Tworzenie stron WWW. Praktyczny kurs

Autor: Marcin Lis
ISBN: 978-83-246-2487-4
Format: 158×235, stron: 384

Naucz się wreszcie samodzielnie tworzyć strony internetowe!

- Jakie techniki koniecznie trzeba poznać, by własnoręcznie projektować witryny WWW?
- Co zrobić, by wszystkie elementy strony znalazły się na swoich miejscach?
- Jak nadać stronie indywidualny charakter i opublikować ją w sieci?

O projektowaniu witryn internetowych napisano już tyle książek, że można by z nich stworzyć osobną bibliotekę. Rzecz jednak w tym, że większość z nich skierowana była do informatyków, a więc ludzi posługujących się specyficznym żargonem i zorientowanych w temacie. Ponadto strony powstałe kilka lat temu nie mają zbyt wiele wspólnego z tymi powstającymi dziś, a techniki ich tworzenia także są już zupełnie inne. Jeśli więc nigdy dotąd nie próbowałeś stworzyć własnej witryny albo zniechęciły Cię kiedyś związane z tym komplikacje, powinieneś czym prędzej sięgnąć po ten poradnik. Zdziwisz się, jak proste i przyjemne może być zbudowanie własnej strony WWW.

Książka „Tworzenie stron WWW. Praktyczny kurs" przeznaczona jest dla początkujących adeptów sztuki tworzenia witryn internetowych – jedynym wymaganiem jest umiejętność w miarę sprawnego posługiwania się komputerem i pracy z dowolnym edytorem tekstu. Wszystkie potrzebne techniki czy zasady formatowania stron zostały tu opisane i zaprezentowane w praktyce. Podczas lektury oraz wykonywania ćwiczeń dowiesz się, jak opracować strukturę strony, a także wykorzystać znaczniki, listy, elementy XHTML, obrazy, odnośniki, multimedia, formularze itp., by stworzyć naprawdę interesującą, nowoczesną i funkcjonalną witrynę internetową.

- Podstawowa struktura strony WWW
- Strona WWW w internecie
- Struktura (X)HTML i podstawowe znaczniki
- Listy, tabele, obrazy i odnośniki
- Kontrola nad tekstem i multimedia na stronie WWW
- Formularze i formatowanie strony za pomocą tabel
- Style CSS
- Własności czcionek i formatowanie tekstu
- Marginesy, obrysy i obramowania
- Wielkości i pozycje elementów witryny
- Sposoby wyświetlania elementów strony
- Układ strony generowany przez CSS

Weź swoją stronę we własne ręce – nikt nie zrobi jej lepiej od Ciebie!

Spis treści

Rozdział 1.

Podstawy

Rozdział pierwszy rozpoczyna naszą przygodę z tworzeniem stron WWW. Składa się z trzech lekcji, w których omówione zostały podstawy niezbędne do dalszej nauki. Zobaczymy, jak wygląda struktura strony WWW, z czego ona się składa i jak w najprostszy sposób wyświetlić zwykły napis. Omówione zostaną również narzędzia pomocne przy tworzeniu witryny i przydatne edytory, dzięki którym ominą nas m.in. problemy z wyświetlaniem polskich znaków. W lekcji 2. zostaną przedstawione sposoby umieszczania strony WWW w internecie, tak by była dostępna dla wszystkich zainteresowanych. Temat ten może być jednak przez początkujących pominięty, gdyż testowanie tworzonych witryn można bez problemów przeprowadzać na domowym komputerze przy użyciu wyłącznie przeglądarki WWW.

Lekcja 1. Pierwsze kroki

- Z czego składa się strona WWW?
- Jak otworzyć plik w przeglądarce?
- Co to jest HTML i XHTML?
- Jakiego języka opisu strony używać?
- Jaka wygląda struktura strony napisanej w HTML-u?
- Co to jest kod źródłowy?
- W jaki sposób uniknąć problemów z polskimi literami?
- Czym jest prolog dokumentu HTML?
- Jakie edytory tekstowe wspomagają tworzenie witryn?
- Czemu warto testować witryny w różnych przeglądarkach?

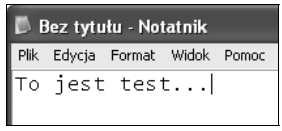
Wczytanie dokumentu do przeglądarki

Gdy przeglądamy strony WWW, widzimy na nich wiele rozmaitych danych: tekst, grafiki, zdjęcia, animacje, dynamiczne menu itp. Tworzy to wrażenie dużej złożoności. Jednak podstawowa strona WWW to zwyczajny dokument tekstowy. Wszystko inne to tylko dodatki umieszczone w osobnych plikach. Oczywiście nie jest to taki dokument, jaki piszemy np. w Wordzie. Ma on swoją strukturę i specjalne znaczniki, dzięki którym przeglądarka wie, jak ma wyświetlić zawartą w nim treść. Tę strukturę i te znaczniki musimy utworzyć sami. To właśnie zadanie każdego twórcy stron WWW.

Jak w najprostszy sposób wyświetlić coś w przeglądarce? To proste. Wystarczy utworzyć zwyczajny dokument tekstowy, wpisać w nim dowolną treść i otworzyć go w Firefoksie, Internet Explorerze bądź innym programie tego typu. Co rozumiemy przez „dokument tekstowy"? To plik zawierający tylko tekst. Można go utworzyć za pomocą dowolnego edytora tekstowego. W systemie Windows najlepiej użyć dostępnego standardowo Notatnika (z menu *Start* wybierz *Programy* [bądź *Wszystkie programy*], *Akcesoria* i *Notatnik*).

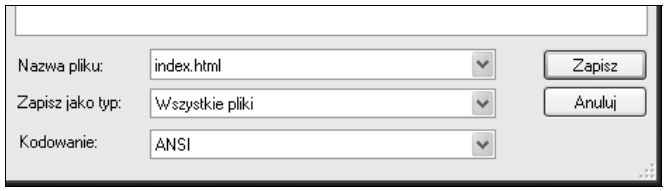
W edytorze należy wpisać dowolną treść (rysunek 1.1), a następnie zapisać plik, wybierając z menu *Plik* pozycję *Zapisz*.

Rysunek 1.1. *Testowy wpis w edytorze tekstowym*



W oknie wyboru nazwy pliku wpisujemy *index.html* lub dowolną inną nazwę (rysunek 1.2). Ważne, aby rozszerzenie nazwy pliku miało postać *html* (lub *htm*, jednak *html* jest lepszym rozwiązaniem[1]). Z listy *Zapisz jako typ* należy wybrać pozycję *Wszystkie pliki*[2], a pole wyboru *Kodowanie* pozostawić bez zmian. Po kliknięciu przycisku *Zapisz* dokument zostanie zapisany i zostanie mu nadane rozszerzenie *html*.

Rysunek 1.2. *Parametry zapisywanego pliku*



[1] Prawidłowym rozszerzeniem plików zawierających dokumenty typu HTML (czyli treść stron WWW) pierwotnie było *.html*. Problem powstał, gdy takie pliki trzeba było zapisać w systemach DOS i opartych na nich wczesnych wersjach Windowsa, które dopuszczały maksymalnie trzyliterowe rozszerzenia. Dlatego też oprócz *html* zaczęło funkcjonować również rozszerzenie *htm*. W dzisiejszych jednak czasach każdy popularny system operacyjny oferuje rozszerzenia plików o długości większej niż 3 znaki, dlatego należy stosować rozszerzenie *html*.

[2] Jeśli w polu *Zapisz typ jako* pozostanie domyślna opcja *Dokument tekstowy*, do wprowadzonej nazwy pliku edytor doda rozszerzenie *txt*. Wtedy, jeśli np. wprowadzoną nazwą było *index.html*, na dysku zostanie zapisany plik o nazwie *index.html.txt*. Dlatego też istnieje konieczność zmiany tej opcji na wskazaną w tekście.

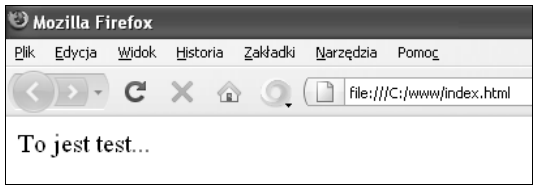
Miejsce zapisania pliku jest dowolne. Można go umieścić nawet na pulpicie. Lepiej jednak przygotować specjalny katalog, w którym będą umieszczane wszystkie pliki testowe i który pozwoli na swobodne testowanie prezentowanych przykładów. Ten katalog można nazwać np. *www*. Taki też katalog będzie się pojawiał w dalszych przykładach.

Po zapisaniu pliku trzeba go wczytać do przeglądarki. Najprościej można to zrobić, klikając dwukrotnie plik w Eksploratorze Windows. Wtedy zostanie uruchomiona domyślna przeglądarka[3], a plik zostanie do niej wczytany. Można też najpierw uruchomić przeglądarkę (dowolną), a następnie wczytać do niej plik. Wtedy:

- W Firefoksie z menu *Plik* wybieramy *Otwórz plik* i wskazujemy plik *index.html*.
- W Internet Explorerze z menu *Plik* wybieramy *Otwórz*, a następnie przeglądaj, i wskazujemy plik *index.html*.
- W Operze z menu *Plik* wybieramy *Otwórz* i wskazujemy plik *index.html*.

Niezależnie od tego, jakiej aplikacji użyliśmy, na ekranie pojawi się, wprowadzona wcześniej w edytorze, treść pliku tekstowego (rysunek 1.3).

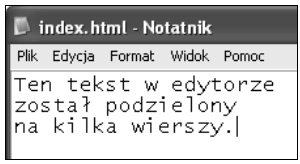
Rysunek 1.3. *Plik tekstowy wczytany do przeglądarki*



Jak widać, najprostsza strona WWW wcale nie musi się składać ze skomplikowanych i mało zrozumiałych dla laika poleceń. Wystarczy napisać trochę tekstu. Niestety takie postępowanie wystarczy jedynie do zaprezentowania prostej treści. Już przy kilku zdaniach natrafimy na prozaiczne problemy, choćby takie jak brak możliwości zastosowania akapitów, zmiany czcionki, nie mówiąc już o budowaniu układu strony.

Warto zrobić prosty test. Zmodyfikujmy treść pliku *index.html* (np. otwierając go ponownie w Notatniku) lub utwórzmy nowy, tak by zawierał treść w kilku wierszach (rysunek 1.4).

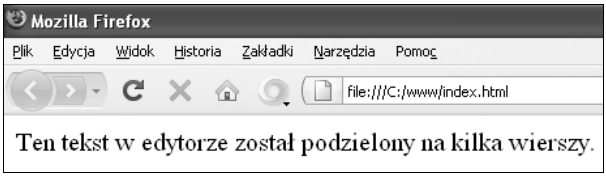
Rysunek 1.4. *Edytor z kilkoma wierszami tekstu*



[3] Przeglądarka domyślna to ta, która została zarejestrowana w systemie jako produkt mający być używany domyślnie przy wszelkich czynnościach związanych z WWW, np. do otwierania plików z rozszerzeniem *.html*. W systemie Windows standardowo domyślną przeglądarką jest Internet Explorer, jednak podczas instalacji innych produktów (Firefox, Opera) każdy z nich oferuje możliwość zmiany tego ustawienia. Tak więc domyślną przeglądarką może być praktycznie dowolny produkt tego typu.

Jeżeli po zapisaniu danych ponownie wczytamy plik *index.html* do przeglądarki, spotka nas niespodzianka. Cały tekst będzie prezentowany tylko w jednym wierszu (rysunek 1.5). Widać więc wyraźnie, że do prawidłowego formatowania strony WWW potrzebne są dodatkowe elementy. Służy do tego język HTML lub XHTML. Dopiero jego zastosowanie pozwala tworzyć poprawne witryny.

Rysunek 1.5. *W przeglądarce tekst prezentowany jest jednym ciągiem*



HTML i XHTML

Językiem służącym do tworzenia stron WWW jest HTML, czyli *Hypertext Markup Language*. Nazywamy go również językiem opisu strony. Składa się on ze znaczników, które nadają znaczenie różnym elementom tekstowym. Pozwalają np. sformatować akapit czy też dodać do dokumentu odnośnik (link, hiperłącze) prowadzący do innej witryny bądź zasobów sieci.

Początki HTML sięgają wczesnych lat 90. XX wieku. Pierwszy dokument dokonujący standaryzacji tego języka pojawił się w 1995 roku i dotyczył standardu HTML 2.0. Obecnie[4] najnowszą oficjalną wersją standardu jest HTML 4.01, który pochodzi z 1999 roku [5]. Nie da się więc ukryć, że ma on już swoje lata, a rozwój został na długi czas zatrzymany. Dopiero w styczniu 2008 roku został opublikowany roboczy szkic wersji 5.0, która wciąż jest w fazie uzgodnień organizacji standaryzacyjnych.

Jedną z przyczyn zatrzymania rozwoju HTML było dosyć powszechne pod koniec lat 90. ubiegłego wieku przekonanie, że kolejne wersje języka opisu strony powinny być oparte na zdobywającym wtedy coraz większą popularność języku XML. Dlatego też powstał język XHTML (*Extensible Hypertext Markup Language*), który w swej pierwotnej wersji 1.0 (styczeń 2000 r.) był stosunkowo prostą adaptacją HTML 4.01 do standardu XML. Powstała również wersja 1.1, która jednak aż do początku 2009 roku nie powinna być formalnie stosowana do zapisu danych typu HTML[6].

Aby jednak nie wnikać w dalsze teoretyczne rozważania na temat języków HTML i XHTML, czas odpowiedzieć na pytanie nurtujące zapewne każdego początkującego twórcę stron WWW — który z tych języków wybrać? Odpowiedź będzie przewrotna. Nie ma formalnej potrzeby dokonywania wyboru — przynajmniej na początku nauki.

---

[4] Słowa te zostały napisane w drugiej połowie 2009 roku.

[5] Oficjalne zatwierdzenie standardu miało miejsce w maju 2000 r. (standard ISO/IEC 15445:2000).

[6] Wynika to z rekomendacji organizacji standaryzacyjnej W3C. Otóż typem danych dla dokumentów HTML i XHTML1.0 może być standardowo rozpoznawany przez wszystkie serwery i przeglądarki (a także inne aplikacje) *text/html*, gdy tymczasem typem danych dla XHTML 1.1 powinno być *application/xhtml+xml*. To ograniczenie zostało zniesione w styczniu 2009 roku.

Wszystkie podstawowe struktury witryny można zapisać tak, aby były zgodne i z HTML, i z XHTML. Co więcej, zwolenników i przeciwników obu języków pogodzi zapewne HTML5 (zgodny z XHTML5)[7].

Wszystkie przykłady prezentowane w książce (chyba że zaznaczono inaczej) będą zgodne zarówno z HTML 4.01, jak i z XHTML 1.0 i 1.1, a w większości przypadków także z nadchodzącą najnowszą wersją HTML5 (różnice między HTML 4.01 i HTML5 będą zaznaczane w opisach).

Podstawowa struktura strony WWW

Struktura strony WWW budowana jest z tzw. znaczników. Można je traktować jak polecenia formatujące. Więcej informacji o znacznikach zostanie przedstawionych w lekcji 3., już teraz jednak powinniśmy zapoznać się z ogólną budową strony. Została ona przedstawiona na listingu 1.1.

Listing 1.1. *Ogólna struktura strony WWW*

```
<html>
<head>
Tutaj należy umieścić treść nagłówka dokumentu
</head>
<body>
Tutaj należy umieścić właściwą treść dokumentu HTML
</body>
</html>
```

Dokument rozpoczynamy od znacznika <html>, a kończymy znacznikiem </html>. Wewnątrz można wyróżnić dwie sekcje. Pierwsza z nich to tak zwany nagłówek, który znajduje się między znacznikami <head> i </head>. Umieszczane są w nim różne informacje o dokumencie, takie jak np. sposób kodowania znaków, tytuł, dane o autorze itp. Te informacje nie są wyświetlane. Właściwa teść dokumentu, czyli to co zobaczy użytkownik odwiedzający witrynę, znajduje się w sekcji wyróżnionej za pomocą znaczników <body> i </body>. Tam można umieścić np. akapity tekstowe.


Uwaga

Treść zaprezentowaną na listingu, a składającą się ze znaczników HTML (XHTML), *nazywamy kodem źródłowym strony WWW*. Używając skrótu, będziemy mówili po prostu o *kodzie strony*.

Taka struktura będzie wspólna zarówno dla kodu HTML, jak i XHTML[8]. Aby więc można było rozróżnić oba typy, a także określić konkretną wersję wybranego języka, niezbędne jest zastosowanie tzw. prologu. Prolog jest elementem niezbędnym dla każdej poprawnej, zgodnej ze standardami strony WWW. Rodzaje prologów zostały opisane w części lekcji zatytułowanej „Prolog dokumentu HTML i XHTML".

[7] Prace nad XHTML 1.2 i 2.0 są obecnie wstrzymywane, a tzw. XHTML5 jest częścią standardu HTML5.

[8] Jest to również struktura dla dokumentów HTML w wersjach poniżej 4, w których nie stosowano elementów prologu.

Pierwsza strona WWW

Czas napisać naszą pierwszą stronę WWW, czyli wypełnić zaprezentowaną wcześniej strukturę prawidłową treścią. Zadaniem tej strony będzie jedynie wyświetlenie jednego akapitu tekstowego składającego się z kilku słów. Kod takiej strony, zgodny ze standardem HTML 4.01, został przedstawiony na listingu 1.2. Wygląda on dosyć skomplikowanie, nie należy się tym jednak przejmować. Wszystkie elementy zostaną wyjaśnione krok po kroku.

Listing 1.2. *Najprostsza strona WWW*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Pierwsza strona WWW</title>
</head>
<body>
<p>To jest akapit tekstowy.</p>
</body>
</html>
```

Zanim jednak zajmiemy się poszczególnymi elementami kodu, zapiszmy go w pliku *index.html* i wczytajmy do przeglądarki, tak jak zostało to opisane w podrozdziale „Wczytanie dokumentu do przeglądarki". Zobaczymy wtedy widok przedstawiony na rysunku 1.6.

Rysunek 1.6. *Kod z listingu 1.2 wczytany do przeglądarki*

Jak widać, na ekranie pojawił się jedynie tytuł strony, widoczny zarówno na pasku tytułu okna przeglądarki, jak i w nagłówku karty (o ile użyta przeglądarka korzysta z kart), oraz treść akapitu tekstowego. Żadna inna treść, a w szczególności znaczniki HTML, nie została wyświetlona.

Przeanalizujmy teraz kod z listingu 1.2, który wygenerował widoczną na rysunku witrynę. Zaczyna się on od tak zwanego prologu (deklaracji typu dokumentu):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
```

Jest to po prostu określenie standardu HTML, w którym została utworzona strona. Więcej informacji o prologach znajduje się w części lekcji zatytułowanej „Prolog dokumentu HTML i XHTML". Na razie przyjmijmy po prostu, że w przypadku standardu HTML 4.01 na początku kodu należy umieścić taki, trzeba przyznać, dosyć dziwnie wyglądający, ciąg znaków.

W sekcji <head> został umieszczony znacznik <title>, który pozwala zdefiniować tytuł strony:

```
<title>Pierwsza strona WWW</title>
```

Tytułem jest cała treść znajdująca się pomiędzy <title> i </title>. W prezentowanym przypadku jest to ciąg Pierwsza strona WWW. Utworzenie tytułu strony jest obligatoryjne. Zawsze należy więc użyć znacznika <title> i musi on być umieszczony w sekcji <head>.

W sekcji <body>, zawierającej właściwą treść witryny, został umieszczony znacznik :

```
<p>To jest akapit tekstowy.</p>
```

Definiuje on jeden akapit tekstowy. Treścią akapitu są wszystkie znaki umieszczone między a . W tym przypadku jest to zatem ciąg To jest akapit tekstowy. Ten ciąg jest wyświetlany w przeglądarce po wczytaniu witryny.

Pierwsza strona w XHTML

Jeżeli zamiast HTML chcemy użyć XHTML, wcale nie musimy wprowadzać znaczących modyfikacji kodu strony. W przypadku prostych witryn zmiany będą dotyczyły tylko samego początku dokumentu, który będzie musiał zawierać inny prolog, a także bardziej rozbudowany wstęp. Najlepiej widać to na listingu 1.3.

Listing 1.3. *Kod prostej strony w XHTML*

```
<?xml version="1.1" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/tr/xhtml11/Dtd/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl">
<head>
<title>Pierwsza strona WWW</title>
</head>
<body>
<p>To jest akapit tekstowy.</p>
</body>
</html>
```

Tym razem jeszcze przed deklaracją typu dokumentu znajduje się wiersz:

```
<?xml version="1.1" encoding="utf-8"?>
```

Określa on wersję dokumentu XML (parametr version), w tym przypadku 1.1, a także sposób kodowania znaków (parametr encoding). Więcej informacji o kodowaniu znaków znajduje się w części zatytułowanej „Polskie litery na stronie WWW”.

Poniżej znajduje się właściwa deklaracja typu dokumentu:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/tr/xhtml11/Dtd/xhtml11.dtd">
```

właściwa dla standardu XHTML 1.1 (deklaracje właściwe dla innych wersji XHTML zostały opisane w kolejnej części lekcji).

Znacznik `<html>` rozpoczynający właściwy dokument HTML został tu uzupełniony o parametr określający definicję tzw. przestrzeni nazw XHTML (`xmlns`), a także użyty język (`pl`):

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl">
```

Te parametry nie są jednak obligatoryjne i znacznik ten mógłby mieć również prostą postać, czyli `<html>`.

Dalsza część kodu jest taka sama jak w przypadku przykładu z listingu 1.2. Tak samo będzie też w większości przykładów prezentowanych w książce. Właściwa treść kodu strony będzie zgodna zarówno z HTML, jak i XHTML, a więc uzyskanie odpowiedniego typu dokumentu będzie można uzyskać, wymieniając jedynie opisane fragmenty prologów. Na listingach będzie natomiast prezentowany wyłącznie prolog dla HTML 4.01.

Prolog dokumentu HTML i XHTML

Prolog dokumentu to informacja o standardzie, w jakim ten dokument został wykonany. Innymi słowy określa on typ dokumentu, jest to deklaracja typu dokumentu — DTD (z ang. *document type declaration*). Dzięki tej deklaracji przeglądarka (lub inny program) będzie wiedziała, jak odczytywać dokument i czego (jakich struktur) może się w nim spodziewać. Dla języka HTML w wersji 4 (4.01) stosuje się 3 rodzaje prologu. Pierwszy ma postać:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
```

i określa ścisłą wersję standardu 4.01. To oznacza, że na stronie nie mogą być umieszczone żadne elementy uznane za przestarzałe (z ang. *deprecated*, stosowane jest też określanie *elementy schyłkowe*), a dokument musi dokładnie spełniać wszelkie wymogi standardu.

Drugi typ to:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

Jest to „przejściowa” (z ang. *transitional*) wersja standardu 4.01, czasami określana jako luźna, ze względu na swobodniejsze, mniej restrykcyjne podejście do respektowania standardu. Można w niej używać znaczników uznanych za przestarzałe.

Trzeci typ to:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
```

Ta wersja oznacza stronę z ramkami (temat ramek zostanie przedstawiony w lekcji 13.). Dokładniej rzecz ujmując, taki dokument jest traktowany jak wersja Transitional uzupełniona o możliwość stosowania ramek.

Każdy z wymienionych prologów może być uzupełniony o odnośnik do dokumentu zawierającego formalny opis składni dla danego standardu. Nie jest to obligatoryjne, ale jest wskazówką dla narzędzi dokonujących walidacji (sprawdzenia poprawności) dokumentów. Najczęściej prolog jest więc o taki odnośnik uzupełniany, mimo że przeglądarki z reguły z tej informacji nie korzystają.

Alternatywne wersje prologów będą zatem miały przedstawione poniżej postaci.

Wersja HTML 4.01 Strict:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
```

Wersja HTML 4.01 Transitional:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
```

Wersja HTML 4.01 Frameset:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

Analogicznie jak w przypadku HTML, dokumenty XHTML również mają swoje prologi. Dla standardu XHTML 1.0 będą one następujące:

Dla ścisłej (strict) wersji standardu:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN">
```

Dla przejściowej (transitional) wersji standardu:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
```

Dla wersji z ramkami (frameset):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN">
```

Określenia „ścisła", „przejściowa" i „z ramkami" mają tu takie samo znaczenie jak w przypadku standardu HTML 4.01.

Jeśli deklaracja DTD ma również zawierać odnośnik do dokumentu z opisem składni, stosuje się następujące prologi:

Dla ścisłej (strict) wersji standardu:

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

Dla przejściowej (transitional) wersji standardu:

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

Dla wersji z ramkami (frameset):

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

Dla XHTML w wersji 1.1 stosuje się prolog o postaci:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Wszystkie prezentowane w książce przykłady (chyba że zaznaczono inaczej) będą zgodne ze ścisłą wersją standardu HTML 4.01 (a także XHTML 1.1) i będzie w nich stosowany prolog o postaci:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
```

Jeśli natomiast dana strona ma być zapisana jako XHTML, powinien być używany prolog:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Sposób umieszczenia prologu w dokumencie HTML i XHTML został zaprezentowany w poprzednich dwóch częściach lekcji.

Formatowanie kodu HTML

Ważną sprawą, na którą osoby początkujące z reguły nie zwracają uwagi, jest sposób formatowania kodu źródłowego strony. Nie ma to większego znaczenia przy niewielkich witrynach, na których znajduje się jedynie kilka akapitów tekstowych, jednak przy bardziej rozbudowanych stronach właściwy zapis bardzo ułatwi późniejszą edycję i poprawki. Chodzi o to, aby tak zapisać kod źródłowy, żeby był jak najbardziej dla nas czytelny. Dlatego też najczęściej wydziela się poszczególne bloki, stosując wcięcia od lewej strony. Kod z listingu 1.2 można by przedstawić tak, jak zaprezentowano to na listingu 1.4.

Listing 1.4. *Kod HTML z wyróżnieniem bloków funkcjonalnych*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Pierwsza strona WWW</title>
  </head>
  <body>
    <p>To jest akapit tekstowy.</p>
  </body>
</html>
```

W ten sposób wyraźnie zostały wydzielone sekcje `<head>` oraz `<body>` (odsunięte o dwa odstępy od lewej strony listingu), a także ich zawartość, czyli znaczniki `<title>` i `<p>` (przesunięcie o dwa odstępy od początku każdej sekcji, czyli o 4 odstępy od lewej strony listingu).

Takie wyróżnienie bloków funkcjonalnych bardzo ułatwia orientowanie się w strukturze kodu, co jest ważne przy tworzeniu nawet niezbyt złożonych witryn. Oczywiście nie ma to żadnego wpływu na sposób wyświetlania danych w przeglądarce (o tym decydują odpowiednie znaczniki), a wpływa wyłącznie na czytelność kodu źródłowego. Rzecz jasna stosowanie wcięć formatujących nie jest obligatoryjne i nie ma konieczności ich używania, jest to jednak dobry zwyczaj, który po prostu warto stosować. Nie ma też jednego ustalonego standardu formatowania, który by mówił, gdzie i ile

dokładnie zastosować odstępów, czy przejść do nowego wiersza. Każdy może tu wypracować swój własny, najbardziej czytelny dla siebie, standard. Zawsze trzeba jednak wziąć pod uwagę, że najważniejszym kryterium jest czytelność kodu oraz to, że w przyszłości może on być analizowany i poprawiany przez innych programistów.

Polskie litery na stronie WWW

Podczas tworzenia strony WWW prędzej czy później napotkamy problem prawidłowego wyświetlania polskich znaków, a mówiąc ogólniej — wszelkich znaków narodowych i niestandardowych wykraczających poza alfabet łaciński (i standard ASCII[9]). Najlepiej więc od razu dowiedzieć się, jak prawidłowo kodować witrynę, tak by każdy użytkownik zobaczył prawidłowy zapis. Wykonajmy jednak najpierw prosty test. W kodzie z listingu 1.2 (lub 1.4) zmieńmy treść akapitu tekstowego, tak aby zawierał polskie znaki. Cała sekcja `<body>` może przyjąć postać widoczną na listingu 1.5.

Listing 1.5. *Treść sekcji <body> z akapitem zawierającym polskie znaki*

```
<body>
  <p>Polskie znaki to m.in.: ąćęłńóśźż</p>
</body>
```

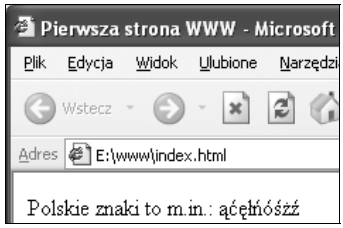
Tę treść zapiszmy w standardowy sposób za pomocą dostępnego w Windows Notatnika w pliku *index.html* i wczytajmy go do przeglądarki. Rezultaty będą ciekawe. Przykładowo przeglądarka Firefox wyświetli ciąg składający się zarówno z prawidłowych polskich liter, liter nieprawidłowych, jak i nieokreślonych znaków (rysunek 1.7).

Rysunek 1.7. *Test polskich znaków — przeglądarka Firefox 3*

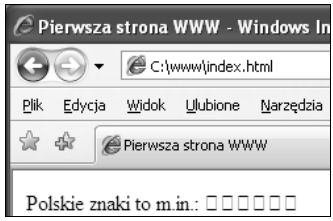
Lepiej sprawdzi się Internet Explorer w wersji 6. Tu polskie litery zostaną wyświetlone poprawnie (rysunek 1.8). Łatwo się domyślić, że jest to efekt zapisania kodu źródłowego w windowsowym Notatniku — dzięki temu edytor tekstu i przeglądarka pracują w tym samym (ale niestety nieprawidłowym, o czym niżej) standardzie kodowania znaków.

Myliłby się jednak ktoś, kto na tej podstawie założyłby, że Internet Explorer zawsze poprawnie wyświetli taką stronę. Otóż już w wersji 7 tej przeglądarki efekt będzie zgoła odmienny. Nie zobaczymy żadnej polskiej litery, a zamiast nich pojawią się znaki zastępcze (rysunek 1.9).

[9] Skrót ASCII pochodzi on angielskiej nazwy *American Standard Code for Information Interchange* i określa jeden z pierwotnych sposobów zapisu znaków dla komputerów osobistych. Niestety uwzględnia jedynie litery i znaki specjalne charakterystyczne dla krajów anglosaskich.



Rysunek 1.8. *Test polskich znaków — przeglądarka Internet Explorer 6*



Rysunek 1.9. *Test polskich znaków — przeglądarka Internet Explorer 7*

Nie powinniśmy jednak winić za ten stan rzeczy przeglądarek. Różne zaszłości historyczne spowodowały bowiem, że polskie litery (a ogólniej — wszelkie znaki narodowe i niestandardowe) mogą być zapisywane w różnych formatach (różnych standardach kodowania[10]). Skoro tak, to naszym zadaniem jest poinformowanie przeglądarki, jakiego standardu kodowania użyliśmy na naszej stronie WWW. Nie jest jej zadaniem „zgadywanie" tej informacji. To oznacza, że w kodzie strony zawsze musimy umieścić znacznik określający sposób kodowania. Ma on następującą postać:

```
<meta http-equiv="Content-Type" content="text/html; charset=kodowanie">
```

i należy go umieścić w sekcji <head>. Ten znacznik mówi przeglądarce, że ma do czynienia z tekstowym dokumentem HTML, w którym znaki zostały zakodowane w standardzie określonym przez wartość parametru *charset*. W miejsce ciągu *kodowanie* należy zatem wstawić określenie standardu kodowania znaków.

W sieci spotkamy strony stosujące następujące kodowania:

- *Windows-1250* — kodowanie znaków charakterystyczne dla systemu Windows. Nie stanowi normy międzynarodowej. Należy unikać stosowania tego zapisu.
- *ISO-8859-2* — popularny w latach 90. XX w. sposób zapisu stanowiący normę międzynarodową[11]. Można go używać, jednak w obecnych czasach lepiej stosować kodowanie UTF-8.
- *UTF-8* — najnowszy z prezentowanych sposób zapisu, według międzynarodowych standardów Unicode i UCS[12]. Unicode pozwala na zapis znaków występujących w większości języków świata. O ile to możliwe, należy korzystać z tego standardu zapisu.

[10] Znaki, które wprowadzamy z klawiatury, są w komputerze zapisywane za pomocą wartości liczbowych. Każdy znak ma przypisany swój kod, np. mała litera *a* ma kod 97 (w standardzie ASCII i pochodnych). Sposób przypisania wartości liczbowych (kodów) do liter nazywamy właśnie standardem kodowania znaków.

[11] Odpowiada to polskiej normie PN-93/T-42118.

[12] Standardy UCS (standard ISO) i Unicode są w większości ze sobą zgodne w zasadzie we wszystkich wersjach, i w praktyce nie rozróżnia się ich, mówiąc po prostu o zapisie Unicode. UTF-8 to po prostu nazwa jednego ze sposobów zapisu znaków według normy Unicode.

Znacznik <meta> będzie więc przyjmował następujące postaci:

Dla Windows-1250:

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
```

Dla ISO-8859-2:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

Dla UTF-8:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Jak zatem spowodować, aby strona z listingu 1.5 była poprawnie wyświetlana we wszystkich współczesnych przeglądarkach? Skoro zapisaliśmy ją w Notatniku w standardowy sposób, to sposobem kodowania jest Windows-1250. Takie też kodowania należy uwzględnić w znaczniku <meta> umieszczonym w sekcji <head>. Kod przyjąłby wtedy postać widoczną na listingu 1.6.

Listing 1.6. *Strona zawierająca określenie standardu kodowania znaków*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
    <title>Pierwsza strona WWW</title>
  </head>
  <body>
    <p>Polskie znaki to m.in.: ąćęłńóśżź</p>
  </body>
</html>
```

Po takim uzupełnieniu kodu polskie znaki pojawią się we właściwej postaci i w Firefoksie, i w Internet Explorerze (a także w innych przeglądarkach jak Opera czy Konqueror).

Jak zostało to jednak wspomniane wyżej, kodowanie typu Windows-1250 jest zdecydowanie niezalecane. Zamiast niego powinniśmy stosować UTF-8. Jak to zrobić? Trzeba skorzystać z odpowiedniego edytora tekstowego. To zagadnienie zostało opisane w kolejnej części lekcji.

Edytory tekstowe dla Windows

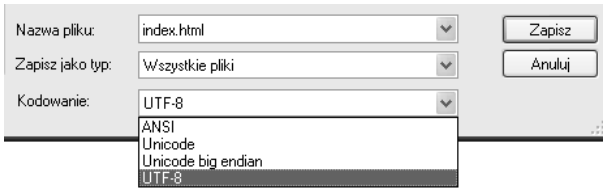
Do tworzenia stron WWW potrzebny jest edytor. Do nauki najlepsze są edytory pracujące w trybie tekstowym, w którym znaczniki i treść, z których składa się witryna, wpisuje się bezpośrednio z klawiatury. Istnieją co prawda programy oferujące graficzny tryb budowania witryny, jednak osoby początkujące raczej nie powinny z nich korzystać, gdyż najpierw należy poznać budowę witryn „od wewnątrz", zaznajamiając się z kolejnymi znacznikami i zależnościami między nimi. Tylko w ten sposób można zostać profesjonalnym twórcą WWW.

Edytory tekstowe można podzielić na dwa rodzaje. Te ogólnego przeznaczenia oraz te wspomagające wprowadzanie kodu HTML. Dla naszych potrzeb wystarczy w zupełności ten pierwszy rodzaj. Ważne, aby taki edytor obsługiwał kodowanie UTF-8. Takie warunki, choć z pewnymi problemami, spełnia nawet windowsowy Notatnik. Lepszym rozwiązaniem jest jednak użycie takich produktów jak Notepad++ czy Notepad2, które dodatkowo oferują bardzo przydatne podświetlanie (kolorowanie) składni HTML (XHTML).

Notatnik Windows

Notatnik jest edytorem tekstu dostępnym nawet w tak leciwych wersjach systemu jak 3.1. Począwszy od Windows 2000, pozwala on na odczyt i zapis plików ze znakami kodowanymi w standardzie UTF-8, nadaje się więc również do tworzenia poprawnych stron WWW. Nie oferuje jednak żadnych dodatkowych udogodnień, brakuje w nim nawet tak prostych funkcji jak numerowanie wierszy. Nie nadaje się więc do tworzenia bardziej złożonych witryn. Na potrzeby tego kursu będzie jednak wystarczający, jeśli więc ktoś nie chce instalować w swoim systemie dodatkowego oprogramowania, może skorzystać z Notatnika.

Aby utworzoną w Notatniku stronę WWW zapisać w kodowaniu UTF-8, z menu *Plik* należy wybrać pozycję *Zapisz jako*. W oknie dialogowym służącym do zapisu plików (rysunek 1.10) w polu *Nazwa pliku* należy wpisać nazwę pliku (np. *index.html*), z listy *Zapisz typ jako* wybrać pozycję *Wszystkie pliki*, a z listy *Kodowanie* koniecznie wybrać pozycję UTF-8.



Rysunek 1.10. *Zapis pliku jako UTF-8 z użyciem systemowego Notatnika*

Tak zapisany plik będzie już zawsze rozpoznawany przez Notatnik jako zapisany w kodowaniu UTF-8. To oznacza, że przy kolejnym otwieraniu go w Notatniku nie trzeba już przeprowadzać żadnych dodatkowych operacji, a zapis można wykonać za pomocą zwykłego polecenia *Zapisz* (menu *Plik*, pozycja *Zapisz* lub kombinacja klawiszy *CTRL+S*).

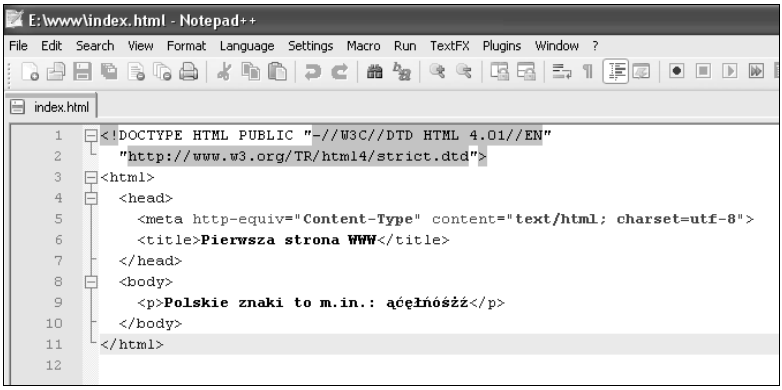
Niestety Notatnik w przypadku plików kodowanych jako UTF-8 zawsze zapisuje na początku pliku tzw. znacznik BOM[13]. To w niektórych przypadkach może spowodować pewne problemy z wyświetlaniem strony. Może się wtedy na jej początku pojawić ciąg trzech znaków widoczny na rysunku 1.11. To rzadka sytuacja, przeglądarki z reguły radzą sobie z takimi plikami, czasem jednak w sieci zobaczymy stronę serwującą nam owe kilka dodatkowych znaków. Najlepszym rozwiązaniem jest więc użycie edytora, który potrafi zapisać znaki bez znacznika BOM.

[13] Znacznik BOM, czyli znacznik porządku bajtów (z ang. *byte order mark*). Są to trzy bajty określające porządek bajtów składających się na znaki Unicode. Dla kodowania UTF-8 znacznik BOM nie jest konieczny, gdyż ustawienie bajtów składających się na znak jest z góry określone.

Rysunek 1.11. *Uwidoczniony w przeglądarce znacznik BOM*

Notepad++

Doskonałym zamiennikiem systemowego notatnika jest Notepad++ (rysunek 1.12). Jest to darmowa aplikacja rozwijana na zasadach wolnego oprogramowania. Najnowszą wersję można pobrać pod adresem *http://notepad-plus.sourceforge.net/*. Edytor ten oferuje wiele ciekawych i przydatnych funkcji. Najważniejsze dla nas to: kolorowanie składni (X)HTML (a także kilkudziesięciu innych języków), co ogromnie zwiększa czytelność kodu, numerowanie poszczególnych wierszy i zapis danych w standardzie UTF-8. Pod podanym adresem oprócz wersji instalacyjnej można również znaleźć pliki z polską wersją językową.



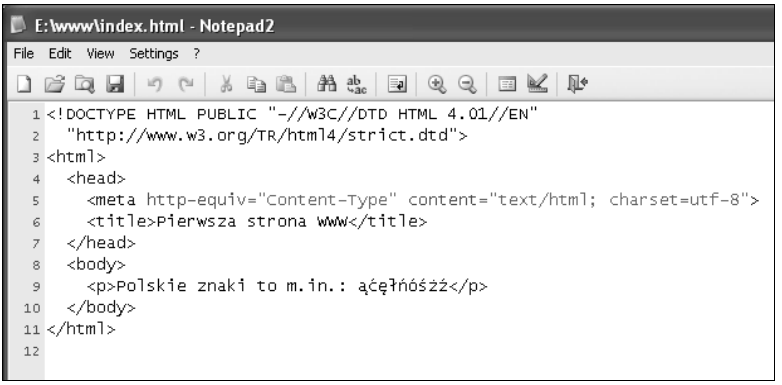
Rysunek 1.12. *Kod strony WWW w edytorze Notepad++*

Aby zastosować kodowanie UTF-8, należy, najlepiej jeszcze przed rozpoczęciem wpisywania tekstu, z menu *Format* wybrać pozycję *Encode In UTF-8 without BOM* (*Koduj w UTF-8 (bez BOM)*). Po wprowadzeniu kodu plik zapisujemy standardowo, wybierając z menu *File* (*Plik*) pozycję *Save* (*Zapisz*) lub wciskając kombinację klawiszy *Ctrl+S*.

W przypadku gdy dysponujemy plikiem, który został zapisany w standardzie Windows-1250 (np. był tworzony w Notatniku i zapisany standardowo, bez zmiany kodowania), przejście na UTF-8 osiągniemy, wybierając z menu *Format* pozycję *Convert to UTF-8 wihout BOM* (*Konwertuj na format UTF-8 bez BOM*). Sposób zapisu pliku na dysku nie zmieni się.

Notepad2

Kolejnym zamiennikiem Notatnika jest Notepad2 (rysunek 1.13). Jego możliwości są mniejsze niż oferowane przez Notepad++, jednak na potrzeby naszego kursu najzupełniej wystarczające. Edytor oferuje kolorowanie składni HTML, numerowanie wierszy kodu i zapis w formacie UTF-8. Aby edytowany dokument został zapisany w kodowaniu UTF-8, z menu *File* należy wybrać pozycję *Encoding*, a następnie zaznaczyć pozycję UTF-8.



Rysunek 1.13. *Strona WWW wczytana do edytora Notepad2*

Inne edytory

Oprócz edytorów tekstowych ogólnego przeznaczenia, których trzy przykłady zostały przedstawione wyżej, w sieci można znaleźć też specjalizowane aplikacje wspomagające tworzenie witryn internetowych. Wśród nich znajdują się zarówno aplikacje komercyjne, jak Adobe Dreamweaver czy polski Pajączek, jak i darmowe aplikacje, z których można wymienić 1st page 2000, ezHTML czy WebSite Pro. Część z nich umożliwia budowanie stron w trybie graficznym, większość jednak pracuje w trybie tekstowych, wspomagając twórcę przez automatyczne wstawianie znaczników, autouzupełnianie kodu czy weryfikację poprawności witryny.

Ta książka to jednak kurs tworzenia stron WWW w HTML i XHTML, a nie instrukcja obsługi konkretnego edytora. Zanim bowiem zacznie się używać zaawansowanych narzędzi, najpierw trzeba poznać zasady konstruowania stron. Dopiero na dalszym etapie nauki można wybrać bardziej zaawansowane narzędzia i świadomie wyselekcjonować takie, które jest najbardziej odpowiednie do potrzeb. Dlatego też w zupełności wystarczającym edytorem będzie Notepad++ bądź Notepad2, a w ostateczności zwykły Notatnik.

Różne przeglądarki

Choć większość użytkowników internetu przyzwyczajona jest tylko do jednej przeglądarki, a wiele osób nawet nie wie, że istnieje ich wiele, twórca stron WWW musi zdawać sobie sprawę, iż każda aplikacja tego typu jest trochę inna. W przypadku prostych stron różnice zapewne się nie ujawnią, jednak wraz z tworzeniem coraz bardziej

zaawansowanych witryn coraz większe znaczenie będzie miało testowanie kodu z uwzględnieniem różnych przeglądarek. Niestety, mimo że będziemy tworzyć witryny w pełni zgodne ze standardami HTML i XHTML, często okaże się, iż występują różnice w prezentacji strony. Może się więc okazać, że do formalnie poprawnego kodu trzeba będzie wprowadzać poprawki.

Oczywiście nie ma sensu testować witryn we wszystkich dostępnych przeglądarkach — jest ich zbyt wiele. Obecnie na rynku liczą się głównie Internet Explorer i Firefox — każda witryna musi więc działać poprawnie w obu tych produktach. Warto również uwzględnić przeglądarkę Opera, która co prawda nie jest tak popularna, ale ma też grono zagorzałych zwolenników. Poza naszym krajem kilkuprocentową popularnością cieszy się również dostępna głównie dla komputerów Apple przeglądarka Safari.

Oprócz testowania różnych typów przeglądarek nie należy zapominać o tym, że mają one różne wersje, które też mogą różnić się między sobą. Te różnice nie są już tak duże, bowiem w obrębie jednej rodziny przeglądarki cechują się — co na nie budzi na pewno żadnego zdziwienia — dużą zgodnością. W zakresie prezentowanym w tym kursie nie będzie takiej potrzeby, ale już w dalszej, profesjonalnej praktyce tworzenia stron należałoby również uwzględniać poszczególne wersje produktów. Obecnie należałoby np. testować strony w Internet Explorerze 6, 7 i 8 (na wersję 6 zwracając najmniejszą uwagę), oraz Firefoksie 3 i 3.5, czyli w tych, które w danym czasie są najbardziej rozpowszechnione.

Lekcja 2. Strona WWW w internecie

- Z czego składa się adres strony WWW?
- Jak umieścić witrynę w internecie?
- Do czego potrzebne jest konto WWW?
- Jak wgrać pliki w witryną na serwer internetowy?
- Jak używać aplikacji FTP?
- Co zrobić, aby mieć własny adres internetowy?
- Czy można utworzyć konto WWW na własnym komputerze?

Adres strony WWW

Gdy chcemy odwiedzić wybraną stronę WWW, wpisujemy w przeglądarce jej adres, np. *http://helion.pl*. Co to oznacza? Jest to polecenie dla przeglądarki, aby połączyła się z adresem *helion.pl* i pobrała domyślną (główną) stronę WWW. Pod wpisanym adresem znajduje się serwer WWW (czyli komputer z odpowiednim oprogramowaniem), który jest w stanie taką stronę wysłać do przeglądarki. Gdy przeglądarka otrzyma dane, wyświetla je w swoim oknie, tak że możemy zobaczyć je na ekranie.

Czym jednak jest strona domyślna czy też główna strona WWW na danym serwerze? Otóż jest to plik z kodem (X)HTML, który został uznany za główną część witryny przez administratora serwera. Z reguły jest to plik o nazwie *index.html*, *index.htm*, *main.html* lub podobny[14]. Często też spotkamy adresy jawnie odwołujące się do takiego pliku, np.: *http://helion.pl/index.htm*. Takie odwołanie oznacza już: połącz się z serwerem znajdującym się pod adresem *helion.pl* i pobierz plik (dane z pliku) o nazwie *index.htm* znajdujący się w głównym katalogu tego serwera.

Dokładnie tak jest. Serwer WWW ma swoją strukturę katalogów, analogiczną do struktury dysku twardego na naszym własnym komputerze[15]. To znaczy, że znak / znajdujący się tuż za nazwą `helion.pl` określa główny katalog serwera WWW, a każdy kolejny taki znak — kolejne podkatalogi. Tym samym odwołanie:

```
http://helion.pl/ksiazki/cwjas2.htm
```

oznacza odwołanie do pliku o nazwie `cwjas2.html` znajdującego się w katalogu `ksiazki`, który jest podkatalogiem głównego katalogu serwera WWW uruchomionego pod adresem `helion.pl`.

Umieszczanie witryny w internecie

Aby nasza strona WWW była dostępna w sieci, musimy ją umieścić na serwerze. Na szczęście nie trzeba uruchamiać go samodzielnie. Istnieje bardzo wiele serwisów umożliwiających tworzenie własnych witryn. Prowadzone są przez firmy, które nazywamy dostawcami usług hostingowych. Każdy, kto założy konto w takim serwisie (nazywamy je kontem WWW), będzie mógł opublikować swoją własną witrynę. Oczywiście do nauki tworzenia stron nie jest to konieczne. Z powodzeniem można korzystać ze sposobu przedstawionego w pierwszej lekcji, czyli wczytywania plików HTML zapisanych na dysku domowego komputera.

Konto WWW

Konta WWW są z reguły płatne. Koszt waha się od kilkudziesięciu do kilkuset złotych rocznie. Praktycznie wszyscy dostawcy usług hostingowych oferują jednak bezpłatny okres próbny. Jest to zazwyczaj od 10 do 30 dni. W tym czasie bez wnoszenia opłat można przetestować konto i sprawdzić, czy jest ono odpowiednie dla naszych potrzeb. Aktualną listę największych polskich firm obsługujących konta WWW można znaleźć m.in. pod adresem *http://www.top100.pl*.

Istnieją także serwisy pozwalające na bezpłatne założenie konta WWW i umieszczenie własnej strony w sieci. Takie konto z reguły ma pewne ograniczenia (niewielka pojemność, brak niektórych usług, czasami również pojawiają się na nim reklamy),

[14] Obecnie bardzo często jest to plik typu PHP, ASP lub podobny, zawierający skrypt wykonywany po stronie serwera. To zagadnienie zdecydowanie wykracza jednak poza ramy tematyczne tej książki. Osobom zainteresowanym tworzeniem stron WWW za pomocą PHP można polecić publikację *PHP5. Praktyczny kurs* (*http://helion.pl/ksiazki/php5pk.htm*).

[15] Ściślej rzecz ujmując, odwzorowanie adresu WWW wcale nie musi odpowiadać faktycznej strukturze plików na serwerze. To jednak zagadnienie czysto techniczne, które nie ma wpływu na naukę tworzenia stron.

jednak do nauki tworzenia WWW będzie też całkowicie wystarczające. Konta tego typu oferują m.in. niektóre portale, jak np. Interia (*http://miasto.interia.pl*), Onet (*http://republika.onet.pl*), Wirtualna Polska (*http://webpark.pl*) i inne.

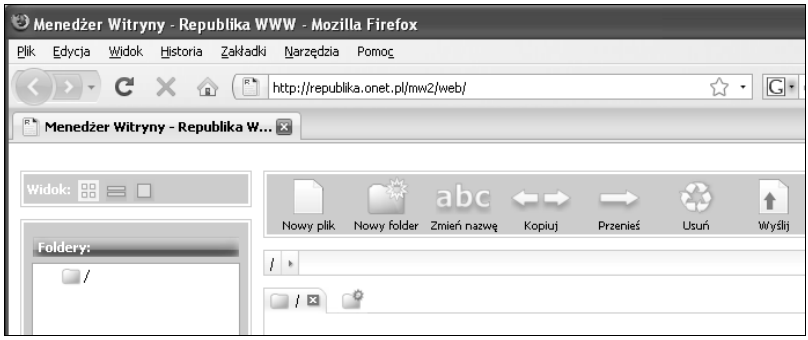
Niezależnie jednak od tego, czy wybierzemy usługę płatną czy też darmową, konieczne będzie założenie konta w danym serwisie. Niestety nie ma jednej procedury zakładania konta WWW, gdyż będzie się ona różnić w zależności od wybranego dostawcy usług, choć można wyróżnić pewne wspólne cechy, takie jak wybór nazwy i podanie danych kontaktowych. Z pewnością jednak nikt nie będzie miał z tym problemu, gdyż nie jest to dużo bardziej złożona procedura niż zakładanie konta pocztowego czy też konta na portalu społecznościowym.

W trakcie zakładania konta będziemy musieli podać jego nazwę oraz wymyślić hasło. Nazwa konta będzie je identyfikowała w danym serwisie i będzie częścią adresu internetowego naszej strony. Przykładowo jeśli nazwą konta będzie `mojastrona`, to adres, pod którym będą widzieć witrynę internauci, może mieć jedną z następujących postaci (zależnie od wybranego dostawcy usług hostingowych):

```
http://mojastrona.republika.pl
http://mojastrona.webpark.pl
http://mojastrona.miasto.interia.pl
```

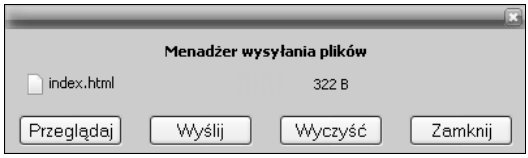
Umieszczanie strony na serwerze

Gdy mamy już konto u dostawcy usług hostingowych (na serwerze), musimy umieścić na nim plik (pliki) z naszą stroną WWW (popularnie mówi się o wgraniu plików na serwer). Sposób wykonania tego zadania jest zależny od rodzaju konta. Niektórzy dostawcy, często jest tak w przypadku kont darmowych dostępnych na popularnych portalach, udostępniają menedżery kont, które umożliwiają wykonanie wspomnianej operacji z poziomu przeglądarki. Przykładowy wygląd takiego menedżera obrazuje rysunek 1.14. Często jednak trzeba będzie skorzystać z aplikacji typu FTP (zostało to opisane w kolejnej części lekcji).



Rysunek 1.14. *Menedżer konta umożliwiający zarządzanie plikami składającymi się na witrynę*

Menedżer konta powinien zawierać część odpowiedzialną za wysyłanie plików. W omawianym przypadku będzie ona dostępna po kliknięciu na ikonę *Wyślij*. Pojawi się wtedy nowy element witryny pozwalający na wskazanie pliku znajdującego się na naszym domowym komputerze i wysłanie danych na serwer (rysunek 1.15).



Rysunek 1.15. *Menedżer plików pozwalający na wysłanie danych na serwer*

Po zakończeniu transferu plik pojawi się w głównym oknie menedżera (rysunek 1.16). To znak, że znajduje się na serwerze. Od tej chwili nasza strona WWW stanie się widoczna w sieci. Będzie się do niej można odwoływać, wpisując adres w pasku przeglądarki. Postać tego adresu jest zależna od rodzaju konta WWW, z którego korzystamy — przykłady zostały podane w poprzedniej części lekcji.

Jeżeli przykładowy plik ma nazwę inną niż *index.html*, adres witryny należy uzupełnić o nazwę pliku (zgodnie z informacjami podanymi na początku lekcji). Przykładowo, jeśli plik miałby nazwę *abc.html*, to odwołanie do witryny mogłoby mieć postać:

http://mojastrona.republika.pl/abc.html

http://mojastrona.webpark.pl/abc.html

http://mojastrona.miasto.interia.pl/abc.html

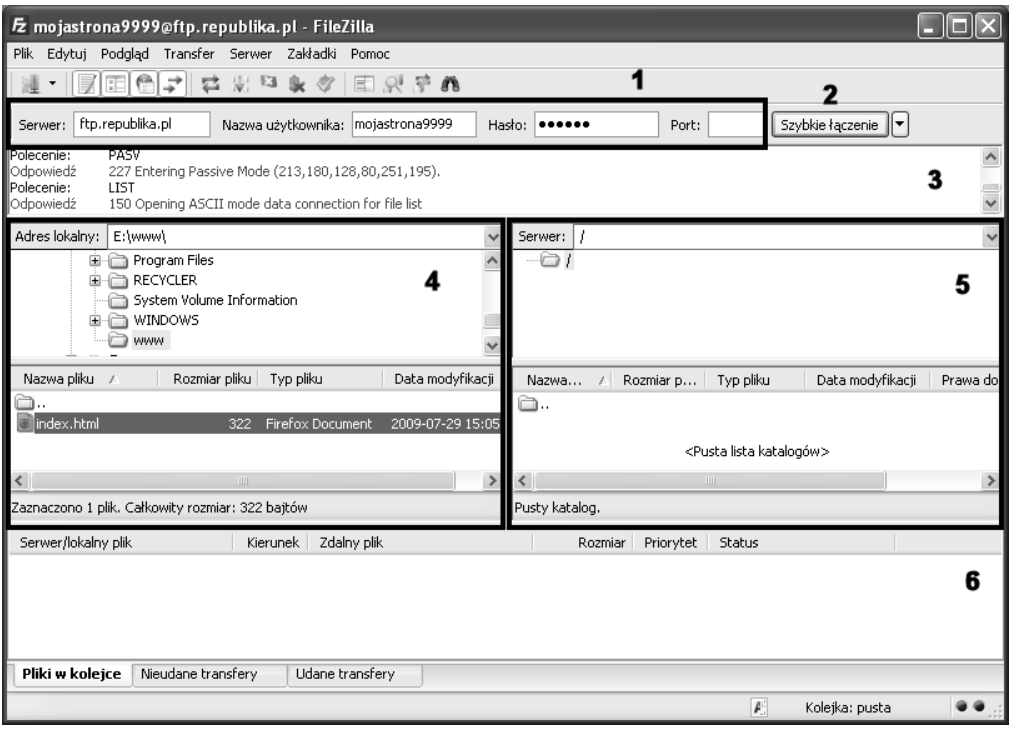
Rysunek 1.16. *Plik wgrany na serwer pojawił się w oknie menedżera konta*

Korzystanie z FTP

Nie wszyscy dostawcy usług hostingowych udostępniają menedżery kont, które pozwalają na wgranie plików składających się na stronę WWW na serwer. Wtedy, aby umieścić dane na serwerze, niezbędne jest skorzystanie z aplikacji FTP. Ten sposób jest też polecany, gdy witryna składa się z dużej liczby plików. Użycie FTP jest wtedy znacznie wygodniejsze.

Czym jest FTP? To protokół przesyłu plików (z ang. *File Transfer Protocol*), czyli metoda przesyłania plików między komputerami. Wiele zbiorów danych i oprogramowania pracuje z wykorzystaniem tej metody. Często nawet nie zdajemy sobie sprawy, że plik pobierany z internetu jest transmitowany właśnie za pomocą FTP. Aby skorzystać z tej metody transmisji, potrzebna jest aplikacja nazywana klientem FTP, na przykład FileZilla, CoreFTP, GoFTP.

Jak użyć FTP, zobaczymy na przykładzie darmowej aplikacji FileZilla, którą można znaleźć pod adresem *http://filezilla-project.org/*. Po pobraniu aplikacji należy ją zainstalować w systemie. Tę czynność wykonujemy w sposób standardowy, tak samo jak w przypadku każdego innego programu dla systemu Windows. Po uruchomieniu aplikacji zobaczymy jej główne okno, takie jak przedstawione na rysunku 1.17.



Rysunek 1.17. *Główne okno programu FTPZilla*

Można w nim wyróżnić kilka obszarów:

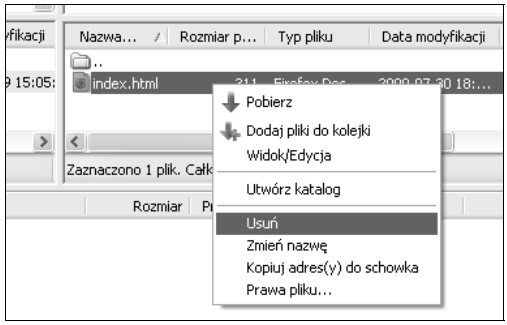
1. W górnej części znajdują się pola pozwalające na wpisanie nazwy serwera FTP, nazwy użytkownika oraz hasła.
2. Przycisk *Szybkie łączenie* pozwala na nawiązanie połączenia z serwerem.
3. Pole poniżej panelu danych przeznaczone jest dla komunikatów technicznych związanych z transmisją FTP. Nie trzeba na nie zwracać uwagi, chyba że wystąpi jakiś błąd. Wtedy można tam znaleźć przyczynę powstania błędu.
4. Okna po lewej stronie pokazują dane znajdujące się na naszym domowym komputerze. Można się w nim przemieszczać po strukturze katalogów i plików.
5. Okna po prawej stronie pokazują dane (strukturę katalogów i plików) znajdujące się na koncie WWW (na serwerze WWW).
6. Dolne okno zawiera dane o transmisji danych. Przydatne jest w przypadku przesyłania wielu plików. Pozwala m.in. ocenić, na jakim etapie znajduje się transmisja.

Aby połączyć się z serwerem, w górnym panelu trzeba wprowadzić dane właściwe dla danego dostawcy usług hostingowych[16], np. w przypadku konta na serwerze *republika.pl* nazwą serwera jest *ftp.republika.pl*, nazwą użytkownika — nazwa konta (np. `mojastrona`), a hasłem — hasło podane przy tworzeniu konta. Pole *Port* najlepiej pozostawić puste. Po wpisaniu niezbędnych informacji należy kliknąć przycisk *Szybkie łączenie*.


Uwaga

Z reguły po zalogowaniu się do konta FTP znajdujemy się w głównym katalogu konta WWW, czyli tym, do którego należy wgrać pliki składające się na witrynę. Nie jest to jednak regułą. Jeśli pojawi się np. katalog o nazwie *public_html*, prawdopodobnie to tam należy wgrać pliki. Dokładną informację o sposobie wgrywania danych na stronę zawsze znajdziemy na stronie WWW dostawcy usług hostingowych.

Po chwili połączenie zostanie nawiązane i w polu z prawej strony (numer 5 na rysunku 1.17) pojawi się zawartość głównego katalogu naszego konta WWW. Jeśli jest ono puste, możemy przystąpić do wgrywania pliku z naszą witryną, jeżeli jednak znajdziemy w nim plik o nazwie *index.html*, *index.php* lub podobny, lepiej go najpierw skasować[17]. Aby skasować plik (pliki), należy kliknąć go prawym przyciskiem myszy i z menu podręcznego wybrać pozycję *Usuń* (rysunek 1.18). Po skasowaniu danych możemy przystąpić do wgrania naszej strony.



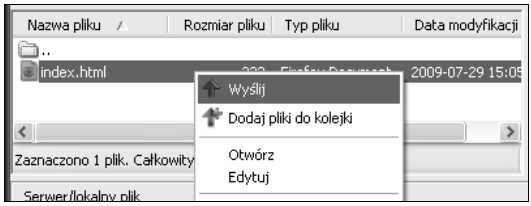
Rysunek 1.18. *Usuwanie pliku z serwera*

Aby wgrać plik z naszego domowego komputera na serwer, należy wykonać następujące czynności (rysunek 1.19):

- Odszukać ten plik w oknie reprezentującym system plików naszego komputera.
- Kliknąć plik prawym przyciskiem myszy i z menu podręcznego wybrać pozycję *Wyślij*.
- Plik zostanie przesłany i pojawi się w oknie reprezentującym system plików serwera.

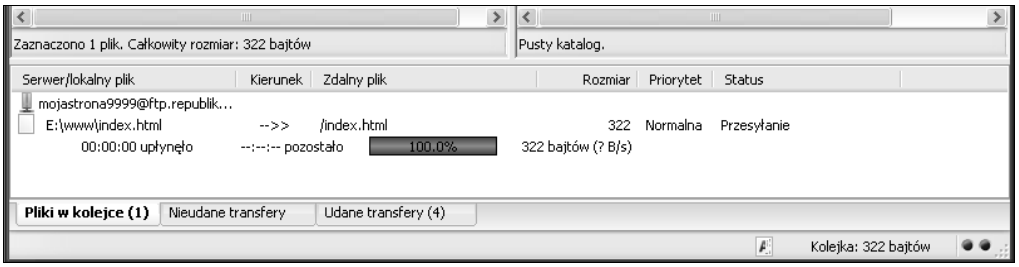
[16] Dane te praktycznie zawsze są dostępne na stronach dostawcy. Jeśli nie możemy ich znaleźć, najlepiej skontaktować się z pomocą techniczną — z pewnością uzyskamy wtedy potrzebne informacje.

[17] Na niektórych kontach po utworzeniu automatycznie jest umieszczany plik (pliki) tworzący stronę domyślną. Taka strona zwykle zawiera informację, że witryna jest dopiero w trakcie tworzenia. Taki plik (*index.html*, *index.php* lub podobny) najlepiej po prostu skasować przed wgraniem naszych danych.



Rysunek 1.19. *Wysyłanie pliku na serwer*

Postęp transferu danych będzie można obserwować w polu znajdującym się w dolnej części okna aplikacji, tak jak jest to widoczne na rysunku 1.20.



Rysunek 1.20. *Postęp transferu pliku na serwer*

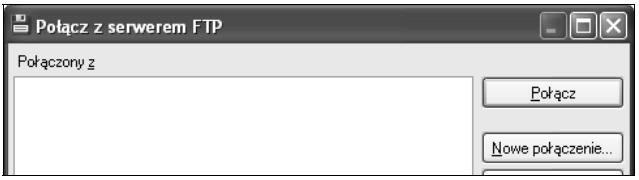
> Jeżeli plików do wysłania jest więcej, przed wywołaniem menu podręcznego należy je zaznaczyć. Można to zrobić dokładnie w taki sam sposób jak w Eksploratorze Windows. Aby zaznaczyć wszystkie naraz, używamy kombinacji klawiszy *Ctrl+A*. Aby zaznaczyć grupę następujących po sobie plików, klikamy pierwszy, wciskamy klawisz *Shift* i klikamy ostatni. Aby zaznaczyć tylko kilka wybranych plików, klikamy je, trzymając wciśnięty klawisz *Ctrl*.

Po wgraniu pliku (plików) z witryną na serwer można wpisać adres strony do przeglądarki i sprawdzić, czy wszystko działa prawidłowo.

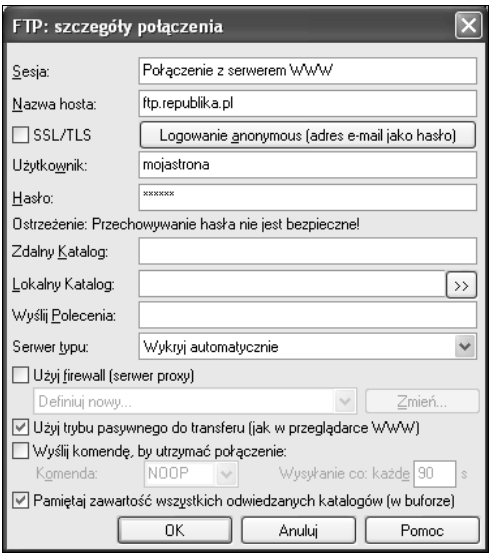
Wysyłanie danych za pomocą menedżera plików

Jeśli na naszym komputerze jest zainstalowany menedżer plików typu Total Commander czy Free Commander, nie musimy instalować aplikacji typu FileZilla. Takie menedżery plików zawierają moduły FTP i z powodzeniem mogą być użyte do transmisji danych na serwer. Zobaczmy, jak taki proces odbywałby się w przypadku aplikacji Total Commander.

Po uruchomieniu programu należy przejść na prawy lub lewy panel i wcisnąć kombinację klawiszy *Ctrl+F* lub też z menu *Sieć* wybrać pozycję *FTP Nowe połączenie*. Na ekranie pojawi się wtedy lista zdefiniowanych połączeń. Jeśli to pierwsze uruchomienie modułu FTP, będzie ona pusta (rysunek 1.21). Aby dodać nowe połączenie, klikamy przycisk *Nowe połączenie*. Pojawi się okno definicji nowego połączenia (rysunek 1.22).



Rysunek 1.21. *Pusta lista zdefiniowanych połączeń FTP*

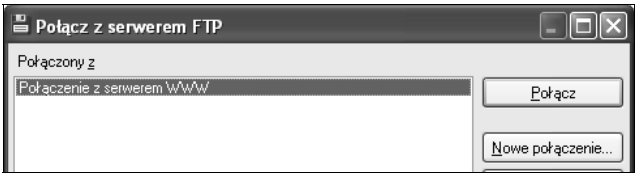


Rysunek 1.22. *Okno definicji nowego połączenia FTP*

Wprowadzamy w nim następujące dane:

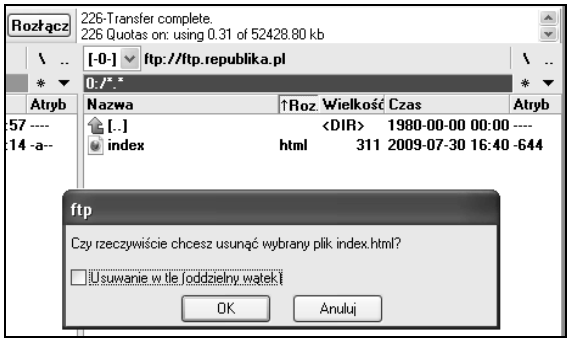
- W polu *Sesja* — nazwę dla połączenia. Może być ona dowolna. Wpisana nazwa będzie widoczna na liście z rysunku 1.21.
- W polu *Nazwa hosta* — adres właściwego dla naszego konta serwera FTP (np. *ftp.republika.pl*).
- W polu *Użytkownik* — nazwę konta (np. *mojastrona*).
- W polu *Hasło* — hasło do konta.

Po wprowadzeniu danych klikamy przycisk *OK*. Powrócimy wtedy do ekranu wyboru połączenia (rysunek 1.21), na którym pojawi się to przed chwilą zdefiniowane (rysunek 1.23). Wystarczy teraz wskazać utworzone przed chwilą połączenie FTP i kliknąć przycisk *Połącz*. Połączenie zostanie nawiązane, a w wybranym panelu pojawi się lista plików i katalogów znajdujących się na serwerze.



Rysunek 1.23. *Lista dostępnych połączeń FTP*

Podobnie jak miało to miejsce w przypadku korzystania z FileZilla, jeśli na serwerze znajdują się pliki składające się na domyślną stronę (*index.html*, *index.php* lub podobne), należy je usunąć. Aby usunąć plik, należy go wskazać i wcisnąć klawisz *F8* (lub *Del*) oraz potwierdzić chęć wykonania operacji, klikając przycisk *OK* (rysunek 1.24).



Rysunek 1.24. *Usuwanie pliku z serwera w programie Total Commander*

Po usunięciu pliku (plików) można przystąpić do przesłania naszej strony na serwer. Pliki (i katalogi) kopiuje się tak samo jak w przypadku dysków lokalnych. Wystarczy zaznaczyć dane przeznaczone do przesłania i wcisnąć klawisz *F5*.

Uwaga

Aby w Total Commanderze zaznaczyć grupę plików, wystarczy wcisnąć klawisz *+* umieszczony na klawiaturze numerycznej. Zaznaczenie wszystkich danych z danego katalogu można też wykonać, wciskając kombinację klawiszy *Ctrl+A*. Pojedyncze pliki i podkatalogi najłatwiej zaznaczać, wciskając klawisz *Insert*.

Własny adres internetowy

Jeśli chcemy mieć swój własny adres internetowy, czyli domenę (np. *helion.pl* czy *marcinlis.com*), musimy go wykupić u tzw. rejestratora. Taką usługę zapewnia większość dostawców usług hostingowych, jednocześnie możemy więc zakupić miejsce na serwerze i adres. Adresy firm zajmujących się rejestracją domen bez problemu znajdziemy, wpisując w przeglądarkę hasło *domeny*. Często przy zakupie domeny otrzymamy też gratis podstawowe konto WWW o niewielkiej pojemności, co może być dobrym rozwiązaniem na początku przygody z własną stroną internetową. Cena domeny zależy od jej rodzaju. Przykładowe relacje cen zostały zobrazowane w tabeli 1.1.

Tabela 1.1. *Relacje cenowe wybranych rodzajów domen internetowych*[18]

Końcówka nazwy domenowej	Cena
.pl	100 zł
.com.pl	75 zł
.waw.pl	40 zł
.com	30 – 50 zł
.net	30 – 50 zł
.org	30 – 50 zł

[18] Dane aktualne w drugiej połowie 2009 roku. Podane ceny nie uwzględniają ewentualnych promocji i rabatów.

Rejestracja domeny standardowo jest ważna przez rok, chyba że zdecydujemy się na wykupienie dłuższego okresu. To oczywiście wiąże się z większym wydatkiem (wielokrotnością podanych kwot). Często też cena pierwotnej rejestracji jest znacznie obniżona, nawet do 1 zł czy nawet 0,01 zł, ale konieczne jest wtedy kontynuowanie rejestracji przez kolejne lata w tej samej firmie rejestrującej, co wcale nie musi być atrakcyjne. Nie należy więc ulegać chwilowym promocjom, ale zawsze dokładnie zapoznać się z warunkami rejestracji.

Po rejestracji domeny konieczne jest powiązanie jej z kontem WWW, na którym znajduje się nasza strona. Jeśli zarówno konto, jak i domena zostały utworzone u tego samego dostawcy, czynność ta będzie uproszczona (w przypadku darmowych kont WWW takie powiązanie z reguły nie jest możliwe — konieczne jest wykupienie pełnego konta). Odpowiednie opcje znajdziemy w panelu administracyjnym. Ponieważ jednak sposób administracji kontem WWW i kontem służącym do rejestracji domen będzie różny u różnych dostawców usług, nie można opisać jednej procedury wykonania niezbędnych czynności. Wszelkie niezbędne dane i opisy zawsze jednak znajdziemy na stronach naszego dostawcy usług hostingowych. W razie wątpliwości należy się kontaktować z pomocą techniczną danej firmy.

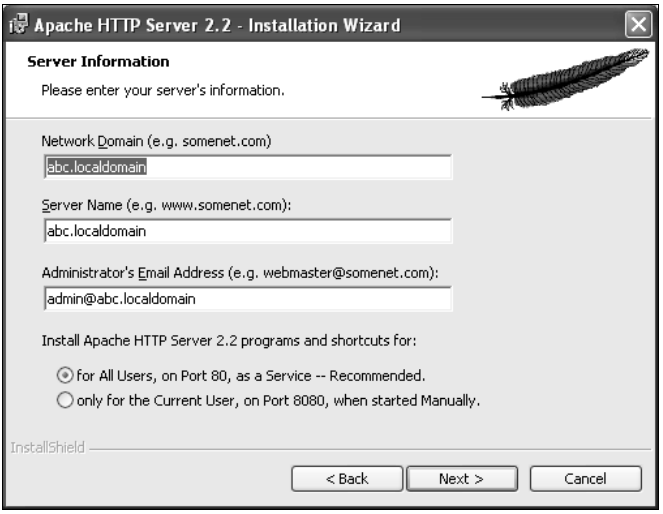
Serwer WWW na domowym komputerze

Jeżeli nie chcemy otwierać konta WWW w internecie, u dostawcy usług hostingowych, ale chcielibyśmy testować naszą stronę w zbliżonym środowisku, możemy uruchomić nasz prywatny serwer WWW na domowym komputerze. Najlepiej użyć do tego oprogramowania serwera WWW o nazwie Apache. Jest on darmowy i rozpowszechniany na zasadach wolnego oprogramowania. Serwer Apache można znaleźć pod adresem *http://httpd.apache.org/*. Wersja dla Windows jest dystrybuowana w postaci pliku instalacyjnego *msi* (dostępne są także kody źródłowe, którymi nie będziemy się zajmować). Dla wersji 2.2.12 jest to *apache_2.2.12-win32-x86-no_ssl.msi*[19]. Po pobraniu należy go uruchomić. Proces instalacji jest typowy jak dla każdej innej aplikacji w systemie Windows. W jego trakcie w oknie konfiguracyjnym, widocznym na rysunku 1.25, można wprowadzić własne nazwy dla domeny i serwera. Ma to jednak znaczenie tylko w przypadku uruchamiania serwera pracującego w internecie. W przypadku instalacji na komputerze domowym (mówimy też w takiej sytuacji o komputerze lokalnym lub o lokalnej instalacji) w celach testowych najlepiej pozostawić wartości domyślne proponowane przez program instalacyjny.

Kolejny ekran pozwala na wybór typu instalacji (rysunek 1.26). Najlepiej skorzystać z opcji typowej (*Typical*). Osoby zaawansowane mogą oczywiście wybrać również instalację użytkownika (*Custom*). Kliknięcie przycisku *Next* spowoduje przejście do kolejnego ekranu, na którym możliwy będzie wybór katalogu docelowego (rysunek 1.27). W zwykłych zastosowaniach nie ma jednak potrzeby zmieniania folderu zaproponowanego przez instalator (typowo jest to *C:\Program Files\Apache Software Fundation\Apache2.2*).

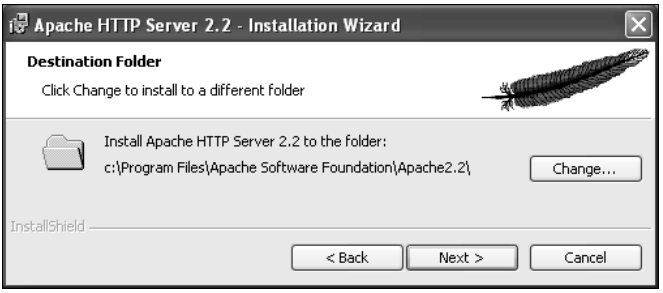
[19] Dostępny jest także plik *apache_2.2.12-win32-x86-openssl-0.9.8e.msi* zawierający moduł szyfrowania. Dla potrzeb niniejszego kursu nie jest on jednak potrzebny.

Rysunek 1.25. *Wybór domeny i nazwy serwera*



Rysunek 1.26. *Ekran wyboru typu instalacji*

Rysunek 1.27. *Ekran wyboru katalogu docelowego*

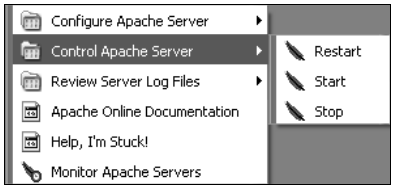


Po zakończeniu instalacji serwer zostanie automatycznie uruchomiony, co będzie sygnalizowane przez zielony kolor ikony znajdującej się na pasku zadań (rysunek 1.28). To oznacza, że nasz komputer przemienił się w serwer WWW (jeśli ikona nie jest widoczna, można ją wywołać z menu *Start/Wszystkie programy/Apache HTTP Server/Control Apache Servers/Monitor Apache Servers*).

Rysunek 1.28.
Ikona obrazująca stan serwera Apache

Stan serwera może być kontrolowany z poziomu głównego menu lub za pomocą aplikacji monitorującej. W tym pierwszym przypadku dostęp do opcji sterujących uzyskamy, wybierając z menu *Start/Wszystkie programy/Apache HTTP Server/* grupę *Control Apache Server* (rysunek 1.29). Uzyskamy wtedy dostęp do pozycji *Start* (uruchamiającej serwer), *Stop* (zatrzymującej serwer) i *Restart* (restartującej serwer).

Rysunek 1.29.
Grupa menu kontrolująca zachowanie serwera

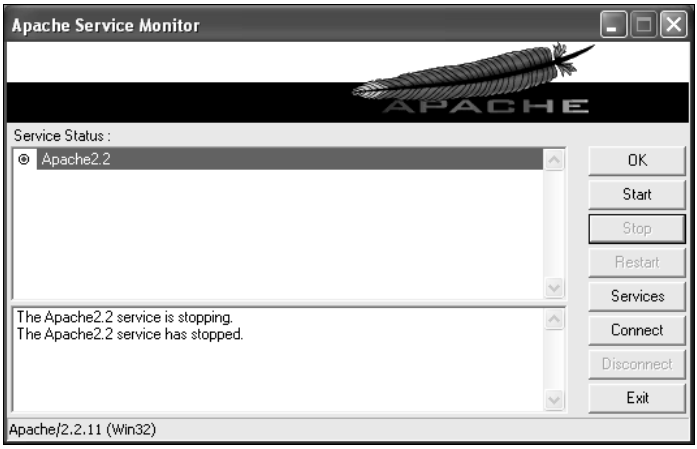


Okno aplikacji kontrolującej zachowanie serwera możemy natomiast wywołać, klikając prawym przyciskiem myszy znajdującą się na pasku zadań, a opisaną wyżej, ikonę serwera. Spowoduje to wywołanie menu podręcznego, w którym należy wybrać pozycję *Open Apache Monitor* (rysunek 1.30).

Rysunek 1.30.
Wywoływanie okna aplikacji monitorującej stan serwera

Po wykonaniu tej czynności na ekranie pojawi się okno monitora serwera widoczne na rysunku 1.31. Górne pole zawiera informacje o stanie serwera (zatrzymany bądź uruchomiony), dolne zawiera natomiast komunikaty techniczne. Widoczny z prawej strony pasek przycisków pozwala z kolei na wykonywanie rozmaitych operacji. Interesujące dla nas są przyciski *Start* i *Stop* wykonujące uruchomienie i zatrzymanie serwera.

Rysunek 1.31.
Okno monitora serwera

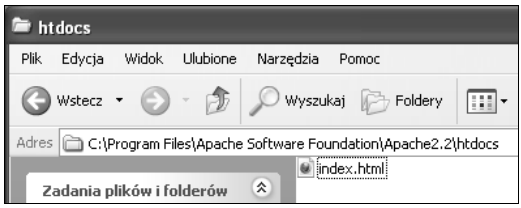


Gdy Apache pracuje (ikona symbolizująca stan jest zielona), możemy traktować nasz własny komputer jak prawdziwy serwer WWW. Wystarczy uruchomić przeglądarkę i w polu adresu wpisać *http://localhost/*. To specjalny adres przypisany naszemu domowemu komputerowi (można też użyć numerycznej postaci *http://127.0.0.1/*). Zobaczymy napis `It works!` (rysunek 1.32). To znak, że instalacja zakończyła się pomyślnie, a serwer działa prawidłowo.

Rysunek 1.32. *Strona domyślna serwera Apache*

Pozostaje nam umieścić na serwerze naszą własną witrynę. Tym razem nie potrzebujemy żadnych dodatkowych aplikacji. Wystarczy skopiować pliki HTML składające się na stronę (np. plik *index.html* o treści z listingu 1.2) do odpowiedniego katalogu i odświeżyć stronę o adresie *http://localhost/*. Tym katalogiem jest podkatalog *htdocs* znajdujący się w katalogu, w którym został zainstalowany Apache. Jeśli więc w trakcie instalacji Apache'a nie zmieniliśmy katalogu docelowego (rysunek 1.27), pliki z witryną należy umieścić w katalogu: *C:\Program Files\Apache Software Fundation\Apache2.2\htdocs* (rysunek 1.33), usuwając wcześniej znajdujący się tam plik *index.html*.

Rysunek 1.33. *Pliki z witryną powinny znaleźć się w katalogu htdocs*



Wpisanie adresu *http://localhost/* ma takie samo znaczenie, jak wpisanie każdego innego adresu internetowego (nasz komputer jest teraz pełnoprawnym serwerem WWW), czyli oznacza pobranie domyślnej strony WWW na serwerze o nazwie *localhost* (komputerze lokalnym, domowym). W domyślnej konfiguracji serwera Apache jest to strona znajdująca się w pliku *index.html*. Jeżeli chcemy uzyskać dostęp do strony znajdującej się w pliku o innej nazwie, przykładowo *main.html*, musimy tę nazwę uwzględnić w adresie. Wtedy odwołanie miałoby postać *http://localhost/main.html*.

W katalogu *htdocs* możemy też utworzyć podkatalogi (w tym celu należy użyć standardowych mechanizmów Windowsa). W takich podkatalogach też możemy umieszczać strony WWW. Przykładowo, jeśli w katalogu *htdocs* (czyli głównym katalogu serwera WWW) utworzymy podkatalog o nazwie *strony*, a w nim umieścimy plik *mojastrona.html*, to aby zobaczyć w przeglądarce stronę WWW zapisaną w tym pliku, musimy użyć odwołania w postaci *http://localhost/strony/mojastrona.html*.

Lekcja 3. Struktura (X)HTML

- Czym są znaczniki (X)HTML?
- Jaka jest konstrukcja znacznika?
- Co to są atrybuty?
- Jakie atrybuty są standardowe?
- Co to są encje?
- Jak uzyskiwać znaki i symbole specjalne?
- Jaki wpływ na witrynę mają tzw. białe znaki?
- Czy warto używać komentarzy?
- Jak wpływać na zachowanie wyszukiwarek i robotów sieciowych?

Znaczniki HTML

Strony WWW tworzone są w języku HTML lub XHTML za pomocą tak zwanych znaczników (z ang. *tags*). Ich przykłady poznaliśmy już w poprzednich lekcjach. To elementy ujęte w nawias ostry np.: `<html>`, `<body>`, `<p>`. Dzięki nim budowana jest struktura strony, a jej elementy otrzymują dodatkowe znaczenie. Na przykład fragment tekstu może być oznaczony jako akapit, możemy też spowodować, że czcionka będzie pogrubiona, dodać odnośnik itp. Schematycznie znacznik można przedstawić jako:

```
<znacznik>zawartość znacznika</znacznik>
```

W ten sposób został zdefiniowany element strony WWW. Widać wyraźnie, że składa się ze znacznika (z ang. *tag*) otwierającego `<znacznik>`, zamykającego `</znacznik>` oraz zawartej między nimi treści. Znacznik zamykający jest identyczny z otwierającym, ale zawiera dodatkowo znak ukośnika. Zawartość elementu może być dowolna, z reguły jest to jakiś tekst. Całej zawartości znacznika nadawana jest funkcja określona przez ten znacznik. Przykładowo, używaliśmy takiej konstrukcji:

```
<p>To jest akapit tekstowy.</p>
```

Oznacza ona, że ciąg znaków `To jest akapit tekstowy.` na być traktowany jako akapit tekstowy. Możemy też umieścić wiele takich znaczników (elementów) jeden po drugim, np.:

```
<p>Pierwszy tekst</p>
<p>Drugi tekst</p>
<p>Trzeci tekst</p>
```

W taki sposób zdefiniowaliśmy występujące jeden po drugim trzy akapity tekstowe. Stronę zawierającą taki kod przeglądarka zinterpretuje tak, jak zostało to przedstawione na rysunku 1.34.

Rysunek 1.34. *Trzy następujące po sobie akapity tekstowe*

Istnieją również znaczniki, które nie mają części zmykającej ani też treści. Mówimy wtedy o znacznikach pustych. Definiują one takie elementy strony jak np. nowy wiersz czy pozioma linia. Tego typu znacznik ma schematyczną postać:

```
<znacznik />
```

Użycie ukośnika jest niezbędne, aby zachować zgodność z kodem XHTML. W przypadku HTML można używać znacznika w postaci:

```
<znacznik>
```

Lepiej jednak użyć pierwszej z zaprezentowanych postaci, gdyż pozwala to zachować zgodność kodu z oboma językami[20]. Przykładem tego typu znacznika jest np.: `<br />` łamiący wiersz tekstu (więcej informacji o tym znaczniku znajduje się w rozdziale 2.).

Zagnieżdżanie znaczników

Znaczniki, czy też dokładniej — elementy HTML, mogą być zagnieżdżane. To znaczy, w obrębie treści jednego elementu mogą się znajdować inne. Wydaje się to oczywiste w przypadku znaczników definiujących strukturę (X)HTML. Wystarczy spojrzeć na listing 1.2. Element `<head>` zawiera treść dotyczącą nagłówka dokumentu, a element `<body>` — właściwą treść dokumentu składającą się z innych znaczników. Ta zasada dotyczy też jednak elementów definiujących treść witryny. Możliwa jest np. sytuacja, którą schematycznie można zobrazować tak:

```
<znacznik1><znacznik2>treść</znacznik2></znacznik1>
```

a nawet tak:

```
<znacznik1>treść1<znacznik2>treść2</znacznik2>treść3</znacznik1>
```

Przy takich konstrukcjach ważne jest, aby zawsze najpierw zamykać (umieszczać znacznik kończący zawierający ukośnik) najbliższy możliwy element. To znaczy, że błędna byłaby konstrukcja w postaci:

```
<znacznik1><znacznik2>treść</znacznik1></znacznik2>
```

[20] Nie zawsze tak jest. Czasami nie można w zgodzie ze standardem HTML użyć formy `<znacznik />`. Takie sytuacje będą zaznaczane w dalszej części książki.

Skoro bowiem najpierw został otworzony <znacznik1>, a potem <znacznik2>, to najpierw należy zamknąć znacznik2 (</znacznik2>), a dopiero potem znacznik1 (</znacznik1>). Przykład użycia zagnieżdżonych znaczników znajduje się na listingu 1.7.

Listing 1.7. *Użycie zagnieżdżonych znaczników*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Moja strona WWW</title>
  </head>
  <body>
    <p>Przykład <b>pogrubienia</b> tekstu</p>
    <p>Przykład <b><i>pogrubienia i kursywy</i></b></p>
    <p>Inny przykład <b>pogrubienia i <i>kursywy</i></b></p>
  </body>
</html>
```

W sekcji <body> zostały umieszczone trzy akapity tekstowe zdefiniowane za pomocą znacznika . Pierwszy akapit ma postać:

```
<p>Przykład <b>pogrubienia</b> tekstu</p>
```

We wnętrzu elementu (treść znajdująca się pomiędzy i) został ulokowany tekst, w którym umieszczono dodatkowy znacznik . Ten znacznik powoduje, że ujęty weń tekst zostanie wyświetlony pogrubioną czcionką. W tym przypadku będzie to słowo pogrubienia.

Drugi akapit ma postać:

```
<p>Przykład <b><i>pogrubienia i kursywy</i></b></p>
```

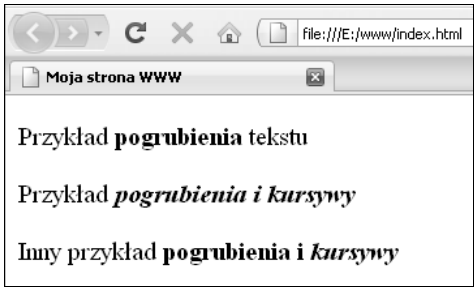
Ta konstrukcja jest podobna do występującej w pierwszym przypadku, ale tym razem we wnętrzu znacznika zostały użyte dwa inne: i . Znacznik powoduje, że ujęty weń tekst będzie wyświetlany za pomocą kursywy. Należy jednak zwrócić uwagę, że ciąg pogrubienia i kursywy znajduje się zarówno we wnętrzu znacznika , jak i znacznika , a zatem ten tekst będzie jednocześnie pogrubiony oraz wyświetlony za pomocą kursywy. Warto zwrócić uwagę, że zgodnie z podaną wyżej zasadą skoro zaraz po znaczniku został użyty znacznik , to znaczniki zamykające zostały użyte w odwrotnej kolejności — najpierw , a potem .

Trzeci akapit ma postać:

```
<p>Inny przykład <b>pogrubienia i <i>kursywy</i></b></p>
```

W nim również użyto wewnętrznych (zagnieżdżonych) elementów i . Tym razem jednak pogrubiony został cały ciąg pogrubienia i kursywy (ujęty w znacznik), ale kursywa została użyta tylko w stosunku do słowa kursywy (ujętego w znacznik).

Gdy zatem wczytamy tak zdefiniowaną witrynę do przeglądarki, ujrzymy widok zaprezentowany na rysunku 1.35.



Rysunek 1.35. *Przykład użycia znaczników zagnieżdżonych do formatowania tekstu*

Znaczników nie możemy jednak mieszać dowolnie. Na przykład te, które służą do formatowania tekstu, możemy podzielić na dwie grupy: blokowe (z ang. *block-level*) oraz wierszowe (z ang. *inline*). Blokowe definiują bloki tekstu, np. akapity, wykazy itp. Wierszowe z reguły zmieniają atrybuty tekstu, np. dodają pogrubienie, kursywę. Jako zasadę można przyjąć, że element blokowy może zawierać dowolną liczbę elementów wierszowych. To wydaje się logiczne — w akapicie można zastosować wiele pogrubień i innych wyróżnień. Nie można jednak użyć konstrukcji odwrotnej — pogrubienie zatem nie może zawierać akapitu. Czyli w elemencie wierszowym nie można umieścić elementu blokowego. Konstrukcja o postaci:

```
<b><p>To jest pogrubiony akapit</p></b>
```

jest więc błędna.

> Mimo że przedstawiona konstrukcja jest nieprawidłowa, przeglądarka wyświetli taki kod poprawnie (na ekranie pojawi się pogrubiony akapit). Dzieje się tak dlatego, że aplikacje tego typu są bardzo odporne na wiele, nawet dużo poważniejszych, błędów w kodzie. Twórcy przeglądarek stosują założenie, że ich produkt powinien jak najlepiej wyświetlić każdą możliwą stronę WWW, nawet jeśli jej autor nie stosuje się do standardów i utworzył niepoprawny kod. Nie oznacza to jednak, że możemy tworzyć niechlujny i niespójny kod (X)HTML. Dlatego też zawsze stosujmy się do standardów i piszmy poprawne strony WWW.

Elementy blokowe zawsze zajmują całą dostępną przestrzeń w poziomie, jeśli więc w kodzie znajdą się jeden obok drugiego, to na witrynie będą prezentowane jeden pod drugim[21]. Przykładowe trzy akapity:

```
<p>pierwszy</p><p>drugi</p><p>trzeci</p>
```

umieszczone są w jednym wierszu, ale na stronie zostaną przedstawione jeden pod drugim (każdy z nich zajmie całą dostępną szerokość). Dzieje się tak właśnie dlatego, że pojedynczy akapit jest elementem blokowym.

Jak jednak odróżnić, które znaczniki można umieszczać w innych, a które nie? Ta wiedza przyjdzie, można powiedzieć, sama wraz z postępem nauki języka (X)HTML.

[21] To zachowanie może zostać zmienione za pomocą stylów CSS.

Atrybuty znaczników

Znacznik otwierający dany element strony WWW może zawierać tak zwane atrybuty. Pojedynczy atrybut składa się z nazwy i przypisanej jej wartości. Za pomocą atrybutów definiujemy dodatkowe parametry znacznika. Schematyczna postać takiej konstrukcji jest następująca:

```
<znacznik atrybut="wartość">treść</znacznik>
```

Wartość atrybutu zawsze należy ująć w cudzysłów[22]. W przypadku gdyby jeden znacznik miał mieć przypisanych wiele atrybutów, oddziela się je od siebie znakiem spacji:

```
<znacznik atrybut1="wartość1" atrybut2="wartość2">treść</znacznik>
```

Jeżeli mamy do czynienia ze znacznikiem pustym, schemat postępowania nie zmienia się. Po prostu znacznik otwierający jest jednocześnie zamykającym:

```
<znacznik atrybut1="wartość1" atrybut2="wartość2" />
```

Pojedynczy atrybut może np. wskazywać nazwę pliku z obrazem graficznym, określać styl danego elementu itp., np.:

```
<img src="obraz.jpg" />
<p style="font-weight:bold">Tekst pogrubiony</p>
```

Atrybuty, które można, a niekiedy wręcz trzeba, zastosować przy znacznikach, będą omawiane wraz z opisem konkretnych znaczników w dalszej części książki. Istnieje jednak pewien zestaw typowych atrybutów, które mogą być stosowane w większości znaczników definiujących elementy witryny. Omówimy je teraz pokrótce. Tę część lekcji można pominąć i powrócić do niej dopiero po zapoznaniu się z konkretnymi znacznikami w rozdziale 2.

Lista typowych atrybutów

Atrybut `acceskey` — umożliwia przypisanie skrótu klawiaturowego pozwalającego na szybki dostęp do danego elementu witryny, gdy do nawigacji używana jest klawiatura. Jako wartość atrybutu należy użyć tylko znaku klawisza aktywującego. Przykład[23]:

```
<input type="text" accesskey="a" />
```

Aktywacja tak wyróżnionego elementu (pola tekstowego, odnośnika itp.) w przeglądarce wymaga jednak wciśnięcia kombinacji składającej się z klawisza aktywującego i klawisza funkcyjnego (z reguły *Alt*, *Shift* itp.). Kombinacje te są zależne od zastosowanej przeglądarki (tabela 1.2).

[22] Wiele przeglądarek zaakceptuje wartości atrybutów podane bez cudzysłowu, ale jest to postępowanie nieprawidłowe, niezgodne ze standardami i może prowadzić do błędów w wyświetlaniu witryny.

[23] Znacznik `<input>` zostanie omówiony w lekcji 12.

Tabela 1.2. *Klawisze aktywujące skróty klawiaturowe dla elementów witryny*

Przeglądarka	Klawisze aktywujące
Firefox	*Alt+Shift*
Google Chrome	*Alt*
Internet Explorer	*Alt*
Konqueror	*Ctrl*
Opera	*Esc+Shift*
Safari	*Ctrl+Option*

Atrybut `class` — określa klasę lub klasy, do których należy atrybut. Zagadnienie to zostanie dokładniej opisane w rozdziale 4. omawiającym style CSS. W skrócie: klasa decyduje o sposobie prezentacji elementu. Wiele elementów strony może należeć do jednej klasy, jeden element może też należeć do wielu różnych klas. W tym drugim przypadku poszczególne nazwy klas należy oddzielić spacjami. Nazwa klasy może składać się jedynie z cyfr, liter, znaku pokreślenia (_) i kreski (-). Przykłady:

```
<p class="akapity_glowne">Tekst akapitu</p>
<p class="akapity_glowne teksty_wyroznione">Tekst akapitu</p>
```

Atrybut `dir` — określa kierunek odczytu tekstu będącego zawartością danego elementu strony. Może przyjmować wartości `ltr` (z ang. *left to right*), co oznacza od lewej do prawej, lub `rtl` (z ang. *right to left*), co oznacza od prawej do lewej. W praktyce bardzo rzadko stosowany. Przykład:

```
<p dir="ltr">Tekst akapitu</p>
```

Atrybut `id` — pozwala nadać danemu elementowi strony unikatowy identyfikator. Taki identyfikator może być później używany do odwoływania się do tego elementu. Nazwa identyfikatora, podobnie jak w przypadku atrybutu `class`, może się składać wyłącznie z liter, cyfr oraz znaków _ i -. Przykład:

```
<p id="akapit1">Tekst akapitu</p>
```

Atrybut `lang` — określa język, w którym został zapisany tekst danego elementu strony. Może być przydatny, gdy np. w tekście witryny umieszczamy wstawki w różnych językach. W typowych zastosowaniach zwykle jest jednak pomijany. Wartością atrybutu powinien być kod języka. Lista wybranych kodów została zebrana w tabeli 1.3. Przykład:

```
<p lang="pl">Tekst akapitu</p>
```

Ten atrybut nie występuje w XHTML 1.1.

Atrybut `style` — określa styl CSS elementu strony WWW. Style zostaną przedstawione w rozdziale 4. Przykład użycia stylu:

```
<p style="font-weight:bold">Przykładowy akapit z pogrubioną czcionką</p>
```

Atrybut `tabindex` — określa kolejność, w której dany element zostanie aktywowany w trakcie przemieszczania się po elementach witryny za pomocą klawisza *Tab*. Przykład użycia:

```
<input type="text" tabindex="2" />
```

Tabela 1.3. *Kody języków stosowane jako wartość atrybutu lang*[24]

Kod	Język	Kod	Język	Kod	Język
cs	czeski	hu	węgierski	no	norweski
de	niemiecki	it	włoski	pl	polski
en	angielski	ja	japoński	ru	rosyjski
fr	francuski	nl	holenderski	zh	chiński

Atrybut `title` — określa tytuł elementu strony. Może być to dowolny tekst. Przeglądarka może wyświetlać taki tytuł jako „podpowiedź" (z ang. *tooltip*), gdy kursor znajdzie się nad danym elementem strony (zostało to zobrazowane na rysunku 1.36). Atrybutu `title` można użyć np. w następujący sposób:

```
<p title="To jest tytuł">Przykładowy tekst akapitu zawierającego atrybut title.</p>
```

Rysunek 1.36. *Wyświetlenie zawartości atrybutu title jako podpowiedzi*

Atrybut `xml:lang` — określa język, w którym został zapisany tekst danego elementu strony. Atrybut charakterystyczny dla XHTML, nie występuje w HTML. Przykładowe kody języków zaprezentowano w tabeli 1.3.

Encje (znaki specjalne)

Niektóre znaki mają w HTML-u specjalne znaczenie. Wiadomo np., że < i > wyróżniają w tekście strony formatujące ją znaczniki. Nieraz jednak będziemy chcieli użyć tych znaków po prostu w tekście witryny. Nie można dopuścić do sytuacji, w której pewne znaki byłyby zabronione ze względu na to, że są używane w języku opisu strony. Jak więc umieścić nawias ostry np. w akapicie? W pierwszej chwili zapewne nasuwa się użycie następującej konstrukcji:

```
<p>Prosty tekst <z nawiasem> ostrym</p>
```

Jednak umieszczenie takiego kodu w sekcji `<body>` i wczytanie do przeglądarki szybko ujawni błąd. Widać to na rysunku 1.37 — zniknęła część tekstu. Pozostał zaś tylko fragment `Prosty tekst ostrym`.

[24] Kody zgodne ze standardem ISO 639-1:2002. Pełną listę można znaleźć w internecie m.in. pod adresem *http://www.w3schools.com/tags/ref_language_codes.asp*.

Rysunek 1.37. *Nieprawidłowo wyświetlona treść akapitu*

Stało się tak, dlatego że przeglądarka, napotkawszy znak < znajdujący się przed literą z, uznała go za rozpoczęcie nowego znacznika. A więc cały fragment <z nawiasem> został potraktowany jak znacznik z atrybutem. Ponieważ nie ma takiego znacznika w żadnym ze standardów (X)HTML, cały ten fragment został zignorowany (przeglądarka nie wyświetla nieznanych jej znaczników).

To jednak wcale nie jedyny problem związany z wyświetlaniem. Są przecież znaki, których nie ma na typowej klawiaturze, a powinna istnieć możliwość wyświetlania ich na stronach WWW. Wtedy właśnie z pomocą przychodzą nam tzw. encje (z ang. *entities*). Encja (z ang. *entity*) to w przypadku HTML pewien kod określający dany znak. Ma on postać:

```
&nazwa;
```

Znaki & i ; wyznaczają początek i koniec encji, a ciąg *nazwa* określa samą encję. Przykładowo < to symbol mniejszości, a > to symbol większości. Często są one używane do definiowania właśnie nawiasu ostrego[25]. A więc prawidłowy zapis tekstu z początku tej części lekcji miałby postać:

```
<p>Prosty tekst &lt;z nawiasem&gt; ostrym</p>
```

Po wczytaniu takiego kodu do przeglądarki zobaczylibyśmy pełny tekst z właściwie odwzorowanymi znakami < i >, tak jak jest to widoczne na rysunku 1.38.

Rysunek 1.38. *Prawidłowe odwzorowanie znaków < i >*

Encje mogą być również zapisywane w postaci numerycznej. Mają wtedy postać:

```
&#kod;
```

o ile stosujemy kody w postaci dziesiętnej, lub:

```
&#xkod;
```

jeżeli stosujemy kody w postaci szesnastkowej. Przykładowo, znak mniejszości można zapisać jako < lub <. Kody encji są maksymalnie czterocyfrowe. W przypadku gdy kod zawiera mniej niż cztery cyfry, można na początku umieścić znak 0.

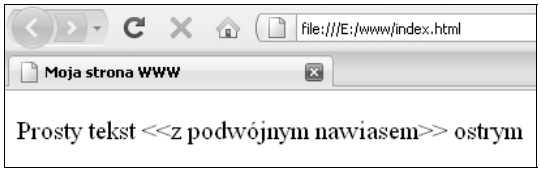
[25] Właściwy nawias ostry (nawias kątowy, z ang. *angle bracket*) powinien być zdefiniowany za pomocą encji ⟨ i ⟩, natomiast znaki cytowania < i > za pomocą encji ‹ i ›.

Prawidłowymi zapisami znaku mniejszości są zatem również < i <. Wszystkie zaprezentowane postaci encji mogą być też dowolnie mieszane w jednym dokumencie. Prawidłowy będzie np. zapis:

```
<p>Prosty tekst &lt;&#60;z podwójnym nawiasem&#x3e;&#x003E; ostrym</p>
```

Przez przeglądarkę zostanie on zinterpretowany, tak jak zostało to przedstawione na rysunku 1.39.

Rysunek 1.39. *Interpretacja różnych typów encji*




Uwaga

Jeżeli stosujemy zapis szesnastkowy kodów encji, w którym występują litery, możemy stosować dowolną ich wielkość, np. &#x003E i &#x003e oznaczają tę samą encję (znak większości). W przypadku korzystania z nazw encji nie ma takiej dowolności. Wtedy wielkość liter ma znaczenie i nie wolno jej zmieniać. Przykładowy zapis Ñ (wielkie N z tyldą) musi mieć dokładnie taką postać, inaczej nie zostanie poprawnie rozpoznany.

W tabelach 1.4 – 1.8 zostały zaprezentowane encje z różnych grup tematycznych wraz z ich nazwami, kodami i opisami.

Tabela 1.4. *Najpopularniejsze i często stosowane encje*

Encja symboliczna	Encja numeryczna szesnastkowa	Encja numeryczna dziesiętna	Reprezentowany symbol	Opis
&quot;	&#x0022;	&#0034;	"	cudzysłów
&amp;	&#x0026;	&#0038;	&	ampersand
&apos;	&#x0027;	&#0039;	'	apostrof prosty
&lt;	&#x003C;	&#0060;	<	znak mniejszości
&gt;	&#x003E;	&#0062;	>	znak większości
				spacja nierozdzielająca

Tabela 1.5. *Encje dotyczące interpunkcji i znaków drukarskich*

Encja symboliczna	Encja numeryczna szesnastkowa	Encja numeryczna dziesiętna	Reprezentowany symbol	Opis
&sect;	&#x00A7;	&#0167;	§	paragraf
&uml;	&#x00A8;	&#0168;	¨	umlaut
&macr;	&#x00AF;	&#0175;	¯	akcent prosty poziomy (kreska górna)
&acute;	&#x00B4;	&#0180;	´	akcent ostry

Tabela 1.5. *Encje dotyczące interpunkcji i znaków drukarskich — ciąg dalszy*

Encja symboliczna	Encja numeryczna szesnastkowa	Encja numeryczna dziesiętna	Reprezentowany symbol	Opis
·	·	·	·	kropka środkowa
ˆ	ˆ	ˆ	ˆ	akcent cyrkumfleksowy (daszek)
˜	˜	˜	˜	tylda
				średnia spacja (en space)
				długa spacja (em space)
				krótka spacja (thin space)
–	–	–	–	półpauza (en dash)
—	—	—	—	pauza, myślnik (em dash)
‘	‘	‘	‘	lewy apostrof górny
’	&#x2019	’	’	prawy apostrof górny
‚	&#x201A	‚	‘	lewy apostrof dolny
“	“	“	“	lewy cudzysłów górny
”	”	”	”	prawy cudzysłów górny
„	„	„	„	lewy cudzysłów dolny
‹	‹	‹	‹	lewy cytat pojedynczy
›	›	›	‹	prawy cytat pojedynczy

Tabela 1.6. *Encje dotyczące symboli walut*

Encja symboliczna	Encja numeryczna szesnastkowa	Encja numeryczna dziesiętna	Reprezentowany symbol	Opis
¢	¢	¢	¢	cent
£	£	£	£	funt brytyjski
¤	¤	¤	¤	znak waluty
¥	¥	¥	¥	jen
€	€	€	€	euro

Tabela 1.7. *Encje dotyczące symboli matematycznych*

Encja symboliczna	Encja numeryczna szesnastkowa	Encja numeryczna dziesiętna	Reprezentowany symbol	Opis
¬	¬	¬	¬	negacja
°	°	°	°	stopnie
±	±	±	±	plus-minus

Tabela 1.7. *Encje dotyczące symboli matematycznych — ciąg dalszy*

Encja symboliczna	Encja numeryczna szesnastkowa	Encja numeryczna dziesiętna	Reprezentowany symbol	Opis
²	²	²	²	druga potęga
³	³	³	³	trzecia potęga
¹	¹	¹	¹	pierwsza potęga
×	×	×	×	mnożenie
÷	÷	÷	÷	dzielenie
ƒ	ƒ	ƒ	*f*	funkcja
‰	‰	‰	‰	promil
&exists;	∃	∃	∃	istnieje
∅	∅	∅	Ø	pusty zbiór
∈	∈	∈	∈	należy do (jest elementem)
∉	∉	∉	∉	nie należy do (nie jest elementem)
∑	∑	∑	Σ	suma
−	−	−	–	minus
∗	∗	∗	*	asterisk (gwiazdka)
√	√	√	√	pierwiastek kwadratowy
∞	∞	∞	∞	nieskończoność
∠	∠	∠	∠	kąt
∧	∧	∧	∧	logiczne AND
∨	∨	∨	∨	logiczne OR
∩	∩	∩	∩	iloczyn zbiorów
∪	∪	∪	∪	suma zbiorów
∫	∫	∫	∫	całka
≈	≈	≈	≈	zbliżony
≠	≠	≠	≠	różny
≡	≡	≡	≡	identyczny
≤	≤	≤	≤	mniejszy lub równy
≥	≥	≥	≥	większy lub równy

Tabela 1.8. *Encje dotyczące różnych znaków specjalnych*

Encja symboliczna	Encja numeryczna szesnastkowa	Encja numeryczna dziesiętna	Reprezentowany symbol	Opis
¦	¦	¦	¦	Przerywana kreska pionowa
©	©	©	©	Znak copyright
®	®	®	®	Znak registered (wszystkie prawa zastrzeżone)
¼	¼	¼	¼	jedna czwarta
½	½	½	½	jedna druga
¾	¾	&#018A;	¾	trzy czwarte
¿	¿	¿	¿	odwrócony pytajnik
™	™	™	™	znak trademark (znak towarowy zastrzeżony)

Spacja, tabulacja i znak nowego wiersza

Tak jak zostało to wspomniane w lekcji 1., kod (X)HTML można formatować dowolnie, wstawiając między znaczniki spacje, znaki tabulacji bądź znaki nowego wiersza (znak nowego wiersza powstaje, gdy wciskamy klawisz *Enter*). Są to tzw. białe znaki. Nie wpływają one na sposób interpretacji strony przez przeglądarkę. Jeśli występują między znacznikami, są po prostu ignorowane[26]. Dlatego właśnie możemy praktycznie dowolnie formatować kod witryny. Między kolejnymi akapitami tekstowymi możemy wprowadzić wiele znaków nowego wiersza, a nie będzie to miało wpływu na sposób prezentacji witryny przez przeglądarkę. Kod z listingu 1.8 (została przedstawiona tylko sekcja <body>) zostanie potraktowany tak samo jak ten z listingu 1.7.

Listing 1.8. *Odstępy między akapitami w kodzie źródłowym*

```
<body>
  <p>Przykład <b>pogrubienia</b> tekstu</p>


  <p>Przykład <b><i>pogrubienia i kursywy</i></b></p>


  <p>Inny przykład <b>pogrubienia i <i>kursywy</i></b></p>
</body>
```

[26] Ściśle rzecz ujmując, to stwierdzenie nie zawsze jest prawdziwe. Występujące między znacznikami białe znaki mogą być interpretowane jako węzły tak zwanego drzewa dokumentu (X)HTML. To jednak zagadnienie zaawansowane i istotne tylko w pewnych sytuacjach przy programowaniu z użyciem języków skryptowych. Szczegółowe wyjaśnienie można znaleźć m.in. w publikacji *JavaScript. Praktyczny kurs* (*http://helion.pl/ksiazki/jscpk.htm*).

Wprowadzone między kolejnymi akapitami odstępy nie mają po prostu żadnego znaczenia dla interpretacji strony. To tylko zmieniony sposób zapisu.

Podobne zasady obowiązują przy zapisie samych znaczników. Znacznik może przecież zawierać atrybuty. Wcale jednak nie muszą być one zapisane w jednym wierszu i oddzielane od siebie jednym znakiem spacji. Mogą też podlegać formatowaniu. Jest to ważne, gdy atrybutów jest wiele. Wtedy zamiast tworzyć mało czytelną konstrukcję w postaci:

```
<znacznik atrybut1="wartość1" atrybut2="wartość2" atrybut2="wartość2">treść
znacznika</znacznik>
```

lepiej rozbić ją na kilka wierszy. O wiele czytelniejszy będzie przecież zapis:

```
<znacznik atrybut1="wartość1"
          atrybut2="wartość2"
          atrybut2="wartość2">
treść znacznika
</znacznik>
```

W takiej sytuacji każdy biały znak inny niż spacja jest traktowany tak jak pojedyncza spacja, a każda dowolna sekwencja białych znaków również tak jak jedna spacja. Tego typu zapis będzie występował na listingach w dalszej części książki.

Choć może wydawać się to dziwne, podobne zasady obowiązują również przy zapisie treści wyświetlanej na stronie. Akapit tekstowy (a także inny element strony odpowiedzialny za wyświetlanie treści — będzie o tym mowa w rozdziale 2.) można zapisać w wielu wierszach, a w jego treści umieścić wiele spacji czy znaków tabulacji. To jednak nie będzie miało żadnego wpływu na postać tekstu wyświetlaną w przeglądarce! Każda sekwencja białych znaków zostanie zamieniona na pojedynczy znak spacji. Spójrzmy na kod przedstawiony na listingu 1.9.

Listing 1.9. *Nietypowo zapisany akapit tekstowy*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Moja strona WWW</title>
  </head>
  <body>
    <p>To   jest

       akapit    te
                              kstowy.</p>
  </body>
</html>
```

Struktura kodu jest taka sama jak w przypadku listingu 1.4, jednak zupełnie inaczej zapisana jest treść akapitu zdefiniowanego za pomocą znacznika <p>. Tym razem jest mało czytelna, to jednak tylko demonstracja. Zgodnie z tym, co zostało napisane powyżej, wszystkie ciągi białych znaków (wszelkiego rodzaju odstępów) zostaną potraktowane tak, jakby były jednym znakiem spacji. Tym samym po wczytaniu takiej strony do prze-

glądarki zobaczymy całkiem czytelną treść, tak jak zostało to zaprezentowane na rysunku 1.40. Jedynym mankamentem jest odstęp występujący w wyrazie `tekstowy`. Jest to interpretacja kodu zgodna z przedstawionymi zasadami.

Rysunek 1.40. *Akapit z listingu 1.8. wyświetlony przez przeglądarkę*

W tym miejscu zapewne nasuwają się pytania:

- W jaki sposób podzielić kod strony na osobne wiersze, skoro podział w kodzie źródłowym jest ignorowany?
- Jak uzyskać odstępy dłuższe niż jedna spacja?

Odpowiedzią są odpowiednie znaczniki. Zostaną przedstawione już w lekcji 4.

Komentarze

W kodzie strony WWW można umieszczać komentarze. To konstrukcja niedoceniana przez początkujących autorów witryn, a jednak bardzo przydatna. W komentarzu można umieścić dowolną informację, najczęściej o przeznaczeniu danego fragmentu kodu. W przypadku prostych witryn nie ma to znaczenia, ale wraz ze wzrostem ilości kodu i skomplikowania strony staje się to nieocenione. Dzięki takim dodatkowym informacjom łatwo jest zorientować się w strukturze kodu, a więc również aktualizować go i poprawiać. Komentarz może też zawierać takie dane jak historia wprowadzanych zmian czy informacje o autorze. Najważniejsze jest jednak wspomniane opisywanie struktury kodu.

Komentarz w (X)HTML zaczyna się od sekwencji `<--`, a kończy na sekwencji `-->`. Wszystko to, co znajdzie się pomiędzy tymi znacznikami, będzie ignorowane przez przeglądarkę w procesie przetwarzania witryny. Dane te nie zostaną jednak usunięte z kodu i zawsze będą widoczne. Komentarz możemy umieścić w jednym wierszu za pewnym znacznikiem, np.:

```
<p><-- To jest przykładowy komentarz za znacznikiem. -->
```

Może też występować samodzielnie, zajmując cały wiersz:

```
<p>Akapit numer 1</p>
<--Ten komentarz zajmuje jeden wiersz kodu.-->
<p>Akapit numer 2</p>
```

Komentarz może też składać się z wielu wierszy kodu:

```
<-- Ten komentarz
składa się w trzech
wierszy.
-->
```

Ta ostatnia możliwość jest często wykorzystywana do tymczasowego wyłączania w celach testowych fragmentów kodu. Jeśli bowiem część znaczników ujmiemy w komentarz, np.:

```
<--
<p>Żaden z tych akapitów</p>
<p>nie zostanie wyświetlony!</p>
-->
```

to przeglądarka pominie je w trakcie przetwarzania kodu strony.

Struktura dokumentu raz jeszcze

Ogólna struktura strony WWW napisanej w HTML i XHTML została przedstawiona w lekcji 1. Warto jednak rozwinąć ten temat, zaczynając od znacznika `<html>` (tę część lekcji można na razie pominąć i przejść od razu do rozdziału 2.). W przypadku kodu XHTML miał on postać:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl">
```

i nie wymagał już żadnych dodatkowych parametrów. W przypadku kodu HTML przedstawiony znacznik nie miał żadnych atrybutów:

```
<html>
```

To prawidłowe postępowanie, aczkolwiek zalecane jest użycie dodatkowego atrybutu określającego główny język strony. Dobrze jest więc użyć atrybutu `lang` (opisanego w sekcji „Atrybuty znaczników"), np.:

```
<html lang="pl">
```

Wtedy głównym językiem strony będzie polski. To oczywiście nie oznacza, że na witrynie nie będzie można używać innych języków, np. jako cytatów. To jedynie informacja, sugestia dla przeglądarki, że ma spodziewać się języka polskiego.

Sekcja <head>

Znacznie bardziej rozbudowana może być również sekcja `<head>`, czyli nagłówek witryny. Żadne dane z nagłówka nie są wyświetlane w treści witryny, ale może dostarczać on wiele informacji dla przeglądarki, a także — co ważne — dla wyszukiwarek internetowych. Główne znaczniki sekcji `<head>` to:

- `<title>` — definiuje tytuł strony, był omówiony w lekcji 1.
- `<base>` — definiuje bazowy URL dla odnośników (ta kwestia nie będzie omawiana).
- `<link>` — określa odwołania do zasobów zewnętrznych, np. stylów lub skryptów (ta kwestia zostanie omówiona w lekcji 14.).
- `<meta>` — określa rozmaite dane dodatkowe (tzw. metadane).

Jedną z postaci znacznika `<meta>` już poznaliśmy. Była to:

```
<meta http-equiv="Content-Type" content="text/html; charset=kodowanie">
```

określająca typ dokumentu oraz rodzaj kodowania znaków. Dzięki niej mogliśmy ustalić sposób zapisu m.in. polskich liter. Informacje podane w tej lekcji pozwalają już stwierdzić, że taka postać znacznika zawiera po prostu dwa atrybuty: `http-equiv` oraz `content`. Oprócz tej kombinacji może też wystąpić druga: `name` oraz `content`. Obie te kombinacje umożliwiają dostarczenie rozmaitych informacji dodatkowych o witrynie. Najpopularniejsze z nich zostaną omówione w poniższych punktach.

Opis strony

Opis strony to krótkie podsumowanie tego, o czym jest witryna i co można na niej znaleźć. Jest on często używany przez wyszukiwarki i może pojawiać się w wynikach wyszukiwania wraz z tytułem zdefiniowanym za pomocą znacznika `<title>`. Opis można dodać za pomocą atrybutu `name` o wartości `description`, schematycznie:

```
<meta name ="description" content="opis strony">
```

Zawartość atrybutu `content` nie powinna być zbyt długa. Maksymalnie dwa, trzy zdania.

Słowa kluczowe

Można zdefiniować zestaw słów kluczowych najlepiej pasujących do naszej strony. Ma to znaczenie dla wyszukiwarek internetowych indeksujących zasoby sieci. Przykłady słów kluczowych: HTML, kurs, WWW. Słowa kluczowe dodamy, stosując atrybut `name` o wartości `keywords`. Poszczególne słowa umieszczamy jako wartość atrybutu `content`, oddzielając je przecinkami, schematycznie:

```
<meta name="keywords" content="słowo1,słowo2,słowo3">
```

Słów kluczowych nie należy umieszczać zbyt wiele. Zwykle wystarczy kilka, maksymalnie kilkanaście.

Autor

Aby dodać informacje o autorze strony (imię, nazwisko, pseudonim), należy skorzystać z atrybutu `name` o wartości `author`, schematycznie:

```
<meta name="author" content="Imię Nazwisko">
```

Ta informacja nie jest nigdzie wyświetlana, ale jest używana przez wyszukiwarki i inne automaty indeksujące zasoby WWW.

Generator strony

Atrybut `name` o wartości `generator` jest używany przez narzędzia tworzące bądź wspomagające tworzenie stron WWW. W takim przypadku atrybut `content` zawiera po prostu nazwę i ewentualnie inne informacje o danym narzędziu (np. numer wersji). Schematycznie wygląda to tak:

```
<meta name="generator" content="nazwa narzędzia">
```

Wygaśnięcie (data ważności)

Istnieje możliwość określenia daty ważności dokumentu HTML. Służy do tego atrybut `http-equiv` o wartości `expires` w połączeniu z atrybutem `content` zawierającym datę utraty ważności przez dokument. Schematycznie taka konstrukcja wygląda następująco:

```
<meta http-equiv="expires" content="czas">
```

Wartość *czas* musi zawierać datę wygaśnięcia ważności w formacie[27]:

```
ttt, dd mmm rrrr gg:ii:ss GMT
```

gdzie:

- *ttt* to trzyliterowy skrót określający dzień tygodnia (tylko dla języka angielskiego, `Mon` — poniedziałek, `Tue` — wtorek, `Wed` — środa, `Thu` — czwartek, `Fri` — piątek, `Sat` — sobota, `Sun` — niedziela),
- *dd* to dzień tygodnia,
- *mmm* to trzyliterowy skrót określający miesiąc (`Jan` — styczeń, `Feb` — luty, `Mar` — marzec, `Apr` — kwiecień, `May` — maj, `Jun` — czerwiec, `Jul` — lipiec, `Aug` — sierpień, `Sep` — wrzesień, `Oct` — październik, `Nov` — listopad, `Dec` — grudzień),
- *rrrr* to rok w formacie czterocyfrowym,
- *gg* to godzina,
- *ii* to minuta,
- *ss* to sekunda.

Czas musi być podany jako GMT (Greenwich Mean Time[28]). Przykład:

```
<meta http-equiv="expires" content="Sat, 06 Aug 2011 10:34:25 GMT">
```

Wskazówki dla robotów

Sieć bez ustanku jest przeczesywana przez roboty przeszukujące, indeksujące i gromadzące informacje. To dzięki nim mamy możliwość korzystania z tak popularnych wyszukiwarek. W znaczniku `<meta>` możemy zawrzeć wskazówki dla robotów (jedynie wskazówki, gdyż nie mamy żadnej gwarancji, że dany robot się do nich zastosuje). Znacznik ma wtedy postać:

```
<meta name="robots" content="wskazówka">
```

Ciąg *wskazówka* może przyjmować jedną z postaci przedstawionych w tabeli 1.9.

[27] Jest to format określony przez dokumenty RFC822/RFC1123. Dopuszczalne, ale niezalecane, jest używanie formatu zgodnego z RFC850.

[28] Czas na południku zerowym. Obowiązujący w Polsce czas środkowoeuropejski letni jest przesunięty o dwie godziny do przodu (GMT+2), a czas zimowy o jedną godzinę do przodu (GMT+1).

Tabela 1.9. *Wartości atrybutu content ze wskazówkami dla robotów sieciowych*

Wartość atrybutu	Znaczenie
index	Indeksuj stronę
noindex	Nie indeksuj strony
follow	Podążaj za odnośnikami na stronie
nofollow	Nie podążaj za odnośnikami na stronie (ignoruj podstrony)
all	Znaczenie takie samo jak index i follow łącznie
none	Znaczenie takie samo jak noindex i nofollow łącznie

Jeżeli chcemy zastosować dwa atrybuty na raz (czyli index połączone z follow lub nofollow bądź noindex połączone z follow lub nofollow), należy je oddzielić znakiem przecinka. Przykłady:

```
<meta name="robots" content="noindex">
<meta name="robots" content="index, nofollow">
<meta name="robots" content="all">
```

Pamięć podręczna

Strony pobrane z sieci przeglądarki przechowują w pamięci podręcznej (cache). Możemy jednak zasugerować przeglądarce, aby tego nie robiła, stosując znacznik <meta> w postaci:

```
<meta http-equiv="pragma" content="no-cache">
```

Podobnie jak w przypadku opisanych wyżej dyrektyw dla robotów, nie mamy gwarancji, że przeglądarka zastosuje się do tego polecenia, z reguły jest ono jednak respektowane.

Automatyczne odświeżanie

Możemy spowodować, aby strona WWW była automatycznie odświeżana. Może to być użyteczne w przypadku witryn prezentujących często aktualizowane dane. Znacznik <meta> powinien mieć wtedy postać:

```
<meta http-equiv="refresh" content="n">
```

gdzie *n* określa liczbę sekund, po których strona zostanie ponownie wczytana, np.:

```
<meta http-equiv="refresh" content="15">
```

Przekierowanie na inny adres

Znacznik <meta> może być również użyty do przekierowania odwiedzającego witrynę pod inny adres. Tego typu efekt jest najczęściej używany, gdy strona zmieniła lokalizację. Ponieważ przekierowanie może być wykonane z opóźnieniem, pod starym adresem można zamieścić stosowną informację. Osoba odwiedzająca będzie więc mogła przeczytać komunikat, po czym zostanie automatycznie wczytana nowa wersja witryny. Schematyczna postać znacznika jest następująca:

```
<meta http-equiv="refresh" content="n; url=http://nowy_adres">
```

gdzie *n* określa liczbę sekund, po których nastąpi przekierowanie, a *nowy_adres* — nowy adres witryny. Przykład:

```
<meta http-equiv="refresh" content="5; url=http://marcinlis.com">
```

Sekcja <body>

W sekcji <body> umieszczamy właściwą treść strony WWW. Sam znacznik <body> ma z reguły postać używaną w dotychczasowych przykładach i nie zawiera atrybutów[29]. Nie znaczy to jednak, że nie może ich zawierać. Można stosować opisane wyżej standardowe atrybuty: class, dir, id, lang, style, title. Warto zwrócić uwagę na zachowanie atrybutu title. Otóż użycie go przy znaczniku <body> spowoduje, że tak określona podpowiedź będzie wyświetlana przy każdym widocznym elemencie witryny, który nie ma zdefiniowanego atrybutu title. Spójrzmy na przykład z listingu 1.10.

Listing 1.10. *Podpowiedź w elemencie <body>*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html lang="pl">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Moja strona WWW</title>
  </head>
  <body title="To jest element witryny bez własnej podpowiedzi">
    <p>To jest akapit tekstowy.</p>
    <p title="Drugi akapit">To jest akapit tekstowy numer 2.</p>
  </body>
</html>
```

To typowa strona WWW z dwoma akapitami tekstowymi. Pierwszy akapit nie ma atrybutu title, a więc nie ma przypisanej własnej podpowiedzi, drugi natomiast taki atrybut ma. To oznacza, że po zatrzymaniu kursora myszy nad pierwszym akapitem pojawi się podpowiedź zdefiniowana w znaczniku <body> (można ją potraktować jako domyślną podpowiedź do każdego elementu witryny), a po zatrzymaniu kursora myszy nad drugim akapitem pojawi się wyłącznie podpowiedź dotycząca tego akapitu (podpowiedź z elementu <body> zostanie zignorowana).

Omawiając znacznik <body>, nie można, niestety, pominąć przestarzałych atrybutów, a to dlatego że wciąż spotyka się w sieci strony, które z nich korzystają. Warto więc wiedzieć, czemu one służą, ale też w żadnym wypadku nie należy ich stosować. Te atrybuty to:

- background — określa obraz stosowany jako tło.
- bgcolor — określa kolor tła.
- text — określa kolor tekstu.

[29] Z wyjątkiem atrybutów związanych ze zdarzeniami. Najczęściej spotka się tu zdarzenie onload. To jednak temat wykraczający poza ramy tematyczne książki. Stosowane bywają także atrybuty związane ze stylami CSS.

- `link` — określa kolor nieodwiedzonego odnośnika.
- `vlink` — określa kolor odwiedzonego odnośnika.
- `alink` — określa kolor aktywnego odnośnika.

Koniecznie trzeba jednak podkreślić raz jeszcze — wszystkie wymienione tu atrybuty są obecnie przestarzałe, a ich stosowanie jest błędem. Zmianę definiowanych przez nie parametrów należy wykonywać za pomocą stylów CSS (rozdział 4.).