helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

PHP. Czarna księga

Autor: Peter Moulding
Tłumaczenie: Marcin Pancewicz
ISBN: 83-7197-695-X
Tytuł oryginału: PHP Black Book
Format: B5, stron: 738

Skrypty PHP są osadzone w dokumentach HTML, dzięki czemu powstają aktywne strony WWW. W tej książce znajdziesz opis metod i funkcji potrzebnych, by nauczyć się budowania dynamicznych witryn WWW. Peter Moulding, od ponad 25 lat zajmujący się programowaniem, pomoże Ci w zdobyciu wiedzy niezbędnej do zastosowania PHP w środowisku Unix lub Windows. Dowiesz się wszystkiego, co jest potrzebne do tworzenia stabilnych i efektywnie działających witryn WWW. W książce zostały opisane wszystkie funkcje PHP, włącznie z funkcjami obsługi baz danych. Dzięki tej książce będziesz mógł dopisać do listy posiadanych umiejętności dogłębną znajomość PHP.

- Programowanie witryn WWW za pomocą PHP
- Tworzenie stabilnego, wydajnego kodu, działającego w systemach Windows i Unix
- Wysyłanie i odbieranie poczty za pomocą stron WWW
- Korzystanie z baz danych MySQL oraz PostgreSQL
- Pobieranie informacji z dowolnej bazy lub serwera
- Stosowanie cookies, sesji, zabezpieczeń oraz profilów użytkowników
- Automatyzowanie administrowania systemem, dostarczanie danych oraz wykonywanie innych codziennych zadań
- Zastosowanie formularzy, tablic, klas i obiektów
- Konfiguracja i wykorzystywanie ważnych technologii sieciowych: XML, LDAP, IMAP, MIME, WDDX, Z39.50, YAZ, XSLT oraz XSL
- Generowanie obrazków JPEG i PNG oraz plików Flash

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

Spis treści

Rozdział 9.

Formularze

Szczegółowe omówienie

Formularze tworzy się w PHP dość łatwo, a dzięki tablicom można w nich bez trudu umieszczać nawet długie listy pozycji. Łatwość prezentacji zmiennych z cookies, kwerend URL oraz formularzy z metodami `GET` i `POST` oznacza, że możesz szybko przejść do wielostronicowych formularzy oraz specjalnych mechanizmów, na przykład wysyłania plików do serwera. Kontrolki formularzy mogą być używane także z czymś, co nie jest powiązane z tradycyjnymi formularzami, na przykład z alternatywną nawigacją, opisywaną w następnym podrozdziale. Możesz używać przycisków *Wyślij* w formularzach jako alternatywy dla obrazków i łączy podczas nawigacji.

W tym rozdziale położę nacisk na sposoby tworzenia pytań w formularzach, tak, abyś mógł łatwo i szybko tworzyć formularze bez konieczności ponownego pisania kodu. Wzorcowa funkcja formularza i pytań, budowana w podrozdziale „Natychmiastowe rozwiązania", pozwoli na dodawanie pytań i formatowanie formularza bez modyfikacji kodu. Jeśli przeczytałeś rozdziały 5. i 20., możesz zechcieć skierować dane otrzymane od funkcji formularza do bazy danych lub pliku XML. Korzystanie z bazy danych będzie odpowiednie dla stron, które ulegają gwałtownym przemianom.

Alternatywna nawigacja

Powszechnym sposobem łączenia stron jest użycie znacznika zakotwiczenia, `<a>`. Znacznik ten przekazuje zmienne poprzez łańcuch URL, ale takie łańcuchy szybko stają się bardzo złożone, gdy ładuje się je ze zmiennymi. Formularze mogą przekazywać wartości za pomocą bardziej przejrzystej metody `POST`, która działa zarówno z bardzo obszernymi wartościami, jak i z bardzo długimi listami wartości (zdarzyło mi się, że za pomocą metody `POST` przekazywałem z jednego formularza do drugiego czterdzieści dziewięć tysięcy nazw plików).

W przypadku formularzy znacznik `<form>` odpowiada znacznikowi `<a>`, znaczniki `<input>` odpowiadają poszczególnym pozycjom łańcucha kwerendy, zaś przycisk wysyłania formularza odpowiada tekstowi, jaki umieszczasz pomiędzy początkowym a końcowym znacznikiem zakotwiczenia. Przedstawiony dalej przykład pokazuje standardowe łącze z przygotowaną kwerendą URL; tuż po nim zostało pokazane to samo łącze, uzyskane

dzięki użyciu formularza. Kwerenda oferuje mechanizm wyłączenia już przeglądanych dokumentów; w pierwszym przykładzie każdy wyłączony dokument jest nazywany z użyciem pola o nazwie `exclude01`. W przykładzie z formularzem wykorzystywane jest w tym celu pole `example[]`, powodujące utworzenie prostej tablicy w PHP. Z czasem, gdy przeglądarka odwiedzi już tuziny dokumentów, pierwsza wersja łącza zacznie powodować w większości przeglądarek błędy, podczas gdy w drugiej wersji może zostać obsłużonych wielokrotnie więcej nazw plików, niż jestem w stanie odwiedzić podczas jednej sesji.

```
$link = "<a href=\"query.html?"
    . "sortby=date&sortorder=descending&country=Suriname"
    . "&type=document&author=cia"
    . "&exclude01=d1991122&exclude02=d3010304"
    . "&find=tropical+rain+forest+conservation\">Find</a>";

$link = "<form action=\"query.html\" method=\"post\">"
    . "<input type=\"hidden\" name=\"sortby\" value=\"date\">"
    . "<input type=\"hidden\" name=\"sortorder\" value=\"descending\">"
    . "<input type=\"hidden\" name=\"country\" value=\"Suriname\">"
    . "<input type=\"hidden\" name=\"type\" value=\"document\">"
    . "<input type=\"hidden\" name=\"author\" value=\"cia\">"
    . "<input type=\"hidden\" name=\"exlude[]\" value=\"d1991122\">"
    . "<input type=\"hidden\" name=\"exlude[]\" value=\"d3010304\">"
    . "<input type=\"hidden\" name=\"find\" value=\""
      . "tropical rain forest conservation\">"
    . "<input type=\"submit\" value=\"Find\">"
    . "</form>";
```

HTML czy czyste PHP?

PHP dobrze łączy się z kodem HTML, ale taki kod trudno odczytać, a przy tym, przy każdym przejściu do HTML, traci się zalety skryptu PHP. Potęgę i elastyczność zyskuje się budując cały formularz, łącznie z każdym znacznikiem i każdym parametrem, w czystym kodzie PHP. Skrypty przedstawione w tym rozdziale generują każdy element znaczników HTML, dzięki czemu mamy absolutną kontrolę nad zawartością i formatem.

W miarę, jak przeglądarki będą coraz lepiej obsługiwać arkusze CSS (Cascading Style Sheets), będziesz mógł redukować bezpośrednie formatowanie zawartości strony, ale pamiętaj, by dokładnie przeanalizować rynek, zanim zastąpisz stare znaczniki `<font>` czymś nowym. Swojego czasu pomagałem agencji marketingowej w przygotowaniu prezentacji dla szefa pewnej bardzo dużej firmy. Szef obejrzał prezentację, skopiował listę referencyjną z witrynami stworzonymi przez agencję, po czym pojechał do domu. Późną nocą użył starego komputera i wolnego modemu swojego syna, aby sprawdzić otrzymane witryny i okazało się, że część z nich była nieczytelna. Agencja straciła to zlecenie.

Rozwiązanie problemów szybkości i zgodności ma zasadnicze znaczenie w formularzach rejestracyjnych witryny, gdyż właśnie te formularze są wyświetlane użytkownikom jako pierwsze. Niektóre witryny tracą tysiące klientów, gdyż ich formularze rejestracyjne nie działają we wszystkich przeglądarkach. Dodając do tego słabe dzienniki, raporty i analizę prowadzone przez wiele witryn, firmy nawet nie wiedzą, dlaczego i ilu klientów tracą. Tworząc początkowy formularz rejestracyjny, formularz zapytania, formularz z informacjami zwrotnymi czy formularz zgłaszania problemu, dokładnie upewnij się, że jest on łatwy w użyciu i doskonale stabilny w każdej z przeglądarek.

Największym problemem może okazać się walka z wydziałem marketingu o pozbycie się ze strony formularza natarczywych pytań, rozpraszającej grafiki i rozbudowanego tekstu. Z jakichś powodów niektórzy ludzie nie krępują się zadawać na stronie pytań, na które klienci rzadko odpowiadają w drukowanych formularzach. Co więcej, niektórzy projektanci czynią takie pytania obowiązkowymi i dziwią się, że klienci nie wypełniają formularzy.

Na nawigowanie za pomocą formularzy wpływa także szybkość, z jaką możesz zmieniać strony; jeśli nie możesz szybko ich zmienić, stają się bezużyteczne i mylące. Okaże się, że kierownicy i pracownicy marketingu zażądają natychmiastowej zmiany strony dziesięć minut po zakończeniu pracy przez zespół testujący; zmiany te mogą obejmować zarówno coś tak prostego, jak usunięcie oferty ze specjalną ceną, jak i coś tak ważnego, jak usunięcie nielegalnej strony, przynoszącej kary w wysokości tysięcy dolarów dziennie. Kilka lat temu, gdy czekałem na windę, by pojechać do domu, usłyszałem krzyk i poszedłem sprawdzić, co się dzieje. Błyskotliwy, młody kierownik marketingu gwałtownie rozrastającej się witryny handlowej wyglądał tak, jakby miał za chwilę dostać ataku serca, ponieważ na stronach ich całkiem nowej linii produktów, a mianowicie papierosów, zabrakło wymaganych ostrzeżeń o zachorowaniach. Na szczęście zaprojektowałem tę witrynę w taki sposób, by móc szybko dokonywać zmian, dlatego mogłem rozwiązać ten problem zanim jeszcze winda przybyła na piętro.

Jedną z cech, jaką daje Ci kod przedstawiony w tym rozdziale, jest spójność. Im bardziej spójnym uczynisz format i układ swoich pytań i formularzy, tym łatwiej będzie klientom wypełniać kolejne formularze w witrynie. Spójność będzie Ci także pomocna na dłuższą metę; jest to kolejnym powodem, by generować wszystko za pomocą kodu PHP i nie korzystać ze wstawek w HTML.

Część kodu w tym rozdziale obsługuje listy wyboru z wieloma pozycjami. Jeśli korzystasz z list wyboru, weź pod uwagę możliwość wpisania dowolnego, alternatywnego tekstu. Na przykład, gdy na pewnych amerykańskich witrynach rozpoczynano sprzedaż poza granicami kraju, formularze zamówień wymagały podania adresu, ale listy wyboru stanu przyjmowały wyłącznie stany amerykańskie, przez co 5,6 miliarda z 5,9 miliarda osób na tej planecie nie mogło podać poprawnego adresu. Obecnie większość stron działa już lepiej i oferuje możliwość wybrania innego stanu (lub pominięcia wyboru stanu), podania kodu pocztowego innego, niż składający się z pięciu cyfr (lub pominięcia i tego wyboru) oraz podania numeru telefonu z długim prefiksem. Na szczęście, w cudownej dobie poczty elektronicznej, numery faksów zwykle nie są już wymaganą odpowiedzią.

Minimalny kod HTML

Formularz HTML wymaga:

- początkowego znacznika, `<form>`,
- któregoś ze znaczników danych, na przykład znacznika `<input>`,
- znacznika akcji, jak `<input type="submit">`,
- znacznika końcowego, `</form>`.

Znaczniki te można umieszczać wewnątrz struktur układu strony (na przykład jako komórki tabeli) lub ująć całą tabelę w znacznik <form> i umieścić poszczególne znaczniki <input> w osobnych komórkach, tak jak w następującym przykładzie:

```
<form action="page.html" method="post">
<input type="text" name="cokolwiek">
<input type="submit" name="doit">
</form>
```

W wielu przeglądarkach znacznik <form> powoduje irytujący efekt wstawienia odstępu. W niektórych początek formularza wstawia podział linii, podobny do podziału występującego w tabelach. Także znacznik końca formularza może wstawiać podział linii lub spację.

Formularze mogą zawierać inne elementy formatowania, na przykład tabele sterujące układem poszczególnych elementów w formularzu. Można także stosować akapity i szeroki zakres elementów formatowania HTML. Istnieją reguły określające, co może trafić do wnętrza formularza oraz co może otaczać formularz. Większość przeglądarek łamie te reguły — niektóre dzielą linie tam, gdzie nie powinny, inne ignorują bardziej skomplikowane opcje formatowania, jeszcze inne akceptują źle sformułowany kod HTML — dlatego bądź raczej konserwatywny i użyj minimum kodu HTML, wymaganego do stworzenia logicznego i czytelnego formularza. Testuj go, korzystając z różnych przeglądarek, gdyż Twoja ulubiona przeglądarka może akceptować takie pomyłki, które w innych przeglądarkach mogą zniszczyć całą stronę.

Gdy zakończysz formularz na innym poziomie znacznika HTML, niż go zacząłeś, spowodujesz problemy. Na przykład, jeśli zaczniesz formularz poza tabelą, a następnie zakończysz go wewnątrz komórki tabeli, niektóre przeglądarki nie będą wiedziały, co z tym zrobić. Twoje struktury powinny być jednolite i symetryczne. Ja zwykle tworzę znacznik <input>, następnie ujmuję go pomiędzy znaczniki <td> i </td>, całość ujmuję znacznikami <tr> i </tr>, potem umieszczam wszystko w znacznikach tabeli, a następnie w znacznikach formularza. Gdy chcę określić położenie formularza na stronie, cały formularz wstawiam do komórki tabeli.

Możesz skorzystać z zagnieżdżania i symetrii znaczników formularzy i tabel, tworząc z każdej z warstw funkcję, a następnie zagnieżdżając te funkcje. Testowałem witryny za pomocą tuzinów przeglądarek w różnych wersjach; każda z listy moich 30 ulubionych przeglądarek dobrze radziła sobie z elementami formularza, ułożonymi na poziomach pasujących do innych znaczników HTML. Tworzenie formularzy za pomocą zagnieżdżonych funkcji PHP jest dużo bardziej stabilne niż tworzenie ich za pomocą Dreamweavera i innych drogich programów do tworzenia układu strony.

Dzwonki i gwizdki

HTML nie posiada znacznika <dzwonek>, zaś PHP nie zawiera funkcji `gwizdek()`. Musisz dokładnie rozważyć poziom złożoności, jaki chcesz umieścić w formularzu, aby spełnić wygórowane wymagania wydziału marketingu. PHP pozwala na tworzenie formularza za pomocą prostych funkcji, tablic i złożonych obiektów. To, czego potrzebujesz, to jak najprostszy sposób zmiany wyglądu formularza, gdy osoby testujące zgłoszą wszystkie problemy, jakie w nim odkryli.

Dobrym sposobem jest oddzielenie formatowania, kolejności i danych, aby można było zmieniać kolejność pytań bez zmiany formatowania oraz by ktoś inny mógł zmienić treść pytania bez wymagania od Ciebie modyfikacji kodu strony. Treść pytań możesz przechowywać w tablicy przetwarzanej w pętli, jak pokazano w następnym przykładzie (szczegóły korzystania z tablic przedstawia rozdział 3.). Kolejność pytań może zostać zmieniona przez zwykłą zamianę kolejności pozycji tablicy.

```
$question[] = "imię";
$question[] = "adres";
$question[] = "miejscowość";
$question[] = "kraj";
while(list($k, $v) = each($question))   {
   print("<br>Podaj: " . $v);
   }
```

W przedstawionym tu kodzie mógłbyś użyć pętli `for()`, ale konstrukcja `while(each())` ma pewne zalety, do których należy zwrócenie właściwej następnej pozycji tablicy nawet wtedy, gdy wewnątrz pętli wstawiasz lub usuwasz pozycje tej tablicy.

Jeśli masz bardziej złożone nazwy, łatwiejsze może okazać się identyfikowanie pozycji według ich indeksu w tablicy. Gdy przetwarzasz listę, możesz użyć indeksu elementu, aby ponownie znaleźć go w tablicy. Rozważmy następującą listę elementów trafiających do formularza:

```
$fruit[] = "Akee";
$fruit[] = "Black Sapote";
$fruit[] = "Durian";
$fruit[] = "Guaraná";
$fruit[] = "Otaheite Gooseberry";
```

Guraná jest ważnym źródłem kofeiny; w jej nazwie zawiera się akcentowane á, które sprawia kłopot osobom próbującym wpisać je w polu tekstowym lub w kodzie. Nazwy tego rodzaju powinny znaleźć się na liście wyboru, abyś nie musiał przetwarzać znaków. Gdy zamieniasz listę na pozycję wyboru, jako identyfikatora użyj indeksu elementu listy.

Przy przetwarzaniu tablic korzystaj z technik przetwarzania tablic i list. Najpierw wyzeruj wskaźnik tablicy, przechodząc na jej początek, po czym przetwórz każdą pozycję za pomocą pętli `while()` oraz instrukcji `list()` i `each()`. W podrozdziale „Natychmiastowe rozwiązania" znajdziesz mnóstwo przykładów kodu. Głównym celem jest uniknięcie użycia konkretnej nazwy jako identyfikatora przez przetwarzanie listy jako całości, oraz uniknięcie kodu, wymagającego identyfikowania konkretnego elementu.

Tak się składa, że tablice w PHP uwzględniają wielkość liter, dlatego pozycja `$fruit["Apple"]` różni się od pozycji `$fruit["apple"]`. Bądź uważny przy łączeniu nazw wyświetlanych z nazwami używanymi do identyfikacji elementów. Osobiście korzystam — w celu usunięcia spacji z wyświetlanej nazwy i zmiany wszystkiego na małe litery — z następującej sztuczki:

```
$display_name = "Macadamia nuts";
$id = strtolower(str_replace(" ", "", $display_name));
```

Techniki zmiany tekstu strony na podstawie języka znajdziesz w rozdziałach 12. i 20.; dowiesz się w nich, jak wprowadzać tekst w formacie o określonej strukturze. Oba rozdziały pokazują sposoby, jakimi możesz się posłużyć, by przetwarzać obszerne listy

tekstu tworzonego przez inne osoby; są to techniki, których możesz użyć do formatowania swoich formularzy. Oddzielając kolejność i formatowanie od treści pytań, możesz skoncentrować się na tworzeniu strony i innych zagadnieniach, na przykład na rodzaju pytań, jakie zadajesz.

Pytania formularza

HTML oferuje różne sposoby zadawania pytań i otrzymywania odpowiedzi, od prostych przycisków opcji i przycisków radiowych po rozbudowane listy wyboru. Chcesz, by pytanie było zrozumiałe dla klienta i by ograniczało jego odpowiedzi do poprawnych możliwości. Możesz łączyć rodzaje pytań; możesz na przykład dać do wyboru trzy przyciski radiowe oraz pole tekstowe, w którym użytkownik może wpisać ewentualny komentarz.

Oto przykładowe pomyłki, które często zdarzają się na amerykańskich witrynach, próbujących prowadzić sprzedaż w Azji i w krajach Pacyfiku. Witryny te:

- zawierają listy wyboru stanu, które pozwalają wybrać wyłącznie stany USA;
- żądają podania stanu, bez brania pod uwagę faktu, iż wiele krajów nie posiada odpowiednika amerykańskiego stanu;
- zakładają, że każdy kraj korzysta z kodów pocztowych;
- zakładają, że kod pocztowy ma zawsze pięć cyfr.

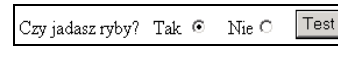
Każdy kraj posiada własną specyfikę, co sprawia, że projektowanie międzynarodowych formularzy jest interesującym zagadnieniem.

Pytania Tak/Nie

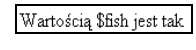
Przedstawione tu pytania pasują do przycisków radiowych w formularzach HTML:

- Czy jadasz ryby? Tak/Nie
- Czy jadasz przetwory mleczne? Tak/Nie
- Czy życzysz sobie miejsce przy oknie? Tak/Nie

Przyciski radiowe są najbardziej przydatne wtedy, gdy chcesz uzyskać tylko jedną odpowiedź z listy kilku możliwych. W przeglądarce pytanie wygląda tak, jak pokazano na rysunku 9.1. Odpowiedź powinna wyglądać tak, jak pokazano na rysunku 9.2.



Rysunek 9.1. *Pytanie Tak/Nie z zastosowaniem przycisków radiowych*



Rysunek 9.2. *Odpowiedź na pytanie Tak/Nie*

A jak wygląda to w kodzie HTML i w PHP? Kod HTML ma postać:

```
<form action="radiobutton.html" method="post">
Czy jadasz ryby?   Tak
```

```
<input type="radio" name="fish" value="tak" checked>
   Nie<input type="radio" name="fish" value="nie">
   <input type="submit" value="Test">
</form>
```

Zwróć uwagę, że oba przyciski radiowe mają nazwę `fish`, oraz oba mają przypisane wartości. Wartość domyślna (w tym przypadku `tak`), jest oznaczona jako `checked`. PHP zwraca wybraną wartość w zmiennej o nazwie `$fish`; w tym przykładzie wyświetlimy ją za pomocą kolejnego kodu. Zwróć uwagę na zastosowane w nim sprawdzenie, czy zmienna `$fish` istnieje (za pomocą instrukcji `if(isset($fish))`); warto o tym pamiętać, tworząc strony z formularzami, które w celu wyświetlenia wyników lub błędów wskazują na siebie same. Na górze strony dopisz komentarz opisujący, co jest wyświetlane przy pierwszym wejściu, co w przypadku wystąpienia błędu oraz gdzie trafi odwiedzający po kliknięciu na przycisku *Wyślij*.

```
if(isset($fish))
   {
   print("Wartością \$fish jest " . $fish);
   }
```

W pytaniu możesz umieścić dowolną ilość odpowiedzi do wyboru, o ile mają te same nazwy. Gdy pominiesz pole `name`, niektóre przeglądarki przyjmą nazwę poprzedniego przycisku radiowego, ale nie sądzę, że działają tak wszystkie przeglądarki.

Gdy pominiesz pole `value`, PHP zwróci wartość `on`. Możesz nadać każdej możliwości inną nazwę i poszukać zmiennej o wartości `on`, ale w gruncie rzeczy jest to dość złożony sposób kodowania pytania z kilkoma odpowiedziami. Aby zilustrować tę trudność, następny fragment kodu zawierać będzie dwie linie z przykładu z pytaniem o ryby; pierwszy znacznik `<input>` ma nazwę `fishyes`, zaś drugi nazwano `fishno`, dzięki czemu możesz sprawdzić istnienie dowolnego z pól. To podejście wydaje się być najprostsze, gdy każde pytanie zawiera niewielką ilość odpowiedzi, ale staje się trudne do opanowania, gdy ilość możliwych odpowiedzi wzrasta.

```
<input type="radio" name="fishyes" value="tak" checked>
   Nie<input type="radio" name="fishno" value="nie">
```

Upewnij się, że każde pytanie ma inne pole `name` i że nazwa tego pola nie koliduje z żadną inną zmienną PHP. Najlepszą techniką może się okazać użycie numeru kolejności pytania, `$k` i budowanie nazwy z zastosowaniem wspólnego przedrostka (na przykład `question`), jak pokazano w kolejnym przykładzie kodu. Zmienne zwrócone z formularza będą miały wtedy nazwy `question0`, `question1`, `question2` i tak dalej.

```
$question = "<input type=\"radio\""
   . " name=\"question" . $k . "\""
   . " value=\"tak\" checked>";
```

Jedna lub więcej odpowiedzi z wielu

Niektóre pytania mogą mieć jedną lub kilka odpowiedzi; mogą także nie mieć żadnych odpowiedzi. Do ich zadawania służy znacznik HTML `<select>`. Znacznik ten pozwala na wskazanie jednej pozycji, ale ma także opcję zaznaczania kilku pozycji naraz (choć nie działa to w każdej przeglądarce ani w każdym systemie operacyjnym). Oto kilka przykładowych pytań oraz kod, rezultatem którego jest lista pokazana na rysunku 9.3.

Rysunek 9.3. *Wybierz swoje ulubione piwo, używając znacznika <select>*

- Jakie są Twoje ulubione owoce?
- Co powinieneś spakować, wybierając się do Sydney?
- Wybierz swoje ulubione piwo.

```
print("<form action=\"query.html\" method=\"post\">"
   . "<select name=\"favorite\"><option>Sparkling Ale</option>"
   . "<option>Pale Ale</option><option>Dark Ale</option>"
   . "<option>Stout</option><option>Premium Ale</option>"
   . "<option>Vintage Ale</option><option>Old Stout</option>"
   . "<option>Special Old Stout</option>"
   . "<option>Genuine Draught</option>"
   . "<option>Light</option><option>DB</option></select></form>");
```

Wyobraź sobie sytuację, w której klient nie może odpowiedzieć na pytanie. Podobają mi się firmy, które wymagają zarejestrowania produktu przed jego użyciem i w samym środku strony rejestracyjnej proszą o wpisanie komentarza na jego temat. W takich przypadkach najuczciwszą odpowiedzią będzie z pewnością:

```
Na razie ten produkt kosztuje mnie jedynie czas i jeszcze nie dał mi niczego użytecznego!
```

Z tą samą sytuacją masz do czynienia, gdy kupujesz wózek dla dziecka i musisz odpowiedzieć na pytanie, ile masz dzieci. Przecież możesz dopiero je planować lub masz po prostu zamiar rozpieszczać swojego psa.

Także brak odpowiedzi może być poprawną odpowiedzią. Pozwól klientom pomijać pytania, na które nie potrafią odpowiedzieć, gromadź inne odpowiedzi na dysku, a następnie zastanów się, jak zadać trudne pytanie następnym razem, gdy klient odwiedzi Twoją witrynę.

Gdy pobierasz wyniki z formularza, otrzymasz pewną ilość zmiennych ustawionych na odpowiedzi tak/nie lub otrzymasz tablicę zawierającą listę wybranych pozycji. Do Ciebie należy wybranie dokładnie tego, czego potrzebujesz dla swojego stylu programowania. Ja zwykle używam listy, aby móc łatwo dodawać nowe elementy.

Wyobraź sobie formularz z pytaniami:

- Czy lubisz banany?
- Czy lubisz pomarańcze?
- Czy lubisz jabłka?

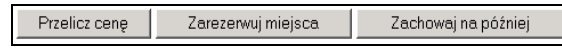
Lista pytań szybko staje się nużąca, dlatego powinieneś zmienić jej format tak, by pytanie „Czy lubisz” wystąpiło raczej jako nagłówek, a nie jako prosta lista owoców. W swoim kodzie PHP możesz mieć formularz zwracający zmienną `$bananas`, zawierającą wartość

`true` lub `false`, zmienną `$oranges` i tak dalej. Osobiście wolę stosować przechowywaną w tablicy listę zawierającą owoce, które zostaną umieszczone na liście wyboru, niż stosować prostą listę z wybranymi owocami, zawartą w innej tablicy. Kolejna pozycja lub kolejny owoc mogą zostać dodane przez kogokolwiek i nie muszę w tym celu zmieniać nawet linijki kodu.

W podrozdziale „Natychmiastowe rozwiązania" w dalszej części tego rozdziału znajdziesz kod pozwalający Twoim klientom na wybór jednej lub więcej pozycji listy. Przejrzyj przykład, aby poznać rodzaj pytania, którego potrzebujesz, a następnie wypróbuj działanie kodu.

Kilka akcji

Formularz kończy się przyciskiem wysyłania oraz przyciskiem zerowania lub anulowania, dzięki czemu klient może wybrać: „Dalej, kupuję to" lub „Rezygnuję z zamówienia". Nie jesteś ograniczony wyłącznie do dwóch możliwości; możesz przygotować ich więcej, jak pokazano na rysunku 9.4.



Rysunek 9.4. *Więcej przycisków akcji w formularzu*

W swoim kodzie PHP możesz łatwo obsłużyć więcej przycisków akcji, ponieważ z każdego z nich otrzymujesz inną zmienną. Jeśli przycisk *Przelicz cenę* miałby nazwę `recalc`, a pozostałe odpowiednio `book` i `save`, skrypt otrzymałby zmienne `$recalc`, `$book` oraz `$save` i do podjęcia odpowiednich działań mógłbyś użyć kodu w rodzaju `if(isset($recalc))`.

Pierwszym znacznikiem HTML w formularzu jest znacznik `<form>`, który zawiera parametr `action="_nazwa_strony_"`, gdzie `_nazwa_strony_` jest nazwą strony, której żądanie zostanie wysłane przez przeglądarkę w momencie, gdy użytkownik kliknie na przycisku wysyłania lub zerowania. Przeglądarka decyduje, czego zażądać, po czym wysyła to żądanie do serwera. Umieszczając nazwę strony w parametrze akcji decydujesz, której strony (czyli którego skryptu) zażąda przeglądarka.

Czy chcesz odesłać użytkownika na tę samą stronę, czy na inną? Gdy wyślesz go na tę samą stronę i przeprowadzisz przetwarzanie w jej nagłówku, możesz z łatwością wyświetlić ponownie tę samą stronę i ten sam formularz, na przykład wraz z komunikatami o błędach i żądaniami uzupełnienia informacji. Gdy skierujesz użytkownika na inną stronę, masz inne możliwości, jak przetworzenie wszystkich danych z formularza na ukrytej stronie, która niczego nie wyświetla, a następnie wysłanie nagłówków przekierowania, odsyłających przeglądarkę użytkownika do właściwej strony.

Czasem możesz chcieć odsyłać klientów do więcej niż jednej strony; możesz to zrobić, umieszczając na pojedynczej stronie więcej niż jeden formularz. Wyobraź sobie stronę zawierającą wyłącznie trzy przyciski z rysunku 9.4; każdy z nich mógłby być przyciskiem wysyłania zupełnie innego formularza, z zupełnie innym zestawem pytań i akcji. Przycisk *Zarezerwuj miejsca* mógłby wskazywać stronę na bezpiecznym serwerze w banku, przycisk *Zachowaj na później* mógłby wskazywać stronę aktualizującą profil użytkownika, zaś przycisk *Przelicz cenę* mógłby po prostu powodować skok do bieżącej strony z przeliczeniem nowej ceny.

Jedyny problem z użyciem kilku formularzy wiąże się z brakiem koordynacji danych. Gdy użytkownik kliknie na przycisku pierwszego formularza na stronie, nie będziesz mógł zebrać danych z drugiego formularza. Aby użyć kilku formularzy, musisz najpierw zebrać wszystkie dane, a następnie użyć osobnych formularzy wyboru na innej stronie.

JavaScript

Często zdarza się, że niedoświadczeni programiści, używający JavaScriptu po raz pierwszy, tworzą w nim rozwijane listy wyboru i przeprowadzają skomplikowaną weryfikację danych. Z drugiej strony, właściciele witryn płacą mi duże pieniądze za usunięcie podobnego kodu JavaScript. Dlaczego? Muszę poprawiać formularze, ponieważ nie działają w niektórych przeglądarkach, a jeśli nawet wydają się działać, okazuje się, że baza danych witryny pełna jest niepoprawnych danych. Do moich zadań należy zarówno sprawienie, by formularze działały w większej ilości przeglądarek, jak i powstrzymanie bezużytecznych informacji przed trafieniem do bazy danych. Aby to osiągnąć, wykonuję pewne kroki.

Usuwanie wymyślnych rozwijanych menu

Pierwszy krok wiąże się z usunięciem wymyślnych rozwijanych menu, stworzonych w JavaScripcie. Standardowy znacznik HTML `<select>` dostarcza rozwijanego menu, które nie tylko jest łatwe do stworzenia w PHP, ale jest także zrozumiałe dla każdego. Cokolwiek bardziej skomplikowanego wprawia niektórych użytkowników w zakłopotanie i może spowodować, że albo otrzymasz dane wpisane w niewłaściwe pole, albo nie otrzymasz tych danych wcale. Powszechnym symptomem jest strona w połowie pusta, ponieważ kliknięcie na kontrolowanym przez JavaScript polu w środku strony spowodowało zniszczenie serii innych, nie powiązanych z nim pól.

Sprawdzanie poprawności pól na serwerze

Drugi krok wymaga, by na serwerze sprawdzane były wszystkie pola. Bez względu na to, jak dobrym uczynisz swój kod JavaScript, klienci mogą obejść ten kod i wysłać Ci niepoprawne dane. Jedna ze sztuczek polega na skopiowaniu strony, zmianie kodu JavaScriptu w kopii, a następnie otwarciu kopii w przeglądarce. Wiele witryn używa niekompletnego i źle sformatowanego JavaScriptu, działającego wyłącznie w Internet Explorerze (powodującego błędy w przeglądarkach takich jak Opera), pozwalając użytkownikom przeglądarek na decydowanie, czy skrypt ma kontynuować działanie, czy też ma się zatrzymać. Możesz otrzymać dane, które tylko częściowo zostały zweryfikowane przez Twój skrypt lub które zostały w całości stworzone przez cudzy skrypt, dlatego wszystko (a już na pewno ceny i podsumowania) musisz sprawdzać w swoim kodzie PHP. Gdy sprawdzasz pole na serwerze za pomocą PHP, nie zaoszczędzisz zbyt wiele czasu, sprawdzając je w przeglądarce za pomocą JavaScriptu.

Gdy próbujesz zmienić JavaScript lub koszyk z zakupami korzystający z cookie, zmiana łącznej kwoty z pięciu tysięcy dolarów na tysiąc dolarów może się udać, jednak zmianę na dziesięć dolarów z pewnością ktoś zauważy.

W którymś z Twoich projektów może się zdarzyć, że ktoś będzie nalegać na użycie JavaScriptu do odrzucania stron, w których wymagane pola nie zostały wypełnione. Gdy

uniemożliwisz potencjalnym klientom wysłanie formularza, mogą opuścić Twoją witrynę i nigdy już do niej nie wrócić. Przesyłając formularz do serwera i sprawdzając go tam, możesz zarejestrować pola, na które nie otrzymujesz odpowiedzi i przekazać te informacje projektantom formularza.

Długie formularze

Gdy klient nie ma potrzeby wypełniania wszystkich pól, wielostronicowe formularze mogą go irytować. W papierowych formularzach można trafić na sekcję opisaną jako przeznaczona wyłącznie dla diabetyków, którą można pominąć lub wypełnić, zależnie od potrzeb. Internetowym odpowiednikiem będzie zadanie na pierwszej stronie ogólnego pytania (Czy jesteś diabetykiem?), a następnie wyświetlenie lub pominięcie strony z pytaniami przeznaczonymi dla diabetyków.

Problem wyborów prezydenckich na Florydzie w 2000 roku pokazał, jak łatwo wprowadzić w zakłopotanie użytkowników formularzy. W komputerze problem jest jeszcze większy, gdyż ludziom trudno jest przewracać strony w przód i w tył i porównywać sekcje formularza. W związku z tym musisz stosować jednolity układ stron, zapewniający wizualne wskazówki dotyczące każdego pytania i rodzaju wymaganej odpowiedzi. W następnych podrozdziałach poznasz zasady tworzenia poprawnych wielostronicowych formularzy.

Dobrym rozwiązaniem dla długich formularzy jest stosowanie jednej funkcji formatującej dla wszystkich pytań każdego formularza znajdującego się na witrynie. Gdy nowy klient wypełni jedną stronę formularza, będzie gotów do wypełniania wszystkich innych formularzy. Oddzielając formatowanie od pytań, możesz łatwej zmieniać kolejność pytań i przerwy pomiędzy stronami.

Podział długich formularzy

Wiele handlowych witryn oferuje przykłady zbyt długich formularzy, stron rejestracyjnych czy ankiet, które zniechęcają potencjalnych klientów. Długie formularze powinny zostać podzielone na grupy powiązanych ze sobą pytań; każda grupa powinna wiązać się z jednym zagadnieniem i powinna znajdować się na osobnej stronie.

Wyobraźmy sobie witrynę, za pośrednictwem której sprzedawana jest żywność; załóżmy, że autorzy tej witryny chcą, by klient przyjmował cotygodniowe wiadomości e-mail oraz chcieliby poznać wiek, dochód i płeć klienta. Niektórzy klienci wypełnią taki formularz, ale wielu po prostu opuści witrynę, zaś wielu innych zrobi to, co ja: wybierze zupełnie nieprawdziwe odpowiedzi na zaczepne i irytujące pytania. W przypadku pytania o pocztę elektroniczną zawsze klikam na przycisku *Nie*, ponieważ zbyt wiele witryn wysyła bezużyteczne, zaśmiecające skrzynkę listy; jako wiek wybieram najniższą lub najwyższą możliwość, zaś jako dochód wskazuję kwotę, jakiej nigdy nie zarobiłby autor fachowej literatury. Od czasu do czasu w pytaniach o płeć mam okazję wybrać pozycję *Inna*.

A gdybyś tak pytanie o adres e-mail umieścił na osobnej stronie, wraz ze wzorem wiadomości? Wiadomość ta mogłaby zawierać interesujące informacje i czytelnik mógłby wybrać twierdzącą odpowiedź na pytanie: „Czy chciałbyś otrzymywać wiadomości podobne do tej?”.

Zamiast bezpośrednio pytać o wiek osoby, odeślij ją na stronę wyświetlającą oferty uzależnione od wieku klienta i dopiero wtedy zadaj pytanie o wiek oraz inne osobiste pytania. Możesz na przykład umieścić zdjęcia uśmiechniętych dzieci na stronie dotyczącej żywności dla dzieci, a obok umieścić zapis: „Aby pomóc nam w dobraniu produktów dla Twojego dziecka, podaj proszę...” i zapytać w nim o wiek wszystkich członków rodziny.

Na każdej stronie spróbuj przedstawiać temat, krótką zachętę do odpowiedzi na pytanie, samo pytanie oraz dodatkowy materiał (nie ma znaczenia, czy ów dodatkowy materiał wykracza poza dolną krawędź strony). Lepiej jest uzyskać poprawną odpowiedź na jedno pytanie, niż otrzymać niepoprawne odpowiedzi na wiele pytań.

Przenoszenie informacji ze strony do strony

Gdy korzystasz z wielostronicowych formularzy, możesz przenosić informacje ze strony do strony za pomocą ukrytych pól, cookies lub rekordów sesji. Typową strategią jest zapytanie o imię osoby na jednej stronie, a następnie wyświetlanie tego imienia na kolejnych stronach, aby stały się one bardziej osobiste. Ukryte pole jest najprostszym sposobem przenoszenia imienia, ale najlepszym jest rekord sesji (o ile istnieje sesja), gdyż dzięki niemu imię jest dostępne na każdej stronie witryny.

Użycie ukrytych pól

Ukryte pola, `<input type="hidden">` są łatwe do użycia dla takich obiektów, jak długie listy; nie działają jednak wtedy, gdy osoba wypełniająca formularz zdecyduje się na przerwanie tego zajęcia, po czym spróbuje dokończyć wypełnianie później. W takich przypadkach mogą pomóc cookies, ale także one posiadają ograniczenia.

Ustawianie cookies dla każdej ze stron

Gdy prosisz użytkownika o wypełnienie wielostronicowego formularza, wartości wpisane na pierwszej stronie mogą zostać umieszczone bezpośrednio w bazie danych lub przeniesione na drugą stronę w ukrytych polach. Jeśli aktualizujesz profil klienta, możesz natychmiast wprowadzać do bazy danych każdą stronę informacji, zaś ze strony do strony nie będą przenoszone żadne dane. Dla odróżnienia, gdy zbierasz informacje w celu stworzenia nowego profilu, możesz nie chcieć zapisywać w bazie danych niczego, aż do chwili, gdy użytkownik kliknie na ostatniej stronie na przycisku zatwierdzenia warunków.

W przypadku nowego profilu możesz przenosić zmienne ze strony na stronę w ukrytych polach, ale dodanie nowych stron w środku wielostronicowego formularza oznacza dodanie nowych pól na wszystkich następnych stronach, co może być równie nużące, jak podatne na błędy. Łatwiejsze w użyciu mogą być pliki cookies, ponieważ możesz umieścić w nich wszystko, tworząc nowy plik cookie dla każdej ze stron. Tylko ostatni skrypt w sekwencji musi przejrzeć każde cookie. Mają one mają także tę zaletę, że pozostają w komputerze klienta przez jakiś czas i pozwalają mu na kontynuowanie wypełniania formularza następnego dnia.

Problem z cookie uwidacznia się dopiero wtedy, gdy zbudujesz już kilka stron wielostronicowego formularza. Pojawiają się wtedy ograniczenia plików cookie, wobec czego

musisz ponownie przemyśleć cały swój projekt. Każda przeglądarka nakłada inne ograniczenia na cookie, dlatego jedynymi wytycznymi będą podane tutaj informacje. Specyfikacja cookie określa, że możesz mieć tylko dwadzieścia pików cookie na domenę, co uniemożliwia stworzenie 21-stronicowego formularza z jednym cookie na każdą ze stron. Maksymalny rozmiar cookie wynosi 4 kB, dlatego nie można gromadzić informacji z długiego formularza w jednym pliku.

Problemy pojawią się także, gdy klienci mają wyłączoną obsługę cookie, mają tak ustawioną przeglądarkę, by usuwała cookie na końcu sesji lub gdy korzystają z oprogramowania, które każdego dnia usuwa cookie z dysku. Nie zadziałają one także wtedy, gdy klient rozpocznie wypełnianie formularza przy komputerze w pracy, po czym spróbuje dokończyć je przy komputerze domowym. Z powodu tych ograniczeń pliki cookie powinny być używane wyłącznie do przechowywania identyfikatorów sesji, tak, aby można było utrzymać aktywne sesje.

Sesje zostaną omówione w rozdziale 19., tu zaś pokażemy jedynie opcje konfiguracji, jakie powinieneś włączyć (dla PHP 4.0.5). W pliku *php.ini* włącz opcję `use_cookies`, aby korzystać z cookie (o ile są dostępne), włącz także opcję `use_trans_sid`, aby automatycznie umieszczała identyfikator sesji w łańcuchach URL, gdy cookie nie są dostępne, na koniec zaś ustaw opcję `url_rewriter.tags`, wskazując listę parametrów znaczników HTML, które mogą przenosić łańcuchy URL. To wystarczy do uruchomienia testowej strony:

```
session.use_cookies = 1
session.use_trans_sid = 1
url_rewriter.tags =
  "a=href,area=href,frame=src,input=src,form=fakeentry"
```

Użycie rekordów sesji

Rekordy sesji mogą mieć nieograniczony rozmiar i ilość pozycji. Są przechowywane w polach tekstowych nowoczesnych baz danych, które pozwalają na przechowanie do czterech miliardów znaków, co zdecydowanie wystarcza dla ogromnej większości zastosowań. Jeśli rekord sesji jest przechowywany w osobnym pliku w nowoczesnym systemie operacyjnym, może mieć rozmiar nawet wielu terabajtów. Jeden taki rekord sesji mógłby przechować cały tekst, pochodzący ze wszystkich formularzy wypełnianych kiedykolwiek przez wszystkich ludzi na tej planecie. Użycie cookie do przechowania identyfikatora sesji oraz rekordu sesji do przechowania pozostałych informacji stanowi doskonałe rozwiązanie dla komunikacji między stronami w obrębie witryny WWW.

Istnieje jednak problem dotyczący rekordów sesji, polegający na ich tymczasowej naturze. Rekordy sesji znikają w momencie zamknięcia przeglądarki, stają się przestarzałe, gdy wyjdziesz na lunch i są zależne od innych rzeczy, jak cookie lub URL, przenoszących identyfikator sesji ze strony do strony. Nie można bez pewnych trudności przywrócić połączenia z sesją, zaś prosty sposób stanowiłby lukę w zabezpieczeniach, gdyż umożliwiałby łatwe włamanie się do sesji. W związku z tym użytkownik nie może zapamiętać strony w środku sesji, a następnie zacząć od niej pracę następnego dnia.

Aby umożliwić użytkownikom połączenie się następnego dnia z tą samą sesją, potrzebujesz systemu logowania i wylogowywania się, z zastosowaniem czegoś w rodzaju identyfikatora użytkownika i hasła. Musisz zapisać wszystkie informacje sesji w profilu

użytkownika i następnego dnia stworzyć nową sesję, wypełnioną wszystkimi szczegółami profilu. Teoretycznie mógłbyś użyć ponownie tego samego identyfikatora sesji, ale w ten sposób umożliwiłbyś mi włamanie się do sesji.

Oto jak włamać się do sesji na słabo zaprojektowanych witrynach: gdy Twój współpracownik pójdzie do domu, przejrzyj jego plik cookie (*cookies.txt* w Netscape), aby znaleźć nazwę witryny oraz powiązany z nią identyfikator sesji. Wyłącz cookie w jego przeglądarce, aby wymusić użycie URL, zaloguj się do tej samej witryny, używając dowolnej innej nazwy logowania, po czym wytnij identyfikator sesji swojego współpracownika i wklej go do swojego łańcucha URL. Rekord sesji pozostaje w pliku w serwerze od 20 minut do kilku godzin, zatem zwykle masz do dyspozycji mnóstwo czasu.

Projektowanie dobrych długich formularzy

Mój zakres zapamiętywania obejmuje dwie strony. Innymi słowy, mogę zapamiętać, co wpisałem na pierwszej stronie, gdy wpisuję drugą, ale nie pamiętam już tego podczas wpisywania trzeciej. Próbuję używać tego zakresu jako odniesienia i uwzględniam go, projektując formularze. Gdy mam przygotować profil użytkownika lub podobny formularz, staram się zaprojektować go w taki sposób, by na pierwszej stronie uzyskać informacje identyfikujące użytkownika; informacje te wykorzystuję jako podstawę dla następnych stron.

Najpierw uzyskaj minimum

Jeśli nie możesz potraktować odwiedzającego jako nowego klienta zanim wypełni obszerny formularz, stracisz potencjalnych klientów. Radzę na pierwszej stronie zapytać jedynie o nazwę i adres email użytkownika, po czym wyświetlić informacje o logowaniu. Pytania marketingowe pozostaw na później.

Zaprojektuj formularz i obsługujący go kod w taki sposób, aby móc zapisać podstawowy adres email, zanim zirytujesz odwiedzającego natarczywym pytaniem lub znudzisz go zbyt wieloma stronami zawierającymi pytania. Zaprojektuj witrynę tak, by użytkownik mógł opuścić w połowie proces rejestracji i wrócić do niego później, ale już z własnym identyfikatorem.

Niech marketing płaci za swoją natarczywość

Jeśli kierownik marketingu chce znać wiek klienta, niech oferuje za to jakąś nagrodę. Na stronie rejestracyjnej zbierz minimalne informacje o kontakcie, podaj informacje o sposobie logowania się do witryny, zaś następne pytania zadawaj tylko wtedy, gdyż użytkownik się zaloguje. Nie mam pojęcia, dlaczego autor witryny, za pośrednictwem której sprzedaje się sprzęt biurowy pyta mnie, czy jestem mężczyzną, czy kobietą — czyżby mężczyznom sprzedawano tylko niebieskie długopisy, a kobietom tylko czerwone?

Jeśli na Twojej witrynie nie sprzedajesz alkoholu, papierosów czy pornografii (lub czegokolwiek innego, na co Twój rząd nakłada ograniczenia wiekowe), nie musisz pytać klienta o wiek, aby pozwolić mu na dokonywanie zakupów. Pytanie o wiek może być pytaniem dodatkowym, któremu może towarzyszyć jakaś zachęta, na przykład darmowy produkt przy następnej dostawie. W ten sposób uzyskasz zarówno wiek klienta, jak i zachęcisz go do zakupienia czegoś w celu zdobycia darmowego produktu — a zachęcenie do zakupów jest zawsze pożądane.

To samo odnosi się do pytań o dochód i płeć. Pozwól użytkownikom zarejestrować się nawet wtedy, gdy nie odpowiedzą na te pytania i daj im jakąś zachętę, aby na nie odpowiedzieli. Aby uzyskać prawdziwe odpowiedzi, upewnij się, że zachęta nie skłania do określonego wyboru. Jeśli zachętą dla młodszych osób byłoby DVD z filmem „XFiles", zaś dla starszych DVD z filmem „Moulin Rouge", podałbym wiek 95 lat!

Stosuj informacje zwrotne

Gdy masz już coś, czego możesz użyć do zidentyfikowania osoby wypełniającej formularz, umieść tę informację w pobliżu początku strony, przypominając użytkownikowi, że jest zalogowany. Pobierając każdą z sekcji formularza wyświetlaj informacje zwrotne, takie jak ekrany podsumowujące, stosowane na witrynie Amazon, przedstawiające kroki już wykonane oraz pozostałe do wykonania. Powszechnym problemem w przypadku źle zaprojektowanych formularzy są klienci, którzy rejestrują się po kilka razy, ponieważ albo zapomnieli, że już się zarejestrowali, albo uważali, że ich rejestracja się nie powiodła.

Możesz z łatwością przechować tablicę zawierającą nazwę użytkownika i wypełnione części formularza, a następnie umieścić te informacje w rogu strony. Czasem cały formularz opieram na tablicy podobnej do następującej:

```
$elements[] = array("name" => "firstname", "length" => 60,
   "question" => "Imię", "complete" => false);
$elements[] = array("name" => "lastname", "length" => 60,
   "question" => "Nazwisko", "complete" => false);
$elements[] = array("name" => "address", "length" => 60,
   "question" => "Adres", "complete" => false);
```

Mogę przetwarzać elementy tablicy tworząc formularz, ustawić limit dziesięciu pytań na stronę i zaznaczać każde pytanie jako wykonane, gdy uzyskam na nie odpowiedź. Gdy w polu wystąpi błąd, mogę dodać element błędu do danej pozycji tablicy. Na stronie obsługującej formularz mogę jeden raz zarejestrować tablicę jako zmienną sesji, po czym, na końcu przetwarzania, mogę zapisać całą tablicę do bazy danych. Do innych ulepszeń, jakie można zastosować, należą stale pokazywane elementy, jak choćby wyświetlana na każdej ze stron nazwa użytkownika.

Różnica pomiędzy nie zadaniem pytania a brakiem odpowiedzi

Gdy dane znajdują się już w bazie danych, możesz skorzystać z jej mechanizmów do oznaczenia pól pustych, nieznanych lub pól o wartości `null`. W MySQL pole łańcucha może zwracać łańcuch pusty lub wartość `null`, w zależności od ustawień w definicji pola (szczegóły znajdziesz w rozdziale 5.). Możesz wykorzystać to do pytań w rodzaju „podaj swoje hobby", konwertując `null` w MySQL na `false` w PHP, wskazujące, że pytanie o hobby nie zostało zadane i powinno zostać wyświetlone następnym razem, gdy klient odczytuje swój profil. Oprócz tego pusty łańcuch MySQL możesz zamienić na łańcuch `""` w PHP, wskazujący, że pytanie o hobby zostało zadane, ale użytkownik jeszcze nie odpowiedział na nie. Jeśli odróżnienie wartości `null` i `false` od łańcucha pustego jest zbyt trudne (jest tak w przypadku niektórych baz danych i PHP3), stwórz osobne pole określające, czy pytanie zostało zadane. Możesz nawet użyć daty, wskazującej, kiedy pytanie zostało zadane i zadawać je dokładnie co rok.

Wysyłanie plików do serwera

Formularze pozwalają także na wysyłanie plików ze stacji roboczej do serwera WWW. Istnieje kilka niewielkich różnic pomiędzy formularzami wysyłania plików a formularzami standardowymi. Dodatkowo obie formy formularzy działają inaczej i wyświetlają różne rzeczy w różnych przeglądarkach i systemach operacyjnych. PHP ułatwia przetwarzanie po stronie serwera, ale nie pomaga w wyjaśnieniu użytkownikom, jak mają korzystać z możliwości wysyłania plików w ich przeglądarkach i systemach operacyjnych.

Pierwszą rzeczą, potrzebną do wysyłania plików, jest znacznik formularza z ustawionym parametrem `enctype`. Skopiuj przedstawiony tu przykład lub chwyć za swój egzemplarz „HTML Black Book" (The Coriolis Group, Inc., 2000) aby poznać informacje na temat `enctype`. Znacznik formularza wygląda następująco:

```
<form enctype="multipart/form-data"
action="apage.html" method="post">

<input type="hidden" name="MAX_FILE_SIZE" value="200000">
```

Będziesz potrzebował pola tekstowego dla nazwy pliku; HTML oferuje specjalne pole nazwy pliku z przyciskiem umożliwiającym przeglądanie plików — pole typu `file`. W tym znaczniku parametr `size` jest rozmiarem pola nazwy pliku, a nie wielkością pliku. Sam znacznik wygląda następująco:

```
<input type="file" name="uploadfile" size="60">
```

Formularz kończy się standardowym znacznikiem wysyłania i znacznikiem końca formularza:

```
<input type="submit" value="Wyślij"></form>
```

PHP posiada plik konfiguracyjny o nazwie *php.ini*, zawierający kilka parametrów ograniczających pliki, które można wysyłać za pomocą formularzy. Musisz włączyć opcję `file_uploads`, ustawić opcję `upload_tmp_dir` na odpowiedni katalog, zaś w opcji `upload_max_filesize` ustawić co najmniej taką wartość, jaką wyrażony jest rozmiar największego pliku, który chcesz wysyłać. Niektóre operacje na plikach są ograniczone przez ilość pamięci przydzielonej PHP, dlatego ustaw wartość `memory_limit` na wartość większą niż `upload_max_filesize`. W tym celu możesz wpisać:

```
file_uploads = On
upload_tmp_dir = T:\upload
upload_max_filesize = 8M
memory_limit = 30M
```

Opcja `file_uploads` pozwala PHP na obsługę wysyłanych plików. Niektórzy uważają, że stanowi to obniżenie poziomu zabezpieczeń, zaś inni za nieakceptowalne zajmowanie zasobów. Gdy się na to zdecydujesz, powinieneś ograniczyć możliwość wysyłania plików do osób, które się w jakiś sposób zalogują, aby móc się z nimi skontaktować w przypadku niewłaściwego zachowania. Wyobraź sobie, że Twój serwer WWW zostaje zalany plikami zawierającymi zdjęcia pornograficzne lub pirackimi piosenkami Nirvany.

Gdy ktoś wysyła plik, trafia on do tymczasowego katalogu wskazanego w opcji `upload_tmp_dir`, dzięki czemu możesz sprawdzić jego zawartość przed przesłaniem go do

katalogu docelowego. Upewnij się, że w tymczasowym katalogu jest dużo miejsca dla równolegle otrzymywanych plików i że jest on okresowo porządkowany, aby serwer nie zatonął w starych plikach.

Wartość `upload_max_filesize` jest próbą ograniczenia wysyłanych plików przez ograniczenie maksymalnego rozmiaru pojedynczego pliku; zdarza się jednak, że użytkownik może przesłać wiele gigabajtów pirackich filmów w częściach po kilka tysięcy bajtów każda, po czym złożyć je automatycznie w całość za pomocą ogólnodostępnego oprogramowania. Ograniczenie rozmiaru pliku nie zabezpieczy Cię przed niepowołanym wykorzystaniem Twojego serwera.

Wartość `memory_limit` odnosi się do PHP i ogranicza dostępną dla niego pamięć. Gdy zwiększysz limit pamięci PHP, możesz ograniczyć możliwość działania serwera WWW lub systemu operacyjnego, dlatego sprawdź ich wymagania co do ilości pamięci. Jeśli Twój skrypt generuje obszerne zmienne lub tablice, zwiększ wartość `memory_limit` do co najmniej dwukrotnej ilości miejsca, jakiej potrzebujesz. Największym obrazkiem w mojej stacji roboczej jest 95-megabajtowe zdjęcie krajobrazu. Oprócz tego posiadam napisane w PHP oprogramowanie do zarządzania obrazkami, odczytujące pary obrazków w celu ich porównania. Ten zajmujący 0,15 MB skrypt wymaga 191 MB do porównania dwóch z moich krajobrazów.

Gdy plik trafia do serwera WWW, jego nazwa jest zawarta w zmiennej o nazwie odpowiadającej parametrowi `name` w znaczniku `<input type="file">`. W pokazanym tu przykładzie parametr `name` zawiera nazwę `uploadfile`, dlatego nazwa otrzymanego pliku będzie zawarta w zmiennej `$uploadfile`. Zmienna ta zawiera pełną ścieżkę i nazwę pliku, potrzebne funkcji `copy()` i innym funkcjom przetwarzającym pliki. Zmienna `$uploadfile_name` będzie zawierać nazwę pliku dostarczonego przeglądarce, zmienna `$uploadfile_size` — rozmiar pliku, zaś zmienna `$uploadfile_type` — typ MIME pliku (o ile przeglądarka dostarczy takich informacji).

Plik trafia na serwerze do tymczasowego katalogu, dlatego, jeśli chcesz go zachować, musisz skopiować go w bezpieczne miejsce. PHP w wersji 4.0.4 i nowszych oferuje w tym celu specjalną funkcję `move()`. W starszych wersjach PHP możesz przeprowadzić kopiowanie, a następnie usuwanie pierwotnego pliku.

Gdy dojdziesz do tego miejsca i wciąż masz kłopoty z wysyłaniem plików, zwykle są one związane z uprawnieniami systemu operacyjnego lub przeglądarki albo z problemami z typem pliku. Jeśli masz problemy z typem pliku, wypisz wartość zmiennej `_type` aby sprawdzić, co przeglądarka uważa za typ pliku.

Błędy dostępu występują wtedy, gdy system operacyjny serwera WWW ogranicza Twój dostęp do katalogu otrzymującego wysłane pliki, co zmusza Cię do błagania i przekupywania administratora systemu, aby zmienił uprawnienia. Spójrz na opcję `upload_tmp_dir`, a następnie poeksperymentuj z ręcznym kopiowaniem z katalogu `tmp` do katalogu docelowego.

Jeśli wszystko inne zawiedzie i nie możesz określić, co PHP robi z plikiem, uruchom stronę zawierającą przedstawiony tu kod PHP, aby poznać każdy parametr serwera Apache z punktu widzenia tego serwera oraz każdą opcję PHP z punktu widzenia PHP:

```
<?php phpinfo(); ?>
```

Natychmiastowe rozwiązania

Tworzenie formularza

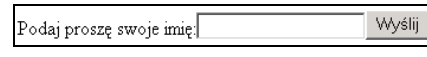
Oto podstawowy formularz w HTML, pytający o imię, po czym wracający do strony o nazwie *createaform.html*:

```
<form action="createaform.html" method="post">
Podaj proszę swoje imię:
<input type="text" name="name">
<input type="submit" name="submit" value="Wyślij">
</form>
```

A oto podstawowy kod PHP, użyty do stworzenia tego formularza:

```
print("<form action=\"createaform.html\" method=\"post\">"
   . "Podaj proszę swoje imię:"
   . "<input type=\"text\" name=\"name\">"
   . "<input type=\"submit\" name=\"submit\" value=\"Wyślij\">"
   . "</form>");
```

Formularz wyświetlony w przeglądarce został pokazany na rysunku 9.5.



Rysunek 9.5. *Prosty formularz z jednym pytaniem i przyciskiem Wyślij*

Na stronie *createaform.html* dane wpisane przez użytkownika zostaną umieszczone w zmiennej o nazwie `$name`. PHP pobiera wszystkie pola formularza (za pomocą metody `POST` lub `GET`) i umieszcza je w zmiennych o nazwach odpowiadających parametrom `name` pól formularza w znacznikach `<input>` w HTML. Oto kod odwołujący się do danych z formularza na stronie *createaform.html*:

```
if(isset($name)) {print("Masz na imię: " . $name);}
```

Tworzenie formularza za pomocą funkcji

W przypadku większych formularzy pokazany w podrozdziale „Tworzenie formularza" kod trudno byłoby w razie potrzeby zmodyfikować, dlatego dobrym rozwiązaniem może być podzielenie go na kilka mniejszych funkcji oraz oddzielenie pytań od formularza. Oto podstawowy kod PHP do budowania znaczników formularza i wstawiania do niego pytań w postaci łańcucha znaków:

```
function form($page, $text)
   {
   return("<form action=\"" . $page . "\" method=\"post\">"
      . $text
      . "<input type=\"submit\" name=\"submit\""
      . " value=\"Wyślij\">"
      . "</form>");
   }
```

Funkcja `form()` przyjmuje jako pierwszy parametr docelową stronę, zaś jako drugi — wszystkie pytania, jakie mają znaleźć się w formularzu. Domyślnie zastosowano metodę `POST` formularza, ponieważ jest ona najlepszym sposobem wysyłania danych formularza. Każdy formularz wymaga przycisku wysyłania, czyli znacznika `<input>` z rodzajem akcji, dlatego do funkcji `form()` wstawiłem trwale taki znacznik. Możesz do niego dodać także przycisk zerowania.

Następny kod tworzy pytanie na podstawie otrzymanych danych tekstowych. Funkcja wymaga podania nazwy pytania, jego treści oraz opcjonalnej domyślnej odpowiedzi:

```
function form_text($name, $question, $default="")
   {
   if(strlen($default))
      {
      $default = " value=\"" . $default . "\"";
      }
   return($question . "  "
      . "<input type=\"text\" name=\"" . $name . "\""
      . $default . ">");
   }
```

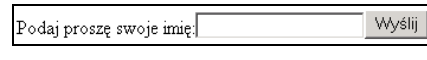
Znaczniki `<input>` mogą zawierać wartości domyślne, zachęcające użytkownika do wypełnienia formularza. W funkcji `form_text()` linia `if(strlen($default))` sprawdza, czy parametr zawiera tekst, zanim umieści go w parametrze `value` znacznika HTML.

Instrukcja `return()` w funkcji `form_text()` zwraca pytanie, kilka spacji oraz znacznik `<input>` z odpowiednimi parametrami. Wymagany jest parametr `$question`, ponieważ jest on jedynym sposobem poinformowania klienta o tym, czego chcesz się dowiedzieć. Obowiązkowe jest także podanie parametru `$name`, ponieważ nie ma innego sposobu zidentyfikowania odpowiedzi zwracanej do skryptu.

W tym momencie możesz tworzyć formularze za pomocą łatwego do rozbudowy kodu, który może być tak ułożony, by w każdej linii zawierał jedno pytanie. Nasz kolejny kod tworzy za pomocą funkcji `form_text()` jedno pytanie, które stanowi parametr wywołania funkcji `form()`, tworzącej cały formularz:

```
print(form ("form.html",
   form_text("name", "Podaj proszę swoje imię:")
   ));
```

W wyniku otrzymujemy formularz pokazany na rysunku 9.6.



Rysunek 9.6. *Formularz z jednym pytaniem*

Tworzenie długiej listy w formularzu

W przypadku dłuższej listy pytań formularz stworzony w podrozdziale „Tworzenie formularza” staje się trudny w użyciu. W związku z tym w kolejnym przykładzie użyjemy tablicy pytań. Najpierw zdefiniujmy tablicę zawierającą pytania ułożone w takiej kolejności, w jakiej mają pojawić się w formularzu:

```
$question["name"] = array("type" => "text",
   "question" => "Podaj proszę swoje imię:");
$question["address"] = array("type" => "text",
   "question" => "Adres:");
$question["city"] = array("type" => "text",
   "question" => "Miasto:");
$question["state"] = array("type" => "text",
   "question" => "Stan:");
$question["country"] = array("type" => "text",
   "question" => "Kraj:");
```

Tablica zawiera pięć elementów, z których każdy jest tablicą zawierającą elementy `name`, `type` oraz `question`; pole `name` jest używane jako klucz tablicy. Pole to identyfikuje zmienną zawierającą odpowiedź na pytanie i jako takie powinno zawierać nazwę zgodną ze standardami nazw zmiennych w Twoim skrypcie PHP. Pole `type` informuje funkcję `form()`, by stworzyła znacznik `<input>` za pomocą funkcji `form_text()`. Pole `question` zawiera wypisywany na stronie tekst dla znacznika `<input>`; tekst ten może zawierać znaczniki formatujące HTML, ponieważ jest zwracany bez żadnych modyfikacji.

Następny fragment kodu tworzy znacznik pola tekstowego. Funkcja ta jest zapożyczona z podrozdziału „Tworzenie formularzy" i została poddana kilku modyfikacjom. Używa nazwy do zidentyfikowania odpowiedzi na pytanie, tekstu pytania oraz opcjonalnej domyślnej odpowiedzi.

```
function form_text($name, $parameters)
   {
   if(!isset($parameters["default"]))
      {
      $parameters["default"] = "";
      }
   if(strlen($parameters["default"]))
      {
      $parameters["default"] = " value=\"" . $parameters["$default"]
         . "\"";
      }
   return($parameters["question"] . "  "
      . "<input type=\"text\" name=\"" . $name . "\""
      . $parameters["default"] . ">");
   }
```

Ta funkcja przyjmuje nazwę pola jako parametr `$name`, zaś pozostałe parametry przyjmuje w tablicy `$parameters`. Jeśli w tablicy `$parameters` nie istnieje element `$parameters["default"]`, funkcja tworzy go. Kod sprawdza długość elementu `$parameters["default"]`, aby stwierdzić, czy istnieje domyślna odpowiedź na pytanie; jeśli jej nie ma, pomija generowanie parametru `value=` znacznika `<input>`. Instrukcja `return()` zwraca niezmodyfikowany element `$parameters["question"]` (dzięki czemu można w nim umieścić znaczniki formatujące), kilka spacji oraz znacznik `<input>`, dzięki któremu użytkownik może wpisać odpowiedź na pytanie.

Zamiast zwykłych spacji, " ", zastosowane zostały niepodzielne spacje HTML, ` `, ponieważ standardowe spacje mogą zostać pominięte (standard HTML traktuje kilka spacji jak jedną; zdarza się także, że ignoruje się pojedyncze spacje).

Weźmy teraz podstawową funkcję formularza z podrozdziału „Tworzenie formularza" i rozszerzmy ją tak, by pobierała pytania z tablicy. Zmodyfikowany kod korzysta

z funkcji `list()` oraz `each()`, pobierając kolejne pytania do tablicy o nazwie `$v` (więcej informacji na temat funkcji `list()` oraz `each()` znajdziesz w rozdziale 3.). Nowy kod dodaje zawartość pola `type` pytania do przedrostka `form_` (tworząc w ten sposób nazwę funkcji), umieszcza całość w zmiennej `$function`, po czym wywołuje tę zmienną jako funkcję (za każdym razem, gdy dodamy nowy typ pytania, musimy stworzyć także funkcję z odpowiadającą mu nazwą). Całość wygląda następująco:

```
function form($page, $question)
   {
   $text = "";
   while(list($name, $v) = each($question))
      {
      if(strlen($text)) {$text .= "<br>";}
      $function = "form_" . $v["type"];
      $text .= $function($name, $v);
      }
   return("<form action=\"" . $page . "\" method=\"post\">"
      . $text
      . "<input type=\"submit\" name=\"submit\""
      . " value=\"Wyślij\">"
      . "</form>");
   }
```

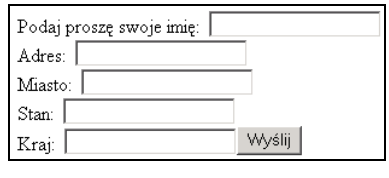
W tej funkcji pętla `while()` przetwarza listę pytań (zawartą w tablicy `$question`) i umieszcza sformatowane pytania w zmiennej `$text`. Ponieważ tablica może być pusta, funkcja zaczyna działanie od stworzenia pustego łańcucha `$text`.

Wewnątrz pętli `while()` kod sprawdza, czy w zmiennej `$text` znajduje się jakiś tekst; jeśli tak, wstawia znacznik podziału `<br>`, aby każde pytanie pojawiło się w osobnej linii. Następnie (ze stałej `form_` i elementu `type` z tablicy) tworzona jest nazwa funkcji, po czym wszystko razem zostaje dopisane do zawartości zmiennej `$text`.

Jeśli nie czytałeś rozdziału 10., możesz zastanawiać się nad wywołaniem `$function()`. Gdy PHP natrafia na zmienną użytą tam, gdzie powinna wystąpić funkcja, używa zawartości tej zmiennej jako nazwy funkcji, którą ma wywołać. Dzięki temu mechanizmowi można dynamicznie budować nazwy funkcji.

Instrukcja `return()` umieszcza na końcu pytania przycisk wysyłania, `<input type="submit">`, po czym całość ujmuje w znaczniki formularza. Pytania nie są wyrównane, ale zajmiemy się tym w następnym rozwiązaniu. Przedstawiony tu kod tworzy formularz pokazany na rysunku 9.7.

Rysunek 9.7. *Formularz z wieloma pytaniami*



Wyrównywanie kolumn

To rozwiązanie uzupełnia kod stworzony w poprzednim rozwiązaniu, wyrównując kolumny formularza i stanowi punkt wyjścia dla centralnego sterowania układem formularza.

W tym przykładzie wszystkie pytania zostaną wyrównane w kolumnach, gdyż przewijanie w pionie jest łatwiejsze, niż przewijanie w poziomie. Możesz z łatwością stworzyć kolejną kolumnę po prawej stronie (zawierającą komentarze do pytań), dodając po prostu kolejny element do tablicy `$question` i generując dodatkową kolumnę w funkcji `form()`.

Zacznijmy od listy pytań. W tym przypadku stworzymy tylko dwa pytania:

```
$question["name"] = array("type" => "text",
   "question" => "Podaj proszę swoje imię:");
$question["address"] = array("type" => "text",
   "question" => "Adres:");
```

Każda pozycja w tablicy `$question` posiada nazwę `name`, identyfikującą pytanie, oraz tablicę zawierającą dwa elementy: `type`, określający rodzaj znacznika `<input>`, oraz `question`, zawierający tekst pytania (łącznie z ewentualnymi znacznikami formatującymi HTML).

Następny fragment kodu to funkcja `form_text()` z podrozdziału „Tworzenie długiej listy w formularzu", w której usunięto tekst pytania, aby można go było formatować wewnątrz funkcji `form()`.

```
function form text($name, $parameters)
   {
if(!isset($parameters["default"]))
      {
      $parameters["default"] = "";
      }
   if(strlen($parameters["default"]))
      {
      $parameters["default"] = " value=\"" . $parameters["default"]
         . "\"";
      }
   return("<input type=\"text\" name=\"" . $name . "\""
      . $parameters["default"] . " >");
   }
```

Funkcja `form_text()` sprawdza, czy element `$parameters["default"]` zawiera jakiś tekst; jeśli tak, dodaje go do parametru `value=` znacznika `<input>`. Jeśli ten element jest pusty, nie ma sensu generować domyślnej wartości pytania.

Instrukcja `return()` umieszcza całość w tekstowym znaczniku `<input>`.

Skopiuj funkcję `form()` z podrozdziału „Tworzenie długiej listy w formularzu" i przenieś element `$v["question"]` na początek funkcji formatującej pytanie. Funkcja `form()` przyjmuje listę pytań w tablicy `$question`, przetwarza ją za pomocą pętli `while()` i zawartość każdego pytania przekazuje do funkcji o nazwie `"form_" . $v["type"]` (gdzie `type` jest elementem `type` z tablicy pytania).

```
function form($page, $question)
   {
   $text = "";
   while(list($name, $v) = each($question))
      {
      if(strlen($text)) {$text .= "<br>";}
      $function = "form " . $v["type"];
      $text .= $v["question"] . "&amp;nbsp;&amp;nbsp;"
```

```
            . $function($name, $v);
        }
    return("<form action=\"" . $page . "\" method=\"post\">"
        . $text
        . "<input type=\"submit\" name=\"submit\""
        . " value=\"Wyślij\"></form>");
    }
```

Szczegóły działania funkcji `form()` zostały opisane w podrozdziale „Tworzenie długiej listy w formularzu", włącznie z wykorzystaniem zmiennej jako nazwy funkcji. Pełny opis stosowania zmiennych jako nazw funkcji znajdziesz w rozdziale 10.

W tej wersji funkcji `form()` po prostu usunąłem tekst pytania z funkcji formatującej i dodałem go wewnątrz funkcji `form()`, dzięki czemu tekst wszystkich pytań jest formatowany w ten sam sposób.

W tym momencie możemy użyć różnych technik formatowania pytań, z zastosowaniem znaczników akapitów, znaczników podziału oraz tabel. W tym przypadku użyję prostej, trzykolumnowej tabeli, z których środkowa posłuży jako odstęp pomiędzy kolumną pytania a kolumną odpowiedzi.

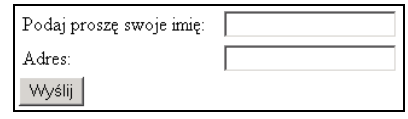
```
function form($page, $question)
    {
    $text = "";
    while(list($name, $v) = each($question))
        {
        if(strlen($text)) {$text .= "<br>";}
        $function = "form_" . $v["type"];
        $text .= "<tr><td>" . $v["question"] . "</td>"
            . "<td>  </td>"
            . "<td>" . $function($name, $v) . "</td>";
        }
    return("<form action=\"" . $page . "\" method=\"post\">"
        . "<table>" . $text . "</table>"
        . "<input type=\"submit\" name=\"submit\""
        . " value=\"Wyślij\">"
        . "</form>");
    }
```

W tej wersji funkcji `form()` zostały zastosowane znaczniki wiersza tabeli (`<tr>`) oraz komórki tabeli (`<td>`), które umieściłem wokół tekstu generowanego wewnątrz pętli `while()`. Dzięki temu każde pytanie znajdzie się w nowym wierszu tabeli. Pierwsza kolumna tabeli zawiera tekst pytania, druga niewielki odstęp, zaś trzecia zawiera znacznik `<input>`, w którym użytkownik może wpisać odpowiedź.

Instrukcja `return()` ujmuje tekst pytań w znaczniki początku i końca tabeli, dzięki czemu na stronie zostają wygenerowane komórki wewnątrz formularza. Ta symetria pozwala nam uniknąć problemów z niektórymi przeglądarkami, które nie akceptują błędnie sformatowanych tabel.

Przycisk *Wyślij* znajduje się poza tabelą i jako taki nie zostanie z nią wyrównany. Spójrz na wynik wyświetlony w przeglądarce, a następnie rozważ dokonanie modyfikacji, którą za chwilę zaproponuję. Formularz wyświetlony w przeglądarce został pokazany na rysunku 9.8.

Rysunek 9.8. *Formularz z kilkoma pytaniami, wyrównanymi w kolumnach*

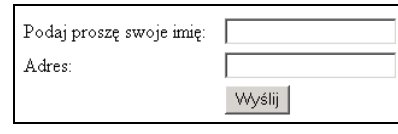


Nie podoba mi się miejsce, w którym znajduje się przycisk *Wyślij*. Uważam, że powinien znaleźć się on bezpośrednio poniżej pól odpowiedzi, dlatego dokonam „ręcznie" jeszcze jednej zmiany. Aby wyrównać przycisk *Wyślij*, zastąpię linię `<table> . $text . <table>` oraz linię znacznika `submit` następującym kodem:

```
. "<table>" . $text . "<tr><td></td><td></td>"
. "<td><input type=\"submit\" name=\"submit\""
. " value=\"Wyślij\"></td></tr></table>"
```

Nowy kod umieszcza przycisk *Wyślij* w nowym wierszu, w trzeciej komórce tabeli, dzięki czemu zostaje on wyświetlony tuż pod ostatnim znacznikiem `<input>`. Gdy przesuwasz wzrok i myszkę w dół strony, trafiasz bezpośrednio na przycisk *Wyślij*. W tym momencie nasz formularz wygląda tak, jak pokazano na rysunku 9.9.

Rysunek 9.9. *Formularz z polami i przyciskiem Wyślij ułożonymi w równą kolumnę*



Wybór jednej z kilku możliwości

Jeśli otwierasz własne biuro podróży i na swojej witrynie pytasz klientów, który kraj chcą odwiedzić, powinieneś zastosować formularz z długą listą wyboru, na której klienci będą mogli zaznaczać cel swojej planowanej podróży. Na podstawie tego założenia w tym rozwiązaniu użyjemy tablicy zawierającej listę nazw krajów, znacznik `<select>` formularza HTML (w celu wyświetlenia listy krajów) oraz zmienną, do której trafi wybrana pozycja.

W tym celu stworzymy najpierw tablicę krajów:

```
$country[] = "Australia";
$country[] = "Austria";
$country[] = "Azerbejdżan";
```

Potrzebujemy funkcji do wyświetlania pytań zawierających listy wyboru; posłuży nam do tego przedstawiona tu funkcja `form_select()`. Przyjmuje ona nazwę tablicy, listę pozycji do umieszczenia na liście wyboru oraz opcjonalny parametr, w którym możemy określić, która z pozycji ma zostać wstępnie zaznaczona.

```
function form_select($name, $parameters)
   {
   $output = "";
   while(list($k, $v) = each($parameters["list"]))
      {
      if(isset($parameters["default"]) and $v == $parameters["default"])
         {
         $output .= "<option selected>" . $v . "</option>";
```

```
        }
      else
        {
        $output .= "<option>" . $v . "</option>";
        }
      }
   return("<select name=\"" . $name . "\">"
      . $output . "</select>");
   }
```

Funkcja `form_select()` jest podobna do funkcji formatujących pytanie, stworzonych w innych rozwiązaniach w tym rozdziale, gdyż przyjmuje nazwę odpowiedzi oraz domyślną wartość, przedstawiającą najbardziej prawdopodobny wybór. Cała reszta jest inna, gdyż w tym przypadku tworzymy znacznik `<select>`.

Znacznik `<select>` pozwala na przedstawienie użytkownikowi listy pozycji, spośród których może dokonać wyboru bez konieczności wpisywania odpowiedzi. W niektórych przeglądarkach i systemach operacyjnych można wybierać więcej niż jedną pozycję, ale tylko wtedy, gdy w znaczniku `<select>` zostanie umieszczony parametr `multiple`.

W funkcji `form_select()` lista odpowiedzi jest zawarta w tablicy `$list`. Pętla `while()` przetwarza tablicę `$list` i generuje poszczególne znaczniki `<option>`, z których każdy zawiera jedną pozycję wyboru. Domyślna pozycja jest zaznaczana przez dodanie słowa kluczowego `selected` do znacznika `<option>`. Znaczniki `<option>` są dodawane do łańcucha `$output`, który na wszelki wypadek jest tworzony jeszcze przed wejściem do pętli `while()`.

Instrukcja `return()` umieszcza nazwę pytania i łańcuch `$output` wewnątrz znacznika `<select>`.

Jak można wyświetlić nowe pytanie? Zmodyfikuj tablicę pytań, używaną w poprzednich przykładach formularzy, tak, aby tablica krajów stała się parametrem `list` dla pytania typu `select`. Gdy to zrobisz, po to, by łatwo obsłużyć ten rodzaj pytania, wystarczy niewielka modyfikacja funkcji `form()` z poprzednich rozwiązań.

```
$question["destination"] = array("type" => "select",
   "question" => "Który kraj chcesz odwiedzić?",
   "list" => $country);
```

Każdy element tablicy `$question` jest tablicą zawierającą nazwę pytania, jego typ oraz treść. W przypadku pytań typu `select` element ten zawiera także listę pozycji do wyboru, które zostaną wyświetlone w znaczniku `<select>`.

Wartość klucza tablicy `$question` odpowiada nazwie zmiennej, jaką otrzyma skrypt pobierający odpowiedzi z formularza. Pole `type` określa funkcję, służącą do sformatowania pytania, zaś pole `question` zawiera wyświetlany tekst pytania.

Możesz dodać opcjonalny element o nazwie `default`, wstawiający domyślną odpowiedź na pytanie. W listach `<select>` domyślna odpowiedź musi pasować do pozycji na liście wyboru, z uwzględnieniem wielkości liter. Bezpieczny sposób dodawania domyślnej odpowiedzi został pokazany w kolejnym fragmencie kodu. Gdy do listy krajów w tablicy `$country` dodajesz pozycję „Anguilla”, najpierw umieść ją w zmiennej `$country_default`, a dopiero potem dodaj tę zmienną do tablicy `$country`:

```
$country_default = "Anguilla";
$country[] = $country_default;
```

Gdy dodajesz pozycję do tablicy `$question`, pobierz wartość elementu `default` bezpośrednio ze zmiennej `$country_default`. W ten sposób dodasz domyślną pozycję wyboru. Kolejny przykład przedstawia poprzednie pytanie z dodaną domyślną pozycją:

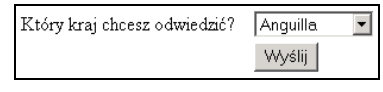
```
$question["destination"] = array("type" => "select",
    "default" => $country_default,
    "question" => "Który kraj chcesz odwiedzić?",
    "list" => $country);
```

Teraz możesz skopiować funkcję `form()` z poprzednich rozwiązań i użyć jej z nowym typem pytania bez jakichkolwiek modyfikacji. W funkcji `form()` lista pytań jest tablicą, której każdy element jest przetwarzany za pomocą funkcji odpowiedniej dla danego typu pytania (wybieranej na podstawie pola `type` w tablicy pytania). W poprzednich przykładach funkcja pytania otrzymywała nazwę pytania i wartość domyślną, lecz w przypadku funkcji `form_select()` musimy przekazać dodatkowy parametr. Funkcja `form()` nadal działa, ponieważ wszystkie parametry poza nazwą pytania są przekazywane w pojedynczej tablicy.

Alternatywą dla tablicy z parametrami jest użycie stałej listy trzech parametrów i modyfikacja istniejącej funkcji `form_text()`, aby przyjmowała trzeci, niepotrzebny parametr; w ten sposób każda z funkcji pytań przyjmowałaby dokładnie trzy parametry.

Pokrewne rozwiązanie: **Znajdziesz na stronie:**
Użycie domyślnych parametrów funkcji 344

Przedstawiony dalej kod rozszerza tabelę o dodatkowy wiersz i przenosi do tego wiersza przycisk *Wyślij*, aby można go było wyrównać do pól odpowiedzi. Uruchom stronę, po czym przeglądarka powinna wyświetlić formularz pokazany na rysunku 9.10.



Rysunek 9.10. *Formularz z rozwijaną listą wyboru*

```
function form($page, $question)
   {
   $text = "";
   while(list($name, $v) = each($question))
      {
      if(strlen($text)) {$text .= "<br>";}
      $function = "form_" . $v["type"];
      $text .= "<tr><td>" . $v["question"] . "</td>"
         . "<td>  </td>"
         . "<td>" . $function($name, $v) . "</td>";
      }
   return("<form action=\"" . $page . "\" method=\"post\">"
      . "<table>" . $text . "<tr><td></td><td></td>"
      . "<td><input type=\"submit\" name=\"submit\""
      . " value=\"Wyślij\"></td></tr></table>"
      . "</form>");
   }
```

Potrzebujemy teraz kodu do wyświetlenia wybranej pozycji. Przedstawiony poniżej fragment kodu sprawdza, czy istnieje zmienna o nazwie `$destination` (gdyż `destination` jest nazwą pytania w tablicy `$question`), po czym wyświetla jej wartość:

```
if(isset($destination))
   {
   print("<br>Wybrałeś kraj: " . $destination);
   }
```

Gdy formularz przekazuje wartość do docelowej strony za pomocą łańcucha URL albo metody `GET` lub `POST`, skrypt PHP w docelowej stronie otrzymuje tę wartość w zmiennej o nazwie zgodnej z nazwą w parametrze `name` tego znacznika HTML, który stworzył tę wartość. Jeśli zdefiniujesz znacznik `<input name="pet">` i ktoś w tym polu wpisze wartość „pies”, skrypt otrzyma zmienną `$pet` zawierającą łańcuch `"pies"`.

Wybór jednej z kilku możliwości za pomocą przycisków radiowych

Gdy podróżujesz samochodem, możesz natychmiast zmienić stację radiową za pomocą jednego z zaprogramowanych wcześniej przycisków. Któryś z twórców dodał taką możliwość także do HTML, w postaci znacznika `<input>` typu `radio`. W przypadku krótkich list wyboru przyciski radiowe są dobrą alternatywą dla znaczników `<select>`, gdyż od razu widać wszystkie możliwości. Z drugiej strony, znacznik `<select>` nadaje się do dłuższych list, ponieważ każda opcja na liście przycisków radiowych zajmuje miejsce na stronie.

Aby użyć przycisków radiowych w przykładzie biura podróży z poprzedniego podrozdziału, zaczniemy od stworzenia tablicy z listą krajów:

```
$country[] = "Australia";
$country[] = "Austria";
$country[] = "Azerbejdżan";
```

Potrzebujemy funkcji do wyświetlania pytań z przyciskami radiowymi; posłuży nam do tego funkcja `form_radio()`. Funkcja ta przyjmuje nazwę pytania oraz tablicę z jego parametrami (włącznie z opcjonalnym parametrem `default`, określającym, która z opcji ma zostać domyślnie zaznaczona).

```
function form_radio($name, $parameters)
   {
   $output = "";
   while(list($k, $v) = each($parameters["list"]))
      {
      if(strlen($output))
         {
         $output .= "<br>";
         }
      $output .= "<input name=\"" . $name . "\"" . " type=\"radio\""
         . " value=\"" . $v . "\"";
      if(isset($parameters["default"]) and
         $v == $parameters["default"])
         {
         $output .= " checked";
```

```
        }
      $output .= ">" . $v . "\n";
      }
   return($output);
   }
```

Funkcja `form_radio()` jest podobna do funkcji `form_select()`, gdyż przyjmuje nazwę pytania i jego parametry, a także obsługuje wybór opcji domyślnej. Różnice wynikają z tego, że w tym przypadku tworzymy znacznik `<input>`. Podobny kod został zastosowany w następnym rozdziale do wygenerowania listy dostępnych możliwości.

Aby wyświetlić nowy rodzaj pytania, skopiuj tablicę pytania z poprzedniego przykładu (jak to pokazano w kolejnym kodzie) i przekaż ją do funkcji `form()`, także skopiowanej z poprzedniego przykładu.

```
$question["destination"] = array("type" => "radio",
   "question" => "Który kraj chciałbyś odwiedzić?",
   "list" => $country);
```

Wartość klucza tablicy `$question` odpowiada nazwie zmiennej, jaką otrzyma skrypt pobierający odpowiedzi z formularza. Pole `type` określa funkcję, służącą do sformatowania pytania, zaś pole `question` zawiera wyświetlany tekst pytania. Możesz dodać opcjonalny element o nazwie `default`, wstawiający domyślną odpowiedź na pytanie.

Wybór kilku z wielu możliwości

W poprzednim przykładzie, w którym otwierałeś własne biuro podróży i pytałeś klientów o kraj, który chcieliby odwiedzić, postanowiłeś, że byłoby przydatne posiadanie formularza zawierającego długą listę krajów, które mogliby wybierać Twoi klienci. W tym rozwiązaniu użyjemy przycisków opcji formularza HTML, tablicy, zawierającej listę nazw krajów oraz tablicy, w której zostaną umieszczone wybrane pozycje. Nasz nowy kod stanowi niewielką modyfikację kodu użytego w podrozdziale „Wybór jednej z kilku możliwości”, zaś znacznik `<select>` został zastąpiony przez znacznik `<input type="checkbox">`. Jeśli użyjesz przedstawionego tutaj kodu, dane zwrócone przez formularz będą miały postać tablicy z listą wybranych pozycji, a nie prostej zmiennej z jedną pozycją.

Być może domyśliłeś się już, że lubię tablice PHP, ponieważ mogą one być używane w formularzach do zwracania list wartości. Wielowymiarowe tablice jednak nie zadziałają i pozostanie nam jedynie zastąpienie standardowego parametru HTML `name="nazwazmiennej"` przyjaznym dla PHP parametrem `name="nazwazmiennej[]"` (choć to ograniczenie może zostać zniesione w przyszłych wersjach przeglądarek, jak i samego PHP).

Zaczniemy od stworzenia tablicy z listą nazw krajów:

```
$country[] = "Australia";
$country[] = "Austria";
$country[] = "Azerbejdżan";
```

Potrzebujemy funkcji, która tworzyłaby znaczniki `<input type="checkbox">` dla każdej z pozycji tej tablicy. Funkcja `form_checkbox()` przyjmuje nazwę pytania oraz listę

jego parametrów, do których należy lista opcji przeznaczonych do wyboru oraz pole default, określające, czy wszystkie opcje mają być domyślnie zaznaczone (użytkownik może wtedy po prostu wyłączyć te opcje, których nie potrzebuje).

Zwróć uwagę na to, że parametr name="" w znaczniku <input> zawiera teraz po nazwie pola nawiasy kwadratowe — [] — informujące PHP, że otrzymana wartość jest następną pozycją tablicy. Nawiasy kwadratowe powodują, że parametr name odpowiada w PHP linii:

```
$destination[] = "_nazwa kraju_";
```

Znacznik <input> zawiera także parametr value="". W HTML możesz go pominąć; przeglądarka użyje wtedy łańcucha zawartego pomiędzy znacznikami <input> a </input>. W PHP i niektórych przeglądarkach pominięcie parametru value da w rezultacie pozycję tablicy zawierającą wartość on. Można tego uniknąć, stosując jawnie parametr value.

W przypadku zastosowania przycisków opcji wybrane pozycje są oznaczane słowem kluczowym checked, a nie selected, dlatego pamiętaj o zmodyfikowaniu odpowiedniej linii kodu. W porównaniu z poprzednim przedstawionym w tym rozdziale rozwiązaniem, ta funkcja używa liniowej metody budowania łańcucha wyjściowego, która lepiej nadaje się do tworzenia długich łańcuchów na podstawie wielu opcji. Znaczniki są budowane część po części, dzięki czemu można w przyszłości dodać kod bez modyfikacji istniejącego kodu. Ponadto na końcu łańcuchów umieszczono znaki \n, dzięki czemu dużo łatwiejsze jest przeglądanie źródła HTML w przeglądarce. Gdy piszesz w PHP inne funkcje konstruujące łańcuchy, możesz porównać to rozwiązanie z tym, które zastosowano w funkcji form_select().

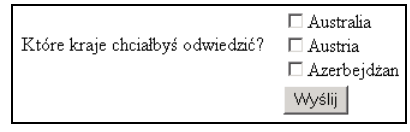
```
function form_checkbox($name, $parameters)
   {
   $output = "";
   while(list($k, $v) = each($parameters["list"]))
      {
      if(strlen($output))
         {
         $output .= "<br>";
         }
      $output .= "<input name=\"" . $name . "[]\""
         . " type=\"checkbox\""
         . " value=\"" . $v . "\"";
      if(is_array($parameters["default"])
      and in_array($v, $parameters["default"]))
         {
         $output .= " checked";
         }
      $output .= ">" . $v . "</input>\n";
      }
   return($output);
   }
```

Funkcja form_checkbox() przyjmuje nazwę pytania oraz tablicę z jego parametrami (do których należą lista opcji oraz ich wartość domyślna).

Pętla while() konstruuje listę w zmiennej $output, dlatego zostaje ona wcześniej zainicjalizowana. Pętla ta przetwarza listę i tworzy po jednym znaczniku <input> dla każdej

opcji. W celu ułożenia listy w pionową tabelę, przed każdym znacznikiem `<input>` zostaje wstawiony znacznik przerwy, `<br>`. Aby zabezpieczyć się przed stworzeniem pustej linii przed pierwszą opcją, znacznik `<br>` jest wstawiany tylko wtedy, gdy w zmiennej `$output` znajduje się jakiś tekst (stąd obecność w kodzie testu `if(strlen($output))`).

Znacznik `<input>` otrzymuje typ `checkbox` oraz nazwę identyfikującą go na docelowej stronie. Znacznik typu `checkbox` powoduje stworzenie przycisku opcji, pokazanego na rysunku 9.11. Każdy przycisk opcji jest osobną jednostką, dzięki czemu użytkownicy mogą do woli włączać i wyłączać opcje.



Rysunek 9.11. *Formularz z kilkoma przyciskami opcji*

Jak można wyświetlić nowe pytanie? Zmodyfikuj tablicę pytań, używaną w poprzednich przykładach formularzy, tak, aby tablica krajów stała się parametrem `list` dla pytania typu `checkbox`:

```
$question["destination"] = array("type" => "checkbox",
   "question" => "Które kraje chciałbyś odwiedzić?",
   "list" => $country);
```

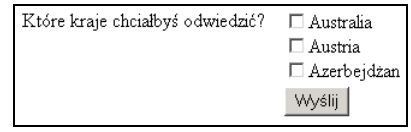
Kiedy to zrobisz, możesz wziąć funkcję `form()` z poprzedniego rozwiązania i bez jakichkolwiek modyfikacji użyć jej do wyświetlenia listy pytań. W ten sposób użyłbyś kodu:

```
function form($page, $question)
   {
   $text = "";
   while(list($name, $v) = each($question))
      {
      if(strlen($text)) {$text .= "<br>";}
      $function = "form_" . $v["type"];
      $text .= "<tr><td>" . $v["question"] . "</td>"
         . "<td>  </td>"
         . "<td>" . $function($name, $v) . "</td></tr>";
      }
   return("<form action=\"" . $page . "\" method=\"post\">"
      . "<table>" . $text . "<tr><td></td><td></td>"
      . "<td><input type=\"submit\" name=\"submit\""
      . " value=\"Wyślij\"></td></tr></table>"
      . "</form>");
   }
```

Funkcja `form()` przyjmuje nazwę strony, do której trafią odpowiedzi oraz listę pytań. Pętla `while()` przetwarza pytania, budując odpowiedni kod HTML. Instrukcja `return()` dodaje przycisk *Wyślij* oraz umieszcza całość w znacznikach formularza i tabeli.

Choć większości osób spodoba się formularz w takiej postaci, jak pokazana na rysunku 9.11, jednak ktoś mógłby poprosić o formularz z treścią pytania wyrównaną do górnej krawędzi, jak na rysunku 9.12. Aby dokonać tej niewielkiej zmiany, a jednocześnie uczynić ją standardową dla wszystkich formularzy, zachowując przy tym łatwość modyfikacji bez potrzeby zmiany kodu, w zmiennej `$option` umieścimy wspólne opcje formatowania:

```
$option["form"]["valign"] = "top";
```



Rysunek 9.12. *Formularz z pytaniem wyrównanym do najwyższej opcji*

Gdy dokonujemy zmian, możemy także dodać linię sterującą odstępem pomiędzy kolumnami, aby móc łatwo dodawać i usuwać odstęp w miarę rozwoju formularzy na stronie. Tablica `$option` może być załadowana ze wspólnego pliku nagłówkowego, administracyjnej bazy danych lub z pliku XML, generowanego przez administracyjną stronę witryny. Aby wprowadzić tę zmianę, możesz napisać:

```
$option["form"]["separator"] = "  ";
```

Teraz powinieneś zmodyfikować funkcję `form()`, tak, aby stosowała formatowanie dla wszystkich pól formularza i zastąpić zakodowany w skrypcie odstęp wartością załadowaną z tablicy `$option`. W innych językach globalne zmienne są deklarowane raz, na początku skryptu. W PHP zmienna nie jest zmienną globalną, dopóki nie wymienisz jej w deklaracji `global` wewnątrz swojej funkcji. To oznacza, że łatwiej jest zadeklarować pojedynczą tablicę w deklaracji `global` i przechowywać wszystko jako pozycje tablicy, niż przechowywać wszystkie wartości w osobnych zmiennych i próbować dodać każdą z nich do deklaracji `global`.

Zanim użyjemy wartości z tablicy `$option`, za pomocą funkcji `$isset()` musimy sprawdzić, czy ta wartość jest ustawiona i odpowiednio zareagować, jeśli została pominięta. W przypadku łańcucha separatora zmieniłem kod tak, by zupełnie pomijał kolumnę separatora tabeli w przypadku, gdy wartość separatora nie została zdefiniowana. Oprócz tego, na wypadek, gdyby ktoś zechciał użyć dodatkowych opcji formatujących znacznik `<td>` kolumny separatora (jak parametr `bgcolor`), łańcuch znacznika umieściłem w osobnej zmiennej `$td`. Aby zobaczyć, jak to działa, spójrz na następujący kod:

```
function form($page, $question)
   {
   global $option;
   $td = "<td";
   if(isset($option["form"]["valign"]))
      {
      $td .= " valign=\"" . $option["form"]["valign"] . "\"";
      }
   $td .= ">";
   $sep = "";
   if(isset($option["form"]["separator"]))
      {
      $sep = $td . $option["form"]["separator"] . "</td>";
      }
   $text = "";
   while(list($name, $v) = each($question))
      {
      if(strlen($text)) {$text .= "<br>";}
      $function = "form_" . $v["type"];
      $text .= "<tr>" . $td . $v["question"] . "</td>" . $sep . $td;
      $text .= $function($name, $v)
          . "</td></tr>";
      }
   return("<form action=\"" . $page . "\" method=\"post\">"
```

```
        . "<table>" . $text . "<tr><td></td><td></td>"
        . "<td><input type=\"submit\" name=\"submit\""
        . " value=\"Wyślij\"></td></tr></table>"
        . "</form>");
    }
```

Funkcja `form()` definiuje symbol `$option` jako zmienną globalną, dzięki czemu może odwoływać się do zmiennej `$option` zdefiniowanej poza funkcją `form()`.

Kod zaczynający się od `$td =` tworzy definicję komórki tabeli. Znacznik `<td>` jest tworzony z parametrami pochodzącymi z tablicy `$option`, dlatego możemy użyć tej tablicy do precyzyjnej kontroli nad formatowaniem komórki tabeli HTML. W tym przykładzie obsługuję jedynie parametr `valign` znacznika `<td>`, sterujący wyrównaniem ułożenia tekstu w komórce tabeli w pionie. Możesz to rozszerzyć na kolory i wszystkie inne parametry znacznika `<td>`.

Dlaczego generujemy formatowanie w ten sposób, a nie za pomocą kaskadowych arkuszy stylów (CSS)? Ponieważ niektóre przeglądarki nie obsługują kaskadowych arkuszy stylów, a inne obsługują je błędnie. Nawet najnowsze przeglądarki nie zgadzają się co do szczegółów interpretacji CSS.

Kod zaczynający się od `$sep =` przygotowuje po prostu komórkę tabeli, która będzie oddzielać od siebie kolumny tekstu i opcji. Użyłem kolumny, ponieważ ekrany są szerokie i do wizualnego rozdzielenia danych potrzeba więcej miejsca w poziomie, niż w pionie.

Linia kodu zaczynająca się od `$text = "<tr>"` jest istniejącą wcześniej linią, w której ustalony znacznik `<td>` został zamieniony na łańcuch `$td`, zaś łańcuch `"<td>  </td>"` został zastąpiony przez `$sep`.

Dodatkowa praca w funkcji `form()` owocuje lepszym, bardziej elastycznym formularzem. Do przetworzenia wyników z tego formularza będziesz potrzebował przedstawionego dalej kodu. Dane znajdują się w zmiennej `$destination`, dlatego najpierw sprawdzamy, czy taka zmienna istnieje, potem sprawdzamy, czy jest tablicą, a dopiero potem przetwarzamy ją w pętli, wyświetlając wartości. Każda zaznaczona pozycja formularza będzie posiadała swoją wartość (z parametru `value=""` formularza) w tablicy. Gdy dane trafią do skryptu, możesz je przetwarzać w dowolny sposób. A oto przykład takiego kodu:

```
$d = "";
if(isset($destination))
   {
   $d = $destination;
   if(is_array($destination))
      {
      $d = "";
      while(list($k, $v) = each($destination))
         {
         $d .= "<tr><td>" . $v . "</td></tr>";
         }
      $d = "<table>" . $d . "</table>";
      }
   }
```

Kod inicjalizuje łańcuch w zmiennej `$d`, w której umieścimy kod HTML i tekst. Instrukcja `if(isset())` sprawdza, czy zmienna `$destination` istnieje (przy pierwszym wejściu na stronę zmienna ta może nie istnieć), po czym kopiujemy jej zawartość do zmiennej `$d`

(ponieważ zmienna $destination może zawierać sformatowany łańcuch). Jeśli $destination okaże się tablicą, zmienna $d jest zerowana, a $destination zostaje przetworzona w pętli. Wewnątrz pętli tworzone są wiersze tabeli, które są dopisywane do zmiennej $d, a na koniec zmienna ta zostaje ujęta pomiędzy znaczniki <table> i </table>.

Zachowywanie odpowiedzi i podświetlanie błędów

W poprzednich rozwiązaniach zakładaliśmy, że klient wypełnia formularz i jednokrotnie klika na przycisku *Wyślij*. Ale co zrobić, gdy chcesz przenieść wartości ze strony do strony w wielostronicowym formularzu lub ponownie wyświetlić tę samą stronę z dopisanymi komunikatami o błędach? W takim przypadku do tablicy $question powinieneś po prostu dodać kolejne pola oraz stworzyć nowe funkcje przetwarzające. W przypadku błędów możesz dodać nową, początkową kolumnę na wypisywane na czerwono komunikaty o błędach, aby nie zmieniać pionowego układu formularza. Użyjesz w zasadzie tej samej tablicy $question, co w poprzednich rozwiązaniach, ale uzupełnionej o dodatkowe wartości. Aby umożliwić łatwe przetestowanie komunikatów o błędach, możesz zmienić pytanie tak, by pozwalało na wybór tylko dwóch krajów, tak jak w następującym fragmencie kodu:

```
$question["destination"] = array("type" => "checkbox",
   "question" => "Wybierz nie więcej niż dwa kraje:",
   "list" => $country);
```

W celu dopasowania odpowiedzi do pytań każde pytanie jest indeksowane nazwą. Dzięki temu, gdy przetwarzasz zmienną odpowiedzi (na przykład $destination), możesz odwoływać się do pytań jako do zmiennych $question["destination"] i z łatwością wstawiać odpowiedzi i komunikaty o błędach jako dodatkowe pozycje.

Gdy w odpowiedzi wystąpi błąd i zechcesz wyświetlić komunikat zwracający na to uwagę użytkownika, jedyne, czego potrzebujesz w kodzie, to linia dodająca komunikat o błędzie do tablicy $question. W przedstawionym tu przykładzie zobaczymy, jak to działa w przypadku, gdy klient zaznaczy więcej niż dwa kraje. Oczywiście, gdy strona będzie już poprawnie wypisywać komunikaty o błędach, możesz opracować swoje własne komunikaty. W celu dodania komunikatu o błędzie do tablicy $question użyjemy kodu:

```
if(count($destination) > 2)
   {
   $question["destination"]["error"] =
      "Zaznaczyłeś więcej niż dwa kraje."
      . "<br><em>(Wskazówka: Dwa to liczba większa niż jeden.)</em>";
   }
```

Musimy się także upewnić, że na nowej stronie pojawią się poprzednie odpowiedzi, aby uchronić użytkowników przed ponownym wpisywaniem wszystkich pozycji. W tym celu po prostu dodamy poprzednie odpowiedzi do tablicy pytań jako nowe wartości domyślne:

```
if(isset($destination) and is_array($destination))
   {
   $question["destination"]["default"] = $destination;
   }
```

Zarówno lista domyślnych odpowiedzi, jak i lista $list mogą być załadowane z bazy danych. Jeśli posiadasz wspólną tabelę bazy danych dla danych pochodzących z tablicy

$list oraz z listy domyślnych odpowiedzi, możesz połączyć obie listy w jedną tablicę i załadować ją za pomocą kwerendy SQL. Oznacza to, że w funkcji form_checkbox() zniknie wartość $parameters["default"], zaś zmienna $v stanie się tablicą zawierającą zarówno wartość do wyświetlenia, jak i jakąś inną wartość, wskazującą, że pozycja powinna zostać domyślnie zaznaczona. W takiej sytuacji za pomocą niewielkiej zmiany w funkcji form() na początku pytania umieścilibyśmy komunikat o błędzie:

```
function form($page, $question)
   {
   global $option;
   $td = "<td";
   if(isset($option["form"]["valign"]))
      {
      $td .= " valign=\"" . $option["form"]["valign"] . "\"";
      }
   $td .= ">";
   $sep = "";
   if(isset($option["form"]["separator"]))
      {
      $sep = $td . $option["form"]["separator"] . "</td>";
      }
   $text = "";
   while(list($name, $v) = each($question))
      {
      if(isset($v["error"]))
         {
         $v["question"] = "<font color=\"Red\">" . $v["error"]
            . "</font><br>" . $v["question"];
         }
      $function = "form_" . $v["type"];
      $text .= "<tr>" . $td . $v["question"] . "</td>" . $sep . $td;
      $text .= $function($name, $v)
         . "</td>";
      }
   return("<form action=\"" . $page . "\" method=\"post\">"
      . "<table>" . $text . "<tr><td></td><td></td>"
      . "<td><input type=\"submit\" name=\"submit\""
      . " value=\"Wyślij\"></td></tr></table>"
      . "</form>");
   }
```

Pierwszą modyfikacją w funkcji form() jest kod przetwarzający element $v["error"]. Element error istnieje tylko wtedy, gdy na nasze pytanie została udzielona błędna odpowiedź, dlatego pierwszym krokiem powinno być sprawdzenie, czy element ten istnieje. Jeśli tak, kod pobiera z niego tekst, ustawia jego kolor na czerwony (za pomocą znacznika <font>) i dodaje go na początku tekstu pytania.