Formularze.

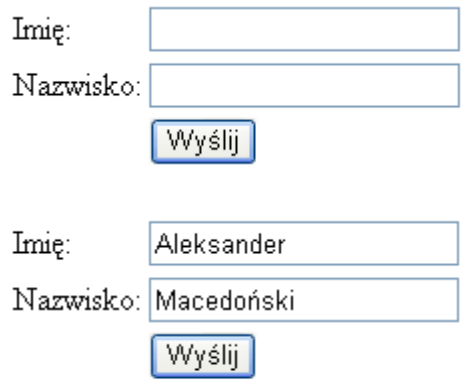
1. Formularz HTML

Formularz HTML definiujemy stosując element FORM. Wewnątrz, po między znacznikami <FORM> oraz </FORM> umieszczamy zawartość formularza, na którą składają się kontrolki (np. INPUT) oraz elementy formatujące (np. TABLE). Typowy formularz składa się z elementu FORM zawierającego tabelę, wewnątrz której umieszczono kilka kontrolek. Listing 1 przedstawia przykładowy formularz.

```
<FORM action="jakis-skrypt.php">
<TABLE>
<TR>
  <TD>Imię:</TD>
  <TD><INPUT name="imie"></TD>
</TR>
<TR>
  <TD>Nazwisko:</TD>
  <TD><INPUT name="nazwisko"></TD>
</TR>
<TR>
  <TD> </TD>
  <TD><INPUT type="submit" value="Wyślij"></TD>
</TR>
</TABLE>
</FORM>
```

Listing 1. Przykładowy formularz.

Zawiera on dwa pola do wprowadzania danych (pola te nazwano *Imię* i *Nazwisko*) oraz przycisk Wyślij. Osoba odwiedzająca witrynę może umieścić kursor wewnątrz pól formularza, wypełnić je, wpisując napisy *Aleksander Macedoński*, po czym przesłać formularz, naciskając przycisk Wyślij. Treść wprowadzona przez użytkownika zostanie przesłana do skryptu o nazwie jakis-skrypt.php. Nazwę skryptu przetwarzającego formularz podajemy jako wartość atrybutu action elementu FORM.



Rysunek 1. Wygląd formularza z listingu 1 przed i po wprowadzeniu danych

Po naciśnięciu przycisku Wyślij, wizyta zostanie przeniesiona pod adres jakis-skrypt.php. W skrypcie tym będą dostępne dane wprowadzone przez użytkownika w formularzu.

Skrypt przetwarzający formularz zawarty w pliku jakis-skrypt.php może być napisany w dowolnym języku programowania dynamicznych stron WWW. Może to być PHP, Perl, ASP, JavaServerPages, skrypty CGI w bashu, C, czy nawet Pascalu. Jednakże trzeba od początku jasno podkreślić, że nie ma możliwości przetworzenia formularza w języku HTML. **Do przetwarzania formularza musimy użyć jednego z języków skryptowych, służących do programowania dynamicznych stron WWW.**

Formularze tworzymy w języku HTML, stosując między innymi elementy FORM oraz INPUT. Natomiast przetwarzanie formularza wykonuje skrypt napisany na przykład w jednym z języków PHP, ASP lub Perl i umieszczony na serwerze.

Zatem korzystanie z formularzy wymaga znajomości zarówno języka HTML jak i języka skryptowego. Język HTML zajmuje się jedynie wyglądem zewnętrznym formularza. Stosując elementy HTML układamy zawartość formularza na stronie oraz ustalamy adres URL skryptu, który będzie zajmował się przetworzeniem danych pochodzących z formularza.

Jeśli formularz przedstawiony na listingu 1 zapiszemy do pliku formularz.html, wówczas cały przykład będzie się składał z dwóch plików. Pierwszym plikiem jest plik formularz.html zawierający kod HTML formularza, zaś drugim plikiem będzie jakis-skrypt.php. Pamiętajmy, że nazwa pliku zawierającego skrypt musi być dokładnie taka, jak wartość atrybutu action formularza.

2. Kodowanie application/x-www-form-urlencoded

Formularz jest przedstawiany w oknie przeglądarki w postaci szeregu kontrolek. Układ graficzny kontrolek nie wpływa na sposób zakodowania danych. Dane wprowadzone do formularza są kodowane przez przeglądarkę. O sposobie kodowania decyduje atrybut enctype elementu FORM. Domyślnym kodowaniem formularzy jest application/x-www-form-urlencoded. Kodowanie to polega na utworzeniu par:

```
nazwakontrolki=wartosc
```

i połączeniu ich separatorem &. Wszystkie znaki specjalne występujące w nazwach lub wartościach kontrolek zostają przedstawione w postaci kodu szesnastkowego poprzedzonego znakiem procentu. Na przykład spacja jest zamieniana na napis %20 (kod ASCII znaku spacja - w systemie dziesiętnym - jest równy 32; liczba 32 w systemie szesnastkowym wynosi 20_{HEX}).

Nazwy zmiennych są pobierane z kodu HTML formularza. Każda kontrolka posiada atrybut name. Atrybut ten ustala nazwę zmiennej. W formularzu przedstawionym na listingu 1 występują dwie kontrolki o nazwach imie oraz nazwisko. Po wprowadzeniu do formularza danych *Aleksander Macedoński*, otrzymamy zakodowany napis:

```
imie=Aleksander&nazwisko=Macedo%F1ski
```

Jest to zawartość formularza z listingu 1 po zakodowaniu w formacie application/x-www-form-urlencoded. Litera „ń” została zamieniona na kod szesnastkowy %F1.

Pierwszy krok interakcji użytkownika z aplikacją internetową polega na wprowadzeniu danych do formularza. Następnie, po naciśnięciu przycisku Wyślij, przeglądarka koduje wprowadzone przez użytkownika dane, po czym wysyła odpowiednie zapytanie.

3. Przesyłanie danych pochodzących z formularza protokołem HTTP

Wszystkie transakcje WWW - a zatem także wysyłanie zawartości formularza - są realizowane przy użyciu protokołu HTTP. Protokół ten definiuje cztery metody przekazywania danych. Metodami tymi są POST, GET HEAD oraz PUT. W stosunku do formularzy zastosowanie znajdują dwie spośród nich: GET oraz POST.

W metodzie GET dane są dołączone do adresu URL i przyjmują postać:

```
http://gdzies.w.sieci/kat/strona.php?imie=Jan&nazwisko=Nowak&plec=M&wiek=35
```

Natomiast w metodzie POST dane z formularza są dołączone na końcu zapytania HTTP (za wszystkimi nagłówkami).

Metodę przekazywania danych formularza ustalamy atrybutem method elementu FORM. Ponieważ wartością domyślną jest GET, zatem formularz:

```
<FORM action="jakis-skrypt.php">
...
</FORM>
```

jest równoważny formularzowi:

```
<FORM action="jakis-skrypt.php" method="GET">
...
</FORM>
```

Formularz przekazywany metodą POST wygląda następująco:

```
<FORM action="jakis-skrypt.php" method="POST">
...
</FORM>
```

W zależności od użytej metody, dane pochodzące z formularza odbieramy na różne sposoby wewnątrz skryptu przetwarzającego formularz.

4. Odbieranie danych pochodzących z formularza w skrypcie php

Tablice $_GET, $_POST oraz $_REQUEST zawierające przetworzone dane pochodzące z formularza i dostępne wewnątrz skryptu php są zmiennymi *superglobalnymi*. Oznacza to, że są one widoczne wewnątrz wszystkich funkcji i metod bez konieczności stosowania słowa kluczowego global. Tablica $_GET zawiera dane przekazane do skryptu metodą GET. Tablica $_POST zawiera dane przekazane do skryptu metodą POST. Natomiast tablica

$_REQUEST zawiera dane pochodzące z ciasteczek, sesji, oraz przekazane metodami POST lub GET.

Dane pochodzące z formularzy przekazywanych metodą GET są dostępne w skrypcie php w tablicy $_GET. Jeśli formularz jest przekazany metodą POST, to należy użyć tablicy $_POST.

Wszystkie trzy wymienione tablice są *tablicami asocjacyjnymi*. Indeksem w powyższych tablicach może być napis. Jakiego indeksu powinniśmy użyć w celu odczytania imienia i nazwiska pochodzących z formularza widocznego na listingu 1? Napis wprowadzony w polu zatytułowanym *Imię* jest dostępny pod indeksem imie, zaś nazwisko - pod indeksem nazwisko. Indeksy imie i nazwisko są wartościami atrybutu name kontrolek INPUT. Jeśli użyto metody $_GET, wówczas imię i nazwisko podane przez internautę w formularzu są dostępne jako:

```
$_GET['imie']
$_GET['nazwisko']
```

Jeśli użyto metody $_POST, to należy użyć:

```
$_POST['imie']
$_POST['nazwisko']
```

Indeksami w tablicach $_POST i $_GET są nazwy kontrolek formularza. Jeśli formularz zawiera kontrolkę:

```
<INPUT name="owoc">
```

wówczas tablice $_POST I $_GET posiadają element o indeksie owoc. Po wprowadzeniu przez użytkownika w polu INPUT napisu gruszka, otrzymamy:

```
$_GET['owoc']  == 'gruszka'
```

lub

```
$_POST['owoc'] == 'gruszka'
```

(w zależności od użytej metody).

Zatem wybór metody przekazywania danych z formularza do skryptu php wpływa na wybór tablicy superglobalnej, z której skrypt będzie pobierał dane. Jeśli stosujemy metodę POST to należy korzystać z tablicy $_POST. Korzystając z metody GET dane pobieramy z tablicy $_GET.

6. Jakie dane zostały przesłane do skryptu?

Wszystkie informacje na temat danych pochodzących z formularza i dostępnych wewnątrz skryptu zwraca funkcja phpinfo(). Jeśli w skrypcie jakis-skrypt.php przetwarzającym formularz z listingu 1 umieścimy kod:

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

```
?>
```

wówczas funkcja phpinfo() wyświetli listę wszystkich zmiennych przekazanych z formularza do skryptu. Na rysunku 3 przedstawiono fragment wyniku funkcji phpinfo() zawierający elementy tablic $_GET oraz $_REQUEST.

PHP Variables

Variable	Value
_REQUEST["imie"]	Aleksander
_REQUEST["nazwisko"]	Macedoński
_GET["imie"]	Aleksander
_GET["nazwisko"]	Macedoński

Rysunek 3. Informacje wyświetlane przez funkcję phpinfo()

Pierwszy z przykładów składa się z dwóch plików: formularz.html oraz jakis-skrypt.php. Plik formularz.html zawiera formularz widoczny na listingu 1, zaś skrypt jakis-skrypt.php zawiera wywołanie funkcji phpinfo().

Drugim sposobem sprawdzenia danych przekazanych do skryptu jest użycie jednej z funkcji var_dump(), var_export() oraz print_r(). Kod:

```
<?php
var_dump($_GET);
?>
```

spowoduje wyświetlenie w oknie przeglądarki następujących informacji:

```
array(2) {
  ["imie"]=>
  string(10) "Aleksander"
  ["nazwisko"]=>
  string(10) "Macedoński"
}
```

Możemy również, stosując funkcję array_keys(), odczytać wszystkie indeksy tablicy $_GET, po czym w pętli foreach wydrukować kolejno wszystkie elementy tablicy:

```
$keys = array_keys($_GET);
foreach ($keys as $key) {
    echo "\$_GET['$key'] == {$_GET[$key]}<BR>";
}
```

Drugi przykład, podobnie jak i pierwszy składa się z dwóch plików. Plik formularz.html jest identyczny jak w pierwszym przykładzie. Natomiast skrypt jakis-skrypt.php z przykładu drugiego wywołuje funkcję var_dump() oraz przetwarza w tablicę $_GET pętlą foreach.

W analogiczny sposób możemy oczywiście użyć powyższych rozwiązań do wyświetlenia zawartości tablicy $_POST:

```
<?php
var_dump($_POST);
?>
```

Dodajmy jeszcze, że oprócz tablic $_GET, $_POST, $_REQUEST w skrypcie jest dostępna również tablica $_SERVER, która zawiera szczegółowe informacje na temat zapytania HTTP. Zmienne:

```
$_SERVER["REQUEST_METHOD"]
$_SERVER["REQUEST_URI"]
```

zawierają informacje na temat metody zapytania HTTP oraz żądanego dokumentu.

7. Nagłówki transakcji HTTP

Droga, jaką odbywają dane wprowadzone do formularza jest następująca:

- użytkownik wypełnia formularz, po czym naciska przycisk Wyślij,
- przeglądarka koduje informacje zawarte w formularzu, a następnie wysyła zapytanie HTTP do serwera,
- oprogramowanie działające na serwerze odbiera zapytanie HTTP,
- zapytanie jest przekazywane przez kolejne warstwy oprogramowania: stos protokołów TCPIP przekazuje zapytanie do procesu Apache, Apache uruchamia maszynę PHP i przekazuje jej zapytanie, zaś maszyna PHP przetwarza zapytanie, uruchamia skrypt i przekazuje do skryptu tablice $_GET, $_POST, itd.
- skrypt przetwarza dane, produkuje wynikowy kod HTML,
- kod zostaje wysłany w odpowiedzi HTTP do przeglądarki.

Jeśli porównamy zapytania wysłane metodą GET i POST, to zauważymy następujące różnice:

- w przypadku metody GET:
  - dane zapytania są zakodowane w adresie URL w postaci:
  - `jakis-skrypt.php?imie=Aleksander&nazwisko=Macedo%F1ski`
  - adres wyświetlany przez przeglądarkę, zawiera informacje o przekazanych zmiennych;
  - stronę możemy odświeżyć przyciskiem Odśwież.
- w przypadku metody POST:
  - dane zapytania są dołączone za nagłówkami;
  - dane nie są widoczne w polu adres przeglądarki;
  - odświeżanie strony powoduje wyświetlenie komunikatu.

Pamiętając o tym, by - ze względów bezpieczeństwa - stosować wyłącznie metodę POST, przejdźmy do omówienia przykładowego formularza i przetwarzającego go skryptu.

8. Przykład użycia formularza - kalkulator

Przygotujmy kalkulator, który dodaje odejmuje oraz mnoży dwie liczby wprowadzone przez użytkownika. Formularz kalkulatora zawiera dwie kontrolki INPUT umożliwiające wprowadzanie liczb oraz przycisk do wysyłania formularza.

Pierwsze rozwiązanie składa się z dwóch plików. Formularz jest zapisany w pliku kalkulator-formularz.html, zaś skrypt przetwarzający - w pliku kalkulator-skrypt.php. Listingi 2 oraz 3 przedstawiają oba pliki. Uwagę należy zwrócić na następujące zagadnienia:

- Wartością atrybutu action jest kalkulator-skrypt.php, więc formularz jest przetwarzany przez skrypt kalkulator-skrypt.php.
- Formularz jest przekazywany metodą POST, zatem danych z formularza będziemy szukali w tablicy $_POST.
- Dwie kontrolki formularza przeznaczone na liczby nazywają się liczba1 oraz libczba2 (spójrzmy na atrybuty name kontrolek INPUT). Liczby wprowadzone przez użytkownika w formularzu będą w skrypcie dostępne jako $_POST['liczba1'] oraz $_POST['liczba1'].
- Funkcja isset() zwraca informację logiczną, o tym, czy zmienna podana jako parametr jest dostępna. Innymi słowy: czy wizyta na stronie odbyła się za pośrednictwem formularza, czy użytkownik bezpośrednio odwiedził plik kalkulator-skrypt.php. Jeśli użytkownik pozostawił puste pola i nacisnął przycisk Wyślij, wówczas zmienne tablicy $_POST są określone, lecz puste (tzn. funkcja isset($_POST['liczba1']) zwraca logiczną prawdę oraz zachodzi równość $_POST['liczba1'] === "").
- Ponieważ użytkownik może wprowadzić w formularzu dowolne dane, niekoniecznie liczby, ale także napisy czy - jak już wspomnieliśmy - pozostawić pola formularza niewypełnione, zatem stosując funkcję is_numeric() sprawdzamy poprawność danych z tablicy $_POST.

```
<FORM action="kalkulator-skrypt.php" method="POST">
<TABLE>
<TR>
  <TD>Pierwsza liczba:</TD>
  <TD><INPUT name="liczba1"></TD>
</TR>
<TR>
  <TD>Druga liczba:</TD>
  <TD><INPUT name="liczba2"></TD>
</TR>
<TR>
  <TD> </TD>
  <TD><INPUT type="submit" value="Wyślij"></TD>
</TR>
</TABLE>
</FORM>
```

Listing 2. Formularz kalkulatora - rozwiązanie składające się z dwóch plików.

```
<?php

if (isset($_POST['liczba1']) && isset($_POST['liczba2'])) {
    if (is_numeric($_POST['liczba1']) && is_numeric($_POST['liczba2'])) {
        echo "W formularzu podano liczby {$_POST['liczba1']} oraz
{$_POST['liczba2']}.<BR>";
        echo "Wyniki działań:<BR>";

        echo "{$_POST['liczba1']} + {$_POST['liczba2']} = ";
        echo $_POST['liczba1'] + $_POST['liczba2'];
        echo "<BR>";

    } else {
        echo "Błędne dane! Jedna lub obie liczby są niepoprawne!<BR>";
```

```
    }
} else {
    echo "Brak danych! Jedna lub obie liczby nie zostały podane!<BR>";
}

?>
```

Listing 3. Skrypt przetwarzający formularz kalkulatora - rozwiązanie składające się z dwóch plików.

W celu ułatwienia powrotu po wykonaniu obliczeń do strony zawierającej formularz dodajmy poniżej skryptu php wykonującego obliczenia hiperłącze:

```
<A href="kalkulator-formularz.html">Powrót</A>
```

9. Umieszczanie formularza i skryptu w jednym pliku

Przedstawiony kalkulator możemy wykonać zapisując w jednym pliku zarówno formularz jak i skrypt php. Nazwijmy tenże plik kalkulator.php.

Przetwarzaniem formularza zajmie się plik kalkulator.php. Musimy więc zmodyfikować wartość atrybutu action elementu FORM. Jest to jedyna zmiana, jaką należy wykonać. Teraz po umieszczeniu skryptu poniżej formularza i ewentualnym zmienieniu nazwy pliku na kalkulator.php otrzymamy skrypt, którego szkielet jest przedstawiony na listingu 4. W miejscu wielokropków należy umieścić - bez żadnych modyfikacji - kod z listingów 2 oraz 3.

```
<BODY>

<FORM action="kalkulator.php" method="POST">
...
</FORM>

<HR>

<?php

if (isset($_POST['liczba1']) && isset($_POST['liczba2'])) {
...
}

?>

</BODY>
```

Listing 4. Kalkulator: formularz i skrypt są zapisane w jednym pliku kalkulator.php.

W przypadku, gdy plik zawierający formularz zajmuje się jego przetwarzaniem, wygodnym może się okazać użycie zmiennej $_SERVER['PHP_SELF']. Zmienna ta zawiera nazwę (dokładniej: względny adres URL) aktualnie wykonywanego skryptu. Ustalając atrybut action następująco:

```
<FORM action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
...
</FORM>
```

możemy dowolnie zmieniać nazwę pliku, bez konieczności modyfikacji atrybutu action.

10. Korzystanie z zewnętrznych skryptów do przetwarzania formularza

We wszystkich poprzednich przykładach przygotowywaliśmy zarówno formularz jak i skrypt przetwarzający. Co więcej, plik zawierający formularz oraz plik ze skryptem znajdowały się w tym samym folderze. Dzięki temu wartością atrybutu action była nazwa pliku, w którym zapisano skrypt.

Nie jest to konieczne. Po pierwsze wartością atrybutu action jest adres URL. Wynika z tego, że formularz i skrypt przetwarzający mogą znajdować się w innych folderach a nawet na innych serwerach. Po drugie, jako wartość atrybutu action możemy podać adres URL skryptu, napisanego przez kogoś innego.

Jako przykład wykonajmy formularze umożliwiające wyszukiwanie informacji w Google oraz walidację kodu HTML i CSS skryptami znajdującym się na witrynie W3C.

Formularz umożliwiający wyszukiwanie informacji w Google został przedstawiony na listingu 5. Skrypt o adresie http://www.google.com/search wyszukuje informacje w bazie danych. Jedyną kontrolką formularza, która umożliwia wprowadzanie danych jest pole INPUT o nazwie q. Zatem, mówiąc w języku php, powiedzielibyśmy, że tekst wpisany w formularzu zostaje przekazany do wyszukiwarki Google w zmiennej $_GET['q'] (formularz nie zawiera atrybutu method, zatem stosowana jest metoda domyślna GET).

```
<FORM action="http://www.google.com/search">
<P>
<INPUT type="hidden" name="hl" value="pl">
<INPUT name="q" value="" maxlength="255">
<INPUT type="submit" value="Szukaj" name="btnG">
</P>
</FORM>
```

Listing 5. Formularz do wyszukiwania w Google.

Formularze korzystające z serwisów walidujących W3C zostały przedstawione na listingach 6 oraz 7. Walidator HTML jest dostępny pod adresem http://validator.w3.org/check, zaś walidator CSS - http://jigsaw.w3.org/css-validator/validator. Oba formularze są przekazywane metodą GET, zaś adres strony do sprawdzenia jest zawarty w jedynym polu tekstowym o nazwie uri. Gdyby serwisy walidacyjne były napisane w php (nie są - są napisane w Perlu), wówczas do adresu wprowadzonego w formularzu odwołalibyśmy się wykorzystując zmienną $_GET['uri'].

```
<FORM action="http://validator.w3.org/check">
<P>
  HTML: <INPUT type="text" name="uri" size="100" value="">
  <INPUT type="SUBMIT" value="Sprawdź HTML">
  <INPUT type="hidden" name="charset" value="(detect automatically)">
  <INPUT type="hidden" name="doctype" value="(detect automatically)">
  <INPUT type="hidden" name="ss" value="">
  <INPUT type="hidden" name="outline" value="">
  <INPUT type="hidden" name="sp" value="">
  <INPUT type="hidden" name="noatt" value="">
</P>
```

```
</FORM>
```

Listing 6. Formularz do walidacji kodu HTML serwisem prowadzonym przez W3C.

```
<FORM action="http://jigsaw.w3.org/css-validator/validator">
<P>
  CSS2: <INPUT type="text" name="uri" size="100">
  <INPUT type="submit" value="Sprawdź CSS">
  <INPUT type="hidden" name="warning" value="1">
  <INPUT type="hidden" name="profile" value="css2">
</P>
</FORM>
```

Listing 7. Formularz do walidacji kodu CSS serwisem prowadzonym przez W3C.