PHP – ściągawka

PHP: HyperText Preprocessor – język programowania służący do tworzenia skryptów i programów wykonujących się na back-endzie, czyli po stronie serwera. Jest używany m.in. do obsługi formularzy, operacji CRUD (tworzenia, odczytu, aktualizacji/zapisu, usunięcia) na plikach, łączenia się z bazami danych, itd.

Do działania skryptów PHP wymagany jest serwer stron internetowych, wyposażony w interpreter PHP. Możemy zasymulować działanie takiego serwera np. dzięki pakietowi XAMPP, w skład którego wchodzą:

- serwer WWW – Apache
- silnik bazodanowy – MariaDB
- interpreter PHP
- interpreter Perl

Kolejne kroki niezbędne do uruchomienia naszej strony z użyciem XAMPP'a:

1. uruchom XAMPP
2. uruchom usługi Apache i MySQL
3. uruchom Explorer, otworzy się folder, w którym znajdują się wszystkie pliki XAMPP'a
4. otwórz folder htdocs
5. załóż w nim własny folder
6. umieść w twoim folderze wszystkie pliki projektu (pamiętaj, że plik strony głównej powinien nazywać się „index.php")
7. uruchom przeglądarkę, w pasku URL wpisz „localhost/twoja_nazwa_folderu"

Skrypty PHP umieszcza się w kodzie strony internetowej pomiędzy znacznikami: <?php ?>

Omówione już komendy PHP:

- $xyz – zmienna o nazwie xyz
- $wiek = 26 – przypisanie zmiennej wiek wartości liczbowej 26
- echo lub print – wypisanie (wyświetlenie) czegoś na stronie
- if, elseif, else – instrukcje warunkowe jeżeli
- for, while – pętle
- fopen(„nazwa pliku", „tryb dostępu") – otwarcie pliku w wybranym trybie (r – odczyt, w – zapis (nadpisywanie), a – zapis (dopisywanie))
- fwrite(plik, treść) – zapisanie w pliku wybranej treści
- fread(plik, rozmiar) – odczytanie z pliku treści o wybranym rozmiarze (wyrażonym w bajtach)
- fgets(plik) – odczytanie z pliku pojedynczej linijki treści (od początku linii, do najbliższego znaku nowej linii \n)
- feof(plik) – sprawdzenie, czy plik odczytano już do końca (eof – end of file)
- fclose(plik) – zamknięcie pliku, by nie zużywał zasobów serwera, oraz by nie było do niego dostępu (dopóki ktoś ponownie go nie otworzy)
- include – dołączenie zewnętrznego pliku PHP (jeżeli plik będzie uszkodzony, strona zwróci tylko ostrzeżenie, reszta kodu zostanie załadowana)
- require – dołączenie zewnętrznego pliku PHP (jeżeli plik będzie uszkodzony, strona zwróci błąd krytyczny, reszta kodu nie zostanie załadowana)

- htmlspecialchars(tekst) – funkcja zamieniająca w tekście znaki specjalne HTML (np. <, >, ©, itd.) na ich „bezpieczne" odpowiedniki (np. <, >, ©, itd.) – stosowana aby zapobiec tzw. „script injection", czyli wprowadzeniu szkodliwego skryptu na stronę poprzez pola formularzy.
- trim(tekst) – funkcja usuwająca z tekstu zbędne spacje
- stripslashes(tekst) – funkcja usuwająca z tekstu zbędne ukośniki
- isset(zmienna/pole_formularza) – funkcja sprawdzająca, czy dana zmienna lub dane pole formularza posiada jakąś wartość (czy nie jest niezdefiniowana (undefined)) – stosowana, by wstrzymać wykonywanie danego kawałka kodu, dopóki wszystkie wymagane zmienne i pola nie będą wypełnione
- filter_var(tekst, filtr) – funkcja sprawdzająca, czy dany tekst spełnia założenia danego filtra (np. czy wpisany przez użytkownika adres e-mail jest zgodny z filtrem FILTER_VALIDATE_EMAIL)
- preg_match(wzorzec, tekst) – funkcja sprawdzająca, czy w danym tekście istnieje tekst, pokrywający się ze wzorcem (np. czy w numerze telefonu znajdują się tylko cyfry) – często łączona z wyrażeniami RegEx
- date(wyrażenia) – funkcja zwracająca obecną datę w wybranym formacie – do wyświetlenia kolejnych części daty służą odpowiednie wyrażenia (patrz Ściągawka 9). Np. date(Y-m-d) zapisze datę w formacie 2024-03-25. Wielkość liter jest ważna!

PHP i formularze – formularze na stronach internetowych tworzy się znacznikiem <form>, który musi posiadać dwa ważne atrybuty: action – dokąd mają trafić dane z formularza, method – jaką metodą mają zostać przekazane te dane (GET lub POST). Przykładowy formularz:

<form action="pobierzFormularz.php" method="POST">

Aby skrypt połączył się z formularzem, będącym na tej samej stronie, należy w atrybucie action wpisać następujący skrypt:

<form action="<?php echo ($_SERVER['PHP_SELF']); ?>"...

Jest to skrypt, który wyświetla wartość stałej serwerowej, przechowującej nazwę pliku, w którym znajduje się ów skrypt.

Każdy formularz posiada pola do wpisywania wartości, tworzone znacznikiem <input>. Te pola posiadają trzy atrybuty: type – rodzaj pola, name – nazwa pola, id – identyfikator pola. PHP korzysta z wartości atrybutu name, aby komunikować się z formularzem i pobierać z niego wartości. Przykładowe pole formularza:

<input type="date" name="dataWizyty" id="dataW">

Każdy formularz powinien kończyć się przyciskiem przesyłającym dane formularza:

<input type="submit" value="Wyślij formularz">

Skrypt PHP, który skomunikuje się z formularzem i pobierze z niego dane może wyglądać tak:

```
<?php
    $data_wizyty = $_POST['dataWizyty'];
?>
```

Ogólna składnia służąca do pobrania wartości z formularza i przypisania go do zmiennej:

$nazwa_zmiennej = $_ GET_LUB_POST[‘name_pola_formularza’];

Aby wyświetlić tę wartość na stronie, można skorzystać z polecenia echo:

```
<?php
    $data_wizyty = $_POST[‘dataWizyty’];
    echo ’’Wizytę umówiono na dzień: $data_wizyty”;
?>
```

Aby zapisać tę wartość w pliku tekstowym, można skorzystać z poleceń fopen, fwrite i fclose:

```
<?php
    $data_wizyty = $_POST[‘dataWizyty’];
    echo ’’Wizytę umówiono na dzień: $data_wizyty”;
    $data_do_zapisu = „$data_wizyty \n”;
    $plik = fopen(’’plik.txt’’, ’’a’’);
    fwrite($plik, $data_do_zapisu);
    fclose($plik);
?>
```

Aby wyświetlić wszystkie linie pliku tekstowego, w którym te dane są zapisane, można skorzystać z poleceń fopen, pętli while, feof, echo, fgets i fclose:

```
<?php
    $plik = fopen(’’plik.txt’’, ’’r’’);
    while(!feof($plik)) {
        echo fgets($plik);
    }
    fclose($plik);
?>
```

RegEx – wyrażenia RegEx są wzorcami, mogącymi oznaczać dowolny znak z określonego zakresu. Są często stosowane np. do sprawdzenia, czy podane przez użytkownika hasło spełnia założenia bezpieczeństwa, np. czy posiada wielką literę, małą literę, znak specjalny, itd.

Wyrażenia RegEx można przypisywać do zmiennych. Samo wyrażenie musi być umieszczone w cudzysłowach i między tzw. delimiterami, np. symbolami ukośnika. Przykładowe wyrażenia RegEx:

- [A-Z] – dowolna wielka litera
- [a-z] – dowolna mała litera
- [abc] – litera a, b lub c
- [^xyz] – dowolny znak, niebędący literą x, y lub z
- [0-9] lub \d – dowolna cyfra
- [^0-9] lub \D – dowolny znak, niebędący cyfrą
- . – dowolny znak
- .{2} – dwa dowolne znaki obok siebie
- g{3,6} – od 3 do 6 liter g obok siebie
- [a-zA-Z]{8, } – minimum 8 liter o dowolnej wielkości obok siebie

Wyrażenia RegEx są wykorzystywane np. w funkcji preg_match(), która sprawdza, czy w danym tekście można odnaleźć fragment, pasujący do wzorca. Wynikiem tej funkcji jest 1 – znaleziono wzorzec, lub 0 – nie znaleziono wzorca. Przykłady:

```
<?php
    $wzor = ”/[A-Z]/”;
    $tekst = ”Witam”;
    echo preg_match($wzor, $tekst);
?>
```

Rezultatem powyższego przykładu będzie „1”, ponieważ w słowie „Witam” można znaleźć wielką literę „W”.

```
<?php
    $wzor = ”/[a-z][A-Z]/”;
    $tekst = ”Witam”;
    echo preg_match($wzor, $tekst);
?>
```

Rezultatem powyższego przykładu będzie „0”. Co prawda w słowie „Witam” można odnaleźć i małą, i wielką literę, ale powyższy wzorzec musi być odnaleziony zgodnie z kolejnością – najpierw mała litera, potem wielka.

```
<?php
    $wzor = ”/[\!\@\#\%]/”;
    $tekst = ”Co się stało, Hawat?”;
    echo preg_match($wzor, $tekst);
?>
```

Rezultatem powyższego przykładu będzie „0”. Wzorzec szuka symbolu wykrzyknika, małpy, hasha lub procentu. Każdy z tych symboli musi być poprzedzony symbolem backslash „\”. W powyższym zdaniu znajduje się znak specjalny „?”, ale nie ma go we wzorcu.

```
<?php
    $wzor = ”/\D/”;
    $tekst = ”Witam”;
    echo preg_match($wzor, $tekst);
?>
```

Rezultatem powyższego przykładu będzie „1”. Znak \D oznacza „jestem dowolnym znakiem, ale nie cyfrą”. Zatem już pierwsza litera „W” powoduje, że wzorzec jest odnaleziony. Z kolei, gdyby wzorcem był znak \d, to funkcja zwróciłaby wartość „0”, ponieważ w tekście nie znajduje się ani jedna cyfra.

```
<?php
    $wzor = ”/o{2}/”;
    $tekst = ”good”;
    echo preg_match($wzor, $tekst);
?>
```

Rezultatem powyższego polecenia będzie „1”. W słowie „good” można odnaleźć dokładnie 2 litery „o” obok siebie.

```
<?php
    $wzor = ”/X{3, }/”;
    $tekst = ”Koszulka XXL”;
    echo preg_match($wzor, $tekst);
?>
```

Rezultatem powyższego polecenia będzie „0”. W tekście „Koszulka XXL” są tylko 2 litery „X” pod rząd. Wzorzec szuka minimum 3.

Jeżeli chcemy sprawdzić kilka wzorców na raz, możemy skombinować je w dużym warunku if. Przykładowo, aby sprawdzić, czy w tekście, wpisanym przez użytkownika w polu formularza o nazwie „haslo” znajduje się 6 dowolnych znaków, ale minimum 2 z nich to dwie wielkie litery obok siebie, oraz że gdzieś znajduje się znak myślnika ”-”, można napisać kod:

```
<form action = ”<?php echo ($_SERVER[‘PHP_SELF’]); ?>” method=”POST”>
    Hasło: <input type=”password” name=”haslo” id=”haslo”><br>
    <input type=”submit”>
</form>
```

```
<?php
    $tekst = $_POST[‘haslo’];
    $wzor1 = ”/.{6, }/”;
    $wzor2 = ”/[A-Z]{2, }/”;
    $wzor3 = ”/\-/”;
    if (preg_match($wzor1, $tekst)) && preg_match($wzor2, $tekst)) && preg_match($wzor3, $tekst)) {
        echo ”Twoje hasło spełnia wszystkie wymagania”;
    }
    else {
        echo ”Twoje hasło nie spełnia wymagań”;
    }
?>
```