# Ajax. Niezbędnik projektanta dynamicznych aplikacji


Autor: Michael Morrison
Tłumaczenie: Maciej Jezierski
ISBN: 978-83-246-1393-9
Tytuł oryginału: Ajax Construction Kit: Building Plug-and-Play Ajax Applications (Negus Live Linux Series)
Format: 170x230, stron: 280

**Wykorzystaj gotowe rozwiązania w technologii AJAX!**

- Jak wykorzystać technologię AJAX?
- Jak wykonać najczęściej spotykane zadania?
- Jak zwiększyć interaktywność serwisów WWW?

Technologia AJAX (skrót od ang. Asynchronous JavaScript and XML) pozwala na tworzenie dynamicznych stron WWW, dostarczających niezapomnianych wrażeń ich użytkownikom. Praktycznie wszystkie współczesne aplikacje WWW oraz duża część witryn internetowych korzystają z dobrodziejstw tego rozwiązania, a jeszcze dziś również i Twoja strona może zyskać na interaktywności.

Dzięki książce „Ajax. Niezbędnik projektanta dynamicznych aplikacji" poznasz zasady działania technologii AJAX, jej zalety oraz wady. Autor prezentuje dziesięć przykładów zastosowania technologii AJAX. Dzięki nim nauczysz się między innymi, w jaki sposób pobierać dane z kanałów RSS, jak wykorzystać format XML oraz jak kontrolować wprowadzane przez użytkownika dane w czasie rzeczywistym. Najważniejsze jest jednak to, że każde z tych praktycznych rozwiązań możesz zaimplementować bezpośrednio na Twojej stronie!

- Zalety technologii AJAX
- Przykładowe rozwiązania, korzystające z AJAX
- Sposoby używania formatu XML
- Zasady funkcjonowania technologii AJAX
- Tworzenie interaktywnych formularzy
- Sposoby wykorzystania kanałów RSS
- Dynamiczne pobieranie danych z innych serwisów
- Sposób na efektowną galerię
- Tworzenie okien informacyjnych
- Przewidywanie treści wprowadzanych przez użytkownika
- Metody i właściwości obiektu XMLHttpRequest
- Przyszłość technologii AJAX

**Twoje strony też mogą być interaktywne!**

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

# Spis treści

# 5

# Pobieranie danych z kanałów RSS

**AJAX W STAROŻYTNEJ GRECJI**

Nie umniejszając roli asynchronicznych aplikacji internetowych, był czas, kiedy Ajax miał większe znaczenie. Mam na myśli czasy starożytnej Grecji, kiedy legendarny grecki bohater Ajaks walczył w wojnie trojańskiej. Mityczna siła i odwaga Ajaksa zostały opisane przez greckiego poetę Homera w poemacie *Iliada*.

Jako ktoś, kto przez kilka lat tworzył i utrzymywał wiele witryn internetowych, bardzo entuzjastycznie podszedłem do kanałów RSS, kiedy zobaczyłem je po raz pierwszy. Umożliwiają one wyświetlanie informacji z innych serwisów i możesz umieścić je na własnych stronach. Zapewniają automatyczną aktualizację witryny, a przynajmniej kanałów informacyjnych, które są na niej umieszczone. Jeśli nie masz wprawy lub nie chcesz skrupulatnie uzupełniać wiadomości, możesz przyjąć, że kanał informacyjny umieszczony na Twojej stronie pozwoli udostępniać interesujące, automatycznie aktualizowane informacje. Kanał informacyjny, który będzie idealnie dopasowany do zawartości Twojej strony, znajdziesz najprawdopodobniej już po chwili poszukiwań.

W tym rozdziale omówię, w jaki sposób Ajax umożliwia dynamiczne odpytywanie kanałów informacyjnych i wyświetlanie ich zawartości na stronie. Przykładowa aplikacja, którą poznasz w tym rozdziale, udostępnia prosty interfejs umożliwiający wybieranie różnych usług informacyjnych i następnie wyświetlenie ich na stronie. Oprócz tego podczas wykonywania żądania Ajaksa zobaczysz mały ładny obrazek, który jest niemal tak samo odjazdowy jak dynamiczne pobieranie wiadomości.

Poniżej znajduje się lista plików wykorzystywanych w aplikacji *Czytnik wiadomości* w tym rozdziale. Są one umieszczone w spakowanym archiwum zawierającym przykładowy kod dołączony do książki i znajdującym się na serwerze FTP wydawnictwa Helion pod adresem *ftp://ftp.helion.pl/przyklady/ajaxnp.zip*, w katalogu *chap05*.

- *newsfeeder.html* — strona główna,
- *newsfeeder.php* — skrypt serwera do pobierania danych z kanałów informacyjnych,
- *newsfeeder.css* — arkusz stylów do formatowania wiadomości,
- *ajaxkit.js* — podstawowe funkcje Ajaksa w bibliotece ajaksowej,
- *domkit.js* — funkcje do obsługi DOM w bibliotece ajaksowej,
- *newspaper.gif* — ikonka gazety wyświetlana obok każdej wiadomości,
- *wait.gif* — animowana ikonka wyświetlana podczas wykonywania żądania Ajaksa.

## Zadanie: dynamiczne odpytywanie kanałów informacyjnych

Zadaniem w tym rozdziale jest wykorzystanie Ajaksa do dynamicznego odpytywania usług internetowych udostępniających kanały informacyjne o dane (wiadomości), a następnie wyświetlanie ich na stronie. Mówiąc dokładniej, poszczególne wiadomości są tytułami pełnych wiadomości, więc właściwie musisz wyświetlać tytuły wiadomości, które będą służyć jako łącza do poszczególnych pełnych wiadomości. Jak się okazuje, każdy element w kanale informacyjnym zawiera między innymi zarówno tytuł, jak i łącze, tak więc kanał informacyjny udostępnia wszystkie dane wymagane do wykonania zadania.

Zanim zagłębimy się w zadanie, musimy przede wszystkim przyjrzeć się samym kanałom informacyjnym i ich roli w internecie. Większość kanałów informacyjnych oparta jest na technologii RSS. Umożliwia ona publikowanie zawartości WWW, co oznacza, że możesz łatwo pobierać informacje o nowościach pojawiających się na różnych witrynach bez konieczności ich odwiedzania. Chociaż skrót RSS był różnie interpretowany, najnowszym i najbardziej odzwierciedlającym jego zadania rozwinięciem jest Really Simple Syndication (ang.

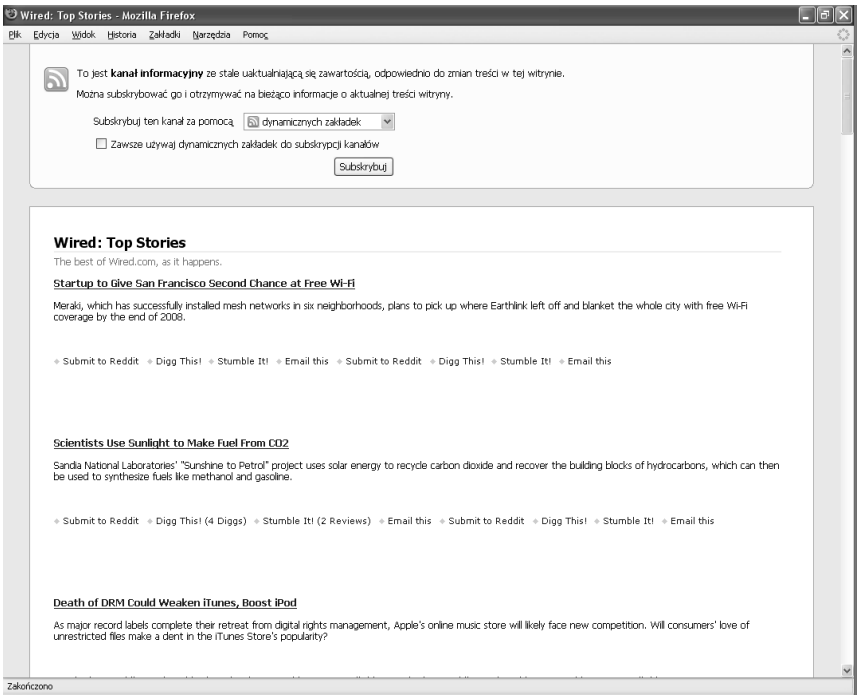
*naprawdę łatwe publikowanie*). Kanały informacyjne RSS są w rzeczywistości dokumentami XML zawierającymi wiadomości. Szerzej omówię je w dalszej części rozdziału.

**UWAGA**

Historia formatu RSS i jego różnych wersji jest dość skomplikowana, ale wystarczy wiedzieć, że najnowszą i najszerzej stosowaną wersją RSS jest RSS 2.0. Możesz także spotkać się z kanałami informacyjnymi w formacie Atom, który jest bardzo podobny do RSS 2.0.

Za pomocą specjalnego oprogramowania nazywanego czytnikiem kanałów możesz śledzić kanały informacyjne pochodzące z różnych witryn i mieć wygodny dostęp do szerokiego zestawu informacji bez potrzeby odwiedzania samych witryn. Możesz porównać RSS do pewnego rodzaju paska informacyjnego z wiadomościami z sieci, który umożliwia bieżące monitorowanie nowych publikacji na Twoich ulubionych witrynach.

Większość współczesnych przeglądarek posiada wbudowaną obsługę przeglądania kanałów informacyjnych, chociaż nie jest ona tak rozbudowana jak w wyspecjalizowanych czytnikach kanałów. Na rysunku 5.1 widać główne wiadomości witryny Wired wyświetlane w przeglądarce Firefox.




**Rysunek 5.1.** *Kanały RSS mogą być wyświetlane w większości przeglądarek internetowych bez potrzeby instalowania specjalnych wtyczek czy oprogramowania*

Oczywiście przeglądanie kanału informacyjnego bezpośrednio w przeglądarce to nie to samo co osadzanie go na własnej stronie. Tym zajmiemy się w naszym zadaniu — wykorzystaniem Ajaksa do dynamicznego pobierania i wyświetlania kanałów informacyjnych na stronie. Ponieważ każdy kanał informacyjny ma swój własny unikalny adres URL, ajaksowa część zadania jest właściwie bardzo prosta — pobranie całego dokumentu XML z określonego adresu. Oznacza to, że nie musisz w jakiś specjalny sposób przetwarzać danych po stronie serwera, co znacznie ułatwia sprawę.

Ponieważ żądanie Ajaksa w tej aplikacji jest tak proste, rozbudujemy trochę interfejs użytkownika, dodając do niego informację, że dane są wczytywane. Jest to o tyle ważne, że niektóre kanały informacyjne zawierają wystarczająco dużo danych, żeby widoczne było opóźnienie w ich odczycie. Nawet kilkusekundowe opóźnienie może być mylące, jeśli nie wiesz, co się dzieje. Eleganckim rozwiązaniem tego problemu jest wyświetlanie podczas wykonywania żądania obrazka sygnalizującego wczytywanie.

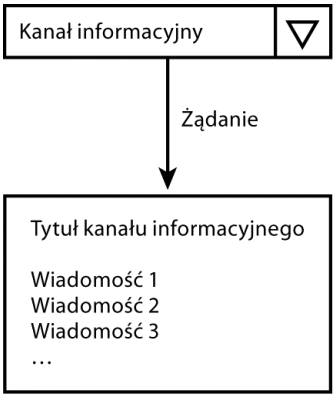
Ostatnią częścią zadania jest umożliwienie użytkownikowi wyboru spośród kilku różnych kanałów informacyjnych. Zapewni to nie tylko dobry sposób przeprowadzenia testów obciążeniowych, ale także większą elastyczność aplikacji z punktu widzenia użytkownika. Oczywiście podczas dołączania RSS do swojej strony możesz zdecydować się na wykorzystanie pojedynczego kanału i zrezygnować z możliwości wyboru. Tak czy siak, przykładowa aplikacja *Czytnik wiadomości* jest dobrym początkiem Twoich przygód z kanałami RSS w Ajaksie.

## Projekt: aplikacja do odczytu wiadomości

Projekt aplikacji *Czytnik wiadomości* rozpoczniemy od żądania Ajaksa, w którym do aplikacji przesyłane są dane z serwera kanałów informacyjnych. Żądanie rozpoczyna się przesłaniem adresu URL do serwera, który następnie zwraca dokument XML zawierający szczegółowe dane kanału informacyjnego, włączając w to listę z poszczególnymi wiadomościami, łączami do nich oraz datami publikacji. Żądanie Ajaksa jest wywoływane przez użytkownika, który wybiera kanał informacyjny na liście.

Biorąc pod uwagę to, że każdy kanał informacyjny ma własny adres URL, który jest wykorzystywany jako podstawa żądania Ajaksa, musisz znaleźć sposób, żeby powiązać URL z nazwą kanału wyświetlaną użytkownikowi na liście wyboru. Atrybut `value` znacznika `<option>` idealnie nadaje się do przechowywania adresu URL kanału i umożliwia łatwe tworzenie listy wyboru.

Lista wyboru kanałów informacyjnych jest głównym elementem interfejsu użytkownika aplikacji *Czytnik wiadomości*. Interfejs ten składa się z dwóch części: listy wyboru kanałów informacyjnych oraz miejsca, w którym wyświetlane są wczytane dane. Dane kanałów informacyjnych w tym przypadku składają się z tytułu kanału oraz poszczególnych wiadomości służących jako łącza do pełnej wersji wiadomości. Na rysunku 5.2 widać, w jaki sposób interfejs użytkownika aplikacji *Czytnik wiadomości* uczestniczy w żądaniu Ajaksa pobierającym kanał informacyjny.




**Rysunek 5.2.** *Interfejs użytkownika aplikacji Czytnik wiadomości składa się z listy wyboru i miejsca, w którym wyświetlany jest tytuł kanału oraz poszczególne wiadomości*

Wcześniej wspomniałem, że aplikacja powinna wykorzystać animowany obrazek, żeby wskazać użytkownikowi, że wykonywane jest żądanie Ajaksa pobierające dane z kanału informacyjnego. Na rysunku 5.3 widać, jak wygląda wykorzystywany w tym celu animowany obrazek *wait.gif*.

**Rysunek 5.3.** *Aplikacja Czytnik wiadomości wykorzystuje animowany obrazek (wait.gif), żeby wskazać użytkownikowi, że wykonywane jest żądanie Ajaksa*

Oczywiście trudno jest pokazać animację na drukowanej stronie, ale ruch obrazka daje użytkownikowi poczucie, że coś się dzieje i strona się nie zawiesiła. Mógłbyś zakwestionować pomysł wyświetlania użytkownikowi w asynchronicznej aplikacji animowanego obrazka oznaczającego wczytywanie. Czy w Ajaksie nie chodzi o to, żeby użytkownik nie musiał na nic czekać? Tak, ale pamiętaj o tym, że aplikacja nie jest blokowana podczas żądania — użytkownik może wybrać inny kanał informacyjny, co spowoduje przerwanie wczytywania bieżącego kanału i załadowanie nowego. Animowany obrazek służy po prostu do wskazania użytkownikowi, że w tle wykonywane są jakieś zadania.

Powinieneś już mieć wystarczające pojęcie o ogólnym projekcie aplikacji, żeby przejść do szczegółów żądania i odpowiedzi Ajaksa.

## Żądanie klienta

Gdyby nie było ograniczeń bezpieczeństwa, żądanie klienta w aplikacji *Czytnik wiadomości* byłoby najprostszą częścią do zaimplementowania, ponieważ po prostu wykonywałbyś żądanie `GET` adresu URL bez żadnych parametrów. Jednak nie można wykonać żądania Ajaksa bezpośrednio do zdalnego serwera internetowego, co oznacza, że będziesz musiał stworzyć własny skrypt po stronie serwera obsługujący takie żądania.

Wszystkim, czego potrzeba do pobrania danych z kanału informacyjnego, jest adres URL serwera, który go udostępnia. Dlatego też w żądaniu aplikacji *Czytnik wiadomości* od klienta do skryptu serwera musi zostać przesłany adres URL kanału informacyjnego. Następnie serwer obsłuży takie żądanie, pobierze dane i odeśle je w odpowiedzi Ajaksa. Kluczowym elementem jest znalezienie odpowiednich kanałów RSS, co przekłada się na poznanie ich adresów URL.

Jeśli zaczniesz się baczniej przyglądać witrynom internetowym, które odwiedzasz, możesz zauważyć na nich łącze „kanały rss" albo nawet ikonkę RSS umieszczone w pobliżu głównego menu lub łączy nawigacyjnych. W ten sposób możesz znaleźć adresy URL kanałów informacyjnych, które są wszystkim, czego potrzebujesz do wczytania kanałów RSS za pomocą Ajaksa.

**UWAGA**

Większość informacyjnych witryn internetowych, takich jak witryny z wiadomościami, udostępnia wiele kanałów o określonej tematyce. Na przykład *USA Today* udostępnia ponad 30 różnych kanałów sportowych, począwszy od szkolnego hokeja, do wyścigów konnych. Żeby uzyskać do nich dostęp, musisz po prostu przewinąć do dołu główną stronę witryny *USA Today* (*http://www.usatoday.com/*) i kliknąć małą pomarańczową ikonkę XML. Taka ikonka jest często wykorzystywana zamiast nowszej ikonki RSS.

## Odpowiedź serwera

Strona serwera aplikacji *Czytnik wiadomości* jest odpowiedzialna tylko za dostarczenie dokumentu XML zawierającego informacje z kanału wiadomości w formacie RSS, który jest formatem XML. W przeciwieństwie do niektórych przykładów w tej książce, w których skrypt PHP przetwarza dane przed zwróceniem ich w odpowiedzi Ajaksa, aplikacja *Czytnik wiadomości* wymaga tylko, żeby skrypt PHP służył jako obejście ograniczenia uniemożliwiającego wysłanie żądania klienta do zewnętrznego serwera. Jest to ograniczenie zabezpieczające kod JavaScript po stronie klienta przed odwoływaniem się poza domenę, z której został wczytany.

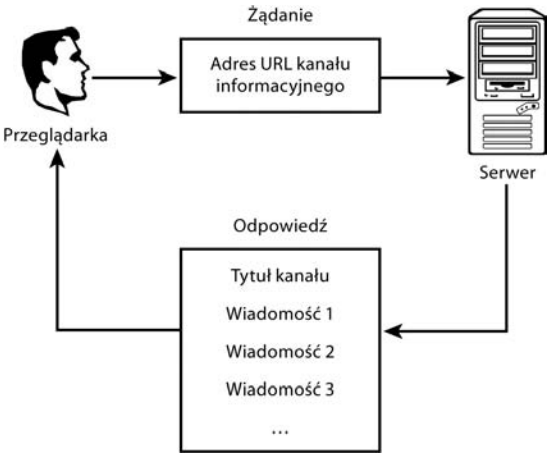
Z tego powodu w aplikacji *Czytnik wiadomości* jest wymagany skrypt po stronie serwera, ale jego jedynym zadaniem jest pobranie danych z zewnętrznego serwera i zwrócenie ich do klienta poprzez odpowiedź Ajaksa. Skrypt serwera możesz sobie wyobrazić jako pośrednika pomiędzy kodem JavaScript po stronie klienta a właściwym serwerem udostępniającym kanały informacyjne. Ponownie kluczowym powodem zastosowania takiego podejścia jest fakt, że kod Ajaksa może wykonywać żądania tylko do Twojego serwera, co stanowi poważny problem, kiedy potrzebujesz pobrać dane z innego miejsca. Rozwiązaniem jest umożliwienie skryptowi serwera pobrania danych i posłużenie się nimi jako podstawą odpowiedzi na żądania Ajaksa.

Zewnętrznymi (zdalnymi) serwerami, o których mówię, mogą być dowolne serwery udostępniające dane w formacie RSS. Jeśli trochę poszukasz, odkryjesz, że niemal każda strona internetowa udostępnia dane w tym formacie. Kiedy za

pomocą przeglądarki znajdziesz kanał informacyjny, zanotuj jego adres URL — wykorzystasz go potem w kodzie podczas wykonywania żądania Ajaksa po kanał RSS.

## Konwersacja pomiędzy klientem a serwerem

Jeśli połączysz wszystko, czego dowiedziałeś się o komunikacji odbywającej się w aplikacji *Czytnik wiadomości*, otrzymasz przepływ danych — taki, jak na rysunku 5.4.




**Rysunek 5.4.** *Aplikacja Czytnik wiadomości wykonuje żądanie zawierające adres URL kanału informacyjnego i odbiera dane XML zawierające ten kanał*

Danymi wysyłanymi do serwera jako parametr żądania Ajaksa jest adres URL kanału informacyjnego. Chociaż dane te są zawsze w formacie URL, mogą mieć różną postać. Na przykład adres URL kanału informacyjnego związanego z witryną internetową *Scientific American* wygląda tak: *http://www.sciam.com/xml/sciam.xml*.

Zauważ, że powyższy adres odnosi się do określonego dokumentu XML — *sciam.xml*. Ma to sens, ponieważ kanały RSS są właściwie dokumentami XML, ale adresy URL kanałów informacyjnych nie zawsze odnoszą się do plików XML. Na przykład adres URL kanału informacyjnego Wired News wygląda tak: *http://feeds.wired.com/wired/topheadlines*.

W powyższym przykładzie URL jest ścieżką — w przeciwieństwie do nazwy pliku. Wynika z tego, że nie jesteś w stanie przewidzieć dokładnej postaci adresu URL kanału komunikacyjnego. Z drugiej strony najczęściej wystarczy po prostu wykorzystanie adresu w takiej postaci, jaką udostępnia witryna internetowa.

Kiedy znasz już adres URL serwera kanału informacyjnego, musisz go umieścić w większym adresie URL, który będzie częścią żądania Ajaksa do skryptu serwera aplikacji *Czytnik wiadomości*. Poniżej znajduje się jego przykład:

```
newsfeeder.php?rssurl=http://www.sciam.com/xml/sciam.xml
```

Powyższy kod pokazuje, w jaki sposób adres URL kanału informacyjnego umieszczany jest jako pojedynczy parametr (`rssurl`) adresu URL. Cały adres URL jest wykorzystywany jako podstawa żądania Ajaksa, które zostanie wysłane do skryptu serwera *newsfeeder.php*. Z kolei skrypt serwera pobierze adres URL z parametru `rssurl` i wykorzysta go do wczytania informacji z kanału informacyjnego.

Teraz już wiesz, że skrypt serwera pobiera adres URL z parametru `rssurl` i następnie wykorzystuje go do pobrania dokumentu RSS. Dokument ten jest następnie zwracany do kodu JavaScript poprzez odpowiedź Ajaksa. Poniżej znajduje się przykład kodu RSS:

```
<?xml version="1.0" encodig="utf-8" ?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
<channel>
   <title>Scientific American - Official RSS Feed</title>
   <link>http://www.ciam.com/</link>
   <image>
      <url>http://www.sciam.com/media/logo/SAlogo_144px.gif</url>
      <title>Scientific American.com</title>
      <link>http://www.sciam.com/</link>
      <width>144</width>
      <height>45</height>
   </image>
   <description>Science news and technology updates from
      Scientific American</description>
   <language>en-us</language>
   <managingEditor>editors@sciam.com<managingEditor>
      <dc:publisher>Scirntific American</dc:publisher>
   <copyright>Copyright 1996-2007 Scientific American</copyright>
   <item>
      <title>News: Snake Bites the Toxic Toad That Feeds It-and
         Spreads Its Poison</title>
      <link>http://www.sciam.com/article.cfm?chanID=sa003&amp;
         articleID=6FD96438-E7F2-99DF-3C8A12ACF6BB25D8&amp;
         ref=rss</link>
      <guid>http://www.sciam.com/article.cfm?chanID=sa003&amp;
         articleID=6FD96438-E7F2-99DF-3C8A12ACF6BB25D8&amp;
         ref=rss</guid>
      <author>JR Minkel &lt;editors@sciam.com&gt;</author>
      <dc:date>2007-01-29T17:00:00-05:00</dc:date>
   </item>
   <item>
      <title>News: Did Honets Abe Have Nerves of Glass?</title>
      <link>http://www.sciam.com/article.cfm?chanID=sa003&amp;
         articleID=6FD6E124-E7F2-99DF-3E0EDA7A34E03A06&amp;
         ref=rss</link>
      <guid>http://www.sciam.com/article.cfm?chanID=sa003&amp;
         articleID=6FD6E124-E7F2-99DF-3E0EDA7A34E03A06&amp;
         ref=rss</guid>
```

```
      <author>Nikhil Swaminathal &lt;editors@sciam.com&gt;</author>
      <dc:date>2007-01-29T17:00:00-05:00</dc:date>
   </item>
...
</channel>
</rss>
```

Na pierwszy rzut oka kod wygląda na skomplikowany, ale do naszych celów możemy go w większej części zignorować. Pamiętaj o tym, że zadaniem aplikacji *Czytnik wiadomości* jest wyświetlenie listy wiadomości, które są łączami do pełnej treści. Możesz wyciągnąć te dane z dokumentu XML i zignorować resztę. Żeby zobaczyć, co mam na myśli, popatrz na kod i znajdź znacznik `<item>`. Teraz spójrz na kod znajdujący się poniżej tego znacznika i znajdź znaczniki `<title>`, `<link>` oraz `<dc:date>`. Kod wygląda tak, jak poniżej:

```
...
   <item>
      <title>News: Snake Bites the Toxic Toad That Feeds It-and
         Spreads Its Poison</title>
      <link>http://www.sciam.com/article.cfm?chanID=sa003&amp;
         articleID=6FD96438-E7F2-99DF-3C8A12ACF6BB25D8&amp;
         ref=rss</link>
...
      <dc:date>2007-01-29T17:00:00-05:00</dc:date>
   </item>
...
```

W znaczniku `<title>` znajduje się tytuł pojedynczej wiadomości, a w znaczniku `<link>` łącze do niej. Znacznik `<dc:date>` może być właściwie traktowany jako znacznik `<date>`, ponieważ `dc` jest przedrostkiem przestrzeni nazw, a nie częścią znacznika. Większość kanałów RSS do zakodowania daty publikacji zamiast znacznika `<date>` używa `<pubDate>`, więc podczas przetwarzania danych XML warto wziąć pod uwagę oba znaczniki. Należy również obsłużyć sytuację, w której nie ma żadnego z tych znaczników, ponieważ data publikacji jest w dokumencie RSS opcjonalna. Z powyższego omówienia wynika, że możesz bez większych problemów przeglądać kod RSS i pobrać z niego potrzebne znaczniki wraz z ich zawartością, żeby stworzyć listę wiadomości.

**UWAGA**

Powód, dla którego do zakodowania daty publikacji stosowane są dwa różne znaczniki, jest złożony i ma związek z konkurencyjnymi specyfikacjami i gwałtownym rozwojem formatu RSS. Nie będę tu przedstawiał nudnej i zbędnej historii RSS, powiem tylko, że najprawdopodobniej znacznie częściej spotkasz się ze znacznikiem `<pubDate>` niż `<date>` (`<dc:date>`). Warto jednak obsługiwać obydwa znaczniki, ponieważ `<date>` jest wykorzystywany przez kilka znanych stron.

Oczywiście będziesz również musiał pobrać nazwę kanału i być może nawet łącze do jego witryny internetowej. Informacje te także znajdują się w znacznikach <title> i <link>, ale tym razem pod znacznikiem <channel>. Kiedy ponownie popatrzymy na kod *Scientific American*, zobaczymy tytuł i łącze kanału informacyjnego:

```
...
<channel>
   <title>Scientific American - Official RSS Feed</title>
   <link>http://www.sciam.com/</link>
....
```

Podsumowując, z dokumentu RSS (XML) kanału wiadomości w aplikacji *Czytnik wiadomości* muszą zostać pobrane następujące informacje:

- tytuł kanału informacyjnego (<title> wewnątrz <channel>),
- łącze kanału informacyjnego (<link> wewnątrz <channel>),
- tytuły wiadomości (<title> wewnątrz <item>),
- łącza wiadomości (<link> wewnątrz <item>),
- daty publikacji wiadomości (<pubDate> lub <date> wewnątrz <item>).

Dwie pierwsze informacje występują w dokumencie kanału informacyjnego tylko raz, natomiast kolejne mogą pojawić się wielokrotnie, jeśli jest wiele wiadomości, a zazwyczaj tak jest. Każda informacja znajduje się w osobnym znaczniku <item>.

Chociaż mógłbym poświęcić jeszcze mnóstwo czasu na omówienie różnych elementów formatu danych RSS, nie zrobię tego — nie ma po prostu takiej potrzeby. Wiesz już wszystko, co trzeba, żeby zająć się implementacją aplikacji *Czytnik wiadomości*, więc do dzieła!

## Implementacja aplikacji Czytnik wiadomości

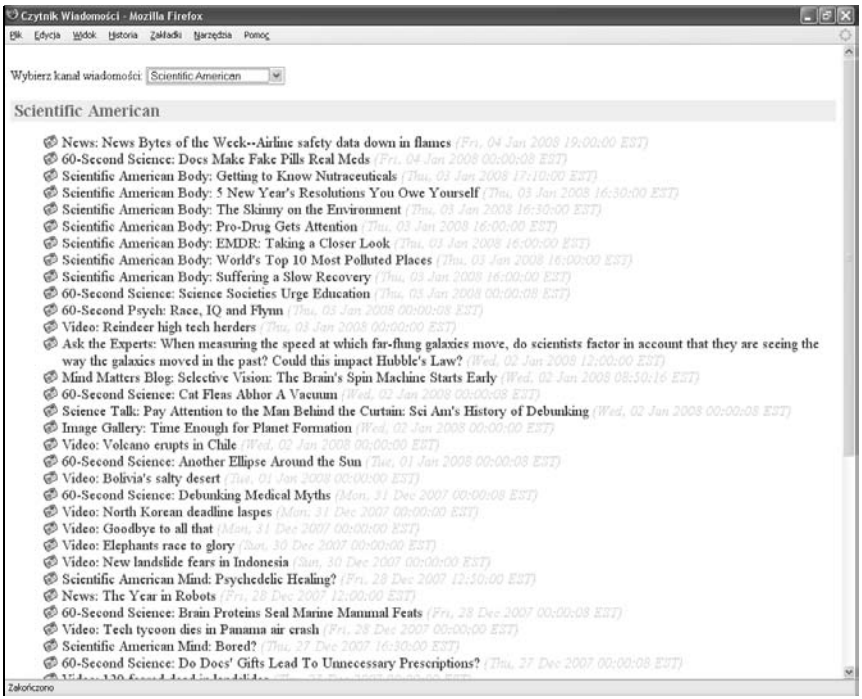
Prawdopodobnie jesteś przygotowany na stworzenie głównych części aplikacji *Czytnik wiadomości*: strony internetowej oraz skryptu serwera. Jest jednak jeszcze trzecia część układanki — arkusz stylów CSS. Nie jest on niezbędny, ale pozwala na nadanie wyświetlanym informacjom ładnego wyglądu. Poniżej znajdują się wymagane elementy aplikacji:

- strona HTML (*newsfeeder.html*),
- arkusz stylów CSS (*newsfeeder.css*),
- skrypt PHP serwera (*newsfeeder.php*).

Pamiętaj o tym, że w skład tych elementów wchodzą także zewnętrzne serwery wskazywane przez adresy URL poszczególnych kanałów informacyjnych. Serwery te nie są częścią aplikacji, to znaczy nie znajdują się na nich żadne pliki ani zasoby, którymi musisz się zajmować, ale bez względu na to pełnią kluczową rolę w funkcjonowaniu aplikacji. Dlatego też kiedy w następnych punktach będziesz pracował nad plikami aplikacji *Czytnik wiadomości*, pamiętaj, że są one uzależnione od zdalnych serwerów, z których pobierane są dane.

## Strona internetowa aplikacji Czytnik wiadomości

Strona aplikacji *Czytnik wiadomości* (*newsfeeder.html*) jest niewątpliwie najważniejszą jej częścią, ponieważ odpowiada za przetwarzanie danych RSS odesłanych przez skrypt serwera i wyświetlenie ich w czytelny sposób. Na rysunku 5.5 widać, że strona zawiera listę wyboru kanałów wiadomości, a pod nią element `div` zawierający łącze z tytułem wybranego kanału oraz listę łączy wiadomości.




**Rysunek 5.5.** *Strona aplikacji Czytnik wiadomości składa się z listy wyboru kanału oraz informacji związanych z aktualnie wybranym kanałem*

Kod HTML strony aplikacji *Czytnik wiadomości* z interfejsem użytkownika znajduje się w znaczniku `<body>`, który przedstawiono poniżej:

```
<body onload="loadFeed()">
   <div id="ajaxState" style="display:none"></div>
   <br />
   <div>Wybierz kanał wiadomości:
      <select id="feed" onchange="loadFeed()">
         <option value="http://feeds.wired.com/wired/topheadlines">
            Wired News</option>
        <option value="http://feeds.engadget.com/weblogsinc/engadget">
            Engadget</option>
        <option value="
            http://feeds.ziffdavis.com/ziffdavis/extremetech">
            ExtremeTech</option>
        <option value="
            http://feeds.feedburner.com/ajaxian">Ajaxian</option>
        <option value="http://www.sciam.com/xml/sciam.xml">
            Scientific American</option>
        <option value="http://news.google.com/?output=rss">Google
        ↪News</option>
        <option value="
            http://www.michaelmorrison.com/mambo/index2.php?
            option=com_rss&feed=RSS2.0&no_html=1">Blog Michaela
            ↪Morrisona</option>
      </select>
   </div>
   <br />
   <div id="feedcontent"></div>
</body>
```

Kiedy pominiesz listę wyboru stworzoną za pomocą znacznika `<select>`, zobaczysz, że kod jest bardzo mały. W rzeczywistości na stronie są tylko dwa widoczne elementy: lista wyboru (`feed`) oraz element `div` (`feedcontent`) wykorzystywany do wyświetlania wiadomości z dynamicznie wczytanych kanałów informacyjnych.

Fragmentem kodu, którego nie powinieneś przegapić w powyższej stronie, jest funkcja obsługi zdarzenia `onload` odpowiedzialna za wczytanie danych z pierwszego kanału na liście wyboru za pomocą funkcji `loadFeed()`. Zanim zajmiemy się tą funkcją, warto wskazać, że możesz dostosować listę kanałów w aplikacji *Czytnik wiadomości*, uzupełniając ją o własne kanały i nie zmieniając żadnego innego fragmentu kodu strony. Inaczej mówiąc, możesz całkowicie dostosować aplikację, nie patrząc na pozostały kod JavaScript — po prostu dodaj nowe znaczniki `<option>` lub zmodyfikuj istniejące, umieszczając w nich tytuły i adresy innych kanałów informacyjnych.

Jeśli chcesz zrozumieć, w jaki sposób działa aplikacja, oczywiście będziesz musiał zagłębić się w kod JavaScript. Poniżej znajduje się kod funkcji `loadFeed()` odpowiedzialnej za inicjalizację żądania Ajaksa wczytującego dane kanału informacyjnego:

```
function loadFeed() {
   // Wyczyść poprzednią zawartość kanału
   document.getElementById("feedcontent").innerHTML =
      "<img src='wait.gif' alt='Wczytywanie...' />";

   // Wyślij żądanie Ajaksa wczytujące nową zawartość kanału
   ajaxSendRequest("GET", "newsfeeder.php?rssurl=" +
      encodeURIComponent(document.getElementById('feed').options[
      document.getElementById('feed').selectedIndex].value),
      handleFeedRequest);
}
```

Pierwsza rzecz, którą robi funkcja `loadFeed()`, jest jedną z najbardziej interesujących funkcjonalności aplikacji — otóż wyświetla ona obrazek oczekiwania (*wait.gif*), który pojawia się podczas przetwarzania żądania. Obrazek był już przedstawiony na rysunku 5.3, ale kiedy zobaczysz go we właściwym kontekście, tak jak na rysunku 5.6, jego rola będzie bardziej oczywista.


**Rysunek 5.6.** *Animowany obrazek wczytywania jest wykorzystywany do wskazania użytkownikowi, że przetwarzane jest żądanie Ajaksa*

Powracając do kodu funkcji `loadFeed()`, obrazek wczytywania jest wyświetlany poprzez zmianę zawartości elementu `div` o identyfikatorze `feedcontent` i umieszczenie w nim znacznika `<img>`. Zmiana ta jest wykonywana poprzez ustawienie właściwości `innerHTML` elementu `div`. Możesz wykorzystać tę właściwość do dynamicznego zmieniania zawartości elementu na stronie. Wykorzystamy ją ponownie za chwilę w kodzie wyświetlającym tytuł kanału informacyjnego i wiadomości w tym samym elemencie `div`.

W ostatnim fragmencie kodu `loadFeed()` funkcja wysyła żądanie Ajaksa. Tworzony jest adres URL służący do komunikacji ze skryptem PHP serwera, zawierający parametr `rssurl`, w którym znajduje się adres URL kanału informacyjnego. Ustawiana jest tu również funkcja obsługi Ajaksa — `handleFeedRequest()`, wykonująca większość zadań w aplikacji.

**UWAGA**

W tym miejscu zdradzę Ci mały sekret — jest to pierwsze miejsce w książce, w którym do zmiany zawartości na stronie HTML wykorzystałem właściwość `innerHTML`. Czuję się trochę winny, ponieważ właściwość ta nie została oficjalnie zaakceptowana jako standard internetowy przez World Wide Web Consortium (W3C), co powoduje, że wykorzystanie jej jest wątpliwe. Jednak jest ona nieprawdopodobnie poręczna i wykorzystywana

w niezliczonych witrynach internetowych przez doświadczonych programistów, wiec istnieje małe prawdopodobieństwo, że w najbliższym czasie przestanie być obsługiwana. Alternatywą jest zmiana zawartości strony wyłącznie za pomocą interfejsu API DOM. Technika ta jest co prawda preferowana, ale w większości przypadków bardziej skomplikowana. Zdecydowałem się na prostotę i dlatego wykorzystałem właściwość `innerHTML`. Jeśli chcesz zapoznać się z dyskusją na temat różnic między `innerHTML` a interfejsem API DOM, wpisz w wyszukiwarce „innerHTML" i „DOM". Znajdziesz wiele skrajnych opinii!

Zanim zajmiemy się kodem funkcji `handleFeedRequest()`, jest jeszcze jeden fragment kodu warty uwagi. Mam na myśli kod znajdujący się tuż pod funkcją `loadFeed()`, który wczytuje obrazek *wait.gif*:

```
var waitImage = new Image();
waitImage.src = "wait.gif";
```

Zadaniem tego kodu jest to, żeby nie było widoczne opóźnienie podczas wczytywania obrazka. Rolą obrazka jest wskazanie użytkownikowi, że w tle coś się dzieje. Dlatego też wczytujemy go jeszcze przed wyświetleniem strony. Jest to możliwe dzięki temu, że kod wczytujący obrazek znajduje się poza jakąkolwiek funkcją JavaScript, co powoduje, że jest automatycznie wykonywany podczas wczytywania strony.

Powracając do funkcji `handleFeedRequest()`, jej zadaniem jest reagowanie na odpowiedź Ajaksa — pobranie zwróconych danych RSS i wyświetlenie wiadomości z kanału. Poniżej znajduje się kod tej funkcji:

```
function handleFeedRequest() {
   if (request.readyState == 4 && request.status == 200) {
      // Zapamiętaj dane z odpowiedzi XML
      var xmlData = request.responseXML;

      // Wygeneruj zawartość, zaczynając od tytułu kanału
      var feedContent = "";
      var channelElem = xmlData.getElementsByTagName("channel")[0];
      feedContent += "<div class='feedtitle'><a href='" +
         getText(channelElem.getElementsByTagName("link")[0]) +
         "'>" + getText(channelElem.getElementsByTagName("title")[0]) +
         "</a></div>";

      // Teraz wygeneruj nienumerowaną listę wiadomości
      feedContent += "<ul>";
      var feedItems = channelElem.getElementsByTagName("item");
      for (var i = 0; i < feedItems.length; i++) {
         var itemTitle =
          ↪getText(feedItems[i].getElementsByTagName("title")[0]);
         var itemLink = getText(feedItems[i].getElementsByTagName(
            "link")[0]);
```

```
            var itemPubDate =
             ↪getText(feedItems[i].getElementsByTagName("pubDate")[0]);
            if (itemPubDate == "")
               itemPubDate =
               ↪getText(feedItems[i].getElementsByTagName("date")[0]);
            if (itemPubDate == "")
               feedContent += "<li><a href='" + itemLink + "'>" +
                  itemTitle + "</a></li>";
            else
               feedContent += "<li><a href='" + itemLink + "'>" +
                  itemTitle + " <span class='itemdate'>(" +
                  itemPubDate + ")</span></a></li>";
         }
         feedContent += "</ul>";

         // Wyświetl zawartość kanału
         document.getElementById("feedcontent").innerHTML = feedContent;
      }
      ajaxUpdateState();
   }
```

Oczywiście jest tu sporo kodu, ale nie jest on specjalnie trudny. Pamiętaj o tym, że tak naprawdę służy on do stworzenia fragmentu HTML. Inaczej mówiąc, funkcja tworzy część strony internetowej, która może zostać umieszczona i wyświetlona w elemencie `div` z zawartością kanału informacyjnego. Wygenerowany HTML zawiera tytuł kanału informacyjnego będący jednocześnie łączem do strony, z której pochodzi. Kod HTML zawiera także nienumerowaną (wypunktowaną) listę wiadomości składających się z tytułu, który jest również łączem, oraz daty publikacji. Należy zauważyć, że wygenerowany kod HTML zawiera odwołania do klas arkusza stylów CSS, które umożliwiają zastosowanie odpowiedniego stylu dla danej zawartości.

Pod koniec kodu funkcji `handleFeedRequest()` możesz zobaczyć linię kodu umieszczającą dynamicznie wygenerowany kod HTML na stronie:

```
document.getElementById("feedcontent").innerHTML = feedContent;
```

Chociaż niewielka, ta pojedyncza linia robi bardzo dużo, biorąc pod uwagę, że jest odpowiedzialna za zamianę obrazka „wczytywanie" na sformatowany tytuł kanału informacyjnego oraz poszczególne wiadomości.

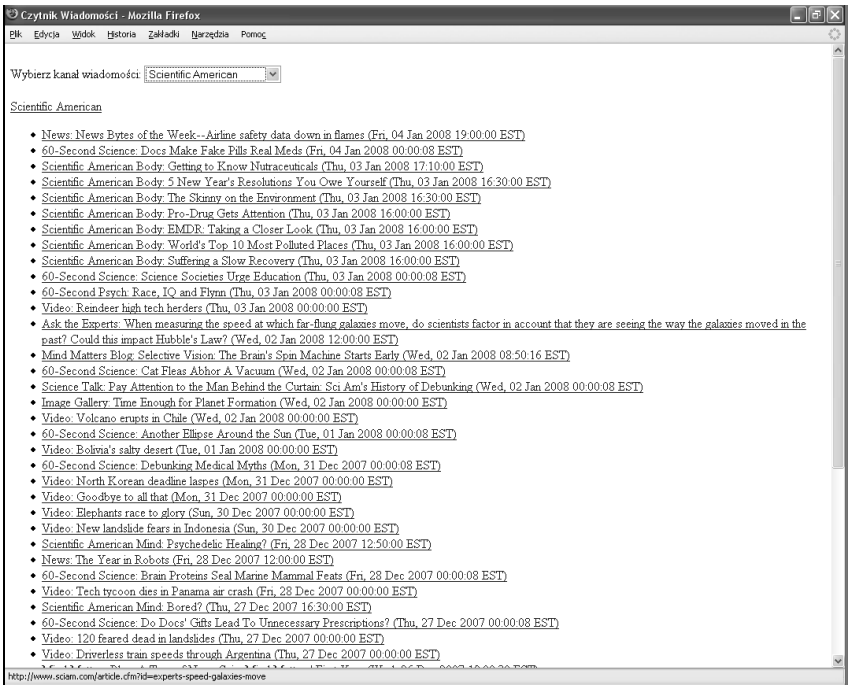
Wygląd strony aplikacji *Czytnik wiadomości* jest wynikiem zastosowania arkusza stylów *newsfeeder.css* dołączanego do strony za pomocą poniższego łącza:

```
<link rel="stylesheet" type="text/css" href="newsfeeder.css" />
```

Arkusz stylów *newsfeeder.css* znajduje się w tym samym katalogu — *chap05* — co reszta plików źródłowych aplikacji *Czytnik wiadomości*. W następnym punkcie omówię go krótko i przedstawię jego wpływ na ostateczny wygląd aplikacji.

## Arkusz stylów aplikacji Czytnik wiadomości

Przyznam się, że niechętnie zamieszczam kod arkusza stylów CSS w książce o Ajaksie. Jednak aplikacja *Czytnik wiadomości* jest znakomitym przykładem tego, jak bardzo można zmienić wygląd aplikacji za pomocą stosunkowo niewielkiej ilości kodu. Jeśli mi nie wierzysz, spójrz na rysunek 5.7 przedstawiający aplikację *Czytnik wiadomości* bez zastosowania arkusza stylów.




**Rysunek 5.7.** *Po usunięciu arkusza stylów aplikacja Czytnik wiadomości wygląda ubogo*

Porównaj ten rysunek z rysunkiem 5.5 przedstawiającym ten sam kanał informacyjny, ale z zastosowaniem arkusza stylów. Moim celem jest pokazanie tego, że chociaż CSS ma niewiele wspólnego z Ajaksem, jednak można za jego pomocą nadać aplikacji bardziej elegancki i profesjonalny wygląd.

Poniżej znajduje się kod CSS odpowiedzialny za style kanału wiadomości w aplikacji *Czytnik wiadomości*:

```
div.feedtitle {
   padding-left : 5px;
   background-color : #EEEEEE;
   font-size : 22px;
   font-weight : bold;
}

div.feedtitle a {
   color : #666677;
```

```
   text-decoration : none;
}

div.feedtitle a:hover {
   color : #333344;
   text-decoration : none;
}
```

Tytuł kanału informacyjnego jest wygenerowanym kodem HTML z atrybutem `class` o wartości `feedtitle`, który łączy go z grupą stylów o tej nazwie. Jako pierwsze style ustawione są lewostronne wyrównanie, kolor tła, rozmiar i grubość czcionki. Następnie ustawiany jest kolor tekstu, który zmienia się, kiedy użytkownik najedzie kursorem myszy na tytuł (który, jak sobie przypominasz, jest łączem). Styl `text-decoration` jest wykorzystywany do nadpisania domyślnego ustawienia HTML powodującego podkreślanie łączy, kiedy użytkownik umieści nad nimi kursor myszy — w tym przypadku czytelniej i bardziej profesjonalnie będzie wyglądać zmiana koloru.

Podobne style CSS zastosowane są do poszczególnych wiadomości wyświetlanych w wypunktowanej liście:

```
li {
   font-size : 18px;
   padding-left : 25px;
   background : url(newspaper.gif) no-repeat;
   list-style : none;
}
```

Powyższy kod jest odpowiedzialny za wyświetlanie ładnej małej ikonki gazety obok każdego elementu listy (patrz rysunek 5.5). Każdy element jest również łączem, dlatego też zastosowana jest podobna koncepcja zmiany koloru po najechaniu kursorem myszy jak w przypadku tytułu kanału. Poniżej znajduje się kod CSS, który realizuje to zadanie:

```
a:link {
   color : #333344;
   text-decoration : none;
}

a:visited {
   color : #9999AA;
   text-decoration : none;
}

a:hover, a:active {
   background : #EEEEEE;
   text-decoration : none;
}
```

W końcu możesz zauważyć, że data publikacji każdej wiadomości jest wyświetlana kursywą i w jasnym kolorze, dzięki czemu tytuł jest lepiej widoczny. Poniżej znajduje się styl CSS odpowiedzialny za wygląd daty publikacji poszczególnych wiadomości:

```
span.itemdate {
   font-style : italic;
   color : #CCCCDD;
}
```

To już wszystko na temat arkuszy stylów w aplikacji *Czytnik wiadomości*. Pozostał jeszcze skrypt PHP serwera i aplikacja będzie gotowa.

## Skrypt serwera aplikacji Czytnik wiadomości

Widziałeś kiedyś sztafetę i miejsce, w którym jeden biegacz przekazuje pałeczkę drugiemu? To prawie to samo, co robi skrypt serwera *newsfeeder.php* w aplikacji *Czytnik wiadomości*. Po uruchomieniu pobiera dane kanału RSS ze zdalnego serwera i następnie przekazuje je do kodu JavaScript po stronie klienta. Informacja, skąd mają zostać pobrane dane, jest przekazywana za pomocą parametru `rssurl` umieszczonego w żądaniu Ajaksa. Poniżej znajduje się kod skryptu PHP:

```
<?php
header('Content-type: text/xml;');

// Zwróć dane kanału RSS
echo file_get_contents($_REQUEST['rssurl']);
?>
```

**UWAGA**

Pamiętaj, że zadaniem tego pośredniczącego skryptu serwera jest obejście ograniczenia uniemożliwiającego wykonywanie żądań Ajaksa do zdalnych serwerów. Gdyby takie ograniczenie nie istniało, skrypt serwera nie byłby potrzebny i klient wykonywałby po prostu żądanie bezpośrednio do serwera udostępniającego dane RSS. Jednak istnieje ważna przyczyna wprowadzenia takiego ograniczenia: bezpieczeństwo! Umożliwienie skryptom klienta wykonywania żądań do zewnętrznych serwerów byłoby zbyt niebezpieczne, dlatego też konieczność stworzenia własnego skryptu pośredniczącego w żądaniach Ajaksa jest niską ceną, jaką trzeba zapłacić za bezpieczny internet.

Po ustawieniu nagłówka zapewniającego, że odpowiedź będzie traktowana jako dokument XML, w skrypcie pozostaje już tylko jedna linijka kodu. Funkcja `file_get_contents()` pobiera dane z określonego adresu URL, czyli robi dokładnie to, czego potrzebujesz. Zatem wszystkim, co robi ten skrypt, jest przekazanie adresu URL do funkcji i następnie wysłanie jej wyniku do klienta. Końcowym

rezultatem jest sformatowany dokument kanału RSS odsyłany do klienta w odpowiedzi Ajaksa.

## Testowanie aplikacji Czytnik wiadomości

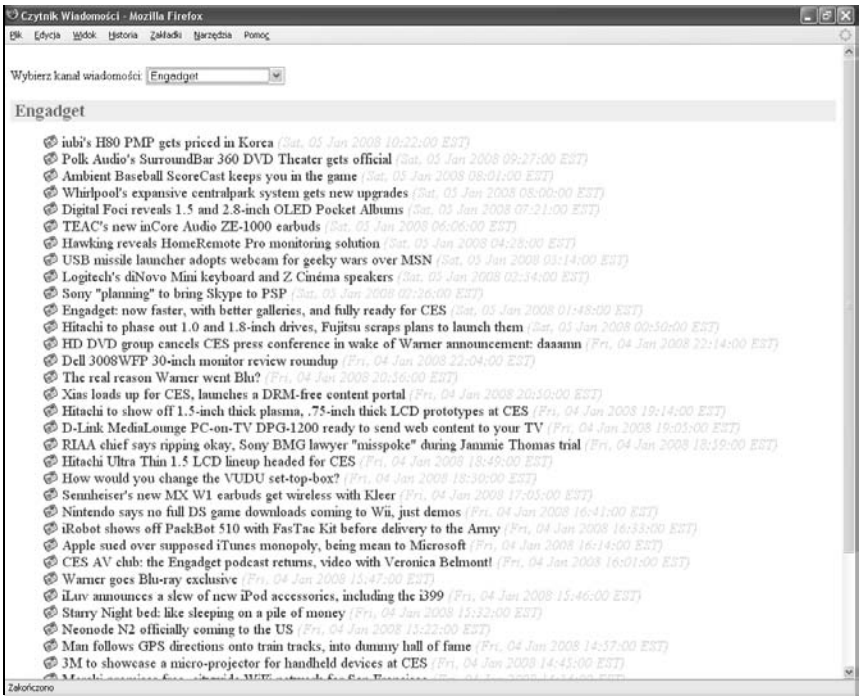
Kiedy po raz pierwszy otworzysz stronę czytnika wiadomości, zostanie wczytany pierwszy kanał na liście wyboru, którym są akurat najnowsze informacje z Wired News. Na rysunku 5.8 widać aplikację, w której wyświetlany jest ten kanał.


**Rysunek 5.8.** *Po otwarciu aplikacji Czytnik wiadomości wczytywany jest pierwszy kanał informacyjny na liście wyboru*

Po wybraniu innego kanału za pomocą listy wyboru na czas wykonywania żądania Ajaksa wyświetlany jest obrazek „wczytywanie". Na rysunku 5.9 widać zmianę kanału z Wired News na Engadget.


**Rysunek 5.9.** *Po zmianie kanału wiadomości wyświetlany jest obrazek „wczytywanie" wskazujący, że w tle wykonywane są jakieś zadania*

Kanały wiadomości nie mają specjalnie dużego rozmiaru, więc ich wyświetlenie nie powinno zająć dużo czasu. Wyświetlane wiadomości zastępują na stronie obrazek „wczytywanie”, co widać na rysunku 5.10.




**Rysunek 5.10.** *Po zakończeniu wczytywania kanału wiadomości z niego zastępują na stronie obrazek „wczytywanie”*

Należy zauważyć, że właśnie stworzyłeś zaskakująco poręczny, niewielki czytnik wiadomości. Mam na myśli to, że po poznaniu sposobu działania Ajaksa w aplikacji *Czytnik wiadomości* możesz ją wykorzystać do przeglądania wiadomości. Kliknij po prostu element na liście, żeby przejść na stronę pełnej wiadomości. W tym przypadku kliknięcie przeniesie Cię na stronę wiadomości witryny Engadget Web.

Oprócz interesujących wiadomości o gadżetach strona Engadget Web jest także świetnym przykładem tego, jak łatwo wykorzystać kanały RSS do publikowania informacji z witryny. Popatrz na górny prawy róg strony Engadget Web, gdzie znajduje się standardowa pomarańczowa ikonka RSS. Kliknij ją, żeby uzyskać adres URL kanału informacyjnego, który możesz dodać do *Czytnika wiadomości*.

## Przerabianie aplikacji Czytnik wiadomości

Żadna aplikacja w tej książce nie jest tak naprawdę ukończona. Mówiąc to, mam na myśli, że przykładowe aplikacje są tylko punktem, od którego możesz zacząć eksperymentowanie z Ajaksem. *Czytnik wiadomości* nie stanowi wyjątku i w rzeczywistości jest znakomitym przykładem aplikacji, którą możesz dostosować do własnych potrzeb.

Być może najbardziej oczywistym sposobem przerobienia aplikacji *Czytnik wiadomości* jest zmiana samych kanałów wiadomości. Jest to także najprostsza zmiana, ponieważ wymaga tylko modyfikacji znaczników `<option>` na stronie — zmiany tytułów i wartości, tak żeby odpowiadały innym kanałom. Pamiętaj o tym, że możesz dodać dowolną liczbę znaczników `<option>`, tworząc listę wyboru z wieloma różnymi kanałami wiadomości. Musisz tylko umieścić nazwę kanału pomiędzy znacznikami `<option>` i `</option>`, a następnie przypisać adres URL do atrybutu `value`.

Możesz również zdecydować, że możliwość wyboru kanałów wiadomości z listy nie jest tym, czego chcesz, i w takim przypadku po prostu osadzić pojedynczy kanał w swojej stronie. Żeby to zrobić, musisz wziąć odpowiedni fragment kodu ze strony *newsfeeder.html* i umieścić go we własnej witrynie. Będzie się to wiązać głównie z przekopiowaniem kodu z sekcji `head` oraz z przeniesieniem funkcji obsługi zdarzenia `onload` znajdującej się w znaczniku `<body>`, która wywołuje funkcję `loadFeed()`. Musisz także stworzyć element `div` o identyfikatorze `feedcontent`. Na końcu będziesz jeszcze musiał przerobić funkcję `loadFeed()`, tak żeby wykorzystywała wybrany przez Ciebie adres URL kanału wiadomości zamiast pobierania go z listy wyboru.

Jeśli zdecydujesz się na osadzenie kanału informacyjnego na stronie, na której jest już jakaś zawartość, prawdopodobnie będziesz chciał ograniczyć liczbę wyświetlanych wiadomości. Na przykład możesz chcieć wyświetlać tylko pięć pierwszych wiadomości. Żeby to zrobić, musisz zmienić pętlę `for` w funkcji `handleFeedRequest()` tak, jak poniżej:

```
for (var i = 0; (i < feedItems.length) && (i < numItems);i++) {
```

W powyższym kodzie zmienna `numItems` wskazuje, ile wiadomości ma zostać wyświetlonych. Możesz użyć zmiennej lub na sztywno zakodować tę liczbę:

```
for (var i = 0; (i < feedItems.length) && (i < 10);i++) {
```

W powyższym przykładzie liczba wyświetlonych wiadomości jest ograniczona do dziesięciu. Pętla `for` w dalszym ciągu sprawdza długość tablicy `feedItems`, ponieważ może się zdarzyć, że liczba wiadomości w kanale będzie mniejsza niż dziesięć, a w takim przypadku zostaną wyświetlone wszystkie dostępne wiadomości.

Ostatnim zadaniem, które możesz wziąć pod uwagę, kiedy masz już wiedzę na temat RSS, jest stworzenie własnego kanału. Jeśli masz bloga lub stronę internetową z regularnie zmienianą zawartością, udostępnienie użytkownikom kanału informacyjnego na Twojej stronie jest bliską przyszłością. Mam nadzieję, że dowiedziałeś się w tym rozdziale wystarczająco dużo na temat formatu RSS, żeby spróbować stworzyć własne dokumenty RSS. Tak naprawdę najlepszym rozwiązaniem jest stworzenie skryptu, który automatycznie będzie konwertował zawartość HTML na format RSS, dzięki czemu Twoja witryna będzie automatycznie publikowana w postaci kanałów wiadomości. Sposób realizacji tego rozwiązania wykracza nieco poza zakres tego rozdziału, ale masz już dobre narzędzie do testowania swoich kanałów wiadomości, kiedy już je stworzysz.

## Podsumowanie

W tym rozdziale wprowadziłem Cię w interesujący obszar zarządzania zawartością internetową znany jako kanały wiadomości, który wykorzystuje standardowy, oparty na XML-u, format RSS (ang. *Really Simple Syndication*). Za pomocą RSS można pobrać skrót zawartości witryny internetowej bez potrzeby jej odwiedzania. Co więcej, można pobrać zawartość z innej witryny i udostępnić ją na własnych stronach. Jest to dobra metoda aktualizowania własnych stron bez konieczności wprowadzania zmian we własnej zawartości. Stworzona przez Ciebie aplikacja *Czytnik wiadomości* pozwoliła zademonstrować, w jaki sposób za pomocą Ajaksa można dynamicznie wczytywać dane RSS i czytelnie prezentować je na stronie, dzięki czemu możliwe jest czytanie i nawigowanie do wiadomości. Aplikacja ta zawiera podstawowy kod, dzięki któremu możesz bez większego wysiłku umieścić kanały wiadomości RSS na własnych stronach.