty `ID` mają dynamicznie zmieniane nazwy na podstawie ich położenia w dokumencie. Dlatego też programiści CSS i JavaScript mają znacznie utrudnione zadanie w kwestii odwoływania się do obiektów strony poprzez ich wartości `ID`. I nie ma tutaj znaczenia, że taki sposób odwoływania jest otwarty, rozpoznawany wszędzie i zgodny ze standardami sieciowymi. W niektórych przypadkach zastosowanie klasy CSS może być odpowiednim rozwiązaniem. Można również spróbować zastosować wymagany atrybut `ID` elementowi nadrzędnemu, a odwołać się do żądanego obiektu poprzez kontekst. Idealnym wyjściem jest, gdy projektanci interfejsu i programiści kodu wewnętrznego współpracują ze sobą podczas ustawiania atrybutów `ID` dla interfejsu użytkownika. Wówczas będzie to najlepszy sposób wykorzystania możliwości kodu zarówno interfejsu jak i wewnętrznego — ale takie rozwiązanie nie jest dostępne w każdym przypadku.

Warto zwrócić uwagę na jeden interesujący fakt. Mianowicie, wymienione problemy można pokonać wkładając dodatkowy wysiłek oraz poświęcając więcej czasu i uwagi aspektom kodu interfejsu użytkownika.

WSKAZÓWKA

Jednym ze sposobów poradzenia sobie z funkcją „inteligentnego generowania" jest ustawienie opcji konfiguracyjnej `ClientTarget` wartości `Uplevel`. W ten sposób ASP.NET przestanie zakładać, że jedyna przeglądarka oferująca zaawansowane funkcje to Microsoft Internet Explorer, i zacznie wysyłać zaawansowany kod do wszystkich. Jednak takie rozwiązanie powoduje powstanie problemów z niektórymi funkcjami, zwłaszcza dotyczącymi ASP.NET 1.1 i JavaScript. Będą one obsługiwane jedynie przez przeglądarkę IE, ponieważ Microsoft w pewnych miejscach zastosował kod obsługiwany tylko i wyłącznie przez tę przeglądarkę. Należy zachować szczególną ostrożność i dokładnie przetestować aplikację ASP.NET jeszcze przed jej wdrożeniem. Co więcej, wymienione funkcje zostały w pewnym stopniu zmienione w ASP.NET 2.0 — kolejny powód do wykonywania wczesnych i częstych testów.

DUŻE USPRAWNIENIA W ASP.NET 2.0

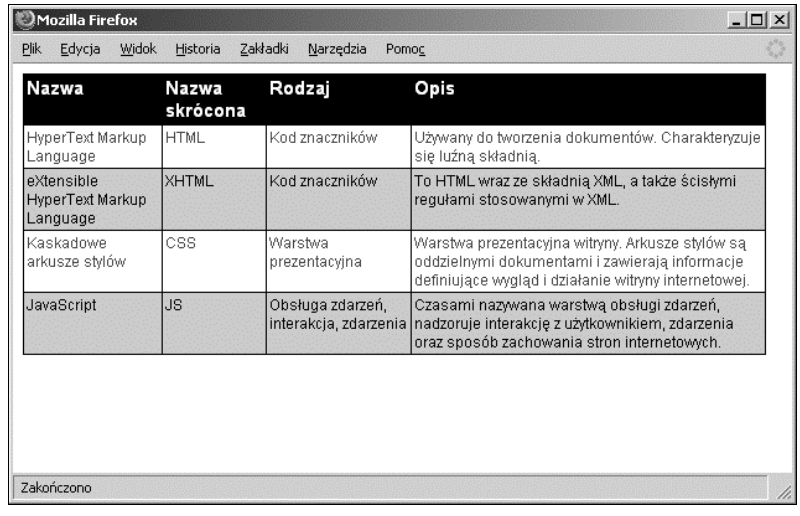
Zarówno w ASP.NET 2.0 jak i Visual Studio.NET 2005 firma Microsoft podjęła duże wysiłki, aby generowany kod był w jeszcze większym stopniu zgodny ze standardami sieciowymi i zapewniał większą dostępność. Inne wprowadzone usprawnienie polega na tym, że nowa wersja środowiska IDE po przejściu do trybu graficznego przepisuje (modyfikuje) znacznie mniejszą ilość kodu.

Kod kontrolek i formularzy generowany automatycznie przez platformy Visual Studio.NET i ASP.NET jest praktycznie zgodny z językiem XHTML w wersji Transitional, co jest pochodną ścisłej natury specyfikacji. Jednak programista nadal musi włożyć dodatkowy wysiłek, aby uniknąć generowania osadzonych atrybutów prezentacyjnych i zwiększyć poziom dostępności kodu. W pewnych sytuacjach lepiej użyć innych kontrolek, które pozwalają na większą elastyczność w zakresie generowanego kodu. Wymienione opcje zostały znacznie usprawnione w ASP.NET 2.0.

Dane wyjściowe ASP.NET

Podobnie jak w przypadku innych języków, także ASP.NET umożliwia wyświetlanie danych wyjściowych pochodzących z bazy danych. Jednak jedynym sposobem zaimplementowanym przez Microsoft jest wykorzystanie znaczników działających po stronie serwera. Wszystkie wymienione znaczniki mają atrybuty prezentacyjne, które w domyśle miały pomoc programiście w nadzorowaniu prezentacyjnych aspektów kodu XHTML danych wyjściowych. Poniżej przedstawiono typowy przykład kontrolek ASP.NET używających funkcji o nazwie `DataGrid` do utworzenia danych wyjściowych w postaci tabeli zawierającej elementy pochodzące z bazy danych. Kod został osadzony w pliku *.aspx* otoczonym pozostałą częścią dokumentu (X)HTML (zobacz rysunek 4.3).

```
<asp:DataGrid ID="catalog" runat="server" AutoGenerateColumns="false"
   CellPadding="2" Width="600" BorderColor="black">
   <ItemStyle Font-Names="Arial" Font-Size="12px"
      ForeColor="#800000" VerticalAlign="top" />
   <AlternatingItemStyle Font-Names="Arial" Font-Size="12px"
      ForeColor="#400000" BackColor="#cccccc" />
   <HeaderStyle Font-Bold="true" Font-Names="Arial" Font-Size="15px"
      ForeColor="white" BackColor="black" VerticalAlign="top" />
   <Columns>
      <asp:BoundColumn DataField="Name" HeaderText="Nazwa" />
      <asp:BoundColumn DataField="abbr" HeaderText="Nazwa skrócona" />
      <asp:BoundColumn DataField="typeof" HeaderText="Rodzaj" />
      <asp:BoundColumn DataField="notes" HeaderText="Opis" />
   </Columns>
</asp:DataGrid>
```



RYSUNEK 4.3. Kontrolka DataGrid w ASP.NET wyświetlająca dane

Powyższy kod powoduje zdefiniowanie tabeli danych wyjściowych oraz naprzemiennych kolorów jej wierszy. Atrybuty opisujące wygląd i sposób działania tabeli są po prostu niektórymi z dostępnych opcji. W rzeczywistości ilość tych opcji jest ogromna. Dane źródłowe tabeli są pobierane poprzez inną warstwę kodu. Wynika to z separacji danych od warstwy prezentacyjnej, która jest doskonałym rozwiązaniem choć w przypadku ASP.NET tylko teorią. Poniżej przedstawiono kod wygenerowany przez powyższy fragment kodu:

```
<table cellspacing="0" cellpadding="2" rules="all" border="1"
id="catalog" style="border-color:Black;width:600px;border-collapse:
collapse;">
   <tr valign="top" style="color:White;background-color:Black;
font-family:Arial;font-size:15px;font-weight:bold;">
      <td>Nazwa</td><td>Nazwa
skrócona</td><td>Rodzaj</td><td>Opis</td>
   </tr>
   <tr valign="top" style="color:Maroon;font-family:Arial;
font-size:12px;">
      <td>HyperText Markup Language</td><td>HTML</td><td>Kod
      ↪znaczników</td>
      <td>Używany do tworzenia dokumentów. Charakteryzuje się luźną
      ↪składnią.</td>
   </tr>
   <tr valign="top" style="color:#400000;backgroundcolor:#
CCCCCC;font-family:Arial;font-size:12px;">
      <td>eXtensible HyperText Markup Language</td><td>XHTML</td>
      <td>Kod znaczników</td><td>To HTML wraz ze składnią XML, a także
      ↪ścisłymi regułami stosowanymi w XML.</td>
```

```
   </tr>
   <tr valign="top" style="color:Maroon;font-family:Arial;
font-size:12px;">
      <td>Kaskadowe arkusze stylów</td><td>CSS</td><td>Warstwa
      ↪prezentacyjna</td>
      <td>Warstwa prezentacyjna witryny. Arkusze stylów są oddzielnymi
      ↪dokumentami i zawierają informacje definiujące wygląd
      ↪i działanie witryny internetowej.</td>
   </tr>
   <tr valign="top" style="color:#400000;backgroundcolor:#
CCCCCC;font-family:Arial;font-size:12px;">
      <td>JavaScript</td><td>JS</td><td>Obsługa zdarzeń, interakcja,
      ↪zdarzenia</td>
      <td>Czasami nazywana warstwą obsługi zdarzeń, nadzoruje
      ↪interakcję z użytkownikiem, zdarzenia oraz sposób zachowania
      ↪stron internetowych.</td>
   </tr>
</table>
```

Przedstawiona tabela jest dość prosta, zawiera naprzemienne kolory wierszy, ustawienia rodziny czcionek oraz elegancko stosuje arkusze stylów CSS zamiast znaczników `<font>`. Jednak powyższy fragment kodu pokazuje przykład rodzaju kodu, który jest powszechnie używany do demonstracji kontrolki `DataGrid`. Arkusze stylów CSS są osadzone w kodzie i wielokrotnie powtarzane, co wynika ze stosowania przez ASP.NET atrybutów takich jak `ForeColor`, `BackColor` i innych. Ponadto pierwszy wiersz tabeli, który z założenia ma być nagłówkiem, to po prostu element `<td>` wraz z pogrubioną czcionką.

Zaprezentowany kod to powód do wstydu, ponieważ w rzeczywistości ASP.NET posiada funkcje pozwalające na przypisywanie klas CSS dla większości wartości. Ponadto w ASP.NET 2.0 jest obsługiwany znacznik `<th>` służący do generowania nagłówka w tabeli. Atrybut `CssClass` występuje w większości znaczników, a także istnieją inne, które pozwalają na przypisywanie klas znacznikom w wielu sytuacjach. Warto zwrócić uwagę na użycie w poniższym kodzie atrybutu `UseAccessibleHeader` kontrolki `DataGrid`. Źle się stało, że programiści muszą stosować wewnątrz kodu własne sposoby na zwiększenie poziomu jego dostępności, ale taka jest natura „bestii" — i na szczęście odpowiednie funkcje są dostępne w ASP.NET 2.0.

W zaprezentowanym poniżej fragmencie kodu dokonano drobnych modyfikacji kodu ASP.NET, które pozwoliły na zwiększenie jakości generowanych danych wyjściowych:

```
<asp:DataGrid ID="catalog" runat="server" AutoGenerateColumns="false"
   CssClass="grid-class" UseAccessibleHeader="true">
   <ItemStyle CssClass="item" />
   <AlternatingItemStyle CssClass="item-alt" />
```

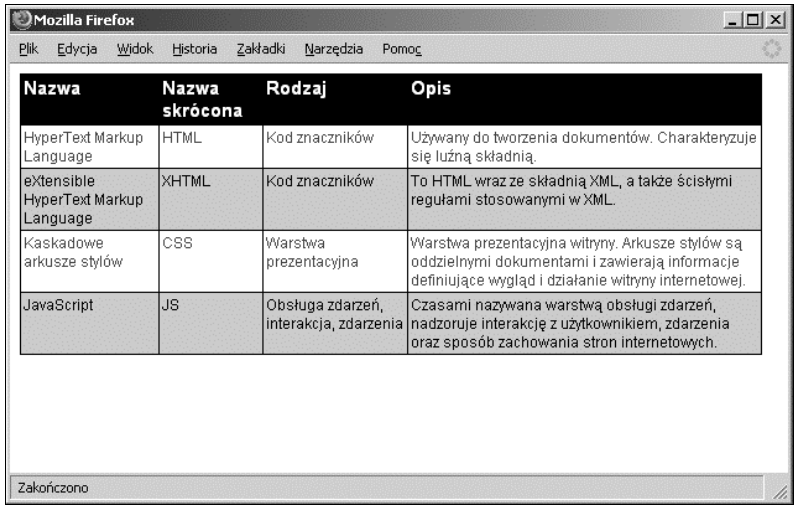
```
   <HeaderStyle CssClass="item-header" />
   <Columns>
      <asp:BoundColumn DataField="Name" HeaderText="Nazwa" />
      <asp:BoundColumn DataField="abbr" HeaderText="Nazwa skrócona" />
      <asp:BoundColumn DataField="typeof" HeaderText="Rodzaj" />
      <asp:BoundColumn DataField="notes" HeaderText="Opis" />
   </Columns>
</asp:DataGrid>
```

Do powyższego fragmentu kodu można dołączyć arkusz stylów CSS, na przykład taki:

```
.grid-class {
   border: 1px solid black;
   width: 600px;
   border-collapse:collapse;
}
.grid-class td {
   vertical-align: top;
   padding: 2px;
   border: 1px solid black;
}
.item {
   font: 12px arial, sans-serif;
   color: #800;
}
th {
   font: bold 15px arial, sans-serif;
   background-color: black;
   color: white;
   vertical-align: top;
   text-align: left;
   padding: 2px;
}
.item-alt {
   font: normal 12px arial, sans-serif;
   background-color: #ccc;
}
```

W takim przypadku wygenerowana tabela HTML będzie sama jak poprzednia (zobacz rysunek 4.4). Jednak użyty do jej utworzenia kod jest lepszy i bardziej zgodny ze standardami sieciowymi. Dane wyjściowe XHTML zostały przedstawione poniżej:

```
<table class="grid-class" cellspacing="0" rules="all" border="1"
id="catalog" style="border-collapse:collapse;">
   <tr class="item-header">
      <th scope="col">Nazwa</th><th scope="col">Nazwa skrócona</th>
      <th scope="col">Rodzaj</th><th scope="col">Opis</th>
```



Nazwa	Nazwa skrócona	Rodzaj	Opis
HyperText Markup Language	HTML	Kod znaczników	Używany do tworzenia dokumentów. Charakteryzuje się luźną składnią.
eXtensible HyperText Markup Language	XHTML	Kod znaczników	To HTML wraz ze składnią XML, a także ścisłymi regułami stosowanymi w XML.
Kaskadowe arkusze stylów	CSS	Warstwa prezentacyjna	Warstwa prezentacyjna witryny. Arkusze stylów są oddzielnymi dokumentami i zawierają informacje definiujące wygląd i działanie witryny internetowej.
JavaScript	JS	Obsługa zdarzeń, interakcja, zdarzenia	Czasami nazywana warstwą obsługi zdarzeń, nadzoruje interakcję z użytkownikiem, zdarzenia oraz sposób zachowania stron internetowych.

RYSUNEK 4.4. Ta sama kontrolka DataGrid w ASP.NET, ale po zastosowaniu usprawnień zwiększających jej zgodność ze standardami sieciowymi

```
   </tr>
   <tr class="item">
      <td>HyperText Markup Language</td><td>HTML</td><td>
      ↪Kod znaczników</td>
      <td>Używany do tworzenia dokumentów. Charakteryzuje się luźną
      ↪składnią.</td>
   </tr>
   <tr class="item-alt">
      <td>eXtensible HyperText Markup Language</td><td>XHTML</td>
      <td>Kod znaczników</td><td>To HTML wraz ze składnią XML, a także
      ↪ścisłymi regułami stosowanymi w XML.</td>
   </tr>
   <tr class="item">
      <td>Kaskadowe arkusze stylów</td><td>CSS</td><td>Warstwa
      ↪prezentacyjna</td>
      <td>Warstwa prezentacyjna witryny. Arkusze stylów są oddzielnymi
      ↪dokumentami i zawierają informacje definiujące wygląd
      ↪i działanie witryny internetowej.</td>
   </tr>
   <tr class="item-alt">
      <td>JavaScript</td><td>JS</td><td>Obsługa zdarzeń, interakcja,
      ↪zdarzenia</td>
      <td>Czasami nazywana warstwą obsługi zdarzeń, nadzoruje
      ↪interakcję z użytkownikiem, zdarzenia oraz sposób zachowania
      ↪stron internetowych.</td>
   </tr>
</table>
```

Warto zwrócić uwagę, że nowy kod zawiera znacznik <th> (kompletny wraz z atrybutem scope) klasy CSS dla większości elementów, łącznie z obsługą naprzemiennego koloru wierszy, oraz nie ma (w większości) osadzonych atrybutów prezentacyjnych. Mimo że powyższy kod na pewno nie jest doskonały, to jednak stanowi duże usprawnienie, a poprawienie danych wyjściowych wymagało drobnych korekt technicznych. Ten rodzaj modyfikacji wymaga pracy ze strony programistów interfejsu, którzy będą znali techniki pracy z kodem wewnętrznym. Albo programistów kodu wewnętrznego, którzy wiedzą, jak programować interfejs.

Znaczniki ASP.NET generujące dane wyjściowe na podstawie danych pochodzących z bazy danych posiadają znacznie więcej opcji. W niektórych sytuacjach zastosowanie innego znacznika może być znacznie lepszym rozwiązaniem, ponieważ zapewni on większy poziom kontroli nad generowanym kodem UI. Przykładowo: kontrolka Repeater stanowi doskonały wybór podczas tworzenia prostych danych wyjściowych, gdyż definiuje ona pętlę danych wyjściowych i używa małych szablonów kodu, w których programista może określić to, co powinno być wyświetlone:

```
<asp:Repeater id="catalog" runat="server">
<HeaderTemplate>
<table class="grid-class">
<tr>
   <th>Nazwa</th>
   <th>Nazwa skrócona</th>
   <th>Rodzaj</th>
   <th>Opis</th>
</tr>
</HeaderTemplate>

<ItemTemplate>
<tr class="item">
   <td><%#Container.DataItem("name")%> </td>
   <td><%#Container.DataItem("abbr")%> </td>
   <td><%#Container.DataItem("typeof")%> </td>
   <td><%#Container.DataItem("notes")%> </td>
</tr>
</ItemTemplate>

<AlternatingItemTemplate>
<tr class="item-alt">
   <td><%#Container.DataItem("name")%> </td>
   <td><%#Container.DataItem("abbr")%> </td>
   <td><%#Container.DataItem("typeof")%> </td>
   <td><%#Container.DataItem("notes")%> </td>
</tr>
```

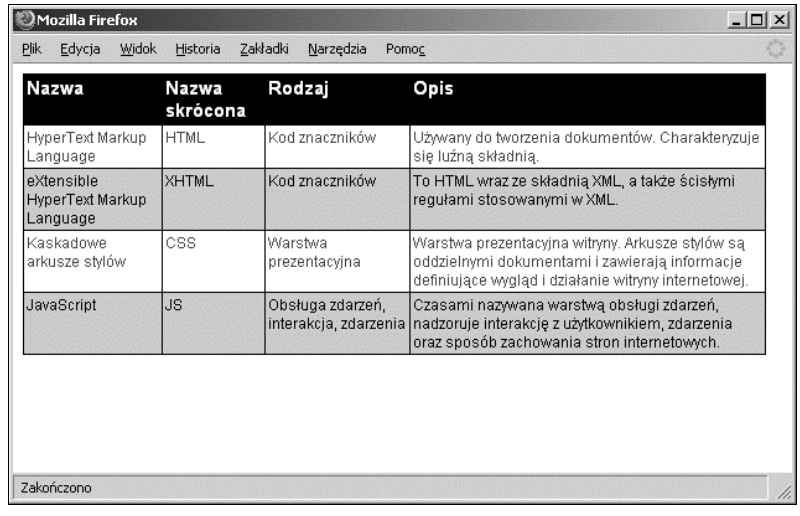
```
</AlternatingItemTemplate>

<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
```

Przedstawiony powyżej kod generuje tabelę HTML, która jest połączona z przedstawionym wcześniej arkuszem stylów CSS. Powstały kod jest czysty i pozbawiony wszelkich niepotrzebnych atrybutów prezentacyjnych. Chociaż kontrolka `Repeater` wymaga użycia nieco większej ilości kodu, to jednak daje programiście większą kontrolę nad sposobem generowania danych wyjściowych (zobacz rysunek 4.5):

```
<table class="grid-class">
<tr>
   <th>Nazwa</th>
   <th>Nazwa skrócona</th>
   <th>Rodzaj</th>
   <th>Opis</th>
</tr>
<tr class="item">
   <td>HyperText Markup Language</td>
   <td>HTML</td>
   <td>Kod znaczników</td>
   <td>Używany do tworzenia dokumentów. Charakteryzuje się luźną
   ↪składnią.</td>
</tr>
<tr class="item-alt">
   <td>eXtensible HyperText Markup Language </td>
   <td>XHTML</td>
   <td>Kod znaczników</td>
   <td>To HTML wraz ze składnią XML, a także ścisłymi regułami
   ↪stosowanymi w XML.</td>
</tr>
<tr class="item">
   <td>Kaskadowe arkusze stylów</td>
   <td>CSS</td>
   <td>Warstwa prezentacyjna</td>
   <td>Warstwa prezentacyjna witryny. Arkusze stylów są oddzielnymi
    ↪dokumentami i zawierają informacje definiujące wygląd
    ↪i działanie witryny internetowej.</td>
</tr>
<tr class="item-alt">
   <td>JavaScript</td>
   <td>JS</td>
   <td>Obsługa zdarzeń, interakcja, zdarzenia</td>
```

RYSUNEK 4.5. Użycie kontrolki Repeater ASP.NET w celu wygenerowania tej samej tabeli, ale z czystszym kodem, przy zachowaniu takiego samego wyglądu




```
   <td>Czasami nazywana warstwą obsługi zdarzeń, nadzoruje interakcję
   ↪z użytkownikiem, zdarzenia oraz sposób zachowania stron
   ↪internetowych.</td>
</tr>
</table>
```

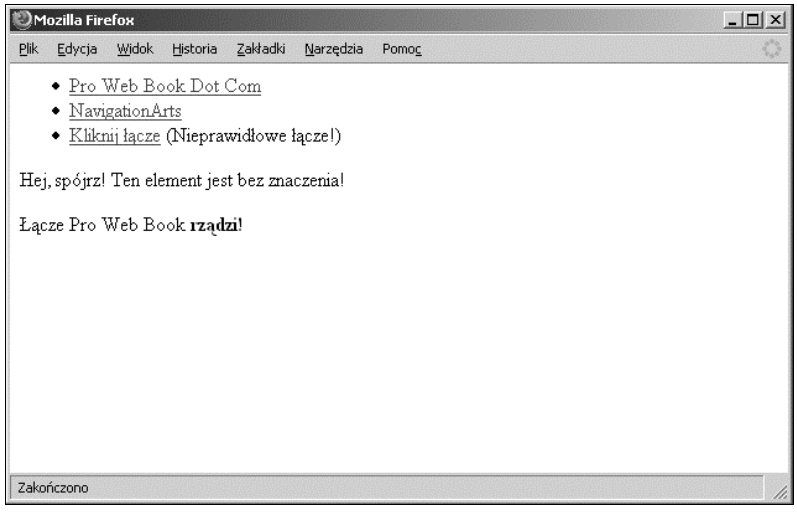
To jest przykład czystego kodu HTML wygenerowanego przez logikę ASP.NET działającą po stronie serwera. Czytelnik przekonał się więc, że rozsądne używanie funkcji aplikacji może zapewnić spory poziom kontroli nad sposobem generowania danych wyjściowych.

Przyglądając się kodowi bazującemu na znacznikach staje się oczywiste, że firma Microsoft przeniosła logikę aplikacji do innych plików, a używanie plików interfejsu *.aspx* z kodem znaczników nie jest złą koncepcją. Niektóre zespoły tworzące interfejsy będą mogły operować i modyfikować kod znajdujący się w plikach *.aspx* bez konieczności poznawania tajników języków C# lub Visual Basic.NET.

Kontrolki HTML ASP.NET, kontrolki Web Control i inne

Istnieją także inne obszary, na których platforma ASP.NET powszechnie generuje kod dla programistów. Zamiast używać prostych znaczników HTML, platforma ASP.NET zawiera funkcje pozwalające na wygenerowanie znaczników HTML. Ponadto zapewnia również dostęp do dokumentu po stronie serwera na sposób podobny do stosowanego w modelu DOM, choć z użyciem własnej składni. Warto przeanalizować przedstawiony poniżej kod oraz rysunek 4.6:

```
<script runat="server">
void Clicker(Object sender, EventArgs e)
{
   Response.Redirect("http://www.cherny.com");
```



RYSUNEK 4.6. Różne kontrolki serwerowe platformy ASP.NET mają możliwość dynamicznego generowania danych wyjściowych

```
}
void Page_Load()
{
   if (!IsPostBack)
   {
      // Kontrolki Web Control.
      myLink.Text = "Pro Web Book Dot Com";
      myLink.NavigateUrl = "http://www.prowebbook.com";
      myLink.CssClass = "webControl";

      // Kontrolki tworzące łącze HTML.
      myOtherLink.InnerText = "NavigationArts";
      myOtherLink.HRef = "http://www.navigationarts.com";
      myOtherLink.Attributes.Add("class","htmlControl");

      // Kontrolka tworząca etykietę.
      myLabel.Text = "Hej, spójrz! Ten element jest bez znaczenia!";

      // Kontrolka tworząca dosłowny tekst.
      myLiteral.Text = " (Nieprawidłowe łącze!)";

      // Kontrolka tworząca pierwszy akapit HTML.
      myPara.InnerHtml = "Łącze Pro Web Book " +
         " <strong>rządzi!</strong>";

      // Kontrolka tworząca drugi akapit HTML.
      myPara2.Visible = false;
   }
```

```
}
</script>
<form id="form1" runat="server">
<ul>
   <li><asp:HyperLink runat="server" id="myLink" /></li>
   <li><a runat="server" id="myOtherLink" /></li>
   <li><asp:LinkButton Text="Kliknij łącze"
      OnClick="Clicker" runat="server" /></li>
</ul>
<p><asp:Label id="myLabel" runat="server" /></p>
<p id="myPara" runat="server" />
<p id="myPara2" runat="server">Ten element będzie ukryty.</p>
</form>
```

Powyższy kod zawiera wiele elementów, na które warto zwrócić uwagę, prezentujących różne punkty widzenia:

- Całość kodu ASP.NET musi być umieszczona w ramach elementu `<form>` wraz z atrybutem `runat="server"`, co umożliwi użycie kontrolki `<asp:LinkButton />`. O ile będzie taka możliwość, to należy unikać stosowania wymienionej kontrolki. Wyjaśnienie zostanie przedstawione wkrótce podczas omawiania wynikowego kodu XHTML.
- Warto zwrócić uwagę, że w nieuporządkowanej liście formularza pierwszym znacznikiem serwerowym ASP.NET jest kontrolka `<asp:HyperLink />`. To kontrolka serwerowa Web Control, która generuje łącze HTML oraz zapewnia programowy dostęp do wszystkich atrybutów i tekstu znacznika, jak to zaprezentowano w bloku `<script>` znajdującym się nad odwołaniem do `myLink`.
- Warto także zauważyć, że drugie łącze nie posiada atrybutu `href` i również jest programowo kontrolowane z poziomu znajdującego się wcześniej bloku `<script>`. Ta kontrolka HTML działa po stronie serwera (wskazuje to atrybut `runat="server"`), który zapewnia programowy dostęp do atrybutów elementu. W ten sposób programista ma większą kontrolę nad danymi wyjściowymi, ale mniejszą nad funkcjami serwerowymi.

Dodanie atrybutu `ID` do kodu `runat="server"` powoduje, podobnie jak w przypadku dowolnego znacznika HTML, uzyskanie dostępu do jego znaczników. Takie rozwiązanie daje olbrzymie możliwości, ponieważ programista może kontrolować wyświetlanie, treść oraz atrybuty za pomocą jedynie kilku wierszy kodu umieszczonego w warstwie programowej aplikacji.

Obie kontrolki tworzenia łączy wirtualnie wykonują te same zadania, natomiast kontrolki Web Control dostarczają dodatkowych funkcji w stosunku do zwykłych kontrolek HTML. Ponadto kontrolki rodzaju Web Control są w mniejszym stopniu przewidywalne i zawierają atrybuty prezentacyjne, które programiści często wykorzystują w celu

szybkiego wprowadzenia drobnych zmian. Podczas używania kontrolek Web Control, programiści powinni ograniczyć się do stosowania atrybutów `CssClass` i unikać używania atrybutów prezentacyjnych. Aby osiągnąć większy poziom przewidywalności wygenerowanych danych wyjściowych, warto stosować kontrolki HTML, natomiast kontrolki Web Control tylko w określonych przypadkach, gdy jest to niezbędne.

Warto szczególnie dokładnie przyjrzeć się końcówce dokumentu:

```
<p><asp:Label id="myLabel" runat="server" /></p>
<p id="myPara" runat="server" />
<p id="myPara2" runat="server">Ten element będzie ukryty.</p>
```

Wszystkie kontrolki są wykonywane po stronie serwera.

Blok `<script>` jest kodem działającym po stronie serwera, który nie jest wysyłany do przeglądarki internetowej użytkownika i najczęściej znajduje się w oddzielnym pliku. Zadaniem tego kodu jest programowe wygenerowanie przedstawionego poniżej kodu XHTML (oprócz innego kodu .NET, który w tej chwili nie ma znaczenia podczas tych rozważań):

```
<script type="text/javascript">
<!--
var theForm = document.forms['form1'];
if (!theForm) {
   theForm = document.form1;
}
function __doPostBack(eventTarget, eventArgument) {
   if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
      theForm.__EVENTTARGET.value = eventTarget;
      theForm.__EVENTARGUMENT.value = eventArgument;
      theForm.submit();
   }
}
// -->
</script>
<ul>
   <li><a id="myLink" class="webControl" href="http://www.prowebbook.
   ↪com">Pro Web Book Dot Com</a></li>
   <li><a href="http://www.navigationarts.com" id="myOtherLink"
   ↪class="htmlControl">NavigationArts</a></li>
   <li><a href="javascript:__doPostBack('ctl00','')">Kliknij
   ↪łącze</a></li>
</ul>
<p><span id="myLabel">Hej spójrz! Ten element jest bez znaczenia!
↪</span></p>
<p id="myPara">Łącze Pro Web Book <strong>rządzi!</strong></p>
```

Lektura obowiązkowa podczas stosowania standardów sieciowych w ASP.NET

Podczas tworzenia kodu ASP.NET bardzo łatwo zbliżyć się do poziomu profesjonalnych standardów sieciowych. To jest po prostu ćwiczenie w podejmowaniu uzasadnionych i rozsądnych decyzji dotyczących stylu programowania.

Na witrynie MSDN (ang. *Microsoft Developers Network*) firma Microsoft zamieściła artykuł, którego celem jest pomoc programistom w używaniu standardów sieciowych oraz funkcji związanych z dostępnością podczas pracy w platformie ASP.NET. Ten artykuł powinien stać się lekturą obowiązkową dla wszystkich programistów ASP.NET. Niestety, jest on głęboko „zaszyty" na witrynie, a omawiane techniki nie zostały przedstawione w większości przykładów ASP.NET umieszczonych w Internecie. Wymieniony artykuł znajduje się na stronie:

http://msdn2.microsoft.com/en-us/library/aa479043.aspx

Wreszcie, firma Microsoft wypuściła na rynek kod typu open source opracowany na bazie możliwości rozbudowy ASP.NET, który pozwala programistom na zmianę danych wyjściowych XHTML z poziomu wielu znaczników ASP.NET wbudowanych w platformę ASP.NET. Wymieniony kod nosi nazwę „ASP.NET 2.0 CSS Friendly Control Adapters" i jest dostępny na stronie:

http://www.asp.net/cssadapters/

Przy odrobinie dodatkowego wysiłku, wygenerowany przez ASP.NET kod działający po stronie serwera może bardzo zbliżyć się do poprawnego (zgodnego ze standardami) kodu zapewniające duży poziom dostępności.

Dodatkowe spostrzeżenia dotyczące wygenerowanych danych wyjściowych są następujące:

- Warto zwrócić uwagę, że w ASP.NET istnieje kilka sposobów wygenerowania prostego łącza. Pierwsze dwa zostały przedstawione wcześniej — opierają się o kontrolki serwerowe i HTML. Kolejny to wykorzystanie jeszcze innej kontrolki o nazwie `<asp:LinkButton />`.
- Kod generujący trzecie łącze wyraźnie pokazuje, dlaczego w większości przypadków najlepszym rozwiązaniem jest uniknięcie stosowania kontrolki `<asp:LinkButton />`. Ta kontrolka tworzy atrybut `href` za pomocą łącza JavaScript, co łamie wszystkie zasady opisane w niniejszej książce.
- Zastosowana w pierwszym akapicie kontrolka `<asp:Label />` zupełnie bez powodu generuje znacznik `<span>`. Kontrolka `Label` jest uznawana za pożyteczną, ponieważ programista może przypisać jej pewne aspekty prezentacyjne. Zamiast niej możliwe jest jednak użycie kontrolki `<asp:Literal />`, która po prostu generuje uroczy komunikat: „(Nieprawidłowe łącze)". Bardzo często można spotkać się ze stwierdzeniem, że używanie tej kontrolki ma jedną dużą wadę, którą jest brak aspektów prezentacyjnych. To jest jednak powód, dla którego jej używanie jest doskonałym pomysłem.

- Za pomocą atrybutu `ID` (tutaj o wartości `myPara`) programista może również kontrolować tekst znaczników kontrolek HTML akapitu, jak to zademonstrowano na powyższym fragmencie kodu.
- Warto zwrócić uwagę, że programista może także wyświetlać lub ukrywać kontrolki HTML. Ostatni akapit w kodzie działającym po stronie serwera został ustawiony jako `visible = false`, a więc nie został wysłany do przeglądarki internetowej użytkownika.

Kontrolki HTML oferują kodowi ASP.NET duże możliwości, ale z różnych powodów samouczki dotyczące ASP.NET pomniejszają ich wykorzystanie. Jednak z perspektywy standardów sieciowych są to narzędzia o potężnych możliwościach i często bardziej pożądane niż kontrolki serwerowe Web Control.

Współpraca po stronie serwera z projektantami UI

Podczas opracowywania aplikacji działających po stronie serwera, programiści oraz projektanci powinni współpracować ze sobą tak blisko, na ile jest to możliwe, w celu zagwarantowania, że kod znaczników UI będzie stosował najlepsze rozwiązania zgodne ze standardami sieciowymi. Bardzo często oznacza to konieczność wyboru właściwego narzędzia po stronie serwera. Utrzymywanie UI z dala od aplikacji oraz używanie klas powoduje, że kod ma czystszą postać. W ten sposób kod związany z projektem wizualnym pozostaje w rękach zespołu UI oraz ogranicza się ryzyko, oraz potencjalną potrzebę angażowania programisty w celu wprowadzenia zmian w interfejsie użytkownika. Kiedy aplikacja staje się bardzo skomplikowana lub bazuje na podstawie starego kodu, konieczne mogą stać się kompromisy. Jednak ostateczne cele, standardy sieciowe i wskazówki powinny zakładać osiągnięcie spójności i utrzymywanie jej w długim okresie czasu.

Zarządzanie treścią

Kiedy witryna internetowa musi zostać uaktualniona przez użytkownika pozbawionego wiedzy technicznej, gdy rozrasta i staje się większa lub zawiera treść, która musi być ponownie użyta, skatalogowana, współdzielona i zapewniająca możliwość jej przeszukiwania, to najczęściej stosowanym rozwiązaniem jest oprogramowanie do zarządzania treścią. Wymienione oprogramowanie zarządzania treścią jest jedną z powszechnie dostępnych aplikacji sieciowych, na którą natknie się każdy zespół programistów sieciowych. Wynika to z faktu, że najczęstszym żądaniem ze strony użytkowników biznesowych jest coraz większa możliwość kontrolowania witryny.

Niestety, aplikacje służące do zarządzania treścią bardzo często generują własny język znaczników lub wymagają włożenia dużych nakładów pracy i wysiłku, aby wejść do świata standardów sieciowych. Na szczęście najczęściej nie jest to aż takie trudne, jak się początkowo wydaje, przynajmniej w przypadku porządnych aplikacji zarządzania

treścią. Szczerze mówiąc, najtrudniejszym etapem jest często określenie, gdzie należy szukać odpowiednich fragmentów kodu, które faktycznie odpowiadają za generowanie danych wyjściowych.

Dostępne są niezliczone systemy zarządzania treścią (CMS) o zróżnicowanym stopniu elastyczności i zgodności ze standardami sieciowymi. Z punktu widzenia standardów sieciowych, elastyczność zależy od niuansów związanych z możliwościami oprogramowania, jak również implementacją, za którą odpowiada już zespół programistów.

Podstawy systemu zarządzania treścią

Lepsze rozwiązania z zakresu systemów CMS pozwalają zespołowi na wygenerowanie kodu UI w ustalony przez niego sposób — czyli wprost przeciwnie do rozwiązań, w których to narzędzie decyduje o postaci kodu. Istnieje kilka sposobów, na które standardy sieciowe mogą współpracować z rozwiązaniami z dziedziny systemów zarządzania treścią:

- Abstrahowanie warstwy prezentacyjnej treści od miejsca przechowywania tej treści.
- Użycie mniejszej liczby szablonów CMS, ale efektywniejsze wykorzystanie arkuszy stylów CSS.
- Ponowne używanie treści ponieważ kod znaczników nie będzie powiązany z warstwą prezentacyjną.
- Uproszczenie zadań autorów tworzących treść ponieważ będą musieli obsługiwać mniejszą ilość aspektów prezentacyjnych.
- Znaczne zwiększenie możliwości przeprojektowania witryny przy jednoczesnym zmniejszeniu wysiłku ze strony zespołu obsługującego system CMS.

System zarządzania treścią i czysta treść

Prosty proces przechowywania treści w centralnym repozytorium wraz z czystym kodem znaczników jest sam w sobie dobrym sposobem na ponowne używanie treści, ponieważ kod znaczników będzie prosty. Ponadto na różnych fragmentach stron, w zależności od kontekstu, będzie można stosować style za pomocą arkuszy stylów CSS. Oprócz tego, poniżej wymieniono niektóre z najlepszych praktyk oraz rozwiązań stosowanych w profesjonalnych systemach zarządzania treścią i projektowania z użyciem arkuszy stylów CSS:

- Projektowanie z użyciem arkuszy stylów CSS na podstawie kontekstu. Przykładowo, opis artykułu może być wygenerowany jako nagłówek `<h1>` stosujący na stronie artykułu dużą czcionkę w kolorze brunatnym, ale jednocześnie małą czcionkę w kolorze czarnym na stronie archiwum. Arkusze stylów CSS pozwalają na kontrolę tego rodzaju różnic.

- Treść powinna być przechowywana w najprostszej z możliwych postaci. Należy stosować minimalną, wystarczającą liczbę klas CSS oraz minimalną (lub zerową) liczbę atrybutów prezentacyjnych elementów.
- Warto stosować semantyczny kod znaczników, oznaczając treść w pełni zrozumiałym, czytelnym i przenośnym formatem, którego można ponownie używać i nadawać mu styl. Należy sobie przypomnieć, że doświadczeni programiści CSS mogą zastosować różne style na podstawie kontekstu w szablonie witryny za pomocą atrybutu `ID` lub klasy.
- Autorzy treści powinni stosować jedynie podstawowe znaczniki HTML struktury strony. Idealnym rozwiązaniem będzie, gdy autorzy nauczą się kilku klas CSS. Powinni również wiedzieć, że nagłówek pierwszego stopnia w obszarze treści to `<h1>`. Ponadto powinni też wiedzieć, że utworzenie zwykłej listy wypunktowanej skutkuje wyświetlaniem małych ikon wypunktowania. Muszą też wiedzieć, że na przykład nagłówek w obszarze treści po prawej stronie będzie wyświetlany w kolorze granatowym. Warstwę prezentacyjną można nałożyć za pomocą arkuszy stylów CSS i nie trzeba będzie jej przechowywać w repozytorium treści systemu CMS. Ogólnie rzecz biorąc, autorzy treści mogą skoncentrować się na treści, a nie jej wizualnych aspektach.
- Kontekst na danej stronie jest użytecznym narzędziem, ale kontekst na witrynie może być równie ważny. W chwili obecnej często stosowanym rozwiązaniem jest podział witryny na kilka sekcji, które stosują odmienne schematy kolorów wyłączone z podstawowej marki. Przechowywana w bazie danych treść sama w sobie nie jest w stanie tego odróżnić, ale powinna być przenośna między sekcjami witryny. Style CSS można definiować w stosunku do znacznika `<body>` lub zbioru atrybutów `ID` sekcji witryny, a więc w ten sposób stosować różne schematy kolorów na wszystkich semantycznych znacznikach dla danego fragmentu witryny.

Dane wyjściowe systemu zarządzania treścią i moduły

Narzędzia zarządzania treścią mogą stanowić wyzwanie, gdy trzeba będzie określić, w jaki sposób kontrolować dane wyjściowe oraz otrzymać kod zgodny ze standardami sieciowymi. Bardzo często najtrudniejszą częścią jest określenie punktu rozpoczęcia pracy oraz *sposobu* kontrolowania danych wyjściowych. Zazwyczaj system CMS generuje stronę sformatowaną za pomocą kodu pochodzącego z kilku różnych obszarów:

- Wbudowane moduły generujące dane wyjściowe na podstawie własnych funkcji lub funkcji dostępnych w systemie CMS.
- Szablony stron używane do wyświetlania różnego rodzaju stron.
- Bazujące na przeglądarce internetowej edytory podobne do procesorów tekstu (używane przez autorów treści).

W kwestii wygenerowania poprawnego kodu zgodnego ze standardami sieciowymi, największymi wyzwaniami związanymi z aplikacją CMS są najczęściej jej wbudowane funkcje, nad którymi zespół programistów ma małą kontrolę lub w ogóle jej nie ma. Na te funkcje należy szczególnie uważać. Jednak niektórzy mogą uważać je za doskonałe funkcje narzędzia, gdyż rzekomo po jego instalacji witryna internetowa będzie gotowa „od ręki".

Ryzykowne elementy obejmują między innymi:

- Moduły administracyjne osadzone na stronach dostępnych publicznie.
- Kod służący do śledzenia ruchu oraz wbudowane skrypty.
- Moduły reklamowe.
- Wszystkie elementy generujące menu.
- Kontrolki specjalne, które mogą generować listy treści (na przykład nowości bądź archiwum produktów).
- Wyniki wyszukiwania

WSKAZÓWKA

Wszystko określane mianem „dostępne od ręki" z natury jest podejrzane, o ile narzędzie nie zostało wcześniej przystosowane do spełniania nowoczesnych standardów.

Narzędzia o potężnych możliwościach pozwalają zespołowi programistów na dostęp do kodu odpowiedzialnego za generowanie danych wyjściowych lub wstawianie własnych modułów, które mogą zastąpić bądź rozszerzyć wbudowane funkcje. W idealnej sytuacji narzędzie będzie zawierało wbudowane szablony lub fragmenty kodu, które można uaktualnić. Szczególnie należy uważać na narzędzia pozwalające na dostosowanie wyglądu i sposobu działania za pomocą czegoś na wzór panelu sterowania, o ile faktycznie to narzędzie nie udostępni kodu, który będzie można przejrzeć i zmodyfikować.

Szablony systemu zarządzania treścią

Większość narzędzi CMS wiąże strony z szablonami, z których każdy jest układem graficznym możliwym do ponownego użycia. Autorzy po prostu wybierają właściwy szablon dla danego rodzaju lub sekcji strony. Im większa liczba szablonów, tym więcej elementów autor musi śledzić — a obsługa witryny staje się coraz bardziej trudniejsza. Rozsądne użycie arkuszy stylów CSS i standardów sieciowych może w rzeczywistości doprowadzić do zmniejszenia liczby szablonów.

Szablony zazwyczaj opierają się na siatce strony, co zwykle oznacza stosowanie różnego kodu znaczników. W świecie standardów sieciowych oraz arkuszy stylów CSS niekoniecznie musi tak być.

Przykładowo, warto przeanalizować przedstawiony poniżej układ strony używającej trzech kolumn (zobacz rysunek 4.7):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
   <title>Pierwszy szablon</title>
   <link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body class="typeA">

<div id="wrapper">
   <ul id="nav">
      <li><a href="...">Nawigacja 1</a></li>
      <li><a href="...">Nawigacja 2</a></li>
      <li><a href="...">Nawigacja 3</a></li>
      <li><a href="...">Nawigacja 4</a></li>
   </ul>
   <div id="content">
      <h1>Obszar treści</h1>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing
         elit. Nam sit amet nulla. Ut ut urna ac lectus.</p>
      <p>Ut ut urna ac lectus tincidunt sollicitudin. Sed rutrum
         interdum lorem. Integer aliquam pellentesque
         neque.</p>
      <p>Curabitur a neque a libero gravida dignissim. Sed
         eget tellus.</p>
   </div>
   <div id="related">
      <h2>Łącza do innych stron</h2>
      <p>Łącza do innych stron i treści</p>
      <ul>
         <li><a href="">Sekcja 1</a></li>
         <li><a href="">Sekcja 1</a></li>
      </ul>
   </div>
</div>

</body>
</html>
```

Przedstawiony na powyższym fragmencie kodu dokument ma trzy elementy `<div>` o następujących wartościach atrybutów `ID`: `nav`, `content` i `related` (czyli z punktu widzenia selektorów CSS będą to `#nav`, `#content` i `#related`). Wymienionym elementom można bardzo łatwo nadać styl za pomocą arkuszy stylów CSS. Treść znajdująca się

RYSUNEK 4.7. Szablon systemu CMS tworzący układ, który opiera się na trzech kolumnach

w elementach `<div>` o nazwach `#content` i `#related` może być tworzona przez autorów treści. Na powyższym fragmencie kodu są one przedstawiane przez sekcje odpowiednio „Lorem ipsum" oraz „Łącza do innych stron". Co więcej, łącza znajdujące się w trzeciej kolumnie mogą być nawet generowane dynamicznie po stronie serwera na podstawie pokrewieństwa z daną treścią. Dlatego też mogą zdarzyć się sytuacje, gdy trzecia kolumna nie będzie potrzebna.

Warto zwrócić uwagę na klasę elementu `<body>`. Autor treści może wykorzystać ją w celu zaznaczenia szablonu i umieszczenia w kolejce odpowiedniej liczby kolumn. Logika programu może z kolei używać klasy elementu `<body>` dokumentu, aby utworzyć strukturę strony składającej się na przykład z nawigacji i pojedynczej kolumny z treścią (zobacz rysunek 4.8). W takim przypadku w narzędziu CMS nie trzeba będzie tworzyć większej liczby szablonów, a autorzy treści nie będą musieli wybierać różnych szablonów.

RYSUNEK 4.8. Po wprowadzeniu drobnych zmian w arkuszu stylów CSS, ten sam szablon tworzy stronę wyświetlającą tylko dwie kolumny

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
   <title>Drugi szablon</title>
   <link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body class="typeB">
<div id="wrapper">
   <ul id="nav">
      <li><a href="...">Nawigacja 1</a></li>
      <li><a href="...">Nawigacja 2</a></li>
      <li><a href="...">Nawigacja 3</a></li>
      <li><a href="...">Nawigacja 4</a></li>
   </ul>
   <div id="content">
      <h1>Obszar treści</h1>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing
         elit. Nam sit amet nulla. Ut ut urna ac lectus.</p>
      <p>Ut ut urna ac lectus tincidunt sollicitudin. Sed rutrum
         interdum lorem. Integer aliquam pellentesque
         neque.</p>
      <p>Curabitur a neque a libero gravida dignissim. Sed
         eget tellus.</p>
   </div>
   <div id="related">
   </div>
</div>

</body>
</html>
```

Element z atrybutem ID o wartości #related został zwinięty, a więc nie generuje danych wyjściowych. Na poziomie systemu CMS zmiana klasy elementy <body> na typeB powoduje zmianę układu strony, ale bez konieczności modyfikacji kodu znaczników. Oznacza to, że w systemie można zrezygnować z jednego dodatkowego szablonu, który w przeciwnym razie byłby potrzebny. W ten sposób zachowana zostaje również separacja między kodem znaczników i warstwą prezentacyjną. Arkusz stylów CSS powyższej strony jest następujący:

```
body { font: normal .9em Georgia; }
body.typeB #related { display:none; }
body.typeB #content { width: 600px; }
#nav { list-style-type: none; width: 100px; float: left;
   margin: 30px 0 0 5px; padding: 0; }
```

```
#content { width: 400px; float: left; margin-left: 10px;
   border: 1px solid red; padding: 3px; }
#related { width: 150px; float: left;
   border: 1px solid red; margin-left: 10px;
   padding: 3px; }
```

Narzędzia służące do zarządzania treścią są po prostu innego rodzaju mechanizmem oprogramowania obsługującym witrynę internetową. W rzeczywistości są powszechnie stosowane na witrynach dużych organizacji, a większość z nich to w pełni wyposażone platformy programistyczne, które można zmusić do wygenerowania danych wyjściowych zgodnych ze standardami sieciowymi. W ten sposób witryna będzie w większym stopniu dostępna. Niezależnie od tego, czy narzędzie powoduje wygenerowanie czystych danych wyjściowych, czy są one przekształcane z formatu XML za pomocą języka XSLT (ang. *eXtensible Stylesheet Language Transformations*), dane wyjściowe powinny być czyste i przemyślane. Warto podkreślić, że efektywne stosowanie standardów sieciowych może prowadzić również do redukcji liczby używanych szablonów CMS oraz znacznie ułatwić pracę autorom treści.

Tryb WYSIWYG dla autorów treści

Podczas pracy w środowiskach zajmujących się treściami publikowanymi w Internecie, większość autorów treści nieposiadających umiejętności technicznych korzysta z pewnej formy interfejsu WYSIWYG podczas tworzenia danej treści. Tego rodzaju narzędzia mieszczą się w szerokim zakresie oprogramowania, na przykład edytor, taki jak Adobe Contribute, służący do prostej obsługi witryny. Nie są one typowymi platformami programistycznymi, takimi jak Adobe Dreamweaver lub Microsoft Expression Web, i posiadają znacznie uboższy zestaw funkcji. Podobnie jak w przypadku każdego innego rodzaju oprogramowania, wymienione narzędzia mają opcje konfiguracyjne pozwalające na osiągnięcie różnego poziomu zgodności ze standardami sieciowymi.

Ogólna zasada jest taka, że najlepsza konfiguracja edytora dla autora treści jest wtedy, gdy większość opcji formatowania zostaje wyłączona. Wynika to z faktu, że funkcje formatowania zwykle polegają na kodzie, którego zespół programistów nie będzie w stanie kontrolować. Efektywna obsługa arkuszy stylów CSS przez te narzędzia jest więc kluczowa.

Edycja za pomocą przeglądarki internetowej

Edytory bazujące na przeglądarce internetowej są dostępne już od pewnego okresu czasu. Firma Microsoft wprowadziła komponent edycyjny na bazie technologii Active-X już w przeglądarce Internet Explorer 4. Od tego czasu dodano obsługę języka JavaScript, a edytory zwykle bazują na skryptach, języku Java, technologiach Flash lub Active-X. Coraz częściej do obsługi edycji w przeglądarce internetowej są dostępne komponenty wykorzystujące skrypty i działają one w większości nowoczesnych przeglądarek internetowych, takich jak Safari i Opera. Ze względu na zachowanie zgodności warto wybrać jeden z tego rodzaju komponentów.

Niestety, standardowo edytory bazujące na przeglądarce internetowej nie są wystarczająco solidne. Przeglądarka Microsoft Internet Explorer generuje znaczniki `<font>`, podczas gdy Mozilla generuje osadzone style. Edytory mogą wygenerować to, co potrafi wygenerować przeglądarka internetowa, ale funkcje edytorów muszą w olbrzymim stopniu zostać zbudowane na bazie kilki sztuczek dostępnych w modelu DOM przeglądarki. Z tego powodu niezmiernie trudno jest na masową skalę opracować własny edytor, ale każdego dnia ich jakość ulega poprawie. Czytelnik na pewno znajdzie nowoczesny edytor, który będzie generował poprawny kod i obsługiwał arkusze stylów CSS.

Konfiguracja edytora

Programiści sieciowi powinni powziąć odpowiednie kroki, aby edytory bazujące na przeglądarce internetowej nie zniweczyły ich ciężkiej pracy włożonej w definiowanie stylów i programowanie witryn internetowej zgodnie ze standardami sieciowymi. Kroki konieczne do podjęcia mogą obejmować między innymi integrację nowego użytkownika lub nawet nowego oprogramowania.

Pod uwagę należy wziąć następujące kroki:

- Przejrzenie kodu danych wyjściowych edytora WYSIWYG w różnych sytuacjach. Generowanie przez tego rodzaju edytory niepoprawnego kodu jest dość powszechne, ale rynek nadal się rozwija i wiele z nich potrafi wygenerować prawidłowe dane HTML lub nawet XHTML, po wprowadzeniu niewielu zmian konfiguracyjnych.
- Niektóre edytory oferują widok wyświetlający kod źródłowy. W zależności od poziomu umiejętności autora treści, taką funkcję trzeba będzie włączyć lub wyłączyć. Część edytorów używa modelu opartego na prawach dostępu, co umożliwia włączenie widoku wyświetlającego kod źródłowy niektórym użytkownikom i wyłączenie go innym.
- Funkcje kontrolujące aspekty prezentacyjne powinny być ograniczone do minimum. Należy wyłączyć funkcje modyfikujące rodzinę czcionki, kolor czcionki i kolor tła. Te parametry powinny być definiowane jedynie w arkuszu stylów CSS.
- Edytor powinien mieć możliwość stosowania klas CSS. Dobry edytor będzie potrafił obsłużyć połączenie pliku CSS z edytorem. Niektóre będą wymagały, aby programista skonfigurował elementy wyświetlane w menu klas CSS. Najlepsze edytory będą obsługiwały kontekst oraz pozwalały na używanie jedynie niektórych klas aplikacji, w zależności od reguł CSS — na przykład niedozwolone będzie użycie reguły `p.error` w elemencie `<span>`.
- Obsługa klas CSS aplikacji powinna obejmować możliwości w zakresie stosowania CSS określonym elementom. W ten sposób będzie można bardzo łatwo zastosować klasę na elemencie `<ul>`, jak również na zagnieżdżonych w nich elementach `<li>`, na podstawie pewnej formy wyboru opcji. Częstym sposobem realizacji takiego zadania jest proste kliknięcie i zaznaczenie drzewka modelu DOM na pasku stanu (`body > div > ul > li > a`).

- Pliki arkuszy stylów CSS powiązane z edytorem mogą być wyodrębnione z głównych plików CSS, ponieważ zawarte w nich reguły mogą okazać się zbyt skomplikowane do interpretacji. Poza tym pełny kontekst reguł, na przykład elementy <p> wewnątrz `#content` zamiast `#related`, może być nieobsługiwany lub po prostu niepotrzebny. W zależności od kontekstu edycji, rozwiązaniem może być użycie wielu plików CSS.
- Edytor powinien obsługiwać podstawowe znaczniki semantycznego XHTML, takie jak wbudowane nagłówki (1-6), formatowanie akapitu, co najmniej dwa rodzaje wypunktowania, cytowanie, preformatowany tekst i adresy.
- Dobry edytor będzie również usuwał śmieci oraz niepoprawne znaczniki z treści wklejanej ze schowka bądź będzie zawierał wiele opcji służących do oczyszczania wklejonej treści. Bardzo często treść skopiowana z procesora tekstu lub strony internetowej i wklejona do edytora WYSIWYG zachowuje oryginalne informacje dotyczące formatowania. Tego rodzaju informacje prowadzą do wygenerowania niepoprawnego kodu i mogą zmodyfikować style, które powinny być ustalane jedynie poza treścią poprzez arkusze stylów CSS.
- Być może trzeba będzie poinformować autorów treści, że jeśli stosowana przez nich aplikacja nie potrafi usunąć niepoprawnych znaczników z wklejanej treści, to powinni je usuwać poprzez wklejenie treści do zwykłego edytora tekstowego, a dopiero następnie do edytora WYSIWYG.
- Idealnym rozwiązaniem jest znalezienie edytora bazującego na przeglądarce internetowej, który będzie obsługiwał jak największą liczbę przeglądarek i oferował wymienione powyżej funkcje. Znalezienie edytora WYSIWYG, który okazuje się opracowany jedynie pod kątem przeglądarki IE, nie jest wcale takie rzadkie. Jednak obecnie dostępne edytory są dostępne zarówno dla platformy Windows, jak i Mac oraz niemalże każdej przeglądarki internetowej.

Stosowanie wymienionych powyżej reguł oraz kryteriów może oznaczać widoczną różnicę w tworzeniu solidnych, zgodnych ze standardami sieciowymi witryn. Użycie niewłaściwego edytora treści może zniweczyć ciężką pracę włożoną w opracowanie witryny. Jakość i wydajność dowolnego edytora stosowanego w środowisku produkcyjnym należy oceniać pod kątem standardów sieciowych i modyfikacji prowadzących do uzyskania kodu, który będzie w jak największym stopniu poprawny.

Firmy trzecie

Większe witryny internetowe często stosują określony wygląd i sposób działania, również na witrynach firm trzecich bądź aplikacji biznesowych hostingowanych gdzie indziej, na przykład na witrynie Investor Relations lub witrynach zajmujących się pośrednictwem w znalezieniu pracy. Te witryny będą posiadały wygląd zgodny z wyglądem witryny podstawowej, a użytkownik może się nawet nie zorientować, że został przeniesiony na inną witrynę.

Podejście zakładające stosowanie standardów sieciowych jest idealne w tego rodzaju sytuacjach, ponieważ nie tylko grafika i skrypty mogą być przechowywane na głównych

serwerach organizacji, ale także wszystkie pliki CSS lub przynajmniej te określające ogólny wygląd i działanie witryny. W takich sytuacjach organizacja może udostępnić firmom trzecim łącza do niektórych lub wszystkich elementów (na przykład plików CSS), a następnie usprawniać je przez cały czas, bez konieczności wprowadzania modyfikacji także w aplikacjach firm trzecich.

W tego rodzaju przypadkach prawdopodobnie najlepszym rozwiązaniem będzie przekazanie firmom trzecim dokumentacji wraz z opisem klas stylów, nagłówkami grafik oraz ogólnymi wskazówkami w zakresie układu witryny. Warto również przekazać przykładowe fragmenty kodu oraz szablony przedstawiające przykład budowania stron i układu witryny.

Jednak podobnie jak w każdej innej organizacji lub platformie oprogramowania, firmy trzecie będą się różniły stopniem stosowania i zgodności ze standardami sieciowymi. Przykładowo: ich platformy mogą być niezgodne z arkuszami stylów opracowanymi w organizacji. W takich przypadkach pewne kompromisy staną się koniecznością. Kompromis może oznaczać wyodrębnienie pewnego fragmentu z pliku CSS lub utworzenie oddzielnych szablonów kodu znaczników. Jednak w perspektywie dłuższego okresu czasu, warto rozważyć i przeanalizować zmianę firmy trzeciej, jeżeli nie będzie ona w stanie nadążyć za rozwiązaniami przemysłowymi.

W jaki sposób podejść do aplikacji sieciowych?

Podejście zakładające używanie standardów sieciowych oznacza ścisłe oddzielenie warstwy prezentacyjnej od treści, co pociąga za sobą także separację logiki biznesowej od warstwy prezentacyjnej. Zespoły programistów budujące oprogramowanie oraz aplikacje sieciowe mogą czerpać duże korzyści, jeśli tworzone przez nich interfejsy będą opierały się na nowoczesnym podejściu, czyli arkuszach stylów CSS zamiast osadzonych atrybutów prezentacyjnych.

Mniejsza ilość osadzonego kodu prezentacyjnego oznacza zmniejszenie ryzyka dla logiki biznesowej podczas modyfikacji projektu witryny lub oprogramowania. Ponadto, w środowisku potencjalnie skomplikowanego oprogramowania, takie podejście może oznaczać, że doświadczeni projektanci interfejsu będą mogli wprowadzać zmiany w układzie bez konieczności uaktualniania kodu wewnętrznego i angażowania programistów kodu wewnętrznego.

Prawdziwym wyzwaniem jest określenie miejsca rozpoczęcia aktualizacji aplikacji sieciowej i poznanie funkcji dostępnych technologii. Nowe funkcje technik wykorzystujących standardy sieciowe mogą stać się potężnym sojusznikiem zespołu zarówno budowania aplikacji sieciowej od samego początku, jak i podczas używania komercyjnych narzędzi służących do zarządzania treścią.