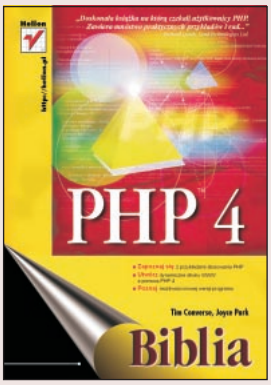
PHP 4. Biblia



Autorzy: Tim Converse, Joyce Park
Tłumaczenie: Paweł Gonera
ISBN: 83-7197-391-8
Tytuł oryginału: PHP 4. Bible
Format: B5, stron: 608
oprawa twarda

Chcesz tworzyć dynamiczne strony WWW współpracujące z bazami danych? Ten rzetelnie opracowany podręcznik zawiera analizy konkretnych problemów. Ułatwi zapoznanie się z najnowszą wersją języka umożliwiającego tworzenie skryptów dołączanych do stron HTML. Jeżeli jesteś projektantem HTML, programistą C lub twórcą stron WWW używającym języków ASP, JSP, Perl lub ColdFusion, podręcznik ten stanie się twoim przewodnikiem po PHP – produkcie dostępnym bezpłatnie. Dzięki zawartym w książce wskazówkom zamiast rozpoczynać pracę „od zera", możesz wykorzystać powszechnie dostępne skrypty, zmieniając je odpowiednio do swoich potrzeb. Kompletny opis PHP 4 to między innymi rozbudowane rozdziały pomagające Ci wykorzystać szybkość relacyjnych baz danych oraz umiejętne łączenie stron WWW z bazami danych.
Naucz się, jak:

- tworzyć witryny ze śledzeniem sesji;
- używać PHP do programowania obiektowego;
- przyłączać kod PHP bezpośrednio do programów do obsługi poczty elektronicznej;
- zabezpieczać witrynę przed atakami;
- wykorzystać praktycznie mechanizmy cookie i przekierowywania.

Jeśli PHP 4 może coś wykonać, ty tym bardziej to potrafisz...

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 12.

Przekazywanie danych pomiędzy stronami

W tym rozdziale:

- Dlaczego HTTP jest jak toczący się kamień?
- Argumenty GET
- Inne użycie adresów URL w stylu GET
- Argumenty POST
- Obsługa zmiennych GET/POST w PHP

W tym rozdziale zajmiemy się przekazywaniem danych, na przykład zmiennych, pomiędzy stronami WWW. Niektóre z tych informacji nie należą jedynie do PHP, ale są konsekwencją interakcji PHP i HTML lub samego protokołu HTTP.

HTTP jest protokołem bezstanowym

Ważne, abyś pamiętał, że sam protokół HTTP jest protokołem bezstanowym. Oznacza to, że każde wywołanie HTTP jest niezależne od innych, nie ma informacji na temat stanu klienta i nie jest zapamiętywane. Każde wywołanie powoduje utworzenie osobnego wątku, który ma za zadanie dostarczyć jeden plik, następnie jest niszczony (uściślając, wraca do puli dostępnych wątków).

Nawet jeżeli projekt twojej witryny zakłada nawigację tylko w jedną stronę (ze strony 1. tylko do strony 2., a następnie do strony 3. itd.), protokół HTTP nigdy nie będzie wiedział, czy użytkownik oglądający stronę 2. załadował wcześniej stronę 1. Nie możesz ustawić na stronie 1. zmiennej i oczekiwać, że zostanie ona zaimportowana do strony 2. tylko przez mechanizmy HTML. Możesz użyć HTML do wyświetlenia formularza, do którego można wprowadzić informacje, ale jeżeli nie zrealizujesz mechanizmów przekazujących dane do innej strony lub programu, zmienne znikną w momencie przejścia na inną stronę.

Dla takich przypadków stworzone zostały technologie przetwarzania formularzy, np. PHP. PHP przechwytuje zmienne przenoszone z jednej strony do drugiej i pozwala na ich późniejsze wykorzystanie. Jest niezwykle użyteczny do funkcji przenoszenia danych i pozwala na ich łatwe zastosowanie w wielu przypadkach.

Istnieją bardziej zaawansowane sposoby zapamiętania interakcji klienta z witryną, jak *cookie* oraz sesje. W tym rozdziale skupimy się tylko na podstawowych technikach przenoszenia danych pomiędzy stronami z zastosowaniem metod `GET` i `POST` z HTML.

Programiści ASP niezmiennie twierdzą, że PHP jest gorszy, ponieważ uważają, że zmienne sesji są czymś niezwykłym. Nie daj się zakrzyczeć, Microsoft używa mechanizmu *cookie* do przechowywania zmiennych sesji, co może powodować wiele problemów. Możesz o tym przeczytać w dodatku B.

Argumenty GET

Metoda `GET` pozwala na przesyłanie argumentów pomiędzy stronami za pomocą fragmentu adresu URL. W przypadku użycia tej metody do obsługi formularzu, `GET` powoduje dodanie nazw zmiennych i ich wartości do adresu URL podanego jako atrybut `ACTION`. Poszczególne zmienne rozdzielone są znakiem zapytania. Kompletny adres URL przesyłany jest do przetworzenia (w tym przypadku przez PHP).

Poniżej przedstawiamy przykładowy formularz HTML używający metody `GET`.

```
<HTML>
<HEAD>
<TITLE>Przykład GET część 1</TITLE>
</HEAD>
<BODY>
<FORM ACTION="http://localhost/php/baseball.php" METHOD="GET">
<P>Root, root, root for the:<BR>
<SELECT NAME="Team" SIZE=2>
<!-- Dobrze jest używać atrybutu VALUE, nawet w przypadkach, gdy
nie jest obowiązkowy. W tym przykładzie jest on absolutnie niezbędny.
-->
<OPTION VALUE="Cubbies">Chicago Cubs (National League)
<OPTION VALUE="Pale Hose">Chicago White Sox (American League)
</SELECT>
<P><INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

Gdy użytkownik wybierze jedną z wartości i kliknie przycisk, przeglądarka sklei bez dodatkowych odstępów następujące elementy:

- adres URL podany w apostrofach po słowie `ACTION` (*http://localhost/php/baseball.php*);
- znak zapytania (?);

- nazwę zmiennej, znak równości i jej wartość (Team=2);
- znak & oraz kolejną parę `NAME=VALUE` (Submit=Submit). Ta część może być powtarzana tyle razy, na ile pozwala ograniczenie długości adresu przez serwer.

Dla naszego przykładu utworzony zostanie następujący ciąg URL:

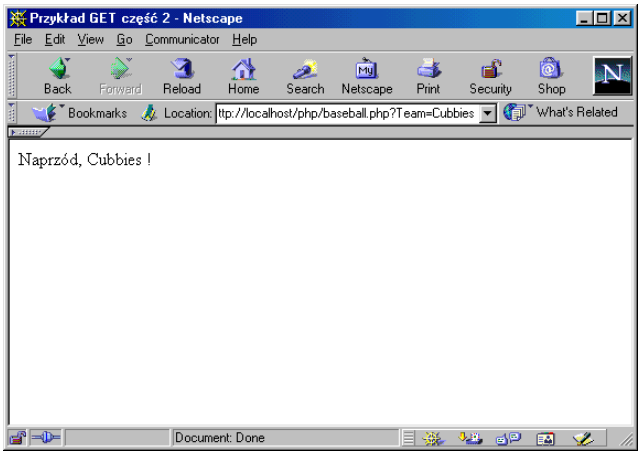
```
http://localhost/php/baseball.php?Team=Cubbies&Submit=Submit
```

Adres ten jest przesyłany przez przeglądarkę jako nowe żądanie. Skrypt PHP, do którego zostało przesłane powyższe żądanie, pobiera z adresu wartości `GET` i używa ich do stworzenia strony. W naszym przypadku wkleimy wybraną wartość do tekstu.

```
<HTML>
<HEAD>
<TITLE>Przykład GET część 2</TITLE>
<STYLE TYPE="text/css">
<!--
BODY {font-size: 24pt;}
-->
</STYLE>
</HEAD>
<BODY)
<P>Naprzód,
<?php print("$Team"); ?> !
</BODY>
</HTML>
```

Końcowy wynik przedstawiony jest na rysunku 12.1.

Rysunek 12.1. *Wynik działania formularza używającego METHOD=GET*



Obsługa formularzy za pomocą metody `GET` posiada jedną wielką zaletę w porównaniu z metodą `POST`. Tworzy ona różne adresy, które użytkownik może zapamiętać. Z wyniku działania formularza bazującego na metodzie `POST` nie da się zrobić zakładki.

Sposób obsługi formularzy za pomocą metody `GET` posiada tyle wad, że standard HTML 4.0 odradza jego stosowanie. Wadami tymi są między innymi:

- `GET` nie jest odpowiedni do logowania użytkownika, ponieważ nazwa użytkownika i hasło są widoczne na ekranie i mogą być zapisane w pamięci odwiedzonych stron w przeglądarce.

- Wysłane przez formularz dane zapamiętywane są w logach serwera WWW.
- Ponieważ `GET` przypisuje dane do zmiennej środowiskowej serwera, jej długość jest ograniczona. Próba przesłania np. 300 słów w adresie URL nie jest najlepszym pomysłem.

Oryginalna specyfikacja HTML określa maksymalną długość żądań na 244 znaki. Mimo że ograniczenie to zostało później zarzucone, używanie dłuższych ciągów nie jest dobrym pomysłem.

Ponieważ można zapamiętywać adresy utworzone przez metodę `GET`, pojawiło się wiele głosów za zachowaniem tej metody przez W3. Tak się stało i mimo że jest to metoda najczęściej używana do obsługi formularzy, jest zalecana tylko dla powtarzalnych zastosowań, czyli takich, które nie powodują stałego efektu ubocznego. Najbardziej odpowiednim zastosowaniem `GET` jest okno przeszukiwania. Dla innych formularzy lepiej użyć metody `POST`.

Inne zastosowania adresów URL w stylu GET

Mimo że w chwili obecnej obsługiwanie formularzy przez metodę `GET` nie jest zalecane, adresy URL w tym stylu są bardzo użyteczne do obsługi nawigacji po witrynie. Szczególnie użyteczne jest użycie ich w witrynach generowanych dynamicznie, które są często tworzone przy użyciu PHP, ponieważ adresy zawierające dołączone zmienne świetnie współpracują z systemem szablonów zależnych od kontekstu.

Załóżmy, że jesteś właścicielem witryny traktującej o zwierzętach hodowanych dla wełny. Po długiej i ciężkiej pracy nad stroną informacyjną i graficzną masz następujące strony:

```
alpaca.html
guanaco.html
llama.html
vicuna.html
```

Lecz gdy witryna zacznie się rozrastać, płaska struktura plików może wymagać dużo pracy przy administracji, szczególnie gdy proste poprawki trzeba będzie wykonać na każdej stronie. Jeżeli struktura stron jest podobna, możesz zastanowić się nad zastosowaniem systemu szablonów razem z PHP.

Można zastosować pojedynczy szablon oraz oddzielne pliki tekstowe dla każdego zwierzęcia (zawierające dane, fotografie itp.):

```
fleecee.php
  alpaca.inc
  guanaco.inc
  llama.inc
  vicuna.inc
```

Można również użyć większej liczby wyspecjalizowanych szablonów:

```
goat.php
  cashmere.inc
insect.php
  silkworm.inc
llamoid.php
  alpaca.inc
rabbit.php
  angora.inc
sheep.php
  merino.inc
```

W obu przypadkach plik szablonu będzie wyglądał podobnie (nie zamieściliśmy wszystkich niezbędnych plików, więc przykład ten nie będzie działał):

```
<HTML>
<HEAD>
<TITLE>Zwierzęta hodowane dla wełny</TITLE>
<STYLE TYPE="text/css">
<!--
BODY {font: verdana; font-size: l4pt}
-->
</STYLE>
</HEAD>
<BODY>
<TABLE BORDER=0 CELLPADDING=0 WIDTH=100%>
<TR>
<!- Panel nawigacji z adresami w stylu GET. -->
<TD BGCOLOR="#428284" ALlGN=CENTER VALIGN=TOP WIDTH=25%>
<P>
<A HREF="fleecee.php?Name=alpaca"><B>Alpaca</B></A>
<BR>
<A HREF="fleecee.php?Name=guanaco"><B>Guanaco</B></A>
<BR>
<A HREF="fleecee.php?Name=llama"><B>Llama</B></A>
<BR>
<A HREF="fleecee.php?Name=vicuna"><B>Vicu&#241a</B></A>
<BR><BR>
</TD>
<TD BGCOLOR="#FFFFFF" ALIGN=LEFT VALIGN=TOP WIDTH=75%>
<? include("$Name.inc"); ?>
</TD></TR></TABLE>
</BODY>
</HTML>
```

Łącza na panelu nawigacji są obsługiwane przez przeglądarkę tak samo jak wynik działania metody `GET`.

Jednak nawet w przypadku takiego rozwiązania masz jeszcze wiele ręcznej pracy: należy się upewnić, że każdy włączany plik jest prawidłowo sformatowany, dodanie nowego pliku do witryny wymaga dodania nowego łącza i innych tego typu prac. Zgodnie z zasadą oddzielania formy od treści możesz przejść na wyższy poziom abstrakcji przy użyciu bazy danych. W tym przypadku adres typu:

```
fleecee.php?animalID=2
```

powinien spowodować sięgnięcie przez szablon do bazy danych (użycie zmiennej numerycznej zamiast słowa przyspiesza interakcję z bazą danych). Taki system powinien automatycznie umieszczać na panelu na podstawie zawartości bazy danych. Dzięki temu możesz tworzyć kolejne strony WWW bez interwencji.

Argumenty POST

POST jest zalecaną metodą obsługi formularzy, szczególnie dla niepowtarzalnych zastosowań (takich, które skutkują trwałymi efektami działania), takich jak dodawanie danych do bazy. Zbiór danych formularza jest zawarty w ciele formularza w czasie jego wysyłania do agenta przetwarzania (w tym przypadku PHP). Nie ma żadnych widocznych zmian w adresie URL w zależności od przesyłanych danych.

Metoda POST posiada następujące zalety:

- Jest dużo bezpieczniejsza od GET, ponieważ wprowadzone przez użytkownika dane nie są widoczne w adresie URL, logach serwera oraz na ekranie (jeżeli są prawidłowo obsłużone).
- Maksymalny rozmiar przesyłanych danych jest dużo większy (kilka kilobajtów zamiast kilku setek znaków).

Metoda ta posiada również wady:

- Nie można utworzyć zakładki z wyniku działania.
- Metoda ta może być niezgodna z niektórymi ustawieniami firewalla, jeżeli z powodów bezpieczeństwa usuwa on dane z formularza.

W książce tej konsekwentnie używamy POST do obsługi formularzy umieszczających dane w systemie. GET używamy do nawigacji po witrynie oraz do obsługi przeszukiwania, czyli zastosowań wymagających odczytania danych i ich wyświetlenia.

Wiele, jeżeli nie większość zastosowań POST powoduje zainicjowanie połączenia w usługowej warstwie systemu. Przykład pokazany na wydruku 12.1 jednak tego nie wykonuje (ponieważ skupiamy się na metodzie POST, usunęliśmy sprawdzanie poprawności danych, więc formularz ten wymaga jeszcze pracy).

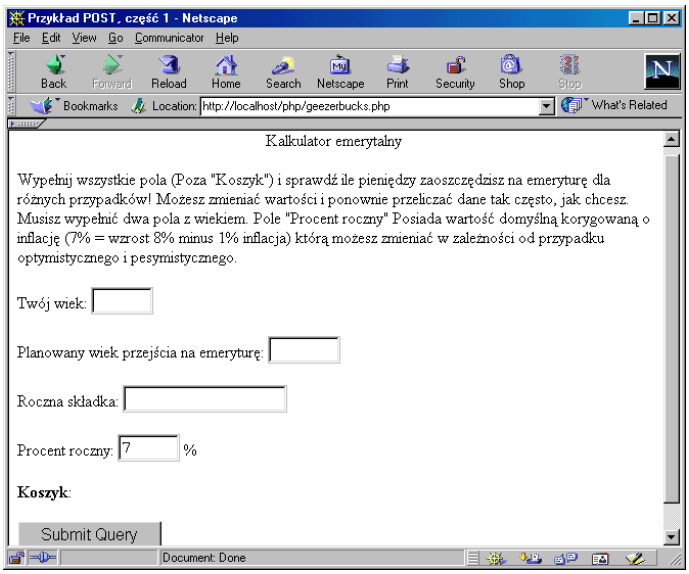
Wydruk 12.1. *Arkusz oszczędności emerytalnych*

```
<HTML>
<HEAD>
<TITLE>Przykład POST, część 1</TITLE>
<STYLE TYPE="text/css">
<!--
BODY        {font-size: 14pt}
.heading    {font-size: 18pt; color: red}
-->
</STYLE>
</HEAD>
<?php
/* To sprawdzenie pozwala na stwierdzenie, czy formularz
jest wyświetlony po raz pierwszy (w tym przypadku wyświetla tylko
domyślny procent roczny), czy już zostały wysłane wprowadzone wartości
 */
if (!IsSet($stage))
  $AnnGain = 7;
else
{
  $Years = $RetireAge - $CurrentAge;
  $YearCount = 0;
```

```
  $Total = $Contrib;
  while($YearCount <= $Years)
  {
    $Total = round($Total * (1.0 + $AnnGain/100) + $Contrib);
    $YearCount = $YearCount + 1;
  }
}
?>
<BODY>
<DIV ALIGN=CENTER ID=Div1 class=heading>
Kalkulator emerytalny
</DIV>
<P class=blurb>Wypełnij wszystkie pola (Poza "Koszyk") i sprawdź,
ile pieniędzy zaoszczędzisz na emeryturę dla różnych przypadków!
Możesz zmieniać wartości i ponownie przeliczać dane tak często,
jak chcesz. Musisz wypełnić dwa pola wieku. Pole "Procent roczny"
posiada wartość domyślną korygowaną o inflację
(7% = wzrost 8% minus 1% inflacja),
którą możesz zmieniać w zależności od przypadku optymistycznego i
pesymistycznego.</P>
<FORM ACTION="<?php print("$PHP_SELF"); ?>" METHOD="POST">
<P>Twój wiek: <INPUT TYPE="text" SIZE=5 NAME="CurrentAge"
VALUE="<?php print("$CurrentAge"); ?>">
<P>Planowany wiek przejścia na emeryturę: <INPUT TYPE="text"
SIZE=6 NAME="RetireAge" VALUE="<?php print("$RetireAge"); ?>">
<P>Roczna składka: <INPUT TYPE="text" SIZE=15
NAME="Contrib" VALUE="<?php print("$Contrib"); ?>">
<P>Procent roczny: <INPUT TYPE="text" SIZE=5 NAME="AnnGain"
VALUE="<?php print("$AnnGain"); ?>"> %
<BR><BR>
<P><B>Koszyk</B>: <?php print("$Total"); ?>
<P><INPUT TYPE="hidden" NAME="stage" VALUE=1>
<P><INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

Na rysunku 12.2 przedstawiony jest formularz z wydruku 12.1.

Rysunek 12.2. *Formularz używający metody POST*



Zarządzanie zmiennymi w PHP

PHP tak efektywnie przenosi dane, ponieważ projektanci podjęli bardzo wygodną, lecz (w teorii) ryzykowną decyzję. PHP automatycznie i niewidocznie dla użytkownika przypisuje zmienne na nowej stronie, gdy wyślesz je za pomocą `GET` lub `POST`. Większość z jego konkurentów wymaga jawnego wykonania tych przypisań na każdej stronie. Jeżeli o tym zapomnisz lub popełnisz błąd, dane nie będą dostępne dla agenta przetwarzania. PHP jest szybszy, prostszy i w większości przypadków odporny na pomyłki.

Najprostszą metodą zilustrowania tej różnicy jest pokazanie różnych metod przetwarzania danych z tego samego formularza. Formularz ten wygląda następująco:

```
<HTML>
<HEAD>
<TITLE>Formularz preferencji słodyczy</TITLE>
</HEAD>
<BODY>
<FORM ACTION="candy.php" METHOD="POST">
Jakie słodycze lubisz najbardziej?<BR>
<INPUT TYPE="radio" NAME="Candy" VALUE="orzeszki ziemne">Orzeszki ziemne<BR>
<INPUT TYPE="radio" NAME="Candy" VALUE="snickers">Snickers<BR>
<INPUT TYPE="radio" NAME="Candy" VALUE="krówki">Krówki<BR>
<INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

Skrypt PHP obsługujący formularz:

```
<HTML>
<HEAD>
<TITLE>Odpowiedź na wybór preferencji</TITLE>
</HEAD>
<BODY>
Hmmm
<?php
print("$Candy! ");
if ($Candy == "orzeszki ziemne")
  print("Istnieje kilka świetnych rodzajów lodów zawierających małe
         lub połamane $Candy.");
else
{
  print("Myślę, że jeszcze nie istnieją lody zawierające $Candy, ");
  if($Candy == "snickers")
    print("ale czy próbowałeś loda $Candy ?");
  elseif($Candy == "krówki")
    print("ale bardzo potrzebne są lody zawierające $Candy.");
}
?>
</BODY>
</HTML>
```

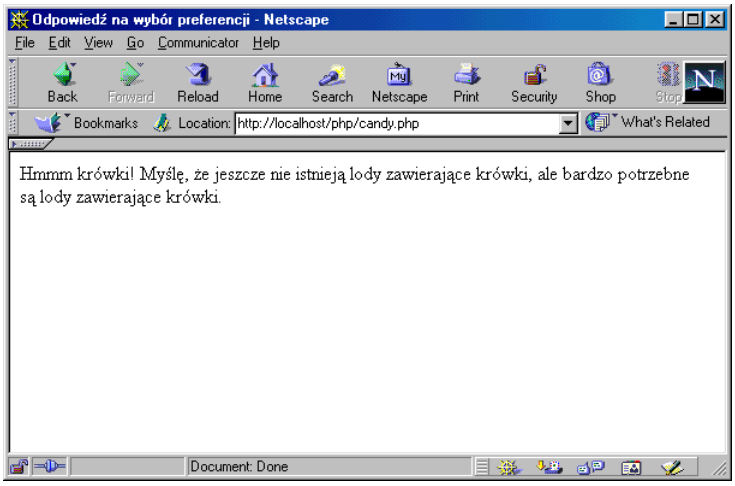
Skrypt ASP realizujący te same funkcje (aby go użyć zmień argument `ACTION` formularza na `candy.asp`):

```
<HTML>
<HEAD>
<TITLE>Odpowiedź na wybór preferencji</TITLE>
</HEAD>
<BODY>
Hmmm
<% Candy = Request.Form("Candy") %>
```

```
<%= Response.Write (Candy)%>
<% If Candy = "orzeszki ziemne" Then %>
Istnieje kilka świetnych rodzajów lodów zawierających małe lub połamane
<% Response.Write(Candy) %>.
<% Else %>
Myślę, że jeszcze nie istnieją lody zawierające <%
Response.Write(Candy) %>,
<% End If %>
<% If Candy = "snickers" Then %>
ale czy próbowałeś loda <% Response.Write(Candy) %> '?
<% Else If Candy = "krówki" Then %>
ale bardzo potrzebne są lody zawierające <%
Response.Write(Candy) %>.
<% End If %>
<% End If %>
</BODY>
</HTML>
```

Wynik jest za każdym razem taki sam jak na rysunku 12.3.

Rysunek 12.3. *Wynik użycia PHP lub ASP*



Używanie GET i POST

Czy wiesz, że PHP może używać zarówno `GET`, jak i `POST` jednocześnie na jednej stronie? Możesz rzucić się w wir dynamicznego tworzenia formularzy!

Szybko pojawia się pytanie: co się stanie, jeżeli (rozmyślnie lub nie) użyję tej samej nazwy zmiennej zarówno w tablicach `GET`, jak i `POST`. PHP przechowuje zmienne `GET`, `POST` oraz `COOKIE` w tablicach o nazwach `$HTTP_GET_VARS`, `$HTTP_POST_VARS` oraz `$HTTP_COOKIE_VARS`, jak również w tablicy `$GLOBALS`. Jeśli wystąpi konflikt, rozwiązywany jest przez nadpisywanie wartości zmiennych w kolejności ustalonej przez opcję `gpc_order` z pliku *php.ini* (należy ustawić również opcję `track_vars`). Późniejszy wygrywa nad wcześniejszym, więc jeżeli użyjesz ustawienia domyślnego `GPC`, *cookie* wygrają z `POST`, a te z `GET`. Możesz zarządzać kolejnością nadpisywania zmiennych, zmieniając kolejność tych trzech liter w odpowiednim wierszu pliku *php.ini*.

Ponieważ PHP posiada zdolność automatycznego przypisywania zmiennych, skrypt PHP pisze się szybciej. W skrypcie PHP po prostu używamy zmiennej bez konieczności pobierania jej wartości lub ponownej definicji. Przykład, jaki podaliśmy, jest trywialny, ponieważ przekazywana jest tylko jedna zmienna. Jednak przy bardziej złożonych formularzach z wieloma zmiennymi lub w przypadku serii formularzy współdzielących wiele zmiennych PHP pozwala zaoszczędzić wiele czasu.

W skrypcie ASP należy odczytać każdą zmienną przysłaną za pomocą metody `POST`, w PHP nie jest to konieczne. W przypadku ASP należy użyć `Request.Form` w przypadku metody `POST` oraz `Request.QueryString` w przypadku metody `GET`. Jeżeli zmienisz metodę przesyłania danych, musisz wszędzie zmienić nazwę kolekcji, co może być źródłem błędów.

Automatyczne przypisywanie zmiennych w PHP również może być źródłem konfliktów nazw zmiennych. PHP po prostu zamieni starą wartość zmiennej na nową. Możesz sterować kolejnością zapisywania zmiennych za pomocą opcji `GPC` z pliku *php.ini*. Najlepszą metodą jest jednak unikanie konfliktów i nadawanie dobrych, zróżnicowanych nazw zmiennych. Większość użytkowników PHP uważa, że użyteczność funkcji automatycznego przypisywania zmiennych przewyższa zagrożenie potencjalnymi konfliktami nazw zmiennych.

Podsumowanie

Protokół HTTP jest protokołem bezstanowym. Oznacza to, że sam HTML nie potrafi wymieniać danych pomiędzy stronami witryny WWW. Może zostać użyty do przekazywania wartości, ale zewnętrzny program obsługi formularza musi wysilić się, aby przekazać wartość. PHP jest prawdopodobnie najbardziej naturalnym programem obsługi formularza.

Dane można przekazywać za pomocą trzech podstawowych metod: `GET`, `POST` oraz `cookie` (którymi zajmiemy się w rozdziale 26.). Metoda `GET` jest używana do tworzenia złożonych adresów URL, stosowanych wraz z szablonami w dynamicznych witrynach. Nie jest to jednak metoda zalecana do obsługi formularzy, w przeciwieństwie do `POST`.