

XUL i Mozilla

Budowanie aplikacji

Spośród wszystkich wynalazków z ostatniego półwiecza tylko kilka pomysłów dotarło do etapu, w którym korzystamy z nich w życiu codziennym.

JONO BACON



Mimo że Sinclair C5 czy Microsoft FoxPro nie dały sobie rady, nadal istnieje kilka produktów i technologii, które mają na to szansę. Jedną z takich technologii jest sieć komputerowa.

Pojawia się wiele wątpliwości, czy faktycznie Internet wywarł tak ogromny wpływ na sposób komunikowania się między ludźmi, robienie zakupów czy inne, codzienne czynności. Pomimo wojny przeglądarek, nieudolnego wprowadzania standardów i prób cenzurowania, Sieć udowodniła, że jest dynamicznym, ogólnosięciowym medium. Niestety, wciąż zdarza się, że słyszymy jeszcze gdzieś na świecie takie słowa: „Jak to? Nie masz jeszcze Internetu?”.

Pomimo postępującej adaptacji Internetu, ograniczenia tego środka przekazu są coraz bardziej oczywiste. Najważniejszym

z nich jest fakt, że dla każdego nowego serwisu internetowego trzeba opracowywać interfejs od początku. Ponadto interfejs powinien zostać zaktualizowany ponownie wraz z każdą zmianą dotyczącą zawartości stron. Jest to nie tylko nieefektywne (będzie przesyłany zbędny kod HTML), ale także tworzy środowisko, w którym dynamiczne zmiany strony są trudniejsze do przeprowadzenia.

Wejście XUL

Prawdziwym sednem problemu jest uzależnienie od języka HTML i funkcjonalności przeglądarki. Mimo że nowsze technologie, takie jak dynamiczny HTML (DHTML) czy obiektowy model dokumentu (DOM), próbują poradzić sobie z tym problemem, muszą one osiągnąć pewien poziom porozumienia między sobą, aby wykonywały

swoje zadania zgodnie z oczekiwaniami.

Programiści stojący za sukcesem popularnej przeglądarki Mozilla wpadli na inny pomysł. Po dyskusjach opracowali oni język XML User Interface Language (XUL). Nazwę tą wymawiamy „zuul” (inspirację zaczerpnięto z filmu „Łowcy duchów”). Podstawową funkcją tego języka jest odtworzenie w ramach serwisów WWW interfejsu użytkownika, zwykle kojarzonego z narzędziami graficznymi typu Qt czy GTK. Korzystanie z przycisków, pasków przewijania, zakładek czy menu – to wszystkie cechy nowego zestawu narzędzi XUL. Większość tych możliwości nie jest dostępna w formie zwykłego kodu HTML. Każda z funkcji zawartych w języku XUL jest wykorzystywana z poziomu plików XML. Dla tych, którzy czują się nieswojo słysząc słowo XML, opiszemy krótko, czym jest ten język.

XML to specjalny zestaw zasad i konwencji, który umożliwia tworzenie języków wyglądających podobnie do języka HTML. Języki takie wykorzystują znaczniki, atrybuty i inne pojęcia składniowe znane z języka HTML. Od strony praktycznej technologia XML umożliwia użytkownikom przede wszystkim definiowanie własnych znaczników i języka, które mogą być używane we własnych aplikacjach. Przykładowo, jeśli chcielibyśmy przechowywać adresy w XML, moglibyśmy stworzyć następujące znaczniki:

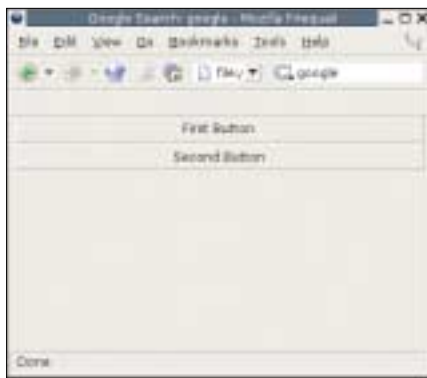
```
<address>
  <forename>Jan</forename>
  <surname>Kowalski</surname>
  <address>ul. Misiowa 3/2, 00-123
  Warszawa </address>
  <phone> 022 123 4567</phone>
</address>
```

Stworzyliśmy w ten sposób własne znaczniki opisujące określony typ danych. Dzięki temu możemy napisać program, który będzie odczytywał te znaczniki i korzystać z prezentowanych danych w określony sposób.

XUL działa podobnie, lecz znaczniki wykorzystuje się tutaj do tworzenia określonych elementów interfejsu. Magia pojawia się dopiero wtedy, gdy Mozilla odczytuje znaczniki i na tej podstawie tworzy elementy interfejsu użytkownika. Plik XML służy tutaj do prostego określania, czego potrzebujemy w swoim interfejsie i w którym miejscu.

Wprowadzenie

W pierwszej części cyklu pokazemy sposób modelowania interfejsu w oparciu o język XUL. Interfejs będzie składał się z różnych elementów, dostępnych w XUL. Mimo że nie będziemy zajmować się obecnie sposobem tworzenia obiektów (opiszemy to w kolejnym numerze LM), zdobędziemy jednak solidne podstawy tworzenia inter-



Rysunek 1: Może to niezbyt mądry skrypt XUL, ale przecież to dopiero początek!

fejsów graficznych. Rozpocniemy od prostego pliku XUL, w którym umieścimy dwa przyciski. Utwórzmy zatem plik o nazwie xul1.xul, zawierający kod programu pokazany na Listingu 1.

Po wpisaniu kodu uruchamiamy przeglądarkę Mozilla i, używając menu File | Open File (Plik | Otwórz plik), otwieramy go w przeglądarce. Powinniśmy zobaczyć wynik podobny do pokazanego na Rysunku 1.

Każdy plik XML, bez względu na to, jakie ma działanie, powinien posiadać na samym początku kilka standardowych linii kodu definiujących wersję XML i używany szablon (jeżeli występuje). W naszym przykładzie są to:

```
<?xml version='1.0'?>
<?xml-stylesheet href='chrome://global/skin/' type='text/css'?>
```

Jak widać, zdefiniowano tutaj wersję 1.0 języka XML. Druga linia określa ścieżkę do szablonu (w naszym przypadku szablon chrome). Ścieżka do szablonu chrome zawiera wewnętrzne obiekty Mozilli, które zarządzają interfejsem użytkownika tej przeglądarki. W kolejnych częściach przyjrzymy się bliżej szablonom (w tym chrome).

Kolejne linie zawiera pierwszy znacznik:

```
<window
  id='firstwindow'
  title='Pierwsze okno XUL'
  xmlns='http://www.mozilla.org/
  keymaster/gatekeeper/there.is.only.xul'>
```

Mimo że mamy tutaj cztery linie kodu, jest to w rzeczywistości jeden znacznik. Rozdzieliliśmy je po prostu w celu zwiększenia czytelności. Każda stworzona strona XUL wymaga znacznika <window>, w którym umieszczane będą obiekty tworzące interfejs. Wewnątrz znacznika znajdują się także trzy atrybuty. Pierwszy z nich (id) to niepowtarzalny odnośnik wskazujący ten znacznik w pliku XML. Atrybut id jest istotny ze względu na możliwość komunikowania się ze znacznikami i aktualizowania ich wraz z wprowadzanymi zmianami i danymi. Bardziej szczegółowo wyjaśnimy funkcję tego atrybutu w dalszej części artykułu, gdy skorzystamy ze standardu DOM do rzeczywistej aktualizacji znaczników. Drugi znacznik (title) zawiera czytelny dla użytkownika ciąg znaków, który będzie wyświetlany na pasku tytułu, gdy tylko otworzymy plik XUL w oddzielnym oknie. Jeżeli odczytamy plik w przeglądarce, tak jak to zrobiliśmy wcześniej – tekst nie zostanie wyświetlony. Ostatnim atrybutem jest xmlns. Jego wartość określa przestrzeń nazw, z której będzie korzystać znacznik <window> i wszystkie znaczniki znajdujące się wewnątrz niego. Przestrzeń nazw do specjalna grupa, którą możemy określić w celu podania źródła pochodzenia danego znacznika. Jest to bardzo pomocne w sytuacjach, kiedy znacznik <window> z innego języka XML występuje wraz ze znacznikiem <window> z języka XUL – przestrzeń nazw odpowiednio je rozdzieli.

Po stworzeniu okna jesteśmy gotowi do umieszczenia w nim nowych elementów. W naszym prostym przykładzie tworzymy dwa przyciski, używając następującego kodu:

```
<button id='button1' label='
  Pierwszy przycisk' />
<button id='button2' label='
  Drugi przycisk' />
```

Obie linie kodu są bardzo podobne do siebie – różni je jedynie zawartość atrybutów id i label. Atrybut id zachowuje się dokładnie tak samo jak <window> – jest wykorzystywany do wskazywania znacznika w późniejszym czasie. Atrybut label zawie-

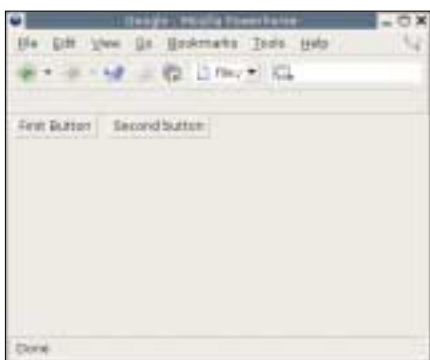
Listing 1: Plik xul1.xul

```
<?xml version='1.0'?>
<?xml-stylesheet href='chrome://global/skin/' type='text/css'?>

<window
  id='firstwindow'
  title='First XUL Window'
  xmlns='http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul'>
  <button id='button1' label='Pierwszy przycisk' />
  <button id='button2' label='Drugi przycisk' />
</window>
```

ra tekst, który pojawi się na przycisku.

Część użytkowników może zastanawiać się, w jakim celu w znaczniku pojawia się ukośnik (/). W odróżnieniu od niektórych form języka HTML, gdzie możliwe jest wstawianie znaczników w dowolne miejsca kodu, język XML jest rygorystyczny, jeśli chodzi o prawidłowe oznaczenia. Dlatego znacznik <button> powinien być zawsze zamykany przez </button>. Ukośnik / na końcu znacznika <button> w przykładzie powyżej jest skróconą formą znacznika </button>. W języku XML spotkamy tę formę zapisu zapisu bardzo często. Na końcu pliku musimy umieścić też znacznik zamykający </window>.



Rysunek 2: Ułożenie elementów okna w poziomie.

Zarządzanie układem strony

W przedstawionym przykładzie od razu rzuca się w oczy sposób umieszczenia przycisków – jeden pod drugim. Jest to domyślne zachowanie dla pojawiających się elementów, dla których nie ustalono określonego położenia. Takie rozwiązanie może znaleźć zastosowanie na prostych stronach internetowych, lecz metoda ta jest dla naszych potrzeb za mało elastyczna. Z pomocą przyjdzie nam Layout Manager.

Zarządzanie układem strony jest czymś naturalnym w programach z interfejsem GUI typu Qt czy GTK. Większość programów składa się z menedżera pionowego i poziomego. Właśnie ta metoda obsługi układu strony została wykorzystana w języku XUL, tak więc mamy do dyspozycji znaczniki <hbox> i <vbox>. Poniżej podaliśmy przykład działania znacznika <hbox>:

```
<hbox>
<button id='button1' label=▶
'Pierwszy przycisk' />
<button id='button2' label=▶
```

```
'Drugi przycisk' />
</hbox>
```

Znacznik <hbox> pozwala na poziome umieszczanie obok siebie elementów okna znajdujących się pomiędzy znacznikami <hbox> i </hbox>. Rezultat jego użycia pokazano na Rysunku 2.

Kolejnym sposobem zarządzania układem strony jest wykorzystanie znacznika <vbox>. Znacznik ten zachowuje się dokładnie jak <hbox>, z tą różnicą, że wyświetla kolejne elementy okna (elementy wewnątrz znaczników <vbox> i </vbox>) w pionie. Jeżeli pod blokiem <hbox> umieścimy podobny blok jak dla <hbox>, zobaczymy wynik działania tego znacznika. Ponadto do przycisków zarządzanych w pionie dopisaliśmy literę (H), a do przycisków zarządzanych w poziomie – literę (V):

```
<hbox>
<button id='button1' label=▶
'First Button (H)' />
<button id='button2' label=▶
'Second button (H)' />
</hbox>
<vbox>
<button id='button1' label=▶
'First Button (V)' />
<button id='button2' label=▶
'Second button (V)' />
</vbox>
```

Najbardziej interesujące efekty osiągniemy jednak dopiero po połączeniu obu znaczników. Spójrzmy na poniższy przykład:

```
<hbox>
<button id='button1' label=▶
'Pierwszy przycisk (H)' />
<button id='button2' label=▶
'Drugi przycisk (H)' />
<vbox>
<button id='button1' label=▶
'Pierwszy przycisk (V)' />
<button id='button2' label=▶
'Drugi przycisk (V)' />
</vbox>
</hbox>
```

W tym przykładzie umieściliśmy przyciski ułożone pionowo wewnątrz przycisków ułożonych poziomo. Przyciski ułożone pionowo względem siebie zostały umieszczone za pozostałymi przyciskami w linii poziomej. Układ elementów okna powinien mieć teraz postać zbliżoną do tej z Rysunku 3.

Podczas umieszczania elementów okna musimy pamiętać o sposobie zarządzania przestrzenią okna. W naszym ostatnim przykładzie przyciski ułożone w poziomie zostały rozciągnięte w dół, aby przygotować miejsce dla dwóch przycisków ułożonych w pionie. Spowodowane jest to umieszczeniem menedżera pionowego wewnątrz menedżera poziomego, co miało wpływ właśnie na elementy ułożone w poziomie.

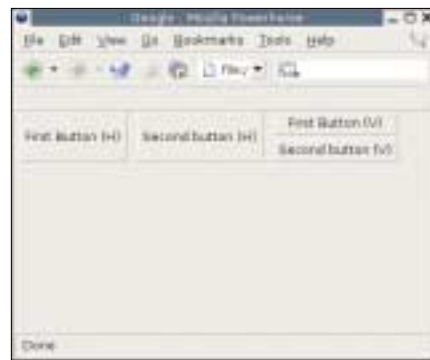
Kolejne elementy okna

Prawdziwa moc języka XUL leży jednak w sposobie, w jaki pozyskuje on informacje od użytkownika. Zachowanie interfejsu nie różni się od tego, co można spotkać w systemach z graficznym interfejsem użytkownika (GUI). W kolejnym przykładzie stworzymy dwie zakładki. Jedna z nich będzie zawierać wielowierszowe okno edycji tekstu, a druga pole listy. Pamiętajmy, że zakładki i pole listy nie są zwykle stosowane w środowisku WWW. Przeanalizujemy jednak ten przykład krok po kroku, pisząc jednocześnie kod strony.

Przede wszystkim tworzymy nowy plik w wersji XML, znacznik szablonu strony i <window>, a następnie dopisujemy poniższe linie:

```
<tabbox>
<tabs>
<tab label='Edytor tekstu' />
<tab label='Lista' />
</tabs>
```

Zaczynamy tu tworzyć nowy element okna zawierający zakładki (<tabbox>). W tabbox musimy umieścić listę zakładek, które z kolei będą mogły przechowywać kolejne elementy okna. Następnie otwieramy znacznik <tabs> i podajemy nazwy zakładek naszego interfejsu. Dla każdej zakładki musimy użyć osobnego znacznika <tab> –



Rysunek 3: Połączenie zarządzania w pionie i w poziomie.

Elementy HTML

W naszej wędrowce po świecie XUL korzystaliśmy do tej pory wyłącznie z menedżerów układu okna i przycisków. Istnieje jednak znaczny wybór elementów okien, z których możemy korzystać. Na początek skupimy się na standardowych elementach pochodzących z języka HTML. Aby skorzystać z elementów tego typu, musimy użyć następującego kodu:

```
<vbox>
<hbox>
  <label value='Checkboxes' />
  <vbox>
    <checkbox id='check1' label=
'Pierwszy' />
    <checkbox id='check2' label=
'Drugi' />
  </vbox>
</hbox>
<hbox>
  <label value='Przyciski radiowe' />
  <vbox>
    <radio id='radio1' label=
'Pierwszy' />
```

```
<radio id='radio2' label=
'Drugi' />
</vbox>
</hbox>
<hbox>
  <label value='Pole tekstowe' />
  <vbox>
    <textbox id='textbox' />
  </vbox>
</hbox>
<hbox>
  <label value='Multiline
Text box' />
  <textbox id='multitextbox'
multiline='true' />
</hbox>
</vbox>
```

Do wyświetlania tekstu w naszym interfejsie XUL wykorzystamy znacznik `<label>`. Jako atrybut tego znacznika podajemy tekst, który zostanie wyświetlony jako etykieta.

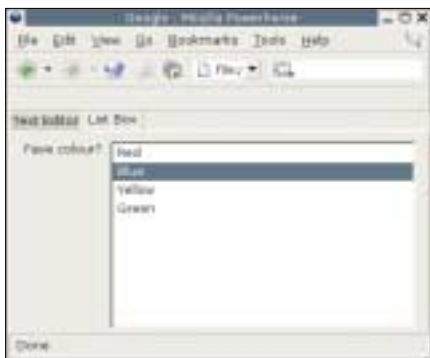


Pierwszy element okna to pole wyboru. Tworzymy je przy pomocy znacznika `<checkbox>` i opisujemy przy użyciu atrybutu `label` (etykieta), czyli opisu, który pojawi się obok pola wyboru. Drugim elementem jest przełącznik radiowy. Sposób użycia znacznika `<radio>` jest taki sam jak w przypadku poprzedniego elementu okna. Kolejnym elementem jest jednowierszowe pole tekstowe. Dla pola tekstowego nie ma przypisanej żadnej etykiety, więc wystarczy w znaczniku `<textbox>` ustawić tylko atrybut `id`. Ostatnim z nich jest wielowierszowe pole tekstowe, aby je stworzyć, dodajemy do zwykłego pola tekstowego parametr `multiline="true"`. Nasz interfejs powinien teraz wyglądać tak jak na Rysunku 4.

w ten sposób określamy, która nazwa dotyczy danej zakładki. Zakładki będą dodawane od lewej do prawej w kolejności określonej w pliku XUL. W naszym przykładzie zakładka Text Editor pojawi się po lewej stronie, a zakładka List Box – po prawej. Teraz musimy stworzyć właściwe panele zakładek – najpierw tworzymy znacznik paneli zakładek:

```
<tabpanel>
```

Teraz możemy stworzyć panele, rozpoczynając od panelu z polem tekstowym. Do utworzenia każdego panelu używamy znacznika `<tabpanel>`, a następnie wypełniamy go kolejnymi elementami okna.



Rysunek 5: Użycie zakładek i pola listy w interfejsie.

Przyjrzyjmy się następnemu przykładowi:

```
<tabpanel id='text'>
  <label value='§§' />
  <label value='Wpisz tutaj jakiś tekst:' />
  <textbox id='textbox'
multiline='true' flex='1' />
</tabpanel>
```

W podanym kodzie pojawił się atrybut `flex`. Jeżeli ustawimy wartość tego atrybutu na 1, element okna zostanie rozciągnięty, zajmując całe dostępne miejsce okna.

Wewnątrz drugiego panelu musi powstać pole listy. Aby utworzyć główne pole, korzystamy ze znacznika `<listbox>`, a następnie

do stworzenia poszczególnych elementów w tym polu używamy znacznika `<listitem>`.

```
<tabpanel id='listbox'>
  <label value='Jaki jest Twój
ulubiony kolor?' />
  <listbox flex='1'>
    <listitem label='Czerwony' />
    <listitem label='Niebieski' />
    <listitem label='Żółty' />
    <listitem label='Zielony' />
  </listbox>
</tabpanel>
```

Na koniec zamykamy panele zakładek i główne pole zakładki:

```
</tabpanel>
</tabbox>
```

Gotowy interfejs możemy zobaczyć na Rysunku 5.

Gotowe interfejsy

Aby podsumować pierwszą część programowania XUL, dokonamy analizy gotowego przykładu interfejsu XUL. Interfejs będzie wykorzystywał kod omówiony wcześniej w tym artykule oraz kilka menu i splitter (regulowany podział strony). W celu utrwalenia wiadomości przeanalizujemy

Listing 2: Główne okno

```
<?xml version='1.0'?>
<?xml-stylesheet href='
chrome://global/skin/'
type='text/css'?>

<window
  id='complete'
  title='Przykładowa aplikacja'
  xmlns='http://www.mozilla.org/
keymaster/gatekeeper/there.is.
only.xul'>
```

zujemy każdą linię kodu.

Zacniemy od znaczników definicji XML i utworzenia głównego okna. Spójrzmy na Listing 2.

Pierwsze elementy okna, które dopiszemy do kodu strony, to kilka menu. Nie zajmowali się jeszcze nimi, ale tworzenie menu podlega tym samym zasadom ogólnym, które omawialiśmy wcześniej. Zacniemy od utworzenia paska menu (menu, w którym umieszczone zostaną kolejne menu) przy pomocy znacznika `<menubar>`.

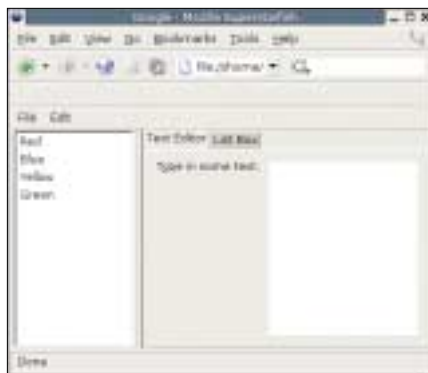
```
<menubar id='menubar'>
```

Następnie dołączamy gotowe menu. W tym przypadku – menu File:

```
<menu id='filemenu' >
  label='Plik'>
  <menupopup id='file-popup'>
  <menuitem label='Nowy' />
  <menuitem label='Otwórz' />
  <menuitem label='Zapisz' />
  <menuseparator />
  <menuitem label='Zakończ' />
  </menupopup>
</menu>
```

Aby utworzyć menu, musimy najpierw użyć znacznika `<menu>`, co pozwoli stworzyć właściwy wpis menu, a następnie tworzymy menu rozwijane i menu podręczne przy pomocy znacznika `<menupopup>`. Na koniec dołączymy kilka pozycji menu przy użyciu znaczników `<menuitem>`. W ten sam sposób tworzymy menu Edit:

```
<menu id='editmenu' >
  label='Edycja'>
  <menupopup id='editpopup'>
  <menuitem label='Cofnij' />
  <menuitem label='Powtóż' />
```



Rysunek 6: Gotowy interfejs XUL.

```
</menupopup>
</menu>
</menubar>
```

Po stworzeniu menu jesteśmy gotowi do rozpoczęcia prac nad głównym obszarem naszego interfejsu. Na Rysunku 6 pokazano wynik naszych działań – nowy wygląd interfejsu.

Nasz interfejs posiada po lewej stronie ekranu pole listy, a po prawej stronie – zakładki. Aby zająć się rozmieszczeniem poszczególnych elementów, musimy najpierw otworzyć znacznik `<hbox>`, a następnie utworzyć pole listy:

```
<hbox>
  <listbox flex='1'>
  <listitem label='Czerwony' />
  <listitem label='Niebieski' />
  <listitem label='Żółty' />
  <listitem label='Zielony' />
  </listbox>
```

Teraz użyjemy specjalnego elementu okna zwanego splitterem. Dzięki niemu dołączymy specjalny pasek, który umożliwi użytkownikowi regulację wielkości elementu okna po lewej i prawej stronie rozdzielacza.

Aby utworzyć ten element okna, używamy znacznika `<splitter/>`:

```
<splitter/>
```

Kolejna część kodu strony jest znanym już nam polem zakładek, które zawiera zakładki do edycji tekstu i pola listy. Spójrzmy na Listing 3. Podany kod strony nie różni się niczym od naszego poprzedniego przykładu:

```
<tabbox>
  <tabs>
  <tab label='Edytor tekstu' />
  <tab label='Lista' />
  </tabs>
  <tabpanel id='text'>
  <label value='Wpisz tutaj >
  jakiś tekst:' />
  <textbox id='textbox' >
  multiline='true' flex='1' />
  </tabpanel>
  <tabpanel id='listbox'>
  <label value='Jaki jest Twój >
  ulubiony kolor?' />
  <listbox flex='1'>
  <listitem label='Czerwony' />
  <listitem label='Niebieski' />
  <listitem label='Żółty' />
  <listitem label='Zielony' />
  </listbox>
  </tabpanel>
  </tabpanel>
  </tabbox>
```

Na koniec zamykamy menedżera poziomu i samo okno:

```
</hbox>
</window>
```

Podsumowanie

W pierwszej części naszego mini-cyklu zajęliśmy się podstawami programowania XUL. Poznaliśmy podstawy, specjalne elementy okien, zarządzanie układem strony, pola zakładek, menu i wiele innych. Zdobylismy umiejętność do tworzenia interesujących interfejsów XUL. Oczywiście istnieje jeszcze wiele nieopisanych tutaj elementów okien, którymi zajmiemy się w kolejnych numerach LM.

W przyszłym miesiącu pokażemy pełnię mocy XUL. Poznamy możliwości skryptów Java w przeglądarce Mozilla, dzięki którym nasze interfejsy będą mogły bliżej współdziałać z użytkownikiem. ■

Listing 3: Pole zakładek

```
<tabbox>
  <tabs>
  <tab label='Edytor tekstu' />
  <tab label='Lista' />
  </tabs>
  <tabpanel id='text'>
  <label value='Wpisz tutaj jakiś tekst:' />
  <textbox id='textbox' multiline='true' flex='1' />
  </tabpanel>
  <tabpanel id='listbox'>
```