



Technologia i rozwiązania

# AJAX i PHP

**Tworzenie interaktywnych  
aplikacji internetowych**

- Jak tworzyć szybsze i sprawniejsze aplikacje internetowe?
- Jak pracować z biblioteką jQuery?
- Jak wprowadzać nowe rozwiązania do już istniejących stron?

**Wydanie II**

**Helion**



Bogdan Brinzarea-Iamandi,  
Cristian Darie, Audra Hendrix

**[PACKT]**  
PUBLISHING

## » Idź do

- Spis treści
- Przykładowy rozdział

## » Katalog książek

- Katalog online
- Zamów drukowany katalog

## » Twój koszyk

- Dodaj do koszyka

## » Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

## » Czytelnia

- Fragmenty książek online

## » Kontakt

Helion SA  
ul. Kościuszki 1c  
44-100 Gliwice  
tel. 32 230 98 63  
e-mail: helion@helion.pl  
© Helion 1991-2010

## AJAX i PHP. Tworzenie interaktywnych aplikacji internetowych. Wydanie II

Autorzy: [Bogdan Brinzarea](#), [Cristian Darie](#)

Tłumaczenie: Julia Szajkowska

ISBN: 978-83-246-2768-4

Tytuł oryginału: [AJAX and PHP: Building Modern Web Applications 2nd Edition](#)

Format: B5, stron: 304



- Jak tworzyć szybsze i sprawniejsze aplikacje internetowe?
- Jak pracować z biblioteką jQuery?
- Jak wprowadzać nowe rozwiązania do już istniejących stron?

Wprowadzenie technologii AJAX pozwoliło na tworzenie bardziej atrakcyjnych i przyjaznych użytkownikowi witryn, które nie wymagają przeładowywania po każdej interakcji. To zapewniło AJAX-owi ogromną popularność wśród webmasterów. Otrzymali bowiem doskonałe narzędzie do projektowania interaktywnych i dynamicznych aplikacji, tak pożądanym w dobie WEB 2.0. Jednak nic nie jest doskonałe i nawet AJAX, mimo wielkiego potencjału, ma swoje słabości i ograniczenia. Jakież? W trakcie lektury książki poznasz plusy i minusy tej technologii!

Intencją autorów było przede wszystkim przekazanie wiedzy niezbędnej do opanowania sztuki tworzenia interaktywnych aplikacji, wykorzystujących PHP, JavaScript, MySQL i jQuery. Dowiesz się stąd również, jak przeprowadzać weryfikację danych wprowadzanych na stronie za pomocą technologii AJAX i jak łączyć ze sobą funkcje programu występujące po stronie serwera z tymi, które pojawiają się po stronie klienta. Poznasz skuteczne metody debugowania kodu. Ponadto na kilku rozbudowanych przykładach nauczysz się sprawnie pracować z biblioteką jQuery. Autorzy pokażą Ci, jak unikać najczęstszych błędów, tworzyć wydajny kod AJAX z myślą o pozycjonowaniu witryny oraz w prosty sposób wprowadzać nowe rozwiązania, także do istniejących już stron internetowych.

- Przygotowanie środowiska pracy
- Wprowadzenie w świat zagadnień technologii AJAX
- JavaScript i klient AJAX
- Programowanie obiektowe w JavaScript
- Skrypty PHP i używanie MySQL po stronie serwera
- Weryfikacja poprawności wprowadzanych danych za pomocą AJAX
- Debugowanie i profilowanie aplikacji AJAX
- Zaawansowane metody budowania aplikacji internetowych
- Arkusze danych w technologii AJAX

**Opanuj sztukę tworzenia aplikacji WEB 2.0!**

# Spis treści

<b>O autorach</b>	<b>7</b>
<b>O recenzencie</b>	<b>9</b>
<b>Wprowadzenie</b>	<b>11</b>
<b>Rozdział 1. Świat technologii AJAX i języka PHP</b>	<b>17</b>
<b>Ogólny zarys</b>	<b>18</b>
Technologia AJAX a Web 2.0	19
<b>Strony internetowe od 1990 roku</b>	<b>20</b>
Protokół HTTP i język HTML	20
PHP i inne technologie strony serwera	22
JavaScript i inne technologie strony klienta	22
Czego zatem brakuje?	24
<b>Świat technologii AJAX</b>	<b>24</b>
Co składa się na narzędzia AJAX?	27
Kiedy warto używać technologii AJAX, a kiedy należy z niej zrezygnować?	28
Narzędzia i źródła	29
<b>Przygotowanie środowiska pracy</b>	<b>30</b>
<b>Prosta aplikacja wykorzystująca AJAX i PHP</b>	<b>31</b>
<b>Podsumowanie</b>	<b>43</b>
<b>Rozdział 2. JavaScript i klient AJAX</b>	<b>45</b>
<b>JavaScript a obiektowy model dokumentu</b>	<b>45</b>
<b>Zdarzenia w języku JavaScript i model DOM</b>	<b>51</b>
<b>I znowu model DOM</b>	<b>55</b>
<b>JavaScript, model DOM i arkusze stylów CSS</b>	<b>59</b>
<b>Używanie obiektów klasy XMLHttpRequest</b>	<b>63</b>
Tworzenie obiektu klasy XMLHttpRequest	63
Obsługa wyjątków w języku JavaScript	64
Tworzenie lepszych obiektów dla przeglądarki Internet Explorer 6	66
Inicjowanie żądania za pomocą obiektu klasy XMLHttpRequest	68
Obsługa odpowiedzi przysyłanych z serwera	70

<b>Praca z dokumentami XML</b>	<b>78</b>
Więcej na temat obsługi błędów i zwracania wyjątków	84
Tworzenie struktury pliku XML	85
<b>Podsumowanie</b>	<b>86</b>
<b>Rozdział 3. Obiektowy JavaScript</b>	<b>87</b>
<b>Dlaczego język JavaScript ma tak duże znaczenie?</b>	<b>88</b>
<b>Idea programowania obiektowego</b>	<b>88</b>
Hermetyzacja	89
Dziedziczenie	90
Polimorfizm	91
<b>Programowanie obiektowe w języku JavaScript</b>	<b>91</b>
W języku JavaScript obiekty są słownikami	92
Funkcje w języku JavaScript	94
Funkcje JavaScript jako obiekty pierwszej klasy	95
Funkcje wewnętrzne	96
Domknięcia	97
Klasy w języku JavaScript	98
Konstruktory	98
Diagramy klas	100
Odwołania do funkcji zewnętrznych	102
Prototypy	103
Właściwości i metody instancji	104
Metody i właściwości statyczne	105
Prywatni uczestnicy klasy	106
Kontekst wykonania w języku JavaScript	107
Kiedy var x, kiedy this.x, a kiedy x?	109
Praca we właściwym kontekście	110
<b>Praktyczne zagadnienia programowania obiektowego w JavaScript — wstęp do notacji JSON</b>	<b>112</b>
Idea formatu JSON	113
Prosty przykład pracy z danymi w formacie JSON	114
<b>Podsumowanie</b>	<b>117</b>
<b>Rozdział 4. Skrypty PHP i używanie MySQL po stronie serwera</b>	<b>119</b>
<b>PHP, DOM i XML</b>	<b>120</b>
<b>Język PHP i format JSON</b>	<b>125</b>
<b>Przekazywanie zmiennych i obsługa błędów w języku PHP</b>	<b>129</b>
<b>Praca z bazą MySQL</b>	<b>139</b>
Tworzenie tabel w bazie danych	139
Przetwarzanie danych	142
Łączenie się z bazą danych i wykonywanie zapytań	143
<b>Podsumowanie</b>	<b>149</b>
<b>Rozdział 5. Weryfikacja poprawności wprowadzanych danych za pomocą AJAX</b>	<b>151</b>
<b>Implementacja weryfikacji poprawności danych z zastosowaniem technologii AJAX</b>	<b>152</b>
<b>Obiekt klasy XMLHttpRequest, wersja 2.</b>	<b>156</b>
<b>Weryfikacja danych z wykorzystaniem możliwości technologii AJAX</b>	<b>164</b>
<b>Podsumowanie</b>	<b>185</b>

<b>Rozdział 6. Debugowanie i profilowanie aplikacji AJAX</b>	<b>187</b>
<b>Debugowanie i profilowanie kodu w przeglądarce Internet Explorer</b>	<b>188</b>
Uruchamianie debugowania w przeglądarkach Internet Explorer 6 i Internet Explorer 7	188
Debugowanie kodu w przeglądarce Internet Explorer 8	189
Inne narzędzia debugujące w przeglądarce Internet Explorer	196
<b>Debugowanie i profilowanie kodu w przeglądarce Firefox</b>	<b>197</b>
Dodatek Firebug	198
Dodatek Venkman JavaScript Debugger	200
Dodatek Web Developer	201
<b>Podsumowanie</b>	<b>202</b>
<b>Rozdział 7. Zaawansowane rozwiązania i metody budowania aplikacji internetowych</b>	<b>203</b>
<b>Pozyskiwanie przewidujące</b>	<b>206</b>
<b>Wskaźnik postępu</b>	<b>207</b>
<b>Nieinwazyjne kodowanie JavaScript</b>	<b>208</b>
<b>Progresywne ulepszanie i eleganckie przemijanie</b>	<b>210</b>
<b>Asynchroniczne wysyłanie plików za pomocą aplikacji AJAX</b>	<b>211</b>
Wysyłanie plików za pomocą protokołu HTTP	212
Asynchroniczne wysyłanie plików z użyciem znacznika iframe i rozwiązań technologii AJAX	212
<b>Wywołania międzydomenowe</b>	<b>218</b>
Realizacja wywołań międzydomenowych za pomocą serwera proxy	219
Realizacja wywołań międzydomenowych za pomocą aplikacji Flash	219
Realizacja wywołań międzydomenowych za pomocą znacznika <iframe>	220
Realizacja wywołań międzydomenowych za pomocą obiektów JSONP	220
<b>Atak typu cross-site request forgery</b>	<b>221</b>
Przejmowanie kontroli za pomocą obiektów JSON	222
Zmniejszenie ryzyka zaistnienia ataku CSRF	222
<b>Ataki typu cross-site scripting</b>	<b>223</b>
Ataki przeprowadzane za pomocą kodu wykorzystującego luki w zabezpieczeniach (ang. exploits)	223
Nietrwały atak typu XSS	223
Trwały atak typu XSS	224
Unikanie ataków typu XSS	224
Weryfikacja danych wejściowych	224
Zmiana zestawu znaków	225
Zabezpieczanie plików cookie	225
<b>Podsumowanie</b>	<b>226</b>
<b>Rozdział 8. Czat bazujący na AJAX i jQuery</b>	<b>227</b>
<b>Czatuj z AJAX</b>	<b>227</b>
<b>Szkielet jQuery</b>	<b>228</b>
Zanim zaczniemy	229
Pierwsze kroki	229
Selektory obiektów modelu DOM w szkielecie jQuery	230
Obiekt osłony szkieletu jQuery	230

Łącuchowanie metod	231
Obsługa wyjątków	231
Prosty przykład	232
Podstawowe idee	233
<b>Czat w technologii AJAX</b>	<b>234</b>
Aplikacja czatu	235
<b>Podsumowanie</b>	<b>261</b>
<b>Rozdział 9. Arkusz danych w technologii AJAX</b>	<b>263</b>
<hr/>	
<b>Implementacja kodu arkusza danych AJAX</b>	<b>265</b>
Analiza kodu	266
Baza danych	266
Style i kolory	267
Po stronie serwera	269
Budowanie arkusza danych krok po kroku	270
<b>Podsumowanie</b>	<b>278</b>
<b>Dodatek. Przygotowanie środowiska pracy</b>	<b>279</b>
<hr/>	
<b>Instalacja pakietu XAMPP</b>	<b>280</b>
Instalacja pakietu XAMPP w systemie Windows	280
Instalacja pakietu XAMPP w systemie Linux	283
<b>Przygotowanie bazy danych ajax</b>	<b>283</b>
<b>Skorowidz</b>	<b>287</b>
<hr/>	

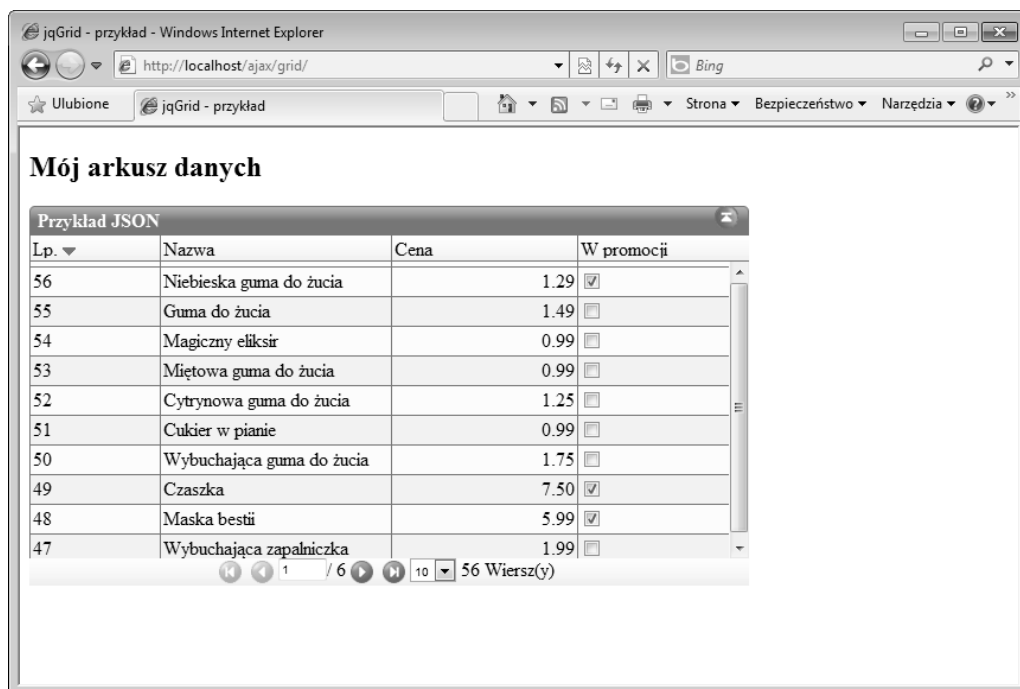
# Arkusz danych w technologii AJAX

Najbardziej powszechną metodą przedstawiania danych jest wykorzystanie formy arkusza danych. W arkuszu można prezentować dane wszelkiego rodzaju — od zawartości książki adresowej do zestawienia danych logistycznych. Ponieważ firmy potrzebowały możliwości przechowywania informacji w scentralizowanych archiwach, nie trzeba było długo czekać na pojawienie się pierwszych aplikacji pozwalających zapisywać dane w internecie czy intranecie w postaci arkusza. Niestety, w porównaniu do swoich stacjonarnych odpowiedników aplikacje te były bardzo mocno okrojone — praca z nimi wymagała ogromnego wysiłku i dużych nakładów czasu. Dodatkowym problemem była implementacja rozwiązania (szczególnie jeśli w grę wchodziła kontrola różnych poziomów dostępu na różnych serwerach), a opóźnienia wynikające z potrzeby odświeżania zawartości strony, związane niejednokrotnie z wykonywaniem najprostszych operacji sortowania czy edytowania, sprawiały, że praca z dostępnymi w sieci arkuszami danych była wysoce niewygodna i mocno obciążająca dla maszyn.

Jesteś bystrym czytelnikiem, więc na pewno domyśliłeś się już, że aktualizację zawartości arkusza można przeprowadzić za pomocą narzędzi dostępnych w technologii AJAX. Już za chwilę pokażemy Ci, jak to zrobić! Tak zaprojektowana aplikacja będzie odświeżała zawartość arkusza bez potrzeby ponownego otwierania strony, będzie potrafiła przechowywać dane po stronie klienta (zamiast wysyłać je za każdym razem na serwer) i będzie umożliwiała zmianę swojego wyglądu w kilku uderzeniach w klawiaturę! Pora pożegnać na zawsze migające ekrany niepełnych zestawień i sesje wygasające tuż przed ukończeniem pracy. Życzymy Ci miłej zabawy!

W tym rozdziale wykorzystamy dodatek szkieletu jQuery o nazwie jqGrid. Dodatek ten jest dostępny za darmo do użytku prywatnego i komercyjnego (choć autorzy nie wzgardzą dobrowolnym wsparciem finansowym). Można pobrać go ze strony <http://www.trirand.com/blog/>. Zapewne domyśliłeś się, że wszystkie operacje po stronie serwera będą realizowane za pomocą skryptu PHP, ale pamiętaj, że dodatek jqGrid potrafi współpracować również z innymi językami skryptowymi. Za pracę arkusza po stronie klienta będą odpowiadały biblioteka jQuery JavaScript i technologia JSON. Wygląd arkusza zdefiniujemy w odpowiednim arkuszu stylów CSS, wykorzystując w tym celu motywy. W ten sposób jego zmiana nie będzie wymagała wiele zachodu. Zacznijmy od zapoznania się z możliwościami dodatku jqGrid. Już wkrótce przekonasz się, że nowo nabyte umiejętności pracy w technologii AJAX pozwolą Ci szybko wykorzystywać możliwości tego narzędzia przy tworzeniu dowolnego serwisu WWW.

Ukończony arkusz będzie wyglądał tak, jak przedstawia to rysunek 9.1.



Rysunek 9.1. Arkusz danych w technologii AJAX zbudowany na szkielecie jQuery

Przyjrzyjmy się kodowi odpowiedzialnemu za utworzenie funkcji arkusza i zabierajmy się do pracy.

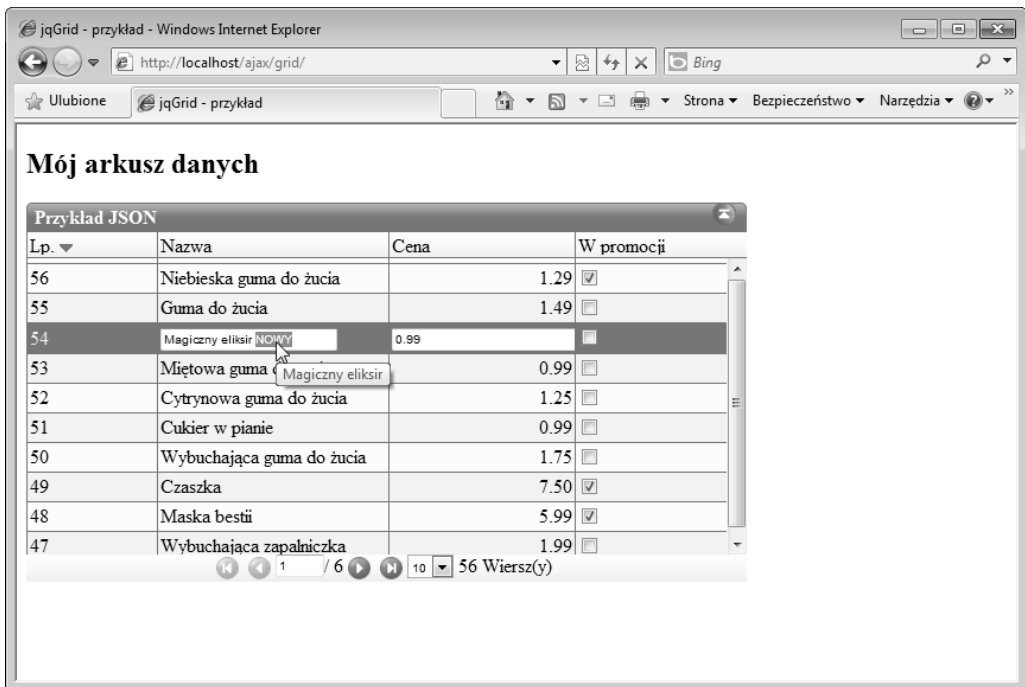


# Implementacja kodu arkusza danych AJAX

Pliki i katalogi niezbędne do omówienia tego przykładu zostały udostępnione w archiwum wszystkich przykładów z książki, ale możesz też przepisać kod osobiście.

Zachęcamy Cię do pobrania przykładowej aplikacji z internetu, ponieważ jest to szybsze i bezpieczniejsze (unikniesz w ten sposób błędów popełnianych podczas przepisywania kodu). Jeżeli zdecydujesz się na to rozwiązanie, wystarczy, że wykonasz następujące kroki:

1. Skopiuj katalog *grid* z archiwum do katalogu *ajax* na serwerze.
2. Połącz się z bazą danych i wykonaj w niej kod SQL zawarty w pliku *product.sql*.
3. Wprowadź do pliku *config.php* dane użytkownika bazy i hasło.
4. Otwórz w przeglądarce adres <http://localhost/ajax/grid>. Efekt powinien wyglądać tak jak na rysunku 9.1.
5. Możesz też sprawdzić od razu funkcję edytowania zawartości. Kliknij wybraną komórkę, wprowadź zmiany i zaakceptuj je klawiszem *Enter*. Rysunek 9.2 pokazuje wygląd aplikacji w trybie edycji danych.



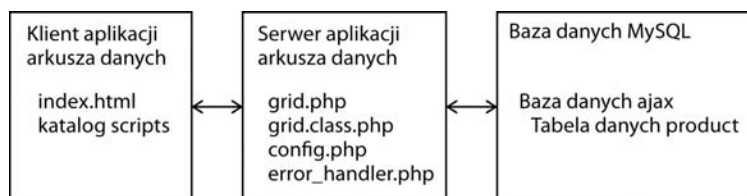
Rysunek 9.2. Edycja wiersza

## Analiza kodu

Jeżeli wolisz własnoręcznie zapisać kod całej aplikacji, znajdziesz go w dalszej części tego rozdziału. Na razie zajmiemy się szybkim przeglądem plików tworzących arkusz danych. Szczegółowe omówienie ich zawartości znajdziesz na końcu rozdziału.

Arkusz danych jest zbudowany z kodu zawartego w kilku plikach.

- Skrypt odpowiedzialny za utworzenie bazy danych arkusza znajduje się w pliku *product.sql*.
- Pliki *config.php* i *error\_handler.php* to nasze standardowe skrypty pomocnicze.
- Działania po stronie serwera są realizowane za pomocą skryptów zapisanych w plikach *grid.php* i *grid.class.php*.
- Część aplikacji działająca po stronie klienta została umieszczona w pliku *index.html*.
- Skrypty jQuery wywoływane z poziomu pliku *index.html* znajdują się w katalogu *scripts*.



Rysunek 9.3. Składniki aplikacji arkusza danych

## Baza danych

Edytowalny arkusz danych wyświetla zawartość fikcyjnej bazy towarów. Na serwerze umieściliśmy tabelę danych *product* zbudowaną z następujących pól:

- *product\_id* — unikatowy numer generowany w bazie danych automatycznie za pomocą autoinkrementacji. Jest to klucz główny tej tabeli.
- *name* — nazwa towaru.
- *price* — cena towaru wystawionego na sprzedaż.
- *on\_promotion* — pole liczbowe przyjmujące wartości 0 lub 1 (odpowiednik wartości logicznych *prawda* i *falsz*). W interfejsie jest ono realizowane za pomocą pola wyboru.

Wybór pola *product\_id* na klucz główny arkusza nasuwa się automatycznie, ponieważ przyjmuje ono unikatową wartość dla każdego z towarów. Pole to nigdy nie będzie puste, gdyż jego zawartość jest uzupełniana automatycznie za pomocą funkcji inkrementacji wartości w chwili dodawania towarów do bazy danych:

```
CREATE TABLE product
(
  product_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  name VARCHAR(50) NOT NULL DEFAULT '',
  price DECIMAL(10,2) NOT NULL DEFAULT '0.00',
  on_promotion TINYINT NOT NULL DEFAULT '0',
  PRIMARY KEY (product_id)
);
```

Zawartość pozostałych pól nie wymaga szerszego omawiania — żadne z nich nie może pozostać puste i każdemu z nich, z wyjątkiem pola `product_id`, przypisujemy ręcznie inną wartość. Pole `tinyint` będzie pojawiać się w arkuszu w postaci pola wyboru, które użytkownik będzie zaznaczał w razie potrzeby. Pole `on_promotion` przechowuje zmienne typu `tinyint`, ponieważ zapisywane będą w nim jedynie wartości 1 lub 0 (prawda lub fałsz).

## Style i kolory

Zostawmy na razie kwestię bazy danych i zajmijmy się aspektami bardziej powiązаныmi z kodem aplikacji. Pora zorientować się wreszcie, w jaki sposób działa nasz arkusz danych.

Wspominaliśmy już, że za wygląd aplikacji będzie odpowiadał osobny arkusz stylów CSS. W pliku `index.html` znajdziesz następujący fragment kodu:

```
<link rel="stylesheet" type="text/css" href="scripts/themes/coffee/grid.css"
      title="coffee" media="screen" />
<link rel="stylesheet" type="text/css" media="screen" href="themes/jqModal.css" />
```

Katalog `themes` zawiera definicje różnych motywów graficznych. W prezentowanym powyżej fragmencie kodu pojawia się nazwa `coffee`, ale możesz zmienić ją na inną, na przykład `green`, jeśli chcesz zmienić kolory arkusza. Możesz też utworzyć (zachowując konwencję nazw) własny motyw graficzny, jeśli przygotujesz odpowiednie pliki graficzne, zapiszesz je w podkatalogu katalogu `themes` i zmienisz nazwę motywu we wskazanym wierszu kodu. Wygląd przycisków aplikacji zależy od ścieżki `imgpath` podanej w pliku `index.html` — `'scripts/themes/green/images'`. Jeżeli chcesz go zmienić, musisz zmienić też nazwę motywu w tej ścieżki.

Rozwiązanie wymagające modyfikowania nazwy katalogu w dwóch miejscach pliku jest potencjalnym źródłem błędów, więc należy zachować szczególną ostrożność w czasie przeprowadzania tej operacji. Z pomocą przyjdzie nam biblioteka jQuery, która pozwoli zastosować sprytne rozwiązanie polegające na dynamicznym wybieraniu arkusza CSS i zmienianiu ścieżki `imgpath` na podstawie określonej wcześniej nazwy motywu.

Sztuczka, którą mamy zamiar się posłużyć, wymaga stworzenia dynamicznie znacznika `<link>` wewnątrz sekcji `<head>` i określenia jego atrybutu `rel`, by wskazywał na określony motyw.

Od tej pory zmiana motywu będzie ograniczać się do podania nowej nazwy zmiennej skryptu JavaScript.

Definicje stylów okna nakładki zawiera plik *jqModal.css*, który jest częścią dodatku *jqModal*. (Funkcje umożliwiające jego działanie znajdują się w pliku *jqModal.js* w katalogu *scripts/js*). Wtyczkę i jej arkusz stylów możesz pobrać ze strony <http://dev.iceburg.net/jquery/jqModal/>.

W sekcji `<head>` pliku *index.html* znajduje się też kilka deklaracji `script src`. Zawierają one nazwy plików JavaScript niezbędnych do stworzenia arkusza danych (oraz nazwę pliku nakładki *jqModal.js*).

```
<script src="scripts/jquery-1.3.2.js" type="text/javascript"></script>
<script src="scripts/jquery.jqGrid.js" type="text/javascript"></script>
<script src="scripts/js/jqModal.js" type="text/javascript"></script>
<script src="scripts/js/jqDnR.js" type="text/javascript"></script>
```

Jak widzisz, do poprawnego funkcjonowania arkusza niezbędnych jest kilka plików. Wszystkie je omówimy bardziej szczegółowo w dalszej części rozdziału.

Sekcja `<body>` pliku *index.html* zawiera deklarację tabeli, która będzie stanowiła zrzut naszego arkusza. W sekcji tej znajdują się też fragmenty kodu odpowiedzialne za wyświetlenie arkusza na stronie i wypełnienie go danymi z bazy.

```
<script type="text/javascript">
  var lastSelectedId;
  var theme = "steel";

  $("head").append("<link>");
  css = $("head").children(":last");
  css.attr({
    rel: "stylesheet",
    type: "text/css",
    href: "scripts/themes/"+theme+"/grid.css",
    title: theme,
    media: "screen"
  });

  $('#list').jqGrid({
    url: 'grid.php',
    datatype: 'json',
    mtype: 'POST',
    colNames: ['Lp.', 'Nazwa', 'Cena', 'W promocji'],
    colModel: [
      {name: 'product_id', index: 'product_id', width: 55, editable: false},
      {name: 'name', index: 'name', width: 100, editable: true,
        edittype: 'text', editoptions: {size: 30, maxLength: 50}},
      {name: 'price', index: 'price', width: 80, align: 'right', formatter:
        ↪ 'currency',
        editable: true},
      {name: 'on_promotion', index: 'on_promotion', width: 80,
        formatter: 'checkbox', editable: true, edittype: 'checkbox'}
    ],
```

```

        rowNum:10,
        rowList:[5,10,20,30],
        imgpath: 'scripts/themes/'+theme+'/images',//alters buttons
        pager: $('#pager'),
        sortname: 'product_id',
        viewrecords: true,
        sortorder: "desc",
        caption:"JSON Example",
        width:600,
        height:250,
        onSelectRow: function(id){
            if(id && id!=lastSelectedId){
                $('#list').restoreRow(lastSelectedId);
                $('#list').editRow(id,true,null,onSaveSuccess);
                lastSelectedId=id;
            }
        },
        editurl:'grid.php?action=save'
    });
    function onSaveSuccess(xhr)
    {
        response = xhr.responseText;
        if(response == 1)
            return true;
        return false;
    }
</script>

```

## Po stronie serwera

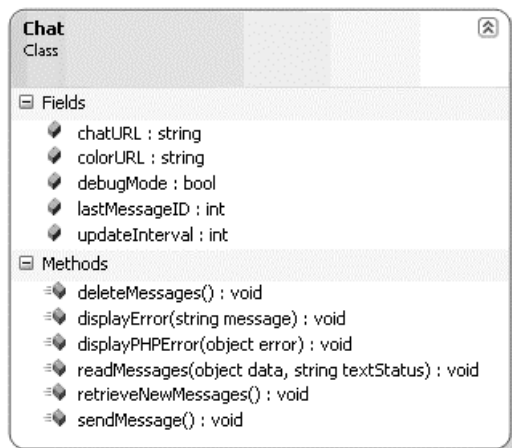
Kod działający po stronie serwera został podzielony na dwa pliki — *grid.php* i *grid.class.php*. Pierwszy z nich jest prostym skrypcem odpowiedzialnym za realizację żądań *load* oraz *save* wysyłanych przez klienta. Ma on następującą strukturę:

```

<?php
... Inicjalizacja
// Otwiera arkusz danych.
if($action == 'load')
{
... Tu otwiera arkusz danych.
}
// Zapisuje arkusz danych.
elseif ($action == 'save')
{
... Tu zapisuje arkusz danych.
}
?>

```

Kod odpowiedzialny za otwieranie i zapisywanie arkusza danych znajduje się w pliku *grid.class.php* w klasie Grid. Budowa tej klasy została przedstawiona na diagramie na rysunku 9.4. Jest ona na tyle prosta, że uznaliśmy, iż nie wymaga szerszego omówienia.



Rysunek 9.4. Diagram klasy Grid

## Budowanie arkusza danych krok po kroku

Jeśli wolisz zapisać cały kod aplikacji własnoręcznie, oto niezbędne wskazówki:

1. Zanim przejdziesz do tworzenia arkusza danych, musisz przygotować zestaw rekordów bazy, z którym będziesz pracować. Wykonaj podany poniżej kod SQL w aplikacji phpMyAdmin. (Żeby oszczędzić Ci pracy, zamieściliśmy tu skróconą wersję kodu dostępnego w archiwum z przykładami).

```
CREATE TABLE product
(
    product_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL DEFAULT '',
    price DECIMAL(10,2) NOT NULL DEFAULT '0.00',
    on_promotion TINYINT NOT NULL DEFAULT '0',
    PRIMARY KEY (product_id)
);

INSERT INTO product(name, price, on_promotion) VALUES('Kostium Dziadka Mroza',
    14.99, 0);
INSERT INTO product(name, price, on_promotion) VALUES('Kostium damy',
    49.99, 1);
INSERT INTO product(name, price, on_promotion) VALUES('Jaskiniowiec',
    12.99, 0);
INSERT INTO product(name, price, on_promotion) VALUES('Kostium ghula',
    18.99, 0);
```

```

INSERT INTO product(name, price, on_promotion) VALUES('Ninja',
15.99, 0);
INSERT INTO product(name, price, on_promotion) VALUES('Mnich',
13.99, 0);
INSERT INTO product(name, price, on_promotion) VALUES('Elvis, czarny kostium',
35.99, 0);
INSERT INTO product(name, price, on_promotion) VALUES('Robin Hood',
18.99, 0);
INSERT INTO product(name, price, on_promotion) VALUES('Pierot',
22.99, 1);
INSERT INTO product(name, price, on_promotion) VALUES('Austin Powers',
49.99, 0);
INSERT INTO product(name, price, on_promotion) VALUES('Obcy',
35.99, 0);
INSERT INTO product(name, price, on_promotion) VALUES('Fantomas',
18.99, 1);
INSERT INTO product(name, price, on_promotion) VALUES('Maska i peleryna
krzykacza', 30.99, 0);

```

2. Sprawdź, czy dane zostały poprawnie wprowadzone do bazy (rysunek 9.5).

The screenshot shows the phpMyAdmin interface for the 'ajax' database. The 'product' table is selected, and the SQL query 'SELECT \* FROM product LIMIT 0, 30' is displayed. The table view shows 9 records with columns: product\_id, name, price, and on\_promotion.

product_id	name	price	on_promotion
1	Kostium Dziadka Mroza	14.99	0
2	Kostium damy	49.99	1
3	Jaskiniowiec	12.99	0
4	Kostium ghula	18.99	0
5	Ninja	15.99	0
6	Mnich	13.99	0
7	Elvis, czarny kostium	35.99	0
8	Robin Hood	18.99	0
9	Pierot	22.99	1

Rysunek 9.5. Tabela product w aplikacji phpMyAdmin

3. W katalogu *ajax* załóż podkatalog *grid*.

4. Skopiuj do niego katalog *scripts* z archiwum z przykładowymi kodami.

5. Utwórz plik o nazwie *config.php* i wpisz w nim następujący kod:

```
<?php
// Definiuje dane niezbędne do połączenia się z bazą.
define('DB_HOST', 'localhost');
define('DB_USER', 'ajaxuser');
define('DB_PASSWORD', 'practical');
define('DB_DATABASE', 'ajax');
?>
```

6. Utwórz nowy plik o nazwie *error\_handler.php* i wprowadź do niego podaną poniżej zawartość.

```
<?php
// Definiuje metodę obsługi błędów.
set_error_handler('error_handler', E_ALL);
// Funkcja odpowiedzialna za obsługę błędów.
function error_handler($errNo, $errStr, $errFile, $errLine)
{
    // Usuwa wszystkie wygenerowane już dane wyjściowe.
    ob_clean();
    // Wysyła komunikat o błędzie.
    $error_message = 'NR BŁĘDU: ' . $errNo . chr(10) .
                    'WIADOMOŚĆ: ' . $errStr . chr(10) .
                    'LOKALIZACJA: ' . $errFile .
                    ', wiersz ' . $errLine;
    echo $error_message;
    // Zapobiega wykonywaniu innych skryptów PHP.
    exit;
}
?>
```

7. W pliku o nazwie *grid.php* umieść podany poniżej skrypt.

```
<?php
// Uruchamia skrypt obsługi błędów i wczytuje klasę Grid.
require_once('error_handler.php');
require_once('grid.class.php');
// Domyślne działanie to otwarcie arkusza.
$action = 'load';
if(isset($_GET['action']))
    $action = $_GET['action'];
// Otwiera arkusz danych.
if($action == 'load')
{
    // Pobiera żądaną stronę.
    $page = $_POST['page'];
    // Pobiera informację o liczbie wierszy, które mają zostać przedstawione w arkuszu.
    $limit = $_POST['rows'];
    // Pobiera identyfikator wiersza wybranego do przeprowadzenia operacji sortowania.
    $sidx = $_POST['sidx'];
    // Pobiera kierunek.
```



```

$sord = $_POST['sord'];

$grid = new Grid($page,$limit,$sidx,$sord);
$response->page = $page;
$response->total = $grid->getTotalPages();
$response->records = $grid->getTotalItemsCount();
$currentPageItems = $grid->getCurrentPageItems();

for($i=0;$i<count($currentPageItems);$i++) {
    $response->rows[$i]['id']=$currentPageItems[$i]['product_id'];
    $response->rows[$i]['cell']=array(
        $currentPageItems[$i]['product_id'],
        $currentPageItems[$i]['name'],
        $currentPageItems[$i]['price'],
        $currentPageItems[$i]['on_promotion']
    );
}
echo json_encode($response);
}
// Zapisuje arkusz danych.
elseif ($action == 'save')
{
    $product_id = $_POST['id'];
    $name = $_POST['name'];
    $price = $_POST['price'];
    $on_promotion = ($_POST['on_promotion'] == 'Yes')?1:0;
    $grid = new Grid();
    echo $grid->updateItem($product_id,$on_promotion,$price,$name);
}
?>

```

8. Utwórz plik o nazwie *grid.class.php* i wpisz w nim następujące instrukcje:

```

<?php
// Otwiera plik konfiguracyjny.
require_once('config.php');
// Rozpoczyna sesję.
session_start();

// Dołącza narzędzia obsługi listy towarów.
class Grid
{
    // Licznik stron arkusza.
    private $mTotalPages;
    // Licznik wpisów w arkuszu.
    private $mTotalItemsCount;
    private $mItemsPerPage;
    private $mCurrentPage;

    private $mSortColumn;
    private $mSortDirection;

```

```

// Funkcja obsługująca połączenie z bazą danych.
private $mMysqli;

// Konstruktor klasy.
function __construct( $currentPage =1, $itemsPerPage=5,
    $sortByColumn='product_id', $sortByDirection='asc')
{
    // Tworzy połączenie z bazą danych MySQL.
    $this->mMysqli = new mysqli(DB_HOST, DB_USER, DB_PASSWORD,
        DB_DATABASE);

    $this->mCurrentPage = $currentPage;
    $this->mItemsPerPage = $itemsPerPage;
    $this->mSortColumn = $sortByColumn;
    $this->mSortDirection = $sortByDirection;
    // Wywołuje funkcję countAllRecords zliczając rekordy arkusza.
    $this->mTotalItemsCount = $this->countAllItems();
    if($this->mTotalItemsCount >0)
        $this->mTotalPages = ceil($this->mTotalItemsCount/$this->mItems
            ↳PerPage);
    else
        $this->mTotalPages=0;
    if($this->mCurrentPage > $this->mTotalPages)
        $this->mCurrentPage = $this->mTotalPages;
}

// Odczytuje stronę towarów i zapisuje ją w zmiennej $this->grid.
public function getCurrentPageItems()
{
    // Tworzy zapytanie SQL, które zwróci stronę towarów.
    $queryString = 'SELECT * FROM product';
    $queryString .= ' ORDER BY ' .
        $this->mMysqli->real_escape_string($this->mSortColumn).
        ' ' . $this->mMysqli->real_escape_string(
        $this->mSortDirection);
    // Nie umieszczaj na stronie $limit*($page - 1).
    $start = $this->mItemsPerPage* $this->mCurrentPage - $this->mItems
        ↳PerPage;
    if ($start<0) $start = 0;
    $queryString .= ' LIMIT ' . $start . ',' . $this->mItemsPerPage;

    // Wykonuje zapytanie.
    if ($result = $this->mMysqli->query($queryString))
    {
        for($i = 0; $items[$i] = $result->fetch_assoc(); $i++);

        // Usuwa ostatni pusty wiersz.
        array_pop($items);
    }
}

```

```

        // Zamyka strumień wyników.
        $result->close();
        return $items;
    }
}

public function getTotalPages()
{
    return $this->mTotalPages;
}

// Aktualizuje informacje o towarze.
public function updateItem($id, $on_promotion, $price, $name)
{
    // Przeprowadza kodowanie danych wejściowych, tak by nie zagrażały zawartości
    // bazy danych.
    $id = $this->mMysqli->real_escape_string($id);
    $on_promotion = $this->mMysqli->real_escape_string($on_promotion);
    $price = $this->mMysqli->real_escape_string($price);
    $name = $this->mMysqli->real_escape_string($name);
    // Tworzy zapytanie SQL, które zaktualizuje rekord towaru.
    $queryString = 'UPDATE product SET name="' . $name . '", ' .
        'price=' . $price . ', ' .
        'on_promotion=' . $on_promotion .
        ' WHERE product_id=' . $id;

    // Wykonuje polecenie SQL.
    $this->mMysqli->query($queryString);
    return $this->mMysqli->affected_rows;
}

// Zwraca całkowitą liczbę rekordów arkusza.
private function countAllItems()
{
    // Zapytanie zwracające licznik rekordów.
    $count_query = 'SELECT COUNT(*) FROM product';
    // Wykonuje zapytanie i przechwytuje wynik.
    if ($result = $this->mMysqli->query($count_query))
    {
        // Pobiera pierwszy zwrócony wiersz.
        $row = $result->fetch_row();
        // Zamyka połączenie z bazą danych.
        $result->close();
        return $row[0];
    }
    return 0;
}

public function getTotalItemsCount()
{

```

```

        return $this->mTotalItemsCount;
    }

    // Koniec klasy Grid.
}
?>

```

9. Na koniec utwórz plik *index.html* i wpisz w nim kod interfejsu aplikacji.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>jqGrid - przykład</title>

        <link rel="stylesheet" type="text/css" media="screen" href="themes/
            ↪jqModal.css" />
        <script src="scripts/jquery-1.3.2.js" type="text/javascript"></script>
        <script src="scripts/jquery.jqGrid.js" type="text/javascript"></script>
        <script src="scripts/js/jqModal.js" type="text/javascript"></script>
        <script src="scripts/js/jqDnR.js" type="text/javascript"></script>
        <script src="scripts/js/grid.locale-pl.js" type="text/
            ↪javascript"></script>
    </head>
    <body>
        <h2>Mój arkusz danych</h2>
        <table id="list" class="scroll" cellpadding="0" cellspacing="
            ↪0"></table>
        <div id="pager" class="scroll" style="text-align:center;"></div>
        <script type="text/javascript">
            var lastSelectedId;
            var theme = "steel";

            $("head").append("<link>");
            css = $("head").children(":last");
            css.attr({
                rel: "stylesheet",
                type: "text/css",
                href: "scripts/themes/"+theme+"/grid.css",
                title: theme,
                media: "screen"
            });

            $('#list').jqGrid({
                url:'grid.php',
                datatype: 'json',
                mtype: 'POST',
                colNames:['Lp.','Nazwa', 'Cena', 'W promocji'],
                colModel:[
                    {name:'product_id',index:'product_id', width:55,editable:false},

```

```

        {name:'name',index:'name', width:100,editable:true,
          editttype:'text',editoptions:{size:30,maxlength:50}},
        {name:'price',index:'price', width:80, align:'right',
        ↪formatter:'currency',
          editable:true},
        {name:'on_promotion',index:'on_promotion', width:80,
          formatter:'checkbox',editable:true, editttype:'checkbox'}
    ],
    rowNum:10,
    rowList:[5,10,20,30],
    imgpath: 'scripts/themes/'+theme+'/images', //Zmienia wygląd przycisków.
    pager: $('#pager'),
    sortname: 'product_id',
    viewrecords: true,
    sortOrder: "desc",
    caption:"Przykład JSON",
    width:600,
    height:250,
    onSelectRow: function(id){
        if(id && id!=lastSelectedId){
            $('#list').restoreRow(lastSelectedId);
            $('#list').editRow(id,true,null,onSaveSuccess);
            lastSelectedId=id;
        }
    },
    editurl:'grid.php?action=save'
});
function onSaveSuccess(xhr)
{
    response = xhr.responseText;
    if(response == 1)
        return true;
    return false;
}
</script>
</body>
</html>

```

10. Wpisz w przeglądarce adres <http://localhost/ajax/grid> i sprawdź, czy arkusz działa poprawnie. Powinien prezentować się tak, jak przedstawiliśmy to na rysunkach 9.1 i 9.2.

Jak widzisz, tak skonstruowany arkusz danych pozwala od ręki edytować zawartość komórek, sortować towary i wprowadzać wszelkie modyfikacje w sposób intuicyjny, reagując na działania użytkownika. Zaprojektowana w ten sposób aplikacja nie wymaga wprowadzania danych seriami, na zasadzie „pliku wsadowego”, więc praca z nią jest bardziej przyjemna i wydajna! Z kolei zastosowanie dostępnych w sieci dodatków i kontrolowanie wyglądu aplikacji za pomocą arkusza CSS sprawia, że programista zyskuje możliwość łatwego przystosowania jej do potrzeb różnych stron. Wystarczy kilka drobnych zmian w kodzie, by program zmienił się zgodnie z wymogami całego projektu, zarówno w warstwie graficznej, jak i w oferowanych funkcjach.

## Podsumowanie

Technologia AJAX nie różni się od innych dziedzin wiedzy — im więcej czasu poświęcisz na ćwiczenie swoich umiejętności, tym bardziej je rozwiniesz. Postaraliśmy się przedstawić Ci zestaw narzędzi pozwalających odnaleźć się w świecie technologii AJAX i pokazać Ci, jak z nich korzystać. Zbudowaliśmy kilka aplikacji od zera, ale też zaprezentowaliśmy metody rozwijania istniejących już programów. Gdy poznasz zasady rządzące tą technologią, znajdziesz się na prostej drodze do sukcesu.

Zawsze z radością wysłuchujemy wszystkich uwag naszych czytelników i chętnie poznajemy ich projekty tworzone w oparciu o proponowane przez nas rozwiązania, dlatego zachęcamy Cię, byś podzielił się z nami swoimi pracami! Mamy nadzieję, że podobał Ci się ten kurs technologii AJAX — czujemy się zaszczytzeni, że zdecydowałeś się poświęcić swój czas na wyprawę w naszym towarzystwie!