

# Php - 2. část

Ing. Martin Dostal, Ph.D.

[madostal@kiv.zcu.cz](mailto:madostal@kiv.zcu.cz)

[github.com/madostal/kiv-web](https://github.com/madostal/kiv-web)

# Obsah

- základní informace
- opakování
- cookies
- session (relace)
- přihlašování uživatelů
- OOP
- databáze - manipulace s databází

# Php - opakování

- Jaké soubory se automaticky otevřou po přístupu na doménu nebo do adresáře?
- Jak se zpracovává php soubor a co se posílá ke klientovi?
- Jak interpret pozná, že má zpracovávat php?

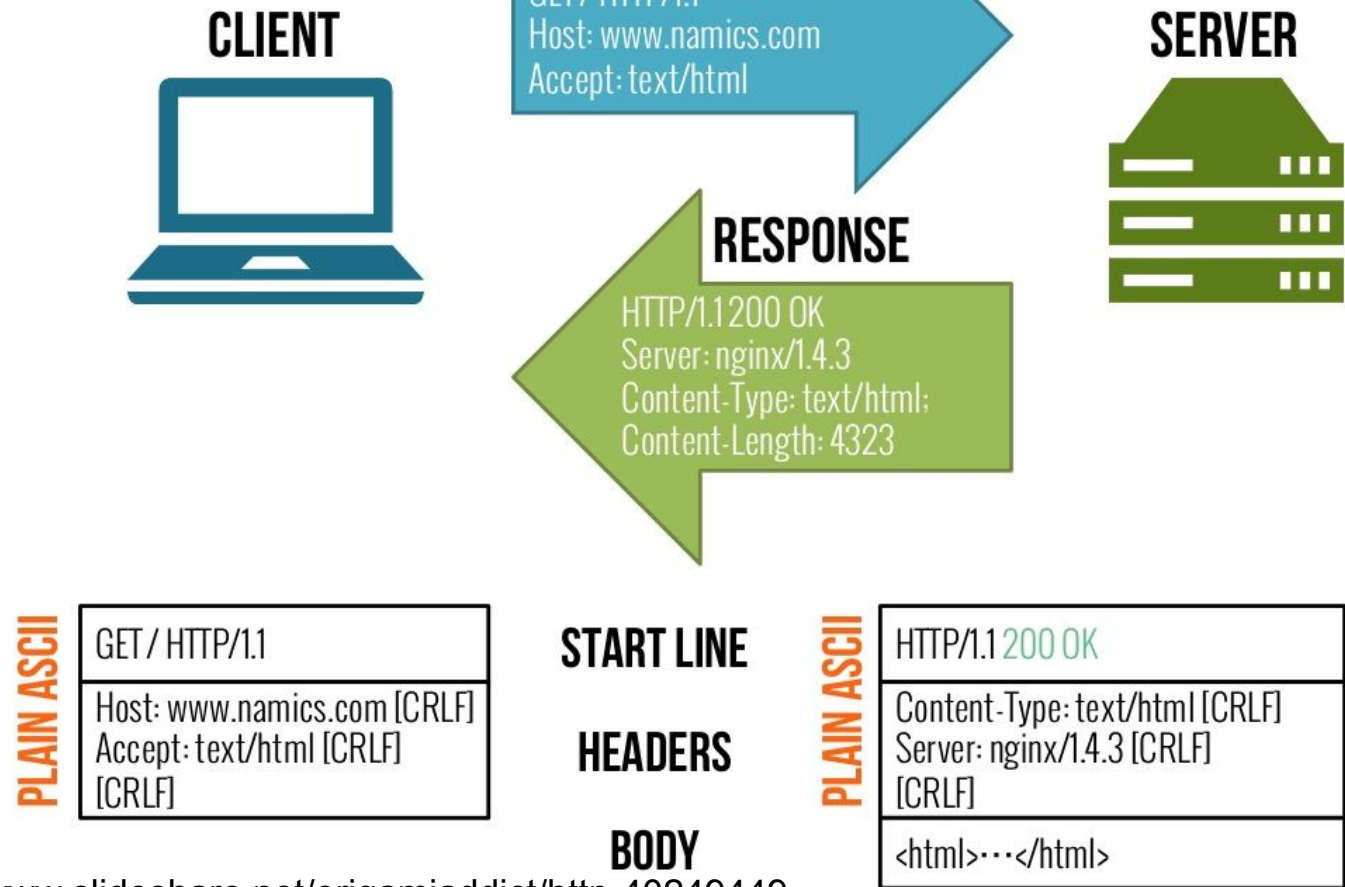
# Co to je 127.0.0.1 ?



# HTTP protokol



# HTTP protokol



# Cookies

Co to je?

Je to dobré?



# PHP - cookies

- jsou malé soubory uložené na klientovi
- obsahují většinou dočasné informace o stavu aplikace
- obsahují ID session - tzv. session ID



# Php - session (relace)

- session řeší problém bezstavosti protokolu HTTP
- drží informace o stavu aplikace a o uživateli
- session umožňuje předávání parametrů, které bychom jinak museli zobrazovat v URL
- data v session jsou fyzicky uložena na serveru a nepřenášejí se ke klientovi - na rozdíl od cookies

# Php - session - základní funkce

- `session_start()` - start session
- `session_destroy()` - ukončení session, které však nesmaže data!!!
- `$_SESSION["promenna"] = "ahoj";`
- `unset($_SESSION["promenna"])` - smaže proměnnou v SESSION,
  - pozor: nikdy nedělejte `unset($_SESSION)` - znemožní to další použití SESSION pro registraci proměnných
- SID - konstanta obsahující ID session, většinou nás nezajímá a vždy necháváme ukládat ID session do cookies

# Session - správné použití

*session1.php:*

```
session_start();           // start session
$_SESSION["color"] = "green"; // založení klíče color
$_SESSION["animal"] = "cat";  // založení klíče animal
```

*session2.php:*

```
session_start();
echo "Favorite color is " . $_SESSION["color"] . "<br>";
echo "Favorite animal is " . $_SESSION["animal"] . ".";
print_r($_SESSION);           // zobrazení dat v SESSION
```

Zdroj a další informace: [http://www.w3schools.com/php/php\\_sessions.asp](http://www.w3schools.com/php/php_sessions.asp)

# Session - správné zrušení

*session3.php*

```
session_start();           // použít session
$_SESSION["data"] = array(); // přepsat proměnnou
unset($_SESSION["data"])   // zabít proměnnou
session_destroy();          // ukončit práci se session, většinou není třeba
```

# Session - konfigurace v php.ini

- session.auto\_start - obvykle off, doporučuji nezapínat
- session.cache\_expire - doba platnosti session
- session.cookie\_lifetime - použijeme většinou 0 = do zavření prohlížeče
- session.save\_path - cesta, kam ukládat soubory se session, pozor, aby tam šlo zapisovat. Nejčastější místo, které způsobí nedostatek inodů na disku z důvodu, že se nemažou session.
- session.use\_cookies - používat cookies, obvykle 1
- session.use\_trans\_sid - zda vkládat identifikaci SESSION do URL, pokud budou vypnuté cookies. Doporučuji vypínat z důvodu bezpečnosti.

# Session v php.ini

Na tyto parametry většinou není vhodné sahat:

- session.cookie\_domain
- session.cookie\_path
- session.name - název SESSION, obvykle je "PHPSESSID"
- session.save\_handler - kam ukládat data, obvykle files

# Session - příklad

<https://github.com/madostal/kiv-web/tree/master/prednasky/session>

# Jak přihlásit uživatele?





# Php - přihlášení uživatele

- HTTP autentizace - vyskočí přihlašovací okno z prohlížeče
  - login a heslo dostanete v superglobální proměnné `$_SERVER`:
    - `$_SERVER['PHP_AUTH_USER']`
    - `$_SERVER['PHP_AUTH_PW']`
- klasický přihlašovací formulář

# Přihlášení uživatele - 1. formulář

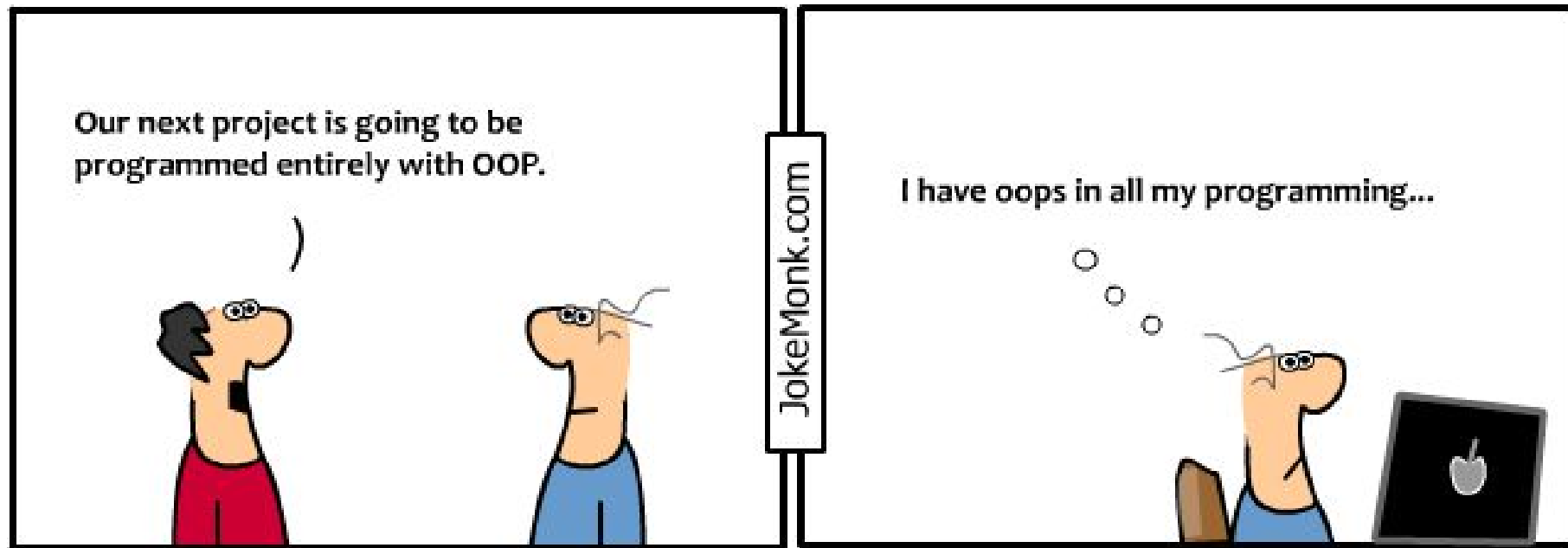
```
<form method="post">  
    <input type="text" name="user[login]" />  
    <input type="text" name="user[heslo]" />  
</form>
```

```
$user = $_POST["user"]; // dostanu login a heslo
```

# Přihlášení uživatele - viz. github



# OOP



# OOP - quick start

```
<?php
```

```
class kalkulacka
```

```
{
```

```
    public function secti($a, $b) {
```

```
        return $a + $b;
```

```
    }
```

```
}
```

```
$kalkulacka = new kalkulacka();
```

```
$c = $kalkulacka->secti(1, 2);
```

```
?>
```

# Class property (vlastnost třídy)

```
<?php
class MyClass
{
    // veřejná proměnná třídy
    public $prop1 = "I'm a class property!";
}
```

```
$obj = new MyClass;
var_dump($obj);
?>
```

# Konstruktor a destruktork

```
class MyClass
{
    public $prop1 = "I'm a class property!";

    public function __construct()    // alternativně MyClass
    {
        echo 'The class "', __CLASS__, '" was initiated!<br />';
    }

    public function __destruct()
    {
        echo 'The class "', __CLASS__, '" was destroyed.<br />';
    }
}
```

# Php - coding standard

- Správné značení souborů:
  - index.php - pouze 1 v rootu domény
  - auto.class.php - soubor obsahující definici třídy auto
  - seznam\_aut.inc.php - soubor, který se includeje do jiného souboru

**Vyberte si nějaký standard a ten dodržujte!**



# Php - připojení k databázi

Existují 3 základní možnosti:

- Nepoužívat: mysql rozšíření pro Php - deprecated od Php 5.5.0
  - `mysql_connect`
  - `mysql_query`
- Mysqli - nová alternativa, podpora pouze pro Mysql, navíc procedurální API
- Nejlepší: PDO - nejvhodnější alternativa i z důvodu snadné záměny databází - podpora 12 různých databázových systémů

# PDO - ukázka připojení z w3schools

```
$servername = "localhost"; $username = "username"; $password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
} catch(PDOException $e){
    echo "Connection failed: " . $e->getMessage();
}
```

Co je tady špatně a mohlo by to být lepší?

# PDO - prepared statements

- je to šablona obsahující SQL kód
- SŘBD ví, že se jedná o šablonu
- mohou obsahovat proměnné - poziční (?) nebo jmenné (:name)
  - Jmenné:

```
$dbh->prepare("INSERT INTO X (name, value)  
VALUES (:name, :value)");
```
  - Poziční:

```
$dbh->prepare("INSERT INTO X (name, value)  
VALUES (?, ?)");
```
- automatická ochrana před SQL injection - zvlášť jde SQL kód a zvlášť hodnota proměnné, ale stejně je dobré vždy kontrolovat všechny vstupy!!!

# PDO - prepared statements - ukázka

```
$conn = new PDO("mysql:host=$servername;dbname=$dbname", $user, $pass);
```

```
// prepare sql and bind parameters
```

```
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
```

```
VALUES (:firstname, :lastname, :email)");
```

```
$stmt->bindParam(':firstname', $firstname); //POZOR: navážu proměnnou!
```

```
$stmt->bindParam(':lastname', $lastname);
```

```
$stmt->bindParam(':email', $email);
```

```
// insert a row
```

```
$firstname = "John"; $lastname = "Doe"; $email = "john@example.com";
```

```
$stmt->execute();
```

# PDO - vložení hodnoty do prep. stmt

Navázání proměnné odkazem

- k vyhodnocení dojde při zavolání execute
- `$stmt->bindParam(':firstname', $firstname);`

Navázání proměnné hodnotou

- vloží se aktuální hodnota proměnné - **doporučeno**
- `$stmt->bindValue(':firstname', $firstname)`

# PDO - proměnná - příklad

```
$sex = 'male';
```

```
$s = $dbh->prepare('SELECT name  
FROM students WHERE sex = :sex');
```

```
$s->bindParam(':sex', $sex);  
// navázání proměnné
```

```
$sex = 'female'; // možnost chyb  
$s->execute();  
// executed with sex = 'female'
```

```
$sex = 'male';
```

```
$s = $dbh->prepare('SELECT name  
FROM students WHERE sex = :sex');
```

```
$s->bindValue(':sex', $sex);  
// navázání hodnoty
```

```
$sex = 'female';  
$s->execute();  
// executed with WHERE sex = 'male'
```

# PDO - metody

- `$conn->exec("sql kód");` // vrátí počet řádků
- `// vytvoří prepared statement`  
`$stmt = $conn->prepare("sql kód ? nebo :navez");`
- `// provede statement a vrátí result`  
`$res = $conn->query($statement)`
- `$conn->lastInsertID()` // poslední vložené ID

Zdroj: [http://www.w3schools.com/php/php\\_mysql\\_update.asp](http://www.w3schools.com/php/php_mysql_update.asp)

# **PDO - ukázka třídy skládající PDO dotazy**

Viz github: [github.com/madostal/prednasky/pdo](https://github.com/madostal/prednasky/pdo)



# Rekapitulace PDO - co bych měl umět?

- Rozdíl mezi PDO a MySQLi?
- Co to je prepared statement a jak to přibližně použít?
- Jaké jsou hlavní výhody?
- PDO - na co si dát pozor?