Cvičebnice PHP

3. ročník

(Programování webových aplikací)

Obsah

Lekce 1: Úvod do PHP

Cíl lekce

Vysvětlit, co je PHP, kde běží, jak se liší od JavaScriptu a jak vytvořit svůj první PHP skript.

Co je PHP

PHP je skriptovací jazyk, který běží na serveru. Na rozdíl od JavaScriptu, který se provádí v prohlížeči, PHP generuje výsledný HTML obsah, který je poté odeslán klientovi.

- JavaScript = běží v prohlížeči (client-side)
- PHP = běží na serveru (server-side)
- PHP zpracovává formuláře, pracuje se soubory a vytváří dynamické weby

Jak spustit PHP

Pro spuštění PHP je potřeba lokální serverové prostředí, například:

- XAMPP (doporučeno pro výuku)
- WAMP / MAMP
- Integrovaný PHP server (z příkazové řádky)

Soubor s příponou `.php` musí být uložen v kořenové složce serveru (např. `htdocs`).

První PHP soubor

Vytvořte soubor s názvem `index.php` a vložte do něj následující kód:

```
<?php
echo "Ahoj světe!";
?>
```

Stránku otevřete na adrese:

`http://localhost/index.php`

Komentáře v PHP

Komentáře fungují podobně jako v JavaScriptu:

```
// jednořádkový komentář

/*
    víceřádkový
    komentář
*/
```

Cvičení

Cvičení 1.1: První výpis v PHP
Upravte kód tak, aby vypsal: *"Vítejte v kurzu PHP!"*

Cvičení 1.2: HTML + PHP
Vytvořte HTML stránku, ve které bude PHP kód vypisovat vaše jméno.

Cvičení 1.3: Více příkazů echo
Vypište na stránku tři různé věty pomocí funkce `echo`.

Lekce 2: Proměnné a datové typy v PHP

Cíl lekce

Seznámit se s vytvářením proměnných v PHP, naučit se pracovat s datovými typy a pochopit rozdíly mezi PHP a JavaScriptem.

Proměnné v PHP

V PHP se všechny proměnné zapisují se znakem `$` na začátku:

```
$jmeno = "Kristýna";
$vek = 18;
$student = true;
```

Rozdíly oproti JavaScriptu:

- Nepoužívá se `let`, `const` ani `var`.
- Proměnná vždy začíná symbolem `$`.
- PHP je volně typovaný jazyk (typ se určí podle hodnoty).

Základní datové typy v PHP

- **string** – text
- **int** – celé číslo
- **float** – desetinné číslo
- **bool** – true / false
- **array** – pole
- **null** – prázdná hodnota

Ukázky:

```
$nazev = "PHP";
$cislo = 42;
$cena = 19.99;
$aktivni = false;
$prazdne = null;
```

Práce s textem (string)

PHP umožňuje zapisovat text dvěma způsoby:

- dvojité uvozovky – zpracovávají proměnné
- jednoduché uvozovky – berou obsah doslova

```
$jmeno = "Kristýna";
echo "Ahoj, $jmeno";  // proměnná se vyhodnotí

echo 'Ahoj, $jmeno'; // proměnná se nevyhodnotí
```

Spojování textu

V PHP se text spojuje operátorem `.` (tečka).

```
$jmeno = "Kristýna";
$pozdrav = "Ahoj " . $jmeno . "!";
```

Cvičení

Cvičení 2.1 – Základní proměnná
Vytvořte proměnnou se svým jménem a vypište ji pomocí `echo`.

Cvičení 2.2 – Text a číslo
Uložte do proměnné cenu produktu a vypište větu: *"Cena produktu je X Kč."*

Cvičení 2.3 – Dvě proměnné
Uložte své jméno a příjmení do dvou proměnných a vypište je v jedné větě.

Cvičení 2.4 – Datový typ
Vytvořte proměnnou s číslem a vypište její datový typ pomocí `var_dump()`.

Cvičení 2.5 – Boolean hodnota
Vytvořte proměnnou s hodnotou `true` a vypište ji.

Cvičení 2.6 – Změna hodnoty
Vytvořte proměnnou `$vek` a postupně do ní přiřaďte tři různé hodnoty. Na konci vypište tu poslední.

Cvičení 2.7 – Spojování textu
Spojte text "Ahoj, "s vaším jménem pomocí operátoru `.`

Cvičení 2.8 – Počítání délky textu
Uložte libovolné slovo do proměnné a vypište jeho délku funkcí `strlen()`.

Cvičení 2.9 – Jednoduchý výpočet
Vytvořte tři proměnné `$a`, `$b`, `$c`. Do `$c` uložte součet `$a` a `$b`.

Cvičení 2.10 – Přepsání proměnné
Uložte do proměnné `$x` číslo 5 a poté do ní uložte text "pět". Zkontrolujte pomocí `var_dump()`, jak se změnil typ.

Cvičení 2.11 – Null hodnota
Vytvořte proměnnou a nastavte její hodnotu na `null`. Vypište ji na stránku.

Cvičení 2.12 – Převod čísel na text
Mějte číslo v proměnné `$cislo`. Spojte ho s textem způsobem: `"Výsledek je: ". $cislo`

Cvičení 2.13 – Řetězce a proměnné v uvozovkách
Mějte proměnnou `$jmeno`. Vyzkoušejte rozdíl mezi:

```
echo "Ahoj $jmeno"; echo 'Ahoj $jmeno';
```

Cvičení 2.14 – Velká a malá písmena
Do proměnné uložte libovolný text a vypište jej velkými písmeny pomocí `strtoupper()`.

Cvičení 2.15 – Věta o vás
Vytvořte tři proměnné: `$jmeno`, `$vek`, `$mesto`. Vypište větu ve formátu: *"Jmenuji se X, je mi Y let a pocházím z města Z."*

Lekce 3: Operátory v PHP

Cíl lekce

Seznámit se s nejpoužívanějšími operátory v PHP, pochopit rozdíly oproti JavaScriptu a naučit se je využívat v praxi.

Aritmetické operátory

Aritmetické operátory v PHP fungují stejně jako v JavaScriptu.

- `+` – sčítání
- `-` – odčítání
- `*` – násobení
- `/` – dělení
- `%` – modulo (zbytek po dělení)

```
$a = 10;
$b = 3;

echo $a + $b; // 13
echo $a % $b; // 1
```

Operátory přiřazení

Podobně jako v JavaScriptu:

- = – základní přiřazení
- += – přičtení hodnoty
- -= – odečtení hodnoty
- *= – vynásobení
- /= – vydělení

```
$x = 5;
$x += 3; // x = 8
```

Porovnávací operátory

- == – porovnání hodnot
- === – porovnání hodnoty i typu
- != – nerovnost
- !== – nerovnost (včetně typu)
- > , < , >= , <=

PHP má rozdíl mezi == a ===, podobně jako JavaScript.

```
echo (5 == "5");  // true
echo (5 === "5"); // false
```

Logické operátory

- `&&` – AND
- `||` – OR
- `!` – NOT

```
$vek = 20;

if ($vek > 18 && $vek < 30) {
    echo "Vhodný věk.";
}
```

Spojování textu

V PHP se text nespojuje pomocí + jako v JavaScriptu, ale pomocí tečky `.`

```
$jmeno = "Kristýna";
echo "Ahoj " . $jmeno . "!";
```

Cvičení

Cvičení 3.1 – Aritmetika základ
Spočítejte součet, rozdíl, součin a podíl čísel 12 a 4.

Cvičení 3.2 – Modulo
Pomocí operátoru `%` zjistěte zbytek po dělení 17 a 5.

Cvičení 3.3 – Kombinované přiřazení
Vytvořte proměnnou `$x = 10`. Poté postupně proveďte: `$x += 5`, `$x -= 2`, `$x *= 3`. Vypište konečný výsledek.

Cvičení 3.4 – Porovnání hodnot
Zjistěte, zda jsou hodnoty `10` a `"10"` stejné pomocí == i ===.

Cvičení 3.5 – Nerovnost
Vyhodnoťte výraz `7 != 5`. Co vrátí PHP?

Cvičení 3.6 – Logické operátory
Zjistěte, zda je číslo v proměnné `$vek` mezi 15 a 30 (pomocí `&&`).

Cvičení 3.7 – OR podmínka
Vytvořte dvě proměnné `$a = true` a `$b = false`. Vyzkoušejte výraz `$a || $b`.

Cvičení 3.8 – Negace
Vyzkoušejte výraz `!false`. Jaký je výsledek?

Cvičení 3.9 – Spojování textu
Spojte dvě proměnné `$jmeno = "Eva"` a `$prijmeni = "Nováková"` do jedné věty.

Cvičení 3.10 – Pozor na sčítání textu
Vyzkoušejte výraz: `"5"+ 5` Co vrátí PHP? Proč?

Cvičení 3.11 – Porovnání typu
Zjistěte, jaký výsledek vrátí výrazy: `0 == "0"` `0 === "0"`

Cvičení 3.12 – Kombinovaný logický výraz
Vyhodnoťte výraz: `(5 > 3) && (2 < 1 || 4 == 4)`

Cvičení 3.13 – Aritmetika + text
Vytvořte proměnné `$a = 6`, `$b = 2`. Vypište text: *"Součet je X a rozdíl je Y"*.

Cvičení 3.14 – Přiřazení s textem
Vytvořte proměnnou `$pozdrav = "Ahoj"` a připojte k ní text ‘"světe!„ pomocí `.=`

Cvičení 3.15 –Vyhodnocení věku
Pomocí proměnné `$vek` zjistěte, zda je člověk dospělý nebo nezletilý.

Lekce 4: Funkce v PHP

Cíl lekce

Pochopit, jak se v PHP definují funkce, jak pracují parametry a návratové hodnoty, a jak se funkce liší od JavaScriptu.

Co je funkce

Funkce je pojmenovaný blok kódu, který vykonává určitou činnost.
V JavaScriptu píšeme:

```
function ahoj() {
    console.log("Ahoj!");
}
```

V PHP píšeme:

```
function ahoj() {
    echo "Ahoj!";
}
```

Funkce se volá úplně stejně:

```
ahoj();
```

Funkce s parametry

Funkce může přijímat vstupní hodnoty (parametry):

```
function pozdrav($jmeno) {
    echo "Ahoj, $jmeno!";
}

pozdrav("Kristýna");
```

Důležité rozdíly oproti JS:

- parametry mají před sebou `$`
- PHP neumí anonymní funkce tak přirozeně jako JS (ale existují)

Návratová hodnota

Pomocí `return` vrátíme hodnotu zpět z funkce:

```
function secti($a, $b) {
    return $a + $b;
}

$vysledek = secti(5, 7);
echo $vysledek;
```

Výchozí hodnoty parametrů

Můžeme nastavit výchozí hodnotu:

```
function info($jmeno = "Neznámý") {
    echo "Uživatel: $jmeno";
}

info();             // Uživatel: Neznámý
info("Radek");      // Uživatel: Radek
```

Typová deklarace

V PHP 7+ můžeme uvést typ parametrů i návratové hodnoty:

```
function secti(int $a, int $b): int {
    return $a + $b;
}

echo secti(4, 5);
```

Pokud bude typ nesprávný, PHP zobrazí chybu.

Funkce uvnitř HTML

Funkce lze používat i v HTML souborech:

```
<?php
function pozdrav($jmeno) {
    return "Ahoj, $jmeno!";
}
?>
<p><?php echo pozdrav("Eva"); ?></p>
```

Cvičení

Cvičení 4.1 – Jednoduchá funkce
Napište funkci, která vypíše text „Ahoj světe“.

Cvičení 4.2 – Funkce s parametrem
Napište funkci `pozdrav($jmeno)`, která vypíše pozdrav.

Cvičení 4.3 – Návratová hodnota
Vytvořte funkci `nasob($a, $b)`, která vrátí součin.

Cvičení 4.4 – Defaultní parametr
Napište funkci, která vypíše větu o člověku. Pokud parametr jméno chybí, použijte hodnotu „Neznámý“.

Cvičení 4.5 – Typované funkce
Napište funkci `odecist(int $a, int $b): int` a vraťte výsledek.

Cvičení 4.6 – Řetězec z funkce
Vytvořte funkci, která vrátí textovou zprávu formátu: *"Uživatel X má Y let."*

Cvičení 4.7 – Více parametrů
Napište funkci, která vypíše větu: *"Součet je X a rozdíl je Y."*

Cvičení 4.8 – Znovupoužitelnost
Vytvořte funkci, která dvakrát vypíše stejný text.

Cvičení 4.9 – Funkce vracející boolean
Napište funkci, která vrátí `true`, pokud je číslo sudé.

Cvičení 4.10 – Funkce uvnitř HTML
Použijte funkci v HTML stránce a zobrazte výsledek v odstavci.

Cvičení 4.11 – Kontrola délky jména

Funkce vrátí true, pokud jméno má alespoň 5 znaků.

Cvičení 4.12 – Funkce s počítáním

Funkce přijme počet kusů a cenu za kus a vrátí celkovou cenu.

Cvičení 4.13 – Funkce s výstupem

Vytvořte funkci, která vypíše hvězdičku `"*"` tolikrát, kolik je argument.

Cvičení 4.14 – Mini validace

Funkce vrátí true, pokud je věk $\geq$ 18, jinak false.

Cvičení 4.15 – Oslovení

Napište funkci, která podle pohlaví vypíše: „Vážený pane“ nebo „Vážená paní“.

Lekce 5: Podmínky v PHP

Cíl lekce

Naučit se pracovat s podmínkami v PHP, pochopit rozdíl mezi jednotlivými typy podmínek a procvičit logické myšlení při větvení programu.

Základní podmínka

Základní struktura:

```
if ($cislo > 10) {
    echo "Číslo je větší než 10";
}
```

Podmínka se vyhodnocuje stejně jako v JavaScriptu.

If / else

```
if ($vek >= 18) {
    echo "Dospělý";
} else {
    echo "Nezletilý";
}
```

Elseif

PHP používá klíčové slovo `elseif`:

```
if ($hodnota > 0) {
    echo "Kladné";
} elseif ($hodnota < 0) {
    echo "Záporné";
} else {
    echo "Nula";
}
```

Logické operátory v podmínkách

Můžeme kombinovat více podmínek:

- `&&` – AND
- `||` – OR
- `!` – NOT

```
if ($vek >= 18 && $vek <= 65) {
    echo "Pracovní věk";
}
```

Častá chyba: = vs ==

```
if ($x = 5) {
    echo "Pozor!";
}
```

Tento kód NEporovnává. Místo toho přiřazuje hodnotu 5 do `$x`. Podmínka je potom vždy pravdivá.
Správně:

```
if ($x == 5) { ... }
```

Vnořené podmínky

```
if ($vek >= 18) {
    if ($vek < 30) {
        echo "Mladý dospělý";
    }
}
```

Je ale lepší použít :

```
if ($vek >= 18 && $vek < 30) {
    echo "Mladý dospělý";
}
```

Cvičení

Cvičení 5.1 – Základní podmínka
Zkontrolujte, zda je číslo v proměnné `$cislo` větší než 10.

Cvičení 5.2 – If / else
Vypište, zda je uživatel dospělý nebo ne.

Cvičení 5.3 – Elseif
Otestujte, zda je číslo kladné, záporné nebo nulové.

Cvičení 5.4 – Kombinovaná podmínka
Zjistěte, zda je číslo mezi 1 a 100.

Cvičení 5.5 – Kontrola typu
Porovnejte `5` a `"5"` pomocí `===`.

Cvičení 5.6 – Sudé nebo liché?
Pomocí podmínky vypište, zda je číslo sudé.

Cvičení 5.7 – Logické operátory
Zjistěte, zda je člověk mladý DOSPĚLÝ (18–30 let).

Cvičení 5.8 – Negace
Vytvořte boolean proměnnou a pomocí `!` zobrazte opačnou hodnotu.

Cvičení 5.9 – = vs ==
Opravte záměrně chybnou podmínku s = na správnou.

Cvičení 5.10 – Pohlaví uživatele
Máte proměnnou `$pohlavi`. Pokud je „muz“, vypište „Vítej, pane“. Pokud „zena“, vypište „Vítej, paní“.

Cvičení 5.11 – Teplota
Pokud je teplota $< 0 \rightarrow$ „mrzne“. Pokud mezi 0–20 $\rightarrow$ „chladno“. Jinak $\rightarrow$ „teplo“.

Cvičení 5.12 – Heslo
Pokud je délka hesla < 8 znaků $\rightarrow$ „slabé“, jinak „silné“.

Cvičení 5.13 – Kombinovaná logika
Je-li uživatel starší 18 nebo má rodiče, kteří souhlasí $\rightarrow$ může se zúčastnit.

Cvičení 5.14 – Výběr akce
Máte proměnnou `$cas`. Pokud je $< 12 \rightarrow$ „Dopolední akce“. Pokud 12–18 $\rightarrow$ „Odpolední akce“. Jinak $\rightarrow$ „Večerní akce“.

Lekce 6: Cykly v PHP

Cíl lekce

Naučit se používat cykly v PHP, pochopit rozdíly mezi jednotlivými typy cyklů a procvičit práci s opakováním kódu.

For cyklus

Funguje podobně jako v JavaScriptu.

```
for ($i = 0; $i < 5; $i++) {
    echo $i;
}
```

Tento kód vypíše čísla 0–4.

While cyklus

While se opakuje, dokud je podmínka pravdivá:

```
$i = 1;

while ($i <= 5) {
    echo $i;
    $i++;
}
```

Do...while

Do...while se provede alespoň jednou:

```
$i = 1;

do {
    echo $i;
    $i++;
} while ($i <= 5);
```

Foreach cyklus

Foreach se používá hlavně pro pole:

```
$ovoce = ["jablko", "banán", "hruška"];

foreach ($ovoce as $item) {
    echo $item;
}
```

Foreach je nejjednodušší cyklus na práci s daty.

Break a continue

break ukončí celý cyklus. **continue** přeskočí aktuální opakování.

```
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) continue;
    if ($i == 8) break;
    echo $i;
}
```

Nejčastější chyby

- Chybějící `$i++` → nekonečný cyklus
- Nesprávná podmínka (např. `while (true)`)
- Použití foreach na neexistující proměnnou

Cvičení

Cvičení 6.1 – Jednoduchý for
Vypište čísla 1–10 pomocí `for`.

Cvičení 6.2 – Sudá čísla
Pomocí `for` vypište pouze sudá čísla od 1 do 20.

Cvičení 6.3 – Odpočítávání
Vypište čísla od 10 do 1.

Cvičení 6.4 – While cyklus
Vypište čísla 1–5 pomocí `while`.

Cvičení 6.5 – Do...while
Napište cyklus, který se vždy provede aspoň jednou.

Cvičení 6.6 – Foreach
Vypište všechna zvířata z pole: `["pes", "kočka", "králík"]`

Cvičení 6.7 – Slova a foreach
Vypište každé slovo z věty: `"Ahoj jak se máš"` (použijte `explode`)

Cvičení 6.8 – Tabulka násobků
Pomocí `for` vypište násobky 7 (7, 14, 21, ... 70).

Cvičení 6.9 – Součet
Vypište součet čísel 1–100.

Cvičení 6.10 – Break
Ukončete cyklus, když `$i == 5`.

Cvičení 6.11 – Continue
Přeskočte číslo 7 a vypište všechna ostatní od 1 do 10.

Cvičení 6.12 – Pole cen
Vypište pomocí foreach všechny ceny z pole: `[99, 149, 349]`

Cvičení 6.13 – Pole se jmény
Mějte pole jmen a vypište pro každé větu: `"Ahoj, Jméno!"`

Cvičení 6.14 – Mini validace
Pomocí cyklu zkontrolujte, zda je v poli číslo 7. (Vypište „nalezeno“.)

Lekce 7: Pole v PHP

Cíl lekce

Pochopit práci s poli v PHP, naučit se vytvářet a upravovat pole, pracovat s asociativními poli a využívat běžné funkce pro jejich zpracování.

Jednoduchá pole

Pole v PHP mohou být vytvořena dvěma způsoby:

```
$ovoce = array("jablko", "banán", "hruška");

$ovoce2 = ["kiwi", "pomeranč", "broskev"];
```

Přístup k prvkům probíhá pomocí indexu:

```
echo $ovoce[0]; // jablko
```

Přidávání a odebírání prvků

- Přidání na konec: `$pole[] = hodnota;`
- Odebrání posledního prvku: `array_pop($pole);`
- Odebrání prvního prvku: `array_shift($pole);`
- Přidání na začátek: `array_unshift($pole, hodnota);`

```
$cisla = [1, 2, 3];
$cisla[] = 4; // [1, 2, 3, 4]
array_pop($cisla); // [1, 2, 3]
```

Asociativní pole

Asociativní pole používají místo indexů pojmenované klíče:

```
$osoba = [
    "jmeno" => "Kristýna",
    "vek" => 18,
    "mesto" => "Brno"
];

echo $osoba["jmeno"]; // Kristýna
```

Víceúrovňová pole

Pole mohou obsahovat další pole:

```
$trida = [
    "A" => ["Petr", "David"],
    "B" => ["Lenka", "Ema"]
];

echo $trida["A"][1]; // David
```

Užitečné funkce

- `count($pole)` – počet prvků
- `in_array(hodnota, $pole)` – ověření existence
- `array_push($pole, hodnota)` – přidání prvku na konec
- `sort($pole)` – seřazení
- `implode(", ", $pole)` – spojení pole do textu

```
$ovoce = ["jablko", "banán", "kiwi"];
echo implode(", ", $ovoce); // jablko, banán, kiwi
```

Cvičení

Cvičení 7.1 – Jednoduché pole
Vytvořte pole tří vašich oblíbených jídel a vypište první.

Cvičení 7.2 – Přidání prvku
Přidejte nový prvek do pole měst.

Cvičení 7.3 – Odebrání prvku
Odeberte poslední prvek z pole čísel.

Cvičení 7.4 – Spojení prvků
Pomocí `implode` vypište všechna slova v poli.

Cvičení 7.5 – Asociativní pole
Uložte o sobě informace: jméno, věk, město. Vypište jméno.

Cvičení 7.6 – Přístup ke klíči
Vyberte a vypište město ze struktury osoby.

Cvičení 7.7 – Update hodnoty
Upravte věk v asociativním poli o 1.

Cvičení 7.8 – Víceúrovňové pole
Vypište druhého studenta z podskupiny „A“.

Cvičení 7.9 – Kontrola existence
Zjistěte, zda v poli `["pes", "kočka"]` existuje "kočka".

Cvičení 7.10 – Počet prvků
Vytvořte pole filmů a vypište počet prvků.

Cvičení 7.11 – Seřazení pole
Seřaďte pole čísel od nejmenšího po největší.

Cvičení 7.12 – Foreach + pole
Pomocí foreach vypište všechna zvířata v poli.

Cvičení 7.13 – Klíče asociativního pole
Vypište všechny klíče z pole osoba (pomocí `array_keys`).

Cvičení 7.14 – Hodnoty asociativního pole
Vypište všechny hodnoty z asociativního pole.

Cvičení 7.15 – Telefonní seznam
Vytvořte asociativní pole s kontakty: jméno → telefon. Poté vypište celý seznam.

Lekce 8: Formuláře (GET a POST)

Cíl lekce

Naučit se pracovat s HTML formuláři, pochopit rozdíl mezi metodami GET a POST, zpracovat odeslaná data v PHP a vyhnout se častým chybám.

Co je formulář

Formulář umožňuje uživateli poslat data na server. V HTML vypadá takto:

```
<form action="index.php" method="GET">
    <input type="text" name="jmeno">
    <button type="submit">Odeslat</button>
</form>
```

action – soubor, kam se data odešlou **method** – způsob odeslání (GET nebo POST)

Metoda GET

GET posílá data v URL adrese:

```
index.php?jmeno=Kristyna
```

V PHP si data přečteme takto:

```
$jmeno = $_GET["jmeno"];
echo $jmeno;
```

GET se hodí pro:

- vyhledávání
- filtrování
- jednoduché testy

Metoda POST

POST posílá data „skrytě“, není vidět v adresním řádku.
Používá se pro:

- přihlášení
- registraci
- citlivá data
- odesílání formulářů s větším obsahem

PHP:

```
$jmeno = $_POST["jmeno"];
```

Jednoduché zpracování

HTML:

```
<form method="POST">
    <input type="text" name="jmeno">
    <button>Odeslat</button>
</form>
```

PHP:

```
if (isset($_POST["jmeno"])) {
    echo "Ahoj, " . $_POST["jmeno"];
}
```

isset() chrání před chybou „Undefined index“.

Úprava odeslaných dat

Uživatel může napsat cokoli. Proto často používáme:

- `trim()` – odstraní mezery kolem textu
- `htmlspecialchars()` – ochrana před HTML
- `intval()` – převede na číslo

```php
$jmeno = trim($_POST["jmeno"]);
$jmeno = htmlspecialchars($jmeno);
```

Vícepoložkový formulář

```html
<form method="POST">
    Jméno: <input type="text" name="jmeno">
    Věk: <input type="number" name="vek">
    <button>Odeslat</button>
</form>
```

```php
if (!empty($_POST["jmeno"]) && !empty($_POST["vek"])) {
    echo "Jmenuji se " . $_POST["jmeno"];
    echo " a je mi " . $_POST["vek"] . " let.";
}
```

Cvičení

Cvičení 8.1 – Jednoduchý formulář
Vytvořte formulář, který po odeslání vypíše vaše jméno.

Cvičení 8.2 – GET metoda
Vytvořte formulář s metodou GET a vypište zadané město.

Cvičení 8.3 – POST metoda
Vytvořte přihlašovací formulář se jménem a heslem.

Cvičení 8.4 – Kontrola vyplnění
Pomocí `empty()` zkontrolujte, zda je pole vyplněno.

Cvičení 8.5 – Ošetření dat
Upravte odeslaný text pomocí `trim()` a `htmlspecialchars()`.

Cvičení 8.6 – Věk jako číslo
Pomocí `intval()` převedete vstup na číslo.

Cvičení 8.7 – Dva vstupy
Vytvořte formulář se jménem a příjmením a vypište celou větu.

Cvičení 8.8 – Výběr z možností
Vytvořte select s barvami a vypište vybranou barvu.

Cvičení 8.9 – Radio button
Vytvořte výběr pohlaví (muž/žena) a vypište odpověď.

Cvičení 8.10 – Checkboxy
Vytvořte seznam koníčků a vypište všechny vybrané.

Cvičení 8.11 – Email
Vytvořte pole pro e-mail a vypište ho po odeslání.

Cvičení 8.12 – Mini validace hesla
Pokud je délka hesla < 8, vypište „Slabé heslo“.

Cvičení 8.13 – Formulář s číslem
Vytvořte formulář, který vezme dvě čísla a vypíše jejich součet.

Cvičení 8.14 – Více polí najednou
Vytvořte formulář se jménem, věkem a městem.

Cvičení 8.15 – Osobní karta
Vytvořte formulář (jméno, věk, město, oblíbená barva). Po odeslání vypište přehlednou „vizitku“.

Lekce 9: Superglobální proměnné

Cíl lekce

Seznámit se s nejdůležitějšími superglobálními proměnnými v PHP, pochopit jejich využití a práci s daty, která přicházejí od uživatele nebo od serveru.

Co jsou superglobální proměnné

Superglobální proměnné jsou speciální pole, která jsou v PHP dostupná odkudkoli — včetně funkcí.
Mezi nejdůležitější patří:

- `$_GET`
- `$_POST`
- `$_REQUEST`
- `$_SERVER`
- `$_SESSION`
- `$_COOKIE`
- `$_FILES`

$_GET

Obsahuje data odeslaná metodou GET (v URL).

```
echo $_GET["jmeno"];
```

Typické použití:

- filtrování
- vyhledávání
- předávání parametrů v URL

$_POST

Obsahuje data odeslaná metodou POST.

```
echo $_POST["email"];
```

Použití:

- přihlašování
- registrace
- odesílání citlivých dat

$_REQUEST

Obsahuje kombinaci:

- $_GET
- $_POST
- $_COOKIE

```
$jmeno = $_REQUEST["jmeno"];
```

Pozor: doporučuje se používat jen pro rychlé testování.

$_SERVER

Obsahuje informace o serveru a aktuálním požadavku.
Příklady:

```
echo $_SERVER["REQUEST_METHOD"];  // GET nebo POST
echo $_SERVER["SERVER_NAME"];     // localhost
echo $_SERVER["PHP_SELF"];        // aktuální soubor
```

$_SESSION

Používá se pro ukládání dat mezi stránkami (přihlášení, košík...).

```
session_start();
$_SESSION["jmeno"] = "Kristýna";
echo $_SESSION["jmeno"];
```

Vyžaduje:

```
session_start();
```

$_COOKIE

Slouží k ukládání dat do prohlížeče.

```
setcookie("barva", "modra", time() + 3600);
echo $_COOKIE["barva"];
```

Data přetrvají i po zavření prohlížeče.

$_FILES

Používá se při uploadu souborů.

```
$_FILES["soubor"]["name"];
$_FILES["soubor"]["size"];
```

Soubory se vždy nahrávají přes POST.

Časté chyby

- **Undefined index** – přístup k neexistujícímu poli Řešení: `isset()` nebo `!empty()`
- Zapomenuté `session_start()`
- Použití `$_REQUEST` místo GET/POST → nejasné chování

Cvičení

Cvičení 9.1 – $_GET

Vypište hodnotu parametru `?jmeno=...`.

Cvičení 9.2 – $_POST
Vytvořte formulář, který odešle jméno metodou POST.

Cvičení 9.3 – isset()
Zkontrolujte, zda byl formulář odeslán pomocí `isset()`.

Cvičení 9.4 – $_REQUEST
Vyzkoušejte načtení dat přes `$_REQUEST`.

Cvičení 9.5 – $_SERVER Vypište typ požadavku (GET/POST).

Cvičení 9.6 – IP adresa
Vypište IP adresu pomocí `$_SERVER["REMOTE_ADDR"]`.

Cvičení 9.7 – SESSION ukládání
Uložte do session své jméno a vypište jej.

Cvičení 9.8 – SESSION změna hodnoty
Změňte hodnotu uloženou v session.

Cvičení 9.9 – COOKIE vytvoření
Nastavte cookie s barvou a vypište ji.

Cvičení 9.10 – COOKIE expirace
Nastavte cookie, která vyprší za 5 minut.

Cvičení 9.11 – Upload souboru
Vytvořte formulář s uploadem a vypište název nahraného souboru.

Cvičení 9.12 – Kontrola typu souboru
Zkontrolujte, zda nahraný soubor má příponu .jpg.

Cvičení 9.13 – $_SERVER informace
Vypište aktuální název souboru pomocí `$_SERVER["PHP_SELF"]`.

Cvičení 9.14 – Zkombinování GET a POST
Odešlete 2 formuláře – jeden GET, jeden POST – a zpracujte oba.

Cvičení 9.15 – Info panel
Vytvořte stránku, která vypíše:

- jméno uživatele (POST)
- všechny parametry v URL (GET)
- IP adresu
- aktuální čas

Lekce 10: Session a Cookies

Cíl lekce

Naučit se ukládat data mezi stránkami pomocí SESSION a COOKIES, pochopit rozdíl mezi těmito technologiemi a použít je v jednoduchých aplikacích.

Co je SESSION

SESSION umožňuje ukládat data na straně serveru a přiřadit je konkrétnímu uživateli.
Použití:

- přihlášení uživatele
- nákupní košík
- uložené informace mezi stránkami

SESSION musí být vždy spuštěna:

```
session_start();
```

Ukládání dat do SESSION

```
session_start();
$_SESSION["jmeno"] = "Kristýna";
```

Čtení:

```
echo $_SESSION["jmeno"];
```

Mazání a ukončení SESSION

Smazání jednoho údaje:

```
unset($_SESSION["jmeno"]);
```

Úplné zrušení session:

```
session_start();
session_destroy();
```

Co jsou COOKIES

COOKIE ukládá data do prohlížeče uživatele.
Typicky se používají pro:

- zapamatování preferencí
- zapamatování jazyka
- sledování přihlášení

Nastavení cookie:

```
setcookie("barva", "modra", time() + 3600);
```

Čtení cookie:

```
echo $_COOKIE["barva"];
```

Expirace, mazání a zabezpečení cookie

Mazání cookie změnou expirace:

```
setcookie("barva", "", time() - 3600);
```

Bezpečnostní volby:

- `secure` – funguje jen přes HTTPS
- `httponly` – cookie je dostupná jen pro PHP, ne pro JavaScript

SESSION vs COOKIE

- SESSION ukládá data na server → bezpečnější
- COOKIE ukládá data v prohlížeči → trvalé mezi relacemi
- SESSION vyžaduje `session_start()`
- `COOKIE fungují bez jakéhokoli startu`

Jednoduché přihlášení pomocí SESSION

```php
session_start();

if (isset($_POST["jmeno"])) {
    $_SESSION["uzivatel"] = $_POST["jmeno"];
}

if (isset($_SESSION["uzivatel"])) {
    echo "Přihlášen jako: " . $_SESSION["uzivatel"];
}
```

Cvičení

Cvičení 10.1 – Spuštění SESSION
Vytvořte soubor se session_start().

Cvičení 10.2 – Uložení jména do SESSION
Uložte své jméno do `$_SESSION`.

Cvičení 10.3 – Čtení SESSION
Vypište uložené jméno.

Cvičení 10.4 – Změna hodnoty
Změňte hodnotu v session na jiné jméno.

Cvičení 10.5 – Mazání hodnoty
Pomocí `unset()` smažte hodnotu v session.

Cvičení 10.6 – Zrušení celé session
Použijte `session_destroy()`.

Cvičení 10.7 – Základní cookie
Vytvořte cookie s oblíbenou barvou.

Cvičení 10.8 – Výpis cookie
Vypište uloženou barvu.

Cvičení 10.9 – Expirace cookie
Nastavte cookie, která vyprší za 10 sekund.

Cvičení 10.10 – Smazání cookie
Odstraňte cookie pomocí expirace v minulosti.

Cvičení 10.11 – Bezpečná cookie
Nastavte cookie s `httponly = true`.

Cvičení 10.12 – Jednoduché přihlášení
Vytvořte formulář, který uloží jméno do session.

Cvičení 10.13 – Kontrola přihlášení
Pokud existuje `$_SESSION["uzivatel"]`, vypište přihlášeného uživatele.

Cvičení 10.14 – Logout
Vytvořte tlačítko pro odhlášení → smaže session.

Cvičení 10.15 – Pamatuj si mě
Vytvořte přihlašovací formulář, který při zaškrtnutí „Pamatuj si mě“ uloží jméno do cookie a zobrazí ho při další návštěvě.

Lekce 11: Práce se soubory

Cíl lekce

Naučit se pracovat se soubory v PHP: číst, vytvářet, upravovat, mazat soubory a správně zpracovat upload od uživatele.

Čtení souborů

Jednoduché přečtení celého souboru:

```
$obsah = file_get_contents("data.txt");
echo $obsah;
```

Čtení řádek po řádku:

```
$file = fopen("data.txt", "r");

while (!feof($file)) {
    echo fgets($file);
}

fclose($file);
```

Zápis do souborů

Přepsání celého souboru:

```
file_put_contents("data.txt", "Nový obsah");
```

Přidání na konec (append):

```
file_put_contents("data.txt", "Další řádek\n", FILE_APPEND);
```

Adresáře

Kontrola, zda existuje složka:

```
if (!is_dir("uploads")) {
    mkdir("uploads");
}
```

Výpis souborů ve složce:

```
$files = scandir("uploads");
print_r($files);
```

Upload souboru

HTML formulář:

```
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="soubor">
    <button>Nahrát</button>
</form>
```

PHP zpracování:

```
if (isset($_FILES["soubor"])) {
    $nazev = $_FILES["soubor"]["name"];
    move_uploaded_file($_FILES["soubor"]["tmp_name"], "uploads/"
        . $nazev);
}
```

Informace o souboru

- název: `$_FILES["soubor"]["name"]`
- typ: `$_FILES["soubor"]["type"]`
- velikost: `$_FILES["soubor"]["size"]`
- dočasná cesta: `$_FILES["soubor"]["tmp`$_n ame$"]

Velikost v KB:

```
$kb = $_FILES["soubor"]["size"] / 1024;
```

Bezpečnost při uploadu

- povolte pouze určité přípony (jpg, png, txt…)
- kontrolujte velikost souboru
- nikdy neukládejte soubory do kořenové složky projektu

Ukázka kontroly typu:

```
$allowed = ["image/jpeg", "image/png"];

if (!in_array($_FILES["soubor"]["type"], $allowed)) {
    echo "Nepovolený soubor!";
}
```

Cvičení

Cvičení 11.1 – Vytvoření souboru
Vytvořte textový soubor a do něj zapište libovolný text.

Cvičení 11.2 – Čtení obsahu
Pomocí `file_get_contents` vypište obsah souboru.

Cvičení 11.3 – Přidání textu
Přidejte další řádek na konec souboru.

Cvičení 11.4 – Čtení po řádcích
Vypište soubor řádek po řádku.

Cvičení 11.5 – Kontrola složky
Ověřte, zda existuje složka „uploads“. Pokud ne, vytvořte ji.

Cvičení 11.6 – Výpis souborů
Vypište soubory ve složce „uploads“.

Cvičení 11.7 – Upload obrázku
Vytvořte formulář pro nahrání obrázku.

Cvičení 11.8 – Uložení souboru
Nahrajte soubor do složky „uploads“.

Cvičení 11.9 – Informace o souboru
Vypište velikost a typ nahraného souboru.

Cvičení 11.10 – Kontrola typu obrázku
Povolte pouze formáty JPG a PNG.

Cvičení 11.11 – Omezení velikosti
Zkontrolujte, zda velikost souboru není větší než 2 MB.

Cvičení 11.12 – Odstranění souboru
Smažte vybraný soubor pomocí `unlink()`.

Cvičení 11.13 – Log soubor
Vytvořte „log.txt“ a zapisujte do něj události.

Cvičení 11.14 – Počítadlo návštěv
V souboru udržujte počet návštěv stránky.

Cvičení 11.15 – Galerie obrázků
Nahrajte více obrázků a vytvořte jejich jednoduchý výpis na stránce.

Lekce 12: Práce s databází (MySQL)

Cíl lekce

Naučit se propojit PHP s databází MySQL, vykonat základní SQL dotazy, získat data z databáze a zobrazit je na webové stránce.

Co je MySQL

MySQL je relační databázový systém. Používá se pro:

- ukládání dat webových aplikací
- registrace/přihlášení uživatelů
- e-shopy
- blogy, systémy, portály

Ke správě databáze v XAMPP se používá:
phpMyAdmin → dostupné na adrese: `http://localhost/phpmyadmin`

Připojení k databázi – mysqli

Základní připojení k databázi:

```
$connection = mysqli_connect(
    "localhost",
    "root",
    "",
    "test_db"
);

if (!$connection) {
    die("Chyba připojení: " . mysqli_connect_error());
}
```

root = výchozí uživatel Heslo je u XAMPP většinou prázdné.

SELECT – načtení dat

```
$result = mysqli_query(
    $connection,
    "SELECT * FROM uzivatele"
);

while ($row = mysqli_fetch_assoc($result)) {
    echo $row["jmeno"] . "<br>";
}
```

mysqli_fetch_assoc() vrací jednotlivé řádky jako asociativní pole.

INSERT – vložení dat

```
$sql = "INSERT INTO uzivatele (jmeno, vek)
        VALUES ('Eva', 22)";

mysqli_query($connection, $sql);
```

UPDATE a DELETE

UPDATE:

```
$sql = "UPDATE uzivatele
        SET vek = 25
        WHERE id = 1";
mysqli_query($connection, $sql);
```

DELETE:

```
$sql = "DELETE FROM uzivatele WHERE id = 3";
mysqli_query($connection, $sql);
```

Vytvoření databáze a tabulky

SQL pro vytvoření databáze:

```
CREATE DATABASE test_db;
```

SQL pro vytvoření tabulky:

```
CREATE TABLE uzivatele (
    id INT AUTO_INCREMENT PRIMARY KEY,
    jmeno VARCHAR(50),
    vek INT
);
```

Nejčastější chyby

- špatný název databáze – „Unknown database“
- použití českých znaků bez nastavení utf8
- nezapnutý MySQL server v XAMPP
- zapomenuté uvození textu (’ ’)

Doporučuje se spustit:

```
mysqli_set_charset($connection, "utf8");
```

Cvičení

Cvičení 12.1 – Připojení k databázi
Vytvořte PHP skript, který se připojí k databázi.

Cvičení 12.2 – Vytvoření databáze
Pomocí phpMyAdmin vytvořte databázi `skola`.

Cvičení 12.3 – Vytvoření tabulky
Vytvořte tabulku `studenti` s poli: id, jmeno, rocnik.

Cvičení 12.4 – INSERT
Vložte do tabulky studenta: „Petr“, 3. ročník.

Cvičení 12.5 – SELECT všech dat
Vypište všechny studenty.

Cvičení 12.6 – SELECT s filtrem
Vypište jen studenty 3. ročníku.

Cvičení 12.7 – UPDATE
Změňte jméno studenta s id=1.

Cvičení 12.8 – DELETE
Smažte studenta s id=3.

Cvičení 12.9 – Výpis jedné hodnoty
Vypište jen jména všech studentů.

Cvičení 12.10 – Ošetření znaků
Nastavte kódování na UTF-8.

Cvičení 12.11 – Připojovací chyba
Simulujte chybu – schválně změňte název databáze a vypište error.

Cvičení 12.12 – Vkládání z formuláře
Vložte studenta do databáze pomocí POST formuláře.

Cvičení 12.13 – Výpis dat v tabulce (HTML)
Zobrazte studenty v HTML tabulce.

Cvičení 12.14 – Počítání výsledků
Vypište počet studentů v databázi.

Cvičení 12.15 – Jednoduchý seznam studentů
Vytvořte stránku, která umožní:

- vložit studenta
- zobrazit všechny studenty
- smazat studenta

Lekce 13: PDO a prepared statements

Cíl lekce

Naučit se moderní a bezpečný způsob práce s databází pomocí PDO, používat připravené dotazy (prepared statements) a ošetřit chyby pomocí výjimek.

Co je PDO

PDO (PHP Data Objects) je moderní rozhraní pro práci s databázemi v PHP.
Výhody PDO:

- jednotné rozhraní – funguje s MySQL, SQLite, PostgreSQL...
- podpora **prepared statements** = ochrana proti SQL injection
- lepší práce s chybami (výjimky)
- čistší a kratší kód než mysqli

My budeme pracovat s PDO + MySQL.

Připojení k databázi pomocí PDO

```
try {
    $pdo = new PDO(
        "mysql:host=localhost;dbname=test_db;charset=utf8",
        "root",
        ""
    );

    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION
       );
    echo "Připojeno!";
}
catch (PDOException $e) {
    echo "Chyba: " . $e->getMessage();
}
```

ATTR_ERRMODE nastaví, aby PDO házelo výjimky při chybě.

Prepared SELECT

```php
$stmt = $pdo->prepare(
    "SELECT * FROM uzivatele WHERE vek > ?"
);

$stmt->execute([18]);

$rows = $stmt->fetchAll();

foreach ($rows as $row) {
    echo $row["jmeno"] . "<br>";
}
```

Výhoda: parametr se vloží bezpečně a neprovádí se přímo v SQL.

Prepared INSERT

```php
$stmt = $pdo->prepare(
    "INSERT INTO uzivatele (jmeno, vek)
     VALUES (:name, :age)"
);

$stmt->execute([
    ":name" => "Jana",
    ":age"  => 21
]);
```

Prepared UPDATE a DELETE

UPDATE:

```
$stmt = $pdo->prepare(
    "UPDATE uzivatele SET vek = ? WHERE id = ?"
);
$stmt->execute([25, 1]);
```

DELETE:

```
$stmt = $pdo->prepare(
    "DELETE FROM uzivatele WHERE id = ?"
);
$stmt->execute([3]);
```

Výjimky a řešení chyb

Pokud nastane chyba v dotazu, PDO vyvolá výjimku typu:

- `PDOException`

Lze ji zachytit:

```
try {
    $pdo->query("SELECT * FROM neexistuje");
}
catch (PDOException $e) {
    echo "Chyba SQL: " . $e->getMessage();
}
```

Cvičení

Cvičení 13.1 – Připojení k DB
Napište skript, který se pomocí PDO připojí k databázi.

Cvičení 13.2 – Vytvoření tabulky
Vytvořte pomocí phpMyAdmin tabulku `produkty` (id, nazev, cena).

Cvičení 13.3 – Prepared SELECT
Vyberte všechny produkty dražší než 100 Kč.

Cvičení 13.4 – Prepared INSERT
Vložte nový produkt pomocí připraveného dotazu.

Cvičení 13.5 – INSERT s pojmenovanými parametry
Použijte `:nazev`, `:cena`.

Cvičení 13.6 – Výpis výsledků v HTML tabulce
Zobrazte produkty v tabulce.

Cvičení 13.7 – UPDATE produktu
Změňte cenu produktu s id = 1.

Cvičení 13.8 – DELETE produktu
Smažte produkt s id = 3.

Cvičení 13.9 – Kontrola vstupů
Ošetřete situaci, kdy uživatel zadá prázdné hodnoty.

Cvičení 13.10 – Try/catch
Simulujte chybu v SQL a zachyťte výjimku.

Cvičení 13.11 – LIMIT + OFFSET
Vyberte prvních 5 produktů.

Cvičení 13.12 – Vyhledávání
Vytvořte formulář pro vyhledání produktů podle názvu.

Cvičení 13.13 – Průměrná cena
Vypočítejte průměrnou cenu všech produktů.

Cvičení 13.14 – Řazení výsledků
Vyberte produkty seřazené podle ceny (vzestupně).

Cvičení 13.15 – Správa produktů (CRUD)
Vytvořte jednoduchou aplikaci:

- výpis všech produktů
- přidání produktu
- úprava produktu
- smazání produktu

Lekce 14: CRUD aplikace

Cíl lekce

Naučit se vytvořit kompletní CRUD aplikaci (Create, Read, Update, Delete) pomocí PHP, PDO a MySQL. Studenti si procvičí práci s databází, formuláři, bezpečnými dotazy a zpracováním vstupů.

Co znamená CRUD

CRUD = základní operace nad daty:

- **Create** – vytvořit nový záznam
- **Read** – načíst a zobrazit data
- **Update** – upravit existující záznam
- **Delete** – smazat záznam

Tuto logiku používá každý web (e-shop, blog, správa uživatelů...).

Databázová tabulka

Vytvořte tabulku „produkty“:

```
CREATE TABLE produkty (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nazev VARCHAR(100),
    cena DECIMAL(10,2)
);
```

READ – výpis dat

```
$stmt = $pdo->query("SELECT * FROM produkty");

$produkty = $stmt->fetchAll();
?>
<table border="1">
<tr><th>Název</th><th>Cena</th></tr>
<?php foreach ($produkty as $p): ?>
<tr>
  <td><?= $p["nazev"] ?></td>
  <td><?= $p["cena"] ?></td>
</tr>
<?php endforeach; ?>
</table>
```

CREATE – přidání nového produktu

HTML formulář:

```
<form method="POST">
    Název: <input name="nazev"><br>
    Cena: <input name="cena"><br>
    <button>Přidat</button>
</form>
```

PHP logika:

```
if ($_POST) {
    $stmt = $pdo->prepare(
        "INSERT INTO produkty (nazev, cena)
         VALUES (?, ?)"
    );

    $stmt->execute([$_POST["nazev"], $_POST["cena"]]);
}
```

UPDATE – úprava produktu

Formulář + úprava hodnot:

```
$id = $_GET["id"];

$stmt = $pdo->prepare("SELECT * FROM produkty WHERE id = ?");
$stmt->execute([$id]);
$produkt = $stmt->fetch();
?>

<form method="POST">
    Název: <input name="nazev" value="<?= $produkt['nazev'] ?>"
       ><br>
    Cena: <input name="cena" value="<?= $produkt['cena'] ?>"><br
       >
    <button>Uložit</button>
</form>
<?php

if ($_POST) {
    $stmt = $pdo->prepare(
        "UPDATE produkty SET nazev=?, cena=? WHERE id=?"
    );
    $stmt->execute([$_POST["nazev"], $_POST["cena"], $id]);
}
```

DELETE – odstranění záznamu

```
$id = $_GET["id"];

$stmt = $pdo->prepare("DELETE FROM produkty WHERE id=?");
$stmt->execute([$id]);
```

Bezpečnost v CRUD

- vždy používat prepared statements
- validovat vstupy (cena musí být číslo)
- chránit akce DELETE potvrzením
- skrýt akce za přihlášením (session)

Cvičení

Cvičení 14.1 – Vytvoření tabulky
Vytvořte tabulku `knihy` (id, nazev, autor, cena).

Cvičení 14.2 – Výpis knih
Načtěte všechny knihy a zobrazte je v HTML tabulce.

Cvičení 14.3 – Přidání knihy
Vytvořte formulář pro přidání nové knihy.

Cvičení 14.4 – Vkládání pomocí PDO
Uložte novou knihu do databáze.

Cvičení 14.5 – Úprava knihy
Umožněte upravit název, autora nebo cenu.

Cvičení 14.6 – Smazání knihy
Přidejte tlačítko „Smazat“.

Cvičení 14.7 – Potvrzení mazání
Před smazáním zobrazte potvrzovací dotaz.

Cvičení 14.8 – Vyhledávání
Umožněte vyhledávat podle názvu.

Cvičení 14.9 – Řazení
Seřaďte knihy podle ceny.

Cvičení 14.10 – Formátování cen
Zobrazujte ceny s `number_format()`.

Cvičení 14.11 – Chybové hlášky
Vypište chybu, když chybí název nebo autor.

Cvičení 14.12 – Validace
Cena musí být číslo — ošetřete vstup.

Cvičení 14.13 – Pagination
Zobrazte 5 knih na stránku.

Cvičení 14.14 – Kategorie
Přidejte ke knihám nový sloupec „žánr“.

Cvičení 14.15 – Knihovna
Vytvořte aplikaci, která umožní:

- zobrazit knihy
- přidat knihu
- upravit knihu
- smazat knihu
- vyhledávat knihy

Lekce 15: Validace formulářů

Cíl lekce

Naučit se validovat uživatelské vstupy, zobrazovat chyby, filtrovat a ošetřovat text, čísla, emaily a citlivá data. Validace je základní první krok k ochraně aplikace.

Proč je validace důležitá

- chrání aplikaci před chybami
- chrání databázi před nevalidními daty
- zabraňuje útokům (XSS, SQLi)
- zvyšuje kvalitu dat

Validace se dělí na:

- **serverovou** (PHP — povinná!)
- **klientskou** (HTML5 — doplňková)

Základní validace

Příklad formuláře:

```
<form method="POST">
    Jméno: <input name="jmeno">
    <button>Odeslat</button>
</form>
```

PHP validace:

```
$chyby = [];

if ($_POST) {
    if (empty($_POST["jmeno"])) {
        $chyby[] = "Jméno je povinné.";
    }

    if (strlen($_POST["jmeno"]) < 3) {
        $chyby[] = "Jméno musí mít alespoň 3 znaky.";
    }
}
```

Výpis chyb:

```
foreach ($chyby as $ch) {
    echo "<p style='color:red'>$ch</p>";
}
```

Validace emailu

```
if (!filter_var($_POST["email"], FILTER_VALIDATE_EMAIL)) {
    $chyby[] = "Email není ve správném formátu.";
}
```

Validace čísel

```php
if (!is_numeric($_POST["cena"])) {
    $chyby[] = "Cena musí být číslo.";
}

if ($_POST["cena"] <= 0) {
    $chyby[] = "Cena musí být větší než 0.";
}
```

Ochrana proti XSS

Nebezpečný vstup:

```php
<script>alert("hack")</script>
```

Bezpečný výstup:

```php
$bezpecne = htmlspecialchars($_POST["text"], ENT_QUOTES);
```

Tím zabráníme spuštění JavaScriptu.

Trim a sanitizace

```php
$text = trim($_POST["text"]);
$text = filter_var($text, FILTER_SANITIZE_STRING);
```

Trim odstraní mezery, sanitizace odstraní nebezpečné znaky.

Cvičení

Cvičení 15.1 – Povinné pole
Ověřte, že jméno není prázdné.

Cvičení 15.2 – Minimální délka
Vyžadujte jméno alespoň 5 znaků.

Cvičení 15.3 – Kontrola emailu
Validujte email pomocí `FILTER_VALIDATE_EMAIL`.

Cvičení 15.4 – Kontrola ceny
Cena musí být číslo > 0.

Cvičení 15.5 – Vypsání všech chyb
Vypište chyby červeně.

Cvičení 15.6 – Trim vstupů
Odstraňte mezery na začátku a konci.

Cvičení 15.7 – Sanitizace
Použijte `FILTER_SANITIZE_STRING`.

Cvičení 15.8 – XSS ochrana
Použijte `htmlspecialchars()`.

Cvičení 15.9 – Validace dvou vstupů
Validujte jméno + věk.

Cvičení 15.10 – Porovnání hesel
Hesla musí být dvě a stejná.

Cvičení 15.11 – Vlastní chybová hláška
Zobrazte „Pole nesmí být prázdné“.

Cvičení 15.12 – Přesměrování po úspěchu
Po validaci přesměrujte na jinou stránku.

Cvičení 15.13 – Validace délky popisu
Min. 10 znaků, max. 200 znaků.

Cvičení 15.14 – Filtrace čísel
Použijte `FILTER_SANITIZE_NUMBER_FLOAT`.

Cvičení 15.15 – Přihlášení
Vytvořte formulář s validací:

- email
- heslo (min. 6 znaků)
- zobrazení chyb

Lekce 16: Include, Require a Templating

Cíl lekce

Naučit se rozdělit projekt do více souborů, používat include/require, oddělit HTML šablony od PHP logiky a tvořit opakovaně použitelné části webu.

Proč rozdělovat projekt

- přehlednější struktura
- snadná údržba
- opakované komponenty (hlavička, patička, menu)
- menší riziko chyb
- čistší HTML i PHP

Velké projekty nikdy netvoříme v jednom souboru.

Include

Načte soubor do aktuálního skriptu.

```
include "header.php";
```

Pokud soubor neexistuje → zobrazí varování, ale skript pokračuje dál.

Require

Require je přísnější verze include:

```
require "database.php";
```

```
Pokud soubor chybí     skript se ukončí s fatální chybou.
```

Používá se zejména pro:

- databázové připojení
- konfigurace

Include_once a Require_once

Zabrání více vložení stejného souboru:

```
include_once "header.php";
require_once "config.php";
```

Doporučená struktura projektu

```
/projekt
    /templates
        header.php
        footer.php
    /scripts
        db.php
    index.php
    detail.php
```

/templates → HTML části **/scripts** → logika (připojení k DB, funkce)

Jednoduché templating řešení v PHP

Vytvořte `header.php`:

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>Moje stránka</title>
</head>
<body>
```

Vytvořte `footer.php`:

```
</body>
</html>
```

Použití:

```
require "templates/header.php";
echo "<h1>Vítejte!</h1>";
require "templates/footer.php";
```

Oddělení logiky od šablony

Soubor `produkty.php`:

```
require "scripts/db.php";

$stmt = $pdo->query("SELECT * FROM produkty");
$data = $stmt->fetchAll();
```

Soubor `produkty.view.php`:

```
<table>
<?php foreach ($data as $p): ?>
<tr>
    <td><?= $p["nazev"] ?></td>
    <td><?= $p["cena"] ?></td>
</tr>
<?php endforeach; ?>
</table>
```

Mini templating engine (pokročilé)

Vytvoření render funkce:

```
function render($template, $data = []) {
    extract($data);
    include $template;
}
```

Použití:

```
render("produkty.view.php", ["data" => $data]);
```

Cvičení

Cvičení 16.1 – Použití include

Vytvořte vlastní `header.php` a `footer.php`.

Cvičení 16.2 – Require

Použijte require pro připojení databáze.

Cvičení 16.3 – Include_once

Zabraňte dvojímu vložení souboru.

Cvičení 16.4 – Struktura projektu
Vytvořte projekt přesně podle doporučené struktury.

Cvičení 16.5 – Jednoduchý templating
Načtěte šablonu header a footer.

Cvičení 16.6 – Oddělení logiky
Načtěte data v jednom souboru a zobrazte je v jiném.

Cvičení 16.7 – Menu komponenta
Vytvořte `menu.php` a zahrňte ho do hlavičky.

Cvičení 16.8 – Galerie šablona
Oddělte HTML výpis obrázků do šablony.

Cvičení 16.9 – Parametry pro view
Předejte proměnnou do šablony přes `extract()`.

Cvičení 16.10 – Mini templating engine
Vytvořte funkci render().

Cvičení 16.11 – Vnořené šablony
Zavolejte view uvnitř jiné šablony.

Cvičení 16.12 – Více stránek
Vytvořte 3 stránky využívající stejný header a footer.

Cvičení 16.13 – Proměnné v šablonách
Použijte PHP proměnné v HTML souboru.

Cvičení 16.14 – Dynamický titul
Nastavte proměnnou `$title` a vložte ji do `header.php`.

Cvičení 16.15 – Web s šablonami
Vytvořte aplikaci s:

- header + footer + menu
- stránkou s výpisem produktů
- stránkou s detailem produktu

Lekce 17: Základy OOP v PHP

Cíl lekce

Seznámit se s objektově orientovaným programováním v PHP: třídy, objekty, vlastnosti, metody a konstruktory.

Co je OOP

OOP (Object-Oriented Programming) je způsob psaní kódu, kdy vytváříme objekty, které mají:

- vlastnosti (data)
- metody (funkce)

Výhody:

- lepší organizace kódu
- znovupoužitelnost
- přehledné větší projekty

Třída a objekt

Ukázka třídy:

```
class Osoba {
    public $jmeno;
    public $vek;

    public function pozdrav() {
        return "Ahoj, já jsem " . $this->jmeno;
    }
}

$osoba = new Osoba();
$osoba->jmeno = "Eva";

echo $osoba->pozdrav();
```

Konstruktor

Konstruktor se volá automaticky při vytvoření objektu.

```
class Auto {
    public $znacka;

    public function __construct($znacka) {
        $this->znacka = $znacka;
    }
}

$a = new Auto("Škoda");
echo $a->znacka;
```

Public, Private, Protected

- **public** – přístupné odkudkoliv
- **private** – přístupné pouze uvnitř třídy
- **protected** – v třídě a jejích potomcích

```
class User {
    private $heslo;

    public function nastavHeslo($h) {
        $this->heslo = $h;
    }
}
```

Gettery a settery

```
class Banka {
    private $zustatek = 0;

    public function vlozit($castka) {
        $this->zustatek += $castka;
    }

    public function getZustatek() {
        return $this->zustatek;
    }
}
```

Dědičnost

```
class Zvire {
    public function zvuk() {
        return "nějaký zvuk";
    }
}

class Pes extends Zvire {
    public function zvuk() {
        return "haf!";
    }
}

$p = new Pes();
echo $p->zvuk();
```

Cvičení

Cvičení 17.1 – Vytvoření třídy
Vytvořte třídu `Student` s vlastnostmi jméno a ročník.

Cvičení 17.2 – Objekt
Vytvořte dva objekty třídy Student.

Cvičení 17.3 – Metoda pozdrav
Přidejte metodu `pozdrav()`.

Cvičení 17.4 – Konstruktor
Vytvořte konstruktor pro jméno a ročník.

Cvičení 17.5 – Private vlastnost
Vytvořte private vlastnost a nastavovací metodu.

Cvičení 17.6 – Getter
Vytvořte getter pro private hodnotu.

Cvičení 17.7 – Bankovní účet
Simulujte vklad a výpis zůstatku.

Cvičení 17.8 – Dědičnost
Vytvořte třídu Zvíře a třídu Kočka, která mění metodu zvuk.

Cvičení 17.9 – Více tříd
Vytvořte třídu Auto a SportAuto, které dědí vlastnosti.

Cvičení 17.10 – Počítadlo
Třída, která počítá kolikrát byla metoda zavolána.

Cvičení 17.11 – Validace v OOP
Třída Uživatel, která při nastavování hesla kontroluje délku.

Cvičení 17.12 – OOP + PDO základ
Vytvořte třídu Database s metodou `connect()`.

Cvičení 17.13 – OOP formulář
Třída formuláře se třemi vlastnostmi.

Cvičení 17.14 – Vypisovač
Třída, která vypisuje seznam položek pomocí metody `render()`.

Cvičení 17.15 – Třída Produkt
Vytvořte třídu Produkt:

- vlastnosti: název, cena
- konstruktor
- metoda zobrazit()

Lekce 18: Bezpečnost v PHP

Cíl lekce

Naučit se chránit PHP aplikace před nejčastějšími útoky: XSS, SQL Injection, úniky dat, CSRF a práci s hesly.

Proč je bezpečnost důležitá

- každý formulář je potenciální vstup útočníka
- databáze obsahuje citlivá data
- vstupy mohou obsahovat skripty nebo útoky
- špatná validace vede k útokům

Bezpečnost je základní součást každé webové aplikace.

XSS – Cross-Site Scripting

Nebezpečný vstup:

```
<script>alert("hack");</script>
```

Pokud se zobrazí bez úpravy, JavaScript se spustí.
Ochrana:

```
echo htmlspecialchars($text, ENT_QUOTES);
```

SQL Injection

Nebezpečný vstup:

```
' OR 1=1 --
```

Útočník může obejít přihlášení nebo upravit data.

Bezpečné řešení → **prepared statements**:

```
$stmt = $pdo->prepare(
  "SELECT * FROM uzivatele WHERE email = ?"
);
$stmt->execute([$email]);
```

Hashování hesel

Nikdy neukládat heslo jako text!

Bezpečné řešení:

```
$hash = password_hash($heslo, PASSWORD_DEFAULT);
```

Ověření hesla:

```
if (password_verify($heslo, $hash)) {
    echo "Přihlášen";
}
```

CSRF – mezistránkový útok

Řešení → token ve formuláři:

```
$_SESSION["token"] = bin2hex(random_bytes(32));
```

Formulář:

```
<input type="hidden" name="token" value="<?= $_SESSION['token']
   ?>">
```

Kontrola:

```
if ($_POST["token"] !== $_SESSION["token"]) {
    die("Neplatný token!");
}
```

Bezpečnostní zásada: Escapovat výstup

Každý text z databáze → zobrazovat takto:

```
echo htmlspecialchars($row["nazev"]);
```

Bezpečné nahrávání souborů

- kontrolovat MIME typ
- kontrolovat příponu
- kontrolovat velikost
- generovat nové názvy souborů

Příklad:

```
$allowed = ["image/jpeg", "image/png"];
if (!in_array($_FILES["file"]["type"], $allowed)) {
    die("Nepovolený typ!");
}
```

Cvičení

Cvičení 18.1 – XSS ochrana
Ošetřete vstup pomocí `htmlspecialchars()`.

Cvičení 18.2 – SQL Injection
Upravte login formulář, aby používal prepared statements.

Cvičení 18.3 – Hashování
Uložte heslo s `password_hash()`.

Cvičení 18.4 – Ověření hesla
Použijte `password_verify()`.

Cvičení 18.5 – Bezpečné nahrávání
Povolte pouze obrázky PNG a JPG.

Cvičení 18.6 – CSRF token
Přidejte do formuláře bezpečnostní token.

Cvičení 18.7 – Escapování výstupu
Escapujte všechny výpisy z databáze.

Cvičení 18.8 – Limity velikosti
Zaveďte limit velikosti uploadu (např. 2 MB).

Cvičení 18.9 – Chybný typ souboru
Zobrazte chybovou hlášku při nahrání .exe.

Cvičení 18.10 – Bezpečné přihlášení
Vytvořte přihlašovací systém s hashováním hesla.

Cvičení 18.11 – Kontrola prázdných polí
Validujte vstupy, aby nebyly prázdné.

Cvičení 18.12 – Sanitizace vstupů
Použijte `FILTER_SANITIZE_STRING`.

Cvičení 18.13 – SQL chyby
Zachyťte SQL chybu pomocí try/catch.

Cvičení 18.14 – Bezpečné odhlášení
Odstraňte session data bezpečně.

Cvičení 18.15 – Bezpečný login
Vytvořte přihlášení s:

- hashováním hesla
- SQL prepared statements
- XSS ochranou
- CSRF tokenem

Lekce 19: JSON, API a práce s daty

Cíl lekce

Naučit se pracovat s JSONem, vytvářet jednoduché API v PHP, odesílat data ve formátu JSON a číst data z externích API.

Co je JSON

JSON = JavaScript Object Notation.
Je to moderní formát pro:

- přenos dat mezi serverem a klientem
- ukládání strukturovaných dat
- komunikaci mezi aplikacemi

Příklad JSONu:

```
{
  "jmeno": "Karel",
  "vek": 22
}
```

PHP: JSON encode/decode

PHP pole → JSON:

```
$data = ["jmeno" => "Eva", "vek" => 20];
$json = json_encode($data);

echo $json;
```

JSON → PHP pole:

```
$json = '{"jmeno":"Eva","vek":20}';
$data = json_decode($json, true);

echo $data["jmeno"];
```

Tvorba jednoduchého API v PHP

Soubor `api.php`:

```php
header("Content-Type: application/json");

$data = [
    ["id" => 1, "nazev" => "Jablko"],
    ["id" => 2, "nazev" => "Hruška"]
];

echo json_encode($data);
```

Když stránku otevřete: → vrátí JSON místo HTML.

API: výpis dat z databáze

```php
header("Content-Type: application/json");

$stmt = $pdo->query("SELECT * FROM produkty");
$data = $stmt->fetchAll(PDO::FETCH_ASSOC);

echo json_encode($data);
```

Čtení dat z externího API

Pomocí file_get_contents:

```php
$json = file_get_contents("https://api.kanye.rest/");
$data = json_decode($json, true);

echo $data["quote"];
```

Poznámka: některá API vyžadují API klíč.

API pomocí POST

Klient odešle JSON:

```
$input = json_decode(file_get_contents("php://input"), true);

echo "Přijato: " . $input["zprava"];
```

Tímto vzniká jednoduchý backend endpoint.

Bezpečnost API

- validovat vstupy
- omezit velikost vstupu
- používat HTTPS
- skrýt citlivé údaje (API klíče)
- escapovat výstup

Cvičení

Cvičení 19.1 – JSON encode
Převeďte pole na JSON.

Cvičení 19.2 – JSON decode
Převeďte JSON na pole a vypište hodnoty.

Cvičení 19.3 – Simple API
Vytvořte jednoduché API vracející seznam jmen.

Cvičení 19.4 – API s databází
Vytvořte API vracející produkty z tabulky produkty.

Cvičení 19.5 – API detail
Vytvořte API vracející jeden produkt podle ID.

Cvičení 19.6 – POST endpoint
Vytvořte API, které přijme JSON a vrátí odpověď.

Cvičení 19.7 – Validace JSONu
Pokud JSON není platný → vraťte chybu.

Cvičení 19.8 – Externí API
Načtěte data z veřejného API.

Cvičení 19.9 – Filtrace dat
Z API s produkty vyberte jen ty dražší než 100 Kč.

Cvičení 19.10 – Uložení JSON do souboru
Vytvořte JSON soubor s produkty.

Cvičení 19.11 – Čtení JSON souboru
Načtěte JSON soubor a vypište data.

Cvičení 19.12 – Error handling
V API vracejte chybu při neplatném ID.

Cvičení 19.13 – Parametry v URL
Předejte parametr: `api.php?minCena=50`.

Cvičení 19.14 – Minimální API dokumentace
Popište, co vaše API vrací.

Cvičení 19.15 – Mini projekt: API pro knihy
Vytvořte API, které umožní:

- výpis knih
- výpis jedné knihy podle ID
- přidání knihy pomocí POST JSON

Lekce 20: Závěrečný projekt

Cíl lekce

Navrhnout a vytvořit kompletní webovou aplikaci v PHP, která kombinuje všechny technologie kurzu: PHP, PDO, CRUD, validaci, bezpečnost, templating, OOP a API.

Popis projektu

Vytvoříte aplikaci typu **Správa položek** (např. knihy, produkty, recepty, filmy...). Aplikace bude mít tyto části:

- registrace uživatele (hashované heslo)
- přihlášení (validace, session)
- výpis položek (READ)
- přidání nové položky (CREATE)
- úprava položky (UPDATE)
- smazání položky (DELETE)
- zabezpečení proti XSS, SQL injection
- API endpoint vracející položky ve formátu JSON
- šablony (header, footer, layout)
- OOP struktura pro připojení k databázi

Doporučená složková struktura

```
/projekt
    /templates
        header.php
        footer.php
        layout.php
    /scripts
        db.php
        auth.php
        validator.php
    /classes
        Database.php
        Item.php
    /api
        items.php
    index.php
    login.php
    register.php
    add.php
    edit.php
    delete.php
```

Registrace uživatele

```php
if ($_POST) {
    $email = $_POST["email"];
    $heslo = password_hash($_POST["heslo"], PASSWORD_DEFAULT);

    $stmt = $pdo->prepare(
        "INSERT INTO uzivatele (email, heslo) VALUES (?, ?)"
    );
    $stmt->execute([$email, $heslo]);
}
```

Tabulka `uzivatele`:

```sql
CREATE TABLE uzivatele (
    id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(100),
    heslo VARCHAR(255)
);
```

Přihlášení

```php
$stmt = $pdo->prepare(
    "SELECT * FROM uzivatele WHERE email=?"
);
$stmt->execute([$_POST["email"]]);
$user = $stmt->fetch();

if ($user && password_verify($_POST["heslo"], $user["heslo"])) {
    $_SESSION["user_id"] = $user["id"];
    header("Location: index.php");
}
```

CRUD položek

CREATE:

```
$stmt = $pdo->prepare(
  "INSERT INTO items (nazev, popis) VALUES (?, ?)"
);
$stmt->execute([$nazev, $popis]);
```

READ:

```
$stmt = $pdo->query("SELECT * FROM items");
$items = $stmt->fetchAll();
```

UPDATE:

```
$stmt = $pdo->prepare(
  "UPDATE items SET nazev=?, popis=? WHERE id=?"
);
$stmt->execute([$nazev, $popis, $id]);
```

DELETE:

```
$stmt = $pdo->prepare(
  "DELETE FROM items WHERE id=?"
);
$stmt->execute([$id]);
```

Templating (šablony)

```
<?php include "templates/header.php"; ?>

<h1><?= $title ?></h1>
<?= $content ?>

<?php include "templates/footer.php"; ?>
```

Soubor `layout.php` slouží k sjednocení vzhledu všech stránek.

API endpoint (JSON)

Soubor `/api/items.php`:

```php
header("Content-Type: application/json");

$stmt = $pdo->query("SELECT * FROM items");
echo json_encode($stmt->fetchAll());
```

OOP třída pro položky

```php
class Item {
    public $nazev;
    public $popis;

    public function __construct($nazev, $popis) {
        $this->nazev = $nazev;
        $this->popis = $popis;
    }
}
```

Bezpečnostní požadavky

- hashování hesel
- escapování HTML výstupu
- validace vstupů
- prepared statements
- session ochrana
- kontrola přihlášení
- CSRF token u formulářů

Cvičení

Cvičení 20.1 – Návrh databáze
Návrh tabulek: uzivatele a items.

Cvičení 20.2 – Registrace
Implementujte registraci s hashováním hesla.

Cvičení 20.3 – Login
Implementujte přihlášení pomocí session.

Cvičení 20.4 – Výpis položek
Načtěte položky z databáze.

Cvičení 20.5 – Přidání položky
Vytvořte formulář a validaci.

Cvičení 20.6 – Úprava položky
Implementujte editaci položky.

Cvičení 20.7 – Smazání položky
Vyřešte bezpečné mazání.

Cvičení 20.8 – Šablony
Použijte `layout.php` pro všechny stránky.

Cvičení 20.9 – Escapování výstupu
Chraňte aplikaci před XSS.

Cvičení 20.10 – Validace vstupů
Ověřte délku názvu a popisu.

Cvičení 20.11 – API výpis
Vytvořte API vracející JSON seznam položek.

Cvičení 20.12 – API detail položky
Vytvořte endpoint `api/item.php?id=`.

Cvičení 20.13 – OOP třída Item
Použijte třídu k reprezentaci položky.

Cvičení 20.14 – Ochrana formulářů
Implementujte CSRF token u formuláře.

Cvičení 20.15 – Kompletní aplikace
Poskládejte vše dohromady a vytvořte funkční web.