Webové aplikace

Sylaby přednášek

Ing. Dalibor Fiala, Ph.D.

2012

Poděkování

V těchto sylabech přednášek předmětu KIV/WEB vyučovaném na Fakultě aplikovaných věd Západočeské univerzity v Plzni je použit materiál poskytnutý Ing. Josefem Steinbergerem, Ph.D., a Doc. Ing. Přemyslem Bradou, MSc., Ph.D., za což jim patří můj dík.

Ing. Dalibor Fiala, Ph.D. Plzeň, září 2012

Obsah

Historie a vývoj webu

1.1 Historie webu, hypertext

- Sir Timothy John Berners-Lee je všeobecně považován za vynálezce World Wide Webu – spojil technologie a principy, které už existovaly dříve
- Základem WWW je **hypertext** = mechanismus provázání textů skrze odkazy
    - 1945 - Vannevar Bush
    - 1965 – Ted Nelson – pojem hypertext, editor Xanadu

1.2 Velká počítačová síť

- 1962 – ARPANET - síť ministerstva obrany USA
- 1974 – Vint Cerf – TCP/IP – základ pro síť obřích rozměrů
- 1983 – ARPANET přechází na TCP/IP
- 1984 – ARPANET rozdělen – vojenská část + část pro vědecké účely (univerzity, později i velké firmy)
- Konec 80. let – konec ARPANET, NSFNET převzal jeho úlohu
- Připojení dalších sítí k NSFNET – vznik INTERNETu

1.3 Značkovací jazyk

- 1969 - **G**oldfarb, **M**osher a **L**orie – GML jazyk – podobnost s HTML

```
:book.
:body.
:h1.Toto je nadpis první úrovně
:p.Toto je odstavec
:ol.
:li.První položka číslovaného seznamu
:li.Druhá položka
:li.Třetí položka
:ul.
:li.Položka vnořeného seznamu
:li.Další položka
:eul.
:eol.
:p.Seznam skončil, začíná další odstavec...
```

1.4 SGML (1)

- Metajazyk GML prošel vývojem a v roce 1980 spatřila světlo světa jeho mutace označovaná **SGML** (*Standard Generalized Markup Language* – Standardní zobecněný značkovací jazyk).
- SGML není tedy nějakou konkrétní množinou značek pro popis dokumentu, ale můžeme si jej představit jako *metajazyk, který umožňuje definovat, jaké značky (elementy) se mohou v textu používat a jak spolu souvisejí (struktura dokumentu).*
- Definice přípustných elementů a vztahů mezi nimi se označuje jako DTD (*Document Type Definition*) a bývá uložena v samostatném textovém souboru.

1.5 SGML (2)

- Př. Chceme ukládat sbírky básní

```
<book>
        <poem>
                <title>Římské náměstí</title>
         <verse>
                <line>Na Římském náměstí</line>
                <line>už chřadne javor.</line>
                <line>Je cosi zlatistého v každém umírání.</line>
                <line>Sem tam slétne list</line>
                <line>a zkouší kolik sáhů</line>
                <line>má do hloubky hrob léta.</line>
         </verse>
        <verse>
                <line>Holubí stín</line>
                <line>jej přetne truchlou stuhou,</line>
                <line>ale nic nevěstí</line>
                <!- další řádky verše ->
         </verse>
        </poem>
        <poem>
                <!- následují další básně ->
        </poem>
</book>
```

1.6 SGML (3)

- Definice příslušného DTD by vypadala takto:

```
<!ELEMENT book     - -    (poem+)>
<!ELEMENT poem     - -    (title?,verse+)>
<!ELEMENT title    - 0    (#PCDATA)>
<!ELEMENT verse    - 0    (line+)>
<!ELEMENT line     0 0    (#PCDATA)>
```

1.7 SGML (4)

- Tim Berners-Lee – HTML (SGML ignorováno)

```
<NEXTID 2> <TITLE>Mamut s rýží a bramborem</TITLE>
<H1>Nadpis končí s koncem řádku
 Obyčejný text odstavce, <HP1> začalo tučné písmo.
<P> Tučný druhý odstavec, <HP2> začala kurzíva.
<P> Třetí tučný odstavec v kurzívě, </HP1> tučné písmo skončilo.
<P> Text čtvrtého odstavce  v kurzívě, </HP2> kurzíva skončila.
<P> <A NAME=1>Odstavec s kotvou</A>.
```

- HTML reformulován do SGML až ve verzi 2.0

1.8 World Wide Web

- Vynálezce webu v roce 1989 uchopil 45 let starý hypertext, velkou síť s 16 let starým protokolem, přidal vlastní napodobeninu SGML — vše smíchal dohromady a World Wide Web byl na světě.

1.9 Historie prohlížečů (1)

- Začátek 90. let – rychlý vývoj HTML, žádné standardy
- 1993 - Mosaic 1.0 (NCSA)
- 1994 – vznik Mosaic Communications, později Netscape Communications
  - Netscape Navigator 1.0 – 80% trhu během jednoho roku
- 1995 – HTML 2.0, potom HTTP 1.0
- Netscape - <frame> + JavaScript

1.10 Historie prohlížečů (2)

- Microsoft – vyměnil podíl ze zisku výsledného prohlížeče za zdrojové kódy Mosaicu (od NSCA – Spyglass, Inc.)
- Vznik Internet Exploreru – distribuce zdarma
- 1996 – IE jako součást WIN NT, potom WIN95
- 1997 – IE 4.0 ve WIN98

  - Kompletně přepsán
  - Zobrazování - jádro Trident
  - Lepší než Netscape 4.0
  - Kompatibilita se stránkami psanými pro Netscape
  - Lepší práce se styly
    - Optimalizace stránek pro IE
    - Netscape přestal být přívětivý

1.11 Historie prohlížečů (3)

- Dynamické stránky využívající možností Exploreru
- Implementace objektového modelu dokumentu nebyla v Netscapu dostatečně univerzální, proto si Microsoft vymyslel vlastní. Bohužel nekompatibilní. Kvůli tomu přestal být Netscape použitelný.
- W3.org – HTML 4 – zhroucení některých verzí Netscapu
- Netscape
  - Stále 35% trhu
  - 1998 – zveřejnění kódu prohlížeče – projekt Mozilla
  - Koupen firmou America Online – 9 mld. dolarů
- Mozilla
  - Nové zobrazovací jádro
  - Dohnal technologický náskok IE
  - Cesta standardů + přihlédnutí k realitě webu
  - Počet uživatelů se zvyšuje

1.12 Historie prohlížečů (4)

- Opera
  - 1996 – vznik v Norsku
  - 2003 – dotáhla náskok IE a Moz.
  - Kompromis mezi implementacemi IE a Moz.
  - Počet uživatelů narůstá
- Mozilla Firefox (2004)
- Google Chrome (2008)
- Explorer, Mozilla, Chrome, Opera = 99,5 % trhu

HTML

1.13 HTML – historie, verze

- Hypertext
- Jazyk popisující strukturu dokumentu
- SGML/XML aplikace
- Verze:
    - HTML 1 – 1990 + TBL, CERN
    - HTML 2.0 – 1995 jako RFC 1866
        - kodifikace (zachycení a standardizace) aktuálního stavu jazyka
        - všechny základní elementy (P, UL, PRE, FORM…)
    - HTML 3.0 – 1995, W3C standard
        - pokus o silný standard, nepoužívané
        - obsahovalo mj. matematické vyznačování

1.14 HTML – verze

- HTML 3.2 – 1997, W3C doporučení
    - kodifikace (zachycení a standardizace) aktuálního stavu jazyka
    - nové elementy: TABLE, DIV, FONT, MAP, APPLET atd.
- HTML 4.0 – 1998, W3C doporučení
    - formálně silný základ, praktické použití; důraz na přenositelnost, přístupnost
    - nové elementy a atributy: STYLE, FRAME, OBJECT, SCRIPT, lang, class, accesskey atd.
    - vylepšení: TABLE, FORM
- HTML 5.0 – 2008, W3C „*working draft*“

1.15 XHTML

- SGML → XML
    - zjednodušení DTD
    - snazší strojové zpracování, výměna dat
    - lepší modularita a rozšiřitelnost jazyka
- XHTML 1.0 – 2000, W3C doporučení
    - HTML 4.01 jako XML aplikace

  - nasměrování k čistému logickému vyznačování
- XHTML 1.1 – 2001, W3C doporučení
  - modularizace XHTML1
- XHTML 2 – dosud jako „working draft“
  - cíl: obecnější textové vyznačování, zcela bez prezentačních prvků

1.16 Text v HTML

```
<h1>HyperText Markup Language</h1>
<p>HTML is the <i>lingua franca</i> for publishing hypertext on the
<abbr title=“World Wide Web”>WWW</abbr>.
<a href="http://www.w3.org/TR/REChtml40/">
HTML 4.0</a> is W3C's recommendation for the latest version of HTML.</p>
```

1.17 Obecné prvky

- Značky: vyznačují elementy obsahu
  - <znacka> obsah </znacka>
  - atributy (u otevírací značky)
  - velikost písmen
  - prázdné elementy
  - img, br – nepárové značky
  - h1 není příkaz, href není parametr
- Komentáře
  - <!-- komentář -->
- Entity
  - < < > > & &
  - é é A A
- Bílé místo
  -

1.18 HTML x XHTML

HTML = aplikace SGML	XHTML = aplikace XML

• Značky
  ○ case insensitive
  ○ možno vynechat uzavírací
• Atributy
  ○ atribut=hodnota
  ○ atribut="hodnota s mezerou"
  ○ atribut
• Ne-SGML data
  ○ <![CDATA[…]]>
  ○ obvykle stačí komentáře
• Renderování
  ○ volná interpretace, tolerance
	• Značky
	  ○ case sensitive: malými
	  ○ uzavírací povinně (...)
	• Atributy
	  ○ povinné uvozovky
	  ○ žádná minimalizace
	• Ne-XML data
	  ○ povinně CDATA sekce
	  ○ styly, JavaScript atd lépe do externích souborů
	• Renderování
	  ○ striktní chování

1.19 Varianty: Strict x Transitional

- Strict
  - pouze logické vyznačování
- Transitional
  - pojem „*deprecated element*“ (celkem 10 v HTML4)
  - Nestandardní rozšíření
    - staré verze HTML (Netscape, Microsoft…)
  - Důsledky
    - sada elementů, struktura těla dokumentu
    - chování prohlížečů (CSS)
- Strict je důležitější než XHTML

1.20 HTML dokument

- Preambule, deklarace
  - XML deklarace
  - DOCTYPE
- Záhlaví - <head>

- title, meta, style...

- Tělo - <body>

1.21 Příklad HTML dokumentu

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN“
   "http://w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns=”http://w3.org/1999/xhtml” lang=”en”>
<head>
      <title>HTML Home Page</title>
      <meta name=”keywords” content=”HTML, XHTML 1.0” />
      <style type="text/css"><!-- body { margin-left: 10%; } // --> </style>
</head>
<body>
      <h1 class="c1">HyperText Markup Language</h1>
      <p>HTML is the <i>lingua franca</i> for publishing hypertext on the <abbr
      title=“World Wide  Web”>WWW</abbr>.
      <a href="http://www.w3.org/TR/REC-html40/">HTML 4.0</a> is W3C's
      recommendation for the latest version of HTML.</p>
</body>
</html>
```

1.22 Preambule, deklarace

- Preambule
  - SGML: implicitní
  - XML: povinná

    ```
    <?xml version="1.0" encoding="UTF-8"?>
    ```

- Deklarace typu dokumentu
  - <!DOCTYPE kořenový_element SYSTEM/PUBLIC "URL">
  - označuje gramatiku
  - !!! povinná (interpretace dokumentu)
  - HTML4 strict

    ```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
    ```

  - HTML4 Transitional

    ```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd“
    ```

  - XHTML Strict

```
<!DOCTYPE HTML PUBLIC "DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
+ XML deklarace
```

1.23 Záhlaví <head>

- Meta-informace o dokumentu
  - není obsah dokumentu, nezobrazuje se
- Dokument element
  - HTML: atributy xmlns, lang…
- Značky záhlaví
  - TITLE
  - META: metainfo (author, keywords) + HTTP
    - vyhledávače, filtrování obsahu, …

      ```
      <meta http-equiv="Content-Type" content="text/html;
      charset=windows-1250"/>
      ```
  - LINK: odkazy na úrovni dokumentů
    - typy: next, prev, contents, index, copyright, … , stylesheet + alternate
  - STYLE
    - type, media
  - SCRIPT, NOSCRIPT
    - type, src

1.24 Tělo <body>

- Obsah dokumentu
  - prohlížeč: *canvas* pro zobrazení
  - BODY
  - FRAMESET
- Atributy
  - style, class, id, title, lang
- Vnořené elementy
  - blokové, v nich textové (*inline*)
  - struktura rámců

1.25 Obsahové elementy

- Blokové

    - zalamují odstavec
    - bloky, tabulky, formuláře
- Textové
    - uvnitř blokových
    - frázové × prezentační
- Generické
    - kontejnery, vazba na CSS
- Obecné atributy
    - všechny elementy
    - id, class, style, title; lang, dir; onSomeEvent

1.26 Základní blokové elementy

- Odstavec – p
- Nadpisy – h1-h6
- Odrážky, číslování – ul/ol li, dl dt dd
- Prezentační atributy – align=center
- Uvozovkování – blockquote – atribut cite="URI"
- Předformátovaný text – pre
- Odřádkování – br

1.27 Základní textové elementy

- Důraz – em
- Zesílení – strong
- Podtržení, škrtnutí – ins, del
- Tučné, kurzíva, podtržení – b,i,u
- Indexy – sub,sup
- Monospaced text – tt
- Další – cite, abbr, q, code
- Nedoporučované:
    - strike, big/small,font
    - <xyz align="left" background="..." > a podobné

1.28 Hypermediální elementy

- Odkaz – a (name, href, title)

- Obrázek – img
- Klikací mapy
  - map, area, src, alt, align, usemap
  - name, shape, coords, href
- Další – object, applet, param

1.29 Generické kontejnery

- Bez formátování
- div
  - blokový
- span
  - řádkový (*inline*)
- Atributy pro stylování
  - id, class, style

1.30 Formulářové elementy

- Interakce klient (uživatel) – server
  - zasílání dat na server, zpracování URI objektem
  - metody GET × POST
- form – method, action, enctype
- input – name, value; type; size, maxlength, checked
  - type: text|password|checkbox|submit|file|hidden
- select – multiple, option selected, optgroup
- textarea - rows, cols
- label for ; fieldset, legend
- Obecné atributy
  - tabindex, accesskey, disabled, readonly

1.31 Tabulky

- Popis dat s tabelární strukturou
  - zneužití: formátování pro HTML 3
- table; caption
  - prezentační atributy: width, border
- tr; th, td
  - colspan, rowspan

    - prezentační atributy: align, valign
- thead, tfoot, tbody
    - kontejnery řádků
- col, colgroup
    - stylování sloupců

1.32 Rámce

- frameset: definice mřížky
    - title; rows, cols – velikosti rámců
    - absolutní: pixely (“30”)
    - relativní: procenta (“25%”),
    - poměry (“4*”)
- noframes
    - alternativní obsah pro non-frame prohlížeče
- frame: úvodní obsah rámce
    - name, src
    - noresize, scrolling = “auto|yes|no”, frameborder
- <a href=”…”> s rámci
    - target=”#name”; _blank, _self, _parent, _top

1.33 Tvorba správného HTML

- Editory
    - textové
    - „značkovací“
    - WYSIWYG
- Generování
    - z dokumentů
    - aplikacemi
- Další nástroje
    - validátor
- Zobrazení
    - zdrojový kód
    - prohlížeče – textové, grafické, čtečky

1.34 Strukturování obsahu

- Varianta „prezentační“
  - HTML 3 bez CSS
  - vyznačit tak, aby se co nejlépe zobrazilo
  - tabulkový layout
- Varianta „**informační**“
  - HTML4/XHTML s CSS
  - vyznačit tak, aby se co nejlépe vyhledávalo
  - důležitý obsah napřed
  - dobrý title a h1/h2

1.35 Přístupnost

- Přístupnost = bezbariérovost
  - prohlížeč, OS, rozlišení, skriptování
  - kultura, motorické schopnosti, vidění
  - extrémně handicapovanými návštěvníky jsou vyhledávací roboti
- Zásady
  - validovat
  - používat informační strukturování
  - titulek, hierarchie nadpisů, oddělená navigace, linearizace tabulek
  - psát stručně a srozumitelně
  - členění textu
  - zpřístupnit formuláře
  - **vyhnout se rámcům**

1.36 Možnosti HTML pro přístupnost

- Elementy
  - h*, p; div
  - em, strong, q, cite…
  - fieldset, legend; optgroup; label for
  - th, thead/tfoot; caption
- Atributy
  - title, lang, dir, accesskey
  - specifické

- table - longdesc
- input - title, tabindex
- img - alt, title, longdesc

Kaskádové styly CSS

1.37 Cascading Style Sheets (CSS)

- Nadstavba HTML, XHTML, XML
- Popis prezentace dokumentů
- Oddělení obsahu a prezentace
- Standardizace W3C
- Verze
    - 1996 – CSS1 – W3C doporučení
        - Hlavní úlohou byla podpora HTML (3.2)
    - 1998 – CSS2 – W3C doporučení
        - Typy médií, tabulky, generování obsahu, XML
    - 2007 – CSS2.1 – W3C doporučení
        - Korekce
    - ? – CSS 3 – „*working draft*“
        - Velké změny, viz stránky W3C

1.38 Základní koncept

- h1 { color: green; }
- Selektor – co bude formátováno
    - Typy prvků
    - Třídy
    - Jednotlivé prvky
- Deklarace – vlastní formát
    - vlastnost: hodnota
    - h1 { color: green; font-weight: bold; }

1.39 První styl

```
/* prvni style – komentare jako v C */
body
{
        font-family: "Times New Roman", serif;
        font-size: 12pt;
}
```

```
h1, h2, h3
{
        font-family: Helvetica, sans-serif;
        color: navy;
}
```

1.40 Propojení stylu a stránky

- Externí styl
  - Styl uložený v .css souboru
  - Prvek link (v head)

    <link rel=stylesheet type="text/css" href="http://style.com/cool.css" title="Cool"></link>
  - Do stylu lze importovat další styl

    @import url(http://style.com/basic);
- Interní deklarace stylu
  - Prvek style (v head)

    <style type="text/css">
    <!--
    h1 { color: blue }
    -->
    </style>

    (Obsah v komentáři kvůli starým prohlížečům, které ještě styly neznaly.)
  - Atribut prvku

    The paragraph is green.

1.41 Základní selektory

- Definují to, co má být formátováno
- Jeden typ prvku: p {...}
- Seskupení prvků: h1, h2 {...}
- Kontextový selektor: ul li {...}
- Selektor třídy: .dulezite {color: red}
  - u prvku
  - prvek ve více třídách:
- Třída pro jeden typ prvků: p.dulezite {...}
- Selektor id: #html {...}, u značky
  - Id musí být v dokumentu jedinečné

1.42 Pseudotřídy

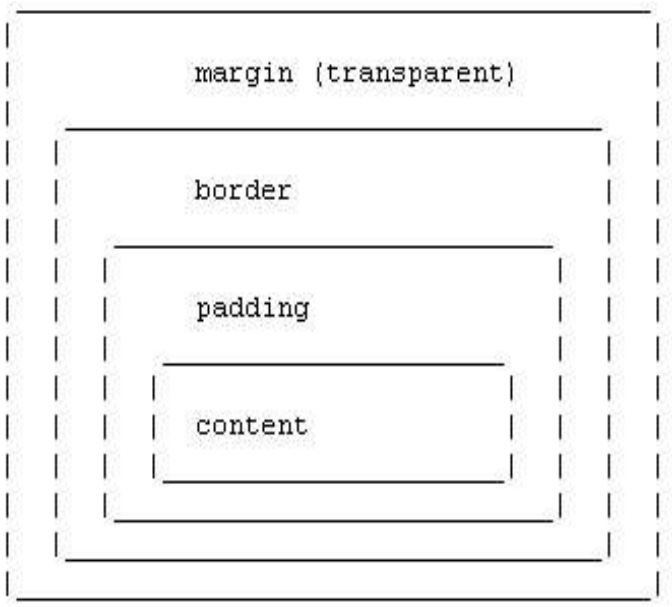
- Některé efekty nejde zachytit HTML strukturou (události – např. hover)
- Odkazy
  - A:link
  - A:visited
  - A:active
  - A:hover
- Formuláře
  - input:focus { color: red; }
- Text
  - P:first-line
  - P:first-letter

1.43 Kombinace selektorů

- P.green em { color: red; }
- #em1, cite, div.navig A { … }
- A.navig:visited { … }

1.44 Formátovací model

- Dva druhy prvků
  - Blokové (*block*): definují obdélníkové bloky s okraji (např. p, div)
  - Řádkové (*inline*): obsah v řádcích uvnitř bloku (např. em, strong, span)
- Margin, border, padding
  - *top/right/bottom/left*



1.45 Vlastnosti a hodnoty

- Zkrácený zápis
  - font-weight: x; font-size: y -> font: x y
- Jednotky: pt, pc, in, cm, mm; em, ex, px, % (absolutní / relativní)
- Barvy: red, #ff33aa, rgb(8,51,0), rgb(5%,0%,1%)
- URL: url("http://www.kiv.zcu.cz/a.gif")
- Speciální hodnoty: thin; Times; NW, S; dotted, solid; …

1.46 Display, seznamy, white-space

- display: block | inline | list-item | none
- list-style-type: disc | lower-roman | none | …
- list-style-position: inside | outside
- list-style: circle outside
- white-space: normal | pre | nowrap

1.47 Fonty

- font-family: "Arial CE", Arial, helvetica, sans-serif
  - (serif, sans-serif, monospace, cursive, fantasy)
- font-weight: normal | bold | lighter | 100-900
- font-style: normal | italic | oblique
- Sdruženě: font: italic 12pt/1.2 "Helvetica CE", Arial, sans-serif

1.48 Text

- text-align: left | center | right | justify
- text-decoration: none | underline | blink | …
- vertical-align: baseline | sub | text-bottom | …
- word-spacing: 1em (letter-spacing)
- text-decoration: none | underline | overline | line-through | blink
- Další: text-indent, text-transform

1.49 Barvy a pozadí prvků

- Barva popředí: color
- Barva pozadí: background-color
- Obrázek na pozadí: background-image: <url>
- Opakování obrázku: background-repeat: repeat | repeat-x | repeat-y | no-repeat

1.50 Boxy

- margin: 1em 2em (top+bottom, right+left)
- border-style: none | dotted | dashed | solid | double | groove | ridge | inset | outset
- border-width: thin thick medium thin
- border: solid 1px blue
- width, min-width, max-width; height

1.51 Pozicování

- Obtékání
  - float: none | left | right
  - clear: none | left | right | both
- Absolutní x relativní pozicování
  - position: absolute | relative | static
  - left, top, bottom, right
- Viz kapitolu 9 ve specifikaci CSS 2.1

1.52 Kaskáda a dědičnost (1)

- Obsah daného prvku dědí vlastnosti rodičovského prvku
- % (např. font-size) relativně k nadřazenému prvku
- Některé vlastnosti se nedědí: background, background image, margin, border nebo padding
- Dědičnost:
  1. Je hodnota někde určena? -> kaskáda
  2. Není-li určena, může se zdědit?
  3. Použij výchozí hodnotu.
- Kaskáda: prvek může mít určitou vlastnost definovánu vícekrát. Která definice opravdu zabere?

1.53 Kaskáda a dědičnost (2)

- Ovlivnění kaskády: !important
- Kaskáda
  1. Najdi všechny definice pro daný prvek
  2. Seřaď je podle váhy (klient < uživatel < autor < autor s !important < uživatel s !important)
  3. Pokud stejná váha – seřaď podle specifičnosti (specifičtější selektory mají přednost před obecnějšími)

4. Seřazení podle pořadí definování

- Příklady specifičnosti:
  - LI specifičnost = 0001
  - UL OL LI specifičnost = 0003
  - LI.red specifičnost = 0011
  - OL LI.red specifičnost = 0012
  - #x34y specifičnost = 0100
  - style="..." specifičnost = 1000

1.54 Další selektory v CSS2

- **Většina nefunguje v IE**
- **Univerzální selektor (*)**
  - * – jakýkoliv prvek
  - DIV * P – P je vnuk DIV
- **Děti (>) a sourozenci (+)**
  - P > EM – <p><i> nevyhovuje
  - H1 + H2 – <h2> po <h1>
- **Atributy ([…])**
  - p[attr] – vyhovuje p s nastaveným attr
  - p[attr=val] – totéž, pokud attr="val"
  - p[attr~=val] – totéž, pokud <p attr="x y val z">
- **Přidané pseudotřídy**
  - p:first-child – první dítě
  - p:lang(en) – když hodnota lang u P je „en“

1.55 Generování obsahu

- p:before – značí začátek prvku
- p:after – značí konec prvku
- content – vložený obsah

```
.poznamka:before
{
        content: "Poznámka: ";
}
```

1.56 Zásady stylování

- Základ
  - pište v XHTML strict
  - nepoužívejte table-layout, rámce, „align“ atd.
- Optimalizujte HTML pro vyhledávače
- Části layoutu do div
- Používejte pro kontrolu více prohlížečů
- Nepoužívejte absolutní velikosti
  - font-size: 7px vs font-size: 7pt
- Neopakujte to, co se dědí
  - body {font-size: 5pt;} p {font-size: 5pt;}
- Třídy nelze zanořovat (P.navig.green)
  - ale lze toto: <p class=”green bold”>
- Používejte alternativní styly: <link rel=”alternate stylesheet” href=”...” title=”...” />

PHP - syntax

1.57 PHP – obecně (1)

- Skriptovací programovací jazyk
- Určený především pro programování dynamických internetových stránek
- Kód lze začlenit do HTML
- PHP skripty jsou prováděny na straně serveru
- K uživateli je přenášen až výsledek jejich činnosti
- Syntaxe jazyka kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java)

1.58 PHP – obecně (2)

- Nezávislý na platformě, skripty fungují bez úprav na různých operačních systémech
- Rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory
- Přístup k většině databázových serverů (mj. MySQL, ODBC, Oracle, PostgreSQL, MSSQL)
- Podpora celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP)
- Kombinace – Apache+MySQL+PHP – často používána k tvorbě webových aplikací
- Od verze 5 OOP podobné Javě

1.59 PHP – historie

- 1994 – Rasmus Lerdorf – pro osobní potřebu, vydána sada skriptů - *Personal Home Page Tools*
- Stále ještě v r. 1994 – PHP spojeno s programem Form Interpreter stejného autora - PHP/FI 2.0
- 1997 - Zeev Suraski a Andi Gutmans – přepsali parser – základ PHP3
- Současně byl název změněn na *PHP Hypertext Preprocessor*
- 1998 – PHP3 – rychlejší, více funkcí, běželo i pod WIN
- 2000 – PHP4
- 2004 – PHP5 – vylepšené OOP podobné Javě

1.60 Ukázka kódu

```
<table>
        <tr>
                <th>Číslo</td>
                <th>Druhá mocnina</td>
        </tr>
<?php
        $i = 1;
        while ($i <= 10) {
                echo "<tr><td>".$i."</td><td>".$i*$i."</td></tr>\n";
                $i++;
        }
?>
</table>
```

1.61 Vlastnosti jazyka PHP

- Jazyk PHP je **dynamicky typový**, tzn. že datový typ proměnné se určí v okamžiku přiřazení hodnoty
- Pole jsou heterogenní, mohou tedy obsahovat jakékoli údaje, stejně tak jako jejich indexy
- Koncovka souborů: .php
- PHP kód uzavřený do <?php ... ?>, mimo HTML kód
- Komentáře jako v Javě: /* … */ a //… (#)
- ; jako oddělovač příkazů
- Volání funkcí: date('H:i, jS F');
- Proměnné s dolarem $i = 5;

1.62 Styly PHP značek

- <?php echo '<p>Text</p>'; ?>
    - Preferovaný styl
    - Nelze zakázat na straně serveru
    - Může být použit v XML dokumentech
- <? echo '<p>Text</p>'; ?>
    - Podle SGML instrukcí
    - Lze zakázat na straně serveru (*short_tag)*
- <script language="php"> echo ' Text '; </script>

  - Jako JavaScript
  - Pokud má HTML editor problémy s jinými styly PHP značek
- <% echo '<p>Text</p>'; %>
  - Lze zakázat na straně serveru (*asp_tags)*

1.63 Přístup k proměnným formuláře

- Odešleme formulář s polem jmeno a jeho hodnotou 'Karel'
- Přístup k tomuto parametru:
  - Zkrácený: $jmeno
    - Musí být povoleno na straně serveru
    - Problémy s laděním
  - Střední: $_POST['jmeno']
    - Doporučovaný
    - Od PHP v. 4.1.0
  - Dlouhý: $HTTP_POST_VARS['jmeno']
    - Zastaralý

1.64 Proměnné

- Identifikátory:
  - Písmena, čísla, podtržítka, (dolary)
  - Nesmí začínat číslicí
  - *Case sensitive* (výjimkou jsou jména funkcí)
  - Proměnná může mít stejné jméno jako funkce – nepoužívat
- Deklarace
  - Proměnná je vytvořena, pokud jí prvně přiřadíme hodnotu

1.65 Datové typy (1)

- PHP podporuje následující datové typy
  - Integer
  - Double
  - String
  - Boolean
  - Array
  - Object
- Nedefinované proměnné mají hodnotu NULL (dat. typ NULL)

- Dat. typ *resource* – manipulace s databází

1.66 Datové typy (2)

- **Slabě typovaný** jazyk

  (a = 0; b = 0.0;)
- Typ proměnné lze změnit

  (b = 'ahoj';)
- Přetypování

  (b = (double) a;)

1.67 Konstanty

- Velkými písmeny
- define('KONSTANTA', 100);
- phpinfo(); - php konstanty

1.68 Rozsah platnosti proměnných

- Místa ve skriptu, kde je proměnná viditelná
- Vestavěné superglobální proměnné jsou viditelné v celém skriptu
- Globální proměnné deklarované ve skriptu jsou viditelné v celém skriptu kromě vnitřku funkcí
- Proměnná použitá uvnitř funkce je lokální
- Proměnná použitá uvnitř funkce, která je deklarovaná jako globální, odkazuje na globální proměnnou
- Superglobální proměnné - $GLOBALS, $_SERVER, $_GET, $_POST, $_COOKIE, $_FILES, $_ENV, $_REQUEST, $_SESSION

1.69 Operátory (1)

- Aritmetické - +,-,*,/,%
- Zřetězení – $a.$b
- Přiřazení - =
  - $b = 6 + ($a = 5); // 11
  - +=, -=, *=, /=, %=, .=
- Pre/Post-Increment/Decrement
  - a=5; echo ++$a; echo $a++;
- Reference
  - $a=5; $b=$a; $a=7; echo $a.” “.$b;

- $a=5; $b=&$a; $a=7; echo $a.” “.$b;

1.70 Operátory (2)

- Porovnávání
  - == (shodné hodnoty) x === (shodné typy)
  - !=, <>, <, >, <=, >=
  - !, &&, ||, and, or (and a or má nižší prioritu)
- Bitové operátory
  - &, |, ~, ^, >>, <<
- Ternární operátor
  - Podmínka ? True hodnota : false hodnota;
- Potlačení výpisu chyby
  - $a = @(57/0);
- Operátor spuštění procesu
  - $out = ‘dir c:’;

1.71 Funkce pro práci s proměnnými

- string gettype(mixed var);
- bool settype(mixed var, string type);
- is_array(), is_double(), is_float(), is_real, is_long(), is_int(), is_integer(), is_string(), is_object()
- Stavu proměnné
  - bool isset(mixed var);
  - void unset(mixed var);
  - bool empty(mixed var);
- Ekvivalent přetypování
  - int intval(mixed var);
  - float doubleval(mixed var);
  - string strval(mixed var);
- mixed není datový typ php

1.72 Řídicí struktury – if

```
if ($cena < 1000) {
        echo "laciné";
}
elseif ($cena < 10000) {
        echo "dražší";
}
else {
```

```
        echo "nejdražší";
}
```

1.73 Řídicí struktury – switch

```
switch ($prom)
{
        case 'a':
                příkaz;
                break;
        case 'b':
                příkaz;
                break;
        default:
                příkaz;
                break;
}
```

1.74 Řídicí struktury – while, for, do

```
while (podmínka)
{
        příkazy;
}
for ($d = 50; $d <= 250; $d += 50)
{
        příkazy;
}
do
{
        příkazy;
} while (podmínka);
```

1.75 Řídicí struktury

- Ukončení smyčky: break
- Skok na další iteraci: continue
- Ukončení php skriptu: exit

PHP - soubory, pole, funkce

1.76 Otevření souboru (1)

- Otevření souboru:
  - $fw = fopen("cesta/soubor", mód);
- *Root* (kořenový adresář) webserveru:
  - $_SERVER['DOCUMENT_ROOT'];
- Módy otevření souboru:
  - r – read
  - r+ – read + write
  - w – write
  - w+ - write+read
  - a – append
  - a+ - append and read
  - b – binární mód
- Pozor na *safe mode* a přístupová práva
- Při neúspěchu vráceno false

1.77 Otevření souboru (2)

```
@ $fw = fopen("cesta/soubor", “r”);
if (!$fw)
{
        echo "Nelze otevřít soubor...";
        exit;
}
```

1.78 Zápis do souboru, zavření souboru

- Otevření souboru pro zápis

  @ $fw = fopen("cesta/soubor", “w”);
- Zápis do souboru
  - fwrite($fw, $retezec);
  - fputs($fw, $retezec);
- Zavření souboru
  - fclose($fw);

1.79 Čtení ze souboru

- Otevření souboru pro čtení

```
@ $fw = fopen("cesta/soubor", “r”);
if (!$fw) { … }
```

- Test konce souboru
  - feof($fr);
- Čtení řádky:
  - fgets($fw, 999) – max 998 bytů
  - fgetss – podobné, vyhazuje HTML/PHP tagy
  - fgetcsv – třetí parametr jako oddělovač
- Čtení celého souboru
  - readfile(soubor), $filearray = file($fr)
- Čtení znaků
  - fgetc($fr);
- Čtení bytů
  - fread($fr, délka);

1.80 Další užitečné funkce

- Test existence souboru
  - file_exists(soubor)
- Zjištění velikosti souboru
  - filesize(soubor)
- Vymazání souboru
  - unlink(soubor)
- Navigace v souboru
  - rewind($fw) – na začátek souboru
  - ftell($fw) – zjištění pozice
  - fseek($fw, offset, start) – nastavení pozice v souboru

1.81 Klasická pole

- $produkty = array('pneu', 'výfuk', 'sklo');
- $produkty[1]; // výfuk
- $cisla = range(1,10); // pole čísel 1-10
- Změna prvku pole: $produkty[1] = 'karosérie'
- Lze přidat nový prvek: $produkty[3] = 'karosérie'

- Procházení pole: for, foreach

1.82 Asociativní pole

- Klíče místo indexů
- $ceny = array('pneu'=>1000, 'výfuk'=>3000, 'sklo'=>2500);
- Přístup k prvku: $ceny['výfuk']
- Přidání nového prvku: $ceny['karosérie'] = 10000;
- Procházení pole – foreach, while, list, each, reset

1.83 Vícerozměrná pole (1)

Kód	Popis	Cena
PN	pneumatika	1000
VY	výfuk	3000
CS	čelní sklo	5000

```
$produkty = array(array('PN', 'pneumatika', 1000),
                  array('VY', 'výfuk', 3000),
                  array('CS', 'čelní sklo', 5000));
echo $produkty[0][1];
```

1.84 Vícerozměrná pole (2)

- Vícerozměrné asociativní pole

```
$produkty = array( array( kod=>'PN',
                    popis=>'pneumatika',
                    cena=>1000),
                  array( kod=>'VY',
                    popis=>'výfuk',
                    cena=> 3000),
                  array( kod=>'CS',
                    popis=>'čelní sklo',
                    cena=> 5000));
echo $produkty[0]['popis'];
```

- Procházení – for + while/list/each

1.85 Řazení polí

- $produkty = array('pneu', 'výfuk', 'sklo');
- sort($produkty);
- *Case sensitive*, velká písmena před malými
- Asociativní pole
  - Podle hodnoty: asort($ceny);
  - Podle klíče: ksort($ceny);
- Reverzní řazení: rsort(), arsort(), krsort()
- Řazení definované uživatelem
  - usort(pole, funkce);

1.86 Pole - další funkce

- Náhodné zamíchání pole: shuffle(pole);
- Zásobníkové vkládání a vybírání: array_push(), array_pop()
- Načtení pole ze souboru:

  $objednavky = file(soubor);
- Zjištění velikosti pole
  - count()
  - sizeof()
  - array_count_values() – vrací pole, klíče jsou původní hodnoty, hodnoty jsou počty původních hodnot
- Extrakce asoc. pole do proměnných: extract(pole)

1.87 Řetězce (1)

- Odříznutí bílých znaků: trim(), ltrim(), chop()
- Tisk řetězců
  - echo – rozdíl mezi "…" a '...'
  - printf – navíc formátování
  - sprintf – vrací zformátovaný řetězec
- Převody znaků: strtoupper(), strtolower()
- Přidání escape znaků při vkládání do databáze:

  AddSlashes(), StripSlashes()
- Rozdělování/slučování řetězců: explode(), implode()/join(), strtok()

1.88 Řetězce (2)

- Podřetězce
  - substr(řetězec, start, délka)
  - Záporná čísla znamenají od konce
- Porovnávání
  - ==
  - strcmp($ret1, $ret2) – vrací <0, 0, >0
- Délka řetězce: strlen($ret)
- Hledání podřetězce v řetězci:

```
strstr($retezec, $hledany);
```

- Pozice v řetězci: strpos($retezec, $hledany)
- Nahrazení: str_replace()

1.89 Require a include (1)

- Oba umožňují vložení kódu z jiného souboru
- Kód se parsuje – <?php ... ?>
- Rozdíly
  - Starší verze – include proveden tolikrát, kolikrát je vložen, a require jednou (nově require_once, include_once)
  - Při neexistenci způsobuje include warning, ale require *fatal error* => require použít u životně důležitých funkcí
  - Soubory vložené pomocí include mohou vracet hodnotu
  - Soubor vložený require je načten, i když je v podmínce a ta není splněná
- Proměnné dostupné v hlavním i ve vloženém souboru

1.90 Require a include (2)

- Hlavní soubor:

```
<?php require('header.php'); ?>
Obsah
<?php require('footer.php'); ?>
```

- header.php
  - Začíná <html>
  - logo, menu atd.
- footer.php
  - Patička – např. copyright

- Končí </html>

1.91 Require a include (3)

openfile.php

```php
<?php
	@ $fp = fopen($name,  $mode);
	if (!$fp)
	{
		echo "Nelze...";
		return 0;
	}
	else
	{
		return 1;
	}
?>
```

main.php

```php
<?php
	$name = 'file.txt';
	$mode = 'r';
	$result = include ('openfile.php');
	if ($result == 1)
	{
		//... lze použít $fp...
	}
?>
```

1.92 Funkce

- Jména funkcí **nejsou** *case sensitive*
- Konvence – malá písmena
- Volání: fname(parametry);
- Deklarace: function fname() {...}
- Funkce lze deklarovat odděleně + require
- Uvnitř funkce lze ukončit php kód a psát html
- Jména funkcí
  - Lze: jmeno(), jmeno2(), jmeno_tri(), _jmenoctyri()
  - Nelze: 5jmeno(), jmeno-sest(), fopen()

1.93 Parametry

- Povinné:

  function create_table($data) {...}
- Volitelné:

  function create_table($table, $border=1) {...}
- Rozsah platnosti lokálních proměnných je pouze uvnitř funkce
- Globální proměnné nejsou viditelné uvnitř funkcí (lze pomocí global $var;)
- Předávání parametrů

  - Hodnotou: function fname($par) {}
  - Odkazem: function fname(&$par) {}
- Vracení hodnot:
  - return;
  - return 5;

PHP a MySQL

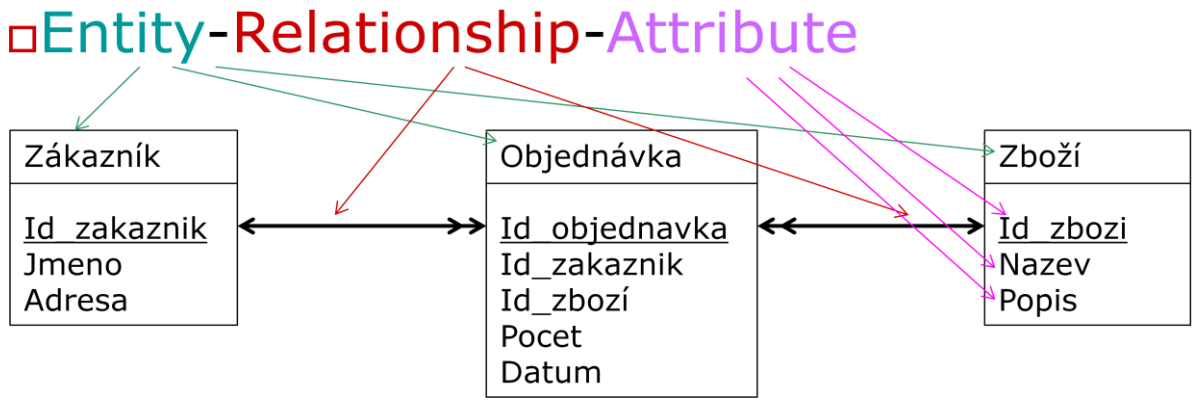
1.94 Databáze

- **Databáze** (neboli **datová základna**) je určitá uspořádaná množina informací (dat) uložená na paměťovém médiu. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim. Tento systém se v české odborné literatuře nazývá **systém řízení báze dat** (SŘBD). Běžně se označením *databáze* – v závislosti na kontextu – myslí jak uložená data, tak i software (SŘBD).

1.95 Relační databázový model

- Nejrozšířenější způsob uložení dat (v logickém smyslu)
- Sdružení dat do relací (tabulek/n-tic)
- Tabulka je struktura záznamů s pevně stanovenými položkami (sloupci/atributy)
- Sloupec definován jednoznačným názvem, typem a doménou
- Návrh databáze – ERA modely

1.96 ERA modely



- Schéma db:

  Zákazník (Id_zakaznik, Jméno, Adresa)

  Objednávka (Id_objednavka, Id_zakaznik, Id_zboží, Počet, Datum)

  Zboží (Id_zbozi, Nazev, Popis)

1.97 Vazby

- Kardinalita
  - 1:1 – oddělení choulostivých dat (zákazník – kreditní karta)

  - 1:N – nejčastější (viz předchozí příklad)
  - M:N – zákazník-zboží, musí se rozkládat na dvě vazby 1:N (viz předchozí příklad)
- Povinnost výskytu entity
  - U každé objednávky musí být zboží
  - Všechno zboží nemusí být alespoň 1x objednáno

1.98 Klíče

- **Primární klíč** – jednoznačně identifikuje záznam v tabulce
- **Cizí klíč** – primární klíč přenesený z jiné tabulky (Id_zakaznika v tabulce *Objednávka*)

1.99 Tvorba modelů

- Zamyslet se nad reálnými objekty a jejich vlastnostmi, které chcete modelovat
- Neukládat redundantní data
- Atomické hodnoty atributů
- Správný výběr klíčů
- Myslet dopředu na to, jaké budete pokládat dotazy
- Vyhněte se spoustě prázdných atributů

1.100 SQL

- *Structured Query Language*
- Standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích
- Příkazy pro manipulaci s daty
  - DML – *Data Manipulation Language*
  - SELECT, INSERT, UPDATE, DELETE, SHOW
- Příkazy pro definici dat
  - DDL – *Data Definition Language*
  - CREATE, ALTER, DROP
- Příkazy pro řízení dat
  - DCL – *Data Control Language*
  - GRANT, REVOKE, COMMIT, ROLLBACK
- Ostatní příkazy – např. nastavení formátu času, tato část není standardizována

1.101 Vytvoření tabulky

```
CREATE TABLE Zakaznik (
        Id_zakaznik INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        Jmeno VARCHAR(30) NOT NULL,
        Adresa VARCHAR(40)
);
```

- NOT NULL = hodnota atribut musí mít hodnotu
- AUTO_INCREMENT = při každém vložení záznamu do tabulky se atribut inkrementuje
- PRIMARY KEY = primární klíč
- Vymazání tabulky:

```
DROP TABLE Zakaznik;
```

1.102 Typy atributů

- **Číselné**
  - Celá čísla: INT[(M)], např INT(3) = tříciferné celé číslo, další možnosti – BIGINT, SMALLINT…
  - Reálná čísla: FLOAT, DOUBLE, např. FLOAT(6,2) = 000.00
- **Řetězcové**
  - Pevná délka: CHAR(5)
  - Proměnná délka: VARCHAR(6)
- **Textové**
  - TEXT, TINYTEXT, LONGTEXT
- **Datum a čas**
  - DATE – YYYY-MM-DD
  - TIME – HH-MM-SS
  - DATETIME – YYYY-MM-DD HH-MM-SS

1.103 Vkládání dat

```
INSERT INTO Zakaznik VALUES (NULL, 'Karel Novák', 'Plzeň');
INSERT INTO Zakaznik (Id_zakaznik, Jmeno) VALUES (NULL, 'Karel Novák');
INSERT INTO Zakaznik
SET Jmeno = 'Karel Novák', Adresa = 'Plzeň';
```

1.104 Výběr dat

- Vyber vše z tabulky Zákazník

```
SELECT *
FROM Zakaznik;
```

- Vyber pouze atributy jmeno a adresa z tabulky Zákazník

```
SELECT Jmeno, Adresa
FROM Zakaznik;
```

- Kritéria: Vyber jméno zákazníka s ID = 2

```
SELECT Jmeno
FROM Zakaznik
WHERE Id_zakaznika = 2;
```

1.105 Operátory v SQL

- =, >, <, >=, <=, !=, <>
- IS NOT NULL (Adresa IS NOT NULL), IS NULL
- BETWEEN (Cena BETWEEN 100 and 1000)
- IN (Mesto IN ('Plzeň', 'Praha')), NOT IN
- LIKE (Jmeno LIKE ('Karel %')), NOT LIKE

1.106 Výběr z více tabulek

- Př.: Vyber všechny objednávky Karla Nováka.

```
SELECT *
FROM Zakaznik, Objednavka
WHERE (Zakaznik.Id_zakaznika = Objednavka.Id_zakaznika) AND (Zakaznik.Jmeno =
'Karel Novák');
```

1.107 Použití aliasů

- Př.: Vyber všechny zákazníky a názvy jejich objednaných zboží.

```
SELECT z.Jmeno, zb.Nazev
FROM Zakaznik z, Objednavka o, Zbozi zb
WHERE (z.Id_zakaznika = o.Id_zakaznika) AND (zb.Id_zbozi = o.Id_zbozi);
```

1.108 Řazení dat

```
SELECT z.Jmeno, o.Nazev
FROM Zakaznik z, Objednavka o
WHERE (z.Id_zakaznika = o.Id_zakaznika)
ORDER BY z.Jmeno DESC;
```

- Vzestupně = ASC, sestupně = DESC

1.109 Seskupování dat

- Př.: Vypiš počty objednávek jednotlivých uživatelů.

```
SELECT z.Jmeno, COUNT(o.Id_objednavka) AS Pocet_objednavek
FROM Zakaznik z, Objednavka o
WHERE (z.Id_zakaznika = o.Id_zakaznika)
GROUP BY z.Jmeno
ORDER BY z.Jmeno;
```

1.110 Limit

- Při výběru lze určit, které řádky máme vrátit:

```
SELECT Jmeno, Adresa
FROM Zakaznik
LIMIT 2, 3;
```

- Vrací tři záznamy, první bude mít index 2 (vrácené záznamy jsou indexovány od 0).

1.111 Úprava a vymazání záznamů

```
UPDATE Zakaznik

SET Adresa = 'Praha'
WHERE Jmeno = 'Karel Novák';

DELETE FROM Zakaznik
WHERE Jmeno = 'Karel Novák';
```

1.112 Přístup do MySQL v PHP (1)

- Vytvoření spojení s databází

```
@ $db = mysql_pconnect("localhost", “root”,  “”);
if (!$db) {
        echo "Nepodařilo se připojit.";
        exit;
}
```

- Vrací handle na databázi nebo false
- pconnect = persistentní spojení s databází
- connect = jednorázové spojení s databází

1.113 Přístup do MySQL v PHP (2)

- Výběr databáze

```
mysql_select_db(“web”);
```

- Vytvoření dotazu

```
$query = "SELECT nazev, cena FROM dil";
```

- Spuštění dotazu

```
$result = mysql_query($query);
```

- Zjištění počtu vrácených záznamů

```
$pocet_zaznamu = mysql_num_rows($result);
```

1.114 Přístup do MySQL v PHP (3)

- Výběr výsledků

```
for($i=0; $i < $pocet_zaznamu; $i++)
{
        $row = mysql_fetch_array($result);
        /* zpracovani vysledku */
        echo $row['id_obj'];
}
```

PHP – OOP, chyby

1.115 OOP (1)

- Funkce a data svázány do objektů
- Objekt je definován třídou
- Třída definuje atributy a metody
- Objekt = instance třídy
- Vytvoření instance – volání konstruktoru
    - V PHP nazýván __construct()
- Dědičnost/zapouzdření/polymorfismus

1.116 OOP (2)

- Názvy tříd
    - První písmeno velké (rozlišení názvu třídy od názvu objektu
    - K simulaci zanoření jmenných prostorů lze použít podtržítko
    - Víceslovné názvy zřetězeny + první písmeno každého slova velké
    - class XML_RSS{}, class Text_PrettyPrinter
- Parametr $this je automaticky vytvořen uvnitř metody objektu a reprezentuje samotný objekt
- Pro přístup k metodám/vlastnostem použijte ->

1.117 Dědičnost

- Dědičný vztah je definován klíčovým slovem extends
- Třída dědí vlastnosti/metody nadřazené třídy, lze je přepisovat/přidat další
- class AdminUser extends User {}
- Musí se ručně volat konstruktor rodičovské třídy:

```
parent::__construct($name, $birthday)
```

1.118 Zapouzdření

- Od PHP5 lze rozdělit viditelnost dat/metod na veřejnou, chráněnou a soukromou
- public – přístupná odkudkoliv
- protected – není zvenku přístupná, ale je přístupná v podtřídě
- private – přístupná pouze uvnitř třídy, ve které je definována

1.119 Statické atributy/metody (1)

- class TestClass {public static $counter}
- Vázány přímo na třídu, ne na objekt
- Volány syntaxí: ClassName::method() a ClassName::property
- Použití $this není možné
- Změna statické vlastnosti se projeví ve všech instancích dané třídy
- self/parent – odkaz na třídu, resp. rodiče
- Přístup ke statickým proměnným uvnitř třídy:

1.120 Statické atributy/metody (2)

```
class TestClass
{
        public static $counter = 0;
        public $id;
        public function __construct()
        {
                $this->id = self::$counter++;
        }
}
```

1.121 Speciální metody

- Konstruktor: __construct()
- __destruct() – metoda volaná při rušení objektu (uvolnění zdrojů)
- V PHP4 jsou objekty předávány hodnotou:

  $obj = new TestClass(); $copy = $obj;

  (3 kopie)
- V PHP5 přiřazení vrací handle na objekt (1 kopie)

1.122 Návrhové vzory (1)

- Zevšeobecněná řešení tříd problémů, se kterými se programátoři setkávají nejčastěji
- Př. práce s DB – třída, která vytvoří spojení, vybere db, provede zadaný dotaz a vrátí výsledek
- Adaptér vzoru – poskytuje přístup k objektu prostřednictvím specifického rozhraní (rozhraní k sadě procedur)

  $dbh = new DB_WEB(*přístup k db*);

  $smtp = $dbh->execute($query);

```
// projít výsledek
```

1.123 Návrhové vzory (2)

- Šablona vzoru
  - Třída, která modifikuje logiku nadtřídy, čímž ji rozšiřuje
  - Např. skrytí parametrů připojení
  - 1 třída pro testovací DB a druhá pro ostrou
- Polymorfismus

```
function show_entry($entry_id, $dbh) {
  …
  $dbh->execute(…)
  …
}
```

1.124 Delegace

- Objekt má jako atribut jiný objekt, který používá k provádění úloh

```
class DB_WEB {
        protected $dbh;
        public function setDB($dbh) {
                $this->dbh = $dbh;
        }
        // funkce pracující s $dbh
}
```

1.125 Rozhraní

- Při delegaci musí být zaručené, že předaná $dbh implementuje potřebné metody
- Rozhraní je vlastně kostra třídy, definuje metody třídy, bez kódu

```
interface DB_Connection {
        public execute($query);
}
```

- class DB_WEB implements DB_Connection {…}
- Třída může implementovat více rozhraní -> lze obejít to, že nelze dědit od více tříd
- Abstraktní třída – může obsahovat jak hotové metody, tak i abstraktní metody, které musí být definovány potomkem

1.126 Kontrola typu

- if (!is_a($dbh,"DB_Connection"))

… chyba

- Od PHP5 – možnost kontroly typu:

  function setDB(DB_Connection $dbh) {…}

1.127 Návrhový vzor *factory*

- Změna z jedné DB na druhou = nahrazení všech výskytů třídy

  z $dbh = new DB_MySQL();

  na $dbh = new DB_Oracle();

- Řešení – funkce typu factory

```
function DB_Connection_Factory() {
        return new DB_MySQL();
}
$dbh = DB_Connection_Factory();
```

1.128 Zpracování chyb (1)

- Vnější chyby – program se nechová tak, jak by se od něj očekávalo, např. spojení s db se nezdaří
- Chyby v kódu – bugy, chybná logika/překlep
- PHP má tři úrovně závažnosti chyb
  - E_NOTICE – upozorňuje na něco, co funguje, ale možná ne tak, jak jste zamýšleli (např. použití proměnné, které ještě nebyla přiřazena hodnota)
  - E_WARNING – nezpůsobuje zastavení nebo změnu toku provádění skriptu (např. vnější chyby fopen, mysql_connect)
  - E_ERROR – chyby, ze kterých se nelze zotavit, zastaví provádění skriptu (např. vytvoření instance neexistující třídy)

1.129 Zpracování chyb (2)

- Vyvolání uživatelské chyby:

  trigger_error($message, E_USER_NOTICE)

- V php.ini lze řídit úroveň chyb, které proniknou do vašeho skriptu
- error_reporting = E_ALL ~ E_NOTICE
- display_errors = on/off
- log_errors = on/off
- error_log = cesta_k_souboru

  (logování: error_log("Retezec chyby");)

- track_errors = on, poslední chybové hlášení bude uloženo v $php_errormsg

1.130 Výjimky

- Kde obsloužit chyby – lokálně nebo u uživatele knihovny?
- Výjimka je struktura řídící tok programu
- Výjimky jsou objekty
- PHP má zabudovanou třídu Exception
- Nezachycená výjimka je fatální chybou

```
try {
        blok, kde může být výjimka vyhozena
}
catch (Exception $e) {
        obsluha výjimky
}
```

1.131 Implementace vlastní výjimky

- Oddědit od Exception
- Přepsat konstruktor:
  __construct($message=false, $code=false)
- Z __FILE__ a __LINE__ lze zjistit poslední volání
- debug_backtrace() – zpětné stopování
- Při obsluze např. print_r($e)

JavaScript

1.132 Úvod

- Multiplatformní, OOP jazyk
- Autorem je Brendan Eich z Netscapu
- Syntaxe podobná C/C++/Javě
- Standardizován v r. 1997
- Spouští se na straně klienta po stažení a zobrazení stránky
- Problém bezpečnosti uživatele – nelze pracovat se soubory

1.133 Kam umístit JavaScript?

- V záhlaví <head>

```
<script language="JavaScript">
<!—
…
// -->
</script>
```

  - JS spuštěn po zavolání (událost)
  - Jistota, že JS je načten před zavoláním
- V těle <body>
  - JS spuštěn při načítání stránky
  - Lze obojí
- Externí soubor:

```
<head><script src="xxx.js”/></head>
```

1.134 První program

```
<?xml version="1.0" encoding="windows-1250"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>
<head>
        <title>První JavaScript</title>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
        1250" />
```

```
<script language="JavaScript">
<!--
function showDateTime()
{
        var now = new Date();
        document.all.hodiny.innerText = now;
        window.status = now;
        window.setTimeout("showDateTime();", 1000);
}
// -->
</script>
</head>
<body onload="showDateTime(); return true;">
<h1>První JavaScript</h1>
<span id="hodiny"></span>
</body>
</html>
```

1.135 Proměnné a výrazy

- 3.14, 6,02e+23, “řetězec”
- Přiřazení: a=5
- Deklarace: (var a=5)
- Proměnné bez určeného typu
- a = ”řetězec”
- b = 10
- c = a + b (výsledek je “řetězec10”)
- Pole: pole[10]

1.136 Podmínky, cykly

- Podmínky stejně jako v Javě
- Cyklus while, for stejně jako v Javě
- break, continue jako v Javě
- Příkaz with
- objekt.vlastnost =

```
with (objekt) {
        příkaz;
}
```

1.137 Funkce

```
function jmeno(par1, par2, ...)
{
        tělo funkce;
}
```

- Lokální parametry: var loc = 10

```
function vzdalenost(x1, x2, y1, y2)
{
        var dx = (x2 - x1);
        var dy = (y2 - y1);
        return Math.sqrt(dx*dx + dy*dy);
}
```

1.138 Vestavěné objekty

- string – práce s řetězci
  - string.length
  - string.substring()
- Math – matematické funkce
  - Math.PI
  - Math.pom(x,y)
- Date – práce s datem a časem
  - datum = new Date(); (musíme použít konstruktor)
  - getDay()

1.139 Objekty prohlížeče – window

- window – hlavní a nadřazený objekt všech ostatních
  - window.alert("Text vypsaný metodou alert");
  - setTimeout() – poté co uplyne určený čas, provede zadaný kód
  - open(), close(), prompt(), confirm()
  - status – obsah (řetězec) stavového řádku
  - navigator – info o prohlížeči
  - location – info o URL stránky
  - history – seznam navštívených stránek
- window.document…
  - Lze psát pouze document…

1.140 Objekty prohlížeče – document

- Obsahem je stránka zobrazená v okně
- Vlastnosti a hodnoty ze zdrojového HTML
- Formulář se jménem form1 bude přístupný přes document.form1
- Obsahuje pole jako např. links nebo forms
- Vlastnosti
  - referer - odkud byl dokument načten
  - title – název stránky
  - location – url dokumentu
- document.write()

1.141 Objekty prohlížeče – form

- Součást objektu dokument
- document.form1.input1.value
- Vlastnosti
  - action – kam má být formulář poslán
  - method – get/post
  - metody – reset(), submit()

1.142 navigator, location, history

- Název prohlížeče:
  - navigator.appName
- Verze prohlížeče:
  - navigator.appVersion
- URL dokumentu: window.location
- Změna location -> přesměrování na danou stránku
- history.back(), history.forward(), history.go(-2), history.go(2)

1.143 Události – příklad

```
<head>
        <script language="JavaScript">
        <!--
        function souradnice(e)
        {
                var sou;
```

```
                    sou = "X: " + e.clientX + " ";
                    sou += "Y: " + e.clientY;

                    if (document.all){
                            document.all.souxy.
                            value = sou; }
                    else {
                            document.getElementById('souxy').value = sou;
                    }
            }
            // -->
            </script>
</head>
<body>
            <h1>Souřadnice</h1>
            <textarea id="souxy" cols="30" rows="5"
                    onmousemove="souradnice(event)">
             </textarea>
</body>
```

1.144 Události

- onclick, ondblclick
- onmouseover, onmouseout, onmousemove
- onload (po načtení dokumentu do prohlížeče), onunload (po skončení práce s dokumentem)
- onkeypress
- Objekt event – původce události
    - srcElement/target
    - clientX, clientY

1.145 Příklady

- Dynamické menu
- Kontrola formuláře

1.146 HTML DOM

- HTML Document Object Model (HTML DOM) definuje standardní způsob pro přistup a manipulaci s HTML dokumenty

- DOM bere HTML dokument jako stromovou strukturu s elementy, atributy a textem
- DOM je W3C standard, 3 úrovně (jedna z nich je HTML DOM)
- Definuje objekty, jejich vlastnosti a metody pro přistup k nim
- Celý dokument je prvek *document*
- Každý HTML tag je prvek *element*
- Text uvnitř HTML elementu je prvek *text*
- Rodiče, potomci, sourozenci

1.147 HTML DOM – vlastnosti

- x = document.getElementById("id_elementu")
- x.innerHTML – vnitřní text (HTML) elementu
- x.nodeName – Název uzlu/elementu
- x.nodeValue – hodnota elementu (vlastní text pro textový uzel)
- x.parentNode – nadřazený uzel
- x.childNodes – pole uzlů-potomků
- x.attributes – pole atributů
- objekt style – lze měnit css styl elementů

1.148 HTML DOM – metody

- x.getElementById(id) – vrací element se specifikovaným ID
- x.getElementsByTagName(name) – vrací pole elementů se specifikovaným názvem tagu
- Obě funkce lze kombinovat
- x.appendChild(node) – vloží potomka uzlu
- x.removeChild(node) – vymaže potomka uzlu

1.149 Závěr

- Projděte si tento tutoriál: http://www.w3schools.com/JS/
- Odhadem cca 2 hod
- To co je v přednášce + spousta příkladů, jednoduché vyzkoušení
- … a umíte základ JS
- Pozor na přenositelnost – aplikaci vždy vyzkoušejte alespoň pod Firefoxem a Explorerem
- Vyskakovací okna, pokud to jde, nepoužívat

HTTP

1.150 HTTP (1)

- **Protokol** je soubor syntaktických a sémantických pravidel určujících výměnu informace mezi nejméně dvěma entitami spojenými například prostřednictvím počítačové sítě.
- HTTP - určený původně pro výměnu hypertextových dokumentů ve formátu HTML
- V současné době je používán i pro přenos dalších informací
- Rozšířením MIME umí přenášet jakýkoliv soubor
- **MIME** (**Multipurpose Internet Mail Extensions**) je internetový standard, který rozšiřuje původní standard RFC822 o možnost posílání zpráv s diakritikou, obrázků, zvuků, elektronický podpis a šifrování zprávy atd.

1.151 HTTP (2)

- Používá se společně s formátem XML pro tzv. webové služby (spouštění vzdálených aplikací)
- Aplikačními branami zpřístupňuje i další protokoly, jako je např. FTP nebo SMTP
- Používá obvykle port TCP/80
- Verze 1.1 protokolu je definována v RFC 2616 (http://tools.ietf.org/html/rfc2616)
- HTTP používá jako některé další aplikace tzv. jednotný lokátor prostředků (**URL**, *Uniform Resource Locator*), který specifikuje jednoznačné umístění nějakého zdroje v Internetu.
- K protokolu HTTP existuje také jeho bezpečnější verze HTTPS, která umožňuje přenášená data šifrovat a tím chránit před odposlechem či jiným narušením.

1.152 Činnost protokolu

- Dotaz-odpověď
- Uživatel/klient/prohlížeč pošle serveru dotaz ve formě čistého textu
    - Označení požadovaného dokumentu
    - Informace o schopnostech prohlížeče
- Server poté odpoví několika řádky textu popisujícími výsledek dotazu
    - Zda se dokument podařilo najít
    - Jakého typu je dokument

- Následují data samotného požadovaného dokumentu
- Další dotaz nezávisí na předchozím -> protokol je **bezstavový**
    - Jak uchovat např. identitu zákazníka?
    - HTTP byl rozšířen o *cookies*
        - Uchování informací o stavu spojení na počítači uživatele

1.153 Ukázka komunikace – dotaz

- Uživatelský program se připojí na server www.kiv.zcu.cz a zašle následující text

```
GET /studies/vyhlasky HTTP/1.1
Host: www.kiv.zcu.cz
User Agent: Mozilla/5.0 Gecko/20040803 Firefox/0.9.3
Accept-Charset: UTF-8,*
```

1.154 Ukázka komunikace – odpověď

```
HTTP/1.0 200 OK
Date: Fri, 15 Oct 2004 08:20:25 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.8
X-Powered-By: PHP/4.3.8
Vary: Accept-Encoding,Cookie
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: cs
Content-Type: text/html; charset=utf-8
```

- Následuje prázdný řádek = konec hlavičky, pak dokument

1.155 Dotazovací metody

- HTTP definuje několik metod, které se mají provést nad uvedeným objektem (dokumentem)
- GET - požadavek na uvedený objekt.
- HEAD - to samé jako metoda GET, ale už nepředává data. Poskytne pouze metadata o požadovaném cíli (velikost, typ, datum změny, …).
- POST - odesílá uživatelská data na server. Používá se například při odesílání formuláře na webu. S předaným objektem se pak zachází podobně jako při metodě GET.
- PUT - nahraje data na server. Objekt je jméno vytvářeného souboru.
- DELETE - smaže uvedený objekt ze serveru. Jsou na to potřeba jistá oprávnění stejně jako u metody PUT.

- TRACE - odešle kopii obdrženého požadavku zpět odesílateli, takže klient může zjistit, co na požadavku mění nebo přidávají servery, kterými požadavek prochází.
- OPTIONS - dotaz na server, jaké podporuje metody.
- CONNECT - spojí se s uvedeným objektem přes uvedený port. Používá se při průchodu skrze proxy pro ustanovení kanálu SSL.

1.156 Stavové kódy

- 1xx – informační
  - 100 *continue* – klient je informován, že server přijal část požadavku, a může pokračovat při jeho odesílání
- 2xx – úspěch – požadavek bych přijat a akceptován
  - 200 OK – server vrací data dle metody požadavku
- 3xx – přesměrování – klient musí provést další akci, aby byl splněn požadavek
- 4xx – chyba klienta
  - 401 *Unauthorized* – požaduje se autentizace uživatele
  - 402 *Payment Required* – rezervováno pro budoucnost
  - 403 *Forbidden* – server zakazuje vstup na stránku
  - 404 *Not Found* – server nenalezl pro URI v požadavku žádný dokument
- 5xx – chyba na straně serveru
  - 500 *Internal Server Error* – např. vyhození neodchycené výjimky
  - 503 *Service Unavailable* – dočasné přetížení serveru nebo jeho údržba

1.157 HTTPS

- Nadstavba HTTP
- Poskytuje zvýšenou bezpečnost před odposloucháváním či podvržením dat
- Data přenášena protokolem HTTP
- Jsou šifrována pomocí SSL nebo TLS, což zaručuje ochranu proti *packet-sniffingu* i *man-in-the-middle* útokům
- Implicitní port 443 (u HTTP je to 80)
- Pro komunikaci pomocí HTTPS musí nejdříve server vlastnit certifikát – digitálně podepsán certifikační autoritou

AJAX

1.158 Co je AJAX

- AJAX není nový programovací jazyk, ale nový způsob použití existujících standardů
- Rychlejší a uživatelsky přívětivější aplikace
- Založen na JavaScriptu a HTTP
- JavaScript může komunikovat přímo se serverem díky objektu **XMLHttpRequest**
    - **Lze vyměňovat data se serverem bez obnovení stránky**
- Asynchronní přenos dat mezi prohlížečem a serverem
- Lze žádat o část stránky (dat), místo celé stránky
- Je to klientská technologie, nezávisí na technologii serveru
- Populární od r. 2005 – Google suggest
- Prohlížeč musí podporovat objekt XMLHttpRequest (různé metody vytvoření v prohlížečích)

1.159 Objekt XMLHttpRequest (1)

- Odlišné vytvoření v různých prohlížečích
- Vlastnost onreadystatechange – obsahuje funkci, která zpracuje odpověď serveru
- Vlastnost readyState obsahuje stav odpovědi
    - Vždy, když je hodnota změněna, spustí se onreadystatechange

      0 = požadavek nebyl inicializován

      1 = požadavek byl připraven

      2 = požadavek byl odeslán

      3 = požadavek je zpracováván

      4 = hotovo
- Test, zda je požadavek hotov (máme celou odpověď)

```
xmlHttp.onreadystatechange=function() {
        if (xmlHttp.readyState==4) {
                // Vezmi data z odpovědi serveru
        }
}
```

- V xmlHttp.responseText je text získaný ze serveru (skriptu)

1.160 Objekt XMLHttpRequest (2)

- Vlastnost responseText obsahuje odpověď
- Metoda open
  - 1. par. – metoda GET/POST
  - 2. par. – URL skriptu na serveru
  - 3. par. – true = asynchronní zpracování
- Metoda send
  - par. – *request*
  - např: 'par1=hodn1& par2=hodn2'

1.161 Serverový skript

- Co vypíše echem, se ukládá do xmlHttp.responseText
- (v případě vygenerování XML – responseXML)
- V případě jednoduché odpovědi řetězec
- V případě strukturované odpovědi XML
- V PHP skriptu je nutno nastavit hlavičku

```
header('Content-Type: text/xml');
```

1.162 Příklady

- Hint
- DB
- XML + DB
  - Verze s textovým vytvořením XML
  - Verze s PHP DOM
- Ajax tutoriál: http://www.w3schools.com/ajax/default.asp
- Plugin Firebug pro Firefox
  - Pomoc při ladění JS
  - Lze sledovat JS volání, chyby
  - https://addons.mozilla.org/cs/firefox/addon/firebug/

Servlety, šablony, konfigurace

1.163 CGI - Common Gateway Interface

- 1993 – Rob McCool
- Protokol pro propojení externích aplikací s webovým serverem
- Umožňuje serveru delegovat požadavek od klienta na externí aplikaci, která dle požadavku vrátí výstup
- Aplikace zpracuje nějaký skript ve webové stránce a webovému serveru vrátí statickou stránku, která je následně poslána klientovi jako výstup jeho požadavku

1.164 CGI (2)

- Na webovém serveru jsou definovány lokace (http://www.kiv.zcu.cz/prog.cgi), které budou obsluhovány CGI programem
- Kdykoliv přijde požadavek, který se shoduje s tímto URL, zavolá se definovaný program
- Data od klienta jsou vstupem programu
- Webový server potom obdrží výstup programu, přidá hlavičky a pošle zpět klientovi
- Pro každý požadavek vzniká nový proces – zvyšuje zatížení serveru (moduly serveru jako PHP, Perl jsou efektivnější)
- Programy (*exe*) zpravidla na web. serveru v adresáři *cgi-bin*

1.165 Java - servlety a JSP (1)

- Javovská technologie pro tvorbu dynamických webů
- 1997 – Sun Microsystems – spec. 1.0
- 2006 – spec 2.5
- Servlet API (javax.servlet) definuje interakci mezi webovým kontejnerem a servletem, interakce se servlety
- Web container
    - komponenta webového serveru
    - Řídí životní cyklus servletů
    - Mapuje URL na určitý servlet
    - Kontroluje přístupová práva
    - Apache Tomcat, Winstone, Jetty

1.166 Java - servlety a JSP (2)

- Servlet je objekt, který obdrží požadavek a vygeneruje odpověď
- Hlavní třída pro vytváření servletů: javax.servlet.http
- Servlety mohou být zabaleny ve WAR souborech jako webové aplikace
- Udržování stavu
  - *Cookies*
  - *Sessions*
  - *URL rewriting*

1.167 Java - servlety a JSP (3)

- Servlety mohou být generovány automaticky – *Java Server Pages* (JSP)
- Aplikace často obsahuje jak JSP, tak servlety
- MVC (*Model-View-Controller*)
- Model = třídy, které zajišťují mapování dat z DB do objektů (*beany*)
- View = JSP stránky – řídí to, jak se stránky zobrazí (HTML + vkládání dynamických dat)
- Controller = servlety, které celou aplikaci řídí (přesměrování na JSP atd.)

1.168 Python a Zope (1)

- Python – vysokoúrovňový jazyk
- 1991 – Guido van Rossum
- Důraz na rychlé programování, ne efektivitu
- Funkce, OOP, dynamický datový systém, automatická správa paměti, odsazování bloků (ne závorkování)
- Open source

1.169 Python a Zope (2)

- Zope – open source, OOP webový server napsaný v Pythonu
- Publikuje objekty Pythonu (objektová databáze) – tedy ne soubory jako např. PHP, JSP a podobné
- Uživatel může manipulovat s objekty přes web
- Zope mapuje URL na objekty – možné i hierarchie objektů (http://www.site.cz/main_object/inner_object), podobně s metodami objektu
- Mechanismy pro tvorbu HTML šablon

1.170 MPO (MVC)

- Často aplikovaný model při programování webu
  - MPO – Model, Pohled, Ovladač
  - MVC – Model, View, Controller
- Model – provádí obchodní logiku (práce s DB)
- Pohled – část zajišťující formátování výstupu systému
- Ovladač – zpracovává vstup a předává ho modelu, řídí aplikaci
- Oddělení aplikační logiky od zobrazování je vhodné
  - Aplikace je pružnější, snadno modifikovatelná
  - Kód vypadá přehlednější
  - Roste opětovná použitelnost částí systému

1.171 Šablony

- Implementace MPO na webu pomocí šablon
- HTML a logika zobrazení obsažena v šabloně
- Aplikační kód neobsahuje žádnou logiku zobrazení, pouze zpracuje požadavek, provede potřebné úkony a předá data šabloně
- Jeden z nejpopulárnějších šablonových systémů – **Smarty**

1.172 Smarty

- Používá speciální značky v souborech šablon
- Šablony jsou kompilovány do cachovaného PHP skriptu
- Instalace
  - Vytvořit adresář pod PHP, zkopírovat do něj stažené knihovny (*smarty.php.net*)
    Do include cesty v php.ini přidat tento adresář (knihovnu Smarty lze kopírovat i s aplikací)
- Vytvořit adresář, kde může smarty načítat svoji konfiguraci a soubory šablon: \templates a \smarty_config

1.173 Konfigurace Apache (1)

- Soubor httpd.conf
- Některá implicitní nastavení nemusí být optimální
- Pro zviditelnění změn se musí server restartovat

- # pro zakomentování řádky
- ServerRoot
    - Cesta ke konfikuraci, logům atd.
    - Místo, kde je Apache nainstalován

1.174 Konfigurace Apache (2)

- Listen 80
- Port, na kterém webserver poslouchá
- ServerName localhost
    - Adresa serveru (www.kiv.zcu.cz)
    - Lokálně při ladění *localhost*
- DocumentRoot
    - Cesta ke kořenovému adresáři webu
    - Standardně ukazuje do htdocs, lze změnit

1.175 Konfigurace Apache – Directory

- Nastavení oprávnění k jednotlivým adresářům
- Nastavení bezpečnosti webu
- Na začátku cesta a jméno adresáře
- Pokud na jeden adresář sedí více pravidel, uplatní se od nejméně specifických k nejvíce
- Pro nastavení všech adresářů použít jako jméno /
- Pravidla se uplatní i na podadresáře
- Lze nastavovat oprávnění i pro soubory
- Deny/Allow from all – zakazuje/povoluje přístup odkudkoliv

1.176 Konfigurace Apache – Aliasy

- Odkazy, kterými můžeme do kořenového adresáře webu přidat adresář, který se nachází někde jinde
- Př.: Alias /mysql/ “c:/phpmyadmin”
- Musí být zapnutý modul mod_alias
- Měly by se nastavit práva pro tento adresář

1.177 Directory index, ErrorDocument

- DirectoryIndex

    - Jména souborů, která jsou brána jako indexová
    - Soubor se použije po zadání adresy bez určení souboru
    - Hodnota např. index.html index.php
- ErrorDocument 404 /404.html
- Lze nastavit různý vzhled chybových hlášení

1.178 Konfigurace Apache - .htaccess

- Zvláštní textový soubor určený k tomu, aby si autor stránek mohl sám upravit některé vlastnosti serveru Apache
- Umístěn v adresáři, pro který chceme měnit vlastnosti (instrukce jako v httpd.conf)
- Příklad – vlastní chybová stránka
- Soubor je skrytý (. na začátku)
- Viz nastavení v httpd.conf

1.179 Konfigurace PHP

- Propojení Apache s PHP v konf. Apache:
    - PHPIniDir
- Konfigurace PHP – php.ini
    - Jméno vlastnosti = hodnota
    - ; pro komentář
    - Defaultně v adresáři PHP
    - Mnoho voleb:
        - Register-globals = off
        - error_reporting = E_ALL
        - display_errors = On
        - Session.gc_maxlifetime = 1440

Web a bezpečnost

1.180 Operační systém

- Záplatovat operační systém
- Mít aktuální antivirový program a firewall (někdy v OS)
- Dobré jsou také jiné programy na *malware*

1.181 Prohlížeč

- Aktualizovat
- Povolit nebo zakázat?
    - Cookies
    - JavaScript
    - Vyskakovací okna
    - Uložení informací z formulářů a textových polí („doplňování slov“)
    - ActiveX a jiný aktivní obsah
- Je možnost za sebou zamést stopy
    - Smazat historii prohlížení a hledání
    - Smazat cookies
    - Smazat hesla a uložené informace z formulářů a textových polí
    - Smazat dočasně uložené stažené soubory

1.182 Chování na webu (1)

- Formuláře
    - Neposkytovat nepovinné údaje
    - Vhodné mít „soukromou“ a „veřejnou“ e-mailovou adresu
- Sociální sítě (Facebook, Twitter…)
    - Skutečně na sebe chci všechno prozradit? (soukromé údaje, fotky…)
    - Účet na Facebooku nelze **smazat**, jen **deaktivovat**
    - Profilová data zůstávají a co se s nimi pak děje?
- Komunikační programy (Skype, ICQ, FreeCall…)
    - Poskytovat jen minimum informací o sobě
    - Udržovat *blacklisty* a *whitelisty*

1.183 Chování na webu (2)

- Internetové nákupy a bankovnictví
  - Bankovní převod bezpečnější než jiné způsoby plateb
  - Nastavit limit pro platby kreditní kartou
  - Možnost povolit a zakázat platby po internetu pro každý nákup
- Osobní webové stránky
  - Co je na webu, je veřejné
  - E-mailovou adresu nepsat jako text kvůli robotům

1.184 Facebook?

- Mark Zuckerberg: „Éra soukromí skončila“
- http://download.zcu.cz/public/Prezentace/seminare_CIV_2011/Odvracena_strana_socialnich_siti.pdf

Literatura

Leiss, Oliver; Schmidt, Jasmin. *PHP v praxi: pro začátečníky a mírně pokročilé.* Vyd. 1. Praha: Grada Publishing, 2010.

Nixon, Robin. *Learning PHP, MySQL, and JavaScript: A Step-By-Step Guide to Creating Dynamic Websites (Animal Guide).* Vyd. 1. Sebastopol: O'Reilly Media, 2009.

Schafer, Steven M. *HTML, XHTML a CSS: Bible pro tvorbu WWW stránek*. Vyd. 4. Praha: Grada Publishing, 2009.

W3C doporučení XHTML 1.0: http://www.w3.org/TR/xhtml1/

W3C návrh HTML5: http://www.w3.org/TR/html5/

W3C návrh CSS3: http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/

W3C specifikace CSS 2.1: http://www.w3.org/TR/CSS21/

W3C specifikace HTML 4.01: http://www.w3.org/TR/html4/