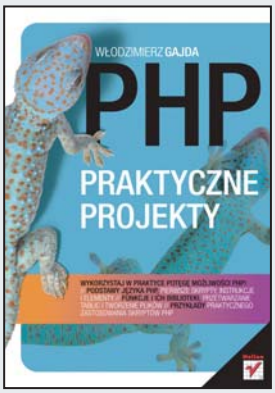
PHP. Praktyczne projekty



Autor: Włodzimierz Gajda
ISBN: 978-83-246-0943-7
Format: 158×235, stron: 552

Wykorzystaj w praktyce potęgę możliwości PHP!

- Podstawy języka PHP, pierwsze skrypty, instrukcje i elementy
- Funkcje i ich biblioteki, przetwarzanie tablic i tworzenie plików
- Przykłady praktycznego zastosowania skryptów PHP

Język PHP nie od dziś stanowi jeden z najłatwiejszych w nauce i najelastyczniejszych języków programowania. Jego ogromne możliwości wykorzystywane są na ogół przy tworzeniu interaktywnych stron internetowych, a obiektowy charakter oraz modułowość stanowią atuty dla programistów, wreszcie zwolnionych z konieczności ustawicznego przepisywania tego samego kodu. Jak każdego innego języka, także i PHP najłatwiej nauczyć się na konkretnych, praktycznych przykładach – wiele z nich zamieszczono właśnie w tej książce.

„PHP. Praktyczne projekty" to naprawdę wyjątkowy podręcznik do nauki PHP. Nie spodziewaj się tu suchych, typowo podręcznikowych teorii! Znajdziesz w nim za to omówienie wszystkich elementów języka PHP, działania skryptów, funkcji i bibliotek wraz z dowcipnymi i przejrzystymi przykładami ich użycia. Pierwsza część książki wprowadzi Cię w świat PHP oraz wyjaśni, jak stosować i łączyć poszczególne polecenia w dobrze działający kod. Z następnych rozdziałów wyniesiesz już umiejętności czysto praktyczne, dotyczące wszystkich obszarów wykorzystania tego języka przy tworzeniu najróżniejszych projektów.

- Instrukcje wyjściowe i sterujące, stałe, zmienne, wyrażenia oraz operatory
- Funkcje w PHP i ich biblioteki
- Programowanie obiektowe i formatowanie kodu PHP
- Podział skryptu na wiele plików i generowanie kodu HTML
- Przetwarzanie napisów litera po literze i podstawy przetwarzania tablic
- Tworzenie i wyszukiwanie plików, krojenie plików tekstowych
- Wyrażenia regularne i przetwarzanie wsadowe
- Wybór podstrony serwisu
- Walidacja zmiennych URL i stosowanie kilku zmiennych URL
- Szablony Smarty i PHP
- Wsadowe tworzenie baz danych i ich wizualne projektowanie w programie MySQL Workbench
- phpMyAdmin, PDO, Propel i inne tajemnicze nazwy
- Funkcje mysql_xxx
- Kontroler jednowymiarowy oraz dwuwymiarowy
- Moduł mod_rewrite
- Wzbogacanie aplikacji o obsługę przyjaznych URL-i oraz routing przyjaznych adresów URL

Przekonaj się, że PHP można nauczyć się nie tylko szybko, ale i przyjemnie!

Spis treści

Rozdział 23.

Pliki tekstowe, zmienne URL i szablony Smarty

Po omówieniu podstawowych zagadnień dotyczących szablonów Smarty przechodzimy do połączenia kilku opanowanych już elementów. Wykorzystując:

- pliki tekstowe,
- zmienne URL
- oraz szablony Smarty,

możemy przystąpić do opracowywania skryptów, których struktura będzie ewoluowała w kierunku architektury MVC.

Projekt 23.1. Powieści Agathy Christie

Folder *agatha-christie-dane/* zawiera pliki tekstowe z opisem powieści Agathy Christie. Każdy plik opisuje jedną powieść. Na przykład w pliku o nazwie *4-50-from-paddington.txt* zawarty jest opis książki zatytułowanej *4.50 from Paddington*:

```
TITLE|4.50 from Paddington
YEAR|1957
TYPE|Novel
BEST|no
DETECTIVE|Miss Marple
METHOD|Poison,Strangling
```

Opis zawiera tytuł utworu, rok wydania, rodzaj, informację logiczną, czy dana książka należy do kanonu najlepszych utworów pisarki, nazwisko detektywa oraz metody zbrodni opisane w książce. Dodatkowo plik *00lista.log* zawiera listę wszystkich powieści zawartych w folderze *agatha-christie-dane/*:

```
4.50 from Paddington*agatha-christie-dane/4-50-from-paddington.txt
A Caribbean Mystery*agatha-christie-dane/a-caribbean-mystery.txt
A Murder Is Announced*agatha-christie-dane/a-murder-is-announced.txt
...
```

Przygotuj skrypt PHP, który przedstawi menu z listą wszystkich tytułów powieści. Każda pozycja menu ma być hiperłączem do strony prezentującej szczegółowe dane wybranego utworu.

Rozwiązanie zadania rozpoczynamy od ustalenia przestrzeni adresów URL. Witryna ma prezentować dwa rodzaje stron: menu z listą utworów oraz szczegółowe informacje na temat wybranej powieści. Strona prezentująca menu będzie miała adres *index.php*, zaś szczegółowe dane książek zaadresujemy:

```
index.php?id=X
```

gdzie `X` będzie identyfikatorem wybranej powieści. Identyfikator ten będzie numerem wiersza w pliku *00lista.log*. Pierwszy wiersz pliku zawiera dane powieści pt. *4.50 from Paddington*, a zatem adres:

```
index.php?id=1
```

Będzie on powodował wyświetlenie szczegółowych danych dotyczących utworu *4.50 from Paddington*. Drugi wiersz w pliku *00lista.log* wskazuje utwór pt. *A Caribbean Mystery*, a więc adres:

```
index.php?id=2
```

będzie prezentował szczegółowe dane tej właśnie powieści. I tak dalej. Przestrzeń adresów URL stosowanych w skrypcie będzie więc następująca:

```
index.php        menu z listą wszystkich powieści
index.php?id=1   szczegóły utworu: 4.50 from Paddington
index.php?id=2   szczegóły utworu: A Caribbean Mystery
index.php?id=3   szczegóły utworu: A Murder Is Announced
...
```

Menu witryny przyjmie postać:

```
<ol>
  <li><a href="index.php?id=1">4.50 from Paddington</a></li>
  <li><a href="index.php?id=2">A Caribbean Mystery</a></li>
  <li><a href="index.php?id=3">A Murder Is Announced</a></li>
  ...
</ol>
```

Skrypt PHP został przedstawiony na listingu 23.1. Po odczytaniu i pokrojeniu pliku *00lista.log* tworzymy w zmiennej `$s` nowy obiekt Smarty. Następnie sprawdzamy, która strona została wybrana. Jeśli zmienna URL o nazwie `id` została podana (tj. funkcja `isset()` zwraca logiczną prawdę) oraz jest to poprawna liczba całkowita z zakresu od 1 do `$d['rows']` (tj. funkcja `str_ievpifr()` ma wartość `true`), to wybrana została strona jednego

z utworów. Numer wybranego utworu[1] umieszczamy w zmiennej `$nr`, po czym odczytujemy i kroimy plik zawierający opis utworu o podanym numerze. Nazwa pliku z opisem utworu jest dostępna w zmiennej `$d` pod indeksami:

```
$d['items'][1][$nr]
```

Pokrojona tablica trafia do szablonu pod nazwą `utwor` (metoda `assign()`), zaś zmienna `$akcja` otrzymuje wartość `utwor`.

Jeśli zmienna URL o nazwie `id` nie została podana, bądź jej wartość jest niepoprawna, to do szablonu trafia lista wszystkich tytułów powieści zawarta w zmiennej `$d['items'][0]`. Tablica ta zostaje przekazana do szablonu pod nazwą `menu`, zaś zmiennej `$akcja` przypisujemy wartość `glowna`.

Na zakończenie tworzenia skryptu PHP do szablonu przekazujemy zmienną `$akcja`, po czym szablon przetwarzamy i wyświetlamy.

Podsumowując, w zależności od wartości zmiennej URL o nazwie `id` do szablonu przekazujemy:

- zmienną `akcja` o wartości `utwor` oraz zmienną `utwor`, która zawiera pokrojony plik z opisem wybranej powieści,
- lub zmienną `akcja` o wartości `glowna` oraz zmienną `menu` zawierającą tytuły wszystkich utworów.

Szablon Smarty jest przedstawiony na listingu 23.2. Zawiera on funkcję `if`, która decyduje o tym, czy wyświetlić należy menu, czy szczegółowe dane jednej z książek:

```
{if $akcja == 'glowna'}
    ...
    wyświetlamy menu
{elseif $akcja == 'utwor'}
    ...
    wyświetlamy szczegółowe dane książki
{/if}
```

Menu jest produkowane za pomocą pętli `section`, która przetwarza tablicę `$menu`. Wartość zmiennej `id` umieszczanej w parametrze `href` hiperłącza powstaje na podstawie zmiennej `iteration` funkcji `section`:

```
{section name=i loop=$menu}
    <li><a href="index.php?id={$smarty.section.i.iteration}">{$menu[i]}</a></li>
{/section}
```

Natomiast szczegółowe dane utworu produkujemy, przetwarzając iteracyjnie tablicę `$utwor`. Ponieważ plik tekstowy zawiera zarówno etykiety (np. `TITLE`, `YEAR`, `DETECTIVE`), jak i dane (np. *4.50 from Paddington*, *1957*, *Miss Marple*), zatem elementy tablicy są wykorzystane zarówno wewnątrz komórek `td`, jak i komórek nagłówkowych `th`.

[1] Numer wybrany z menu jest zmniejszany o jeden, gdyż w menu stosujemy numerację od 1 do *n*, zaś indeksem tablicy jest liczba od 0 do *n* – 1.

Menu zawarte w liście `ol` jest zaprezentowane w postaci czterech kolumn dzięki następującym stylom CSS:

```
ol {
    float: left;
    list-style-type: none;
}
li {
    width: 300px;
    float: left;
}
```

Listing 23.1. *Projekt 23.1 pt. Powieści Agathy Christie — skrypt index.php*

```
$p = file_get_contents('00lista.log');
$d = string2VArray($p, '*');
$s = new Smarty();
if (isset($_GET['id']) && str_ievpifr($_GET['id'], 1, $d['rows'])) {
    $nr = $_GET['id'] - 1;
    $utwor = string2HArray(file_get_contents($d['items'][1][$nr]));
    $s->assign('utwor', $utwor['items']);
    $akcja = 'utwor';
} else {
    $s->assign('menu', $d['items'][0]);
    $akcja = 'glowna';
}
$s->assign('akcja', $akcja);
$s->display('index.tpl');
```

Listing 23.2. *Projekt 23.1 pt. Powieści Agathy Christie — szablon index.tpl*

```
<body>
<h1><a href="index.php">Agatha Christie</a></h1>
{if $akcja == 'glowna'}
    <h2>15 września 1890 &mdash; 12 stycznia 1976</h2>
    <ol>
        {section name=i loop=$menu}
            <li><a
href="index.php?id={$smarty.section.i.iteration}">{$menu[i]}</a></li>
        {/section}
    </ol>
{elseif $akcja == 'utwor'}
    <table>
        {section name=i loop=$utwor}
            <tr>
                <th>{$utwor[i][0]}</th>
                <td>{$utwor[i][1]}</td>
            </tr>
        {/section}
    </table>
{/if}
</body>
```

Projekt 23.2. Ogonki DE, FR, PL, RU

Napisz skrypt, który przedstawi w postaci witryny WWW zestawienie znaków diakrytycznych występujących w kilku językach. Witryna ma mieć menu główne prezentujące listę wszystkich dostępnych języków. Po wybraniu języka z listy należy przedstawić — w postaci tabeli HTML — listę znaków diakrytycznych występujących w danym języku wraz z przykładami użycia. Domyślną stroną witryny ma być pierwszy z dostępnych języków.

W zadaniu wykorzystaj plik *jezyki.txt* oraz pliki *de.txt*, *fr.txt*, *pl.txt* i *ru.txt*. Plik *jezyki.txt* zawiera listę wszystkich dostępnych języków:

```
Francuski|FR|fr.txt
Niemiecki|DE|de.txt
Polski|PL|pl.txt
Rosyjski|RU|ru.txt
```

To na jego podstawie ma powstać menu witryny. Każdy z plików *de.txt*, *fr.txt*, *pl.txt* oraz *ru.txt* wymienionych w pliku *jezyki.txt* ma identyczną strukturę. Pierwsza kolumna zawiera znak, druga kolumna zawiera encję, a trzecia — przykład użycia danej litery. Początkowe wiersze pliku *fr.txt* są następujące:

```
&#x00e0;|&amp;#x00e0;|&#x00e0; bient&#x00f4;t
&#x00c0;|&amp;#x00c0;|&#x00c0; BIENT&#x00d4;T
&#x00e2;|&amp;#x00e2;|g&#x00e2;teau
...
```

Ustalmy adresy URL, jakie będą stosowane w rozwiązaniu. Plik *jezyki.txt* zawiera kolejno wymienione języki francuski, niemiecki, polski oraz rosyjski. Zatem adresy URL przyjmą postać:

- `index.php?id=1` — strona prezentująca ogonki francuskie,
- `index.php?id=2` — strona prezentująca ogonki niemieckie,
- `index.php?id=3` — strona prezentująca ogonki polskie,
- `index.php?id=4` — strona prezentująca ogonki rosyjskie.

Stroną domyślną będzie strona prezentująca litery francuskie. Podanie adresu *index.php* będzie więc powodowało wyświetlenie strony opisującej język francuski.

Menu strony w kodzie HTML przyjmie postać:

```
<ol id="menu">
<li><a href="index.php?id=1">Francuski</a></li>
<li><a href="index.php?id=2">Niemiecki</a></li>
<li><a href="index.php?id=3">Polski</a></li>
<li><a href="index.php?id=4">Rosyjski</a></li>
</ol>
```

Skrypt *index.php* jest przedstawiony na listingu 23.3. Pracę rozpoczynamy od odczytania i pokrojenia pliku *jezyki.txt*. Następnie w zmiennej `$nr` ustalamy numer domyślnej podstrony serwisu, po czym sprawdzamy obecność i poprawność zmiennej `$_GET['id']`. Jeśli zmienna

ta jest obecna (tj. funkcja `isset()` zwraca `true`) oraz poprawna (tj. funkcja `str_ievpifr()` zwraca `true`), to wartość zmiennej `$_GET['id']` pomniejszoną o jeden zapamiętujemy w zmiennej `$nr`.

Gdy w zmiennej `$nr` ustalony jest numer wybranego języka (jest to albo wartość domyślna 0, albo wartość wybrana z menu), odczytujemy i kroimy odpowiedni plik tekstowy. Jego nazwa jest dostępna w zmiennej `$k['items'][2][$nr]`. Na zakończenie tworzymy obiekt Smarty i przekazujemy do niego dwie zmienne: pokrojony plik *jezyki.txt* (zmienna szablonu będzie się nazywała `menu`) oraz pokrojony plik z ogonkami wybranego języka (zmienna szablonu będzie się nazywała `jezyk`).

Szablon *ogonki.tpl* został przedstawiony na listingu 23.4. Menu powstaje przez iteracyjne przetworzenie tablicy `$menu`. Tym razem zamiast funkcji `section` stosujemy funkcję `foreach`:

```
<ol id="menu">
{foreach from=$menu item=opcja name=m}
<li><a href="index.php?id={$smarty.foreach.m.iteration}">{$opcja}</a></li>
{/foreach}
</ol>
```

W podobny sposób na stronie WWW umieszczamy tabelę prezentującą litery z ogonkami:

```
{foreach from=$jezyk item=litera}
<tr>
    <td>{$litera[0]}</td>
    <td>{$litera[1]}</td>
    <td>{$litera[2]}</td>
</tr>
{/foreach}
```

Listing 23.3. *Projekt 23.2 pt. Ogonki DE, FR, PL, RU — skrypt index.php*

```
$k = string2VArray(file_get_contents('jezyki.txt'));

$nr = 0;
if (isset($_GET['id']) && str_ievpifr($_GET['id'], 1, $k['rows'])) {
    $nr = $_GET['id'] - 1;
}

$jezyk = string2HArray(file_get_contents($k['items'][2][$nr]));

$s = new Smarty;
$s->assign('menu', $k['items'][0]);
$s->assign('jezyk', $jezyk['items']);
$s->display('ogonki.tpl');
```

Listing 23.4. *Projekt 23.2 pt. Ogonki DE, FR, PL, RU — szablon index.tpl*

```
<body>

<ol id="menu">
{foreach from=$menu item=opcja name=m}
<li><a href="index.php?id={$smarty.foreach.m.iteration}">{$opcja}</a></li>
{/foreach}
```

```
</ol>

<div id="content">
<table>
<tr>
    <th>znak</th>
    <th>encja</th>
    <th>przykład</th>
</tr>
{foreach from=$jezyk item=litera}
<tr>
    <td>{$litera[0]}</td>
    <td>{$litera[1]}</td>
    <td>{$litera[2]}</td>
</tr>
{/foreach}
</table>
</div>

</body>
```

Projekt 23.3. Fraszki

Przygotuj witrynę prezentującą fraszki Jana Kochanowskiego, wykorzystując szablony Smarty. Użyj danych z projektu 19.4.

Projekt ten stosuje następujące adresy URL:

```
index.php?id=1    fraszka Do gościa
index.php?id=2    fraszka Na nabożną
index.php?id=3    fraszka Na starą
...
```

skrypt powinien więc generować takie oto menu HTML:

```
<ol>
  <li><a href="index.php?id=1">Do gościa</a></li>
  <li><a href="index.php?id=2">Na nabożną</a></li>
  <li><a href="index.php?id=3">Na starą</a></li>
  ...
</ol>
```

Rozwiązanie zadania jest przedstawione na listingach 23.5 oraz 23.6. Fraszki są zawarte w plikach tekstowych w folderze *fraszki/*. Skrypt *index.php* rozpoczyna się więc od wyszukania wszystkich fraszek za pomocą funkcji `glob()`. Następnie tworzymy tablicę `$tytuly`, w której umieszczamy tytuły wszystkich znalezionych fraszek. Dzięki użyciu w pętli `foreach` zmiennej `$k` oraz dzięki wykorzystaniu indeksu `$k +1` indeksacja w tablicy `$tytuly` rozpocznie się od 1, a nie od 0.

Następnie przeprowadzamy walidację zmiennej `$_GET['id']`. Jeśli zmienna ta jest podana, a jej wartość jest poprawna, to zapamiętujemy ją w zmiennej `$nr`. W przeciwnym razie zmienna `$nr` przyjmuje wartość domyślną 1 (czyli domyślnie wyświetlamy treść pierwszej dostępnej fraszki).

W kolejnym kroku na podstawie zmiennej `$nr` odczytujemy plik z treścią fraszki, po czym tworzymy dwie zmienne: `$tytul` oraz `$tresc`. Pierwsza z nich będzie zawierała tytuł, a druga treść wybranej fraszki.

Na zakończenie przygotowane zmienne przekazujemy do szablonu. Będą to:

- `tytuly` — indeksowana od 1 tablica z tytułami wszystkich fraszek (potrzebna do menu),
- `tytul` — tytuł wybranej fraszki,
- `tresc` — treść wybranej fraszki.

Przetworzenie szablonu wykonujemy, tym razem wywołując funkcję `fetch()`:

```
$strona = $s->fetch('index.tpl');
```

Metoda ta zwraca w postaci napisu przetworzony szablon. Szablon ten możemy zapisać w pliku, w bazie danych lub wysłać do przeglądarki za pomocą instrukcji `echo`:

```
echo $strona;
```

Przedstawiony na listingu 23.6 szablon *index.tpl* wykorzystuje trzy zmienne: `$tytuly`, `$tytul` oraz `$tresc`. Menu strony powstaje na podstawie tablicy `$tytuly`. W pętli `foreach` stosujemy atrybut `key`, dzięki czemu mamy dostęp do identyfikatora kolejnej fraszki. Identyfikatorem tym[2] jest indeks w tablicy `$tytuly`:

```
<ol>
{foreach from=$menu key=myId item=fraszka}
  <li><a href="index.php?id={$myId}">{$fraszka}</a></li>
{/foreach}
</ol>
```

Treść fraszki przekształcamy modyfikatorem `nl2br` i umieszczamy wewnątrz elementu `p`:

```
<p>{$tresc|nl2br}</p>
```

Zaś tytuł fraszki pojawia się w tytule strony WWW:

```
<title>Jan Kochanowski: Fraszki: {$tytul}</title>
```


Wskazówka

Szablony Smarty pozwalają na iteracyjne przetwarzanie tablic. Dzięki temu kod PHP nie zawiera żadnych znaczników HTML. Menu HTML wykorzystuje funkcję `foreach` i jest zawarte w szablonie *.tpl*. Takie rozwiązanie jest zastosowane w projekcie 23.3.

Jeśli szablon jest wypełniany danymi przy użyciu funkcji `str_replace()`, kod PHP będzie zawierał niektóre znaczniki, gdyż metoda ta nie pozwala na iteracyjne przetwarzanie tablic. Analizując kod projektu 19.4, zauważymy, że menu HTML jest generowane wewnątrz skryptu PHP w funkcji `generuj_menu()`. Taka separacja prezentacji i przetwarzania nie jest pełna. Jest to rozwiązanie gorsze od rozwiązania stosującego Smarty.

[2] W poprzednich projektach tego rozdziału identyfikator wybranej pozycji menu powstawał na podstawie zmiennej `{$smarty.section.i.iteration}` lub `{$smarty.foreach.m.iteration}`.

Listing 23.5. *Projekt 23.3 pt. Fraszki — skrypt index.php*

```
$plks = glob('fraszki/*.txt');
$tytuly = array();
foreach ($plks as $k => $plk) {
    $p = file($plk);
    $tytuly[$k + 1] = trim($p[1]);
}

if (isset($_GET['id']) && str_ievpifr($_GET['id'], 1, count($plks))) {
    $nr = $_GET['id'];
} else {
    $nr = 1;
}

$plk = file($plks[$nr - 1]);
$tytul = trim($plk[1]);

$plk[0] = '';
$plk[1] = '';
$tresc = trim(implode('', $plk));

$s = new Smarty();
$s->assign('menu',  $tytuly);
$s->assign('tytul', $tytul);
$s->assign('tresc', $tresc);

$strona = $s->fetch('index.tpl');
echo $strona;
```

Listing 23.6. *Projekt 23.3 pt. Fraszki — szablon index.tpl*

```
<html>
  <head>
    <title>Jan Kochanowski: Fraszki: {$tytul}</title>
  </head>
<body>
...
<ol>
{foreach from=$menu key=myId item=fraszka}
  <li><a href="index.php?id={$myId}">{$fraszka}</a></li>
{/foreach}
</ol>
...
<div id="tresc">
    <p>{$tresc|nl2br}</p>
</div>
...
</body>
</html>
```

Projekt 23.4. Liczba mnoga rzeczowników angielskich

Przygotuj witrynę prezentującą zestawienie metod tworzenia liczby mnogiej rzeczowników w języku angielskim. Dane projektu są zawarte w folderze *plural-nouns/*. Plik *00lista.log* zawiera listę różnych przypadków. Każdy wiersz tego pliku opisuje jedną metodę tworzenia liczby mnogiej. Pierwsze trzy wiersze pliku mają postać[3]:

```
&rarr;-s*plural-nouns/01.txt
&rarr;-es*plural-nouns/02.txt
-y&rarr;-ies*plural-nouns/03.txt
...
```

Separatorem w pliku jest znak *.

Pierwszy wiersz opisuje podstawową metodę tworzenia liczby mnogiej, dodanie końcówki *-s* (`&rarr;-s`). Przykłady rzeczowników tej grupy są zawarte w pliku *plural-nouns/01.txt*.

Drugi wiersz opisuje drugą metodę: dodawanie końcówki *-es* (`&rarr;-es`). Przykłady dla tej grupy są zawarte w pliku *plural-nouns/02.txt*.

Wreszcie trzeci wiersz pliku *00lista.log* opisuje grupę rzeczowników, w której końcówka *-y* zostaje zastąpiona końcówką *-ies* (`-y&rarr;-ies`). Przykłady takich rzeczowników są zapisane w pliku *plural-nouns/03.txt*.

I tak dalej.

Każdy z plików *01.txt*, *02.txt*, *03.txt*, ... ma tę samą strukturę. Na przykład plik *03.txt* zawiera:

```
spy:spies
poppy:poppies
penny:pennies
```

Są to kolejne przykłady tworzenia liczby mnogiej poprzez zamianę końcówki *–y* na końcówkę *–ies*. Separatorem w pliku jest znak dwukropka. Pierwszy wyraz jest w liczbie pojedynczej, zaś drugi — w liczbie mnogiej.

Na podstawie pliku *00lista.log* stwierdzamy, że witryna będzie stosowała następujące URL-e:

```
index.php?id=1    pierwszy rodzaj rzeczowników; plik:  01.txt; etykieta menu: &rarr;-s
index.php?id=2    drugi rodzaj rzeczowników; plik 02.txt; etykieta menu: &rarr;-es
index.php?id=3    trzeci rodzaj rzeczowników; plik 03.txt; etykieta menu: -y&rarr;-ies
itd.
```

[3] Encja `&rarr;` to strzałka w prawo.

Zatem w języku HTML menu przyjmie postać:

```
<ol id="menu">
<li><a href="index.php?id=1">&rarr;-s</a></li>
<li><a href="index.php?id=2">&rarr;-es</a></li>
<li><a href="index.php?id=3">-y&rarr;-ies</a></li>
...
</ol>
```

Listingi 23.7 oraz 23.8 przedstawiają rozwiązanie zadania, czyli skrypt *index.php* oraz szablon *index.tpl*.

Przetwarzanie w skrypcie PHP rozpoczynamy od pokrojenia pliku *00lista.log*. Następnie przeprowadzamy walidację zmiennej `$_GET['id']`. Po wykonaniu walidacji zmienna `$nr` zawiera indeks wybranego przypadku. Na podstawie numeru `$nr` odczytujemy i kroimy plik z przykładami dla danego przypadku:

```
$plurals = file_get_contents($menu['items'][$nr][1]);
$plurals = string2HArray($plurals, ':');
```

oraz ustalamy tytuł, który umieścimy w elemencie `h1`. Tytuł ten pochodzi z pokrojonego pliku *00lista.log*, z elementu o indeksie `$nr`:

```
$title = $menu['items'][$nr][0];
```

Trzy zmienne:

- `$menu` — pokrojony plik *00lista.log*,
- `$title` — etykietę wybranego przypadku,
- `$plurals` — przykłady wybranego przypadku

przekazujemy do szablonu pod tymi samymi nazwami, zaś przetworzony szablon wysyłamy do przeglądarki.

Szablon Smarty jest przedstawiony na listingu 23.8. Zmienna `$menu` jest wykorzystana do wygenerowania menu witryny:

```
<ol id="menu">
{section name=i loop=$menu.items}
<li><a
href="index.php?id={$smarty.section.i.iteration}">{$menu.items[i][0]}</a></li>
{/section}
</ol>
```

Tytuł pojawia się w elemencie `h1`:

```
<h1>{$title}</h1>
```

zaś tablica `$plurals` służy do wygenerowania tabeli z przykładowymi formami liczby mnogiej:

```
{section name=i loop=$plurals.items}
<tr>
    <td>{$plurals.items[i][0]}</td>
    <td>{$plurals.items[i][1]}</td>
</tr>
{/section}
```

Listing 23.7. *Projekt 23.4 pt. Liczba mnoga rzeczowników angielskich — skrypt index.php*

```
$p = file_get_contents('plural-nouns/00lista.log');
$menu = string2HArray($p, '*');

if (isset($_GET['id']) && str_ievpifr($_GET['id'], 1, $menu['rows'])) {
    $nr = $_GET['id'] - 1;
} else {
    $nr = 0;
}

$plurals = file_get_contents($menu['items'][$nr][1]);
$plurals = string2HArray($plurals, ':');

$title = $menu['items'][$nr][0];

$s = new Smarty;
$s->assign('menu', $menu);
$s->assign('title', $title);
$s->assign('plurals', $plurals);
$s->display('index.tpl');
```

Listing 23.8. *Projekt 23.4 pt. Liczba mnoga rzeczowników angielskich — szablon index.tpl*

```
<ol id="menu">
{section name=i loop=$menu.items}
<li><a
href="index.php?id={$smarty.section.i.iteration}">{$menu.items[i][0]}</a></li>
{/section}
</ol>
...
<h1>{$title}</h1>
...
<table>
<tr>
    <th>singular</th>
    <th>plural</th>
</tr>
{section name=i loop=$plurals.items}
<tr>
    <td>{$plurals.items[i][0]}</td>
    <td>{$plurals.items[i][1]}</td>
</tr>
{/section}
</table>
```

Projekt 23.5. Sortowanie danych względem kilku kolumn

Plik *dane.txt* zawiera dane osobowe: imię, nazwisko, wiek, płeć oraz staż pracy:

```
Jan:Nowak:78:M:43
Tomasz:Kwiecień:34:M:12
Anna:Jarocka:43:K:22
...
```

Napisz skrypt PHP, który przedstawi dane z pliku w postaci tabeli HTML. Zadanie rozwiąż w taki sposób, by dane można było sortować rosnąco oraz malejąco względem dowolnej kolumny.

Dane zawierają pięć kolumn, zaś każda kolumna może być uporządkowana na dwa sposoby: rosnąco lub malejąco. Zatem potrzebujemy dziesięciu różnych adresów URL. W projekcie 21.4 użyliśmy dwóch zmiennych URL. Jedna z nich decydowała o numerze kolumny, a druga — o tym, czy sortowanie ma być rosnące, czy malejące, na przykład:

```
ligi.php?liga=1&kolumna=5&order=asc
```

Tym razem użyjemy jednej zmiennej o nazwie `s`, która przyjmie wartości 1, –1, 2, –2, 3, –3 itd. Wartość 1:

```
index.php?s=1
```

oznacza, że dane należy posortować rosnąco względem pierwszej kolumny, zaś wartość –1:

```
index.php?s=-1
```

ustala porządek malejący względem pierwszej kolumny. Oto lista pierwszych sześciu adresów URL:

```
index.php?s=1      sortowanie rosnące wg pierwszej kolumny (imie)
index.php?s=-1     sortowanie malejące wg pierwszej kolumny (imie)
index.php?s=2      sortowanie rosnące wg drugiej kolumny (nazwisko)
index.php?s=-2     sortowanie malejące wg drugiej kolumny (nazwisko)
index.php?s=3      sortowanie rosnące wg trzeciej kolumny (wiek)
index.php?s=-3     sortowanie malejące wg trzeciej kolumny (wiek)
```

Powyższe adresy będą użyte w komórkach nagłówkowych tabeli HTML. Kolumna pierwsza będzie zawierała nagłówek:

```
<th><a href="sortuj.php?s=1">Imię</a></th>
```

Po kliknięciu powyższego hiperłącza zostanie wydrukowana strona WWW zawierająca dane posortowane rosnąco względem imion. Hiperłącze zawarte w nagłówku zostanie zamienione na:

```
<th><a href="sortuj.php?s=-1">Imię</a></th>
```

tak, by ponowne kliknięcie nagłówka kolumny *Imię* spowodowało posortowanie malejące względem imion.

Rozwiązanie zadania jest przedstawione na listingach 23.9 oraz 23.10. Skrypt *index.php* z listingu 23.9 rozpoczyna się od zdefiniowania tablicy `$kolumny`. Zawiera ona etykiety wszystkich kolumn oraz wartości zmiennej `s`. Dzięki podaniu indeksu pierwszego elementu tablicy (tj. `1 =>`) indeksacja elementów rozpocznie się od 1. Tablica ta zostanie wykorzystana do wygenerowania hiperłączy zawartych w komórkach nagłówkowych tabeli.

Następnie tworzymy tablicę `$wartosci_url`, którą wykorzystamy do sprawdzenia poprawności zmiennej `$_GET['s']`. Walidacja wartości zmiennej `s` polega na stwierdzeniu, że taka zmienna istnieje (funkcja `isset()`) oraz że jest to jedna z dopuszczalnych wartości, zawartych w utworzonej wcześniej tablicy `$wartosci_url`:

```
if (isset($_GET['s']) && in_array($_GET['s'], $wartosci_url)) {
    ...
} else {
    ...
}
```

Jeśli zmienna `s` jest poprawna, to w zmiennej `$kryterium` zapamiętujemy jej wartość, po czym za pomocą podanych instrukcji:

```
$indeks = abs($kryterium);
$kolumny[$indeks]['sortowanie'] = -$kryterium;
```

przestawiamy porządek sortowania wybranej kolumny na przeciwny. Dzięki temu na stronie prezentującej dane posortowane rosnąco (np. `index.php?s=3`) pojawi się hiperłącze odsyłające do strony sortującej malejąco (np. `index.php?s=-3`).

W przypadku gdy zmienna `s` nie jest podana lub jest niepoprawna, ustalamy domyślne sortowanie rosnące względem nazwisk (tj. kolumny drugiej). W takim przypadku hiperłącze w nagłówku kolumny drugiej musi wskazywać adres `index.php?s=-2`:

```
$kryterium = 2;
$kolumny[2]['sortowanie'] = -2;
```

W następnym kroku odczytujemy i kroimy plik danych. Pokrojoną dwuwymiarową tablicę przypisujemy do zmiennej `$dane`. Następująca potem instrukcja `switch`, sterowana zmienną `$kryterium`, sortuje tablicę `$dane` względem wybranego kryterium. Na zakończenie do szablonu przekazujemy posortowane dane (zmienna `$dane`) oraz tablicę, na podstawie której powstaną komórki nagłówkowe (zmienna `$kolumny`).

Szablon przedstawiony na listingu 23.10 zawiera pętle `foreach` oraz pętlę `section`. Pierwsza z nich generuje komórki nagłówkowe tabeli, a druga — umieszcza w tabeli posortowane dane. Zauważ, że hiperłącza zawarte w tabeli wykorzystują dwie składowe elementów tablicy `$kolumny`. Składowymi tablicy `$kolumny` są tablice asocjacyjne o indeksach `sortowanie` i `etykieta`:

```
array(
    'sortowanie' => 2,
    'etykieta' => 'Nazwisko'
)
```

Dzięki temu w pętli `foreach` możemy użyć zmiennych `$kolumna.sortowanie` oraz `$kolumna.etykieta`:

```
<a href="sortuj.php?s={$kolumna.sortowanie}">{$kolumna.etykieta}</a>
```

Listing 23.9. *Projekt 23.5 pt. Sortowanie danych względem kilku kolumn — skrypt index.php*

```
$kolumny = array(
    1 => array(
        'sortowanie' => 1,
        'etykieta' => 'Imię'
    ),
    array(
        'sortowanie' => 2,
        'etykieta' => 'Nazwisko'
```

```
    ),
    ...
);

$wartosci_url = array(1, -1, 2, -2, 3, -3, 4, -4, 5, -5);

if (isset($_GET['s']) && in_array($_GET['s'], $wartosci_url)) {
    $kryterium = (int)$_GET['s'];
    $indeks = abs($kryterium);
    $kolumny[$indeks]['sortowanie'] = -$kryterium;
} else {
    $kryterium = 2;
    $kolumny[2]['sortowanie'] = -2;
}

$p = file_get_contents('dane.txt');
$tmp = string2VArray($p, ':');
$dane = $tmp['items'];

switch ($kryterium) {
    case 1:
        array_multisort(
            $dane[0], SORT_ASC,  SORT_STRING,
            $dane[1], SORT_ASC,  SORT_STRING,
            $dane[2], SORT_ASC,  SORT_NUMERIC,
            $dane[3], SORT_ASC,  SORT_STRING,
            $dane[4], SORT_ASC,  SORT_NUMERIC
        );
    break;

    case -1:
        array_multisort(
            $dane[0], SORT_DESC, SORT_STRING,
            $dane[1], SORT_ASC,  SORT_STRING,
            $dane[2], SORT_ASC,  SORT_NUMERIC,
            $dane[3], SORT_ASC,  SORT_STRING,
            $dane[4], SORT_ASC,  SORT_NUMERIC
        );
    break;
    ...
}

$smarty  =  new Smarty();
$smarty->assign('dane', $dane);
$smarty->assign('kolumny', $kolumny);
$smarty->display('szablon.tpl');
```

Listing 23.10. *Projekt 23.5 pt. Sortowanie danych względem kilku kolumn — szablon szablon.tpl*

```
<table>

<tr>
  <th>lp.</th>
{foreach from=$kolumny item=kolumna}
  <th><a href="sortuj.php?s={$kolumna.sortowanie}">{$kolumna.etykieta}</a></th>
```

```
{/foreach}
</tr>

{section name=w loop=$dane[0]}
<tr>
  <td>{$smarty.section.w.iteration}.</td>
  <td>{$dane[0][w]}</td>
  <td>{$dane[1][w]}</td>
  <td>{$dane[2][w]}</td>
  <td>{$dane[3][w]}</td>
  <td>{$dane[4][w]}</td>
</tr>
{/section}

</table>
```

Projekt 23.6. Fotogaleria

Folder *foto/* zawiera foldery *200/*, *400/*, *600/*, *800/* oraz *max/*. W każdym z nich umieszczono te same zdjęcia w różnych rozdzielczościach. Fotografia *foto-01.jpg* zawarta w folderze *200/* ma szerokość 200 pikseli. To samo zdjęcie w większych wymiarach jest zapisane w folderach *400/*, *600/*, *800/* oraz *max/*:

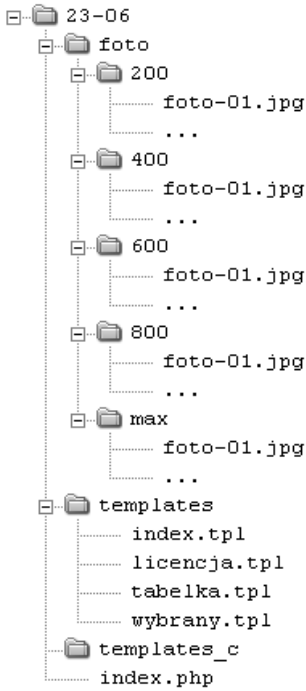
```
foto/400/foto-01.jpg – zdjęcie o szerokości 400 pikseli
foto/600/foto-01.jpg – zdjęcie o szerokości 600 pikseli
foto/800/foto-01.jpg – zdjęcie o szerokości 800 pikseli
foto/max/foto-01.jpg – zdjęcie o szerokości 4288 pikseli
```

Napisz skrypt PHP, który zdjęcia z folderu foto przedstawi w postaci wygodnej do przeglądania strony WWW. Na stronie głównej umieść tabelę miniatur wszystkich zdjęć. Serwis wzbogać o licencję wyjaśniającą warunki wykorzystywania zdjęć.

Opisywany skrypt będzie stosował następujące adresy URL:

- `index.php` — strona główna prezentująca tabelkę miniatur,
- `index.php?id=1` — strona prezentująca pierwsze zdjęcie,
- `index.php?id=2` — strona prezentująca drugie zdjęcie,
- `index.php?id=3` — strona prezentująca trzecie zdjęcie,
- ...
- `index.php?id2=1` — strona prezentująca licencję.

Rozwiązanie zadania składa się ze skryptu *index.php* oraz czterech plików *index.tpl*, *licencja.tpl*, *tabelka.tpl* i *wybrany.tpl*. Drzewo katalogów rozwiązania jest przedstawione na rysunku 23.1.



Rysunek 23.1. *Struktura folderów w projekcie 23.6*

Skrypt *index.php* jest przedstawiony na listingu 23.11. Został on podzielony na cztery etapy: inicjalizację, walidację, przekazanie danych do szablonu oraz przetworzenie szablonu. Inicjalizacja polega na dołączeniu bibliotek i przypisaniu domyślnych wartości wszystkim zmiennym. Zmienna `$akcja` będzie sterowała przebiegiem przetwarzania na etapie trzecim. Jej domyślną wartością jest `404` — wartość powodująca wyświetlenie komunikatu *Błąd! Podana strona nie istnieje!* Oprócz tego podczas inicjalizacji wyszukujemy wszystkie pliki o rozszerzeniu *.jpg* z folderu *foto/200/*. Nazwy znalezionych plików[4] zostają zapamiętane w tablicy asocjacyjnej `$indeksy`.

Funkcja `array_push()` występująca w inicjalizacji powoduje dodanie elementu na końcu tablicy. Wywołanie:

```
array_push($indeksy, $pozycja);
```

jest równoważne:

```
$indeksy[] = $pozycja;
```

Drugi etap, walidacja, ma za zadanie ustalenie, która podstrona serwisu została odwiedzona. Jeśli użyto adresu *index.php* (tj. `empty($_GET)` zwraca logiczną prawdę), to do zmiennej `$akcja` przypisujemy wartość `tabelka`. Jeśli podana jest zmienna `id` o poprawnej wartości, to `$akcja` przyjmuje wartość `fotka`, zaś `$wybrany` numer wybranej fotografii. Jeśli

[4] Nazwy znajdujące się w tablicy `$indeksy` nie zawierają nazw folderów, tylko nazwy plików. Nazwy folderów usuwamy, wywołując funkcję `basename()`.

natomiast podana została zmienna `id1` o wartości 1, to zmiennej `$akcja` przypisujemy wartość `licencja`. Po zakończonej walidacji zmienna `$akcja` przyjmuje jedną z czterech wartości: `404`, `tabelka`, `fotka` lub `licencja`. Na podstawie tych wartości na etapie trzecim do szablonu przekazujemy odpowiednie dane.

Jeśli `$akcja` ma wartość `404`, to do przeglądarki wysyłamy nagłówek HTTP:

```
header('HTTP/1.x 404 Not Found');
```

informujący o tym, że taka strona nie istnieje. Jeśli wartością zmiennej `$akcja` jest `tabelka`, to do szablonu trafia tablica `$indeksy` przygotowana na etapie walidacji:

```
$s->assign('mini', $indeksy);
```

W trzecim przypadku, czyli gdy wybrana została konkretna fotografia, do szablonu przekazujemy jej numer:

```
$s->assign('wybrany', $wybrany);
```

nazwę pliku graficznego:

```
$s->assign('obraz_filename', $obraz_filename);
```

numer następnej i poprzedniej fotografii:

```
$s->assign('next', $next);
$s->assign('previous', $previous);
```

oraz liczbę zdjęć:

```
$s->assign('liczba_ilustracji', $pliki_count);
```

Zwróć uwagę, że jeśli wybrana jest pierwsza fotografia, to jako numer poprzedniego zdjęcia przekazujemy wartość logiczną `false`. Podobnie: jeśli wybrano ostatnią fotografię, to zmienna `next` przekazana do szablonu przyjmie wartość `false`.

Ostatni przypadek, czyli wyświetlenie licencji, nie wymaga wykonania żadnych akcji, gdyż treść licencji jest zawarta w szablonie.

Etap czwarty, czyli przetworzenie szablonu, sprowadza się do wywołania funkcji `display()`.

Szablon rozwiązania został podzielony na cztery osobne pliki *.tpl*. Plikiem głównym jest przedstawiony na listingu 23.12 plik *index.tpl*. Zawarta w nim wieloczłonowa instrukcja `if` steruje dołączaniem kolejnych plików. Jeśli wartością zmiennej `$akcja` jest `404`, wówczas szablon będzie zawierał komunikat:

```
<p class="e404">Błąd! Podana strona nie istnieje!</p>
```

W przeciwnym razie, jeśli `$akcja` ma wartość `tabelka`, to do szablonu dołączymy plik *tabelka.tpl*:

```
{include file="tabelka.tpl"}
```

Kolejnym przypadkiem jest wartość `fotka`. Tym razem dołączamy plik *wybrany.tpl*:

```
{include file="wybrany.tpl"}
```

Jeśli natomiast wartością zmiennej `$akcja` jest `licencja`, to do szablonu dołączamy plik *licencja.tpl*:

```
{include file="licencja.tpl"}
```

Funkcja Smarty `{include}` powoduje dołączenie w miejscu wywołania pliku, którego nazwa jest podana w parametrze `file`. Dołączany szablon ma pełny dostęp do zmiennych zawartych w szablonie. Zatem w przedstawionym na listingu 23.13 pliku *tabelka.tpl* możemy wykorzystywać zmienną `$mini` przekazaną do szablonu na etapie trzecim skryptu *index.php*:

```
case 'tabelka':
    $s->assign('mini', $indeksy);
    break;
```

Szablon *tabelka.tpl* powoduje wyświetlenie tabeli miniaturowych zdjęć o szerokości 200 pikseli.

Kolejny listing, 23.14, prezentuje fragment szablonu *wybrany.tpl*. Szablon ten jest przetwarzany na stronie prezentującej szczegółowe dane wybranej fotografii. Na stronie przedstawiamy zdjęcie o szerokości 600 pikseli:

```
<img id="obraz" src="foto/600/{$obraz_filename}" alt="" />
```

oraz wskaźnik pozwalający na wygodne przejście do następnej lub poprzedniej fotografii. W szablonie tym wykorzystujemy zmienne `{$next}`, `{$previous}`, `{$obraz_filename}`, `{$wybrany}` i `{$liczba_ilustracji}`:

```
$s->assign('next',     $next);
$s->assign('previous', $previous);
$s->assign('obraz_filename', $obraz_filename);
$s->assign('wybrany', $wybrany);
$s->assign('liczba_ilustracji',  $pliki_count);
```

Zmienna `$next` przyjmuje wartość `false` w przypadku, gdy wybrana jest ostatnia fotografia. Wówczas na stronie nie pojawia się hiperłącze do następnej fotografii, a jedynie wyblakła ikonka informująca o tym, że dotarliśmy do ostatniego zdjęcia. Ikonka ta jest zawarta w pliku *next-brak.png*. O tym, czy wydrukować należy hiperłącze, czy wyblakłą ikonę, decyduje funkcja `{if}`:

```
{if $next}
  <li>
    <a title="Następna ilustracja" href="index.php?id={$next}">
      <img src="img/next.png" alt="Następna ilustracja" />
    </a>
  </li>
{else}
  <li><img src="img/next-brak.png" alt="Następny" /></li>
{/if}
```

Listing 23.11. *Projekt 23.6 pt. Fotogaleria — skrypt index.php*

```
//ETAP I: INICJALIZACJA
require_once 'Smarty.class.php';
require_once 'walidacja.inc.php';
$s = new Smarty();
```

```
$akcja = '404';
$wybrany = false;

$pliki  = glob('foto/200/*.jpg');
$pliki_count = count($pliki);

$indeksy = array();
foreach ($pliki as $k => $p) {
    $pozycja = array(
        'filename' => basename($p),
        'id' => $k + 1
    );
    array_push($indeksy, $pozycja);
}

//ETAP II: WALIDACJA
if (empty($_GET)) {
    $akcja = 'tabelka';
} else if (
    (count($_GET) == 1) &&
    isset($_GET['id']) &&
    str_ievpifr($_GET['id'], 1, $pliki_count)
) {
        $akcja = 'fotka';
        $wybrany = $_GET['id'];
} else if (
    (count($_GET) == 1) &&
    isset($_GET['id2']) &&
    ($_GET['id2'] == 1)
) {
        $akcja = 'licencja';
}

//ETAP III: PRZEKAZANIE DANYCH DO SZABLONU
$s->assign('akcja', $akcja);
switch ($akcja) {
case '404':
    header('HTTP/1.x 404 Not Found');
    break;

case 'tabelka':
    $s->assign('mini', $indeksy);
    break;

case 'fotka':
    $next = $wybrany + 1;


    if (!str_ievpifr((string)$next, 1, $pliki_count)) {
        $next = false;
    };

    $previous = $wybrany - 1;

    if (!str_ievpifr((string)$previous, 1, $pliki_count)) {
```

```
        $previous = false;
    };

    $obraz_filename = basename($pliki[$wybrany - 1]);

    $s->assign('next',     $next);
    $s->assign('previous', $previous);
    $s->assign('obraz_filename', $obraz_filename);
    $s->assign('wybrany', $wybrany);
    $s->assign('liczba_ilustracji',  $pliki_count);

    break;

case 'licencja':
    break;
}

//ETAP IV: PRZETWORZENIE SZABLONU
$s->display('index.tpl');
```

Listing 23.12. *Projekt 23.6 pt. Fotogaleria — skrypt index.tpl*

```
<body>
<div id="pojemnik">
        <h1>FOTO.GAJDAW.PL</h1>
        <ul id="navbar">
            <li><a href="index.php">Fotografie</a></li>
            <li><a href="index.php?id2=1">Licencja</a></li>
        </ul>
        {if $akcja == '404'}
            <p class="e404">Błąd! Podana strona nie istnieje!</p>
        {elseif $akcja == 'tabelka'}
            {include file="tabelka.tpl"}
        {elseif $akcja == 'fotka'}
            {include file="wybrany.tpl"}
        {elseif $akcja == 'licencja'}
            {include file="licencja.tpl"}
        {/if}
</div>
</body>
```

Listing 23.13. *Projekt 23.6 pt. Fotogaleria — szablon tabelka.tpl*

```
<ul class="fotki">
  {foreach from=$mini item=fotka}
    <li>
      <a href="index.php?id={$fotka.id}">
        <img src="foto/200/{$fotka.filename}" alt="" />
      </a>
    </li>
  {/foreach}
</ul>
```

Listing 23.14. *Projekt 23.6 pt. Fotogaleria — szablon wybrany.tpl*

```
<img id="obraz" src="foto/600/{$obraz_filename}" alt="" />

<ul class="wskaznik">
  {if $previous}
    <li>
      <a title="Pierwsza ilustracja" href="index.php?id=1">
        <img src="img/first.png" alt="Pierwsza ilustracja" />
      </a>
    </li>
  {else}
    <li>
      <img src="img/first-brak.png" alt="Pierwsza ilustracja" />
    </li>
  {/if}

...

</ul>
```

Projekt 23.7. Witryna GIMP w zastosowaniach

Przygotuj witrynę prezentującą szczegółowe informacje o książce pt. *GIMP w zastosowaniach*. Dane do wykonania projektu są zawarte w plikach tekstowych, plikach graficznych oraz plikach *.zip*.

Menu witryny wykonaj na bazie pliku *dane/menu.txt* o zawartości:

```
BRAK*dane/teksty/menu_404.txt*0*Błąd
OD AUTORA*dane/teksty/menu_od-autora.txt*1*Od autora
PODZIĘKOWANIA*dane/teksty/menu_podziekowania.txt*1*Podziękowania
...
SPIS TREŚCI*dane/teksty/menu_skrocony-spis-tresci.txt*0*Spis treści
```

Każdy wiersz pliku opisuje jedną opcję menu. Kolejnymi kolumnami są:

- etykieta, którą należy umieścić w menu,
- nazwa pliku tekstowego z treścią, którą należy wyświetlić na danej stronie,
- informacja logiczna: czy dany wiersz powinien być widoczny w menu,
- oraz tytuł strony.

Wiersz:

```
OD AUTORA*dane/teksty/menu_od-autora.txt*1*Od autora
```

mówi o tym, że w menu wystąpi pozycja *OD AUTORA*, która będzie powodowała wyświetlenie treści z pliku *dane/taksty/menu_od-autora.txt*.

Ćwiczenia opisane w książce są wymienione w pliku *dane/cwiczenia.txt*, którego początkowe wiersze są takie jak poniżej:

```
02-01-01*jpg*DANE*11*2*1*1*
02-01-02*jpg*DANE*13*2*1*2*
02-01-03*jpg*DANE*14*2*1*3*
...
```

Kolejnymi kolumnami są:

- nazwa pliku graficznego,
- rozszerzenie,
- informacja o tym, czy w projekcie wykorzystano jakieś dane,
- numer strony w książce, na której występuje dane ćwiczenie,
- numer rozdziału,
- numer podrozdziału
- oraz numer ćwiczenia.

Wiersz:

```
02-01-03*jpg*DANE*14*2*1*3*
```

informuje o tym, że na stronie 14. w książce opisane zostało ćwiczenie numer 3 z rozdziału 2. z podrozdziału 1. Ilustracja do ćwiczenia jest zawarta w plikach:

```
images/01-01-03.jpg
images/01-01-03_small.jpg
```

Rozwiązanie ćwiczenia znajduje się w pliku:

```
zip/02-01-03-rozw.zip
```

zaś dane do wykonania ćwiczenia w pliku:

```
zip/02-01-03-dane.zip
```

Ostatni z plików zawartych w folderze *dane/*, plik o nazwie *dane/rozdzialy.txt*, zawiera zestawienie wszystkich rozdziałów i podrozdziałów. Jego początkowe wiersze są następujące:

```
1. Wprowadzenie*5*1***R*4*0
1.1. Instalacja programu*6*1*1**P*0*0
1.2. Interfejs programu*6*1*2**P*0*0
1.3. Ćwiczenia*8*1*3**P*0*0
1.4. Podziękowania*9*1*4**P*0*0
2. Podstawy pracy z GIMP-em*11*2***R*4*
...
```

Separatorem jest znak *, zaś kolejne kolumny to:

- tytuł rozdziału lub podrozdziału poprzedzony numerem,
- strona w książce, na której rozdział się rozpoczyna,
- numer rozdziału,
- numer podrozdziału,

- skrócony tytuł (jeśli pusty — to identyczny z tytułem z pierwszej kolumny),
- litera R (rozdział) lub P (podrozdział)
- oraz liczba podrozdziałów.

Wiersz:

```
2. Podstawy pracy z GIMP-em*11*2***R*4*
```

opisuje rozdział (litera R) o tytule *2. Podstawy pracy z GIMP-em*. Rozdział ten rozpoczyna się na stronie 11., ma numer 2 i zawiera cztery podrozdziały. Natomiast wiersz:

```
5.2. Modyfikacja kształtu selekcji*53*5*2**P*0*
```

opisuje podrozdział (litera P) zatytułowany *5.2. Modyfikacja kształtu selekcji*. Podrozdział ten rozpoczyna się na stronie 53. w rozdziale o numerze 5 i ma numer 2.

Opisywany projekt możesz obejrzeć w internecie pod adresem *http://gwz.gajdaw.pl*.

Witryna GIMP w zastosowaniach wykorzystuje następujące adresy URL:

- `index.php?m=X` — opcje menu,
- `index.php?rozdz=X` — lista ćwiczeń z rozdziału,
- `index.php?rozdz=X&podrozdz=Y` — lista ćwiczeń z podrozdziału,
- `index.php?id=X` — ćwiczenie X.

Adresy wskazujące opcje menu stosują jako identyfikatory numery wierszy w pliku *dane/menu.txt*. Na przykład adres `index.php?id=1` wskazuje stronę *OD AUTORA*, zaś `index.php?m=2` stronę *PODZIĘKOWANIA*. Numeracja tym razem rozpoczyna się od 0.

Rozwiązanie zadania zostało podzielone na następujące elementy:

- Kontroler *index.php*, który przetwarza żądania HTTP.
- Szablony *layout.tpl*, *tresc.tpl* oraz *cwiczenie.tpl* (w folderze *templates/*), które odpowiadają za prezentację wyników w postaci dokumentów HTML.
- Klasy pomocnicze `Menu`, `Rozdzialy`, `Rozdzial`, `Cwiczenia`, `Cwiczenie` zawarte w pliku *include/klasy.inc.php*. Klasy te odpowiadają za krojenie plików *dane/menu.txt*, *dane/rozdzialy.txt* oraz *dane/cwiczenia.txt*.

Kontroler *index.php* został przedstawiony na listingu 23.15. Jest on podzielony na cztery fragmenty:

- inicjalizację,
- walidację,
- przekazanie danych do szablonu
- oraz przetworzenie szablonu.

Zmienna `$akcja`, utworzona na etapie inicjalizacji, decyduje o tym, która z podstron serwisu zostanie wyświetlona. Domyślnie jest to strona błędu 404 z komunikatem *Podana strona nie istnieje*. Jeśli walidacja przebiegnie poprawnie, tzn. wybrany adres jest:

- adresem jednej ze stron menu (np. `index.php?m=3`),
- adresem strony rozdziału (np. `index.php?rozdz=5`),
- adresem strony podrozdziału (np. `index.php?rozdz=5&podr=2`)
- lub adresem strony ćwiczenia (np. `index.php?id=69`),

to zmienna `$akcja` przyjmie jedną z wartości: `menu`, `rozdzial`, `podrozdzial`, `cwiczenie`.

Na trzecim etapie, na podstawie wartości zmiennej `$akcja`, odczytujemy odpowiednie dane i przekazujemy je do szablonu. Zarówno walidacja, jak i odczytanie danych wykorzystują klasy `Menu`, `Rozdzialy`, `Rozdzial`, `Cwiczenia`, `Cwiczenie`. Bezpośrednio w skrypcie *index.php* nie pojawia się ani odczyt ani krojenie plików tekstowych. Na zakończenie szablon jest przetwarzany i wysyłany do przeglądarki.

Zarys szablonu *layout.tpl* został przedstawiony na listingu 23.16. Zawiera on menu pionowe oraz pojemnik, w którym umieszczamy treść witryny. Menu pionowe powstaje na podstawie pokrojonego pliku *dane/menu.txt*. Zawiera ono tylko te opcje, które w przedostatniej kolumnie mają wartość 1, na przykład:

```
METRYKA*dane/teksty/menu_metryka.txt*1*Metryka
```

Treść podstrony jest formatowana przez szablon *tresc.tpl*, o czym decyduje funkcja `{include}`:

```
<div id="tresc">
  {include file="tresc.tpl"}
</div>
```

Formatowanie w pliku *tresc.tpl* odbywa się na podstawie wartości zmiennej `$akcja`. Widoczna na listingu 23.17 funkcja `{if}` wybiera odpowiedni przypadek i wstawia kod HTML oraz zmienne szablonu.

Listing 23.15. *Projekt 23.7 pt. GIMP w zastosowaniach — kontroler index.php*

```
// ETAP PIERWSZY: INICJALIZACJA
require_once 'Smarty.class.php';
require_once 'include/klasy.inc.php';

$smarty =  new Smarty();

if (empty($_GET)) {
    $_GET['m'] = '1';
}

$tutulStrony = '';

$menu        = new Menu();
$rozdzialy   = new Rozdzialy();
$cwiczenia   = new Cwiczenia();
```

```
$akcja          = '404';
$rozdzial    = false;
$podrozdzial = false;
$cwiczenie   = false;


// ETAP DRUGI: WALIDACJA
if (
    isset($_GET['m']) &&
    $menu->isValidMenuId($_GET['m'])
) {

    $akcja = 'menu';
    $menu->setId($_GET['m']);

} else if (
    isset($_GET['rozdz']) &&
    $rozdzialy->isValidRozdzialId($_GET['rozdz'])
) {

    $akcja = 'rozdzial';
    $rozdzial = $rozdzialy->getRozdzial($_GET['rozdz']);
    $rozdzial->updatePodrozdzialy($rozdzialy);

    if (
        isset($_GET['podr']) &&
        $rozdzial->isValidPodRozdzialId($_GET['rozdz'])
    ) {

        $akcja = 'podrozdzial';
        $podrozdzial = $rozdzialy->getPodRozdzial($_GET['rozdz'], $_GET['podr']);

    }

} else if (
    isset($_GET['id']) &&
    $cwiczenia->isValidCwiczenieId($_GET['id'])
) {

    $akcja = 'cwiczenie';
    $cwiczenie = $cwiczenia->getCwiczenie($_GET['id']);

}

// ETAP TRZECI: PRZEKAZANIE DANYCH DO SZABLONU
switch ($akcja) {

case '404':
    $menu->setId(0);
    $tutulStrony = $menu->getTitle();
    $smarty->assign('trescStrony', $menu->getContents());
    header('HTTP/1.x 404 Not Found');
    break;

case 'menu':
    $tutulStrony = $menu->getTitle();
    $smarty->assign('trescStrony', $menu->getContents());
    break;
```

```
case 'rozdzial':
    $rozdzial->updateCwiczenia($cwiczenia);
    $smarty->assign('rozdzial', $rozdzial);
    preg_match('/^[0-9]+\.(.*)$/', $rozdzial->FTytulSkrocony, $regs);
    $tutulStrony =  $regs[1];
    break;

case 'podrozdzial':
    $podrozdzial->updateCwiczenia($cwiczenia);
    $smarty->assign('rozdzial',    $rozdzial);
    $smarty->assign('podrozdzial', $podrozdzial);
    preg_match('/^[0-9]+\.([0-9]+\.)?(.*)$/', $podrozdzial->FTytulSkrocony, $regs);
    $tutulStrony =  $regs[2];
    break;

case 'cwiczenie':
    $smarty->assign('cwiczenie', $cwiczenie);
    $rozdzial = $rozdzialy->getRozdzial($cwiczenie->FRozdzial);
    $rozdzial->updatePodrozdzialy($rozdzialy);
    $smarty->assign('rozdzial', $rozdzial);

    if ($rozdzial->isValidPodRozdzialId($cwiczenie->FPodRozdzial)) {
        $podrozdzial = $rozdzialy->getPodRozdzial(
            $cwiczenie->FRozdzial,
            $cwiczenie->FPodRozdzial
        );
        $podrozdzial->updateCwiczenia($cwiczenia);
        $smarty->assign('podrozdzial', $podrozdzial);
    }
    $tutulStrony =  'Ćwiczenie ' . $cwiczenie->FPelnyNumer;
    break;

}

// ETAP CZWARTY: PRZETWORZENIE SZABLONU
$smarty->assign('akcja', $akcja);
$smarty->assign('menu', $menu->getVisible());
$smarty->assign('tytulStrony', 'GIMP w zastosowaniach: ' . $tutulStrony);
$smarty->display('layout.tpl');
```

Listing 23.16. *Projekt 23.7 pt. GIMP w zastosowaniach — szablon layout.tpl*

```
    <div id="menuPion">
      <ul>
        {section name=i loop=$menu}
          {if $menu[i][2]}
            <li>
              <a href="index.php?m={$menu[i][4]}">
                {$menu[i][0]} <span>&spades;</span>
              </a>
            </li>
          {/if}
        {/section}
      </ul>
    </div>
...
```

```
    <div id="tresc">
      {include file="tresc.tpl"}
    </div>
```

Listing 23.17. *Projekt 23.7 pt. GIMP w zastosowaniach — szablon tresc.tpl*

```
{if $akcja == 'cwiczenie'}
    {include file="cwiczenie.tpl"}
{elseif $akcja == 'rozdzial'}
    <h3>{$rozdzial->FTytulPelny}</h3>
    {if $rozdzial->FTrescCwiczen}
        {$rozdzial->FTrescCwiczen}
    {else}
        <p>W rozdziale tym nie są omówione żadne ćwiczenia praktyczne.</p>
    {/if}
{elseif $akcja == 'podrozdzial'}
    <h3>{$rozdzial->FTytulPelny}</h3>
    <h3>{$podrozdzial->FTytulPelny}</h3>
    {if $podrozdzial->FTrescCwiczen}
        {$podrozdzial->FTrescCwiczen}
    {else}
        <p>W rozdziale tym nie są omówione żadne ćwiczenia praktyczne.</p>
    {/if}
{else}
    {$trescStrony}
{/if}
```

Czego powinieneś nauczyć się z tego rozdziału?

Opanowanie plików tekstowych, adresów URL oraz szablonów Smarty pozwala tworzyć ciekawe i całkiem spore witryny. Projekty takie mogą nawet być wykorzystywane w praktyce do realizacji komercyjnych zleceń. Dowodem tego jest projekt 23.7, który wykorzystałem do opublikowania ćwiczeń omówionych w mojej książce.

Najważniejszą cechą opisanych w tym rozdziale projektów jest to, że zbliżają Cię one do architektury MVC. Na razie są to jeszcze rozwiązania dość prymitywne, jednak wyznaczają zasadniczy kierunek. Podział na takie elementy, jak: kontroler, czyli skrypt przetwarzający żądania HTTP, szablony nadające wizualny format danych oraz funkcje i klasy pobierające informacje (odpowiedniki klas `Menu`, `Cwiczenie`, `Cwiczenia`, `Rozdzial` i `Rozdzialy` z projektu 23.7), zarysuje się jeszcze wyraźniej, gdy zaczniemy wykorzystywać bazy danych. Można powiedzieć, że zasadnicza część pracy jest już za nami, pozostaje rozszerzenie projektów o bazy danych oraz wykorzystanie technik programowania obiektowego.