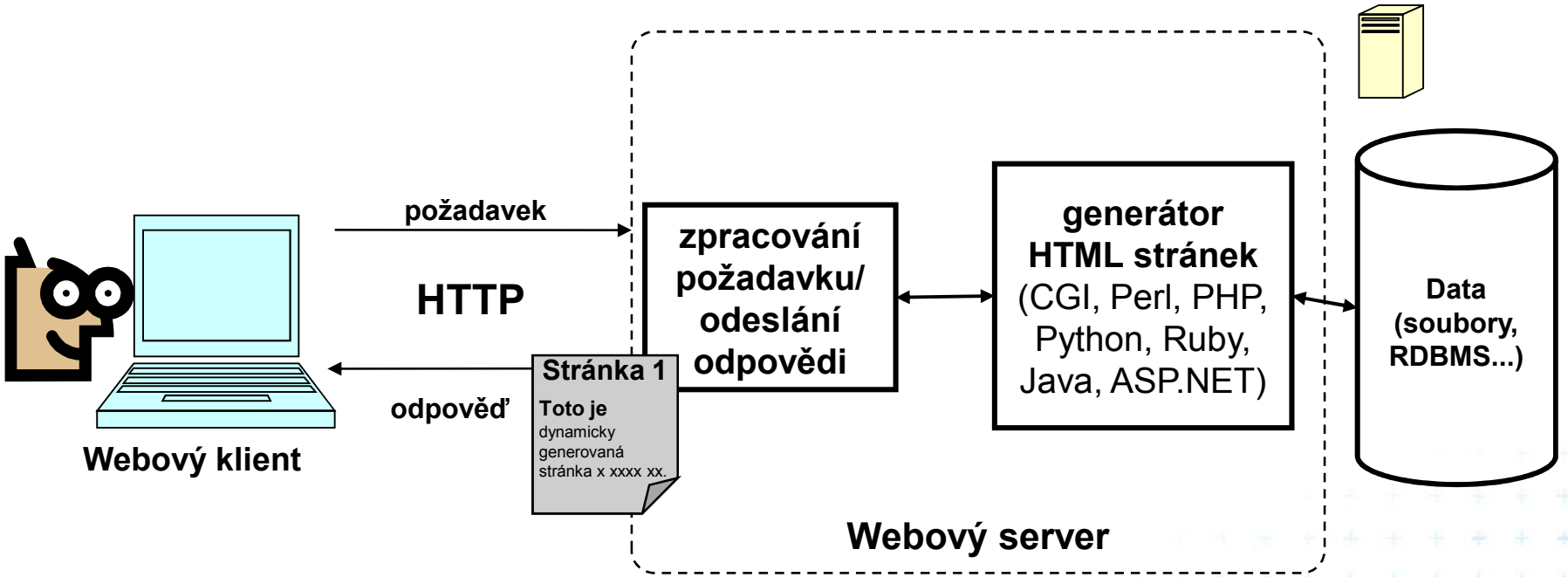
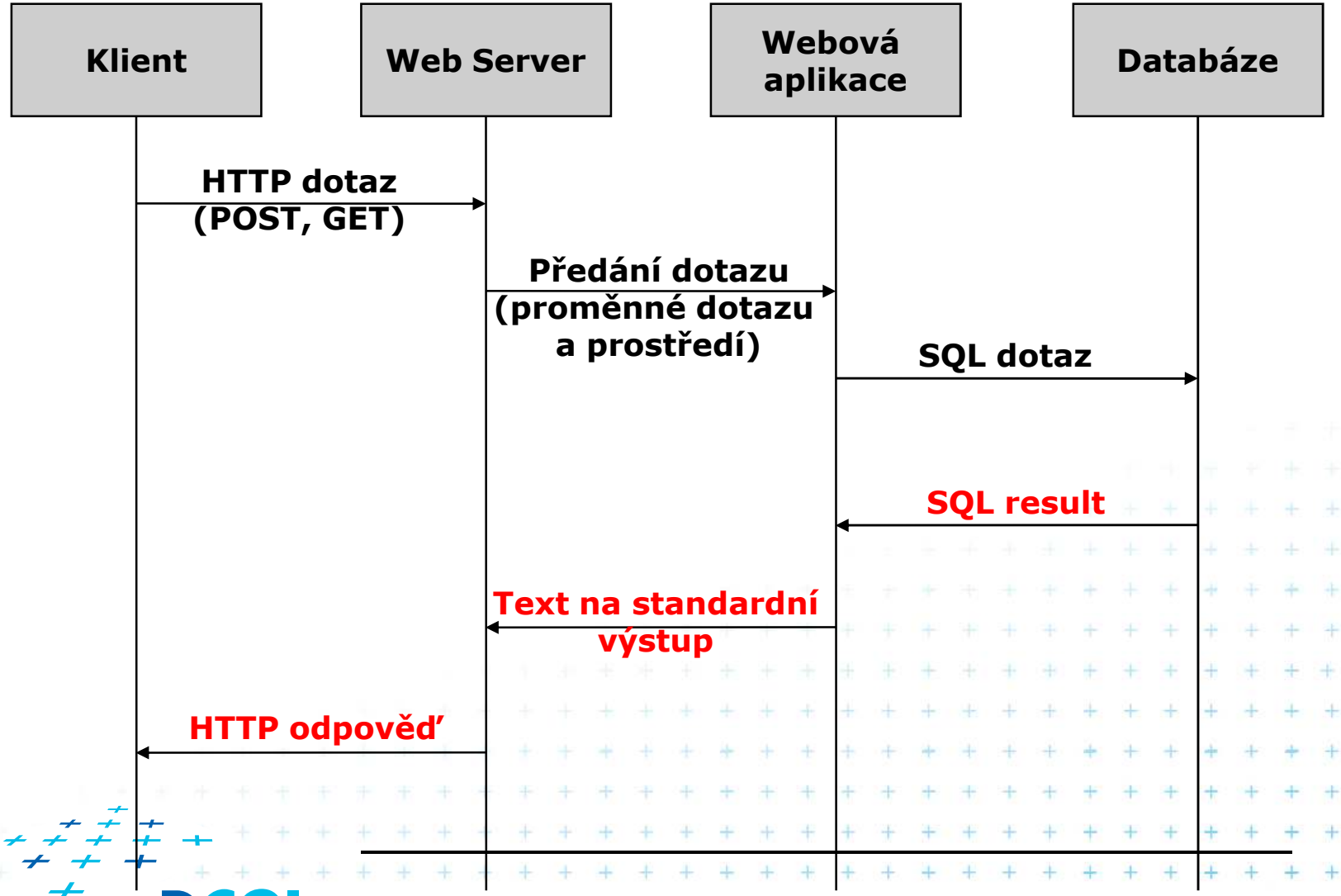
# Základy jazyka PHP

Martin Klíma

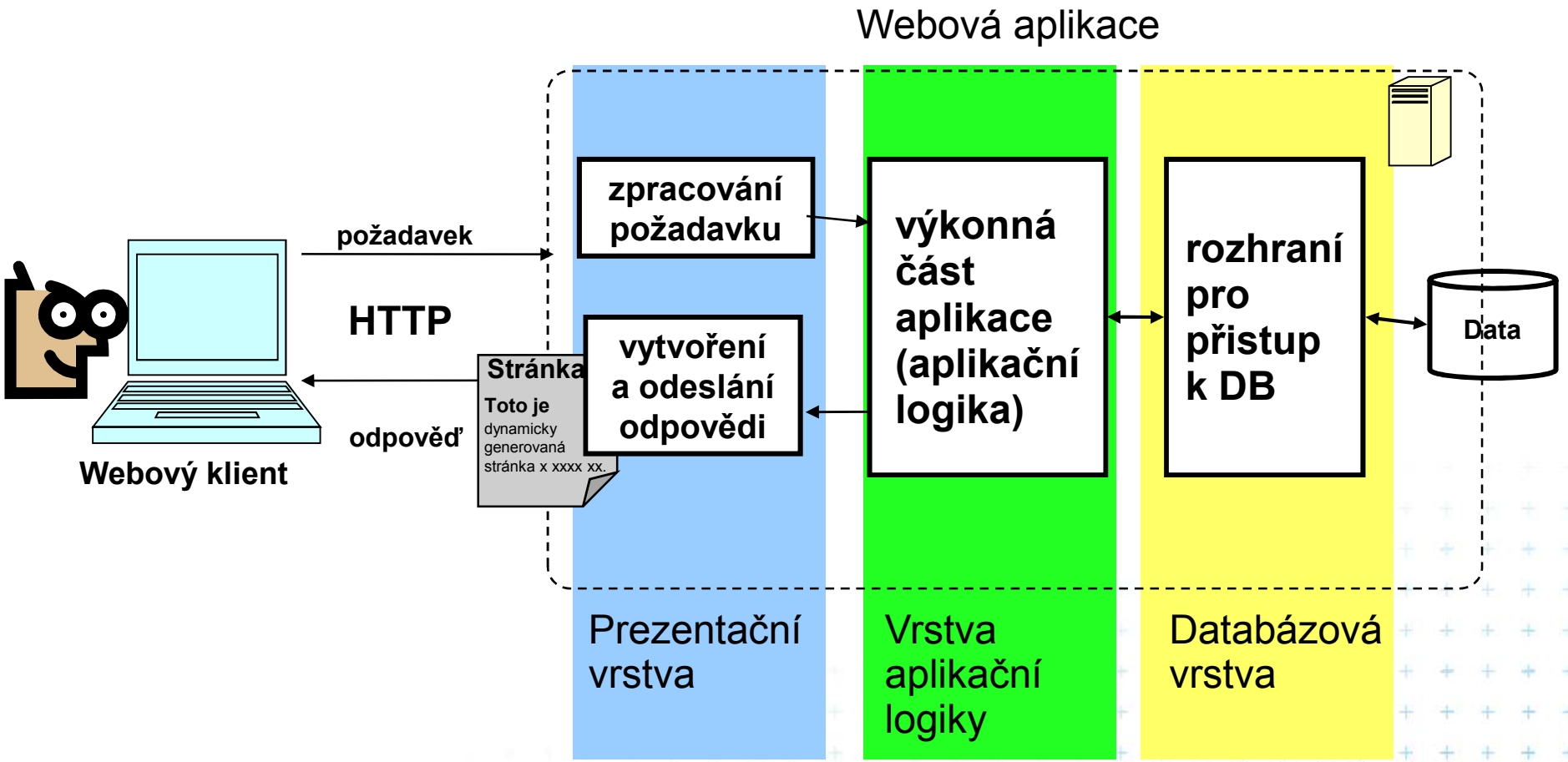

# Dynamický web



# Dynamický web – proces vzniku stránky



# Architektura webové aplikace – 3 vrstvy



# PHP (PHP Hypertext Preprocessor)

- Procedurální jazyk
- C-like syntax
- Množství funkcí v integrovaných knihovnách i modulech
- Jednoduchá integrace do webových serverů
- Od verze PHP5 solidní podpora OOP
- Není plně objektový


# Historie PHP

Počáteční autor: Rasmus Lerdorf

Další důležití autoři: Zeev Surashi a Andi Gutmans

- 1995: PHP 1.0 - Personal Home Page Tools (PHP Tools)
- 2001: PHP 4.1.0 – všechny základní rysy dneška
- 2004: PHP 5.0.0 – nový objektový model
- 2008: PHP 5.3 – bezpečnostní vylepšení, jmenné prostory,…



# PHP

## Výhody

- Jednoduchý na učení
- Podpora pro webové funkce přímo v jazyce
- Multiplatformní
- Open Source
- Web hosting - masová podpora
- Velká komunita vývojářů

## Nevýhody

- Netypovost, nekompiluje se, rychlost, neobjektovost
- Někdy těžkopádný
- Některé věci v jazyce jsou nedotažené
- Svádí k mizernému způsobu programování


# C-like syntax

- Příkazy oddělené středníkem ;
- Bloky sdružené pomocí {  … }
- Komentáře:
  - // řádkové
  - /* blokové  */
- Přiřazení pomocí ‘=’  $a=6
- Porovnání < , >,  ==, !=
- Proměnné jsou uvedeny pomocí znaku $ - $promenna
- Kontrolní struktury: if (cond) {..} else {..},  while (cond) {..} , switch/case, for (startcond; increment; endcond) {  }, foreach ($pole as $prvek),
- Prvky pole přístupné pomocí [ ] : $x[4]  je 5. element pole $x
- Asociativní pole:  $asocPole[“nazev_prvku”] = 'hodnota prvku';
- Funkce volané jménem s argumenty v pevném pořadí uzavřenými do ( ):  substr (“retezec”,0,2)
- Case sensitive  -  $promenna není to samé jako $Promenna



# Ukázka syntaxe

```
$pole = array("Martin","Martin","Tomas","Radek","Petr");
// vypis
for ($i = 0; $i <= 2; $i++) {
  if (($i % 2) == 0) {
      echo (substr($pole[$i], 0, 1) . "<br />");
  } else {
      echo ('tohle je else');
  }
}
```



# PHP proměnné, datové typy

**Přetypování:**

```
$x = (int) $y;
$x = (bool) $y;
$x = (float) $y;
$x = (double)$y;
$x = (string) $y;
$x = (array) $y;
$x = (object) $y;
```

**Dotazování:**

```
is_int($x);
is_bool($x);
is_float($x);
is_double($x);
is_string($x);
is_array($x);
is_object($x);
is_resource($x);
```



# PHP proměnné, datový typ

```php
<?php

   $x = "vyska";
   $$x = 10;

   echo $vyska; // 10
   echo "<br/>";
   echo $$x; //10

   $y = 1; //int
   $y = 1.0; // float
   $y = "abc"; // string


?>
```

# Proměnné - viditelnost

```
<?
 $a = 1; $b = 2;
function Sum() {
    global $a, $b;
    $b = $a + $b;
 }

Sum();
echo $b;
?>
```



# Proměnné viditelnost

- Superglobální proměnné
  - $_GET
  - $_POST
  - $_REQUEST
  - $_COOKIE
  - $_SESSION
  - $_ENV
  - $_FILES
  - $_SERVER
  - $GLOBALS

  - Tyto proměnné jsou viditelné vždy a všude. Není nutné na ně volat global


# Proměnné pokřačování

- Reference a hodnota

```php
<?php
        $jmeno = "František";

        echo "Jméno: ".$jmeno;

        //prirazeni hodnotou
        $jmeno2 = $jmeno;

        // reference na $jmeno
        $jmeno_ref = &$jmeno;

        //zmenime hodnotu promenne $jmeno
        $jmeno = "Hugo";

        echo "<br/>";

        //test
        echo "Jmeno: $jmeno, Jmeno2: $jmeno2, Jmeno_ref: $jmeno_ref";
?>
```



# Zvláštnosti

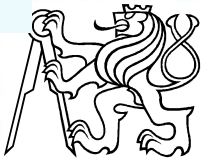
- Datový typ string

Při použití " se obsah řetězce vyhodnotí a proměnná se nahradí hodnotou

Při použití ' se obsah řetězce nevyhodnocuje

```php
<?php

   $jmeno = "František";
   $prijmeni = "Vomáčka";

   echo "$jmeno, $prijmeni";
   echo "<br/>";
   echo '$jmeno, $prijmeni';
?>
```





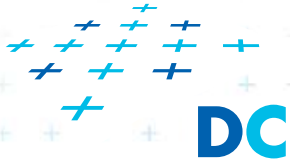
# Konstanty

```
define (“MOJE_KONSTANTA”,1);
define (“TVOJE_KONSTANTA”,”hodnota1”);

echo MOJE_KONSTANTA;
echo TVOJE_KONSTANTA;
```

pozor

```
echo moje_konstanta; // vypíše moje_konstanta a varování
```

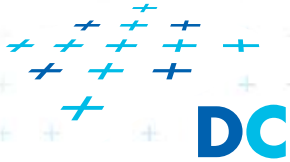
Pozor! Všechny konstanty jsou ve stejném jmenném prostoru, to může dělat problém při vkládání souborů, např. knihoven




# Předdefinované proměnné

$_GET
$_POST
$_REQUEST
$_COOKIE
$_SESSION
$_ENV
$_FILES
$_SERVER
$GLOBALS

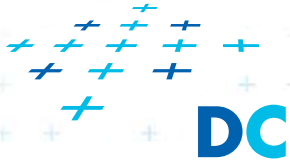
$php_errormsg
$HTTP_RAW_POST_DATA
$http_response_header
$argc
$argv




# Předdefinované konstanty

…těch je mnoho, viz

http://www.php.net/manual/en/reserved.constants.php




# Pole

- Pole je nejsilnější datový typ (hned po objektech)
- Všechna pole jsou asociativní

```
$pole = array();

$pole = array(4,5,6,7,8,9); // indexy od 0

$pole = array(5=>10, 20,30); // první index=5, další 6

$pole = array("pondělí"=>1, "úterý"=>2, "středa"=>3); // indexy jsou
    retezce

$pole[] = “xxx”; $pole[] = “yyy”;
```


# Pole

- Vícerozměrná pole

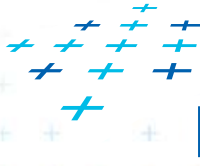
```
$pole[1][30] = 20;
$pole[3][10] = 22;
```

- Iterování polí

```
foreach ($pole as $klic=>$hodnota) {
   echo "$klic = $hodnota <br>";
 }
foreach ($pole as $hodnota) {
   echo "hodnota pole: $hodnota <br>";
}
```


# Dotazování, mazání

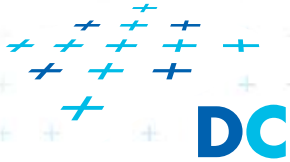
```
if (isset ($pole[1][5]))
    echo "pole[1][5] je nastaveno";
 else echo "pole[1][5] neni nastaveno";

unset ($pole[1][5]);
```

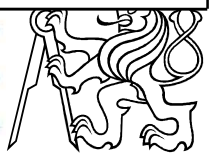



# Operátory

| |
|---|
| , |
| or |
| xor |
| and |
| print |
| = += -= *= /= .= %= &= \|= ^= <<= >>= |
| ? : |
| \|\| |
| && |
| \| |

| |
|---|
| ^ |
| & |
| == != === !== |
| < <= > >= |
| << >> |
| + - . |
| * / % |
| ! ~ ++ -- (int) (float) (string) (array) (object) @ |
| [ |
| new |



# Řídící struktury

- if
- else
- elseif
- while
- do..while
- for
- foreach
- break
- continue
- switch
- declare
- return


# If

```php
<?php

 $pole = array (1,2,5,6);

 if (sizeof($pole)>3)
  echo "velikost &gt; 3";
 elseif (sizeof($pole)>1)
  echo "velikost &gt; 1";
 else
  echo "pole je prázdne";

?>
```



# while, do - while

```php
$pole = array(1,2,4,5,6);
$i=0;
while($i < sizeof($pole)) {
        echo $pole[$i]."<br/>";
        $i++;
}

$i = 0;
do {
        echo $pole[$i]."<br/>";
        $i++;
} while ($i< sizeof($pole));
```



# For, foreach

```php
$pole = array("pondělí"=>1, "úterý"=>2, "středa"=>3);

// selhava, protoze pole nema ciselne indexy
for ($i=0; $i<sizeof($pole); $i++) {
        echo $pole[$i]."<br/>";
}

// univerzalni, v poradku
foreach ($pole as $klic=>$hodnota) {
        echo "index: $klic hodnota: $hodnota <br/>";
}
```



# switch

```
$jmeno = "Josef";
switch ($jmeno) {
        case "Karel": echo "ahoj Karle"; break;
        case "Zdenek": echo "ahoj Zdenku"; break;
        case "Jarmila": echo "ahoj Jarmilo";break;
        case "Josef":
        case "Pepa": echo "ahoj Pepo"; break;
        default: echo "ahoj neznámý";
}
```


# Funkce a procedury

…jedno jsou

Funkce může a nemusí vracet hodnotu.

Návratový typ není deklarován.

```
function sectiPrvkyPole ($pole) {
 if (!is_array($pole)) return false;
 $vysledek = 0;
 foreach ($pole as $hodnota) {
   $vysledek += $hodnota;
 }
 return $hodnota;
}

$pole = array(1,2,3);

echo sectiPrvkyPole($pole);
```

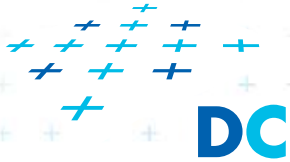


# Zpracování chyb v PHP4 a 5

error_reporting()

set_error_handler()

```
error_reporting(E_ALL);

function my_error_handler ($severity, $msg, $filename,
$line_num) {
        // dostanu info o chybe a muzu si s ni delat co
chci
        echo "Závažnost: $severity <br/>Hláška: $msg
<br/> Soubor: $filename <br/> Číslo řádku: $line_num
<br/>";
}
set_error_handler("my_error_handler");
echo $xxx;
```




# Vyjímky v PHP5

- Je zde zaveden lepší způsob ošetřování vyjímek.
- Podobnost s Javou.
- Jestliže je vygenerována vyjímka (chyba), je vyroben nový objekt.
- Každá vyjímka je rozšířením třídy Exception.
- Odvozením nové třídy lze vyrábět vlastní vyjímky.



# Vyjímky PHP 5

```
class DevZeroException extends Exception {}
class NegativValueException extends Exception {}

function deleni ($a, $b) {
        try {  if ($b == 0) throw new DevZeroException();
               if ($a<0 || $b<0) throw new
NegativValueException();
               return $a/$b;
        }
        catch (Exception $e) {
               echo "doslo k nejake vyjimce!!!!";
               return false;
        }
//      catch (DevZeroException $e) { echo "nulou nelze delit";
//             return false;}
//      catch (NegativValueException $e2) {echo "negative value
odchyceno v ramci funkce"; return false;}
}
deleni(1,2);
deleni(1,0);
deleni(-1,5);
```

# Vkládání souborů

- Použití knihoven = externí soubory
- Do místa direktivy vloží obsah referovaného souboru

- 2 základní způsoby
  - pouze jednou
  - iterativně

- Direktivy programu

```
include    "soubor"
include_once "soubor"

require "soubor"
require_once "soubor"
```

rozdíl mezi include a require:
funkční rozdíl není

include při nenalezení souboru
pokračuje v běhu programu,
require končí kritickou chybou


Dotazy?

# DĚKUJI ZA POZORNOST