UNIVERZITA J. E. PURKYNĚ V ÚSTÍ NAD LABEM
Přírodovědecká fakulta

Programování pro internet

KI/PIN

doc. Viktor Maškov, DrSc.

Ústí nad Labem 2020

Úvodní slovo
Kurz dává studentům teoretické znalosti XML technologii a praktické zkušenosti práce s XML soubory. V kurzu se uvažuje návrh webových aplikací (včetně aplikací pracujících s databázemi). V průběhu kurzu budou vytvořeny a zpracovány několik souborů XML. Uvažují se otázky extrakce informací z obsahu XML a její zobrazení na webových stránkách (s použitím css, xslt, js a php). V průběhu kurzu budou vytvořeny webové aplikace, jejichž návrh zahrnuje rovněž témata jako AJAX a práce s XML soubory, cookies a sezení, objektově orientované programování, grafiku, práce s databázemi.

Obsah:

Kapitola 1. Značkovací jazyk XML

Cíl kapitoly:
- Prakticky použit značkovací jazyk XML
- Procvičit vytvoření souborů XML

Slovníček pojmů: značkovací jazyk XML, syntaxe, elementy XML, XML atributy

Výkladová část:
Syntaxe
1. První řádek → XML deklarace (XML verzi a kódovaní)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

- implicitně je dokument v kódu ISO 10646;
- pro komunikaci se světem se používá UTF-8 (kompatibilní s ASCII). Obsahuje všechny znaky všech abeced;
- pro češtinu lze použít ISO-8859-1 (Latin-1/West European) nebo Windows-1250;

2. Každý element musí mít počáteční a koncovou značku

```
<dopis> ……. ….. </dopis>
```

<dopis> - počáteční značka

</dopis> - koncová značka
Deklarace není XML element, a proto nemusí mít koncovou značku.
3. Značky rozlišují malá a velká písmena
<Message>správný</message>
<message> není správný </message>
4. XML elementy musí být dobře vnořeny do sebe (nesmí se křížit).
Text je bold a italic
5. XML dokument musí mít kořenový element. Všechny ostatní elementy musí být uvnitř kořenového elementu.
6. Hodnoty atributů musí být v uvozovkách.

XML elementy

- XML dokumenty (elementy) jsou rozšiřitelné.
- XML elementy mají vztah.
  Vztah mezi elementy je vnímán jako vztah mezi rodiči a dětmi.
- Elementy mají obsah.
  Obsah: elementy; směsný; jednoduchý; prázdný.
- XML element může mít atributy v počáteční značce.
  <dopis datum="12.02.2008">
- Pojmenování elementů.
  Pravidla:
  - jméno může obsahovat písmena, čísla a jiné znaky;
  - jméno nesmí začínat číslem nebo interpunkčním znakem;
  - jméno nesmí začínat xml
  - jméno nemůže obsahovat mezery.
  - " : " neměl by být použit

XML atributy

Atributy se používají, aby dodaly další informace o elementu.
Často atributy dodávají informaci, která je irelevantní datem, ale je důležitá pro SW, který bude zpracovávat tento element.
<file type="gif">computer.gif</file>

Zadání pro cvičení:

1. Vytvořit XML soubor „Studium"
- údaje (student);
- pro každý studijní rok a pro každý semestr (studijní údaje).
2. Vytvořit XML soubor „Fakulta"
- obecná informace;
- informace pro každou katedru.
3. Vytvořit XML soubor „Rodina"
- údaje omezit na rodiče, děti a sourozenci (bratr, sestra).

Úkoly pro samostatnou práci:
Prostudovat značkovací jazyk XML. Použit prezentaci na vtan.ujep.cz (Složka IS, soubor L-Xml-1.ppt).

Kapitola 2. Specifikace XML (DTD)

Cíl kapitoly:

- Prakticky použit DTD
- Procvičit vytvoření DTD a validaci XML souborů

Slovníček pojmů: DTD (definice typu dokumentu), deklarace elementů, deklarace atributů, deklarace entit.

Výkladová část:
DTD definuje legální elementy XML dokumentu.
DTD definuje strukturu dokumentu.
• Definice typu dokumentu (Document Type Definition - DTD) popisuje pomocí regulérních výrazů strukturu (gramatiku) dokumentu.
• Deklarace typu XML dokumentu může odkazovat na externí entitu, která obsahuje deklaraci značek, nebo může tuto deklaraci obsahovat přímo - interně, nebo obojí.
• DTD je tvořena spojením těchto deklarací, interní mají přednost.

Příklad: externí a interní DTD
- Externě:

```
<?xml version="1.0"?>
<!DOCTYPE dopis SYSTEM „dopis.dtd">
<dopis>Sdělení</dopis>
```

- Interně:

```
<!DOCTYPE root-element [element-declarations]>
                or
<?xml version="1.0" encoding=„ISO-8859-1"?>
<!DOCTYPE dopis [
<!ELEMENT dopis (#PCDATA)>
]>
<dopis>Sdělení</dopis>
```

■ Z hlediska DTD všechny XML dokumenty jsou vytvořené z následných bloků:
- Elementy;
- Atributy;
- Entity;
- PCDATA;
- CDATA.

■ Základní značky DTD
• deklarace typu dokumentu - <!DOCTYPE … >
• deklarace typu elementu - <!ELEMENT … >
• deklarace seznamu atributů - <!ATTLIST … >
• deklarace entity - <!ENTITY … >

Úplná deklarace elementu s potomky

```
<!ELEMENT rodic (potomek1, potomek2)>
<!ELEMENT potomek1 (#PCDATA)>
<!ELEMENT potomek1 (#PCDATA)>
```

■ Regulérní výrazy v deklaracích DTD
, … sekvence
| … selekce
? … iterace (0 nebo 1)
+ … iterace (1 a více)
… iterace (0 a více)

Deklarace atributu
<!ATTLIST element-jmeno atribut-jmeno atribut-typ implicitni-hodnota>
Příklad.
DTD: <!ATTLIST sázka type CDATA “check” >
XML: <sázka type=“check”/>

■ Typy hodnot atributů:
• CDATA - řetězec
• ID - jednoznačný identifikátor (v rámci dokumentu)
• IDREF - odkaz na ID jiného elementu (IDREFS -
elementů) v rámci dokumentu
• ENTITY - odkaz na jméno entity, která není analyzována
XML procesorem, ale jinou aplikací (ENTITIES- seznam entit)
• NMTOKEN - hodnota (validní XML jméno)
• výčtový typ

Deklarace atributů
<!ATTLIST mark
cislo ID #REQUIRED
vypisovat CDATA #FIXED "yes"
typ (natural|adopted) "natural">

• atributy elementu mark
• atribut cislo je unikátní identifikace (ID) a je povinný (#REQUIRED)
• atribut vypisovat obsahuje text (CDATA), je konstantní (#FIXED) a má implicitní neměnnou
hodnotu (yes)
• atribut typ je výčet (natural nebo adopted), implicitní hodnota je natural
Příklad použití ID a IDREF

<?xml version="1.0" encoding="windows-1250"?>
<!-- DTD pro tabulku zaměstnanců -->
<!ELEMENT zaměstnanci (zaměstnanec)*>
<!ELEMENT zaměstnanec (jméno,nástup,plat)>
<!ATTLIST zaměstnanec
ČÍSLO ID #REQUIRED
VEDOUCÍ IDREF #IMPLIED>
<!ELEMENT jméno (#PCDATA)>
<!ELEMENT nástup (#PCDATA)>
<!ELEMENT plat (#PCDATA)>

Deklarace entit
<!ENTITY STATEMENT
"This is well-formed XML">
• Entita je něco, co lze použít kdekoliv v XML dokumentu
• Zde je definována entita STATEMENT
• Libovolný odkaz ve tvaru &STATEMENT; je v textu nahrazen řetězcem
This is well-formed XML

Externí deklarace entity
<!ENTITY entity-jmeno SYSTEM “URL”>
Příklad.
DTD: <!ENTITY book SYSTEM “http://www.someweb.com/book.dtd”>
XML: <catalog>&book;</catalog>

Interní deklarace entity
<!ENTITY entity-jmeno “entity-hodnota”>
Příklad.
DTD: <!ENTITY autor “Tolstoj”>
XML: <spisovatel>&autor;</spisovatel>

Zadání pro Cvičení:
1. Vytvořit DTD pro soubor „Studium“
2. Vytvořit DTD pro soubor „Fakulta“
3. Vytvořit DTD pro soubor „Rodina“

Úkoly pro samostatnou práci:
Prostudovat DTD. Použit prezentace na vtan.ujep.cz (Složka IS, soubor L-Xml-1.ppt).

Kapitola 3. XML Schéma

Cíl kapitoly:
- Prakticky použit XML Schéma
- Procvičit vytvoření XML Schéma a validaci XML souborů

Slovníček pojmů: syntaxe XML Schéma, XSD jednoduché elementy, XSD atributy, Fazety, XSD komplexní elementy.

Výkladová část:
- XML Schéma je alternativa pro DTD, popisuje strukturu XML dokumentu a definuje legální stavební bloky dokumentu.
- Jazyk XML Schéma uveden jako XML Schéma Definition (XSD).

Struktura bloky a dokumentu:
- Elementy, atributy, elementy-potomky, počet potomků, pořadí potomků, typy dat elementů a atributů, implicitní a fixované hodnoty elementů a atributů, atd.

Výhody XML Schema:
□ schopná rozšíření
- znovu použít své Schéma v jiných Schématech
- vytvořit vlastní typy dat odvozené ze standardních typů
- použít v jednom XML dokumentu odkazy na několik XML Schémat

□ napsána v jazyce XML
□ podporuje typy dat a prostor jmen

Syntax XML Schéma
- musí začínat s XML deklarace;
- musí mít jeden unikátní kořenový element;
- počáteční značka musí mít odpovídající koncovou značku;
- elementy jsou citlivé na velikost písma;
- všichni elementy musí být zavřené;
- elementy musí být náležitě vnořené
- všichni hodnoty atributů musí být v uvozovkách
- entity musí být použité pro speciální znaky

Deklarace Schématy často vypadá takto:

```
</ xml version=“1.0” ?>
<xs:schema xmlns:xs=“http://www.w3.org/2001/XMLSchema”
                targetNamespace=“http://www.someweb.com”
                xmlns=“http://www.someweb.com
                elementFormDefault=“qualified”>
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
</xs:schema>
```

■ Následující fragment

```
xmlns:xs=“http://www.w3.org/2001/XMLSchema”
```

ukazuje, že elementy a typy dat, které budou použité ve Schématu, pocházejí z "http://www.w3.org/2001/XMLSchema" prostoru jmen.
Fragment ukazuje také, že tyto elementy a typy dat musí být použité s předponou xs:

■ Následující fragment
targetNamespace="http://www.someweb.com"
ukazuje, že elementy definované v tomto Schématu (dopis, komu, odkoho, hlavicka, obsah) pocházejí z "http://www.someweb.com"

■ Následující fragment
xmlns="http://www.someweb.com"
ukazuje, že implicitní prostor jmen je "http://www.someweb.com"

■ Následující fragment
elementFormDefault="qualified"
ukazuje, že každý element, který bude použít v konkrétním XML dokumentu, a je deklarovaný v toto Schémata, musí být kvalifikovaný prostorem jmen.

XSD jednoduché elementy

■ XML Schéma definuje elementy XML souborů.
Jednoduchý element – je XML element, který obsahuje pouze text.
Jednoduchý element nemůže obsahovat jiné elementy a atributy.
■ Text může být různého typu:
- standardního typu (boolean, string, date, integer atd.)
- uživatelského typu (definujete sami)
■ Je možné použít omezení na data nebo potřebovat aby data odpovídali určité šabloně.

Definování jednoduchého elementu
Syntax:

```
<xs:element name="xxxxx" type="yyyyy"/>
```

Vestavěné typy dat:
- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

XSD atributy (xs:attribute)

■ Všichni atributy jsou deklarované jako jednoduchý typ.
Syntaxe:

```
<xs:attribute name="xxxxx" type="yyyyy"/>
```

Vestavěné typy dat:
- xs:string - xs:decimal - xs:integer - xs:boolean - xs:date - xs:time

Fazety

XML Schema umožňuje dodat vaše vlastní omezení pro elementy a atributy. Tyto omezení se nazývají „fazety".
1. Omezení hodnot
Následující příklad definuje element „cena" s omezením. Hodnota elementu (cena) nemůže být nižší než 10 a větší než 100.
2. Omezení množiny hodnot (tj. definuje hodnoty, které jsou dovolené)

Aby omezit obsah XML elementu tak, že hodnotami elementu mohou být jenom určitá množina hodnot, musíme použit omezení typu výčet.
3. Omezení na řetězec čísel nebo písmen, které mohou být použité v obsahu elementu.
Pro také omezení se používají vzorek omezení.
Příklad definuje element „dopis“ s omezením. Přípustnou hodnotou pro element je pouze hodnota, která sestává z jednoho malého písmena (od a do z):

```
<xs:element name="dopis">
  <xs:simpleType>
     <xs:restriction base="xs:string">
       <xs:pattern value="[a-z]"/>
     </xs:restriction>
  </xs:simpleType>
</xs:element>
```

4. Další omezení hodnot
Příklad definuje element „dopis“ s omezením. Přípustnou hodnotou pro element je pouze hodnota, která sestává z malé písmen (od a do z), přičemž každé písmeno se může vyskytnout 0 (nula) nebo více krát:

```
<xs:pattern value="([a-z])*"/>
```

5. Omezení pro neviditelné znaky (bílé znaky)
Bílé znaky: posuv o řáder; znak tabulátoru (šipka); mezery.
■ Aby určit jak bílé znaky musí být zpracované, můžeme použit speciální omezení WhiteSpace.
6. Omezení délky
Pro omezení délky hodnoty se používá omezení: length, maxLength a minLength.

XSD komplexní elementy
■ Komplexní element obsahuje další elementy a/nebo atributy.
Jsou čtří typy komplexních elementů:
1) prázdné elementy;
2) elementy, které obsahují pouze druhé elementy;
3) elementy, které obsahují pouze text;
4) elementy, které obsahují jak druhé elementy, tak i text
■ Každý z těchto elementů může rovněž obsahovat i atributy.

Indikátory
■ Pomocí indikátorů je možné kontrolovat použití elementů XML dokumentu.
Existují 7 indikátorů:
Indikátory pořadí:
- All
- Choice
- Sequence

2) Indikátory výskytu:
- maxOccurs
- minOccurs

3) Indikátory skupiny:
- Groupname
- attributeGroup name

Zadání pro Cvičení:
1. Vytvořit XML Schema pro soubor „Studium“.
2. Provést seskupování elementů v souborech Fakulta.xsd a Studium.xsd. Udělat odkaz na vytvořenou skupinu.
3. Provést seskupování atributů v souborech Fakulta.xsd a Studium.xsd. Udělat odkaz na vytvořenou skupinu.
4. Nahrazení elementu (Element substitution). Umožnit napsání v jiném jazyce názvu elementu ROK“ (např. YEAR). Provést potřebné změny v Studium.xsd.

5. Pro simpleType použit co nejvíce odpovídající vestavěný Typ!

Úkoly pro samostatnou práci:
Prostudovat XML Schema. Použit prezentace na vtan.ujep.cz (Složka IS, soubor L-xml-3-1.ppt).

Kapitola 4. Zobrazení XML souborů

Cíl kapitoly:
- Vysvětlit metody zobrazení XML souborů.
- Procvičit použití různých metod zobrazení XML souborů.

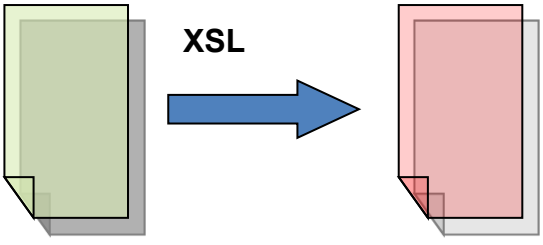
Slovníček pojmů: css, xslt, javascript, elementy XSLT,

Výkladová část:
■ Prohlížeč zobrazí XML dokument v původním tvaru.
■ Existují různá řešení problému zobrazení XML dokumentů pomocí:
- CSS;
- XSL;
- JavaScript;
- XML Data Islands (XML ostrůvky dat).

1. Použití CSS
Pomocí CSS můžeme formátovat XML dokument.
XML soubor:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="table.css" ?>
<table>
<name>Stůl</name>
<width>100</width>
<height>50</height>
</table>
```

2. Použití XSL
XSL (eXtensible Stylesheet Language). XSL popisuje, jak by XML element měl být zobrazen.



Připojení XSL souboru k XML dokumentu

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="sbirka.xsl"?>
<sbírka>
…….
</sbírka>
```

Elementy XSLT
- <xsl:template>
- <xsl:value-of>

- <xsl:for-each>
- <xsl:sort>
- <xsl:if>
- <xsl:choose> (<xsl:when> a <xsl:otherwise>)
- <xsl:apply-templates>

Element <xsl:template>
■ XSL Style Sheet sestává z jednoho nebo více pravidel, která se jmenují šablony (templates).
■ Každá šablona obsahuje pravidla, která mají být aplikována na uzel (element), specifikovaný pomocí atributu match.
■ Atribut match může také definovat šablonu pro kompletní XML dokument.
Hodnotou atributu match je XPath výraz
(např. match="/" definuje celý dokument, t.j. šablona bude aplikována na kořenový element dokumentu).

■ Obsahem elementu <xsl:template> je HTML kód, který bude interpretován prohlížečem při zobrazení dokumentu.

Element <xsl:value-of>
■ Element <xsl:value-of> se používá pro kopirování hodnoty uzlu, vybraného pomocí atributu select, a vložení zkopirované hodnoty do výstupního souboru.
■ Hodnotou atributu select je XPath výraz.
XPath "pracuje" podobné navigace v systému souborů (lomítko (/) vybírá podadresáře).

Element <xsl:for-each>
■ Element <xsl:for-each> umožňuje vybrat každý element ze specifikované množiny uzlů.
■ Množina uzlů může být specifikovaná pomocí atributu select.
<xsl:for-each select="sbírka/kniha">
V případě, že element <sbírka> obsahuje více než jeden potomek <kniha> , bude vybrán každý element <kniha>.

Filtrování pro výstupní soubor
Selekční hranice se přidává do atributu select

<xsl:for-each select="sbírka/kniha[autor='Gogol']">

Legální operace:
= (rovná se) např. [autor='Gogol']
!= (nerovná se)
< (méně než)
> (větší než)

Element <xsl:sort>
■ Element <xsl:sort> slouží pro uspořádání výstupní informace
■ Element <xsl:sort> se nachází uvnitř elementu <xsl:for-each>
■ Atribut select ukazuje na element, který má být uspořádán
<xsl:sort select="rok_vydání"/>

Element <xsl:if>
■ Element <xsl:if> se používá pro spuštění podmínkového testu, který provede testování obsahu XML dokumentu.
■ Element <xsl:if> se nachází uvnitř elementu <xsl:for-each>
Syntaxe
<xsl:if test="výraz">

. . . . . .
. . . výstupní data když výraz je správný
. . . . . .
</xsl:if>

Příklad.
<xsl:if test="náklad > 90000">
<tr>
<td><xsl:value-of select="autor"/></td>
<td><xsl:value-of select="název"/></td>
<td><xsl:value-of select="náklad"/></td>
</tr>
</xsl:if>

Element <xsl:choose>
■ Element <xsl:choose> se používá spolu s elementy <xsl:when> a <xsl:otherwise>, a má za cíl provést několik podmínkových testů.
Syntaxe
<xsl;choose>
<xsl:when test="výraz">
. . . . výstupní data
</xsl:when>
<xsl:otherwise>
. . . . výstupní data
</xsl:otherwise>
</xsl:choose>
Umístění
<xsl:choose>
<xsl:when test="náklad > 90000">
<td bgcolor="red"><xsl:value-of select="název"/></td>
</xsl:when>
<xsl:otherwise>
<td bgcolor="#999"><xsl:value-of select="název"/></td>
</xsl:otherwise>
</xsl:choose>

Element <xsl:apply-templates>
■ Element <xsl:apply-templates> aplikuje šablonu na tento element nebo na jeho potomky.
■ Když přidáme atribut select do elementu <xsl:apply-templates> šablona bude aplikována jenom na potomka, který odpovídá hodnotě atributu.
■ Můžeme použit atribut select, abychom určili pořadí zpracování potomků.

Příklad.
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>sbirka</h2>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>

```
<xsl:template match="kniha">
  <p>
   <xsl:apply-templates select="autor"/>
   <xsl:apply-templates select="název"/>
  </p>
 </xsl:template>

  <xsl:template match="autor">
  Autor: <span style="color:red">
  <xsl:value-of select="."/></span>
  <br/>
  </xsl:template>

 <xsl:template match="název">
  Autor: <span style="color:green">
  <xsl:value-of select="."/></span>
  <br/>
  </xsl:template>
</xsl:stylesheet>
```

XML Ostrůvky dat

■ IE dovoluje použít neoficiální značky <xml> pro vytvoření XML "Ostrůvky dat".

***Zadání pro Cvičení*:**

A. „XML a CSS". Pro soubor studia.xml zobrazit

1. Seznam předmetů (pouze názvy předmětů)
 - s odražkami
- číslovaný seznam
2. Předměty (1 rok, zimní semestr) v tabulce.
   - kb
   - vyučující
  - kod předmětu
      atd
3. Tabulku pro předmět MRL.
4. Pro poslední semestr
  - seznam předmětů
  - každý předmět má vlastní poznámku

B. Pro soubor studia.xml zobrazit (Použit XSLT a xPath)

1. Číslovaný seznam všech předmětů.
2. Tabulky pro každý semestr (pouze předměty s kb>2).
3. Tabulka pro poslední semestr, ve které předměty jsou uspořádány podle počtu kb. Pod tabulkou „Celkem kreditních bodů = ".
4. Seznam semestrů podle celkového počtu kb.
5. Seznam pro první semestr. Předměty, které vyučují různé katedry, musí mít různé pozadí (barva).

Úkoly pro samostatnou práci:

Prostudovat metody zobrazení XML souborů. Použit prezentace na vtan.ujep.cz (Složka IS, soubor L-xml-2-n.ppt).

Kapitola 5. HTML DOM a XML DOM

Cíl kapitoly:

- Vysvětlit podstatu DOM pro HTML a XML.
- Procvičit použití DOM pro různé úkoly.

Slovníček pojmů: DOM (document object model), hierarchie uzlů, strom uzlů, přístup k uzlům.

Výkladová část:
■ W3C DOM je platforma a jazykově neutrální rozhraní, které dovoluje programům a skriptům dynamicky přístupovat a modifikovat obsah, strukturu a styl dokumentu.
■ W3C DOM je roztříděný na tři různé části
- Core (jádro) DOM;
- XML DOM;
- HTML DOM.

■ XML DOM definuje standardní způsob pro přístup a manipulace XML dokumentů.

■ DOM vnímá XML dokument jako strom (stromová struktura). Všechny elementy, jejich obsah (text) a atributy mohou být přístupné skrz DOM strom. Obsah elementů může být modifikován nebo odstraněn, nové elementy mohou být vytvořené. Elementy, jejich text a jejich atributy jsou považovány za uzly.
■ Všechno v XML dokumentu je uzel.
- celý dokument je dokumentní uzel;
- každá XML značka je elementní uzel;
- text umístěný v elementu je textový uzel;
- každý atribut je atributní uzel;
- komentář je komentářní uzel.

Hierarchie uzlů
- Uzly mají hierarchický vztah navzájem.
- Každý element, atribut, text v XML dokumentu představuje uzel v stromě.
- Strom začíná z dokumentního uzlu a pokračuje po větvích, pokud nedosáhne všechny textové uzly na nejnižší úrovni.
- Termíny "rodič" a "potomek" se používají pro popis vztahu mezi uzly.
- Uzel, který nemá potomky je listový uzel.

Protože XMLdata jsou strukturovaná a představená ve formě grafu,
je možné prohlédnout data (pohybovat se po stromě) beze znalosti přesné struktury stromu a typu dat.

Strom uzlů
- vrcholný uzel se jmenuje kořenový uzel.
- každý uzel s výjimkou kořenového má přesně jednoho rodiče.
- uzel může mít libovolný počet potomků.
- list je uzel, který nemá potomky.
- sourozenci jsou uzly, které mají stejného rodiče

Přístup k uzlům
■ DOM umožňuje přístup ke každému uzlu v XML dokumentu.
Nalezení a přístup k uzlům:
- Pomocí metody getElementsByTagName( );
- Pomocí vlastnosti parentNode, firstNode a lastChild elementního uzlu.

getElementsByTagName( )
■ Metoda getElementsByTagName( ) může najít kterýkoli element v XML dokumentu.
■ Metoda pracuje bez ohledu na strukturu stromu (tj. bez ohledu na to, kde se uzel nachází).
■ Metoda vrací všechny elementy jako seznam uzlů.
■ Metoda může být použitá pro kterýkoli element.
Syntaxe:

```
getElementsByTagName("tagname");
```

Příklad.

xmlDoc.getElementsByTagName(“kniha”);

parentNode, firstChild a lastChild

Tyto vlastnosti dovolují pohybovat se po stromě na krátké vzdálenosti.

Kořenový uzel

Pomocí vlastnosti documentElement bude vrácen kořenový uzel

document.documentElement

Informace o uzlech

■ Vlastnosti nodeName, nodeValue a nodeType obsahují informace o uzlech.

1. nodeName

nodeName vlastnost obsahuje jméno uzlu

- nodename elementného uzlu je jméno značky (tag);
- nodename atributného uzlu je jméno atributu;
- nodename textového uzlu je vždycky # text;
- jméno dokumentního uzlu je vždycky # document.

2. nodeValue

Pro textové uzly vlastnost nodeValue obsahuje text;

Pro atributní uzly vlastnost nodeValue obsahuje hodnotu atributu;

Pro elementní a dokumentní uzly vlastnost nodeValue není k dispozici.

3. nodeType

Vlastnost nodeType vrací typ uzlu.

NodeList

■ Když používáme vlastnosti nebo metody jako childNodes nebo getElementsByTagName( ) obdržíme objekt NodeList.

■ Objekt NodeList představuje uspořádaný seznam uzlů. Uzly v seznamu mohou být přístupné přes jejich indexové číslo.

NamedNodeMap

■ Když použijeme vlastnost attributes pro nějaký element, obdržíme objekt NamedNodeMap.

■ Objekt NamedNodeMap reprezentuje neuspořádaný seznam atributních uzlů.

■ Uzly v seznamu NamedNodeMap mohou být přístupné přes jejich jména

Zadání pro Cvičení:

Pro soubory studia.xml , fakulta.xml a rodina.xml

- Vytvořit objektový model (strom uzlů) a popsat rodičovské vztahy mezi uzly.
- použit jazyk xPath pro procházení stromem a vyhledávání uzlů.

Úkoly pro samostatnou práci:

Prostudovat objektový model dokumentu a přistup k uzlům stromu. Použit prezentace na vtan.ujep.cz (Složka IS, soubor L-xml-2-n.ppt).

Kapitola 6. Javascript a HTML

Cíl kapitoly:

- Vysvětlit podstatu DOM pro HTML XML.
- Procvičit použití DOM pro různé úkoly.
- Procvičit práci Javascript s DOM HTML

Slovníček pojmů: DOM HTML, javascript, strom uzlů, přístup k uzlům.

Výkladová část:

HTML DOM

```
<html>
    <head>
      <title>Nazev</title>
    </head>
    <body>
       <h1>Zahlavi</h1>
       <p>Odstavec
            <a href="adresa">Odkaz</a>
        </p>
     </body>
</html>
```

Volání getElementsByTagName na dalších uzlech

```
var seznamy = document.getElementsByTagName("ul");
var druhyseznam = seznamy[1];
var polozkydruhehoseznamu = druhyseznam.getElementsByTagName("li");
```

Hledání všech elementů

```
document.all
document.getElementsByTagName(" * ");
```

Navigace po DOM Stromu

Vlastnosti:

- parentNode;
- childNodes;
- firstChild;
- lastChild;
- next Sibling;
- previousSibling;

Práce s Atributy

```
<p id="one">
var odstavec = document.getElementById("one");
var odstatribut = odstavec.getAttribute("id");
      nebo
var odstatribut = odstavec.id;
```

 Můžeme nastavit atribut

```
odstavec.setAttribute("id", "two");
```

funkce setAttribute může nejenom změnit existující atribut ale i přidat další atributy.

```
odstavec.setAttribute("nazev", "prvni_odstavec");
```

Změna stylů

 Každý elementní uzel má Vlastnost style.

Příklad:

```
odstavec.style.color = "red";
odstavec.style.backgroundColor ="yellow";
```

Vlastnosti CSS, které obsahují (-) musí být ve stylu Camel. Například:

text-indent musí být textIndent

Zadání pro cvičení:

HTML a JavaScript

1. Vytvořit nečíslovaný seznam. Položky seznamu jsou čísla. Při klepnutí na seznam, položka s maximální hodnotou se změní barvu (např. na červenou).
2. Vytvořit tabulku (5 řádků a 4 sloupci). Při klepnutí na tabulku první řádek bude zelený a druhý – žlutý.
3. Tabulka. Při klepnutí:
   - první řádek se nezmění;
   - další řádky budou mít různou barvu (sudé – modrou a liché – zelenou);
4. Tabulka. Obsah tabulky: čísla. Při klepnutí buňka, která obsahuje maximální číslo, bude mít žluté pozadí.

Úkoly pro samostatnou práci:
Prostudovat objektový model dokumentu HTML a přistup k uzlům stromu pomoci JavaScript. Použit prezentace na vtan.ujep.cz (Složka IS, soubor L-xml-2-n.ppt a soubor Lec-js-n1.ppt).

Kapitola 7. Javascript a XML DOM

Cíl kapitoly:

- Vysvětlit podstatu DOM pro XML soubory.
- Procvičit použití DOM pro různé soubory XML.
- Procvičit použití javascript pro přistup a práci s DOM XML

Slovníček pojmů: DOM XML, javascript, dokumentní uzel, elementní uzel, uzel atributu.

Výkladová část:

Typy uzlů:

- Document
- Element
- Text
- Attribute
- ProcessingInstruction
- Comment
- CDATASection
- Entity
- Notation

```
<?xml version="1.0" ?>
<!--  Nějaký komentář  -->
<student oc="F06375">
  <rok>
    . . . . . .
```

 Uzel Document je kořen uzlů hierarchie DOM.
 Uzel Document má vlastnost documentElement, která specifikuje uzel Element reprezentující kořenový element dokumentu XML (např. Student).

Vlastnost childNodes

 Protože item je výchozí hodnota objektu NodeList, můžete ji vynechat

```
DruhyUzel=Element.childNodes(1);
```

 Objekt NodeList má vlastnost length, která specifikuje počet uzlů obsažených v kolekci.

Získání znakových dat elementu

- Vlastnost text poskytuje text elementu, který je reprezentován daným uzlem a text obsažený ve všech jeho elementech potomků.
 Vlastnost nodeValue uzlu Element je vždy nastavena na null. Když element obsahuje znaková data, je tento text uložen v uzlu potomka Text a je možné získat tento text pomocí vlastnosti nodeValue.

XML: <nazev lang="en">Table</nazev>

Získání textu :

první postup x.text
druhý postup x.firstChild.nodeValue

Získání hodnoty atributu

Pro přistup k uzlem Attribute musíte použít vlastnost attributes
Element.attributes

 Metoda getAttribute(nazev_atributu)
např. Element.getAttribute("lang");

Příklad:
<kniha id="5756" lang="en" typ="sport"> ……. Uzel x

x.attributes -> vrací kolekci NamedNodeMap uzlů Attribute

NamedNodeMap = x.attributes

 Objekt NamedNodeMap má vlastnost getNamedItem(nazev_atributu)
Vlastnost getNamedItem vrací uzel, který má daný název

Konkrétní uzel Attribute má následující vlastnosti:
- nodeName (obsahuje název atributu)
- nodeValue (obsahuje hodnotu atributu)

Zadání pro Cvičení:
JS a XML DOM. Použit soubor studia.xml a vytvořit:
1. Seznam předmětů pro každý semestr (pouze předměty s kb>2).
2. Seznam předmětů pro semestr s nejvyšším celkovým počtem kreditů.
3. Seznam předmětů, který obsahuje jeden předmět z každého semestru. Vybraný předmět má nejvyšší počet kb. Zvýraznit pozadí předmětu, který má nevyšší počet kb ze všech předmětů seznamu, žlutou barvou.

Úkoly pro samostatnou práci:
Prostudovat objektový model dokumentu HTML a přistup k uzlům stromu pomoci JavaScript. Použit prezentace na vtan.ujep.cz (Složka IS, soubor L-xml-2-n.ppt a soubor Lec-js-n1.ppt).

Kapitola 8. Jazyk PHP

Cíl kapitoly:
- Prostudovat základní syntaxe jazyka PHP
- Procvičit vytvoření souborů PHP

Slovníček pojmů: řetězce, pole, funkce, výrazy, operátory.

Výkladová část:
<?php kód PHP ?>

```
  <?          kód PHP     ?>
  <%          kód PHP     %>
<script language="php"> kód PHP</script>

<?php
      příkaz 1;                              - volání funkce
      příkaz 2;              příkaz =        - početní operací
       . . . .                               - přiřazení hodnoty
      příkaz N;
  ?>
```

Komentář

3 druhy:

```
          /*  ----------------  */     zasahuje několik řádků
          //  ------------------        vše do konce řádku
          #  ------------------         vše do konce řádku
```

Proměnné

- $ - proměnné uvozeny tímto znakem
- za znakem $ následuje identifikátor proměnné
- v názvech proměnných jsou rozlišována malá a velká písmena

proměnné není třeba předem deklarovat či inicializovat

Konstanty

- funkce define()
- konstanta nemůže být změněná po její deklarace
- konstanta nepotřebuje znak $

Syntaxe

```
       define(„název konstanty“, „hodnota“, [nerozlíšení písmen])
```

- implicitně konstanty rozlišují malá a velká písmena
- existuje dohoda, že konstanty mají velká písmena

```
<?php
      define("PI","3.14",TRUE);
       print pi;
```

Pole

- jedno jméno -> řada hodnot
- k jednotlivým hodnotám přistupujeme pomocí indexu

Index v hranatých závorkách za jménem proměnné

```
          <?php  print $p[10];   ?>
```

Řetězce

Deklarace

```
      <?php  $retezec="Nějaký text.";  ?>
```

nebo

```
      <?php  $retezec=´Nějaký text.´; ?>
```

■ Použijete-li uvnitř řetězce v uvozovkách proměnnou, bude jméno proměnné nahrazeno hodnotou.

```
      <?php
          $prom="strom";
          print "hodnota je:$prom";
      ?>
```

Výpis textu

Výpis textu do kontextu stránky HTML !

```
       print (argument)
       echo (argument1, argument2, .. , argumentN)
```

- Rozdíl mezi příkazy *print* a *echo* spočívá v tom, že pomocí příkazu print lze vypsat jeden textový řetězec, zatímco příkazu echo můžete předat více textových řetězců (tj. Má více argumentů).

■ Příkazy nejsou funkce. Nemusíte argumenty uzavírat v závorkách.
Když příkaz echo má několik argumentů, závorky nepoužíváme.

```
<?php
   echo "Slovo", " ", "Dalsi text", " ", "Konec";
?>
```

■ Abychom posloupnost čísel (např. 1977) mohli brát jako řetězec, vkládáme ji do uvozovek nebo apostrofů.
■ Řetězec uzavřený do apostrofů není zpracováván.

Funkce pro práci s textovými řetězci

strlen() – pro zjištění délky

```
int  strlen (string  str)
```

strpos() – pro nalezení textu v řetězci

```
int  strpos (string kupka_sena, string jehla, [, int start])
```

Argument *start* je pro určení potici začátku hledání.

```
<?php
    $reteyec="Nějaký text .... konec.";
    $pozice=strpos($reteyec, "ryba");
     print "Hledané slovo začíná na pozici $pozice";
 ?>
```

substr() – pro výběr podřetězce z textového řetězce

```
string  substr(string řetězec, int start [, int délka])
```

Výrazy

- samotná hodnota
- samotná proměnná
- volání funkce
- kombinace výše uvedeného spojené operátorem

■ Libovolný výraz zároveň dává <u>hodnotu</u> odpovídající jeho obsahu.
<u>Operátory</u> umožňují spojovat hodnoty, konstanty, proměnné a volání funkcí do složitějších výrazů.

Operátory

- Matematické operátory:

```
+   -   *  /   %
```

Operátory porovnání:

```
==   !=   <>   <   >   <=   >=
```

operátor = (rovnítko) je operátorem přiřazení.

- Logické operátory (je možné převést libovolný výraz na hodnotu TRUE či FALSE

Jako logická nepravda (FALSE) je vyhodnoceno:

- číslo 0 (nebo 0.00)
- prázdný řetězec (" ")
- řetězec obsahující pouze znak 0
- prázdné pole
- prázdný objekt
- hodnota NULL

Vše ostatní je vyhodnoceno jako logická pravda (TRUE).

Negace

Negace je operátor, který z pravdy (true) dělá nepravdu (false).
Pro negace se používá znak ! Uváděný před hodnotou.

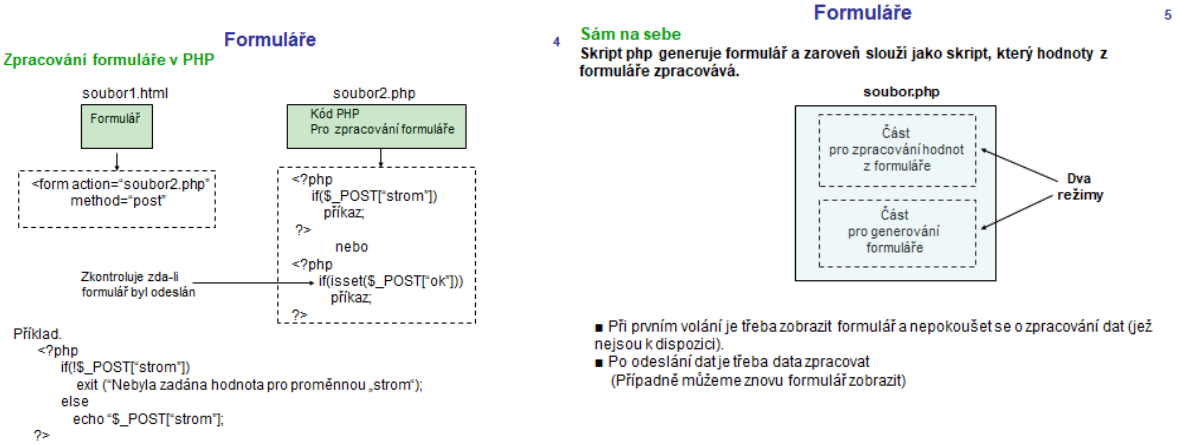
```
<?php
    $p=33;
    $np=!$p;  // $np je nyní 0
```

```
    $nnp=!$np;   //$nnp  je nyní 1
?>
```

■ negace nenulového čísla je 0
■ negace 0 je hodnota „true“ odpovídající číselné hodnotě 1

Konjunkce, Disjunkce, Exkluzivní disjunkce

- Aby dvě podmínky platily zároveň ----- > && nebo and (konjunkce)
- Když stáčí platnost jedné podmínky ze dvou ---→ || nebo or (disjunkce)
- Když platí pouze jedna podmínka ze dvou --→ xor (exkluzivní disjunkce)



Zadání pro Cvičení:

1. Kombinace PHP a HTML.
Použít tři různé způsoby pro vytvoření následujícího textu:

text
text
text

2. Práce s textovými řetězci
- Najít slovo „UJEP“ v textu (viz Soubor) a zobrazit pomocí stylů (jiná barva, barva pozadí, bliká atd.);
- Máte řetězec „Předmět: 2kb“. Vybrat „2“ a tisknout na stránce.

3. Vytvořit asociativní pole. Klíč=název předmětu, Hodnota=počet kb; Pro poslední rok.
- Výstup na stránce: klíč -> hodnota.
- Pomocí switch vypsat předměty s kb=2 a kb=3.

4. Tisknete pole, které vrátí funkce getdate(), v tvaru sloupce.

5. Vytvořte vícerozměrné pole (Pro semestr [zimní, letní] a předměty v těchto semestrech).

6. Vytvořte Funkce „Pozvánka“
- proměnné : $jmeno , $datum
- text pozvánky je libovolný, ale pozvánka se začíná „Vážený pane nebo paní + Jméno“ a končí „Datum“.

7. Vytvořte Funkce „Pozvánka“
- proměnné : $jmeno , $datum
- text pozvánky je libovolný, ale pozvánka se začíná „Vážený p.Jméno“ a končí „Datum“.

8. Vytvořte formulář pro pozvání.
Políčka formuláře:

- pro jméno
- pro pohlaví
- pro titul

Po zadávání údajů skript musí vrátit Text pozvánky. Např.

Vážený/ná Mgr. Novák/ová
Text pozvánky
Datum

9. Vytvořte Formulář-kviz.

Vytvořte políčka pro přihlášku účastníka. (jméno).

Pro každou otázku přepravte 3 odpovědí (ve formě – radio). Určete body pro každou otázku. Určete celkový počet bodů, který je nutný pro postup do dalšího kola. Při splnění podmínky, účastník soutěží postupuje dál (dostane oznámení). V opačném případě dostane oznámení, že musí kviz opakovat.

10. Zpracujte formuláři na Vaše Web stránce.

11. Napsat php soubor (formulář) pro odeslání souboru na server. Odeslat soubor na server.
Při zpracování souboru na serveru:
- vytvořit tabulku s následujícími sloupci:
• název souboru;
• typ;
• size;
• cesta k adresáři, ve kterém je uložen.

Název tabulky je „Údaje o odeslaném souboru“.

12. Odeslat obrázek na server a zobrazit na web stránce.

Úkoly pro samostatnou práci:
Prostudovat základní syntaxe PHP. Použit prezentace na vtan.ujep.cz (Složka IS, soubory L1-syntaxe.ppt ,
L2-struktury.ppt a L3-formulare.ppt).

Kapitola 9. PHP (ukládání dat, Cookie a session)

Cíl kapitoly:

- Prostudovat funkce pro otevření souborů, čtení ze souboru, zápis do souboru, odesílání souborů.
- Procvičit ukládání dat, práci s Cookie a vytvoření sezení (session)

Slovníček pojmů: cookie, session (sezení).

Výkladová část:

Ukládání dat z webového formuláře

```
<html>
  <form action="soub1.php" method="post">
        <?php
            for($i=1; $i<=5; $i++):
              print "$ičlen:";
         ?>
    <input type="text" name="jmeno[ ]"/><br/>
        <?php  endfor;  ?>
    <input type="submit" name="ok" value="Odeslat"/>
  </form>
</html>
```

Kód pro zpracování údajů:

```
<?php
     $jmeno=$_POST[“jmeno”];
        for($i=0; $<5; ++$i)
         print ”$i.clen:”.$jmeno[$i].”<br/>”;
?>
```

Vkládání souborů s kódem (v jazyce PHP)

■ Ke vkládání kódu umístěného v jiném souboru se používají příkazy:

include() require()

```
<?php
   $cislo=7;
   $total=include(“soubor.inc”);
    print “Vložený kód vrati $total”;
?>
```

soubor.inc

```
  <?php
     $hodnota=($cislo+10);
      return $hodnota;
   ?>
```

• Nelze soubor „soubor.inc“ spustit samostatně, protože z nadřazeného kódu získává hodnotu proměnné $cislo.

Otevření souboru

■ PHP pracuje se soubory na serveru prostřednictvím *deskriptorů* (token, který umožňuje získat přístup k identifikovanému prostředku).

■ *Deskriptor* získáme při otevření souboru.

int fopen (string název_souboru, string režim)

Touto funkcí lze otevřít soubor na místním serveru, případně prostřednictvím protokolů HTTP nebo FTP rovněž soubory na vzdáleném serveru.

■ Argumentem režim určujeme režim práce se souborem.

- *číst (r)* - *zapisovat (w)* - *připojit (a)*

Znak „+” označuje požadavek na rozšířené možnosti;
Písmeno „b“ – označuje práce v binárním režimu.
Příklad: “w+b”.

Funkce pro zavření souboru

bool fclose(int deskriptor)

```
<?php
    $soubor=fopen(“soub.txt”,”w+”);
      if($soubor)
          print “soubor byl otevřen.”;
      else
         die(“Soubor se nepodařilo otevřit.”);
    fclose($soubor);                    ?>
```

Čtení ze souboru

■ Údaje jsou v souborech (používaných PHP aplikacemi) většinou ukládány v podobě textových řetězců.

■ Základem získávání těchto údajů je čtení řádků a přesunutí ukazatele na příslušný řádek.

string fread (int deskriptor, int délka)
string fgets (int deskriptor, int délka)

Funkce *fread()* přečte určitou délku bajtů ze souboru. Čtení končí, pokud jsou všechny určité bajty přečteny nebo je dosažen konec souboru.
Funkce *feof()* zjistí, je-li dosažen konec souboru

bool feof (int deskriptor)

Načítání víceřádkového souboru

```
<?php
    $soubor=fopen("some.txt","r");
      if($soubor)
        print "Soubor byl úspěšně otevřen.<br/>";
      else
         die("Soubor se nepodařilo otevřit.");
     $text=fread($soubor,100);
       echo $text;
     fclose($soubor);
?>
```

Čtení textového souboru po řádcích

■ Použitím funkcí ***fgets()*** a ***feof()*** načteme celý textový soubor v cyklu po jednotlivých řádcích.

```
<?php
      $soubor=fopen("prvni.txt","r") or die("Nok");
      while(!feof($soubor))
          {
            $radek=fgets($soubor,500);
             print "$radek<br/>";
          }
       fclose($soubor);
 ?>
```

Zápis do souborů

■ Jazyk PHP můžeme využít k odesílání textových řetězců do textových souborů (tj. ukládat data do textových souborů).

int fwrite (int deskriptor, string řetězec [, int délka])
int fputs (int deskriptor, string řetězec [, int délka])

■ Funkce ***fwrite()*** zapíše obsah řetězce do souboru specifikovaného argumentem deskriptor.
Je-li použit argument délka, zápis skončí po zapsání (délka) bajtů.
■ Funkce ***fputs()*** je pouze aliasem funkce fwrite().

```
<?php
   $nazev="prvni.txt";
     print "Přídání textu do souboru prvni.txt<br/>";
    $text="Mistní lidé začali šetřit už i na jidle.";
    $soubor=fopen($nazev,"w") or die("Nok");
     fwrite($soubor,"$text");
     fclose($soubor);
?>
```

Odesílání souborů na server

■ Informace o souboru přijímaném na serveru najdeme v poli $_FILES.
■ Jednotlivé prvky pole $_FILES
$_FILES['userfile']['name'] –originální název souboru na klientském počítači
$_FILES['userfile']['type'] – typ MIME souboru, pokud prohlížeč tuto informaci poskytuje
$_FILES['userfile']['size'] – velikost přenášeného souboru v bajtech
$_FILES['userfile']['tmp_name'] – dočasný název souboru, pod nímž byl přenášený soubor uložen na server
■ Soubor byl umístěn v dočasném adresáři, odkud byl po dokončení skriptu vymazán.

Abychom mohli s přeneseným souborem dále pracovat, musíme ho v rámci skriptu PHP přemístit z dočasného adresáře do cílového umístění na serveru.

bool move_uploaded_file(string název_souboru, string cíl)

Oba argumenty funkce jsou tvořeny informacemi získanými z pole $_FILES

```
$nazev_souboru=$_FILES[‘fupload’][‘tmp_name’];
$cil=“. /Soubory_klienta/”.$_FILES[‘fupload’][‘name’];
```

Odesílání souborů na server

```
<html>
 <head><title>Odesilani souboru na server</title></head>
 <body>
   <form  enctype=“multipart/form-data”
          action=“some.php”
          method=“POST”>
    <input type=“file” name=“upload” style=“width:500px;”/><br/>
    <input type=“submit” name=“ok” value=“Prenos souboru”/>
  </form>
 </body>
</html>
```

Cookies

■ HTTP je bezestavový protokol.
■ Cookies jsou jedním z rozšíření, které tuto situaci řeší.
■ Cookie je krátký textový řetězec, jež si mezi sebou vyměňují web server a klient v rámci *hlavičky* HTTP.

Postup

1) Server vytvoří Cookie a zašle ji klientovi, který ji u sebe uloží.
2) Když klient příští bude přistupovat na tentýž server, klient tento Cookie opět zašle v rámci požadavku web serveru.

V Cookie je možné uchovávat různou informace:
- počet návštěv;
- identifikátor pro poslední prohlíženou stránku;
- identifikaci klienta, apod.

■ Cookies jsou uchovávány v běžných textových řetězcích (souborech) na klientských počítačích.
-- Cookies jsou nezakódované –

Práce s Cookies v PHP

1. <u>Odesílání Cookie klientovi</u>

• Chcete-li klientovi poslat nějakou informaci prostřednictvím Cookies, použijte funkci SetCookie().

SetCookie(název, hodnota, platnost)

- jediným povinným argumentem je název (tj. jméno, pod kterým Cookie je k dispozici).
❑ zadáte-li pouze název, je daná Cookie z počítače klienta smazána.
❑ nezadáte-li dobu platnosti, je daná Cookie platná pouze pro dané sezení.
❑ doba platnosti je zadávána ve standardním unixovém formátu (tj. ve vteřinách od roku 1970).

Můžeme použít funkce time(), jež vrací aktuální čas.

Příklady funkce SetCookie().

```
setCookie(“Pokus”, “hodnota”) // platná pro sezení.
setCookie(“Pokus”) // smazání hodnoty.
setCookie(“Poslední_navštívená_stránka”, $PHP_SELF, time()+60*60*24) // platná 1 den.
setCookie(“Počet”, $Pocet_navstev, time()+60*60*24*7) // platná 1 týden.
```

■ Cookies jsou odesílány v hlavičce HTTP. To znamená, že je třeba funkci SetCookie() volat ještě před tím, než skript vypíše jakýkoli text na výstup.

2. <u>Zpracování příchozí Cookie (z požadavku klienta)</u>

■ Pro přístup k proměnným došlým prostřednictvím Cookies slouží globální asociativní pole COOKIE

```
<?php
    if($_COOKIE[“Nazev”]
        příkaz;
?>
```

Nevýhody Cookies:

■ Každý web prohlížeč umožňuje klientovi rozhodnout, zda Cookies zasílané serverem automaticky přijímat, odmítat, nebo se dotazovat.

■ Cookies nejsou vhodné pro posílání většího množství dat.

■ Cookies nejsou příliš bezpečné.

⍓ Mechanizmus ***session proměnných*** umožňuje využit alternativních prostředků.

Podstata.

- k uživateli a (zpět) putuje pouze nějaký náhodný řetězec či velké číslo sloužící jako identifikace uživatele.
- všechna data, spojená s prací uživatele na daném webu, např.
  počet návštěv,
  poslední navštívená stránka,
  obsah nákupního košíku,
  uživatelské jméno apod.

jsou uložena na samotném serveru (zpravidla v DB).

Postup

1) Při každém přístupu na server je zjišťováno, zda již pro vás existuje platná session.
2) Pokud neexistuje, je vytvořena.
3) Pokud existuje, pozná to server bud´
   - podle zaslané Cookie s *identifikací* vašeho sezení;
   - nebo pomocí parametru předaného v URL.

Identifikace sezení: je to obvykle dlouhé jedinečné číslo – aby nemohlo dojít k záměně sessions mezi dvěma uživateli.

4) Na serveru (obvykle v DB) ukládána všechna data, spojená s jednotlivými sezeními – tj. vlastní proměnné, které můžeme potřebovat.

Práce s session proměnné

■ Ve všech stránkách, kde chcete sessions používat, je třeba volat funkci ***session_start().***

Funkce session_start()

Funkce se pokusí od uživatele převzít identifikaci existujícího sezení:
- v případě úspěchu najde a zpřístupní data uložená pro toto sezení;
- v případě neúspěchu založí nové sezení a odešle uživateli jeho identifikaci.

■ Pro celý web je třeba použít direktivu v php.ini

```
session.auto_start=1
```

v tom případě není třeba funkce explicitně volat v kódu.

■ V současné době se preferuje metoda, při které se prostě proměnné sezení vytvářejí jako jakékoli jiné proměnné. Jediný rozdíl je v tom, že se na ně musíte odkazovat v kontextu superglobální proměnné **$_SESSION**.

Příklad. Chceme nastavit proměnnou sezení s názvem *username*.

```
<?php
   session_start();
   $_SESSION[“username”]=“Novak”;
    echo “Vaše uživatelské jméno je”.$_SESSION[‘username’];
?>
```

■ Pro odstranění proměnné sezení používáme funkci unset().

např. unset($_SESSION[‘username’]);

Ukládání informace o sezení

■ direktiva ***session.save_handler(files,mm,sqlite,user)*** v php.ini určuje, jak se budou ukládat informace o sezení.

- do souborů (files);
- do sdílené paměti (mm);
- do databáze SQLite (sqlite);
- prostřednictvím uživatelsky definovaných funkcí (user).

■ Když ukládáme do souborů, musíme v php.ini nastavit hodnotu direktivy **session.save_path** na existující složku.

■ Volba (user) je nejkomplikovanější. Nicméně, je nejflexibilnější a nejvyspělejší, protože se dají vytvořit vlastní zpracovatelé, aby ukládali informace na jakákoli média, kam si vývojář přeje (např. do MySQL).

Příklad. Ověření uživatele.

1. Vytváření tabulky v DB
2. Vytváření přihlašovacího formuláře
3. Ověření uživatele pomocí proměnné sezení.

Zadání pro Cvičení:

1. Vytvořit na serveru vlastní složku. Odeslat soubor na server a umístit jej do vytvořené složky.
2. Přečíst 650 bajtů z tohoto souboru a zobrazit ve formátovaném bloku s rámečkem.
3. Připsat do souboru předem přepravený text. Vypsat výsledný soubor do web stránky.
4. Připsat dva řádky textu na začátek souboru a jeden řádek na konec souboru. Vypsat výsledný soubor do web stránky.
5. Uložit ve své složce několik souborů pomocí formuláře
   - zobrazit seznam souborů a podadresářů, které jsou ve vaše složce;
   - vytvořit s názvu souboru odkaz s tím abychom mohli otevřít tento soubor.
6. Nahrát obrázek na web stránku. Dále nahrát další obrázek, který nahradí předchozí obrázek.

Cookie a sezení

1. Vytvořit web stránku. Použít Cookie pro počet návštěv.

Při první návštěvě napište „Vítáme na Naše stránce“
Při 5 návštěvě zobrazte pozdrav
Při 10 návštěvě napište nějaký text (např. Nemůžeš si najít lepší web?)

2. Vytvořit web s 4 stránky. Na úvodní stránce umístěte seznam článků. Na dalších stránkách umístěte názvy jednotlivých článků (jeden článek na každé stránce).

Při návštěvě webu pročíst druhý a třetí články. Vypnout prohlížeč a pak opět navštivte web. Musí být zobrazen seznam článků, které jste již prohledal minulé.

3. Vytvořit web s následujícími stránkami.

Obsah úvodní stránky (stránka 1):

Obecné otázky

1. Otázka 1
   - odpověď 1
   - odpověď 1
   - odpověď 1
2. Otázka 2
   - odpověď 1
   - odpověď 1
   - odpověď 1
3. Otázka 3
   - odpověď 1
   - odpověď 1

 odpověď 1
Tlačítko „Odeslat“

Obsah stránky 2:
Neuspěl jste
n e b o
Vyberte téma pro další otázky
Seznam témat:
- Téma 1
- Téma 2

Obsah stránky 3:
Téma 1
1. Otázka 1
 odpověď 1
 odpověď 1
 odpověď 1
2. Otázka 2
 odpověď 1
 odpověď 1
 odpověď 1
3. Otázka 3
 odpověď 1
 odpověď 1
 odpověď 1
Tlačítko „Odeslat“

Obsah stránky 4:
Téma 2
1. Otázka 1
 odpověď 1
 odpověď 1
 odpověď 1
2. Otázka 2
 odpověď 1
 odpověď 1
 odpověď 1
3. Otázka 3
 odpověď 1
 odpověď 1
 odpověď 1
Tlačítko „Odeslat“

Obsah stránky 5:
Dostal jste N (např. 8 bodů) bodů.
Váša známka je X (např. „1“)

Úkoly pro samostatnou práci:
Prostudovat metody ukládání dat, použití cookies a vytvoření sezení. Použit prezentace na vtan.ujep.cz (Složka IS, soubory L4-ukladani-dat.ppt a soubor L-CookieSession.ppt).

Kapitola 10. PHP (grafika)

Cíl kapitoly:
- Prostudovat funkce grafické knihovny a jejich použití.

- Procvičit použití grafické knihovny v PHP

Slovníček pojmů: cookie, session (sezení).

Výkladová část:
Ke generování dvojrozměrné grafiky se (občas) používá ***tabulka HTML***, v níž vytváříme grafy „natažením“ vodorovného rozměru (šířky) barevných čtverečků.
■ PHP řeší podporu grafiky pomocí grafické ***knihovny GD***.
Grafy HTML
- Graf je z hlediska zobrazení na stránce HTML vlastně tabulkou.

Podstata.
● Vytvoříme obrázek o rozměrech 10x10px v požadované barvě a pojmenujeme ho (např. obr.bmp).
● Základem grafu je tabulka, která bude obsahovat dva sloupce a tolik řádků, kolik hodnot chceme v grafu zobrazit.
- v prvním sloupce bude popis sledovaného parametru;
- ve druhém sloupci zobrazíme náš obrázek zvětšený na požadovanou velikost.

Postup.
1. Vytvoříme statický prototyp stránky HTML, na níž vykreslíme provzor grafu.
Kód v prvním sloupci je název (popis) parametru. Ve druhém sloupci použijeme obrázek o rozměrech 10x10px (obr.bmp).
2. Zvětším obrázek na 90, 150 a 60 pixelů. (pro 1% prezentované hodnoty vyhradíme 3 pixely). Když hodnoty jsou možnosti výběru – max = 300px.
3. Musíme provést matematický výpočet pro zajištění rozumných proporcí grafu:
Chceme-li zobrazit max hodnotu jako obdélník široký 300px vypočítáme šířky ostatních sloupců podle vzorce

šířka=interval(hodnota/maximum)*300

Dinamický graf (soubor gr1.php)

```
<html>
  <head><title>Dynamický graf</title></head>
    <body>
      <?php
$nazvy=array(1=>”Artist1”,“Artist2”,“Artist3”); //Hodnoty, které slouží jako podklad pro
$hodnoty=array(1=>560,920,135);                  // tvorby grafu
  // Nalezeni maxima
  $max=max($hodnota);
   echo ‘<table border=“0”><tbody>’;
  //Vypocet a zobrazeni grafu
 foreach($hodnoty as $index=>$hodnota)
    {
      echo “<tr><td>$nazvy[$index]</td>”;
      $sirka=intval(($hodnota/$max)*300)+5;
      echo “<td><img src=‘obr.bmp’ height=’15’ width=“ .$sirka. ”>” . $hodnota. “</td></tr>”;
    }
  echo “</tbody></table><br/>”;
?>
</body>
</html>
```

Grafická knihovna GD
■ Knihovna GD umožňuje vykreslování geometrických obrazců včetně barevných výplní.
■ Nejnovější verze knihovny GD generují výstup ve formátu ***png***. Domovská stránka knihovny:
http://www.boutell.com/gd
■ v souboru **php.ini** je nutné odstranit středník na počátku řádku s textem:
extension=php_gd.dll

❒ Získáme informace o GD pomocí funkce ***gd_info()***

```
<?php
      print "<pre>";
      print_r(gd_info());
      print "</pre>;
?>
```

Vložení obrázku generovaného pomocí kódu PHP do web stránky

Vložení statického obrázku

```
<html>
  <body>
     <img src="obr.bmp" alt="Obrazek"/>
  </body>
</html>
```

Generování obrázku pomocí PHP

■ Úlohou grafické knihovny GD je na základě předpisu v jazyce PHP generovat obrázek v požadovaném formátu a vložit ho do kontextu výsledné webové stránky.

Funkce:

```
      resource  imagecreate (int x, int y)
      resource  imagecreatetruecolor (int x, int y)
```

Funkce vrací ***identifikátor*** prázdného obrázku o rozměrech x a y.

Funkce:

```
     int  imagecolorallocate (resource obrázek, int červená, int zelená, int modrá)
```

vrací identifikátor barvy pro obrázek vytvořený funkcí imagecreate().

Funkce:

```
          int  imagepng (resource obrázek, [, string  název_souboru])
```

odešle obrázek generovaný pomocí knihovny GD jako datový proud.

Funkce:

```
          int  imagedestroy (resource  obrázek)
```

uvolní místo v paměti vyhrazené pro manipulaci s obrázkem.

Příklad

```
<?php
  header("Content-type: image/png");
  $obr= imagecreate(200,200);
  $green=imagecolorallocate($obr, 0, 255, 0);
  imagepng($obr);
?>
```

■ Kdybychom se pokusili přidat nadpis do obrázku, bude se jednat o pokus o vložení binární podoby do kontextu stránky HTML

☝ Jedním z možných postupů pro dynamické vkládání generovaných obrázků do stránek HTML je klasický element HTML img, v němž jako název souboru s obrázkem uvedeme název souboru obsahující kód v jazyce PHP.

Vložení obrázku generovaného pomocí kódu PHP do web stránky

```
<html>
  <body>
    <h1>Obrazek</h1>
      <img src="obr.php" alt="Obrazek" align="center"/>
  </body>
</html>
```

Kreslení základních geometrických obrazců

■ Jakýkoli obrázek vytvořený v jazyce PHP vzniká převážně jako kompozice složená z jednoduchých geometrických obrazců.

Funkce pro vykreslení jednoduchých geometrických obrazců

```
int  imageline (resource obrazek, int x1, int y1, int x2, int y2, int barva)
```

int imagerectangle (resource obrazek, int x1, int y1, int x2, int y2, int barva)
int imagefilledrectangle (resource obrazek, int x1, int y1, int x2, int y2, int barva)
int imagearc (resource obrazek, int cx, int cy, int w, int h, int s, int e, int barva)
- identifikátor obrazku;
- souřadnice;
- barva.

Funkce:

int imagesetstyle (resource obrázek, array styl)

Definuje různé vzory výplně (např. jemnou mozaiku).

Kreslení základních geometrických obrazců

■ Pro výplň obrazce barvou slouží funkce:

int imagefilltoborder (resource obrázek, int x, int y, int ohraničení, int barva)

Po jejím užití se určená barva „rozlije“ až k hranicím vymezeným ***argumenty x a y*** nebo plochami vykreslenými barvou určenou v ***argumentu ohraničení***.

Příklady kreslení:

```
imageline ($obr, 0, 0, 200, 200, $green);        // přimka
imagerectangle ($obr, 0, 0, 250, 250, $blue);   // čtyřúhelník

$styl=array($red, $blue);
imagesetstyle($obr, $styl);
imagefilledrectangle($obr, 0, 0, 100, 100, IMG_COLOR_STYLED);
imagearc ($obr, 300, 100, 200, 0, 360, $black);   // elipsa
```

Kreslení textu

■ Gragická knihovna GD má integrovány rovněž funkce pro vykreslování textu přímo do generovaného obrázku.

int imagestring (resource obrázek, int písmo, int x, int y, string s, int sloupec)
int imagestringup (resource obrázek, int písmo, int x, int y, string s, int sloupec)

Funkce ***imagestring()*** vykreslí text vodorovně, funkce ***imagestringup()*** zdola nahoru.

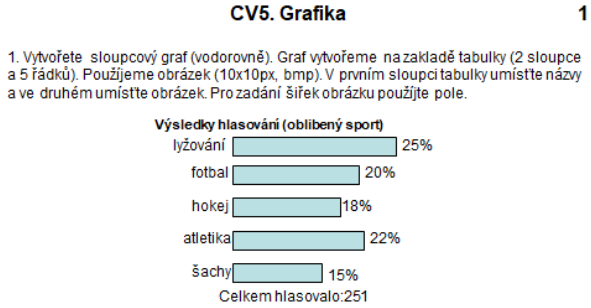
Kreslení textu

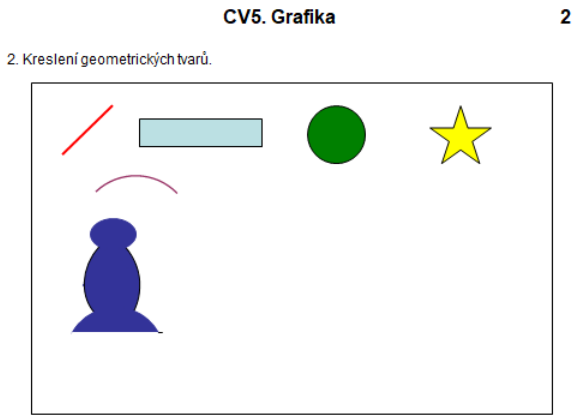
Příklad.

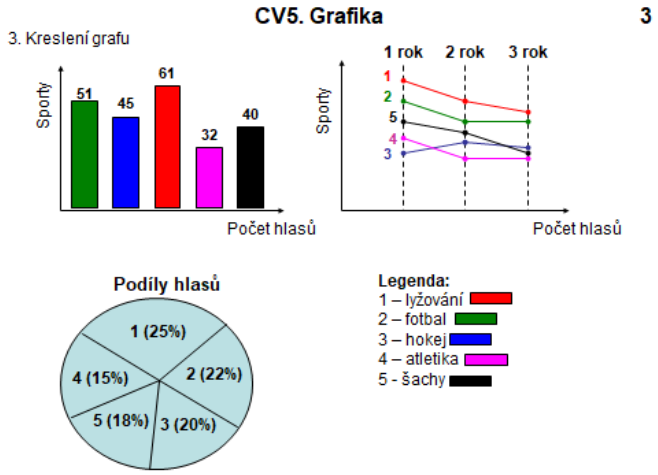
```
<?php
  header(“Content-type:image/png”);
  $obr=imagecreate(600, 300);
  $black=imagecolorallocate($obr, 0, 0, 0);
  $red=imagecolorallocate($obr, 255, 0, 0);
  $blue=imagecolorallocate($obr, 0, 0, 255);
     for ($i=1; $i<=5; $i++)
       {
         imagestring ($obr, $i, 100, 10*$i, ‘Text’, $red);
       }
     for ($i=1; $i<=5; $i++)
       {
         imagestringup ($obr, $i, 10*$i, 250, ‘Text’, $black);
       }
  imagepng($obr);
?>
```

■ Argument písmo určuje druh písma, jenž bude použit k vykreslení textu. Hodnoty 1, 2, 3, 4 a 5 jsou vyhrazeny pro písma zabudovaná do knihovny GD.

Zadání pro Cvičení:







Úkoly pro samostatnou práci:
Prostudovat metody kreslení pomocí PHP. Použit prezentace na vtan.ujep.cz (Složka IS, soubor Lec-Grafika-n.ppt).

Kapitola 11. PHP (práce s objekty)

Cíl kapitoly:

- Prostudovat objektové programování v PHP
- Procvičit vytvoření tříd a objektů v PHP a jejich použití

Slovníček pojmů: třída, objekt,

Výkladová část:
Třída a Objekt
Vlastnosti (členské proměnné) jsou interní proměnné dané třídy.

- ❑ Existují Vlastnosti společné pro všechny Objekty dané třídy (tzn. Statické vlastnosti)
- ❑ Vlastnosti se liší řízením přístupu (viditelnost).

Modifikátory přístupu:
- public (vlastnost viditelná odkudkoli)
- protected (vlastnost je k dispozici pouze v dané třídě a v třídách odvozených)
- private (vlastnost je k dispozici pouze ve třídě, v niž je definována)

Příklad:

```
<?php
  class Pokus1 {
```

```
                    public $prom1;  // k dispozici odkudkoli
                    protected $prom2; // k dispozici v dané třídě nebo v odvozených třídách
                    private $prom3; // k dispozici pouze ve této třídě
                     }
        ?>
```

Metody (členské funkce) jsou funkce, jež manipulují s daty objektu (instance). Funkce jsou svázány s daty uvnitř definice třídy.

Spojení dat i metod pro manipulaci s nimi -> **zapouzdření**

- Třídy mohou mít Statické metody (tj. metody, které je možné volat bez vytvoření instance objektu)

Modifikátor ***static*** pro deklarace statické metody.

Vytváření instancí

Instance Třídy = Objekt

Objekty jsou vytvářeny pomocí operátora *new* a určení jejího jména

Příklad:

```
<?php
  $můj_nový_objekt=new Nějaká_Třída();
?>
```

Proměnná můj_nový_objekt reprezentuje Objekt se všemi jeho Vlastnosti a Metodami.

Práce s vlastnostmi objektu

- Přístup k jednotlivým vlastnostem objektu -> prostřednictvím identifikátoru objektu.

Odkaz na Jméno vlastnosti třídy --→ vždy přes jméno objektu !! (Např. $Ob1->lom)

Příklad:

```
<?php
  $nazevObjektu->nazevVlastnosti="hodnota";
?>
```

Deklarace metod

- Deklarace metod obvykle následuje za deklarací vlastností.
- Metody stejně jako vlastnosti odkazované prostřednictvím identifikátoru třídy.

Příklad:

```
<?php
   class MyClass {
      public $částka;
      public $stav;
       .  .  .  .  .  .
     public function výběr($částka) {
        $this->stav - =$částka;
      }
  ?>
```

Slovo this: Při každém přístupu k vlastnosti třídy je třeba zadávat jméno příslušného objektu. V definice třídy jméno objektu, který bude vytvořen, není známo. Pro přístup ke všem vlastnostem (proměnným) slouží identifikátor this.

Identifikátor this odkazuje na aktuální instanci třídy (tj. na Objekt).

Konstruktor

Konstruktor – je speciální metoda třídy, která slouží k její inicializaci.

Konstruktor se jmenuje __construct().

Obvykle není explicitně volán. V PHP operátor *new* slouží pro vyvolání konstruktoru.

```
// deklarace konstruktoru
public function __construct($par1=3 , $par2="text", $par3=true)
  {
```

```
    $this->par1=$par1;
    $this->par2=$par2;
   $this->par3=$par3;
  }
```

Tento zápis umožňuje ruzné volání:

1. $MyObjekt= new Myclass(); // volání bez parametrů, budou použity všechny standardní hodnoty (tj. $par1=3 , $par2="text", $par3=true)
2. $MyObjekt= new Myclass(5); // vlastnost $par1 bude iniciálizována hodnotou 5
3. $MyObjekt= new Myclass(23, "nový_text"); // vlastnost $par1 bude iniciálizována hodnotou 23, a vlastnost $par2 hodnotou "nový_text"
4. $MyObjekt= new Myclass(44, "další_text", false); // nastaví všechny vlastnosti podle zadaných hodnot

Dědičnost

Dědičnost – metoda, která umožňuje od jedné třídy odvozovat třídu novou (potomka).

- Nová Třída dědí všechny vlastnosti a metody svého předka. Navíc může je upravit.

Definice: klíčové slovo *extends*.

```
<?php
    class NovaTrida extends PuvTrida {
          . …….
        }
 ?>
```

- Nová třída *NovaTrida* může přidat nové vlastnosti a přepsat (nahradit novou verzí) některé z metod původní třídy *PuvTrida.*

Zadání pro Cvičení:

1. Vytvořit třídu „Student" a uložit do složky: "Třídy".
   Ve třídě „Student" použit vlastnosti:

- oc;
- příjmení;
- ročník;
- počet kb v stávajícím semestru.

Použit metody pro:

- změnu ročníku;
- přidávání kreditů;
- zobrazení všech udajů pomocí tabulky.

---

2. Vytvořit třídu „Studenti" (např. pro 3 studenty). Vytvořit metody:
   - SetRocnik;
   - AddKB;
   - ZobrazTabulku.

3. Rozšířit třídu „Studenti" abychom mohli přidávat další studenty.

Úkoly pro samostatnou práci:

Prostudovat vytvoření tříd a objektů v PHP. Použit prezentace na vtan.ujep.cz (Složka IS, soubor OOP-nn.ppt).

Kapitola 12. PHP a XML DOM

Cíl kapitoly:

- Vysvětlit použití DOM pro XML soubory

- Procvičit zpracování souborů XML (DOM) pomocí PHP

Slovníček pojmů: DOM, PHP, strom uzlů, přistup k uzlům, kořenový uzel.

Výkladová část:

W3C DOM je platforma a jazykově neutrální rozhraní, které dovoluje programům a skriptům dynamicky přistupovat a modifikovat obsah, strukturu a styl dokumentu.

■ W3C DOM je roztříděný na tři různé části
- Core (jádro) DOM;
- XML DOM;
- HTML DOM.

■ XML DOM definuje standardní způsob pro přístup a manipulace XML dokumentů.

■ DOM vnímá XML dokument jako strom (stromová struktura). Všechny elementy, jejich obsah (text) a atributy mohou být přístupné skrz DOM strom. Obsah elementů může být modifikován nebo odstraněn, nové elementy mohou být vytvořené. Elementy, jejich text a jejich atributy jsou považovány za uzly.

■ Všechno v XML dokumentu je uzel.
- celý dokument je dokumentní uzel;
- každá XML značka je elementní uzel;
- text umístěný v elementu je textový uzel;
- každý atribut je atributní uzel;
- komentář je komentářní uzel.

Hierarchie uzlů

- Uzly mají hierarchický vztah navzájem.
- Každý element, atribut, text v XML dokumentu představuje uzel v stromě.
- Strom začíná z dokumentního uzlu a pokračuje po větvích, pokud nedosáhne všechny textové uzly na nejnižší úrovni.
- Termíny "rodič" a "potomek" se používají pro popis vztahu mezi uzly.
- Uzel, který nemá potomky je listový uzel.

Protože XML data jsou strukturovaná a představená ve formě grafu,
je možné prohlédnout data (pohybovat se po stromě) bez znalosti přesné struktury stromu a typu dat.

Strom uzlů

- vrcholný uzel se jmenuje kořenový uzel.
- každý uzel s výjimkou kořenového má přesně jednoho rodiče.
- uzel může mít libovolný počet potomků.
- list je uzel, který nemá potomky.
- sourozenci jsou uzly, které mají stejného rodiče

Přístup k uzlům

■ DOM umožňuje přístup ke každému uzlu v XML dokumentu.

Nalezení a přístup k uzlům:
- Pomocí metody getElementsByTagName( );
- Pomocí vlastnosti parentNode, firstNode a lastChild elementního uzlu.

getElementsByTagName( )

■ Metoda getElementsByTagName( ) může najít kterýkoli element v XML dokumentu.
■ Metoda pracuje bez ohledu na strukturu stromu (tj. bez ohledu na to, kde se uzel nachází).
■ Metoda vrací všechny elementy jako seznam uzlů.
■ Metoda může být použitá pro kterýkoli element.

Syntaxe:

```
getElementsByTagName("tagname");
```

Příklad.
xmlDoc.getElementsByTagName(“kniha”);

Seznam uzlů
■ Seznam uzlů obvykle bude uložen do proměnné (např. „ x “)
var x=xmlDoc.getElementsByTagName(“kniha”);

Nyní proměnná x obsahuje seznam všech elementů <kniha>, které jsou v XML dokumentu. Pomocí indexového čísla máme přístup k jednotlivým elementům (např. var y=x[3]).

■ Vlastnost length umožňuje prohlédnout (přístup) každý element v seznamu.

```
var x=xmlDoc.getElementsByTagName(“kniha”);
for ( var i=0; i<x.length; i++)
 {
   // nějaké operace s elementy <kniha>
 }
```

parentNode, firstChild a lastChild
Tyto vlastnosti dovolují pohybovat se po stromě na krátké vzdálenosti.

Kořenový uzel
Pomocí vlastnosti documentElement bude vrácen kořenový uzel
document.documentElement

Informace o uzlech
■ Vlastnosti nodeName, nodeValue a nodeType obsahují informace o uzlech.

nodeName
nodeName vlastnost obsahuje jméno uzlu
- nodename elementného uzlu je jméno značky (tag);
- nodename atributného uzlu je jméno atributu;
- nodename textového uzlu je vždycky # text;
- jméno dokumentního uzlu je vždycky # document.

nodeValue
Pro textové uzly vlastnost nodeValue obsahuje text;
Pro atributní uzly vlastnost nodeValue obsahuje hodnotu atributu;
Pro elementní a dokumentní uzly vlastnost nodeValue není k dispozici.

nodeType
Vlastnost nodeType vrací typ uzlu.

NodeList
■ Když používáme vlastnosti nebo metody jako childNodes nebo getElementsByTagName( ) obdržíme objekt NodeList.
■ Objekt NodeList představuje uspořádaný seznam uzlů. Uzly v seznamu mohou být přístupné přes jejich indexové čislo.

Obdržení délky seznamu NodeList
Vlastnost length vrací počet uzlů v seznamu NodeList.
getElementsByTagName(“title”).length;
Výstup: 4
■ Jestliže známe délku seznamu, můžeme cyklicky prohlížet seznam a vybrat hodnoty, které potřebujeme.

Příklad. Prohlížení všech elementů <autor> a tisk jejich hodnot.

```
// proměnná x bude obsahovat seznam NodeList
  var x=getElementsByTagName("autor");
     for (i=0; i<x.length; i++)
      {
        document.write(x[ i ].childNodes[0].nodeValue);
        document.write("<br/>");
      }
```

Výstup:
Tolstoj
Čechov
Gogol

NamedNodeMap

■ Když použijeme vlastnost attributes pro nějaký element, obdržíme objekt NamedNodeMap.
■ Objekt NamedNodeMap reprezentuje neuspořádaný sezman atributních uzlů.
■ Uzly v seznamu NamedNodeMap mohou být přístupné přes jejich jména

Obdržení délky seznamu NamedNodeMap
Vlastnost length vrací počet uzlů v sezmamu NamedNodeMap.

```
getElementsByTagName("autor")[0].attributes.length;
```

Výstup: 1

Obdržení hodnoty atributu
■ metoda getNamedItem( ) objektu NamedNodeMap vrací specifikovaný uzel.

Obdržení hodnoty atributu

Příklad. Následující fragment kódu ukazuje jak tisknout hodnotu atributu „id" každého elementu <kniha>.

```
xmlDoc=loadXMLDoc("simple-1.xml");
 var x=xmlDoc.getElementsByTagName("kniha");
   for (i=0; i<x.length; i++)
    {
      // proměnná attlist bude obsahovat seznam NamedNodeMap
    var atributseznam=x.[i].attributes;
    var atribut=atributseznam.getNamedItem("id");
    document.write(atribut.value + "<br/>");
   }
```

Výstup:
1
2
3

 Nyní můžeme používat metody objektu DOMDocument pro získání v něm obsažených dat.
Metoda getElementsByTagName(). Metoda vrací objekt NodeList (seznam uzlů).
Uzel má svůj datový obsah, který získáme pomocí vlastnosti nodeValue.

```
 <?php
 $knihy=$instanceDOM->getElementsByTagName("kniha");
  foreach($knihy as $kniha)
         {
       $autor=$kniha->getElementsByTagName("autor");  ?>
           Autor: <?php $autor->item(0)->nodeValue  ?><br/>
<?php  }   ?>
```

 Ačkoli se objekt DOMNodeList svými vlastnostmi velmi podobá poli, není to přesně totéž. Není možné přímo přistupovat k jeho prvkům uvedením indexu prvku v hranatých závorkách, jak je tomu v případě poli PHP.

Namísto toho je třeba použít metodu item, která vrátí požadovaný prvek poli. Index prvku, který chceme vrátit, bere metoda item jako svůj parametr.

Získání hodnoty atributu elementu

Metoda getAttribute(). Metoda bere jako parametr název atributu, který se má v objektu DOMElement najit, a vrací jeho hodnotu.

```
Autoři:<br/>
<?php foreach($autori as $autor)  {  ?>
   Autor-language: <?php $autor->getAttribute("lang") ?><br/>
 <?php  }  ?>
```

 Metoda getAttribute jednoduše vrátí hodnotu atributu.

Třída Node

Vlastnosti: firstChild; lastChild; parentNode; previousSibling; nextSibling; childNodes.

```
<?php
   $zpravy=$instanceDOM->getElementsByTagName("zprava");
     foreach($zpravy as $zprava)  {
           $potomci=$zprava->childNodes;  ?>
       Vydavatel : <?php $potomci->item(1)->nodeValue  ?><br/>
<?php  }  ?>
```

Zadání pro Cvičení:

Vytvořit xml soubor pro knihovnu.

```
<knihovna>
    <sekce nazev="">
        <kniha jazyk=" ">
                <autor>
                <nazev>
                <cena>
                 <vydavatelství>
                 <rok>
          <kniha>
     .   .   .
 </ knihovna>
```

1. Zobrazit všechny knihy (autor, název, cena), které jsou napsané v Angl. jazyce.
2. Zobrazit všechny knihy (autor, název, vydavatelstvo), u kterých cena < N kč.
3. Zobrazit názvy sekcí, které mají zastaralé knihy (alespoň jedná kniha má rok vydání < X rok).
4. Vytvořit vstupní políčko pro zadávání slova. Zobrazit všechny knihy, názvy kterých obsahují toto slovo. Zobrazit autora a název. Hledané slovo zvýraznit v názvu knihy žlutou barvou.

Úkoly pro samostatnou práci:

Prostudovat práci PHP s XML DOM. Použit prezentace na vtan.ujep.cz (Složka IS, soubor L-xmldom-php.ppt).

Kapitola 13. PHP (simpleXML)

Cíl kapitoly:

- Vysvětlit práci PHP se soubory XML pomocí knihovny SimpleXML
- Procvičit zpracování souborů XML pomocí funkcí knihovny SimpleXML

Slovníček pojmů: SimpleXML, metody: attributes(), asXML(), children(), xpath().

Výkladová část:
■ Dokument se načte do Objektu a pak se přistupuje k uzlům dokumentu pomocí odkazů na členské proměnné.
❒ Pomocí funkcí:
simplexml_load_file("some.xml") nebo
simplexml_load_string($xml)
získáme Objekt reprezentující kořenový element dokumentu. Jeho Vlastnosti odpovídají dětským elementům daného elementu.

```
$xml = simplexml_load-file("some.xml");
kořenový element ($xml)
```

Metody:
- attributes()
- asXML()
- children()
- xpath()

■ attributes()

```
simplexml_elememt->attributes()
```

V případě, že souroženců (elementů) je více, můžete se také přímo odkázat na atribut konkrétního elementu.

```
echo $xml->kniha[3]->autor->attributes();
```

V případě, že element má více atributů použijeme cyklus.

```
foreach ($xml->kniha[0]->autor->attributes() as $nazev=>$hodnota)
  {
    echo "$nazev : $hodnota <br/>";
  }
```

■ asXML()

```
string    simplexml_elememt->asXML()
```

Metoda vrátí dobře zformovaný řetězec XML 1.0 založený na objektu SimpleXML.

```
echo htmlspecialchars($xml->asXML(); - vrátí původní dokument XML
```

■ children()

```
simplexml_elememt->children()
```

Metoda vrátí běžné pole (číselné indexy), obsahující dceřiné elementy.

```
<?php
  $xml=simplexml_load_file("some.xml");
   foreach ($xml->kniha[4]->autor->children()  as $child)
     {
       echo "$child<br/>";
     }
?>
```

■ xpath()

```
array   simplexml_elememt->xpath(string  path)
```

xPath – je standard W3C, který nabízí syntax pro identifikaci uzlů XML založenou na cestě.

```
<?php
   $xml=simplexml_load_file("some.xml");
    $autory=$xml->xpath("/knihovna/kniha/autor");
      foreach ($autory as $autor)
        {
          print "$autor<br/>";
        }
```

```
?>
```

■ Pomocí funkce xPath() můžeme také selektivně získat (na základě konkrétní hodnoty) element a jeho děti.

```
<?php
  $xml=simplexml_load_file("some.xml");
   $kniha=$xml->xpath("/knihovna/kniha[autor='Kundera']");
    echo $kniha[0]->nazev;
 ?>
```

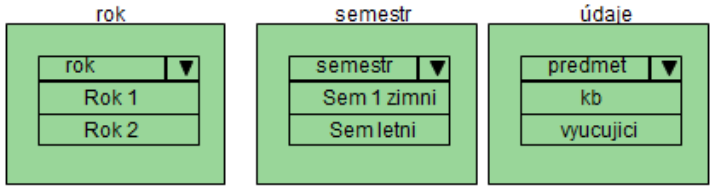
■ Dotazy, které vracejí jen hodnotu, nejsou podporovány. Výsledek dotazu je vždy vrácen jako pole objektů ***SimpleXMLElement***, a to i v případech, kdy výsledkem dotazu jsou jiné uzly než elementy (třeba atributy nebo textové uzly).
Aktuálním uzlem pro vyhodnocování relativních dotazů je vždy kořenový uzel.
Pro zpracování výsledku dotazu používáme příkaz ***foreach***.

```
foreach ($xml->xpath("//item") as $item)
  {
     echo $item->nazev;
  }
```

Pro případ, kdy očekáváme jako výsledek jednu hodnotu, můžeme využít buď Dočasné pole
$a=$xml->xpath("/ ……"); echo "Nazev: ".$a[0];
Nebo můžeme využít toho, že funkce array_pop() vrátí poslední prvek pole:
echo "Nazev: ", array_pop($xml->xpath("/ …"));

Zadání pro Cvičení:
Použit soubor studia.xml

1. Vytvořit formulář (seznam) pro Rok, Semestr a hledané údaje



Pomocí formulářů zadat dotaz na xml soubor (studia.xml). Vyhledávání a zobrazení výsledků provést pomocí funkcí knihovny SimpleXML.
Umožnit vícenásobný výběr pro seznam (údaje).

Úkoly pro samostatnou práci:
Prostudovat práci PHP s XML pomocí knihovny SimpleXML. Použit prezentace na vtan.ujep.cz (Složka IS, soubor
SimpleXML.ppt).

Kapitola 14. PHP (práce s databázemi)

Cíl kapitoly:

- Vysvětlit funkce PHP pro práci s databázemi
- Procvičit výběr a ukládání dat do DB pomocí PHP

Slovníček pojmů: MySQLi, dotazy, objektový a procedurální styl

Výkladová část:
Výběr DB

```
select_db()
```

Jako argument zadáme jméno DB, ke které chceme se připojit.

Např. $mysqli->select_db(“sportovci”)

Dotazy

$mysqli->query()

Argumentem této funkci je SQL příkaz, jenž chcete DB serveru poslat k provedení.

■ Dotaz předavaný funkci mysqli->query() by nemél končit středníkem.

Vracená hodnota je identifikátorem odkazujícím na výsledek tohoto dotazu, který je možné zpracovávat dalšími funkcemi.

Po provedení query() máme k dispozici další funkce:

např, num_rows() - vrací počet řádku obsažených ve výsledku výberového dotazu.

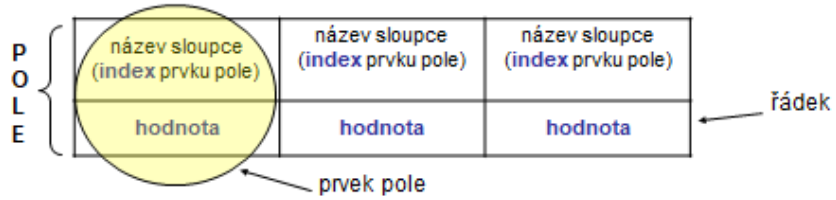
MySQL a PHP 9

PHP

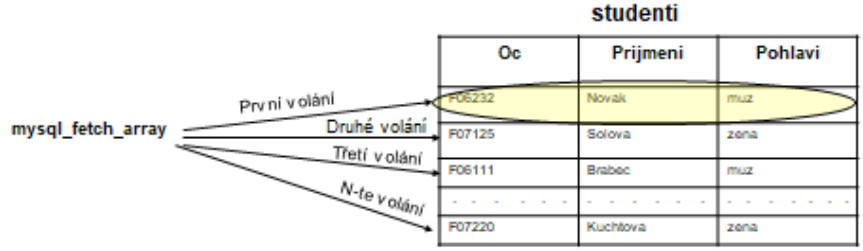
K hodnotám výsledku dotazu se můžeme dostat pomocí několika funkcí:

1. mysql-fetch-array()

Funkce zpřistupní vždy jeden řádek s výsledku v asociativním poli.



■ Každé následující volání mysql_fetch_array() vrací vždy následující řádek z výsledku generovaného původním příkazem SELECT.



Šablona volání funkce: fetch_array()

$row = $vysl->fetch_array(MYSQLI_NUM);

nebo

$row = $vysl->fetch_array(MYSQLI_ASSOC);

2. fetch-row()

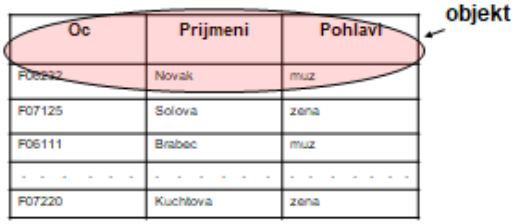
Funkce vrací běžné pole (tj. číselné indexy), tedy bez možnosti přístupu prostřednictvím názvů jednotlivých polí.

3. mysql-fetch-object()

Funkce vrací Obekt. Jednotlivé položky výsledku (tj. řádky) jsou k dispozici jako členské proměnné.

```
class some{
  vlastnost-Oc;
  vlastnost-Prijmeni;
  vlastnost-Pohlavi;
  }

$radek->Prijmeni;
// vrátí Novak
```



- **K hodnotam nelze přistupovat prostřednictvím jejich pořadí, je třeba znát názvy sloupců.**

```
<?php
  mysql_connect();
  mysql_select_db("fakulta");
  $vysl=mysql_query("SELECT * FROM studenti");
    while ($radek=mysql_fetch_object($vysl))
          {
             echo $radek->Oc."<br/>";
             echo $radek->Prijmeni . "<br/>";
              print "<br/><br/>";
          }
 ?>
```

4. ***mysqli_data-seek*** (source id_vysledku, int cislo_zaznamu)
Funkce přesune ukazatel na záznam specifikovaný číslem záznamu.

5. ***mysqli_result*** (source id_vysledku, int cislo_zaznamu [, mixed sloupec])
Získá data z jednoho sloupce specifikovaného řádku. Zatímco se sloupec musí
Volat názvem, řádek musí být pozice vyjádřená celým číslem.

Zadání pro Cvičení:
1. Vytvořit web aplikaci. Na úvodní stránce zobrazit vstupní textové políčka pro zadávání údajů o osobách (3 osoby). Zadávat pouze jednou částku pro každou osobu (např. 100). Částky budou uloženy do DB. Při opakovaném zadávání, částky se budou sečítat. Kdykoliv po kliknutí na tlačítko „Stav“, budou zobrazeny souhrnné údaje o všech osobách. Údaje musí být uspořádané podle částek.

2. Vytvořit DB pro uložení a vyzvedávání informace o studentech. Použít Třídu studenti pro zobrazení tabulky se studenty.

Úkoly pro samostatnou práci:
Prostudovat práci PHP s DB. Použit prezentace na vtan.ujep.cz (Složka IS, soubor L-DB.ppt).

Literatura

základní:

1. David Sklar. PHP 7, Zoner Press, 2018, 368 stran. ISBN: 978-80-7413-363-3.
2. Luke Welling. PHP a MySQL. Computer Press, 2017, 800 stran. ISBN: 9788025148921.
3. Pavel Herout. XSLT 2.0 a SVG prakticky – Xpath 2.0 a Java. Ebook, 2010.

doporučená:

1. J. N. Robbins. Learning Web Design. O'Reilly Media, 2018, 808 pages.
2. C. L. Phang. Web Coding Bible. 2015, 474 pages.
3. HTML 5 Black Book, Covers CSS 3, JavaScript, XML, xHTML, AJAX, PHP and Jquery. Dreamtech Press, eBook, 77472 KB, 2016.
4. MG Martin. PHP: Advanced Detailed Approach to Master PHP Programming Language for Web Development. 2019, 83 page.
5. Antonio Lopez. Learning PHP 7. Packt Publishing, 2016. ISBN: 9781785880544.