Technologia i rozwiązania

PHP i jQuery Receptury

Zbiór kilkudziesięciu sprawdzonych receptur i szybkich rozwiązań problemów, niezbędnych do sprawnego tworzenia interaktywnych aplikacji!

- Jak pracować z bazą danych w kodzie PHP i jQuery?
- Jak pracować z formularzami i dodawać do nich efekty specjalne?
- Jak tworzyć interaktywne, hierarchiczne menu?

Helion

Vijay Joshi

[PACKT] PUBLISHING open source*

PHP i jQuery. Receptury

Autor: Vijay Joshi
Tłumaczenie: Wojciech Moch
ISBN: 8978-83-246-3350-0
Tytuł oryginału: PHP jQuery Cookbook
Format: 170×230, stron: 328

Zbiór kilkudziesięciu sprawdzonych receptur i szybkich rozwiązań problemów niezbędnych do sprawnego tworzenia interaktywnych aplikacji!

- Jak pracować z bazą danych w kodzie PHP i jQuery?
- Jak pracować z formularzami i dodawać do nich efekty specjalne?
- Jak tworzyć interaktywne, hierarchiczne menu?

Język PHP jest podstawowym językiem wybieranym przez twórców stron internetowych, a jQuery – jedną z najczęściej stosowanych bibliotek w sieci. To oczywiste: obie technologie są lekkie, łatwe w użyciu i nauce, a przy tym oferują ogromne możliwości tworzenia dynamicznych witryn i interaktywnych aplikacji WWW. W dodatku razem tworzą doskonale uzupełniający się zestaw wszechstronnych narzędzi dla webmasterów. Jednak w świecie informatyki nic nie jest ani doskonałe, ani dziecinnie proste - dlatego nawet w pracy z takim tandemem możesz napotkać pewne często powtarzające się trudności, które czasem znacznie opóźniają realizację projektu. By tego uniknąć, wykorzystaj ten zbiór ponad sześćdziesięciu prostych, ale wyjątkowo skutecznych receptur i rozwiązań, niezwykle przydatnych przy tworzeniu interaktywnych aplikacji WWW.

W tej przejrzyście napisanej książce znajdziesz wybór najważniejszych zadań i problemów, a także czytelnie przygotowane instrukcje radzenia sobie z nimi. Dzięki temu będziesz mógł jeszcze szybciej i sprawniej tworzyć aplikacje WWW z wykorzystaniem PHP i jQuery, nawet jeśli jesteś początkującym programistą lub webmasterem. Niezależnie od tego, czy chcesz nauczyć się na bieżąco kontrolować dane z formularzy, tworzyć wtyczki, przeciągać elementy, tworzyć atrakcyjne menu i przyjazne formularze, korzystać z API YouTube, czy współpracować z bazą danych – wystarczy sięgnąć po właściwe rozwiązania. Znajdziesz tu również receptury buforowania żądań AJAX oraz obsługi błędów, a także kilka zaawansowanych technik wykorzystania PHP i jQuery do tworzenia bardziej rozbudowanych stron. Dowiesz się między innymi, jak obejść ograniczenia przeglądarek, takie jak żądania przesyłane między domenami, i jak wykorzystać narzędzie Firebug.

Dzięki tej książce:

- zaczniesz od podstaw, aby na koniec poznać triki profesjonalnych twórców stron
- przygotujesz interaktywne, dynamiczne i hierarchiczne menu
- zastosujesz ciekawe efekty specjalne do elementów strony
- wykorzystasz bazę danych w kodzie PHP i jQuery
- za pomocą technologii AJAX poprawisz interakcję użytkownika ze stroną
- dowiesz się, jak wykorzystać formaty XML i JSON do skutecznej wymiany danych
- przygotujesz różne narzędzia do budowania aplikacji WWW
- skontrolujesz dane z formularzy zarówno po stronie klienta, jak i serwera

Wykorzystaj wszystkie możliwości tkwiące w technologiach PHP i jQuery!
Poznaj rozwiązania typowych problemów, które możesz napotkać!

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

Spis treści

Podziękowania 7

O autorze 9

O recenzentach 11

Wstęp 13

Rozdział 1. Obsługa zdarzeń w jQuery 17

Wprowadzenie 17
Wykonywanie funkcji w momencie załadowania strony 18
Wiązanie i odwiązywanie elementów 20
Dodawanie zdarzeń do elementów, które zostaną utworzone później 24
Przesyłanie formularza za pomocą jQuery 27
Kontrola brakujących obrazków 29
Tworzenie funkcji zaznacz/usuń zaznaczenie wszystkich pól wyboru 32
Przechwytywanie zdarzeń myszy 36
Tworzenie skrótów klawiszowych 39
Wyświetlanie tekstu wybranego przez użytkownika 43
Przeciąganie elementów na stronie 47

Rozdział 2. Łączenie PHP i jQuery 51

Wprowadzenie 51
Pobieranie danych z PHP za pomocą jQuery 53
Automatyczne tworzenie tekstu zapytania na podstawie elementów formularza 57
Wykrywanie żądań AJAX w skryptach PHP 60
Wysyłanie danych do PHP 62
Przerywanie żądań AJAX 66
Tworzenie pustej strony i ładowanie jej w częściach 69
Obsługa błędów w żądaniach AJAX 73
Blokowanie w przeglądarce buforowania żądań AJAX 77
Ładowanie JavaScriptu na żądanie, aby zmniejszyć czas ładowania strony 79

Rozdział 3. Praca z dokumentami XML 83

Wprowadzenie 83
Ładowanie danych XML z plików oraz ciągów znaków za pomocą SimpleXML 86
Korzystanie z elementów i atrybutów za pomocą SimpleXML 89
Wyszukiwanie elementów za pomocą XPath 94
Odczytywanie dokumentów XML za pomocą rozszerzenia DOM 98
Tworzenie dokumentów XML za pomocą rozszerzenia DOM 102
Modyfikowanie dokumentów XML za pomocą rozszerzenia DOM 105
Parsowanie dokumentów XML za pomocą biblioteki jQuery 109

Rozdział 4. Praca z formatem JSON 113

Wprowadzenie 113
Tworzenie danych w formacie JSON za pomocą PHP 115
Odczytywanie danych w formacie JSON za pomocą PHP 117
Przechwytywanie błędów analizy danych w formacie JSON 120
Korzystanie z danych w formacie JSON za pomocą jQuery 122

Rozdział 5. Praca z formularzami 129

Wprowadzenie 129
Dynamiczne dodawanie pól do formularza 130
Wyszukiwanie na stronie tekstu wprowadzonego przez użytkownika 133
Szukanie pustych pól za pomocą biblioteki jQuery 137
Sprawdzanie poprawności liczb za pomocą biblioteki jQuery 141
Sprawdzanie poprawności adresów e-mail i adresów WWW za pomocą wyrażeń regularnych 144
Wyświetlanie błędów w czasie wprowadzania danych, czyli sprawdzanie danych na żywo 148
Mocniejsza kontrola formularza, czyli ponowne kontrolowanie w PHP 152
Tworzenie systemu do głosowania 158
Zezwalanie na kod HTML w polach tekstowych i ograniczanie zbioru dozwolonych znaczników 163

Rozdział 6. Efekty specjalne w formularzach 167

Wprowadzenie 167
Tworzenie gry w kółko i krzyżyk 168
Informowanie użytkownika o przetwarzaniu żądania AJAX 175
Tworzenie rozwijanych i zwijanych ramek (harmonijek) 181
Stopniowe ukrywanie elementu po jego zaktualizowaniu 185
Wyświetlanie pływającego okienka na żądanie 188
Aktualizowanie pozycji w koszyku na zakupy 192

Rozdział 7. Tworzenie menu nawigacyjnych 201

Wprowadzenie 201
Tworzenie prostych menu rozwijanych 202
Tworzenie menu zmieniającego kolor tła po wskazaniu myszą 206
Tworzenie menu harmonijkowego 209

Tworzenie menu pływającego 216
Tworzenie interfejsu do nawigacji w kartach 221
Dodawanie nowych kart 225
Tworzenie kreatora za pomocą kart 231

Rozdział 8. Wiązanie danych w PHP i jQuery 239

Wprowadzenie 239
Pobieranie danych z bazy i wyświetlanie ich w formie tabeli 240
Zbieranie danych z formularza i zapisywanie ich w bazie 246
Wypełnianie powiązanych ze sobą list rozwijanych 251
Sprawdzanie w bazie danych dostępności nazwy użytkownika 257
Podział dużych ilości danych na strony 262
Dodawanie funkcji automatycznych podpowiedzi do pól tekstowych 267
Tworzenie chmury znaczników 275

Rozdział 9. Rozbudowywanie stron za pomocą PHP i jQuery 281

Wprowadzenie 281
Wysyłanie żądań między domenami z wykorzystaniem serwera proxy 282
Tworzenie międzydomenowych żądań za pomocą biblioteki jQuery 288
Tworzenie strony przewijającej się w nieskończoność 293
Tworzenie wtyczki do biblioteki jQuery 298
Wyświetlanie kanałów RSS za pomocą PHP i jQuery 302

Dodatek. Firebug 309

Wprowadzenie 309
Badanie elementów strony 311
Edytowanie kodu HTML i stylów CSS 313
Debugowanie kodu JavaScript 315

Skorowidz 319

ROZDZIAŁ 4

Praca z formatem JSON

W tym rozdziale zajmiemy się:

- tworzeniem danych w formacie JSON za pomocą PHP,
- odczytywaniem danych w formacie JSON za pomocą PHP,
- przechwytywaniem błędów analizy danych w formacie JSON,
- korzystaniem z danych w formacie JSON za pomocą jQuery.

Wprowadzenie

W ostatnim czasie format JSON (ang. *JavaScript Object Notation*) stał się bardzo popularnym formatem wymiany danych i coraz więcej programistów zaczyna przedkładać go nad format XML. Coraz więcej witryn WWW korzysta z tego formatu jako domyślnej formy udostępniania danych.

JSON jest formatem tekstowym, który jest niezależny od języka programowania, ale stanowi wewnętrzną formę prezentacji danych w języku JavaScript. Jest to format znacznie szybszy i lżejszy niż XML, ponieważ w porównaniu do niego wymaga zastosowania mniejszej liczby znaczników.

Ze względu na to, że format JSON jest wewnętrznym formatem danych języka JavaScript, możemy znacznie łatwiej korzystać z niego po stronie klienta w aplikacjach korzystających z technologii AJAX.

Obiekty zapisane w formacie JSON zaczynają się od otwierającego nawiasu klamrowego ({), a kończą się zamykającym nawiasem klamrowym (}). Zgodnie ze specyfikacją formatu dozwolone jest w nim stosowanie następujących typów danych:

- **obiektów** — Obiekt jest kolekcją par klucz-wartość zamkniętych pomiędzy nawiasami klamrowymi ({ i }), a rozdzielanych przecinkiem. Klucze i wartości są od siebie oddzielane za pomocą dwukropka (:). O takich obiektach można myśleć jak o tablicach asocjatywnych lub tabelach skrótów. Klucze są tutaj prostymi ciągami znaków, a wartościami mogą być tablice, ciągi znaków, liczby, wartości logiczne lub wartość `null`.
- **tablic** — Podobnie jak w innych językach tablica jest uporządkowanym zbiorem danych. Dane w tablicy rozdzielane są za pomocą przecinka, a wszystkie muszą znaleźć się między nawiasami kwadratowymi ([i]).
- **ciągów znaków** — Każdy ciąg znaków musi być zamknięty za pomocą znaków cudzysłowu.
- **liczb** — To ostatni typ danych.

Dane w formacie JSON mogą być tak proste jak poniższe:

```
{
  "name":"Superman", "address": "gdzieś"
}
```

Poniżej przedstawiam też przykład korzystających z tablic:

```
{
  "name": "Superman", "phoneNumbers": ["8010367150", "9898989898", "1234567890"
]
}
```

I jeszcze jeden, troszkę bardziej skomplikowany przykład, w którym użyłem obiektów, tablic oraz wartości:

```
{
  "people":
  [
    {
      "name": "Vijay Joshi",
      "age": 28,
      "isAdult": true
    },
    {
      "name": "Charles Simms",
      "age": 13,
      "isAdult": false
    }
  ]
}
```

Należy tutaj wspomnieć o ważnym szczególe:

```
{
  'name':'Superman', 'address': 'gdzieś'
}
```

Powyższy ciąg znaków jest prawidłowym obiektem JavaScript, ale nie jest zgodny z formatem JSON. Format ten wymaga, aby nazwa i wartość były umieszczone pomiędzy znakami cudzysłowu, i nie dopuszcza stosowania znaków apostrofu.

Inną rzeczą, o której trzeba pamiętać, jest konieczność stosowania właściwego zestawu znaków.

Proszę pamiętać, że format JSON wymaga zapisywania danych w kodowaniu UTF-8, natomiast język PHP domyślnie zakłada stosowanie kodowania ISO-8859-1.

Trzeba też pamiętać o tym, że format JSON nie jest językiem JavaScript, a jedynie specyfikacją formatu danych lub jednym z wielu elementów języka JavaScript.

Skoro wiemy już, czym jest format JSON, możemy przystąpić do tworzenia receptur, w których nauczymy się korzystać z tego formatu za pomocą języka PHP i biblioteki jQuery.

Utwórz zatem nowy katalog i nadaj mu nazwę `rozdzial4`. Wszystkie receptury z tego rozdziału będziemy umieszczać w tym katalogu, dlatego skopiuj do niego jeszcze plik *jquery.js*.

Chcąc skorzystać z funkcji obsługi formatu JSON w języku PHP, należy zainstalować interpreter PHP w wersji 5.2 lub wyższej.

Tworzenie danych w formacie JSON za pomocą PHP

W tej recepturze można tworzyć struktury danych w formacie JSON za pomocą tablic i obiektów PHP.

Przygotowania

W katalogu *rozdzial4* utwórz nowy katalog i nazwij go *receptura1*.

Jak to zrobić?

1. W katalogu *receptura1* utwórz nowy plik i nazwij go *index.php*.
2. Do pliku wpisz kod PHP tworzący ciąg znaków w formacie JSON na podstawie tablicy.

```
<?php
  $travelDetails = array(
    'origin' => 'Delhi',
    'destination' => 'London',
    'passengers' => array
    (
      array('name' => 'Mr. Perry Mason', 'type' => 'Adult', 'age'=> 28),
      array('name' => 'Miss Irene Adler', 'type' => 'Adult', 'age'=> 28)
    ),
    'travelDate' => '17-Dec-2010'
  );
  echo json_encode($travelDetails);
?>
```

3. Uruchom plik w przeglądarce, a wyświetlony zostanie ciąg znaków w formacie JSON. Po wprowadzeniu odpowiednich wcięć całość będzie wyglądała następująco:

```
{
   "origin":"Delhi",
   "destination":"London",
   "passengers":[
      {
         "name":"Mr. Perry Mason",
         "type":"Adult",
         "age":28
      },
      {
         "name":"Miss Irene Adler",
         "type":"Adult",
         "age":28
      }
   ],
   "travelDate":"17-Dec-2010"
}
```

Jak to działa?

Język PHP udostępnia nam funkcję `json_encode()`, która tworzy ciąg znaków w formacie JSON na podstawie obiektów i tablic. Funkcja ta przyjmuje dwa parametry: pierwszy z nich jest wartością przeznaczoną do zakodowania, a w drugim można podać opcje sterujące kodowaniem znaków specjalnych. Ten drugi parametr jest opcjonalny.

W podanym tu kodzie utworzyliśmy dość złożoną tablicę asocjatywną przechowującą dane o podróżach dwojga pasażerów. Przekazanie tej tablicy do funkcji `json_encode()` pozwoliło nam uzyskać reprezentację tej tablicy w formacie JSON.

I coś jeszcze

Wstępnie zdefiniowane stałe

Dowolną z podanych niżej stałych można przekazać jako drugi parametr funkcji `json_encode()`.

- **`JSON_HEX_TAG`** — Zamienia znaki < i > na encje `\u003C` i `\u003E`.
- **`JSON_HEX_AMP`** — Zamienia znak `&s` na `\u0026`.
- **`JSON_HEX_APOS`** — Zamienia znak `'` na `\u0027`.
- **`JSON_HEX_QUOT`** — Zamienia znak `"` na `\u0022`.
- **`JSON_FORCE_OBJECT`** — Wymusza, żeby wartość zwracana w ciągu znaków była obiektem, a nie tablicą.

> Podane tu stałe zdefiniowane są w języku PHP od wersji 5.3 wzwyż.

Zobacz również

- Recepturę „Odczytywanie danych w formacie JSON za pomocą PHP”.
- Recepturę „Przechwytywanie błędów analizy danych w formacie JSON”.

Odczytywanie danych w formacie JSON za pomocą PHP

W przeciwieństwie do poprzedniej receptury tym razem zajmiemy się odczytywaniem ciągów znaków w formacie JSON i tworzeniem na ich podstawie obiektów lub tablic, a także dekodowaniem takich ciągów znaków dzięki udostępnianym przez język PHP funkcjom JSON.

Przygotowania

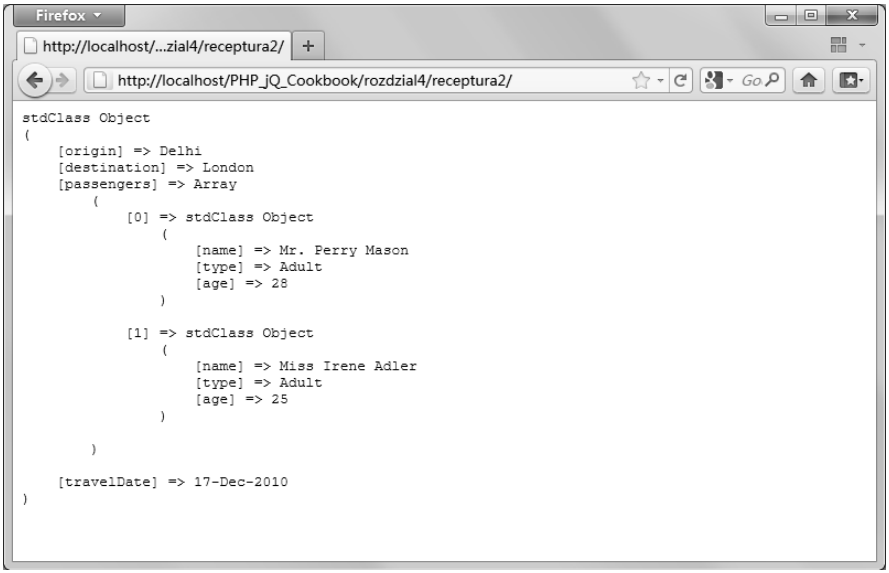
W katalogu *rozdzial4* utwórz nowy katalog o nazwie *receptura2*.

Jak to zrobić?

1. W katalogu *receptura2* utwórz plik o nazwie *index.php*.
2. Spróbuj teraz zmienić ciąg znaków w formacie JSON w obiekt, używając do tego metody `json_decode()`. Po zakończeniu konwersji wypisz wynikowy obiekt na ekranie. Na potrzeby funkcji `json_decode()` możesz użyć wyniku działania poprzedniej receptury, ponieważ na pewno będzie to prawidłowy ciąg znaków JSON.

```
<?php
$json = <<<JSON
{
  "origin":"Delhi",
  "destination":"London",
  "passengers":
  [
    {
      "name":"Mr. Perry Mason",
      "type":"Adult",
      "age":28
    },
    {
      "name":"Miss Irene Adler",
      "type":"Adult",
      "age":25
    }
  ],
  "travelDate":"17-Dec-2010"
}
JSON;
echo '<pre>';
$objJson = json_decode($json);
print_r ($objJson);
echo '</pre>';
?>
```

3. Uruchom plik *index.php* w przeglądarce, a zobaczysz wynik przekształcenia ciągu znaków JSON do postaci obiektu. Chcąc skorzystać z wartości zapisanych w tym obiekcie, można się do nich odwołać dokładnie tak samo, jak do wartości każdego innego obiektu w języku PHP.



Jak to działa?

Funkcja `json_decode()` przekształca prawidłowo zbudowane ciągi znaków JSON w obiekty języka PHP. Przyjmuje ona trzy parametry, które opisuję poniżej:

- Ciąg znaków w formacie JSON.
- Opcjonalny parametr `assoc`, którego domyślną wartością jest `false`. Jeżeli zmienimy ją na `true`, to funkcja `json_decode()` będzie przekształcała obiekty w tablice asocjatywne.
- Opcjonalny parametr `depth`, który definiuje maksymalną głębokość zagnieżdżania rekursywnych struktur w ciągu znaków JSON. Przed wersją PHP 5.3 parametr ten miał domyślną wartość `128`, ale od wersji 5.3 jego wartość wzrosła do `512`.

W podanym kodzie do zdefiniowania ciągu znaków JSON wykorzystaliśmy składnię HEREDOC. Następnie przekazaliśmy ten ciąg znaków do funkcji `json_decode()`, która przekształca go w obiekt.

Teraz możemy już korzystać z wartości tego obiektu za pomocą standardowych operatorów języka PHP. Na przykład datę wycieczki możemy pobrać za pomocą następującej instrukcji:

```
$objJson->travelDate
```

Podobnie instrukcja:

```
$objJson->passengers[1]->name
```

zwróci nam nazwisko drugiego pasażera, czyli *Miss Irene Adler*.

Zobacz również

- Recepturę „Tworzenie danych w formacie JSON za pomocą PHP”.
- Recepturę „Korzystanie z danych w formacie JSON za pomocą jQuery”.
- Recepturę „Przechwytywanie błędów analizy danych w formacie JSON”.

Przechwytywanie błędów analizy danych w formacie JSON

Błędy są nieodłączną częścią procesu tworzenia aplikacji. Dlatego właśnie powinniśmy jak najlepiej obsługiwać powstające błędy, tak żeby ułatwić życie użytkownikom naszych produktów. Podczas tworzenia i dekodowania ciągów znaków w formacie JSON mogą powstać różne błędy. Na przykład wartość przekazana do funkcji może mieć nieprawidłową postać i naruszać zasady budowy obiektów JSON. W takich przypadkach musimy wyłapać wszystkie błędy i odpowiednio je obsłużyć.

W tej recepturze będziemy się zajmować obsługą błędów w funkcjach obsługujących format JSON. Wykorzystamy przy tym metody wbudowane w interpreter PHP, przeznaczone do wykrywania błędów w strukturach JSON.

> Proszę pamiętać o tym, że obsługa błędów w strukturach JSON dostępna jest dopiero od wersji PHP 5.3. W związku z tym proszę się upewnić, że mamy zainstalowaną właściwą wersję interpretera.

Przygotowania

W katalogu *rozdzial4* utwórz nowy katalog o nazwie *receptura3*. Sprawdź też, czy masz zainstalowany interpreter w wersji 5.3 lub nowszej.

Jak to zrobić?

1. W katalogu *receptura3* utwórz nowy plik o nazwie *index.php*.
2. Korzystając z tego samego ciągu znaków w formacie JSON, którego użyliśmy w poprzedniej recepturze, spróbuj przekształcić go w obiekt. Następnie za pomocą instrukcji `switch` sprawdź, czy w czasie przekształcania nie wystąpiły żadne błędy, i odpowiednio wypisz wyniki na ekranie.

```
<?php
$json = <<<JSON
{
  "origin":"Delhi",
  "destination":"London",
  "passengers":
  [
    {
      "name":"Mr. Perry Mason",
      "type":"Adult",
      "age":28
    },
    {
      "name":"Miss Irene Adler",
      "type":"Adult",
      "age":25
    }
  ],
  "travelDate":"17-Dec-2010"
}
JSON;
$objJson = json_decode($json);
switch(json_last_error())
{
  case JSON_ERROR_NONE:
    echo'Data wylotu:' . $objJson->travelDate;
    break;
  case JSON_ERROR_DEPTH:
    echo 'Ciąg znaków JSON przekroczył maksymalną głębokość stosu.';
    break;
  case JSON_ERROR_CTRL_CHAR:
    echo 'Błąd znaków kontrolnych';
    break;
  case JSON_ERROR_SYNTAX:
    echo 'Nieprawidłowa struktura JSON : Proszę sprawdzić składnię';
    break;
}
?>
```

3. Uruchom plik *index.php* w przeglądarce. Ciąg znaków JSON ma prawidłową strukturę, dlatego na ekranie powinien pojawić się tekst *Data wylotu: 17-Dec-2010*. Teraz usuń przecinek z wiersza `"destination":"London"`. Po zapisaniu pliku załaduj go ponownie w przeglądarce. Tym razem pojawi się w niej komunikat *Nieprawidłowa struktura JSON : Proszę sprawdzić składnię*.

Jak to działa?

Od wersji PHP 5.3 dostępna jest funkcja `json_last_error()`, która nie pobiera żadnych parametrów, ale zwraca dane ostatniego błędu, jaki wystąpił podczas analizy ciągu znaków JSON. Zwraca ona wartość całkowitą, na podstawie której można określić rodzaj błędu. W PHP zdefiniowane zostały też stałe dla poszczególnych rodzajów błędów:

- **`JSON_ERROR_NONE`** — oznacza, że ciąg znaków JSON był prawidłowy i przy jego analizie nie powstały żadne błędy.
- **`JSON_ERROR_SYNTAX`** — oznacza, że w ciągu znaków JSON pojawiły się błędy składniowe.
- **`JSON_ERROR_CTRL_CHAR`** — napotkano nieprawidłowy znak kontrolny.
- **`JSON_ERROR_DEPTH`** — ciąg znaków JSON przekroczył maksymalną dopuszczalną głębokość stosu.

Zobacz również

- Recepturę „Odczytywanie danych w formacie JSON za pomocą PHP".

Korzystanie z danych w formacie JSON za pomocą jQuery

Wiemy już, jak należy generować obiekty JSON w języku PHP. Możemy zatem wykorzystać tę wiedzę w małym praktycznym projekcie. Napiszemy tutaj przykład, w którym będziemy żądali danych w formacie JSON od skryptu PHP (do tworzenia żądania wykorzystamy oczywiście bibliotekę jQuery), a następnie wyświetlimy je na stronie WWW.

Przygotowania

W katalogu *rozdzial4* utwór nowy katalog o nazwie *receptura4*.

Jak to zrobić?

1. W utworzonym przed chwilą katalogu *receptura4* utwórz plik o nazwie *index.html*.
2. Do tego pliku zapisz kod HTML tworzący pustą listę rozwijaną oraz pustą listę wypunktowaną. W sekcji `head` zdefiniuj też kilka stylów CSS, tak żeby te elementy ładniej wyglądały.

```
<html>
  <head>
    <title>Korzystanie z danych w formacie JSON</title>
    <style type="text/css">
      body,select,ul{ font-family:"trebuchet MS",verdana }
      ul{ list-style::none;margin:0pt;padding:0pt;}
    </style>
  </head>
  <body>
    <h3>Wybierz datę, aby zobaczyć szczegóły lotu</h3>
    <p>
      <select id="travelDates">
      </select>
      <ul>
        <li id="origin"></li>
        <li id="destination"></li>
        <li id="travellers"></li>
      </ul>
    </p>
  </body>
</html>
```

3. Teraz dopiszemy kod używający biblioteki jQuery. Po pierwsze tuż przed zamykającym znacznikiem `body` dodaj referencję do pliku biblioteki. Następnie dopisz własny kod, który przygotuje żądanie danych w formacie JSON przesłane do pliku *json.php*. W momencie otrzymania odpowiedzi lista rozwijana zostanie wypełniona i dołączona zostanie do niej funkcja obsługi zdarzenia `change`. Po wybraniu wartości z listy wywoływana jest kolejna funkcja, która poszuka w otrzymanej wcześniej odpowiedzi w formacie JSON wybranej daty i wyświetli pasujące do niej informacje.

```
<script type="text/javascript" src="../jquery.js"></script>
<script type="text/javascript">
  $(document).ready(function ()
  {
    var jsonResult;
    $.getJSON("json.php",displayData);
    function displayData(data)
    {
      jsonResult = data;
      var str = '<option value="">wybierz datę</option>';
      for(var i=0; i<data.length;i++)
      {
        str+= '<option value="' + data[i].travelDate + '">' + data[i].
        ↪travelDate + '</option>';
      }
      $('#travelDates').html(str);
      $('#travelDates').change(function()
      {
        if($(this).val() != '')
        {
```

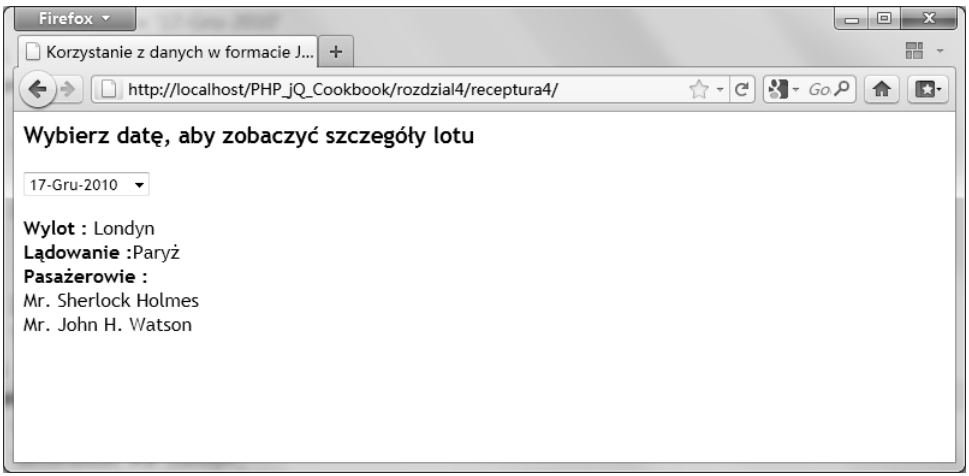
```
            displayDetails($(this).val());
          }
        });
      }
      function displayDetails(selectedDate)
      {
        for(var i=0; i<jsonResult.length;i++)
        {
          var aResult = jsonResult[i];
          if(aResult.travelDate == selectedDate)
          {
            $('#origin').html('<strong>Wylot : </strong>'+
              aResult.origin);
            $('#destination').html('<strong>Lądowanie :</strong>'+ aResult.
            ↪destination);
            var travellers = aResult.passengers;
            var strTraveller = '<ul>';
            for(var j=0; j<travellers.length;j++)
            {
              strTraveller+= '<li>';
              strTraveller+= travellers[j].name;
              strTraveller+= '</li>';
            }
            strTraveller+= '</ul>';
            $('#travellers').html('<strong>Pasażerowie : <br/></strong>
            ↪'+ strTraveller);
            break;
          }
        }
      }
    });
  </script>
```

4. Teraz przyjrzyjmy się plikowi obsługującemu żądanie. Utwórz zatem nowy plik i nadaj mu nazwę *json.php*. W pliku tym zapisz tablicę z informacjami o lotach oraz dane kilku podróżników, a następnie przekształć ją do formatu JSON i odeślij do przeglądarki.

```
<?php
$travelDetails = array(
  array(
    'origin' => 'London',
    'destination' => 'Paris',
    'passengers' => array
    (
      array('name' => 'Mr. Sherlock Holmes', 'age'=> 34),
      array('name' => 'Mr. John H. Watson', 'age'=> 32)
    ),
    'travelDate' => '17-Dec-2010'
```

```
    ),
    array(
      'origin' => 'Delhi',
      'destination' => 'London',
      'passengers' => array
      (
        array('name' => 'Mr. Albert Einstein', 'age'=> 51),
        array('name' => 'Mr. Isaac Newton' ,'age'=> 43)
      ),
      'travelDate' => '25-Jan-2011'
    ),
    array(
      'origin' => 'Delhi',
      'destination' => 'London',
      'passengers' => array
      (
        array('name' => 'Prof. John Moriarty', 'age'=> 44),
        array('name' => 'Miss Irene Adler', 'age'=> 28)
      ),
      'travelDate' => '30-Mar-2011'
    )
  );
  header('Content-Type:text/json');
  echo json_encode($travelDetails);
  ?>
```

5. I to już wszystko! Otwórz w przeglądarce plik *index.html*. Pojawi się wtedy lista wyboru, w której dostępnych będzie kilka dat. Wybierz jedną z nich, a zobaczysz dodatkowe informacje na temat wybranej opcji.



Jak to działa?

Gdy struktura DOM dokumentu jest już gotowa, wywołujemy metodę `getJSON()`. Do tej pory poznaliśmy metody `get()` i `post()`. Tym razem korzystamy ze specjalizowanej metody używanej w przypadku, gdy chcemy pobrać z serwera dane w formacie JSON. Oto lista parametrów, jakie możemy przekazać funkcji `getJSON()`:

- **URL** — Adres URL, na który ma zostać wysłane żądanie.
- **Dane** — Dane, które mają zostać wysłane do serwera.
- **Funkcja wywołania zwrotnego** — Ta funkcja zostanie wywołana po odebraniu odpowiedzi z serwera. W przypadku otrzymania danych JSON biblioteka jQuery najpierw je przeanalizuje i utworzy obiekt JavaScriptu, a następnie przekaże je do wskazanej w parametrze funkcji.

W naszym przykładzie żądanie jest wysyłane do pliku *json.php*, w którym zadeklarowana została tablica z wieloma elementami. Następnie za pomocą metody `json_encode()` tablica ta przekształcana jest w ciąg znaków JSON i wysyłana do przeglądarki.

W momencie otrzymania odpowiedzi biblioteka jQuery przekształca ją w obiekt i przekazuje do funkcji wywołania zwrotnego o nazwie `displayData()`. Otrzymany obiekt zapisywany jest w globalnej zmiennej `jsonResult`.

Następnie iterujemy po tym obiekcie i wstawiamy do listy rozwijanej kolejne daty wylotów. Ze względu na to, że dane są teraz obiektem JavaScriptu, musimy stosować też składnię tego języka.

Po wypełnieniu danymi listy rozwijanej dołączamy do niej funkcję obsługi zdarzenia `change`. W momencie wybrania wartości z listy wywołana zostanie zatem funkcja `displayDetails()`. Funkcja ta poszukuje w obiekcie wybranej przez użytkownika daty. Po jej znalezieniu odczytuje ona informacje o miejscu wylotu, przylotu oraz listę pasażerów. Wszystkie te dane są następnie wyświetlane na stronie.

I coś jeszcze

Inne metody pozwalające na pobranie danych JSON

Jak wspominałem już wcześniej, funkcja `getJSON()` została zaprojektowana do używania wyłącznie w sytuacji, w której z góry wiemy, że w odpowiedzi serwer przyśle nam dane w formacie JSON. Działanie tej funkcji można też symulować za pomocą innych metod biblioteki jQuery, takich jak `get()` lub `post()`, a nawet za pomocą niskopoziomowej funkcji `ajax()`.

```
$.get(
  'json.php',
```

```
    displayData,
    'json'
);
```

Dzięki podaniu w ostatnim parametrze wartości `json` biblioteka jQuery będzie zakładała, że otrzymana od serwera odpowiedź jest ciągiem znaków JSON i spróbuje przekształcić go w obiekt. Podobne zachowanie można też zrealizować za pomocą funkcji `post()` i `ajax()`.

Obsługa błędów podczas odbierania danych JSON

Przedstawione powyżej sposoby pobierania danych wykorzystujące metody `getJSON()` lub `get()` nie pozwalają na samodzielną obsługę błędów. Na przykład, jeżeli żądaliśmy danych za pomocą metody `getJSON()`, a serwer prześle nam źle zbudowany ciąg znaków, to całe żądanie zostanie uznane w tle za nieudane. Istnieją zatem dwa sposoby radzenia sobie w takiej sytuacji. Albo skorzystamy z funkcji `ajaxError()`, która jest uruchamiana, w przypadku gdy w żądaniu AJAX powstaną jakieś błędy, albo użyjemy niskopoziomowej funkcji `ajax()`, która pozwala na wskazanie funkcji wywoływanej w przypadku wystąpienia błędu. Obie te funkcje zostały omówione dokładniej w recepturze „Obsługa błędów w żądaniach AJAX" w poprzednim rozdziale.

Parsowanie ciągu znaków w formacie JSON

Oprócz używania funkcji `getJSON()` i podawania odpowiedniego typu danych w żądaniach AJAX możemy też samodzielnie parsować ciągi znaków JSON i przekształcać je w obiekty. Biblioteka jQuery udostępnia metodę `parseJSON()`, która przekształca ciąg znaków w obiekt języka JavaScript.

```
var objJSON = jQuery.parseJSON('{"klucz":"wartość"}');
```

Od tego momentu w zmiennej `objJSON` zapisany jest obiekt JavaScriptu.

Inną, niezalecaną, metodą jest wykorzystanie funkcji `eval()` udostępnianej przez język JavaScript.

```
var objJSON = eval('(' + '{"klucz":"wartość"}' + ')')
```

Zastosowanie funkcji `eval()` może mieć fatalne skutki dla naszej strony, ponieważ funkcja ta wykona wszelkie przekazane jej dane. Oznacza to, że zalecanym sposobem jest używanie funkcji `parseJSON()` lub specjalnych metod biblioteki jQuery przeznaczonych do obsługi żądań AJAX.

Zobacz również

- Recepturę „Tworzenie danych w formacie JSON za pomocą PHP".
- Recepturę „Pobieranie danych z PHP za pomocą jQuery" z rozdziału 2.

Skorowidz

A

AJAX, 13, 52
  Asynchroniczny, 52
  JavaScript, 52
  XML, 52
akapit, 43
aktualizowanie pozycji w koszyku, 192
aktywacja łącza Czytaj więcej, 215
animacja menu, 205
aplikacje WWW, 13
atrybut
  checked, 35
  href, 205
  name, 59, 162
  src, 31, 292
  value, 162
  votes, 162
automatyczne podpowiedzi, 267

B

baza danych exampleDB, 240
biblioteka
  libxml, 86
  SimpleXML, 89
blokowanie buforowania żądań, 77
błędy, *Patrz* rodzaje błędów, 122

C

chmura znaczników, 275, 278
czyszczenie danych, 240

D

debugowanie kodu, *Patrz* Firebug
dodawanie nowych kart, 225
DOM, 14, 98, 102

E

efekt podświetlenia, 186
element
  channel, 306
  rss, 306
  textarea, 250

F

filtrowanie zawartości listy, 251
filtrowanie znaczników, 163
filtry, 157
filtry poprawiające dane, 157
Firebug, 15, 199, 309
  debugowanie kodu JavaScript, 315
  modyfikowanie stylu konkretnego elementu, 314
  Nowe wyrażenie czujki, 316
  panel do edytowania kodu HTML i stylów CSS, 313
  przyciski dodatku Firebug
    CSS, 310
    DOM, 310
    HTML, 310
    Konsola, 309
    Sieć, 310
    Skrypt, 310

Firebug
sprawdzania stylów CSS, 311
sprawdzanie żądań AJAX, 317
format
HTML, 52
JSON, 52, 113
tekstowy, 52
własny, 52
formularz, 28
dynamiczne dodawanie pól, 130
kontrola po stronie klienta, 152
kontrola po stronie serwera, 152
sprawdzanie danych na żywo, 148
sprawdzanie poprawności adresów e-mail i adresów WWW, 144
sprawdzanie poprawności danych, 28
sprawdzanie poprawności liczb, 141
szukanie pustych pól, 137
formularz rejestracyjny, 257
funkcja
$(document).scrollTop(), 191, 220
addClass(), 225
addEvents(), 80
animate(), 191, 209, 220
checkForWin(), 171
clearSelection(), 135
data(), 267
displayData(), 126
displayDetails(), 126
displaySelectedValues(), 237
dragElement(), 48
each(), 112
eval(), 127
file_get_contents(), 287, 306
filter_var(), 156
getCurrentTabIndes(), 236
getData(), 215
getHTML, 56
getList(), 254
getPositions(), 45
hide(), 221
highlight(), 135
hover(),39, 174, 205, 209
index(), 225
isNaN(), 144
isset(), 156
isUserAtBottom(), 297
json_decode(), 119
json_encode(), 116, 279
json_last_error(), 122
libxml_get_errors(), 88
libxml_use_internal_errors(), 87
live(), 132, 230, 274
loadData(), 297
preventDefault(), 29
real_escape_string(), 250
removeClass(), 225
replace(), 137
session_start(), 198
setcookie(), 162
setInterval(), 188
setTimeout(), 302
show(), 180, 221
simplexml_load_file(), 87, 198
class_name, 88
filename, 88
options, 88
simplexml_load_string(), 88
sleep(), 178
slideToggle(), 221, 307
strip_tags(), 166
test(), 147
toggleClass(),112, 274
trim(), 140
validate(), 138
YFT(), 188
funkcja
obsługi błędu, 76
obsługi zdarzenia change, 123
wywołania zwrotnego, 22, 126
showVideoList(), 293
zaznacz/usuń zaznaczenie, 32

G

globalny obiekt game, 173

H

harmonijka, 181

I

identyfikator
all, 97
cart, 195
check, 259
cloud, 276, 280
container, 46, 169, 294
countryList, 256
error, 268
goTo, 267
information, 256
last, 97
loading, 180
myForm, 28
navigation, 263
next, 267
order, 237
prev, 267
result, 56, 90
rightPanel, 215
selectLanguage, 245
stateList, 256
suggest, 268
suggestions, 268
tip, 37
total, 97
townList, 256
year, 97
instrukcja echo, 56, 65
interaktywne aplikacje WWW, 13
interpreter PHP, 115

J

język
HTML, 83
PHP, 84
XML, 83
jQuery, 13, 39
JSON, 14
korzystanie z danych, 122
odczytywanie danych, 117
parsowanie ciągu znaków, 127
przechwytywanie błędów, 120, 127
stałe, 117
tworzenie danych, 115
typy danych, obiekt, 114
typy danych, tablica, 114
typy danych,ciąg znaków, 114
typy danych,liczba, 114
JSONP, 292

K

karta aktywna, 224
karta elementy główne, 222
karta ustawienia domyślne, 230
karty, 221
klasa
active, 185
autosuggest, 268
cross, 174
DOMAttr, 104
DOMDocument, 84, 101, 162
DOMElement, 104
dragMe, 47
hide, 230
highlight, 133
hoverMe, 37
last, 234
menu, 216
menuHeader, 205
menuItem, 205, 216
MySQLi, 244
next, 236
number, 144
numeric, 142
prev, 236
RegExp, 136
required, 137, 142
round, 174
SimpleXMLElement, 88
site, 147
tabContent, 225
toggle, 32
klucz
HTTP_X_REQUESTED_WITH, 61
valid, 152
what, 56
komunikat Formularz poprawny, 144
konstruktor, 245
konstruktor klasy, 88, 104

L

LIBXML_NOBLANKS, 101
lista wypunktowana, 32, 205, 245

Ł

ładowania plików na żądanie, 79

M

menu, 201
  harmonijkowe, 209
  pływające, 216
  rozwijane, 202, 205
  zmieniające kolor tła, 206
metoda
  $(document).scrollTop(), 220
  $.each(), 31
  $objXML.load(), 101
  .bind(), 20, 24
  .css(), 22
  .live(), 25
  .load(), 18
  .ready(), 18
  .unbind(), 20, 24
  abort(), 275
  ajax(), 65, 82
  ajaxError(), 77
  alert(), 58
  animate(), 220
  appandChild(), 104
  append(), 132
  asXML(), 93
  createElement(), 104
  css(), 50
  data(), 152
  delegate(), 112
  die(), 26
  each(), 144
  fadeIn(), 39
  fadeOut(), 39
  fetch_assoc(), 245, 279
  find(), 112
  flickr.photos.search(), 286
  GET, 59
  get(), 53, 65, 215
  getElementsByTagName(), 101, 108
  getJSON(), 126
  getScript(), 76, 81
  hover(), 39
  json_encode(), 279
  live(), 132, 274
  next(), 221
  parseJSON(), 127
  POST, 59
  post(), 62, 65
  query(), 245, 250
  removeChild(), 109
  save(), 104, 162
  serialize(), 59
  serializeArray(), 59
  slideDown(), 205
  slideUp(), 185, 205
  toggleClass(), 112
  toggleSlide(), 185
  xpath(), 95
metody wyższego poziomu, 76
modyfikowanie wartości węzłów, 93
MySQL, 239

N

nagłówek
  X-Requested-With, 61
  XMLHttpRequest, 61
nagłówek karty, 222
nakładka, 180
narzędzie phpMyAdmin, 240
natychmiastowe ładowanie strony, 69

O

obiekt
  $book, 93, 108
  $quote, 108
  $story, 108
  $title, 108
  DOMNodeList, 101
  event, 29
  jQuery.fn, 301
  LibXMLError, 88
  mysqli, 245
  photo, 287
  settings, 302
  SimpleXMLElement, 87

obliczenia aktualnej pozycji strony, 297
obsługa błędów, 73
odwiązywanie elementów, 20
okienko podpowiedzi, 39
opcja
  cache, 78
  duration, 191
  queue, 191

P

parametr
  alt, 292
  api_key, 286
  end, 301
  farmId, 287
  format, 287
  id, 287
  nojsoncallback, 287
  numComments, 293
  pageNumber, 267
  per_page, 287
  q, 292
  rating, 293
  secret, 287
  serverId, 287
  size, 287
  start, 301
  step, 301
  tags, 287
  thumbnail, 293
  thumbnailHeight, 293
  thumbnailWidth, 293
  videoURL, 293
pasek narzędzi Web developer, 317
PHP, 13
plik
  calculate.php, 195
  check.php, 60, 259
  common.xml, 85
  data.php, 54, 214, 295
  effects.core.js, 186
  effects.highlight.js, 186
  error.html, 29
  feed.php, 304
  jquery.js, 115, 247, 290
  json.php, 124
  main.css, 168
  result.php, 243
  tags.php, 277
  validate.php, 165
pływające okienko, 188
pobieranie danych z PHP, 53
pole
  ID, 245
  languageName, 245
  textarea, 246
  wyboru, 20
pozycja
  bezwzględna, 273
  typu absolute, 50
  względna, 273
  zaznaczonego tekstu, 44
prawidłowa budowa dokumentu XML, 84
product, 237
programowe wysyłanie danych, 28
przeciąganie elementów, 47
przekazywanie metodzie .ready() funkcji, 19
przerywanie żądań AJAX, 66
przestrzeń nazw game, 170
przesyłanie formularza, 27
przezroczystość menu, 209
przycisk
  input, 26
  submit, 27
przykład odpowiedzi serwera Flickr, 282

Q

quantity, 237

R

rodzaje błędów, stałe w PHP, 122
rozszerzenie MySQLi, 240

S

sekcje harmonijki, 184
selektor
  .container:visible, 185
  .toggle:checked, 35
  dwukropek (:), 36
  gt, 36

selektory jQuery, 32
serwer Flickr, 286
SimpleXML, 14
składnia HEREDOC, 119
skrypt PHP validate.php, 164
slideToggle, 42
span, 137, 162, 216
struktura tablicy $booksInfo, 198
style CSS, 32, 168, 203, 276
superglobalna tablica $_GET, 91
superglobalna tablica $_SERVER, 61

T

tabela users, 260
tablica
  $_GET, 240
  $_POST, 107, 240
  $_POST['browser'], 162
  $_POST['sites'], 133
  $arr, 279
  $booksInfo, 198
  $errorArray, 156
  $names, 56
  $result, 279
  asocjatywna, 245
tekst zaznaczony przez użytkownika, 43
tworzenie kreatora, 231

U

uniksowy znacznik czasu, 163
usuwanie pozycji z koszyka, 199
usuwanie węzłów, 109

W

WampServer, 15
wartość
  $_POST['save'], 156
  $find, 257
  absolute, 37, 189
  currentPage, 267
  event.which, 274
  false, 28
  fast, 221
  information, 257
  json-in-script, 292
  klucza find, 257
  none, 37
  normal, 221
  scrollTop, 191
  slow, 221
  states, 257
  stories, 93
  true, 28
węzły, 162
  podrzędne, 101
wiązanie
  danych, 239
  elementów, 20
  metoda skrócona, 23
  wielu zdarzeń, 23
właściwość
  attributes, 101
  dataType, 111
  document.selection, 45
  firstChild, 101
  increasing, 302
  item, 101
  message, 88
  nodeName, 102
  nodeType, 102
  nodeValue, 101, 102
  pageX, 39
  pageY, 39
  position, 189
  selectionEnd, 46
  selectionStart, 46
  which, 42
  window.getSelection, 45
włączenie menu przeglądarki, 42
wskaźnik myszy, przechwytywanie zdarzeń 36
współrzędna wskaźnika myszy, 50
wtyczka
  cashCounter, 301
  Google Page Speed, 312
  Yahoo! YSlow, 312
  easing, 186
  biblioteki jQuery, 298
wyłączenie domyślnego zachowania przeglądarki, 42

wyrażenie
$('container').hide(), 214
$(document).height(), 297
$(window).scrollTop(), 297
//book/name[], 98
__construct(), 245
regularne, 147
wyszukiwanie tekstu, 133
wyświetlanie i ukrywanie podmenu, 204
wyświetlanie kanałów RSS, 302
wyświetlanie podpowiedzi, 37
wywołanie $.get()
dane, 56
funkcja obsługująca, 56
typ danych, 56
URL, 56
wywołanie funkcji zwrotnej, 81
wywoływanie zdarzeń, 23
wzorzec adresu URL dla Flickr, 287
wzorzec adresu URL dla YouTube, 292

X

XML, 14
dodawanie elementów, 93
ładowanie danych, 86
modyfikowanie dokumentów, 93, 105
odczytywanie dokumentów, 98
parsowanie dokumentów, 109
tworzenie nowych dokumentów, 102
wyszukiwanie elementów, 94
XMLHttpRequest, 52
XPath, 94

Y

YFT, 186

Z

zapis $(this), 22
zapytanie
DELETE, 250
INSERT, 247
SELECT, 250
SHOW, 250
UPDATE, 250
zastępnik, 29
zdarzenie, 17
blur, 22
change, 21
click, 21, 24, 25, 152
dblclick, 24
focus, 22, 151
hover, 208
keydown, 22, 40
keypress, 24
keyup, 22, 40, 151, 273
load, 24
mousedown, 24, 48
mousemove, 24, 39, 48
mouseout, 24
mouseover, 24, 274
mouseup, 24
onkeydown, 262
przeciąganie, 18
przewijanie okna, 190
scroll, 24, 191
select, 24
skróty klawiszowe, 18, 39
submit, 24, 247
success, 56
unload, 24
zmienna
$_COOKIE, 162
$action, 92
$bookId, 92
$objXML, 92
$result, 245
$resultStr, 245
$strResponse, 93
altPressed, 42
base, 188
changeBy, 302
currentTabIndex, 236
dataValid, 140, 152
defaultOffset, 191
divLeft, 50
divTop, 50
emailPattern, 147
from, 267
interval, 188
mousex, 50
mousey, 50

zmienna
moviesPerPage, 267
offsetTop, 191
page, 215
player, 175
posts, 307
searchText, 136
target, 256
textOnPage, 46
to, 267
totalMovies, 267
url, 256
who, 174
xhr, 275
zmienne globalne, 50
jsonResult, 126
znacznik
form, 155
img, 31, 176
script, 292
znaczniki w harmonijce, 185

Ż

żądania
AJAX, 54, 251, 293
pomiędzy domenami, 282
typu GET, 65
typu POST, 65
AJAX do skryptu search.php, 284

PHP i jQuery. Receptury

Język PHP jest podstawowym językiem wybieranym przez twórców stron internetowych, a jQuery — jedną z najczęściej stosowanych bibliotek w sieci. To oczywiste: obie technologie są lekkie, łatwe w użyciu i nauce, a przy tym oferują ogromne możliwości tworzenia dynamicznych witryn i interaktywnych aplikacji WWW. W dodatku razem tworzą doskonale uzupełniający się zestaw wszechstronnych narzędzi dla webmasterów. Jednak w świecie informatyki nic nie jest ani doskonałe, ani dziecinnie proste — dlatego nawet w pracy z takim tandemem możesz napotkać pewne często powtarzające się trudności, które czasem znacznie opóźniają realizację projektu. By tego uniknąć, wykorzystaj ten zbiór ponad sześćdziesięciu prostych, ale wyjątkowo skutecznych receptur i rozwiązań, niezwykle przydatnych przy tworzeniu interaktywnych aplikacji WWW.

W tej przejrzyście napisanej książce znajdziesz wybór najważniejszych zadań i problemów, a także czytelnie przygotowane instrukcje radzenia sobie z nimi. Dzięki temu będziesz mógł jeszcze szybciej i sprawniej tworzyć aplikacje WWW z wykorzystaniem PHP i jQuery, nawet jeśli jesteś początkującym programistą lub webmasterem. Niezależnie od tego, czy chcesz nauczyć się na bieżąco kontrolować dane z formularzy, tworzyć wtyczki, przeciągać elementy, tworzyć atrakcyjne menu i przyjazne formularze, korzystać z API YouTube, czy współpracować z bazą danych — wystarczy sięgnąć po właściwe rozwiązania. Znajdziesz tu również receptury buforowania żądań AJAX oraz obsługi błędów, a także kilka zaawansowanych technik wykorzystania PHP i jQuery do tworzenia bardziej rozbudowanych stron. Dowiesz się między innymi, jak obejść ograniczenia przeglądarek, takie jak żądania przesyłane między domenami, i jak wykorzystać narzędzie Firebug.

Dzięki tej książce:

- zaczniesz od podstaw, aby na koniec poznać triki profesjonalnych twórców stron
- przygotujesz interaktywne, dynamiczne i hierarchiczne menu
- zastosujesz ciekawe efekty specjalne do elementów strony
- wykorzystasz bazę danych w kodzie PHP i jQuery
- za pomocą technologii AJAX poprawisz interakcję użytkownika ze stroną
- dowiesz się, jak wykorzystać formaty XML i JSON do skutecznej wymiany danych
- przygotujesz różne narzędzia do budowania aplikacji WWW
- skontrolujesz dane z formularzy zarówno po stronie klienta, jak i serwera

Wykorzystaj wszystkie możliwości tkwiące w technologiach PHP i jQuery!
Poznaj rozwiązania typowych problemów, które możesz napotkać!

helion.pl księgarnia internetowa
Nr katalogowy: 6853
Księgarnia internetowa: http://helion.pl
Zamówienia telefoniczne: 0 801 339900, 0 601 339900

Helion
Sprawdź najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-246-3350-0

Cena 57,00 zł