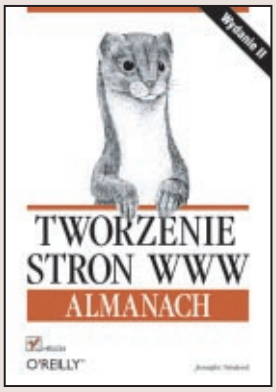
Tworzenie stron WWW. Almanach



Autor: Jennifer Niederst
Tłumaczenie: Mikołaj Barwicki,
Jacek Smycz, Monika Stępińska
ISBN: 83-7197-675-5
Tytuł oryginału: Web Design In A Nutshell
Format: B5, stron: 658

Niniejsza książka jest poświęcona głównie frontowym aspektom tworzenia stron WWW, takim jak pisanie HTML, tworzenie grafiki oraz multimedia. Nie jest ona źródłem wiedzy na temat programowania, skryptów ani funkcji serwera; starałam się jednak dostarczyć wystarczającej ilości informacji na temat tła, aby umożliwić projektantom poznanie w pewnym stopniu terminologii i technologii. Książka jest przeznaczona dla szerokiego kręgu odbiorców – od profesjonalistów, poszukujących szczegółowych informacji, do początkujących, którzy mogą oczekiwać wyczerpującego wyjaśnienia nowych koncepcji i konkretnych znaczników. Książka została podzielona na sześć części; w każdej z nich podano ogólny zakres tematów.

Część I, „Środowisko WWW"
W części I zostały wprowadzone niektóre ogólne koncepcje na temat pracy sieci WWW, które powinny umożliwić projektantom orientację we właściwościach tego środowiska. W końcowej partii znajduje się wprowadzenie do podstawowych koncepcji serwerów i Uniksa.

Część II, „Tworzenie"
W części II autorka skupiła się na omówieniu znaczników HTML i ich wykorzystaniu. Większość rozdziałów rozpoczyna się od wyliczenia znaczników z krótkimi opisami (w celu ułatwienia dostępu), po czym następują bardziej szczegółowe wyjaśnienia i praktyczne porady na temat ich stosowania.

Część III, „Grafika"
Rozdziały w części III zawierają dodatkowe informacje na temat formatów plików graficznych sieci WWW, a także opisy dostępnych narzędzi i praktyczne porady na temat tworzenia i optymalizacji grafiki.

Część IV, „Multimedia i interaktywność"
Rozdziały w części IV są poświęcone animacji, dźwiękowi i interaktywnym możliwościom sieci WWW.

Część V, „Zaawansowane technologie"
Część V zawiera przegląd podstawowych technologii pozwalających na implementację zaawansowanych własności na stronie WWW.

Dodatki
Dodatki zawierają wiele przydatnych tabeli znaczników HTML i elementów CSS.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

32

WAP i WML

W ciągu ostatnich lat technologie bezprzewodowe i Internet wtargnęły jak burza do powszechnego użytku, naturalne zatem wydaje się ich połączenie. Jedną z technik przesyłania informacji do urządzeń bezprzewodowych (głównie telefonów komórkowych, ale także pagerów i osobistych, elektronicznych organizerów) jest WAP, czyli protokół aplikacji bezprzewodowych (*Wireless Application Protocol*). WAP jest zestawem protokołów i specyfikacji umożliwiających telefonom komórkowym dostęp do informacji Interneto-podobnych.

Jednym z elementów standardu WAP jest język znaczników do zastosowań bezprzewodowych (*Wireless Markup Language*, WML), stosowany do budowy bezprzewodowych aplikacji na takiej samej zasadzie, jak HTML jest używany do tworzenia witryn internetowych. WML jest pochodną XML-a, co znaczy, że jest określony poprzez definicję typu dokumentu, czyli DTD (więcej informacji o XML-u znajduje się w rozdziale 30.).

Celem protokołu WAP nie jest przenoszenie istniejących witryn do telefonów komórkowych, ponieważ jest to niepraktyczne ze względu na mały rozmiar wyświetlaczy i ograniczone pasmo transmisji. Jest to raczej system do budowy konkretnych aplikacji, skrojonych do potrzeb urządzeń przenośnych. WAP nadaje się do przesyłania krótkich, zwięzłych porcji danych, takich jak kursy giełdowe, wyniki sportowe, programy kin itd. Nie nadaje się do złożonych dokumentów o skomplikowanej strukturze, jakie powszechne są dziś w sieci WWW.

Wielu webmasterów odczuwa presję związaną z rozwojem sieci w kierunku urządzeń bezprzewodowych. Wydaje się jednak, że osoba, która nie zajmuje się poważnie tworzeniem aplikacji bezprzewodowych, nigdy nie będzie musiała projektować pod kątem urządzeń przenośnych. Z drugiej strony oczywiście dobrze jest znać podstawy tego, co dzieje się w przestrzeni bezprzewodowej.

Rozdział ten otwiera krótkie wprowadzenie do protokołu WAP i projektowania aplikacji. Druga część skupia się na języku WML i sposobie jego działania, włączając w to opis elementów i atrybutów zawartych w obecnej specyfikacji WML-a.

Opis protokołu WAP

Jak już wspomniano, WAP nie jest niezależnym bytem, lecz raczej listą protokołów i specyfikacji. Jest opracowywany pod kierunkiem Wireless Application Protocol Forum, założonego przez gigantów przemysłu bezprzewodowego, jakimi są Nokia, Motorola, Ericsson oraz Unwired Planet (które przekształciło się w Phone.com, a obecnie nazywa się Openwave). Witryna WAP Forum (*http://www.wapforum.org*) jest dobrym źródłem informacji o aktualnym stanie technologii komunikacji bezprzewodowej, włączając w to pełną dokumentację specyfikacji WAP.

Środowisko urządzeń bezprzewodowych

WAP jest dostosowany do specyficznych ograniczeń i wymagań telefonów komórkowych i tym podobnych urządzeń. Środowisko to charakteryzuje się następującymi cechami:

- mały rozmiar wyświetlacza — można wyświetlić jednocześnie niewielką porcję informacji,
- ograniczona moc obliczeniowa,
- brak pełnej klawiatury, co utrudnia wprowadzanie informacji przez użytkownika; niektóre urządzenia przenośne posiadają pełną klawiaturę, ale są to wyjątki,
- niskie prędkości przesyłu — prędkości transmisji telefonów komórkowych oscylują obecnie wokół 9600 bitów na sekundę,
- kosztowny dostęp — użytkownicy z reguły płacą za każdą minutę lub przesłany kilobajt danych; istotne jest, aby dostęp do potrzebnych danych był szybki.

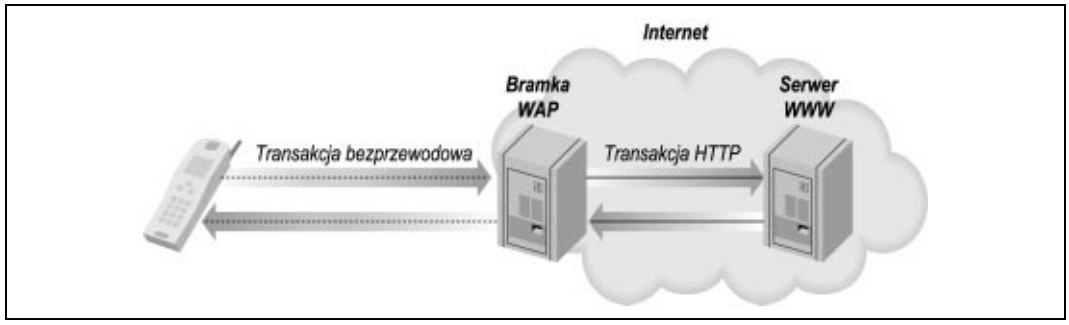
Wszystkie te czynniki wyznaczają kierunek rozwoju technologii WAP, jak i sposób projektowania aplikacji dla urządzeń bezprzewodowych.

Jak funkcjonuje WAP

Aplikacja WAP musi zostać sformatowana w językach WML i WMLScript, który jest językiem skryptowym służącym do wprowadzania do aplikacji elementów interaktywnych. Pliki te mogą być udostępniane przez zwykły serwer HTTP.

Korzystające z WAP-u urządzenia komunikują się serwerem za pośrednictwem *bramki WAP* (Rysunek 32.1). Bramka jest połączeniem między Internetem, a siecią komórkową, a jej zadaniem jest konwersja żądań protokołu WAP na żądania HTTP. Urządzenie bezprzewodowe (takie jak telefon komórkowy) zgłasza do bramki żądania za pośrednictwem fal radiowych. Bramka poprzez Internet oraz protokół HTTP zgłasza żądanie i pobiera dokument WML z serwera sieciowego.

Po dotarciu do bramki, a przed wysłaniem do telefonu, dokument zostaje *skompilowany* do postaci Binarnej WAP XML (WBXML). Proces kompilacji zmniejsza rozmiar pliku poprzez zastąpienie znaczników specjalnymi kodami znakowymi oraz poprzez usunięcie komentarzy i zbędnych pustych znaków. Po skompilowaniu dokument zostaje przesłany za pośrednictwem fal radiowych do telefonu komórkowego.



Rysunek 32.1. Urządzenia bezprzewodowe komunikują się z serwerem sieciowym za pośrednictwem bramki

Alternatywy wobec WAP

Podczas gdy WAP zdobywa światową popularność, nie jest jedynym systemem wymiany informacji z urządzeniami bezprzewodowymi. Zanim powstał WAP, strony były transmitowane za pośrednictwem HTTP oraz uwzględniającego ograniczenia urządzeń bezprzewodowych, języka znacznikowego HDML (opisanego w dalszej części rozdziału w ramce „O HDML-u"). Wiele telefonów wciąż obsługuje standardowe dokumenty HTML.

iMode (stworzony przez NTT DoCoMo, *http://www.nttdocomo.com*) jest kolejnym systemem komunikacji bezprzewodowej, który zdobył popularność w Japonii. Telefony iMode posiadają z reguły wielokolorowe ekrany, zdolne do wyświetlania obrazów czy nawet gier. Dokumenty iMode są tworzone z wykorzystaniem podzbioru języka HTML (pełna lista znajduje się na *http://www.nttdocomo.com/i/tag/index.html*).

Wiele osób postrzega obecną wersję protokołu WAP jako rozwiązanie przejściowe, które zostanie następnie zastąpione przez IPv6 po stronie protokołów i XML po stronie języka. Oczywiście, w świecie telekomunikacji, rozwiązania przejściowe wchłonięte przez rynek potrafią utrzymywać się zaskakująco długo, tak więc prawdopodobnie WAP w obecnej wersji przez jakiś czas jeszcze nie zniknie.

Bramki WAP są z reguły własnością operatorów komórkowych, tak więc nie ma potrzeby posiadania własnej bramki, by oferować dostęp do aplikacji WAP (wyjątkiem są tutaj banki i inne tego rodzaju instytucje). Dokumenty mogą być udostępniane za pośrednictwem zwykłego, odpowiednio skonfigurowanego serwera sieciowego. Bramka WAP odnajdzie żądane informacje tak, jak każdy inny klient sieciowy.

Publikacja dokumentów WAP

Aby udostępnić dokumenty konieczne do funkcjonowania aplikacji WAP, należy skonfigurować serwer tak, by rozpoznawał kilka nowych typów (plików) MIME, opisanych w tabeli 32.1. Pojęcie typów MIME zostało objaśnione w rozdziale 4. Instrukcja serwera zawiera informację, jak instalować nowe typy plików.

Tabela 32.1. Typy MIME związane z WAP

Opis	Typ MIME	Rozszerzenie

Plik WML	`text/vnd.wap.wml`	*.wml*
Skompilowany plik WML	`application/vnd.wap.wmlc`	*.wmlc*
Plik WMLScript	`text/vnd.wap.wmlscript`	*.wmls*
Skompilowany plik WMLScript	`application/wnd.wap.wmlscriptc`	*.wmlsc*
Mapa bitowa do wyświetlania w urządzeniach bezprzewodowych	`image/vnd.wap.wbmp`	*.wbmp*

Tworzenie aplikacji WAP

Przed zapoznaniem się ze szczegółami specyfikacji WAP, warto zwrócić uwagę na kilka aspektów tworzenia dokumentów WAP.

Nowy model

Osoba przyzwyczajona do tworzenia stron WWW będzie musiała dostosować (czyli zredukować) sposób myślenia do poziomu urządzeń przenośnych. Ze względu na ograniczenia wielkości ekranu, mocy obliczeniowej i prędkości transmisji zwykłe, bogate w informacje strony sieciowe są niepraktyczne. Warto chyba porzucić nawyk myślenia w kategoriach „dokumentów" (informacji wyświetlanej, którą czyta użytkownik) i zacząć się posługiwać kategorią „aplikacji" (opartą na wyborze i interakcji z użytkownikiem). Aplikacje WAP z reguły składają się z ekranów o minimalnej zawartości oraz listy opcji. Informacja i interakcja ma pierwszorzędne znaczenie, jako że w zasadzie nie ma miejsca na wyświetlanie grafiki.

Kolejną różnicą jest fakt, że w przeciwieństwie do sieci, gdzie pożądane jest, by użytkownik zatrzymał się na witrynie jak najdłużej, jakość aplikacji WAP może być mierzona tym, jak szybko użytkownik jest w stanie odszukać informację bądź dokonać transakcji i opuścić witrynę.

Przeglądarki WAP

Urządzenia przenośne do pobierania i wyświetlania informacji z sieci wykorzystują specjalne oprogramowanie, określane czasem nazwą „mikroprzeglądarek", ze względu na ich rozmiar i pojemność. Tak samo jak w przypadku WWW nie wszystkie przeglądarki WAP mają takie same możliwości. Wciąż eksploatowane są starsze urządzenia, wyposażone w uboższe wersje przeglądarek, a aplikacja, która funkcjonuje prawidłowo na jednym urządzeniu, może nie działać na innym.

Należy też zwrócić uwagę na fakt, że specyfikacja WML zezwala na pewną dowolność w interpretacji elementów i funkcji, tak więc nawet przeglądarki w 100% zgodne z WML-em mogą wykazywać pewne różnice w implementacji standardu.

Większość obsługujących WAP urządzeń jest wyposażonych w Openwave Mobile Browser (dawniej zwany UP.Browser), opracowany przez Openwave (które kiedyś nazywało się Unwired Planet, stąd „UP"). Lista urządzeń korzystających z przeglądarek Openwave znajduje się na stronie Openwave Mobile Browser Phone Reference pod adresem *http://developer.phone.com/resources/phones.html*.

Nie było zaskoczeniem, że i Microsoft zainteresował się światem urządzeń bezprzewodowych i zaprezentował przeglądarkę Microsoft Mobile Explorer dla telefonów komórkowych. MME jest prze-

glądarką obsługującą dokumenty zarówno WML, jak i HTML. Lista urządzeń wyposażonych w MME wciąż się wydłuża. Więcej informacji o MME znajduje się na stronie: *http://www.microsoft.com/mobile/phones/mme/default.asp*.

Urządzenia przenośne

Na rynku są setki modeli urządzeń przenośnych. Niestety nie istnieje standard konfiguracji sprzętowej, więc trudno przewidzieć, jak konkretna aplikacja będzie działała u docelowego użytkownika. Oto kilka uwarunkowań sprzętowych, z którymi zmagają się twórcy aplikacji WAP:

Wielkość ekranu

Urządzenia różnią się między sobą wielkością wyświetlaczy. O znaczeniu tego zjawiska niech świadczy fakt, że większość wyświetlaczy w telefonach ma rozdzielczość od 95 do 120 pikseli w poziomie i 50 do 90 pikseli w pionie. Nowsze telefony i notesy elektroniczne mogą posiadać większe wyświetlacze (około 300×100 pikseli).

Rozdzielczość ekranu jest trudna do określenia z poziomu aplikacji. Co więcej, np. w modelu Nokia 7110 piksele mają wysokość większą niż szerokość (w stosunku 1,25:1), co może powodować rozciągnięcie elementów graficznych.

Tekst

Ponieważ dokumenty WAP są oparte przede wszystkim na tekście, istotniejsze może być określanie wielkości ekranu jako liczby wyświetlanych znaków. Najczęściej przenośne przeglądarki wyświetlają jednocześnie jedynie 3 do 6 linijek tekstu, po 12 do 20 znaków każda.

Tekst może być wyświetlany w sposób jednorodny, tak że wszystkie znaki są tej samej szerokości, lub w sposób proporcjonalny — wtedy znaki mają różne szerokości. Może to powodować trudności w określeniu liczby znaków przypadających na linię tekstu.

Paleta kolorów

Znaczna większość urządzeń przenośnych posiada 1-bitowe, czarno-białe wyświetlacze LCD. Wraz ze wzrostem mocy obliczeniowej w ciągu najbliższych lat należy się spodziewać rozpowszechnienia wyświetlaczy ze skalą szarości oraz 8-bitowym kolorem. Urządzenia z takimi wyświetlaczami są już popularne w Japonii.

Klawisze programowe

Urządzenia przenośne są przeważnie wyposażone w *klawisze programowe*, czyli takie, których funkcje mogą zostać oprogramowane przez aplikację. Liczba klawiszy, sposób rozmieszczenia oraz sposób ich oprogramowywania zależą od urządzenia. Czasami klawisze programowe są wyświetlane graficznie na ekranie, co powoduje, że trudno przewidzieć sposób, w jaki użytkownik będzie sterował aplikacją.

Ograniczenia rozmiaru

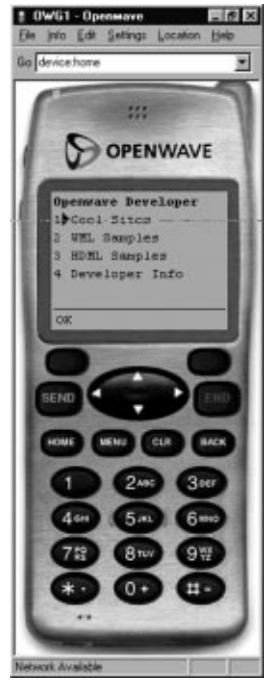
Rozmiar każdego dokumentu *.wml* w aplikacji WAP jest ograniczony do 1400 bajtów, choć większość twórców zmniejsza rozmiar pliku do 500 bajtów, co zwiększa wydajność. Ograniczenie rozmiaru odnosi się do dokumentów *skompilowanych*. Robocze dokumenty mogą być nieco większe, należy więc zwracać uwagę na rozmiar skompilowanego pliku, wskazywany przez emulator bądź inne narzędzie projektowe.

Jeśli rozmiar dokumentu przekracza 1400, musi on zostać podzielony na logicznie oddzielne, mniejsze pliki.

Emulatory WAP

Do testowania swoich dokumentów twórcy aplikacji WAP z reguły korzystają z emulatorów WAP. Emulator (bądź „symulator") jest programem dla zwykłego komputera, który pozwala na testowanie wyglądu i działania aplikacji. Eliminuje to konieczność zakupu kilku telefonów komórkowych do testowania dokumentów.

Zaletą emulatorów jest to, że wyglądają one jak prawdziwe urządzenia (rysunek 32.2) — można nawet sterować aplikacją naciskając przyciski tak, jak w rzeczywistym telefonie. Niestety, emulatory nie zawsze tak *działają*. Można oczekiwać pewnych niezgodności w sposobie wyświetlania tekstu, a więc, aby uniknąć niespodzianek, przed opublikowaniem aplikacji należy ją przetestować na rzeczywistym urządzeniu.



Rysunek 32.2. Okno emulatora OpenWave Simulator

Poniżej wymieniono kilka popularnych emulatorów WML wraz z ich adresami WWW. Openwave Simulator jest najbardziej popularny ze względu na rozpowszechnienie przeglądarki Openwave Mobile Browser oraz jej poprzednika, UP.Browser. Wszystkie te programy występują jedynie w wersjach dla Windows.

Openwave Simulator (dawniej UP.Simulator)

http://developer.phone.com/download/

Microsoft Mobile Explorer Emulator

http://www.microsoft.com/mobile/phones/mme/default.asp

Nokia Toolkit

http://forum.nokia.com

Ericsson WapIDE SDK

http://www.ericsson.com/developerszone/

Motorola Mobile ADK

http://developers.motorola.com/developers/wap/index.html

Przeglądarka Opera w wersji 5.0 zawiera eksperymentalną obsługę WAP-u i WML-a. Podczas gdy nie jest to pełny emulator, może on być przydatny w trakcie testowania kodu WML czy po prostu przy wyświetlania aplikacji WAP. Więcej informacji znajduje się na stronie przeglądarki Opera pod adresem *http://www.opera.com*.

Oprócz programów emulacyjnych, do przeglądania stron WML można także użyć emulatorów sieciowych, takich jak Wapemulator.com (*http://wapemulator.com*) czy Gelon.net (*http://www.gelon.net*). Emulatory te mają ograniczoną funkcjonalność i dokładność, ale dają dobry ogólny pogląd i pozwalają oglądać strony WAP z komputera osobistego.

O HDML-u

Zanim WAP Forum opublikowało specyfikacje WAP-u i WML-u, urządzenia przenośne posługiwały się dokumentami w formacie HDML (Handheld Device Markup Language). WML, jako oparty na XML-u i o większych możliwościach, oficjalnie zastąpił HDML, ale ten ostatni będzie funkcjonował tak długo, jak występować będą obsługujące go starsze urządzenia. HDML składa się z podzbioru HTML-a oraz kilku dodatkowych znaczników, przeznaczonych do nawigacji za pośrednictwem urządzeń przenośnych.

Specyfikacja HDML-a zdążyła zaledwie uzyskać status „złożonej do zatwierdzenia" przez W3C, po czym została prześcignięta przez nowy i ulepszony WML. Proponowana specyfikacja HDML-a jest dostępna pod adresem: *http://www.w3.org/TR/NOTE-Submission-HDML-spec.html*. Aby udostępniać pliki HDML, serwer musi obsługiwać typ MIME określany jako `text/x-hdml` z rozszerzeniem *.hdml*.

Omówienie HDML-a można znaleźć w artykule Webmonkey pt. „Intro to HDML" pod adresem: *http://hotwired.lycos.com/webmonkey/99/48/index3a.html*.

Możliwe jest uaktualnienie istniejących aplikacji w HDML-u do WAP/WML. Dokładna dokumentacja na ten temat dostępna jest pod adresem: *http://developer.phone.com/technotes/hdml2wml/index.html*

Wprowadzenie do WML-a

Pora wreszcie przejść do sedna aplikacji bezprzewodowych — języka WML. Niniejsza część przybliża działanie WML-a, ale bynajmniej nie wyczerpuje tematu. Więcej informacji można

znaleźć w dokumentacji wymienionej na końcu niniejszego rozdziału. Pełny zestaw elementów i atrybutów WML-a zostanie omówiony w następnej części.

Struktura dokumentu

Specyfika środowiska urządzeń przenośnych powoduje, że traci sens znane z HTML-a podejście projektowe oparte na pojęciu „strony". Do aplikacji WML-owych bardziej stosują się pojęcia związane z „kartami". Aplikacja składa się z jednej bądź więcej *talii* (dokumentów *.wml*), z których każda zawiera pewną liczbę *kart* (elementów WML-a składających się na dokument). Karta zawiera ograniczoną ilość informacji, równoważną zaledwie kilku ekranom, podczas gdy ekran zawiera jedynie trzy do sześciu linii tekstu.

Ponieważ WML jest pochodną XML-a (rozdział 30.), dokumenty WML muszą być zarówno *poprawne* (czyli zgodne z DTD związanym z językiem WML), jak i *poprawnie sformułowane* (odpowiadające ścisłym regułom składni znaczników XML-a).

Poniżej znajduje się bardzo prosty dokument WML, nazwany *jenskitchen.wml*. Stanowi on „talię" złożoną z dwóch „kart".

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
                     "http://www.wapforum.org/DTD/wml_1.2.xml">
<wml>
   <card id="intro">
      <p>Witamy w kąciku kucharskim Jen</p>
   </card>

   <card id="book1">
      <p>101 przepisów z dzikimi grzybkami</p>
      <p>Jennifer Niederst</p>
      <p>Cena: 19.95PLN</p>
   </card>
</wml>
```

Przykład ten rozpoczyna się standardowymi deklaracjami `XML` i `DOCTYPE` (więcej informacji o prologu dokumentu WML znajduje się w rozdziale 30.). Element `<wml>` definiuje talię WML. Każdy dokument WML rozpoczyna się i kończy znacznikami `<wml>` i mogą one wystąpić w dokumencie tylko raz.

Talia może składać się z jednej lub wielu kart. Powyższy przykład zawiera dwie karty, określone przez element typu `<card>`. Karty mogą zawierać różne elementy, ale ich zawartość musi być ograniczona znacznikami akapitów (`<p>`). Ponieważ XML jest pochodną XML, każdy akapit musi posiadać znacznik zamykający `</p>` (w odróżnieniu od HTML-a, gdzie akapit mógł pozostać nie zamknięty). Warto zwrócić uwagę, że każda karta otrzymała nazwę poprzez atrybut `id`, dzięki czemu można się do niej odwoływać w dalszej części dokumentu.

Formatowanie tekstu

Zgodnie ze specyfikacją WML-a, tekst (a właściwie każdy element) musi być zawarty w akapitach (znacznikach `<p>`). Element `<p>` posiada dwa atrybuty: `align` oraz `mode`. `align` działa tak samo, jak wyrównanie tekstu w HTML-u. Atrybut `mode` może mieć wartość `wrap` („przenoś

do nowej linii") lub `nowrap` („nie przenoś do nowej linii"). Domyślnie tekst jest przenoszony do nowej linii, ale można to wyłączyć, by akapit zawierał się w jednej linii. Należy jednak pamiętać, że podczas gdy pewne urządzenia przewijają obraz w poziomie, inne nie mają takiej możliwości, tak więc tekst o ustawieniu `nowrap` może być niedostępny dla części użytkowników. Zarówno parametr `align`, jak i `mode` mogą być ignorowane przez niektóre przeglądarki, dlatego należy zwrócić uwagę, by dokument działał także w takim przypadku.

Znaki nowej linii można dodać do tekstu poprzez znacznik `<br/>`. Ponieważ WML korzysta z reguł składni XML-a, konieczne jest dopisanie na końcu znacznika znaku ukośnika, tak aby domknąć element.

Specyfikacja wymienia również następujące elementy służące do bezpośredniego formatowania tekstu:

`<b>`	Pogrubienie
`<big>`	Nieco większy niż otaczający tekst
`<em>`	Wyszczególnienie (pogrubienie i kursywa)
`<i>`	Kursywa
`<small>`	Nieco mniejszy niż otaczający tekst
`<strong>`	Tekst „wzmocniony" (pogrubiony, kursywa lub pogrubiona kursywa)
`<u>`	Podkreślenie

Niestety, nie ma gwarancji, że tekst będzie wyświetlany zgodnie z formatem określonym przez znaczniki. Niektóre urządzenia w zupełności ignorują formatowanie, więc znaczenie tekstu nie powinno być zdeterminowane przez styl.

Dodawanie łączy

Tak jak w sieci WWW, łącza do innych stron są integralną częścią aplikacji WAP. WML oferuje znajomy znacznik `<a>`, pozwalający na tworzenie łączy. W poniższym przykładzie w jednej z kart występuje łącze wskazujące inną kartę poprzez jej nazwę, wpisaną w znaczniku łącza. W podobny sposób tworzone są łącza w HTML-u. W momencie gdy użytkownik wybiera tekst łącza, do okna przeglądarki ładowana jest druga karta.

```
<wml>
   <card id="intro">
      <p>Witamy w kąciku kucharskim Jen<br/>
      <a href="#book1">Wypróbuj naszą książkę!</a>
      </p>
   </card>

   <card id="book1">
      <p>101 przepisów z dzikimi grzybkami</p>
      <p>Jennifer Niederst</p>
      <p>Cena: 19.95PLN</p>
   </card>
</wml>
```

Element `<a>` pozwala jedynie na tworzenie łączy do innych kart bądź talii. Specyfikacja WML oferuje bardziej uniwersalne narzędzie służące do nawigowania pomiędzy kartami: element `<anchor>`. Element ten może służyć jako łącze do konkretnej karty, której adres jest nieznany w momencie tworzenia dokumentu, jak na przykład poprzednia karta bądź karta o adresie poda-

nym przez użytkownika (poprzez zmienną). Znacznik <anchor> służy jako kontener dla dwóch innych elementów WML-a, <go> i <prev>, którym zawdzięcza swoją funkcjonalność.

Poniższy przykład wykorzystuje znacznik <anchor> z elementem <go> do zbudowania prostego łącza (działanie jest identyczne jak w poprzednim przykładzie).

```
<anchor>
   Wypróbuj naszą książkę!
   <go href="#book1" />
</anchor>
```

Element <anchor> może być również wykorzystany do utworzenia własnego przycisku typu „Poprzednia strona" poprzez wykorzystanie elementu <prev>, tak jak to zaprezentowano w poniższym przykładzie:

```
<anchor>
   Powrót!
   <prev/>
</anchor>
```

Obrazy

Mimo że aplikacje WAP składają się w głównej mierze z tekstu, możliwe jest dodanie do karty prostych obrazów (rysunek 32.3). Aby obraz mógł być wyświetlony w aplikacji WAP, musi być zapisany w specjalnie zoptymalizowanym formacie Wireless Bitmap (WBMP). Pliki WBMP są 1-bitowymi obrazami graficznymi, składającymi się jedynie z czarnych i białych pikseli. Zaleca się, by obrazy były możliwie jak najmniejsze. Powierzchnia grafiki nie powinna przekraczać 150 pikseli kwadratowych. Należy pamiętać, że niektóre mikroprzeglądarki w ogóle nie obsługują grafiki, dlatego zawsze należy dołączyć opis tekstowy.

Rysunek 32.3. Przykłady grafiki WBMP w aplikacji WAP

Obrazy dodawane są do dokumentu za pośrednictwem elementu <img/>. Należy się upewnić, że znajduje się on w znaczniku <p>, tak jak w poniższym przykładzie:

```
<card>
<p><img src="logo.wbmp" alt="Logo książki kucharskiej" /></p>
</card>
```

Niektóre urządzenia przenośne mają zapisaną w pamięci bibliotekę niewielkich obrazów, które można włączyć do dokumentu WML za pośrednictwem atrybutu localsrc, umieszczonego w znaczniku obrazu. Zaletą lokalnych obrazków jest to, że redukują one ilość danych koniecznych

do pobrania z serwera, a zatem wyświetlane są szybciej niż zewnętrzne pliki WBMP. Warto również zapewnić wskaźnik do zewnętrznej grafiki, na wypadek gdyby lokalne obrazki nie były dostępne. Następujący przykład ilustruje, jak należy wyświetlić wbudowaną ikonę karty kredytowej z lokalnej biblioteki obrazków, oraz podaje alternatywny plik *.wbmp*. Tekst atrybutu `alt` będzie wyświetlany w urządzeniach nie obsługujących grafiki.

```
<img localsrc="creditcard" src="card.wbmp" alt="credit card symbol" />
```

Pełna lista obrazów bibliotecznych, wraz z ich nazwami, znajduje się pod adresem: *http://developer.phone.com/htmldoc/41/wmlref/taglist.html#575099*.

Tworzenie grafiki WBMP

Obecnie dostępnych jest niewiele narzędzi służących do tworzenia plików WBMP. Jednakże RCP Distributed Systems rozpowszechnia darmową wtyczkę UnWired, która umożliwia tworzenie WBMP w programach Adobe Photoshop 5 i wyższych oraz w JASC Paint Shop Pro (lub dowolnym innym pakiecie graficznym obsługującym wtyczki). Wtyczka dostępna jest pod adresem: *http://www.rcp.co.uk/distributed/downloads/*.

Istnieje również napisane w Javie narzędzie o nazwie *pic_2_wbmp*, które przekształca istniejące pliki BMP na format WBMP. Program dostępny jest pod adresem: *http://www.gingco.de/wap/*.

Zapis grafiki w formacie WBMP umożliwia również produkt firmy Macromedia – Fireworks 4 i jego następca – Fireworks MX.

Tabele

WML oferuje te same podstawowe elementy do budowy tabel co HTML. Sama tabela jest definiowana poprzez znaczniki `<table>`. Tabela zawiera pewną liczbę rzędów (`<tr>`), a każdy rząd składa się z pewnej liczby komórek z danymi (`<td>`), zawierających treść tabeli. W przeciwieństwie do tabel w HTML-u, w języku WML można jawnie określić liczbę kolumn w tabeli poprzez atrybut `columns` w znaczniku `<table>`. Wyrównanie tekstu jest ustawiane dla każdej kolumny na poziomie tabeli (szczegóły składni opisane są w zestawieniu elementów w dalszej części rozdziału).

W poniższym przykładzie utworzono tabelę o trzech kolumnach. Zawartość pierwszej kolumny wyrównywana jest do lewej, zaś treść pozostałych dwóch kolumn — do prawej strony. Rezultat pokazano na rysunku 32.4.

Rysunek 32.4. Tabela WML w emulatorze

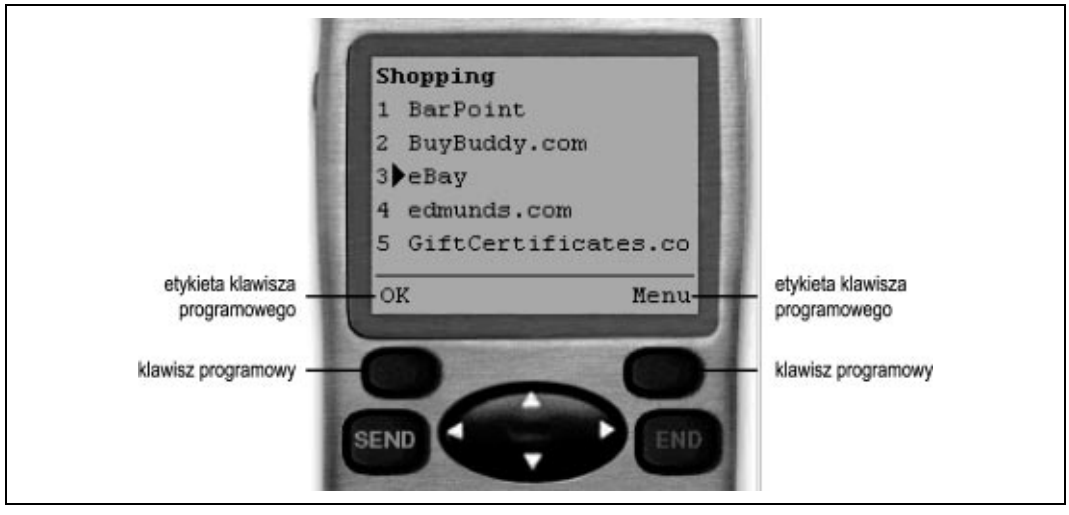
```
<table columns="3" align="LRR">
<tr>
   <td>Month</td>
```

```
      <td>Min</td>
      <td>Max</td>
   </tr>
   <tr>
      <td>Feb</td>
      <td>4</td>
      <td>41</td>
   </tr>
   <tr>
      <td>Mar</td>
      <td>25</td>
      <td>62</td>
   </tr>
   </table>
```

Należy zachować ostrożność w stosowaniu tabel w aplikacjach WML. Tabele bardzo łatwo rozrastają się do rozmiarów przekraczających szerokość ekranu, co powoduje, że użytkownicy nie mający dostępu do przewijania poziomego tracą część informacji. Niektóre urządzenia mogą w ogóle ignorować tabele, po prostu wyświetlając zawartość komórek w liście, w kolejności, w jakiej występują w pliku źródłowym. W przypadku stosowania tabel należy do minimum ograniczyć liczbę kolumn i wielkość komórek tabeli, upewnić się, że sposób prezentacji ma sens, oraz przetestować projekt na kilku różnych urządzeniach.

Definiowanie klawiszy programowych

Telefony komórkowe i inne urządzenia przenośne są z reguły wyposażone w klawisze programowe, przyciski, których funkcja może zostać zaprogramowana w dowolny sposób (rysunek 32.5). Klawisze programowe są najczęściej przyciskami na obudowie urządzenia, ale mogą być też wyświetlane na ekranie. Ponieważ metody implementacji klawiszy programowych są różne w rozmaitych urządzeniach, projektowana aplikacja może się prezentować na wiele sposobów.



Rysunek 32.5. Przykłady klawiszy programowych w Openwave Simulator

Dobrym miejscem na zastosowanie klawiszy programowych są funkcje i łącza niekoniecznie należące do treści wyświetlanego dokumentu, takie jak przycisk „Powrót” lub łącze do głównego me-

nu aplikacji. Funkcje są przyporządkowywane klawiszom za pośrednictwem elementu <do>. Element ten posiada jeden wymagany atrybut, type, który określa sposób działania klawisza. Atrybut type może przybrać jedną z siedmiu wartości:

accept	OK lub potwierdzenie informacji
prev	Cofnij się do poprzednio wyświetlanej karty (tak jak przycisk „Wróć")
help	Żądanie pomocy
reset	Wyczyść stan (zmiennych) karty bądź talii
options	Wybór z listy opcji
delete	Usuń element
unknown lub "" (*pusty*)	Inne działanie

Element <do> wykorzystuje atrybut label do określenia tekstu przypisanego przyciskowi. Zaleca się, aby długość etykiet była ograniczona do około sześciu znaków, co zwiększa prawdopodobieństwo prawidłowego działania aplikacji w różnych urządzeniach.

Definiowanie klawiszy programowych jest skomplikowanym zagadnieniem, wykraczającym poza ramy niniejszego opracowania specyfikacji WML, dlatego przytoczone przykłady ograniczone są do najprostszych. W pierwszym przykładzie element <do> służy do utworzenia klawisza programowego „Wróć". Sposób przyporządkowania tego zadania do klawisza pozostawiono konkretnemu urządzeniu, tak więc jest to poza kontrolą projektanta aplikacji. Uwagę zwraca fakt, że ponieważ zadania nie należą do treści dokumentu, nie muszą być ograniczone znacznikami akapitu.

```
<do type="prev" label="Wróć">
   <prev/>
</do>
```

Innym znaczeniem często nadawanym klawiszowi programowemu jest łącze do innego dokumentu bądź karty, jak w poniższym przykładzie.

```
<do type="accept" label="Lista">
   <go href="list.wml"/>
</do>
```

Aby dodać do danej karty kilka zadań, korzysta się z typu zadania options. Opcje mogą być wyświetlane w przeglądarce jako menu bądź jako odnośnik do osobnej strony zawierającej listę odnośników. Sposób implementacji zależy od urządzenia użytkownika. W poniższym przykładzie dodano trzy funkcje: łącze do strony wyszukiwarki, łącze do listy oraz przycisk „Wróć".

```
<do type="options" label="Szukaj">
   <go href="search.wml"/>
</do>
<do type="options" label="Lista">
   <go href="list.wml"/>
</do>
<do type="options" label="Wróć">
    <prev/>
</do>
```

Interaktywność

Interaktywność jest nieodzownym atrybutem każdej aplikacji. Specyfikacja WML zawiera kilka elementów służących do pobierania danych od użytkownika oraz dynamicznego generowania treści w oparciu o te dane. Tak jak HTML, WML zawiera podstawowe elementy formularzy: `<input>` służący do umieszczenia w dokumencie pola tekstowego, `<select>` do definiowania listy elementów `<option>` oraz `<fieldset>` do grupowania zawartości formularza w logiczne segmenty.

Element `<setvar>` służy do przypisywania wartości *zmiennej*, tymczasowo przechowującej porcję informacji, takich jak dane od użytkownika, URL czy dowolna informacja tekstowa. W poniższym przykładzie wartość zmiennych zawierających informację o książce (skrót tytułu oraz cenę) jest ustawiana w momencie wyboru przez użytkownika łącza „Zakup książkę". Informacja jest zapisywana i wykorzystana dalej w aplikacji, na przykład do budowania listy wybranych pozycji.

```
<card id="book1" title="101 Grzybków">
<p><b>101 przepisów z dzikimi grzybkami</b></p>
<p>Autor: Jennifer Niederst<br/>
   Cena: 19.95PLN<br/>
   Ta książka uratuje Twoją imprezę.
   <anchor>
      Zakup książkę!
      <go href="purchase.wml">
         <setvar name="B" value="101Grzybków"/>
         <setvar name="P" value="19.95"/>
      </go>
   </anchor>
</p>
</card>
```

WMLScript jest językiem skryptowym strony klienta, który nadaje aplikacjom WML prawdziwą funkcjonalność. Omówienie języka WMLScript wykracza poza ramy rozdziału, ale każdy, kto poważnie myśli o tworzeniu aplikacji bezprzewodowych, powinien zapoznać się z tym językiem. Niestety, jest on obsługiwany tylko przez najnowsze przeglądarki WAP (wersje 4 i wyższe), ale bez wątpienia w ciągu najbliższych lat WMLScript stanie się istotnym narzędziem dla twórców aplikacji WAP. Opracowania dotyczące języka WMLScript zostaną wymienione na końcu rozdziału.

Elementy i atrybuty WML

W niniejszej części rozdziału znajduje się krótkie zestawienie elementów i atrybutów zawartych w specyfikacji WML 1.2 (obecnie najnowszej). Ponieważ jest to nowy standard, nie wszystkie przeglądarki obsługują go w całości, należy więc dokładnie testować projektowane aplikacje.

Istnieje kilka podstawowych atrybutów, które mogą być zastosowane niemal w każdym elemencie.

`xml:lang`

Określa język elementu.

`id`

Nadaje elementowi nazwę, pod którą będzie dostępny później.

`class`

Określa nazwę klasy elementu (dzięki czemu elementy mogą być grupowane).

<a>

```
<a>...</a>
```

Definiuje łącze do innego zasobu (konkretnej karty lub innego dokumentu *.wml*).

Atrybuty

`href=`*`url`*

Wymagany. Adres zasobu.

`title=`*`tekst`*

Krótki opis łącza.

`accesskey=`*`klawisz`*

Przypisuje łącze do klawisza.

<access>

```
<access/>
```

Określa nazwę domeny. Tylko dokumenty (karty) z tej domeny mają dostęp do aktualnego dokumentu. Element `<access>` musi być zawarty wewnątrz znaczników `<head>`.

Atrybuty

`domain=`*`nazwa domeny`*

Domena, która ma dostęp do karty.

`path=`*`ścieżka dostępu`*

Określa ścieżkę dostępu w obrębie domeny.

<anchor>

```
<anchor>...</anchor>
```

Określa wewnętrzne łącze do innego zasobu. Może być użyty w połączeniu z elementami `<go>` i `<prev>`.

Atrybuty

`title=`*`tekst`*

Krótki opis łącza.

`accesskey=`*`klawisz`*

Przypisuje łącze do klawisza.

<b>

```
<b>...</b>
```

Pogrubienie tekstu.

<big>

```
<big>...</big>
```

Oznacza tekst nieco większy niż standardowy.


```
<br/>
```

Oznacza znak nowej linii.

<card>

```
<card>...</card>
```

Jednostka dokumentu WML wyświetlana na ekranie urządzenia. Karty są logicznymi segmentami funkcjonalności aplikacji.

Atrybuty

`title=`*`tekst`*

Krótki opis tekstowy karty. Może, ale nie musi być wyświetlany przez urządzenie.

`newcontext=true|false`

W przypadku wartości `true` atrybut powoduje ponowną inicjalizację stanu przeglądarki, kasując historię nawigacji i wszystkie przechowywane zmienne.

`ordered=true|false`

Określa sposób organizacji treści karty. Wartość `true` oznacza, że karta jest częścią grupy elementów przetwarzanych sekwencyjnie, natomiast `false` oznacza brak kolejności.

<do>

```
<do>...</do>
```

Użyty w karcie, definiuje zadanie przeważnie przypisywane do klawisza programowego urządzenia. Zadanie jest określane przez elementy `<go>`, `<prev>`, `<noop>` lub `<refresh>`, zawarte wewnątrz elementu `<do>`.

Atrybuty

`type=accept|prev|help|reset|options|delete|unknown` (*lub* `""`)

Wymagany. Określa przeznaczenie elementu `<do>`, dzięki czemu przeglądarka może go zinterpretować.

`label`

Określa tekstową etykietę przycisku bądź funkcji.

`optional=true|false`

Wartość `true` oznacza, że przeglądarka może zignorować element. Domyślną wartością jest `false` (element nie może zostać zignorowany).

`name`

Nadaje elementowi nazwę.


```
<em>...</em>
```

Oznacza tekst wyszczególniony (sposób wyświetlania zależy od urządzenia).

<fieldset>

```
<fieldset>...</fieldset>
```

Definiuje logiczne segmenty w dokumencie. Dotychczas znacznik ten nie jest zbyt dobrze obsługiwany.

Atrybuty

`title=tekst`

Krótki opis tekstowy segmentu.

<go>

```
<go/>
```

Oznacza zadanie, które służy do nawigacji do określonego pliku (tak jak łącze). Atrybut `xml:lang` jest niedozwolony w tym elemencie.

Atrybuty

`href=`*`url`*

Wymagany. Określa docelowy adres URL.

`sendreferer=true|false`

Wartością domyślną jest `false`. Przy wartości `true`, przeglądarka musi podać położenie talii zawierającej to zadanie przy wysyłaniu żądania do serwera.

`method=get|post`

Metoda zgłoszenia żądania (danych formularza) HTTP. Musi być jedną z wartości `get` i `post`. Wartością domyślną jest `get`.

`enctype=`*`kodowanie`*

Rodzaj zawartości formularza. Wartością domyślną jest `application/x-www-form-urlencoded`.

`accept-charset=`*`lista zestawów znaków`*

Lista stron kodowych koniecznych do przetworzenia danych formularza.

<head>

```
<head>...</head>
```

Określa opcjonalny nagłówek dokumentu.

<i>

```
<i>...</i>
```

Oznacza kursywę.


```
<img/>
```

Umieszcza w tekście rysunek.

Atrybuty

`src=`*`url`*

Wymagany. Położenie wyświetlanego obrazu.

`alt=`*`tekst`*

Określa zastępczy tekst na wypadek, gdy obraz jest niedostępny.

`localsrc=`*`nazwa lokalnego obrazu`*

Oznacza predefiniowaną ikonę z wbudowanej biblioteki obrazów.

`align=top|middle|bottom`

Określa sposób wyświetlania obrazu w odniesieniu do linii bazowej otaczającego go tekstu.

`vspace=`*`liczba`*

Wstawia określoną liczbę pustych pikseli nad i pod obrazem.

`hspace=`*`liczba`*

Wstawia określoną liczbę pustych pikseli po lewej i prawej stronie obrazu.

`height=`*`liczba`*

Określa wysokość obrazu w pikselach.

`width=`*`liczba`*

Określa szerokość obrazu w pikselach.

<input>

```
<input/>
```

Dodaje do dokumentu pole tekstowe.

Atrybuty

`name=`*`tekst`*

Wymagane. Nazwa zmiennej, w której przechowywany będzie wprowadzony tekst.

`type=text|password`

Określa rodzaj wprowadzanego tekstu. Domyślna wartość `text` oznacza normalne wprowadzanie tekstu, natomiast `password` ukrywa wpisywane litery, wyświetlając gwiazdki lub kropki.

`value=`*`wartość`*

Określa domyślną wartość, wyświetlaną po załadowaniu elementu.

`format=`*`format`*

Określa format wprowadzanego tekstu, co ma na celu ograniczenie do określonych schematów alfanumerycznych (takich jak data lub numer karty kredytowej).

`emptyok=true|false`

Określa, czy pole może pozostać puste.

`size=`*`liczba`*

Szerokość pola tekstowego (w znakach).

`maxlength=`*`liczba`*

Maksymalna długość wprowadzanego tekstu (w znakach).

`tabindex=`*`liczba`*

Określa kolejność w jakiej przełączane są aktywne elementy formularza.

`title=`*`tekst`*

Określa opis ekranu tekstowego. Opis ten może, ale nie musi być wyświetlony przez przeglądarkę.

`accesskey=`*`klawisz`*

Przypisuje pole tekstowe do klawisza.

<meta>

```
<meta/>
```

Zawiera informację o dokumencie. Musi być umieszczony wewnątrz elementu `<head>`.

Atrybuty

`http-equiv=`*`nazwa`*

Określa nazwę nagłówka HTTP.

`name=`*`nazwa`*

Określa nazwę dla metadanych.

`forua=true|false`

Określa, czy metadane są przeznaczone dla użytkownika (przeglądarki). W przypadku wartości `false` metadane są usuwane z dokumentu przed wysłaniem do przeglądarki.

`content=`*`tekst`*

Wymagany. Określa treść metadanych.

`scheme=`*`tekst`*

Wykorzystywany do interpretacji wartości danych.

<noop>

```
<noop/>
```

Oznacza, że nic nie należy robić. Może zablokować przeglądarkę.

<onevent>

```
<onevent/>
```

Wykonuje zadanie uruchamiane określonym zdarzeniem.

Atrybuty

`type=onenterbackward|onenterforward|ontimer`

Wymagany. Określa rodzaj zdarzenia, które uruchamia zadanie zawarte w znaczniku `<onevent>`. Wartość `onenterbackward` oznacza nawigację wykorzystującą historię przeglądarki. `onenterforward` oznacza dowolny inny rodzaj nawigacji. `ontimer` wykonuje zadanie po określonym czasie.

<option>

```
<option>...</option>
```

Określa jedną z opcji w liście `<select>`.

Atrybuty

`value=`*`wartość`*

Oznacza opcjonalną zmienną związaną z opcją, na przykład przechowującą skróconą wersję pełnej etykiety opcji.

`title=`*`tekst`*

Nadaje opcji nazwę.

`onpick=`*`id karty`*

Określa identyfikator karty wyświetlanej po wybraniu opcji.

<optgroup>

```
<optgroup>...</optgroup>
```

Wyróżnia grupy elementów `<option>` w obrębie listy `<select>`.

Atrybuty

`title=`*`tekst`*

Nadaje tytuł tworzonej grupie.

<p>

```
<p>...</p>
```

Określa akapit tekstowy.

Atrybuty

`align=left|center|right`

Wyrównuje tekst w poziomie.

`mode=wrap|nowrap`

Określa tryb dzielenia linii tekstu w akapicie. `nowrap` powoduje, że tekst jest wyświetlany w jednej linii, co może wymagać poziomego przewijania.

<postfield>

```
<postfield/>
```

Definiuje parę nazwa-wartość, która zostanie wysłana do serwera.

Atrybuty

`name=`*`tekst`*

Wymagany. Określa nazwę pary.

`value=`*`wartość`*

Wymagany. Określa zmienną (wartość) pary.

<prev>

```
<prev>...</prev> lub <prev/>
```

Nakazuje przeglądarce cofnięcie się do poprzednio wyświetlanej karty. Standardowy atrybut `xml:lang` nie jest dozwolony w tym elemencie.

<refresh>

```
<refresh>...</refresh>
```

Oznacza operację odświeżania, która uaktualnia zawartość okna przeglądarki (co oznacza wyczyszczenie historii i przechowywanych zmiennych).

<select>

```
<select>...</select>
```

Definiuje listę opcji w formularzu. Element `<select>` zawiera pewną liczbę elementów `<option>` i może także zawierać elementy `<optgroup>`.

Atrybuty

`title=`*`tekst`*

Określa tytuł, który może, ale nie musi być wyświetlany przez przeglądarkę.

`name=`*`tekst`*

Nazwa zmiennej (wartość jest nadawana w momencie, gdy użytkownik dokona wyboru którejś z opcji).

`value=`*`wartość`*

Pozwala na określenie wartości początkowej.

`iname=`*`wartość`*

Określa zmienną, która przechowywać będzie indeks wybranej opcji.

`ivalue=numer indeksu`

Określa domyślną opcję poprzez numer jej indeksu.

`multiple=true|false`

Oznacza możliwość wyboru wielu opcji z listy. Domyślną wartością jest `false` (dozwolona tylko jedna opcja).

`tabindex=liczba`

Określa kolejność, w jakiej przełączane są aktywne elementy formularza.

<setvar>

```
<setvar/>
```

Określa parę nazwa-wartość dla zmiennej. Jeśli zmienna o danej nazwie już istnieje, jej wartość jest nadpisywana. Element `<setvar>` może być użyty tylko wewnątrz znaczników `<refresh>` i `<go>`.

Atrybuty

`name=nazwa`

Wymagany. Określa nazwę zmiennej.

`value=wartość`

Wymagany. Określa wartość zmiennej.

<small>

```
<small>...</small>
```

Oznacza tekst o rozmiarze nieco mniejszym niż domyślny.


```
<strong>...</strong>
```

Oznacza tekst wyraźnie wyszczególniony. Sposób implementacji zależy od urządzenia.

<table>

```
<table>...</table>
```

Oznacza początek i koniec tabeli.

Atrybuty

`title=tekst`

Określa tytuł tabeli, który może, ale nie musi być wyświetlany przez przeglądarkę.

`align=kod wyrównania`

Określa poziome wyrównanie zawartości komórki w tabeli. Wartość ma postać ciągu znaków L, C i R (oznaczających odpowiednio: lewo, środek, prawo), po jednym dla każdej kolumny

tabeli. Tak więc wyrównanie dla tabeli o czterech kolumnach może być zdefiniowane przez `align="RLRC"`.

`columns=`*`liczba`*

Wymagane. Określa liczbę kolumn w tabeli. Jest to różnica w stosunku do składni znacznika tabeli w HTML.

<td>

`<td>...</td>`

Definiuje pojedynczą komórkę w tabeli. Zawartość komórki musi znajdować się wewnątrz znaczników `<td>`.

<template>

`<template>...</template>`

Definiuje szablon dla wszystkich kart w talii, zawierający definicje klawiszy programowanych, stworzone za pośrednictwem elementów `<do>`.

<timer>

`<timer/>`

Ustawia licznik czasu, który może być zastosowany do uruchamiania zadań.

Atrybuty

`name=`*`nazwa`*

Nazwa zegara, do której odwołuje się element obsługujący zdarzenie.

`value=`*`wartość`*

Wymagany. Ustawia długość odmierzanego czasu (w dziesiątych częściach sekundy).

<tr>

`<tr>...</tr>`

Określa rząd w tabeli. Jego zawartością jest pewna liczba elementów `<td>` (komórek tabeli). Atrybut `xml:lang` nie ma zastosowania do tego elementu.

<u>

`<u>...</u>`

Oznacza tekst podkreślony.

<wml>

`<wml>...</wml>`

Oznacza początek i koniec talii. Jest elementem głównym dokumentu (talii) WML.

Informacje o WAP i WML

W sieci dostępnych jest kilka znakomitych witryn poświęconych WAP i WML:

WAP Forum

http://www.wapforum.org

Oficjalna witryna grupy opracowującej standard WAP.

AllNetDevices Wireless FAQ

http://www.allnetdevices.com/faq/

Świetne miejsce na rozpoczęcie nauki o WAP i WML.

Openwave (dawniej Phone.com oraz Unwired Planet)

http://www.openwave.com

Openwave jest producentem przeglądarek i narzędzi do projektowania aplikacji WAP.

Openwave Developer Program

http://developer.phone.com

Prowadzona przez Openwave witryna dla projektantów WAP zawiera wyczerpujące dane. Warto zobaczyć.

W3Schools.com

http://www.w3schools.com/wap/

http://www.w3schools.com/wmlscript/

Witryna W3Schools oferuje samouczki zarówno WML, jak i WMLScript.

WMLScript Primer

http://www.webreference.com/js/column62/

Artykuł z WebReference.com dobry dla początkujących w dziedzinie WMLScript.