

ZACZNIJ TAM, GDZIE INNI KOŃCZĄ!

# PHP

# Zaawansowane programowanie

Peter MacIntyre, Brian Danchilla, Mladen Gogala

Tytuł oryginału: Pro PHP Programming

Tłumaczenie: Jakub Hubisz

ISBN: 978-83-246-3922-9

Original edition copyright 2011 by Peter MacIntyre, Brian Danchilla, and Mladen Gogala.  
All rights reserved.

Polish edition copyright 2012 by HELION SA.  
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Wydawnictwo HELION dołożyło wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION  
ul. Kościuszki 1c, 44-100 GLIWICE  
tel. 32 231 22 19, 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:  
<ftp://ftp.helion.pl/przyklady/phpzap.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres  
<http://helion.pl/user/opinie/phpzap>  
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# Spis treści

O autorach .....	11
O korektorze merytorycznym .....	13
Przedmowa .....	15
Wprowadzenie do PHP .....	17
<b>Rozdział 1. Obiektość .....</b>	<b>21</b>
Klasy .....	21
Dziedziczenie i przeciążanie .....	23
„Magiczne” funkcje .....	27
Metody __get i __set .....	27
Metoda __isset .....	28
Metoda __call .....	28
Metoda __toString() .....	29
Kopiowanie, klonowanie oraz porównywanie obiektów .....	29
Interfejsy, iteratory i klasy abstrakcyjne .....	31
Kontekst klasy i elementy statyczne .....	35
Podsumowanie .....	36
<b>Rozdział 2. Wyjątki i referencje .....</b>	<b>37</b>
Wyjątki .....	37
Referencje .....	41
Podsumowanie .....	45
<b>Rozdział 3. Mobilne PHP .....</b>	<b>47</b>
Różnorodność urządzeń .....	47
Rozpoznanie urządzenia .....	48
Aplikacja kliencka .....	48
Wbudowane funkcje PHP .....	48
Rozpoznawanie możliwości urządzenia .....	51
WURFL .....	51
Tera-WURFL .....	57

Narzędzia renderujące .....	60
WALL .....	60
Reagujący CSS .....	62
Emulatory i SDK .....	62
Tworzenie dla systemu Android .....	62
Adobe Flash Builder dla PHP .....	62
Kody QR .....	63
Podsumowanie .....	64
<b>Rozdział 4. Media społecznościowe .....</b>	<b>65</b>
OAuth .....	65
Twitter .....	66
API publicznego wyszukiwania .....	66
Prywatne REST API .....	67
Wykorzystanie mechanizmu OAuth w celu powiązania strony z systemem logowania .....	77
Dodatkowe metody API i przykłady jego wykorzystania .....	80
Facebook .....	83
Dodanie linku wylogowania z Facebooka .....	88
Żądanie dodatkowych uprawnień .....	89
Graph API .....	89
Podsumowanie .....	91
<b>Rozdział 5. Nowości technologiczne .....</b>	<b>93</b>
Przestrzenie nazw .....	93
Przestrzenie nazw i autoładowanie .....	96
Przestrzenie nazw — podsumowanie .....	96
Funkcje anonimowe .....	96
Nowdoc .....	97
Lokalne instrukcje goto .....	100
Standardowa biblioteka PHP — SPL .....	100
SPL — podsumowanie .....	103
Rozszerzenie phar .....	103
Podsumowanie .....	106
<b>Rozdział 6. Tworzenie formularzy i zarządzanie nimi .....</b>	<b>107</b>
Walidacja danych .....	107
Wczytywanie plików i obrazów .....	113
Konwersja obrazów i miniatury .....	114
Wyrażenia regularne .....	115
Integracja języków .....	118
Podsumowanie .....	119
<b>Rozdział 7. Integracja z bazami danych. Część I .....</b>	<b>121</b>
Wprowadzenie do MongoDB .....	122
Zapytania w MongoDB .....	126
Modyfikowanie dokumentów w MongoDB .....	130
Agregacje w MongoDB .....	132
Podsumowanie MongoDB .....	134

Wprowadzenie do CouchDB .....	134
Wykorzystanie interfejsu Futon .....	135
Podsumowanie CouchDB .....	140
Wprowadzenie do SQLite .....	141
Podsumowanie SQLite .....	149
Podsumowanie .....	149
<b>Rozdział 8. Integracja z bazami danych. Część II .....</b>	<b>151</b>
Wprowadzenie do rozszerzenia MySQLi .....	151
Podsumowanie rozszerzenia MySQLi .....	158
Wprowadzenie do PDO .....	158
Podsumowanie PDO .....	161
Wprowadzenie do ADOdb .....	161
Podsumowanie ADOdb .....	165
Wyszukiwanie pełnotekstowe przy wykorzystaniu Sphinxsa .....	165
Podsumowanie .....	173
<b>Rozdział 9. Integracja z bazami danych. Część III .....</b>	<b>175</b>
Wprowadzenie do Oracle .....	175
Podstawy. Połączenie i wykonywanie zapytań .....	177
Interfejs tablicowy .....	180
Procedury i kursory w PL/SQL .....	183
Praca z typami LOB .....	186
Inne podejście do połączeń — pule połączeń .....	190
Zestawy znaków w bazie danych i PHP .....	192
Podsumowanie .....	193
<b>Rozdział 10. Biblioteki .....</b>	<b>195</b>
SimplePie .....	196
TCPDF .....	199
Pobieranie danych ze stron internetowych .....	204
Integracja z Mapami Google .....	209
Wiadomości e-mail i SMS .....	211
gChartPHP — biblioteka wykorzystująca Google Chart API .....	215
Podsumowanie .....	219
<b>Rozdział 11. Bezpieczeństwo .....</b>	<b>221</b>
Nigdy nie ufaj danym .....	221
register_globals .....	222
Białe i czarne listy .....	222
Dane formularzy .....	223
\$_COOKIES, \$_SESSION i \$_SERVER .....	224
Żądania Ajax .....	224
Powszechne ataki .....	225
Polityka tego samego pochodzenia .....	225
XSS (Cross Site Scripting) .....	225
CSRF (Cross-Site Request Forgery) .....	228
Sesje .....	229
Zapobieganie atakom typu SQL injection .....	229

Wyrażenia filtrujące .....	230
Plik php.ini i ustawienia serwera .....	233
Środowisko serwerowe .....	233
Zabezpieczanie pliku php.ini .....	234
Algorytmy hasel .....	235
Podsumowanie .....	236
<b>Rozdział 12. Programowanie zwinne z wykorzystaniem Zend Studio dla Eclipse, Bugzilli, Mylyn i Subversion .....</b>	<b>237</b>
Zasady programowania zwinnego .....	237
Rajd programowania zwinnego .....	238
Wprowadzenie do programu Bugzilla .....	239
Mylyn dla Eclipse .....	240
Bugzilla i Mylyn w połączeniu z Eclipse .....	242
Maksymalizowanie korzyści .....	245
Podsumowanie .....	246
<b>Rozdział 13. Refaktoryzacja, testy jednostkowe i ciągła integracja .....</b>	<b>249</b>
Refaktoryzacja .....	249
Niewielka refaktoryzacja .....	250
Większy przykład .....	253
Testy jednostkowe .....	265
Ciągła integracja .....	279
Serwer ciągłej integracji .....	280
System kontroli wersji .....	280
Analiza statyczna .....	281
Budowanie automatyzacji .....	282
Uruchomienie serwera Jenkins .....	282
Podsumowanie .....	285
<b>Rozdział 14. XML .....</b>	<b>287</b>
Podstawy XML .....	287
Schematy .....	288
SimpleXML .....	289
Parsowanie XML z tekstu .....	289
Parsowanie XML z pliku .....	290
Przestrzenie nazw .....	294
RSS .....	296
Generowanie dokumentów XML za pomocą SimpleXML .....	298
DOMDocument .....	303
XMLReader i XMLWriter .....	305
Podsumowanie .....	306
<b>Rozdział 15. JSON i Ajax .....</b>	<b>307</b>
JSON .....	308
PHP i JSON .....	309
Ajax .....	312
Tradycyjny model WWW .....	313
Model Ajax .....	313
Zdarzenia synchroniczne kontra asynchroniczne .....	315

Obiekt XMLHttpRequest .....	316
Wykorzystanie obiektu XMLHttpRequest .....	317
API JavaScript wyższego poziomu .....	322
Przykłady jQuery .....	322
Przesyłanie danych z Ajaksa do skryptu PHP .....	327
Prosty program graficzny .....	328
Utrzymanie stanu .....	330
Podsumowanie .....	335
<b>Rozdział 16. Konkluzja .....</b>	<b>337</b>
Zasoby .....	337
www.php.net .....	337
www.zend.com .....	338
devzone.zend.pl .....	338
www.phparch.com .....	338
Konferencje .....	339
Certyfikacja PHP .....	340
Podsumowanie .....	341
<b>Dodatek Wyrażenia regularne .....</b>	<b>343</b>
Składnia wyrażeń regularnych .....	343
Przykłady wyrażeń regularnych .....	344
Opcje wewnętrzne .....	347
Chciwość .....	347
Funkcje wykorzystujące wyrażenia regularne .....	348
Zamiana ciągów — preg_replace .....	348
Inne funkcje .....	350
<b>Skorowidz .....</b>	<b>353</b>





## ROZDZIAŁ 3



# Mobilne PHP

Tworzenie aplikacji mobilnych staje się z roku na rok coraz popularniejsze. O zwiększenie udziału w rynku walczą iPhone, Android czy BlackBerry. Każdy producent smartfona potrzebuje aplikacji dla swojego urządzenia, by przyciągnąć jak najwięcej użytkowników. Ponadto istnieją tablety, takie jak iPad, PlayBook i Galaxy, oraz czytniki, np. Kindle czy Nook. Nawet standardowe telefony komórkowe mają przeglądarki i różne dodatki.

Na każdym urządzeniu mobilnym posiadającym dostęp do internetu można przeglądać strony internetowe i uruchamiać aplikacje utworzone w technologii PHP. Dlatego potrzebny jest sposób sensownego prezentowania zawartości stron na mniejszych urządzeniach. Z tego rozdziału dowiesz się, jak rozpoznać urządzenie klienckie za pomocą żądania HTTP, poznasz WURFL i Tera-WURFL.

Obecnie działają tysiące urządzeń mobilnych umożliwiających przeglądanie stron internetowych. Może się wydawać, że tworzenie oprogramowania dla starszych przeglądarek jest trudne, ale urządzenia mobilne są jeszcze mniej ustandaryzowane. Na szczęście są dostępne systemy pomocne w procesie tworzenia takiego oprogramowania. Mając na uwadze renderowanie na urządzenia mobilne, pokażemy, jak sprawić przy użyciu WALL, by znaczniki były bardziej abstrakcyjne, jak automatycznie zmieniać rozmiar obrazków i spowodować, by CSS był bardziej płynny.

Zaprezentujemy także emulatory urządzeń, omówimy tworzenie aplikacji PHP na urządzenia z Androidem i program Flash Builder dla PHP.

## Różnorodność urządzeń

Podczas pracy z urządzeniami mobilnymi jednym z największych wyzwań jest zapewnienie czytelności strony po jej wyrenderowaniu. Przy tworzeniu aplikacji internetowych na komputery osobiste sprawdzamy najpopularniejsze przeglądarki, takie jak Chrome, Firefox, Safari, Opera czy Internet Explorer, i być może inne systemy, np. Windows XP, Windows 7, Linux bądź Mac OS X. Zapewnienie wsparcia dla różnych kombinacji przeglądarek i systemów może być pracochłonne.

W przypadku urządzeń mobilnych renderowanie jest jeszcze mniej ustandaryzowane i dużo bardziej złożone. Na przykład prawie wszystkie współczesne komputery osobiste umożliwiają wyświetlanie tysięcy kolorów i zapewniają rozdzielczość minimum 800 na 600 pikseli. Jednakże telefony komórkowe, smartfony, tablety, czytniki e-booków i inne urządzenia mobilne mogą mieć ograniczoną paletę kolorów lub tylko skalę szarości. Rozmiary wyświetlaczy także znacznie się różnią. To są tylko trzy parametry — istnieją setki innych, którymi mogą się różnić poszczególne urządzenia. Omówimy kilka z tych parametrów w dalszej części tego rozdziału.

W przeciwieństwie do tworzenia stron przeznaczonych dla komputerów stacjonarnych naiwne próby programowania dla każdego urządzenia oddzielnie są nierealne lub przynajmniej zajęłyby zbyt wiele czasu i pracy, niż ktokolwiek byłby skłonny na to poświęcić. Zamiast tego omówimy systemy pozwalające na określenie urządzenia i wyrenderowanie strony dynamicznie oraz płynne zmienianie stylów CSS.

## Rozpoznanie urządzenia

Pierwszym krokiem różnicowania zawartości strony jest sprawdzenie, dla jakiego urządzenia ta strona będzie renderowana. Przeanalizujemy kilka technik pozwalających na ustalenie, jakie urządzenie jest używane.

### Aplikacja kliencka

U podstaw każdego systemu detekcji urządzeń leży nagłówek aplikacji klienckiej wysyłany w standardowym zapytaniu HTTP. W PHP możemy uzyskać dostęp do informacji o aplikacji klienckiej dzięki superglobalnej zmiennej serwerowej `$_SERVER['HTTP_USER_AGENT']`. Nagłówek takiej aplikacji może zawierać informacje o przeglądarce, silniku renderującym i systemie operacyjnym. Ma postać podobną do poniższego, wygenerowanego dla przeglądarki Firefox 4:

```
Mozilla/5.0 (Windows NT 5.1; rv:2.0) Gecko/20100101 Firefox/4.0
```

Z tego nagłówka możemy wyczytać, że systemem operacyjnym klienta jest Windows, silnikiem renderującym jest Gecko, natomiast przeglądarką jest Firefox w wersji 4.0.

- 
- **UWAGA.** Rozpoznawanie urządzeń nie jest pewne. Nagłówki dla dwóch różnych urządzeń mogą nie być unikalne, chociaż rzadko tak bywa; mogą także być fałszowane, co zostanie omówione w dalszej części rozdziału.
- 

### Wbudowane funkcje PHP

PHP ma funkcję `get_browser`, która próbuje uzyskać informacje o wykorzystywanej przeglądarce. Funkcja działa na podstawie pliku *browscap.ini*. W tym kontekście jest jak prostsza, ograniczona wersja systemu WURFL, który będzie omówiony dalej.

- 
- **UWAGA.** Ta funkcja wymaga zainstalowanego w systemie pliku *browscap.ini* oraz ustawienia ścieżki do niego w pliku *php.ini*, na przykład:

```
browscap = "C:\twoja\sciezka\do\browscap.ini"
```

Więcej informacji o funkcji `get_browser` można uzyskać pod adresem <http://php.net/manual/pl/function.get-browser.php>, natomiast aktualne pliki *browscap.ini* pod adresem <http://browsers.garykeith.com/downloads.asp>.

---

Jeżeli pierwszy parametr będzie ustawiony na wartość `null` lub przekazany zostanie nagłówek aplikacji klienckiej, uzyskamy informacje o przeglądarce. Możemy także przekazać inny nagłówek, aby uzyskać informacje o nim. Drugi parametr jest opcjonalny. Ustawienie go na wartość `true` spowoduje zwrócenie wyniku w postaci tabeli zamiast domyślnego obiektu (listingi 3.1 i 3.2).

#### Listing 3.1. Wykorzystanie funkcji `get_browser`

```
<?php
echo $_SERVER ['HTTP_USER_AGENT'] . "\n\n";
var_dump ( get_browser ( null, true ) );
//Równie dobrze mogliśmy przekazać nagłówek aplikacji klienckiej jako pierwszy parametr
//var_dump ( get_browser ( $_SERVER ['HTTP_USER_AGENT'], true ) );
?>
```

#### Listing 3.2. Wynik dla przeglądarki Chrome

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.65
Safari/534.24
```

```

array
  'browser_name_regex' => string '$^.*$$' (length=6)
  'browser_name_pattern' => string '*' (length=1)
  'browser' => string 'Default Browser' (length=15)
  'version' => string '0' (length=1)
  'majorver' => string '0' (length=1)
  'minorver' => string '0' (length=1)
  'platform' => string 'unknown' (length=7)
  'alpha' => string '' (length=0)
  'beta' => string '' (length=0)
  'win16' => string '' (length=0)
  'win32' => string '' (length=0)
  'win64' => string '' (length=0)
  'frames' => string '1' (length=1)
  'iframes' => string '' (length=0)
  'tables' => string '1' (length=1)
  'cookies' => string '' (length=0)
  'backgroundsounds' => string '' (length=0)
  'cdf' => string '' (length=0)
  'vbscript' => string '' (length=0)
  'javaapplets' => string '' (length=0)
  'javascript' => string '' (length=0)
  'activexcontrols' => string '' (length=0)
  'isbanned' => string '' (length=0)
  'ismobiledevice' => string '' (length=0)
  'issyndicationreader' => string '' (length=0)
  'crawler' => string '' (length=0)
  'cssversion' => string '0' (length=1)
  'supportscss' => string '' (length=0)
  'aol' => string '' (length=0)
  'aolversion' => string '0' (length=1)

```

Jak widać, funkcja nie zwraca żadnych informacji dla tej nowej przeglądarki. Dzieje się tak dlatego, że plik *browscap.ini* załączony do serwera WAMP (Windows, Apache, MySQL, PHP) ma już ponad rok. Rozwiązaniem problemu jest pobranie aktualnej wersji pliku. Może także być potrzebny restart serwera, jeżeli plik zostanie złożony w pamięci podręcznej. Po aktualizacji pliku otrzymamy dokładniejsze informacje, pokazane na listingu 3.3.

**Listing 3.3.** Wynik dla przeglądarki Chrome po aktualizacji pliku *browscap.ini*

```

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.65
Safari/534.24
array
  'browser_name_regex' => string '$^mozilla/5\.0 \\..*windows nt 6\.1.*wow64.*\)'
  ↪applewebkit/.*\.(khtml, like gecko).*chrome/11\..*safari/.*$' (length=108)
  'browser_name_pattern' => string 'Mozilla/5.0 (*Windows NT 6.1*WOW64*) AppleWebKit/* (KHTML,
  ↪like Gecko)*Chrome/11.*Safari/*' (length=90)
  'parent' => string 'Chrome 11.0' (length=11)
  'platform' => string 'Win7' (length=4)
  'win32' => string '' (length=0)
  'win64' => string '1' (length=1)
  'browser' => string 'Chrome' (length=6)
  'version' => string '11.0' (length=4)
  'majorver' => string '11' (length=2)
  'frames' => string '1' (length=1)
  'iframes' => string '1' (length=1)
  'tables' => string '1' (length=1)
  'cookies' => string '1' (length=1)

```

```
'javascript' => string '1' (length=1)
'javaapplets' => string '1' (length=1)
'cssversion' => string '1' (length=1)
'minorver' => string '0' (length=1)
'alpha' => string '' (length=0)
'beta' => string '' (length=0)
'win16' => string '' (length=0)
'backgroundsounds' => string '' (length=0)
'vbscript' => string '' (length=0)
'activexcontrols' => string '' (length=0)
'isbanned' => string '' (length=0)
'ismobiledevice' => string '' (length=0)
'issyndicationreader' => string '' (length=0)
'crawler' => string '' (length=0)
'aolversion' => string '0' (length=1)Using Regex
```

Jeżeli zależy Ci na sprawdzeniu tylko kilku głównych urządzeń mobilnych, możesz wykorzystać wyrażenia regularne w celu przeszukania nagłówka aplikacji klienckiej. Na listingu 3.4 sprawdzamy, czy zapytanie przyszło z jednego z popularnych telefonów. Jeżeli łańcuch zostanie znaleziony, przekierowujemy żądanie do odrębnej strony przeznaczonej dla urządzeń mobilnych oraz ładujemy alternatywny szablon i arkusz stylów. Opcja `/i` w wyrażeniu regularnym (Regex) powoduje, że nasze zapytanie ignoruje wielkość znaków. Znak `|` oznacza „lub” — zostanie znaleziony zarówno łańcuch „iPhone”, jak i „iPad”, ale nie „iPod”. Podobnie będą znalezione „windows ce” oraz „windows phone”, ale nie „windows xp”. Zajrzyj do dodatku „Wyrażenia regularne”.

**Listing 3.4.** Wykorzystanie wyrażen regularnych w celu zweryfikowania urządzeń mobilnych

```
<?php
if (preg_match ( '/i(Phone|Pad)|Android|Blackberry|Symbian|windows (ce|phone)/i' ,
    $_SERVER ['HTTP_USER_AGENT'] ) ) {
    //przekierowanie, załadowanie innych szablonów, arkuszy stylów
    header ( "Location: mobile/index.php" ) ;
}
?>
```

Aby wykryć więcej urządzeń, potrzebujemy znacznie większego wyrażenia regularnego. Możemy skorzystać z popularnej strony <http://detectmobilebrowsers.com/>, pozwalającej generować wyrażenie regularne w kilku różnych językach programowania i dla różnych frameworków. Wygenerowany skrypt przekieruje klienta na witrynę przeznaczoną dla urządzeń mobilnych. Listing 3.5 pokazuje przykładowy skrypt wygenerowany przez wyżej wymienioną stronę.

**Listing 3.5.** Wyrażenie regularne wygenerowane przez stronę [detectmobilebrowsers.com](http://detectmobilebrowsers.com/)

```
<?php
$useragent=$_SERVER['HTTP_USER_AGENT'];
if(preg_match('/android.+mobile|avantgo|bada\b|blackberry|blazer|compal|elaine|fennec|hiptop|iemo-
bile|ip(hone|od)|iris|kindle|lge |maemo|midp|mmp|opera m(ob|in)i|palm(
os)?|phone|p(ixi|re)\|plucker|pocket|psp|symbian|treo|up\.(browser|link)|vodafone|wap|windows
(ce|phone)|xda|xiino/i',$useragent)|preg_match('/1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a
wa|abac|ac(er|oo|s\-
)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-m|r |s
)|avan|be(ck|ll|ng)|bi(1b|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-(n|u)|c55\|capi|ccwa|cdm\-
|cell|chtm|cldc|cmd\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-
d)|e1(49|ai)|em(12|u1)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|_)g1 u|g560|gene|gf\5|g\-
mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-\hi(pt|ta)|hp( i|ip)|hs\-\c|ht(c(\-|
|_|a|g|p|s|t)|tp)|hu(aw|tc)|i\-(20|go|ma)|i230|iac( |\-
|\)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt( |\)|k1on|kpt
|kwc\-\|kyo(c|k)|le(no|xi)|lg( g|v|(k|l|u)|50|54|e\-\e\|a-w)|l1bw|lynx|m1\-
w|m3ga|m50\|ma(te|u|x)|mc(01|21|ca)|m\-\cr|me(di|rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-
```

```
|o|v|)|zz|)|mt(50|p1|v |)|mwbp|mywa|n10[0-2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\-
|on|t|f|w|g|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\-([1-
8]|c)|)|phil|pire|p1(ay|uc)|pn\2|po(ck|rt|se)|prox|psio|pt\g|qa\-\a|qc(07|12|21|32|60|\-[2-7]|i\-\
)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\|sa(ge|ma|mm|ms|ny|va)|sc(01|h\-\oo|p\-\)|sdk\|se(c(\-
|0|1)|47|mc|nd|ri)|sgh\-\|shar|sie(\-|m)|sk\-\0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-\|v\-\
|v |)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-\|tdg\-\|tel(i|m)|tim\-\|t\-\
mo|to(p1|sh)|ts(70|m\-\|m3|m5)|tx\-\9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-3]|v\-\
v)|vm40|voda|vu1c|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-| |)|webc|whit|wi(g
|nc|nw)|wm1b|wonu|x700|xda(\-|2|g)|yas\-\|your|zeto|zte\-\|i'|,substr($useragent,0,4))
header('Location: http://detectmobilebrowser.com/mobile');
?>
```

To rozwiązanie będzie poprawne tylko w niektórych przypadkach. Aby uzyskać dokładniejsze wyniki oraz rozpoznać możliwości urządzenia, potrzebny jest zaawansowany system. Ten system to WURFL, omówiony w kolejnym podrozdziale.

## Rozpoznawanie możliwości urządzenia

System WURFL pozwala wyjść poza proste wykrywanie rodzaju urządzenia i ustalić, jakie są jego możliwości.

### WURFL

Wireless Universal Resource FiLe (WURFL) jest plikiem XML opracowanym przez Luca Passanigo, zawierającym informacje o możliwościach urządzeń mobilnych.

### Wprowadzenie

Aktualnie w pliku WURFL jest ponad pięćset różnych właściwości urządzeń mobilnych. Implementacje WURFL zostały opracowane w wielu językach i na wiele platform, włączając w to Javę i .NET. Dla PHP oficjalne API to *The New PHP WURFL API*, dostępne pod adresem <http://wurfl.sourceforge.net/nphp/>.

Właściwości urządzeń są ułożone hierarchicznie. Jeżeli właściwość nie jest wyszczególniona dla danego modelu, to sprawdzany jest ogólniejszy typ urządzenia. Jeśli właściwość ponownie nie zostanie znaleziona, WURFL sprawdza kolejny ogólniejszy typ urządzenia i powtarza ten proces, dopóki nie dotrze do korzenia pliku. Struktura hierarchiczna oszczędza przestrzeń na dysku i przyspiesza wyszukiwanie. WURFL próbuje także wykorzystywać wersje pliku XML spakowaną za pomocą ZipArchiwe, pakietu dostępnego w PHP od wersji 5.2.0. Ponieważ wersja ZIP pliku ma aktualnie poniżej megabajta (MB), a wersja rozpakowana pliku ponad 16 MB, jest to duże usprawnienie.

Niektóre użyteczne właściwości to rozdzielczość ekranu, kodeki i formaty lub wsparcie dla JavaScriptu, Javy czy Flasha.

- 
- **UWAGA.** Plik XML jest tworzony głównie przez developerów i użytkowników, więc może zawierać błędy. Ponadto na rynek ciągle trafiają nowe urządzenia. Pomimo rozmiarów i kompletności pliku WURFL nie powinniśmy nigdy ufać mu w stu procentach. Jeżeli potrzebujesz szybko wykorzystać informacje o urządzeniu, możesz wyszczególnić jego właściwości oraz umieścić informacje o nim w głównym pliku XML.

Jeżeli dokładność informacji ma kluczowe znaczenie, można użyć systemów o znacznie większej skuteczności.

---

### Instalacja

Na potrzeby wszystkich przykładów z tego rozdziału umieścimy bibliotekę WURFL w katalogu *wurfl*, który będzie się znajdował w głównym katalogu web jako *./wurfl/*. Wykorzystamy standardowy plik konfiguracyjny, a obiekt WURFLManager będziemy za każdym razem pozyskiwać za pomocą kodu z listingu 3.6.

**Listing 3.6.** *Utworzenie obiektu WURFLManager: wurflSetup.php*

```

<?php
error_reporting(E_ALL);
define( "WURFL_DIR", dirname(__FILE__) . '/wurfl/WURFL/' );
define( "RESOURCES_DIR", dirname(__FILE__) . "/wurfl/examples/resources/" );
require_once WURFL_DIR . 'Application.php';
function getWurflManager() {
    $config_file = RESOURCES_DIR . 'wurfl-config.xml';
    $wurfl_config = new WURFL_Configuration_XmlConfig( $config_file );
    $wurflManagerFactory = new WURFL_WURFLManagerFactory( $wurfl_config );
    return $wurflManagerFactory->create();
}
?>

```

**Rozpoznawanie urządzeń za pomocą WURFL**

W naszym pierwszym przykładzie rozpoznawania urządzenia wyświetlimy stos urządzenia przy wykorzystaniu WURFL PHP API. Wyświetlimy hierarchię aplikacji klienckiej z zastosowaniem właściwości fallback oraz id (listing 3.7).

**Listing 3.7.** *Wyświetlenie stosu dla aplikacji klienckiej od szczegółowej do ogólnej*

```

<?php
error_reporting(E_ALL);
require_once('wurflSetup.php');
$wurflManager = getWurflManager();
$urzadzenie = $wurflManager->getDeviceForHttpRequest($_SERVER);
print "<p>Stos Id: <br/>";
while ($urzadzenie != null)
{
    print $urzadzenie->id . "<br/>";
    if (!$urzadzenie->fallBack || $urzadzenie->fallBack == "root")
    {
        break;
    }
    $urzadzenie = $wurflManager->getDevice($urzadzenie->fallBack);
}
print "</p>";
?>

```

Poniżej znajduje się wynik działania skryptu na komputerze stacjonarnym dla przeglądarki Firefox 4:

```

Stos Id:
firefox_1
firefox
generic_web_browser
generic_xhtml
generic

```

oraz dla przeglądarki Chrome:

```

Stos Id:
google_chrome_1
google_chrome
generic_web_browser
generic_xhtml
generic

```

- **UWAGA.** Uruchomienie skryptu po raz pierwszy może zająć dużo czasu, ponieważ WURFL buduje bufor. Konieczne może być zwiększenie wartości dla dyrektywy `max_execution_time` w pliku *php.ini*.

Jeżeli chcemy emulować inne urządzenie, możemy wprowadzić inną zmienną serwerową zawierającą nagłówek aplikacji klienckiej. Zmodyfikowana wersja listingu 3.7 pokazana jest na listingu 3.8. Wynik skryptu przedstawiono na listingu 3.9.

**Listing 3.8.** *Emulowanie innego urządzenia — wprowadzenie innej zmiennej serwerowej*

```
<?php
error_reporting(E_ALL);
require_once('wurflSetup.php');
$wurflManager = getWurflManager();
$_SERVER['HTTP_USER_AGENT'] = "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X; en-us)
↳AppleWebKit/532.9 (KHTML, like Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7";
$urzadzenie = $wurflManager->getDeviceForHttpRequest( $_SERVER );
print "<p>Stos ID: <br/>";
while ( $urzadzenie != null ) {
    print $urzadzenie->id . "<br/>";
    if ( !$urzadzenie->fallBack || $urzadzenie->fallBack == "root" )
    {
        break;
    }
    $urzadzenie = $wurflManager->getDevice( $urzadzenie->fallBack );
}
print "</p>";
?>
```

**Listing 3.9.** *Wynik działania WURFL dla emulowanego iPhone'a 4*

```
Stos ID:
apple_iphone_ver4_sub405
apple_iphone_ver4
apple_iphone_ver3_1_3
apple_iphone_ver3_1_2
apple_iphone_ver3_1
apple_iphone_ver3
apple_iphone_ver2_2_1
apple_iphone_ver2_2
apple_iphone_ver2_1
apple_iphone_ver2
apple_iphone_ver1
apple_generic
generic_xhtml
generic
```

## Rozpoznawanie i wyświetlanie właściwości urządzenia za pomocą WURFL

Na listingu 3.10 pokazano dostępne grupy właściwości, które możemy sprawdzić. Wyświetlimy wszystkie dostępne właściwości dla grup `display` i `css`. Wynik pokazany jest na listingu 3.11.

**Listing 3.10.** *Wyświetlanie dostępnych grup atrybutów*

```
<?php
error_reporting(E_ALL);
require_once('wurflSetup.php');
$wurflManager = getWurflManager();
```

```

$urzadzenie = $wurflManager->getDeviceForHttpRequest( $_SERVER );
$grupy_wlasciwosci = $wurflManager->getListOfGroups();
asort( $grupy_wlasciwosci );
foreach ( $grupy_wlasciwosci as $c ) {
    print $c . "<br/>";
}
?>

```

**Listing 3.11.** Wynik działania skryptu z listingu 3.10

```

ajax
bearer
bugs
cache
chtml_ui
css
display
drm
flash_lite
html_ui
image_format
j2me
markup
mms
object_download
pdf
playback
product_info
rss
security
sms
sound_format
storage
streaming
transcoding
wap_push
wml_ui
wta
xhtml_ui

```

Aby wyświetlić listę wszystkich dostępnych właściwości, możemy zmodyfikować skrypt z listingu 3.10, tak by wykorzystywał metodę `getCapabilitiesNameForGroup`. Zmodyfikowany skrypt pokazany jest na listingu 3.12. Początkowa część wyniku widnieje na listingu 3.13.

**Listing 3.12.** Wyświetlanie listy wszystkich właściwości, które mogą być sprawdzone

```

<?php
error_reporting(E_ALL);
require_once('wurflSetup.php');
$wurflManager = getWurflManager();
$urzadzenie = $wurflManager->getDeviceForHttpRequest( $_SERVER );
$grupy_wlasciwosci = $wurflManager->getListOfGroups();
asort( $grupy_wlasciwosci );
foreach ( $grupy_wlasciwosci as $c ) {
    print "<strong>" . $c . "</strong><br/>";
    var_dump( $wurflManager->getCapabilitiesNameForGroup( $c ) );
}
?>

```



**Listing 3.13.** Początek wyniku skryptu z listingu 3.12

```

ajax
array
  0 => string 'ajax_preferred_geoloc_api' (length=25)
  1 => string 'ajax_xhr_type' (length=13)
  2 => string 'ajax_support_getelementbyid' (length=27)
  3 => string 'ajax_support_event_listener' (length=27)
  4 => string 'ajax_manipulate_dom' (length=19)
  5 => string 'ajax_support_javascript' (length=23)
  6 => string 'ajax_support_inner_html' (length=23)
  7 => string 'ajax_manipulate_css' (length=19)
  8 => string 'ajax_support_events' (length=19)
bearer
array
  0 => string 'sdio' (length=4)
  1 => string 'wifi' (length=4)
  2 => string 'has_cellular_radio' (length=18)
  3 => string 'max_data_rate' (length=13)
  4 => string 'vpn' (length=3)

```

...

Możemy zmodyfikować skrypt z listingu 3.12 tak, aby wyświetlały się tylko niektóre właściwości urządzenia, by wspierane właściwości były oznaczone kolorem zielonym i znacznikami (renderowanymi przez encje HTML) oraz by niewspierane właściwości były oznaczone kolorem czerwonym i przekreśleniem (listing 3.14). Wynik pokazany jest na rysunku 3.1.

**Listing 3.14.** Wyświetlanie właściwości urządzeń wraz z kolorowaniem

```

<?php
error_reporting ( E_ALL );
require_once ('wurflSetup.php');
$wurflManager = getWurflManager ();
$urządzenie = $wurflManager->getDeviceForHttpRequest ( $_SERVER );
$grupy_wlasciwosci = $wurflManager->getListOfGroups ();
asort ( $grupy_wlasciwosci );
foreach ( $grupy_wlasciwosci as $grupa ) {
    //Wyświetlamy właściwości tylko z niektórych grup.
    if (in_array ( $grupa, array ("ajax", "css", "image_format" ) ) ) {
        print "<strong>" . $grupa . "</strong><br/>";
        print "<ul>";
        foreach ( $wurflManager->getCapabilitiesNameForGroup ( $grupa ) as $nazwa ) {
            $c = $urządzenie->getCapability ( $nazwa );
            if ( $c == "false" ) {
                $c = "<li><span style='color:red; text-decoration:line-through;'>";
                $c .= $nazwa . "</span>";
            } else if ( $c == "true" ) {
                $c = "<li><span style='color:green;'> &#10003; ";
                $c .= $nazwa . "</span>";
            } else {
                $c = "<li>" . $nazwa . ": <em>" . $c . "</em>";
            }
            print $c;
            print "</li>";
        }
        print "</ul>";
    }
}
?>

```

```

ajax
  • ajax_preferred_geoloc_api: none
  • ajax_xhr_type: standard
  • ✓ ajax_support_getelementbyid
  • ✓ ajax_support_event_listener
  • ✓ ajax_manipulate_dom
  • ✓ ajax_support_javascript
  • ✓ ajax_support_inner_html
  • ✓ ajax_manipulate_css
  • ✓ ajax_support_events

css
  • css_gradient: none
  • css_border_image: none
  • css_rounded_corners: none
  • css_sprites
  • ✓ css_supports_width_as_percentage

image_format
  • greyscale
  • ✓ jpg
  • ✓ gif
  • transparent_png_index
  • epoc_bmp
  • ✓ bmp
  • wbmp
  • ✓ gif_animated
  • colors: 65536
  • svgt_1_1_plus
  • svgt_1_1
  • transparent_png_alpha
  • ✓ png
  • tiff

```

**Rysunek 3.1.** Wynik działania skryptu z listingu 3.14, wyświetlającego właściwości urządzenia w przejrzystej formie

W ostatnim skrypcie wykorzystującym WURFL API wyświetlimy niektóre właściwości urządzenia (listing 3.15).

**Listing 3.15.** Wyświetlenie niektórych właściwości iPhone'a 4 dotyczących dźwięku i wyświetlania

```

<?php
error_reporting(E_ALL);
require_once('wurflSetup.php');
$wurflManager = getWurflManager();
$_SERVER['HTTP_USER_AGENT'] =
    "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X; en-us) AppleWebKit/532.9 (KHTML, like
    ↳ Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7";
$urządzenie = $wurflManager->getDeviceForHttpRequest($_SERVER);
//wyświetlenie interesujących nas pól
//informacje dotyczące wyświetlania
print "<h2>" . $urządzenie->id . "</h2>";
print "<p><strong>Wyświetlanie: </strong><br/>";
print $urządzenie->getCapability( 'resolution_width' ) . " x "; //szerokość
print $urządzenie->getCapability( 'resolution_height' ) . " : "; //wysokość
print $urządzenie->getCapability( 'colors' ) . ' kolorów<br/>';
print "podwójna orientacja: " . $urządzenie->getCapability( 'dual_orientation' ) . "</p>";

```

```
//informacje dotyczące dźwięku
print "<p><strong>Wspierane formaty audio:</strong><br/>";
foreach ( $wurflManager->getCapabilitiesNameForGroup( "sound_format" ) as $nazwa ) {
    $c = $urządzenie->getCapability( $nazwa );
    if ( $c == "true" ) {
        print $nazwa . "<br/>";
    }
}
print "</p>";
?>
```

Wywołanie skryptu z listingu 3.15 wyświetla następujące informacje:

---

```
apple_iphone_ver4_sub405
Wyświetlanie:
320 x 480 : 65536 kolorów
podwojna-orientacja: true
Wspierane formaty audio:
aac
mp3
```

---

## Tera-WURFL

Implementacja WURFL, nazwana Tera-WURFL, dostępna jest pod adresem <http://www.tera-wurfl.com>. Poprzednio opisane PHP WURFL API budowane jest głównie z myślą o trafności wyników. Tera-WURFL skupione jest na wydajności. Aby osiągnąć wyższą wydajność, zamiast dużego pliku XML jest wykorzystywana baza danych. Aktualne Tera-WURFL wspiera MySQL, Microsoft SQL Server i MongoDB. Tera-WURFL może być do pięciu razy wydajniejszy niż zwykły WURFL (sprawdza się to w 99% przypadków), zapewnia też lepsze wykrywanie właściwości dla urządzeń stacjonarnych. Dodatkowo Tera-WURFL pozwala na pokazanie zdjęcia urządzenia, które zostało użyte. W dalszej części wyjaśnimy, jak wyświetlić zdjęcie urządzenia.

## Instalacja

Aby zainstalować Tera-WURFL, należy wykonać następujące kroki:

1. Utworzyć bazę danych oraz zmodyfikować dane dostępowe w pliku konfiguracyjnym *TeraWurflConfig.php*.
2. Otworzyć stronę administracyjną <http://localhost/Tera-WURFL/admin/>. Jeżeli pojawi się błąd dotyczący brakujących tabel, nie przejmuj się — tabele zostaną utworzone, kiedy będą wczytywane dane.
3. Możesz wczytać lokalny plik XML lub zdalny plik XML.

---

■ **UWAGA.** Jeżeli pojawi się informacja o błędzie, np. „fatal error maximum function nesting level of '100' reached aborting”, musisz tymczasowo wyłączyć opcję xdebug w pliku *php.ini* lub zwiększyć liczbę możliwych zagłębień xdebug poprzez ustawienie w pliku *php.ini* dyrektywy `xdebug.max_nesting_level=100` na wyższą wartość, np. 500.

---

## Rozpoznawanie urządzeń za pomocą Tera-WURFL

W pierwszym przykładzie, pokazanym na listingu 3.16, wykorzystamy nagłówek urządzenia klienckiego dla iPhone'a 4 i sprawdzimy, czy Tera-WURFL został zainstalowany poprawnie i czy go rozpozna.

**Listing 3.16.** Kod Tera-WURFL sprawdzający urządzenie klienckie

```
<?php
error_reporting(E_ALL);
```

```

require_once('Tera-WURFL/TeraWurfl.php');
$teraWURFL = new TeraWurfl();
$iphone = "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X; en-us) AppleWebKit/532.9
↳KHTML, like Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7";
if ( $teraWURFL->getDeviceCapabilitiesFromAgent( $iphone ) ) {
    print "ID: ".$teraWURFL->capabilities['id']."<br/>";
} else {
    print "urządzenie nie zostało rozpoznane";
}
?>

```

Wynik działania skryptu to:

ID: apple\_iphone\_ver4\_sub405

Gdybyśmy nie przekazali nagłówka urządzenia klienckiego jako parametru, tak jak w przypadku WURFL, wynikiem byłby klient, który został wykorzystany do otwarcia strony.

## Rozpoznawanie właściwości urządzenia i tworzenie ich listy za pomocą Tera-WURFL

Skrypt z listingu 3.17 tworzy listę właściwości dotyczących wyświetlania i dźwięku dla iPhone'a 4. Jest to wersja skryptu z listingu 3.15 wykorzystująca Tera-WURFL.

**Listing 3.17.** Wyświetlenie właściwości dotyczących dźwięku i wyświetlania dla iPhone'a 4 przy użyciu Tera-WURFL

```

<?php
error_reporting(E_ALL);
require_once('Tera-WURFL/TeraWurfl.php');
$teraWURFL = new TeraWurfl();
$iphone = "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X; en-us) AppleWebKit/532.9
↳(KHTML, like Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7";
if ( $teraWURFL->getDeviceCapabilitiesFromAgent( $iphone ) ) {
    $marka_urzadzenia = $teraWURFL->getDeviceCapability( "brand_name" );
    $model_urzadzenia = $teraWURFL->getDeviceCapability( "model_name" );
    $model_dodatkowe_info = $teraWURFL->getDeviceCapability( "model_extra_info" );

    //Wyświetlenie informacji, które nas interesują.
    print "<h2> " . $marka_urzadzenia . " " . $model_urzadzenia . " " . $model_dodatkowe_info .
    ↳"</h2>";

    //informacje dotyczące wyświetlania
    print "<p><strong>Informacje dotyczące wyświetlania:</strong><br/>";
    print $teraWURFL->getDeviceCapability( 'resolution_width' ) . " x "; //szerokość
    print $teraWURFL->getDeviceCapability( 'resolution_height' ) . " : "; //wysokość
    print $teraWURFL->getDeviceCapability( 'colors' ) . ' kolorów<br/>';
    print "podwójna orientacja: " . $teraWURFL->getDeviceCapability( 'dual_orientation' );
    print "</p>";

    //informacje dotyczące dźwięku
    print "<p><strong>Wspierane formaty audio:</strong><br/>";
    foreach ( $teraWURFL->capabilities['sound_format'] as $nazwa => $wartosc ) {
        if ( $wartosc == "true" ) {
            print $nazwa . "<br/>";
        }
    }
}
print "</p>";

```

```

} else
{
    print "urządzenie nie zostało rozpoznane";
}
?>

```

Wynik działania skryptu to:

#### Apple iPhone 4.0

*Informacje dotyczące wyświetlania:*

320 x 480 : 65536 kolorów

podwójna orientacja: 1

*Wspierane formaty audio:*

aac

mp3

## Wyświetlenie zdjęcia urządzenia przy wykorzystaniu Tera-WURFL

W ostatnim przykładzie dotyczącym Tera-WURFL pokazemy, jak wyświetlić zdjęcie urządzenia. Najpierw należy pobrać zestaw zdjęć urządzeń znajdujący się pod adresem <http://sourceforge.net/projects/wurfl/files/WURFL%20Device%20Images/>. Następnie trzeba rozpakować archiwum i umieścić jego zawartość w miejscu dostępnym dla usługi web. My utworzymy folder *zdjecia\_urzadzen* w katalogu */Tera-WURFL/*. Rozbudujemy poprzedni przykład z listingu 3.17. Najpierw dodamy do skryptu kolejny plik z biblioteki Tera-WURFL, a później kod pobierający odpowiedni obrazek i wyświetlający go (listing 3.18). Wynik jego działania jest pokazany na rysunku 3.2.

### Listing 3.18. Wyświetlenie zdjęcia urządzenia

```

<?php

error_reporting ( E_ALL );
require_once ('Tera-WURFL/TeraWurfl.php');
require_once ('Tera-WURFL/TeraWurflUtils/TeraWurflDeviceImage.php');

$teraWURFL = new TeraWurfl ();
$iphone = "Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_0 like Mac OS X; en-us)? AppleWebKit/532.9
↳(KHTML, like Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7";

if ($teraWURFL->getDeviceCapabilitiesFromAgent ( $iphone )) {
    $marka_urzadzenia = $teraWURFL->getDeviceCapability( "brand_name" );
    $model_urzadzenia = $teraWURFL->getDeviceCapability( "model_name" );
    $model_dodatkowe_info = $teraWURFL->getDeviceCapability( "model_extra_info" );

    //Wyświetlenie informacji, które nas interesują.
    print "<h2>" . $marka_urzadzenia . " " . $model_urzadzenia . " " . $model_dodatkowe_info .
    ↳"</h2>";

    //zdjęcie
    $zdjecie = new TeraWurflDeviceImage ( $teraWURFL );
    //adres na serwerze
    $zdjecie->setBaseURL ( '/Tera-WURFL/zdjecia_urzadzen/' );
    //adres w systemie plików
    $zdjecie->setImagesDirectory ( $_SERVER ['DOCUMENT_ROOT'] . '/Tera-WURFL/zdjecia_urzadzen/' );

    $zdjecie_src = $zdjecie->getImage ();
    if ($zdjecie_src) {
        print '';
    } else {

```



**Rysunek 3.2.** Wynik działania skryptu z listingu 3.18 — wyświetlenie zdjęcia urządzenia

```

    echo "Zdjęcie niedostępne";
}

//informacje dotyczące wyświetlania
print "<p><strong>Informacje dotyczące wyświetlania: </strong><br/>";
print $teraWURFL->getDeviceCapability( 'resolution_width' ) . " x "; //szerokość
print $teraWURFL->getDeviceCapability( 'resolution_height' ) . " : "; //wysokość
print $teraWURFL->getDeviceCapability( 'colors' ) . ' kolorów<br/>';
print "podwójna orientacja: " . $teraWURFL->getDeviceCapability( 'dual_orientation' );
print "</p>";

//informacje dotyczące dźwięku
print "<p><strong>Wspierane formaty audio:</strong><br/>";
foreach ( $teraWURFL->capabilities['sound_format'] as $nazwa => $wartosc ) {
    if ( $wartosc == "true" ) {
        print $nazwa . "<br/>";
    }
}
print "</p>";
} else
{
    print "urządzenie nie zostało rozpoznane";
}
?>

```

## Narzędzia renderujące

Aby dynamicznie modyfikować zawartość strony dla różnych urządzeń mobilnych, możemy wykorzystać abstrakcyjne znaczniki przy zastosowaniu Wireless Abstraction Library (WALL), automatyczną zmianę rozmiarów obrazków i właściwości CSS dotyczące mediów.

### WALL

Luca Passani poza WURFL opracował także WALL — bibliotekę tworzącą abstrakcyjne znaczniki dla urządzeń mobilnych. Co to dokładnie znaczy? Najpierw trzeba zauważyć, że w odróżnieniu od stron dla normalnych przeglądarek na urządzenia stacjonarne, pisanych w HTML lub XHTML, dla urządzeń mobilnych jest więcej wariantów znaczników zawierających rozbieżności.

Najpopularniejszymi zestawami znaczników dla urządzeń mobilnych są:

- XHTML MP (Mobile Profile),
- CHTML (Compact HTML),
- HTML.

Część wspólna akceptowanych tagów we wszystkich tych językach jest ograniczona. Np. znacznik nowej linii utworzony jako `<br>` zamiast `<br />` lub na odwrót może zostać zignorowany bądź wywołać błąd w zależności od zastosowanego języka.

Korzystając z WALL, możemy zapisać nową linię jako `<wall:br />`. Dzięki WURFL znajdziemy język odpowiedni dla danego urządzenia poprzez sprawdzenie właściwości `preferred_markup`. Mając tę informację, WALL wyrenderuje odpowiednie znaczniki `<br>` lub `<br />`. WALL napisano pierwotnie dla Java Server Pages (JSP). Biblioteka WALL4PHP, dostępna pod adresem <http://laacz.lv/dev/Wall/>, została utworzona na potrzeby PHP. Istnieje także zaktualizowana wersja biblioteki, utrzymywana przez twórców Tera-WURFL, dostępna pod adresem <https://github.com/kamermans/WALL4PHP-by-Tera-WURFL>. Na potrzeby przykładów zastosujemy pierwotną implementację. Dokładne instrukcje integracji bibliotek WALL i WURFL można znaleźć pod adresem <http://www.tera-wurfl.com/wiki/index.php/WALL4PHP>.

Plik PHP wykorzystujący WALL mógłby wyglądać jak na listingu 3.19.

**Listing 3.19.** Dokument wykorzystujący WALL

```
<?php require_once('WALL4PHP/wall_prepend.php'); ?>
<wall:document><wall:xmlpidtd />
<wall:head>
  <wall:title>WALL jest super</wall:title>
</wall:head>
<wall:body>
  <wall:h1>Nagłówek</wall:h1>
  <wall:block>To będzie paragraf w HTML</wall:block>
  <wall:menu autonumber="true">
    <wall:a href="http://ur1A">A</wall:a>
    <wall:a href="http://ur1B">B</wall:a>
  </wall:menu>
</wall:body>
</wall:document>
```

Efekt renderowania będzie zależny od urządzenia. Jeżeli urządzenie wspiera HTML, kod zostanie wyrenderowany zgodnie z listingiem 3.20.

**Listing 3.20.** Kod wyrenderowany dla urządzenia wspierającego HTML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN" "http://www.wapforum.org/DTD/
↳xhtml-mobile10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>WALL jest super</title>
</head>
<body>
  <h1>Nagłówek</h1>
  <p>To będzie paragraf w HTML</p>
  <ol>
    <li><a accesskey="1" href="ur1A">A</a></li>
    <li><a accesskey="2" href="ur1B">B</a></li>
  </ol>
</body>
</html>
```

## Reagujący CSS

Aby rozmieszczenie elementów na stronie było odpowiednie dla urządzenia, możemy zastosować pływające kontenery oraz zmieniać rozmiar zdjęć, jak pokazano powyżej. Możemy także wykorzystać arkusze przeznaczone specjalnie dla urządzeń mobilnych. Niedawnym usprawnieniem w CSS są zapytania o media. Właściwości, jakie można sprawdzić, to `width`, `height`, `device-width`, `device-height`, `orientation`, `aspect-ratio`, `device-aspect-ratio`, `color`, `color-index`, `monochrome`, `resolution`, `scan` i `grid` (listing 3.21).

**Listing 3.21.** Przykładowe zapytania o media za pomocą CSS3

```
@media screen and (min-device-width:400px) and (max-device-width:600px){
    /* ograniczenie szerokości urządzenia */
}
@media screen and (orientation:landscape){
    /* dobre dla urządzeń mogących działać w dwóch płaszczyznach, takich jak iPad i Kindle */
}
```

Głębsza analiza CSS wyszłaby poza zakres tej książki, jednakże pod adresem <http://www.netmagazine.com/tutorials/adaptive-layouts-media-queries> dostępny jest doskonały artykuł. Opisane tam techniki mogą usprawnić wyświetlanie stron na urządzeniach mobilnych. Więcej informacji dotyczących zapytań o media w CSS3 znajduje się pod adresem <http://www.w3.org/TR/css3-mediaqueries/>.

## Emulatory i SDK

W ramach pomocy dla twórców mających ograniczony budżet, niepozwalający na zakup telefonów na potrzeby testów, oraz w celu ułatwienia pracy powstało wiele emulatorów i zestawów narzędzi dla programistów (Software Developer Kit — SDK) piszących aplikacje przeznaczone dla urządzeń mobilnych. Niektóre narzędzia emulują jedno urządzenie, inne kilka jednocześnie. Oto wybrane adresy, pod którymi są one dostępne:

- Android: <http://developer.android.com/guide/developing/tools/emulator.html>
- Apple: <http://developer.apple.com/devcenter/ios/index.action>
- BlackBerry: <http://www.blackberry.com/developers/downloads/simulators/>
- Kindle: <http://www.amazon.com/kdk/>
- Opera Mini: <http://www.opera.com/mobile/demo/>
- Windows: <http://create.msdn.com/en-us/resources/downloads>

## Tworzenie dla systemu Android

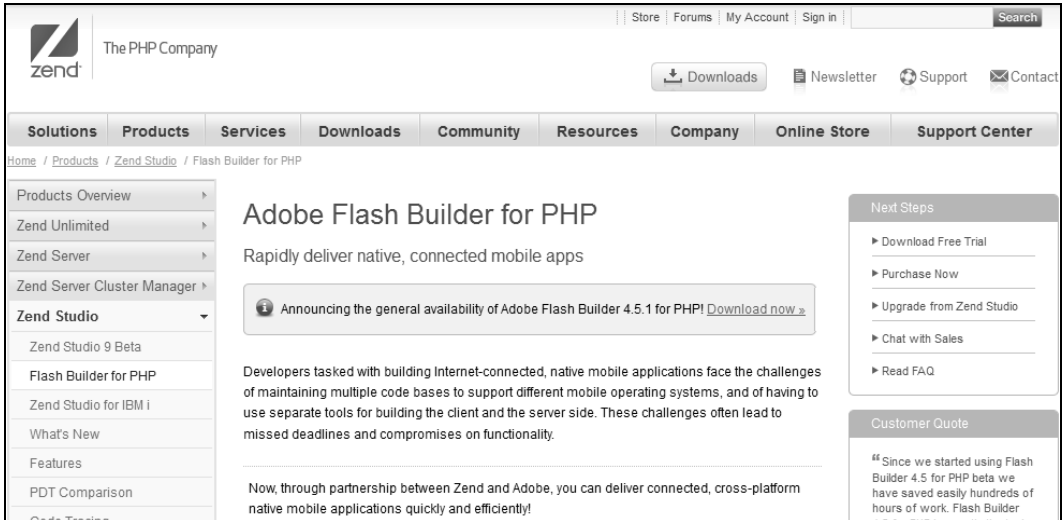
System Android wydany przez Google może uruchamiać aplikacje w Java i natywnym C. Projekt warstwy skryptów dla Androida (SL4A) dostępny pod adresem <http://code.google.com/p/android-scripting/> pozwala na tworzenie aplikacji w językach skryptowych. Jednakże PHP aktualnie nie należy do wspieranych języków.

Aby stworzyć aplikacje dla Androida w PHP, możemy wykorzystać projekt open source PHP for Android, dostępny pod adresem <http://www.phpforandroid.net/>. Narzędzie to oferuje nieoficjalne wsparcie dla PHP wewnątrz SL4A poprzez plik APK (Android Package).

## Adobe Flash Builder dla PHP

Niedawno Zend ogłosiło połączenie sił z Adobe w celu wprowadzenia wsparcia dla PHP w aplikacji Flash Builder 4.5 (rysunek 3.3). Więcej informacji o Flash Builder dla PHP można znaleźć pod adresem <http://www.zend.com/en/products/studio/flash-builder-for-php/>. Flash Builder dla PHP zawiera Zend Studio w zintegrowanym środowisku programistycznym (IDE). Jako frontend może być wykorzystany Flex, natomiast jako backend — PHP.





**Rysunek 3.3.** Ogłoszenie dotyczące programu Flash Builder na stronie Zend

IDE ma ułatwiać tworzenie kodu i zapewniać lepszą jego przenośność pomiędzy platformami mobilnymi. Może nawet skompilować kod Flex, tak aby był wykonywany natywnie na urządzeniach opartych na systemie iOS, takich jak iPhone i iPad.

## Kody QR

Kody QR (Quick Response) są czymś w rodzaju dwuwymiarowego kodu kreskowego. Zostały wprowadzone w Japonii ponad dwadzieścia lat temu w celu katalogowania części samochodowych. Nowoczesne urządzenia mobilne z wbudowanymi aparatami fotograficznymi przyczyniły się do rozpowszechnienia tego rozwiązania.

Kod QR zazwyczaj reprezentuje adres URL, ale może zawierać więcej tekstu. Pokażemy, w jaki sposób łatwo wygenerować kody QR za pomocą trzech różnych bibliotek. Dwie z nich, TCPDF i Google Chart API, są omówione dokładniej w rozdziale 10.

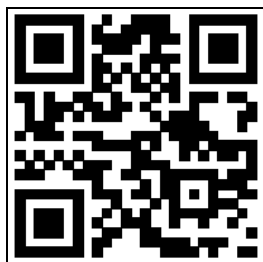
Pierwsza biblioteka, za której pośrednictwem wygenerujemy kod QR, jest dostępna pod adresem <http://www.tcpdf.org/>. Za pomocą TCPDF możemy wygenerować kod QR jako PDF, nie możemy jednak generować kodów jako odrębnych plików graficznych. Zobacz listing 3.22 i rysunek 3.4 przedstawiający wygenerowany kod QR.

**Listing 3.22.** Generowanie kodu QR w pliku PDF przy wykorzystaniu biblioteki TCPDF

```
<?php

error_reporting(E_ALL);
require_once('/tcpdf/config/lang/eng.php');
require_once('/tcpdf/tcpdf.php');

$pdf = new TCPDF(); //utworzenie obiektu TCPDF
$pdf->AddPage(); //dodanie nowej strony
$pdf->write2DBarcode( 'Witaj, świecie kodów QR', 'QRCODE' );
//napisz 'Witaj, świecie kodów QR' jako kod QR
$pdf->Output( 'qr_code.pdf', 'I' ); //wygeneruj i wyslij plik pdf
?>
```



**Rysunek 3.4.** Kod QR dla ciągu znaków „Witaj, świecie kodów QR”. Każda biblioteka powinna wygenerować taki sam obrazek

Aby zapisać kod QR do pliku, możemy wykorzystać bibliotekę `phpqrcode`, dostępną pod adresem <http://phpqrcode.sourceforge.net/index.php> (listing 3.23).

**Listing 3.23.** Generowanie kodów QR bezpośrednio do pliku lub przeglądarki za pośrednictwem `phpqrcode`

```
<?php
require_once('phpqrcode/qrlib.php');

QRcode::png( 'Witaj, świecie kodów QR', 'qrcode.png' ); //do pliku
QRcode::png( 'Witaj, świecie kodów QR' ); //bezpośrednio do przeglądarki
?>
```

Możemy także wykorzystać Google Chart API znajdujące się pod adresem <http://code.google.com/p/gchartphp/> (listing 3.24).

**Listing 3.24.** Generowanie kodów QR za pośrednictwem biblioteki `qrcodephp`

```
<?php

error_reporting(E_ALL);
require_once ('GChartPhp/gChart.php');

$qr = new gQRCode();
$qr->setQRCode( 'Witaj, świecie kodów QR' );
echo "<img src=\"".$qr->getUrl()."\" />";
?>
```

## Podsumowanie

W tym rozdziale omówiliśmy rozpoznawanie urządzeń mobilnych i ich właściwości. Obecnie nie ma idealnego systemu wykrywania urządzeń, jednak to, co mamy, jest względnie niezawodne. Programista powinien być czujny i mieć aktualne pliki, niezależnie od tego, czy wykorzystuje `browsercap`, `WURFL`, czy inny system.

Pokazaliśmy także narzędzia do tworzenia abstrakcyjnego kodu, automatycznej zmiany rozmiarów obrazków oraz płynnej zmiany rozmiarów zawartości strony. Kiedy tylko jest to możliwe, używaj narzędzi, które wykonają pracę za Ciebie. Mogą one sprawdzić, jakie urządzenie zostało użyte i co ono potrafi; mogą też pomóc w ustaleniu, w jaki sposób przetransformować istniejące style, obrazki i kod. Automatyzacja i przydatne biblioteki powinny być stosowane we wszystkich dziedzinach tworzenia oprogramowania.

Technologie mobilne ciągle się rozwijają, więc także i metody tworzenia oprogramowania szybko ulegają zmianie. Aby zostać dobrym twórcą oprogramowania dla urządzeń mobilnych, musisz przyswajać sobie dobre praktyki, poznawać najnowsze technologie oraz wchodzące na rynek SDK i API.

# Skorowidz

- \$\_COOKIEs, 224
- \$\_FILES, 113
- \$\_GET, 222
- \$\_POST, 222
- \$\_SERVER, 224
- \$\_SERVER['HTTP\_USER\_AGENT'], 48
- \$\_SESSION, 224
- \$argc, 39
- \$argv, 39
- \$this, 22
- \_\_autoload, funkcja, 25, 96
- \_\_call, funkcja, 28
- \_\_clone, funkcja, 30
- \_\_construct, 22
- \_\_get, funkcja, 27, 28
- \_\_isset, funkcja, 28
- \_\_set, funkcja, 27, 28
- \_\_toString, funkcja, 29
- \_\_unset, funkcja, 28

## A

- ACID, 121, 122
- ACME, 26
- Adobe Flash Builder, 62, 63
- ADODB, 35, 161, 162, 163, 164, 165
- Ajax, 224, 225, 307, 312, 313
  - model, 313, 314
  - przesyłanie danych do skryptu PHP, 327
- akcesory, 22
- algorytmy haszujące, 235
- analiza statyczna, 281
- Android, 62
- anonimowe, funkcje, 96, 97
- Apache, 233
- aplikacje mobilne, 47

- array\_map, funkcja, 96
- array\_reduce, funkcja, 96, 97
- autoładowanie, 21, 96
- automatyzacja, 282

## B

- bazy
  - CouchDb, 134, 140
  - MongoDB, 122, 134
  - MySQL, 151, 154
  - NoSQL, 121, 122
  - Oracle, 175, 176, 177
  - relacyjne, 141, 142, 143
  - SQLite, 141, 142, 143, 144, 149
  - zgodność kodowania, 192
- bezpieczeństwo, 221
  - ataki, 225
  - białe i czarne listy, 222, 223
  - CSRF, 228
  - dane formularzy, 223, 224
  - sesje, 229
  - SQL injection, 229
  - XSS, 225, 226, 227
- biała lista, 222, 223
- biblioteki, 195
  - gChartPHP, 215, 216
  - Google Chart API, 215, 216
  - php-google-map-api, 209, 210
  - PHPMailer, 211, 214
  - phpQuery, 205, 208, 209
  - PHPUnit, 265, 266, 267
  - SimplePie, 196, 199
  - Simpletest, 265, 266, 267
  - TCPDF, 199, 200

blob, 155  
 bloki obsługi błędów, 39  
 bloki obsługi wyjątków, 39  
 browscap.ini, 48, 49  
 Bugzilla, 239, 242, 243

## C

call\_user\_func\_array, funkcja, 155  
 catch, blok, 39  
 certyfikacja, 340, 341  
 ciągła integracja, 249, 279  
 clone, 30  
 COMMIT, polecenie, 154  
 compile, metoda, 117  
 CoonFoo, konferencja, 340  
 CouchDB, 134, 137, 138, 140  
     Futon, interfejs, 135, 136  
     widoki, 139  
 create\_function, funkcja, 96  
 Cross Site Scripting, *Patrz* XSS  
 Cross-Site Request Forgery, *Patrz* CSRF  
 CruiseControl, 280  
 CSRF, 228  
 CSS, 62  
 CSV, 152  
 CURL, 67  
 CURRENT\_AS\_FILEINFO, 103  
 CURRENT\_AS\_PATHNAME, 103  
 czarna lista, 222, 223

## D

Danchilla, Brian, 11  
 DBMS\_OUTPUT, 184  
 denormalizacja, 130  
 destruktor, 24  
 DISABLE STORAGE IN ROW, 187  
 DOM, 287, 303, 304  
 DRCP, *Patrz* Oracle, rezydentne pule połączeń bazy danych  
 DTD, 288  
 dziedziczenie, 23, 24  
     klasy abstrakcyjne, 25, 31  
     klasy nadrzędne, 24

## E

empty, funkcja, 28  
 enkapsulacja, 21, 24  
 Exception, klasa, 37  
 exec, metoda, 117  
 explode, funkcja, 350

## F

Facebook, 65  
     API, 83  
     dodanie linku wylogowania, 88  
     Graph API, 89  
     tworzenie aplikacji, 83, 84  
     weryfikacja konta, 83  
     wyszukiwanie albumów i fotografii, 90  
     żądanie dodatkowych uprawnień, 89  
 FileSystemIterator, klasa, 102, 103  
 Filter, biblioteka, 110  
 filter\_has\_var, funkcja, 232, 233  
 filter\_id, funkcja, 233  
 filter\_input, funkcja, 233  
 filter\_input\_array, funkcja, 233  
 filter\_list, funkcja, 233  
 filter\_var, funkcja, 110, 230  
 filter\_var\_array, funkcja, 232  
 final, 24  
 formularze, 107  
     bezpieczeństwo, 223, 224  
     walidacja danych, 107  
     walidacja w JavaScriptcie, 107, 108  
     walidacja w PHP, 107, 109, 110  
 funkcje  
     anonimowe, 96, 97  
     magiczne, 27

## G

gChartPHP, biblioteka, 215, 216  
 GD, biblioteka, 114  
 get, akcesor, 22  
 GET, metoda, 73, 108  
 get\_browser, funkcja, 48  
 getimagesize, funkcja, 114  
 globalne tabele tymczasowe, 175  
 Gogala, Mladen, 11  
 Google Chart API, biblioteka, 215, 216  
 goto, 93, 100  
 Gutmans, Andi, 18

## H

hash\_algos, funkcja, 235  
 hasła, algorytmy, 235  
 heredoc, 98, 99

## I

iconv, moduł, 119  
 include, dyrektywa, 25  
 indeksy tekstowe, 165

insert, polecenie, 146  
 interfejsy, 31, 32  
 International PHP Conference, 340  
 is\_uploaded\_file, funkcja, 113  
 ISO-8859-1, 118  
 Iterator, interfejs, 32  
 iterator, 31, 32, 33

**J**

JavaScript  
   walidacja formularzy, 107, 108  
   wyrażenia regularne, 117  
 Jenkins, 280  
   uruchomienie serwera, 282, 283, 284  
 jQuery, 322, 323, 325, 330  
 JSON, 121, 307, 308, 309  
 json\_decode, funkcja, 309  
 json\_encode, funkcja, 309  
 json\_last\_error, funkcja, 309

**K**

klasy, 21, 25  
   abstrakcyjne, 25, 31  
   destruktor, 24  
   dziedziczenie, 23, 24, 25  
   elementy, 21  
   elementy statyczne, 35  
   konstruktor, 22, 24  
   metody, 21  
   nadrzędne, 24  
   terminologia, 21  
   właściwości, 21  
 klonowanie, 29, 31  
 klucz główny, 141  
 kodowanie, problemy, 119  
 kody QR, 63, 64  
 konferencje  
   ConFoo, 340  
   International PHP Conference, 340  
   Open Source India, 340  
   OSCON, 340  
   ZendCon, 340  
 konstruktor, 22, 24  
 kontekst klasy, 35  
 kopiowanie, 29  
   głębokie, 30  
   płytkie, 30

**L**

Lerdorf, Rasmus, 17  
 LOAD DATA, polecenie, 152

LOB, 186, 187, 188, 189, 190  
   deskryptor, 188  
   lokatory, 187

**M**

MacIntyre, Peter, 11  
 magiczne funkcje, 27  
 mail, funkcja, 211  
 mapreduce, framework, 132  
 Mapy Google, 209  
 match, metoda, 117  
 mb\_convert\_encoding, funkcja, 119  
 mb\_ereg\_replace, funkcja, 119  
 mb\_strlen, funkcja, 119  
 mb\_substr, funkcja, 119  
 MD5, algorytm, 235  
 media społecznościowe, 65  
 metaznaki, 343, 344  
 metody, 21  
 mobilne, urządzenia, 47  
   Android, 62  
   emulatory, 62  
   możliwości, 51  
   rozpoznawanie, 48, 50  
   rozpoznawanie za pomocą Tera-WURFL, 57, 58  
   rozpoznawanie za pomocą WURFL, 52, 53, 54, 55, 56  
   różnorodność, 47  
   SDK, 62  
 mongo, polecenie, 123  
 MongoDB, 122, 123, 134  
   agregacje, 132  
   argument safe, 126  
   instalacja, 122  
   kolekcje, 124  
   modyfikowanie dokumentów, 130  
   operator \$where, 128  
   plik logu, 124  
   typy danych, 123  
   zapytania, 126, 127, 128  
 MongoRegex, 128  
 mt\_rand, funkcja, 235  
 Multibyte String Function, biblioteka, 119  
 MVCC, 134  
 Myer, Thomas, 13  
 Mylyn, 240, 242, 243  
 MySQL, 154  
 MySQLi, 151, 152, 155, 156, 158

**N**

narzędzia renderujące, 60  
 NLS\_LANG, 193  
 nonce, 224

NoSQL, 121  
 CouchDb, 134, 140  
 MongoDB, 122, 123  
 nowdoc, 97, 98, 99

## O

OAuth, 65, 66  
 etapy uwierzytelniania, 65  
 obiektowe, programowanie, 25, 37  
 obiektowość, 21  
 obiekty  
 klonowanie, 29  
 kopiowanie, 29, 30  
 porównywanie, 29, 31  
 obrazy  
 konwersja, 114  
 miniatury, 114  
 wczytywanie, 113  
 oci\_bind\_by\_name, 184  
 oci\_connect, 179  
 oci\_error, 179  
 oci\_execute, 179, 186  
 oci\_fetch\_array, 180  
 oci\_fetch\_row, 179  
 oci\_new\_descriptor, funkcja, 188  
 OCI\_NO\_AUTO\_COMMIT, 179, 186  
 oci\_num\_fields, 179  
 oci\_parse, 179  
 OCI8, rozszerzenie, 177, 179  
 OCI-Lob, klasa, 188, 189  
 onreadystatechange, funkcja, 316  
 OOP, 18, 37  
 Open Source India, konferencja, 340  
 open, funkcja, 316  
 operatory  
 ==, 31  
 ===, 31  
 Oracle, 175, 176, 177  
 dzielone obszary globalne, 190  
 globalny obszar procesu, 190  
 interfejs tablicowy, 180  
 Oracle\*Text, 186  
 pule połączeń, 190  
 rezydentne pule połączeń baz danych, 190  
 synonimy, 176  
 wykonywanie zapytań, 177  
 OSCON, konferencja, 340

## P

parsowanie stron, 204, 205  
 PCRE, biblioteka, 111, 118, 343  
 PDF, generowanie, 199

PDO, 158, 159, 161  
 PEAR, 40  
 PEAR\_Exception, 40  
 Perl Compatible Regular Expressions, *Patrz* PCRE, biblioteka  
 PGA, *Patrz* Oracle, globalny obszar procesu  
 phar, rozszerzenie, 103, 104, 105, 106  
 PHP, 17, 18  
 biblioteki, 195  
 certyfikacja, 340, 341  
 geneza, 17  
 konferencje, 339, 340  
 przyszłość, 20  
 standardowa biblioteka, 25, 100  
 zasoby, 337  
 Zend Corporation, 18  
 php.ini, 221, 233  
 zabezpieczanie, 234  
 php\_info, funkcja, 114  
 php-google-map-api, biblioteka, 209, 210  
 PHPMailer, biblioteka, 211, 214  
 PHP-on-Couch, biblioteka, 134  
 phpQuery, biblioteka, 205, 208, 209  
 phpUnderControl, 280  
 PHPUnit, biblioteka, 265, 266, 267  
 PL/SQL, 182, 183  
 procedury i kursory, 183  
 pliki, wczytywanie, 113  
 polityka tego samego pochodzenia, 225  
 porównywanie obiektów, 29, 31  
 Posix, 343  
 POST, metoda, 73, 108  
 preferred\_markup, 61  
 preg\_grep, funkcja, 350, 351  
 preg\_match\_all, funkcja, 111  
 preg\_replace, funkcja, 348, 349  
 preg\_split, funkcja, 350  
 private, 22  
 programowanie  
 obiektowe, 25, 37  
 zwinne, 237, 238, 239  
 protected, 22, 23  
 przeciążanie, 23, 24  
 przestrzenie nazw, 93, 94, 95, 96  
 public, 22

## Q

QR, kody, 63, 64  
 Quick Response, 63

## R

RAC, 176, 177  
 rand, funkcja, 235  
 Really Simple Syndication, *Patrz* RSS  
 refaktoryzacja, 249, 250, 257, 278  
   przykłady, 249  
 referencje, 37, 41, 42, 43, 44, 45  
 Regex, 50  
 Regex Tester, 115  
 register\_globals, 222  
 RELAX NG, 288  
 replace, metoda, 117  
 require, dyrektywa, 25  
 ROLLBACK, polecenie, 154  
 rozszerzenie  
   MySQLi, 151, 152, 155, 156, 158  
   OCI8, 177, 179  
   phar, 103, 104, 105, 106  
   SimpleXML, 287, 289, 292, 298, 303  
   XMLReader, 305, 306  
   XMLWriter, 305, 306  
 RSS, 296, 297

## S

scrum, 238  
 search, metoda, 117  
 serwer  
   ciągłej integracji, 280  
   ustawienia, 233  
 sesje, 229  
 set, akcesor, 22  
 set\_exception\_handler, funkcja, 39, 40  
 SGA, *Patrz* Oracle, dzielone obszary globalne  
 SHA1, algorytm, 235  
 SimplePie, biblioteka, 196, 199  
 Simpletest, biblioteka, 265, 266, 267  
 SimpleXML, rozszerzenie, 287, 289, 292, 303  
   generowanie dokumentów XML, 298  
 singleton, 78  
 Sphinx, 165, 166, 168, 171, 172  
 SPL, 25, 35, 100, 103  
 SplFileInfo, klasa, 102  
 split, metoda, 117  
 SplMaxHeap, klasa, 100  
 społecznościowe, media, 65  
 SQL injection, 229  
 SQLite, 78, 141, 142, 143, 144, 149  
 SSL, 224  
 strony, parsowanie, 204, 205  
 Subversion, 280  
 Suraski, Zeev, 18  
 SVN, 210, 280  
 system kontroli wersji, 280

## Ś

środowisko serwerowe, 233

## T

tabele tęczowe, 235  
 TCPDF, biblioteka, 63, 199, 200  
 Tera-WURFL, 57  
   instalacja, 57  
   rozpoznawanie urządzeń, 57  
   rozpoznawanie właściwości urządzeń, 58  
   wyświetlenie zdjęcia urządzenia, 59  
 Test Driven Development, 278  
 test, metoda, 117  
 testy funkcjonalne, 269  
 testy jednostkowe, 249, 265, 266, 269, 278, 279  
 throw, 39  
 try, blok, 39  
 Twitter, 65, 66  
   API publicznego wyszukiwania, 66  
   dodatkowe metody API, 80  
   REST API, 67, 73  
   tworzenie nowej aplikacji, 67, 68  
   uwierzytelnianie przez odwołanie zwrotne, 74  
   uwierzytelnianie przy użyciu numeru PIN, 70, 71, 72  
   uwierzytelnianie przy użyciu tokenu dostępu, 69  
   zakładanie konta, 67  
 twitteroauth, 67

## U

uniqid, funkcja, 235  
 unset, funkcja, 24  
 urządzenia mobilne, 47  
   emulatory, 62  
   możliwości, 51  
   rozpoznawanie, 48, 50  
   rozpoznawanie za pomocą Tera-WURFL, 57, 58  
   rozpoznawanie za pomocą WURFL, 52, 53, 54, 55, 56  
   różnorodność, 47  
   SDK, 62  
 UTF-8, 118, 119  
 utf8\_encode, funkcja, 119

## W

WALL, 60, 61  
 WALL4PHP, 61  
 WAMP, 49  
 więzy integralności, 141  
 Wireless Abstraction Library, *Patrz* WALL  
 Writer, rozszerzenie, 305

WURFL, 51  
  instalacja, 51  
  rozpoznawanie urzędzeń, 52, 53  
  rozpoznawanie właściwości urzędzenia, 53, 54, 55, 56  
WURFLManager, 51  
wyjątki, 37, 38, 39  
  wywołanie, 37  
wykresy świecowe, generowanie, 217, 218  
wyrażenia filtrujące, 230  
wyrażenia regularne, 50, 115, 116, 117, 343, 346, 347, 348, 350  
  dostępne znaki, 116  
  funkcje PCRE, 118  
  JavaScript, 117  
  metaznaki, 343, 344  
  modyfikatory globalne, 347  
  przykłady, 344  
  składnia, 343  
wyszukiwanie pełnotekstowe, 165, 166

## X

XHTML, 287, 292  
XML, 51, 287  
  parsowanie z pliku, 290  
  parsowanie z tekstu, 289  
  przestrzenie nazw, 294  
  schematy, 288

XML Schema, 288  
XMLHttpRequest, 316, 317  
XMLReader, rozszerzenie, 305, 306  
XMLWriter, rozszerzenie, 306  
XPath, 292, 293, 296  
XSS, 225, 226, 227

## Z

ZCE, 18, 341  
zdarzenia synchroniczne, 37  
Zend, 18, 338, 340  
ZendCon, konferencja, 340  
złączenie, 147  
zwinne, programowanie, 237, 238, 239  
zasady, 237



# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

## PHP. Zaawansowane programowanie

PHP jest obecnie najpopularniejszym językiem programowania aplikacji internetowych, a jego znajomość staje się koniecznością dla każdego programisty. *PHP. Zaawansowane programowanie* zapozna Cię z nowymi możliwościami wersji 5.3.x, takimi jak przestrzenie nazw, funkcje anonimowe, Nowdoc, SPL oraz archiwizacja Phar. Doświadczeni programiści PHP znajdą tu przejrzyste wyjaśnienia i przydatne fragmenty kodów dotyczących programowania obiektowego, mobilnych urządzeń klienckich, skalowalnych źródeł danych, XML-a, AJAX-a, JSON-a oraz bezpieczeństwa.

Autorzy tej książki skupiają się na zaawansowanych zagadnieniach związanych z językiem PHP. W trakcie lektury dowiesz się, jak stworzyć aplikację dla platform mobilnych, zintegrować swój serwis z takimi portalami, jak Facebook i Twitter. Znajdziesz tu rozdziały poświęcone ważnemu tematowi programowania baz danych — od mało znanych baz NoSQL, poprzez CouchDB, MongoDB i SQLite, aż do Oracle, rozszerzenia MySQLi, PDO, ADOdb oraz system wyszukiwania pełnotekstowego Sphinx. Nauczysz się korzystać z bibliotek open source oraz parsować wiadomości RSS, generować dokumenty PDF, pobierać dane ze stron WWW, korzystać z bibliotek Map Google i Google Chart, a także tworzyć wiadomości e-mail i SMS. Ta książka zaczyna się w miejscu, w którym inne kończą omawianie PHP. Jeżeli tworzysz nowatorskie aplikacje albo chcesz zintegrować się z serwisami społecznościowymi, musisz ją mieć!

- Integracja z serwisami Facebook i Twitter
- Dokumentowanie kodu
- Wykorzystanie wyspecjalizowanych baz danych
- Wsparcie dla platform mobilnych

**Apress®**

Nr katalogowy: 8622

Księgarnia internetowa:  
<http://helion.pl>

Zamówienia telefoniczne:  
0 801 339900  
0 601 339900

**helion.pl**  
księgarnia internetowa

Sprawdź najnowsze promocje:  
• <http://helion.pl/promocje>  
Książki najchętniej czytane:  
• <http://helion.pl/bestsellery>  
Zamów informacje o nowościach:  
• <http://helion.pl/nowości>

**Helion**

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

sięgnij po WIĘCEJ



KOD KORZYŚCI

Cena 59,00 zł

ISBN 978-83-246-3922-9



9 788324 639229