1. Úvod a premenné

PHP je skriptovací jazyk, ktorý beží na strane servera, čo znamená, že používateľ vidí v prehliadači až výsledný HTML kód.

Syntax: Kód píšeme medzi značky `<?php` a `?>`. Každý príkaz musí končiť stredníkom ;

Výstup: Na výpis textu používame príkaz `echo`.

Premenné:

- Začínajú znakom `$`, napr. `$vek = 15;`.

PHP automaticky rozlišuje typy ako celé čísla (int),

- desatinné čísla (float)
- textové reťazce (string)
- polia.

Príklad:

```
<?php
$meno = "Peter";
echo "Ahoj, $meno!"; // Vypíše: Ahoj, Peter!
?>
<?php
$meno = "Peter"; // Premenná začína znakom $
$vek = 16;       // PHP samo rozpozná, že ide o číslo

// Spájanie textu a premenných sa robí pomocou bodky
echo "Ahoj, volám sa " . $meno . " a mám " . $vek . " rokov.";
?>
```

2. Podmienky a logika

Podmienky umožňujú programu reagovať na rôzne situácie podľa pravdivosti výrazov.

Konštrukcia if/else: Ak podmienka v zátvorke platí, vykoná sa prvý blok, inak sa vykoná časť `else`.

Operátory: Používame == (rovnosť), `!=` (nerovnosť), < (menší), > (väčší). Pre prísnu kontrolu hodnoty aj dátového typu slúži ===.

Príklad:

```
<?php
$body = 75;
if ($body >= 50) {
    echo "Urobil si skúšku.";
} else {
    echo "Bohužiaľ, neuspel si.";
}
?>
```

3. Formuláre a bezpečnosť

Formuláre sú kľúčové pre interakciu s používateľom.

Metódy GET a POST: `GET` posiela údaje cez URL adresu (vhodné pre vyhľadávanie), kým `POST` ich posiela "neviditeľne" v tele požiadavky (vhodné pre heslá a citlivé údaje).

Spracovanie: PHP pristupuje k údajom cez polia `$_GET` alebo `$_POST` podľa toho, aká metóda bola použitá v HTML formulári.

Bezpečnosť: Vždy používajte funkciu `htmlspecialchars()`, ktorá zabraňuje XSS útokom tým, že prevedie nebezpečné znaky na neškodný text.

Príklad spracovania:

```
<?php
$meno = $_POST['meno'] ?? "Hosť";
echo "Vitaj, " . htmlspecialchars($meno);
?>
```

Údaje z formulárov sa najčastejšie posielajú metódou `POST` (skryte v tele požiadavky) alebo `GET` (viditeľne v URL adrese).

```
// HTML časť (formular.html)
// <form method="POST" action="spracuj.php">
//    Meno: <input type="text" name="uzivatel">
//    <input type="submit" value="Odoslať">
// </form>

// PHP časť (spracuj.php)
<?php
```

```
$meno = $_POST['uzivatel']; // Získanie údaja z poľa $_POST

// htmlspecialchars() zabraňuje XSS útokom
echo "Vitaj, " . htmlspecialchars($meno) . "!";
?>
```

Vysvetlenie: `POST` je vhodnejší pre citlivé údaje. Funkcia `htmlspecialchars()` mení nebezpečné znaky (ako `<` alebo `>`) na text, aby útočník nemohol do stránky vložiť škodlivý kód.

4. Cykly

Tu sú praktické ukážky základných cyklov v PHP:

1. Cyklus `for` (Pevný počet opakovaní)

Používa sa vtedy, keď presne vieme, koľkokrát sa má kód vykonať. Využíva riadiacu premennú (často `$i`), ktorá sa v každom kroku mení.

```
<?php
// Výpis čísel od 1 do 100
for ($i = 1; $i <= 100; $i++) {
    echo $i . " ";
}
?>
```

Zdroj:

2. Cyklus `while` (Opakovanie podľa podmienky)

Tento cyklus opakuje príkazy dovtedy, kým platí zadaná podmienka. Je ideálny napríklad pri čítaní údajov, kde vopred nepoznáme presný počet krokov.

```
<?php
$i = 1;
while ($i <= 10) {
    echo $i . " ";
    $i++; // Dôležité: musíme manuálne zvýšiť hodnotu, inak cyklus nikdy neskončí
}
?>
```

Zdroj:

3. Cyklus `foreach` (Prechádzanie polí)

Je to zjednodušená verzia cyklu určená špeciálne na prácu s poliami. V každom kroku vráti aktuálnu položku z poľa.

```
<?php
$mena = array("Peter", "Anna", "Michal");
foreach ($mena as $meno) {
    echo "Ahoj, " . $meno . "<br>";
}
?>
```

Okrem týchto existuje aj cyklus **do while**, ktorý podmienku vyhodnocuje až na konci, takže kód prebehne vždy aspoň raz.

Tu sú tri ukážky zložitejšieho využitia PHP, ktoré kombinujú cykly, logiku a prácu s údajmi:

1. Tabuľka malej násobilky (Vnorené cykly)

Tento príklad využíva jeden cyklus vo vnútri druhého. Vonkajší cyklus (`$j`) vytvára riadky a vnútorný cyklus (`$i`) vypočítava hodnoty v stĺpcoch.

```
echo '<table border="1">';
for ($j = 1; $j <= 10; $j++) {
    echo '<tr>';
    for ($i = 1; $i <= 10; $i++) {
        echo '<td>' . ($i * $j) . '</td>'; // Násobí aktuálny riadok stĺpcom
    }
    echo '</tr>';
}
echo '</table>';
```

2. Vykreslenie šachovnice (Logika a Modulo)

Tu kombinujeme vnorené cykly s operátorom modulo (`%`), ktorý zisťuje zvyšok po delení. Ak je súčet súradníc `$i + $j` párny, políčko bude čierne, inak biele.

```
echo '<table border="1">';
for ($i = 0; $i < 8; $i++) {
    echo '<tr>';
    for ($j = 0; $j < 8; $j++) {
        if (($i + $j) % 2 == 0) {
            $styl = 'style="background: black; width: 30px; height: 30px;"';
        } else {
```

```
            $styl = 'style="background: white; width: 30px; height: 30px;"';
        }
        echo "<td $styl></td>";
    }
    echo '</tr>';
}
echo '</table>';
```

3. Práca s viacrozmerným poľom

V praxi sa údaje často ukladajú do polí, ktoré obsahujú ďalšie polia (napr. zoznam používateľov). Na ich výpis sa používa cyklus `foreach`.

```
$ludia = array(
    array('meno' => 'Pavol Novák', 'vek' => 20),
    array('meno' => 'Tomáš Márny', 'vek' => 50),
    array('meno' => 'Jana Nová', 'vek' => 35)
);

echo '<table border="1"><tr><th>Meno</th><th>Vek</th></tr>';
foreach ($ludia as $clovek) {
    echo '<tr>';
    echo '<td>' . htmlspecialchars($clovek['meno']) . '</td>';
    echo '<td>' . htmlspecialchars($clovek['vek']) . '</td>';
    echo '</tr>';
}
echo '</table>';
```

5. Práca s poliami (Arrays)

Pole je dátová štruktúra, ktorá umožňuje uložiť viacero hodnôt do jednej premennej namiesto vytvárania desiatok samostatných premenných. V PHP rozlišujeme tri základné typy polí:

Číselne indexované polia: Predstavte si ich ako rad očíslovaných priehradiek v pamäti, kde sa začína vždy od nuly. Ak do nich vložíte dáta, PHP im automaticky priradí indexy 0, 1, 2 atď..

Príklad: `$znamky =;` – tu je na indexe 0 hodnota 1.

Asociatívne polia: Namiesto čísiel používajú textové kľúče, čo robí kód oveľa čitateľnejším. Kľúč a hodnotu spájame operátorom `=>`.

Príklad: `$uzivatel = array("meno" => "Peter", "vek" => 18);` – namiesto hľadania v indexe 0 presne vieme, že pod kľúčom "meno" nájdeme krstné meno.

Viacrozmerné polia: Sú to vlastne „polia v poliach“. Používajú sa na ukladanie tabuľkových dát, kde každá položka hlavného poľa je ďalším (často asociatívnym) poľom s údajmi o konkrétnom objekte, napríklad o používateľovi.

Na spracovanie polí je najlepší cyklus **foreach**, ktorý prejde všetky položky bez toho, aby ste museli riešiť, koľko ich presne je.

6. Práca s databázami (MySQL)

Databáza slúži na to, aby dáta (ako mená používateľov alebo heslá) nezmizli po zatvorení prehliadača, ale zostali trvalo uložené na serveri.

Štruktúra: Dáta sú v MySQL organizované do **tabuliek**, ktoré majú stĺpce (napr. *id, meno, email*) a riadky (konkrétne záznamy o ľuďoch).

Pripojenie: Aby PHP mohlo s databázou hovoriť, musí sa k nej pripojiť pomocou servera, mena, hesla a názvu databázy. Bez tohto kroku nie je možné vykonať žiadnu operáciu.

SQL príkazy: Komunikácia prebieha cez jazyk SQL. Príkaz **SELECT** vytiahne dáta z tabuľky (napr. pri prihlasovaní), kým **INSERT** do nej vloží nové dáta (napr. pri registrácii).

Bezpečnosť: Toto je kritická časť. Heslá sa do databázy nikdy neukladajú v čitateľnej forme, ale musia sa „hashovať“ (napr. cez `password_hash()`), aby ich útočník nemohol prečítať ani pri úniku dát. Taktiež je nutné ošetrovať vstupy proti útokom typu SQL Injection a XSS pomocou funkcií ako `htmlspecialchars()`.

Dáva ti to takto podrobnejšie zmysel? Ak chceš, môžeme si teraz ukázať, ako vyzerá konkrétny kód na pripojenie k databáze a uloženie údajov z formulára.

Tu je podrobný rozpis toho, ako prepojiť PHP s databázou MySQL a uložiť do nej údaje zadané používateľom.

7. Príprava pripojenia (connect.php)

Aby skript vedel komunikovať s databázou, musí najprv definovať prístupové údaje k serveru. Tieto údaje sa zvyčajne ukladajú do samostatného súboru, ktorý sa potom vkladá do ostatných skriptov pomocou príkazu `include`.

Server: Adresa, kde databáza beží (napr. 'localhost').

Login a Heslo: Prihlasovacie údaje vytvorené v administrácii hostingu.

Názov databázy: Konkrétna databáza, v ktorej máte vytvorené tabuľky.

```
<?php
```

```
$db_server = 'názov_servera';
$db_login = 'uzivatel';
$db_password = 'heslo';
$db_name = 'názov_databázy';

// Vytvorenie spojenia so serverom a výber databázy
$spojeni = @MySQL_Connect($db_server, $db_login, $db_password);
@MySQL_Select_DB($db_name) or die('Chyba pripojenia');
mysql_query("set names utf8"); // Nastavenie kódovania pre správnu slovenčinu
?>
```

8. HTML formulár pre zber dát

Formulár slúži na to, aby používateľ mohol zadať svoje údaje. Používame metódu **POST**, pretože je bezpečnejšia pre osobné údaje a neobmedzuje dĺžku odosielaných dát. Každý prvok `input` musí mať priradený atribút `name`, pod ktorým PHP tento údaj rozpozná.

```
<form action="ulozit.php" method="post">
    Prezývka: <input type="text" name="nick"><br>
    Email: <input type="text" name="email"><br>
    Heslo: <input type="password" name="heslo"><br>
    <input type="submit" name="submit" value="Registrovať sa">
</form>
```

9. Spracovanie a uloženie údajov (ulozit.php)

V tomto kroku PHP prijme dáta zo superglobálneho poľa `$_POST`, ošetrí ich proti útokom a zapíše do databázovej tabuľky pomocou SQL príkazu `INSERT INTO`.

```
<?php
include "connect.php"; // Načítame konfiguračný súbor s pripojením

if(isset($_POST['submit'])) { // Kontrola, či bol formulár skutočne odoslaný
    // Ošetrenie vstupov proti nebezpečným znakom
    $nick = mysql_real_escape_string($_POST['nick']);
    $email = mysql_real_escape_string($_POST['email']);
    $heslo = md5($_POST['heslo']); // Základné zahashovanie hesla

    // SQL dotaz na vloženie dát do tabuľky 'uzivatele'
    $sql = mysql_query("INSERT INTO uzivatele VALUES ('', '$nick', '$heslo',
'$email')")
           or die(mysql_error());

    echo "Registrácia bola úspešne dokončená!";
}
?>
```

Dôležité upozornenie: Hoci zdroje uvádzajú funkcie `mysql_`, v modernom PHP sa odporúča prejsť na bezpečnejšie rozhrania ako **PDO** alebo **mysqli**, pretože staré funkcie už nie sú podporované.

Načítanie údajov a ich zobrazenie v tabuľke je základom dynamických webov. Celý proces prebieha v troch krokoch:

A. SQL dopyt

Pomocou príkazu `SELECT` povieme databáze, ktoré údaje chceme vytiahnuť. Ak chceme všetky údaje z tabuľky „uzivatele“, dopyt bude vyzerať takto: `SELECT * FROM uzivatele`.

B. Spracovanie výsledkov (Cyklus)

Keď nám databáza vráti dáta, PHP ich spracováva riadok po riadku pomocou funkcie `mysql_fetch_array()`, ktorá premení každý záznam na pole. Na prechádzanie všetkých záznamov použijeme cyklus `while`.

C. Výpis do HTML tabuľky

Dáta vkladáme priamo do HTML značiek `<table>`, `<tr>` (riadok) a `<td>` (bunka).

Príklad kódu:

```
<?php
include "connect.php"; // Pripojenie k DB

$vysledok = mysql_query("SELECT login, email FROM uzivatele");

echo '<table border="1">
        <tr><th>Meno</th><th>E-mail</th></tr>';

// Cyklus while beží, kým sú v tabuľke ďalšie riadky
```

```
while ($riadok = mysql_fetch_array($vysledok)) {
    echo '<tr>';
    // htmlspecialchars chráni pred vložením škodlivého kódu
    echo '<td>' . htmlspecialchars($riadok['login']) . '</td>';
    echo '<td>' . htmlspecialchars($riadok['email']) . '</td>';
    echo '</tr>';
}

echo '</table>';
?>
```

Čo sa v kóde deje?

Dopyt: Príkaz `mysql_query` pošle požiadavku serveru.

Získanie dát: `mysql_fetch_array` vytiahne jeden riadok z výsledkov.

Zobrazenie: `echo` vypíše hodnoty priamo do buniek tabuľky.

Bezpečnosť: Funkcia `htmlspecialchars()` zabezpečí, že sa text z databázy zobrazí bezpečne a nespustí prípadný útočníkov skript.