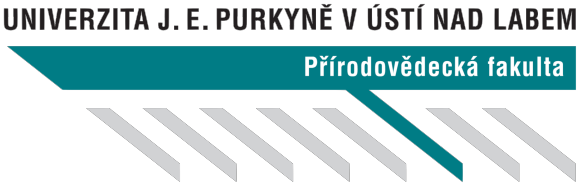
Univerzita Jana Evangelisty Purkyně
v Ústí nad Labem
Přírodovědecká fakulta



Programování pro internet

Příprava na zkoušku

Vypracoval: Kopyto

Studijní program: Aplikovaná informatika
Studijní obor: Informační systémy

Ústí nad Labem 2024

Obsah

1. XML

XML (eXtensible Markup Language) je značkovací jazyk, který umožňuje uživatelům definovat svá vlastní strukturovaná data. Základní syntaxe XML vyžaduje, aby každý dokument obsahoval jeden kořenový element, který obaluje všechny ostatní elementy. XML soubory musí být ”well-formed”, což znamená, že musí správně dodržovat strukturu značek a pravidla syntaxe, jako je správné uzavírání značek a uvozování atributů hodnotami v uvozovkách.

pod tu to otázky spadaj pod otázky: základní syntaxe, jmenný prostor, schéma a datatypes

1.1 základní syntaxe

Pravidla pro psaní XML souboru zahrnují použití správného prologu, což je volitelná deklarace na začátku souboru specifikující verzi XML a použitou znakovou sadu (např. <?xml version=”1.0”encoding=”UTF-8”?>). Jména elementů a atributů by měla být srozumitelná a relevantní k obsahu, který reprezentují, a neměla by obsahovat mezery ani speciální znaky. Co se týče entit, XML definuje několik vestavěných entit pro speciální znaky (např. < pro znak menší než, > pro znak větší než, & pro ampersand, atd.), které umožňují vkládat do dokumentů speciální znaky. Výše uvedené aspekty jsou klíčové pro správné psaní a porozumění XML.

1.2 Jmenný prostor

Jmenný prostor (namespace) v XML je metoda, která umožňuje rozlišovat elementy a atributy, které by mohly mít stejná jména, ale různé významy, v závislosti na kontextu, ve kterém jsou použity. Jmenné prostory jsou definovány pomocí atributu xmlns v základním nebo jiném elementu XML dokumentu, čímž se určuje URI (Uniform Resource Identifier), které slouží jako unikátní identifikátor pro daný jmenný prostor.

Pro přidání jmenného prostoru do elementu v XML dokumentu je běžně používán atribut xmlns, například <knihy xmlns=”http://www.example.com/knihy». Tento přístup označuje, že element knihy a všechny jeho dětské elementy patří do jmenného prostoru http://www.example.com/knihy. Když chcete použít elementy z více jmenných prostorů v jednom dokumentu, můžete definovat prefixy, jako je xsl:template, kde xsl je prefix odkazující na jmenný prostor definovaný například takto: xmlns:xsl=”http://www.w3.org/1999/XSL/Transform”.

Tato struktura umožňuje XML dokumentům být flexibilní a zároveň přesně definované, což je klíčové pro jejich správnou funkčnost a integraci v různých aplikacích a systémech.

1.3 obsah elementu

Obsah elementu v XML se týká všeho, co je umístěno mezi otevírací a zavírací značkou tohoto elementu. Tento obsah může zahrnovat text, další elementy (což umožňuje vytvářet hierarchické struktury), komentáře, a entity (které umožňují zahrnutí speciálních znaků, jako jsou například &, <, >). Kromě toho může obsah zahrnovat data v různých formátech, jako jsou čísla, řetězce nebo datumy, v závislosti na specifikaci a účelu daného XML dokumentu. Význam a struktura obsahu dokumentu je definována schematem nebo typem dokumentu, který XML používá, což umožňuje flexibilní a přesnou manipulaci s daty.

1.4 XML schéma

XML schema, obvykle definované ve W3C XML Schema Definition Language (XSD), umožňuje návrhářům specifikovat pravidla a omezení pro strukturu XML dokumentu, včetně typů dat pro jednotlivé elementy a atributy, jejich výskyt v dokumentu, a vztahy mezi různými elementy. Pomocí XML schema můžete například definovat, že určitý element musí obsahovat pouze číselné hodnoty, nebo že jiný element může obsahovat jiné elementy v určitém pořadí. XML schema také podporuje vytváření vlastních datových typů a použití omezení, jako jsou minimální a maximální hodnoty nebo specifické formáty pro řetězce. Tato schémata jsou klíčová pro automatizovanou validaci XML dokumentů, což zajišťuje, že data vyhovují definovaným standardům a jsou konzistentní napříč různými systémy a aplikacemi.

```
<?xml version="1.0" encoding="UTF-8"?>
<knihovna>
    <kniha id="001">
        <nazev>ůPrvodce galaxií pro řstopae</nazev>
        <autor>Douglas Adams</autor>
        <vydano>1979</vydano>
    </kniha>
    <kniha id="002">
        <nazev>1984</nazev>
        <autor>George Orwell</autor>
        <vydano>1949</vydano>
    </kniha>
</knihovna>
```

Obrázek 1.1: Ukázka XML souboru

```
<?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>
```

Obrázek 1.2: Ukázka přidání schematu

1.5 XML datové typy

XML Schema poskytuje širokou škálu vestavěných datových typů, které umožňují přesně specifikovat a omezovat data v XML dokumentech. Základní datové typy zahrnují string pro textové řetězce, integer pro celá čísla, boolean pro pravdivostní hodnoty, a date pro data. Kromě těchto základních typů, XML Schema definuje i složitější datové typy jako decimal pro čísla s desetinnými místy, duration pro časové úseky, nebo gYearMonth pro specifikaci roku a měsíce. Uživatelé také mohou vytvářet vlastní datové typy pomocí omezení (restriction) na stávající typy, což umožňuje velmi specifickou validaci dat podle potřeb aplikace. Tato flexibilita a přesnost jsou klíčové pro efektivní využití XML v různých aplikacích a systémech.

2. DTD a XSD

2.1 DTD

DTD (Document Type Definition), což je jeden z prvních a základních mechanismů pro definování struktury a pravidel pro XML dokumenty. DTD umožňuje specifikovat, které elementy, atributy a entity mohou v dokumentu existovat, jaké jsou jejich vztahy a jak často se mohou v dokumentu objevit. Toto je klíčové pro zajištění konzistence dat mezi různými systémy a pro validaci struktury XML dokumentů před jejich zpracováním.

DTD je obvykle zapsáno buď přímo v XML dokumentu, nebo jako externí soubor, který je poté odkazován z XML. K definování pravidel v DTD se používají deklarace elementů a atributů. Například, můžete definovat, že každý dokument musí obsahovat kořenový element <knihovna>, který může obsahovat několik <kniha> elementů, přičemž každý <kniha> element musí obsahovat <nazev>, <autor>, a <vydano>.

2.1.1 Datové typy obsahu

1. **#PCDATA** - Znamená "Parsed Character Data", což umožňuje vložení běžného textu. Tento typ se používá, když chcete, aby element obsahoval jen text.
2. **Prázdný** - Pomocí klíčového slova EMPTY můžete specifikovat, že element nesmí obsahovat žádný obsah ani žádné další elementy.
3. **Smíšený obsah** - Můžete definovat element, který může obsahovat kombinaci textu a dalších elementů. To se specifikuje pomocí modelu obsahu kombinujícího #PCDATA s názvy dalších elementů.

2.1.2 Datové typy atributů

1. **CDATA** - Pro běžné textové řetězce.
2. **ID** - Unikátní identifikátor v rámci dokumentu.
3. **IDREF/IDREFS** - Odkaz na element s typem ID.
4. **NMTOKEN/NMTOKENS** - Token, který musí odpovídat XML názvovým konvencím (např. musí začínat písmenem).
5. **ENUMERATION** - Umožňuje definovat seznam povolených hodnot.
6. **NMOTATION** - Odkaz na notaci definovanou v DTD.

```
<!DOCTYPE knihovna [
<!ELEMENT knihovna (kniha+)>
<!ELEMENT kniha (nazev, autor, vydano)>
<!ELEMENT nazev (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT vydano (#PCDATA)>
]>
```

Obrázek 2.1: Ukázka DTD souboru

2.2 XSD

XSD je mocný nástroj pro definování struktury a validaci XML dokumentů. Na rozdíl od DTD poskytuje XSD podrobnější specifikace pro datové typy, podporuje jmenné prostory a umožňuje vytváření složitých a omezujících pravidel pro XML dokumenty. XSD se používá pro definování pravidelné struktury XML dokumentů, což zajišťuje, že data splňují definované formáty a omezení. To je nezbytné pro aplikace, které vyžadují vysokou úroveň integrity dat, jako jsou finanční systémy, zdravotnické informační systémy, a webové služby, kde je důležitá správná a konzistentní interpretace dat mezi různými systémy a platformami. Hlavní kategorie datových typů

1. **Primitivní datové typy** - To jsou základní typy, ze kterých jsou odvozeny všechny ostatní typy.
2. **Odvozené datové typy** - Tyto typy jsou odvozeny z primitivních typů a mohou zahrnovat dodatečná omezení.
3. **Speciální datové typy**

2.2.1 Primitivní datové typy

- **‘xs:string‘** - Pro textové řetězce.
- **‘xs:boolean‘** - Pro pravdivostní hodnoty (true/false).
- **‘xs:decimal‘, ‘xs:integer‘, ‘xs:float‘, ‘xs:double‘** - Pro číselné hodnoty, kde ‘xs:decimal‘ a ‘xs:integer‘ jsou pro celá a desetinná čísla bez ztráty přesnosti, zatímco ‘xs:float‘ a ‘xs:double‘ jsou pro plovoucí desetinné čárky s možnou ztrátou přesnosti.
- **‘xs:date‘, ‘xs:time‘, ‘xs:dateTime‘, ‘xs:duration‘** - Pro datum, čas a jejich kombinace.

2.2.2 Odvozené datové typy

- **‘xs:byte‘, ‘xs:short‘, ‘xs:long‘, ‘xs:unsignedShort‘, atd.** - Různé formáty číselných typů pro specifické potřeby, jako jsou velikosti a znaménka.
- **‘xs:nonNegativeInteger‘, ‘xs:positiveInteger‘** - Pro nezáporná a kladná celá čísla.

2.2.3 Speciální datové typy

- **‘xs:ID‘, ‘xs:IDREF‘, ‘xs:ENTITY‘, atd.** - Speciální účelové typy pro unikátní identifikátory, odkazy na identifikátory a jména entit.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="knihovna">
        <xs:complexType>
            <xs:sequence>
                <xs:element
                    name="kniha"
                    maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element
                                name="nazev"
                                type="xs:string"/>
                            <xs:element
                                name="autor"
                                type="xs:string"/>
                            <xs:element
                                name="vydano"
                                type="xs:date"/>
                        </xs:sequence>
                        <xs:attribute
                            name="id"
                            type="xs:string"
                            use="required"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Obrázek 2.2: Ukázka XSD souboru

3. XSL

XSL zahrnuje XSLT (XSL Transformations), XPath (XML Path Language) a XSL-FO (XSL Formatting Objects), které umožňují transformovat, vyhledávat a formátovat XML data pro různé výstupní formáty, jako je HTML, text, nebo PDF.

XSLT je pravděpodobně nejčastěji používaná část XSL a slouží k transformaci XML dokumentů do jiných formátů (např. HTML) pomocí šablonového přístupu. Tento proces transformace umožňuje zobrazit XML data ve webových prohlížečích nebo je integrovat do jiných aplikací.

```
<?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>Seznam knih</h2>
        <table border="1">
          <tr>
            <th>Název</th>
            <th>Autor</th>
          </tr>
          <xsl:for-each select="knihovna/kniha">
            <tr>
              <td><xsl:value-of select="nazev"/></td>
              <td><xsl:value-of select="autor"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Obrázek 3.1: Ukázka XSL souboru

3.0.1 Šablony a Výběry (Templates and Matching)

- **‘<xsl:template match=”...»‘** - Definuje pravidla pro zpracování určitých částí XML dokumentu. Atribut ’match’ umožňuje specifikovat XPath výrazy, které identifikují elementy nebo atributy pro aplikaci šablony.

3.0.2 Podmíněné Zpracování (Conditional Processing)

- **‘<xsl:if test=”...»‘** - Provádí test, a pokud je výsledek pravdivý, aplikuje určitý kód.
- **‘<xsl:choose>‘, ‘<xsl:when>‘, a ‘<xsl:otherwise>‘** - Slouží pro vytváření komplexnějších podmíněných struktur, kde ’<xsl:choose>’ funguje jako kontejner pro jednu nebo více ’<xsl:when>’ větví a volitelný ’<xsl:otherwise>’ pro výchozí chování.

3.0.3 Iterace (Looping)

- **‘<xsl:for-each select=”...»‘** - Iteruje přes množinu uzlů specifikovanou XPath výrazem v ’select’ a aplikuje vložené šablony na každý uzel.

3.0.4 Řazení (Sorting)

- **‘<xsl:sort select=”...”order=”ascending|descending”data-type=”text|number»‘** - Používá se uvnitř ’<xsl:for-each>’ nebo ’<xsl:apply-templates>’ pro řazení uzlů podle textu nebo číselných hodnot.

3.0.5 Vkládání Hodnot (Outputting Values)

- **‘<xsl:value-of select=”...»‘** - Extrahuje text z vybraného uzlu a vkládá ho do výstupního dokumentu.
- **‘<xsl:copy-of select=”...»‘** - Kopíruje celé uzly z vstupního dokumentu do výstupu, včetně všech dětských uzlů.

3.0.6 Tvorba a Použití Proměnných (Variables and Parameters)

- **‘<xsl:variable name=”...”select=”...»‘** - Definuje proměnnou, která může být použita v rámci šablony.
- **‘<xsl:param name=”...»‘** - Definuje parametr, který může být předán do šablony z vnějšího kontextu nebo jiné šablony.

3.0.7 Rekurze (Recursion)

- **‘<xsl:apply-templates select=”...»‘** - Aplikuje šablony na vybrané uzly, což může zahrnovat rekurzivní volání na uzly, které samy obsahují další uzly pro zpracování.

4. XML DOM

XML DOM (Document Object Model) je konvence pro interakci a manipulaci s XML dokumenty. XML DOM definuje strukturu XML dokumentů jako strom objektů, kde každý objekt reprezentuje část dokumentu, jako jsou elementy, atributy, text a tak dále. Tento model umožňuje programátorský přístup k dokumentům, což znamená, že můžete programově přidávat, měnit, a mazat uzly v XML dokumentu.

4.1 K čemu je XML DOM potřeba

XML DOM je nezbytný pro dynamickou interakci s XML dokumenty v různých programovacích prostředích, jako jsou webové prohlížeče, serverové aplikace a mnoho dalších, kde je potřeba XML dokumenty číst, upravovat, nebo generovat. Výhodou použití DOM je jeho nezávislost na platformě a jazyku, což znamená, že stejný DOM kód může fungovat ve více prostředích.

4.2 Správa

Správu XML DOM (Document Object Model) standardů zajišťuje W3C (World Wide Web Consortium), který je mezinárodní komunitou, jež spolupracuje na vývoji webových standardů. W3C vytvořilo a udržuje mnoho specifikací souvisejících s webovými technologiemi, včetně HTML, CSS, a XML. DOM specifikace, která zahrnuje i XML DOM, jsou součástí těchto standardů a W3C pravidelně aktualizuje a zlepšuje tyto specifikace, aby odpovídaly novým technologickým trendům a potřebám vývojářů.

4.3 Hierarchie uzlů XML DOM

XML DOM reprezentuje XML dokument jako hierarchický strom objektů, který umožňuje snadnou manipulaci s uzly. Každý uzel v stromu odpovídá určité části dokumentu. Zde je výpis hlavních typů uzlů:

- **Dokument** - Vrcholový uzel reprezentující celý XML dokument.
- **Element** - Uzly, které odpovídají značkám v XML. Mohou obsahovat další elementy, texty, komentáře, atd.
- **Atribut** - Uzly, které definují vlastnosti elementů.
- **Text** - Uzly, které obsahují textová data mezi značkami elementů.
- **Komentář** - Uzly, které obsahují komentáře v XML dokumentech.

4.4 Funkce uzlů

Každý uzel má několik vlastností a metod, které umožňují manipulaci s ním, například:

- **childNodes** - Seznam všech dětských uzlů.
- **parentNode** - Odkaz na rodičovský uzel.
- **appendChild()** - Přidává nový uzel do dětských uzlů.
- **removeChild()** - Odstraňuje uzel z dětských uzlů.
- **getAttribute()**, **setAttribute()** - Metody pro práci s atributy elementů.

4.5 Práce s uzly

Práce s XML DOM umožňuje efektivní a dynamické manipulace s dokumenty pro různé aplikace od webů po serverové aplikace. To zahrnuje přidávání, měnění, a mazání uzlů, stejně jako navigaci a filtrování v rámci stromu uzlů.

5. xPath

XPath, zkratka pro XML Path Language, je jazyk pro výběr uzlů z XML dokumentů. XPath umožňuje navigovat přes elementy a atributy v XML dokumentech, což je užitečné pro různé účely, jako je extrakce informací, manipulace s daty nebo při transformacích XML pomocí XSLT. XPath definuje cestu k určitému uzlu nebo skupině uzlů a používá se pro jednoduché až složité dotazy na strukturu XML dokumentu.

5.1 Správa

XPath je spravován W3C (World Wide Web Consortium), což je mezinárodní komunita, která vyvíjí otevřené standardy pro zajištění dlouhodobého růstu Webu. W3C udržuje a aktualizuje specifikace XPath, aby byly kompatibilní s ostatními technologiemi a vyhovovaly novým potřebám uživatelů a vývojářů.

5.2 Struktura XPath

XPath využívá cesty podobné adresám ve filesystemech k lokalizaci informací v XML dokumentech. Několik základních syntaktických prvků zahrnuje:

- **/** - označuje kořenový uzel dokumentu a zahajuje absolutní cestu.
- **//** - vyhledává uzly v dokumentu od aktuálního uzlu, které odpovídají vzoru bez ohledu na jejich hloubku.
- **.** - reprezentuje aktuální uzel.
- **..** - reprezentuje rodičovský uzel aktuálního uzlu.
- **@** - používá se pro výběr atributů.

6. Postata DOM pro XML a HTML

1. **Hierarchická struktura:** DOM transformuje celý HTML nebo XML dokument do stromové struktury, kde každý uzel stromu reprezentuje část dokumentu, jako jsou elementy, text, komentáře a atributy. Tento strom umožňuje snadný přístup a manipulaci s jednotlivými uzly.

2. **Programové rozhraní:** DOM poskytuje univerzální rozhraní, díky kterému mohou programy a skripty dynamicky interagovat s dokumentem. Například, JavaScript může přidávat, odstraňovat nebo měnit elementy, reagovat na události uživatelů, měnit styly a tak dále.

3. **Nezávislost na platformě a jazyku:** Ačkoli DOM je úzce spojen s webovými prohlížeči a JavaScriptem, je to standardizovaná technologie, kterou spravuje W3C (World Wide Web Consortium) a je nezávislá na platformě nebo programovacím jazyku. To znamená, že jakýkoliv jazyk, který podporuje DOM (např. Python s knihovnou pro práci s webovými dokumenty), může manipulovat s HTML nebo XML dokumenty.

4. **Dynamické změny dokumentu:** DOM umožňuje skriptům provádět změny v dokumentu "za běhu", což znamená, že změny se projeví ihned v prohlížeči bez potřeby znovu načítat stránku. Toto je zásadní pro dynamické webové aplikace, jako jsou jednostránkové aplikace (SPA), které se spoléhají na rychlé a plynulé uživatelské rozhraní.

5. **Podpora událostí:** DOM definuje, jak jsou události zpracovány v dokumentech. Skripty mohou reagovat na události uživatelů, jako jsou kliknutí myši, stisknutí kláves, pohyby myši atd., což umožňuje vytvářet interaktivní webové stránky.

7. Javascript a XML DOM

JavaScript je dynamický skriptovací jazyk, který se používá pro vytváření interaktivních webových stránek a aplikací. XML DOM (Document Object Model) je programové rozhraní, které umožňuje skriptům, jako je JavaScript, dynamicky přistupovat k a manipulovat s XML dokumenty. XML DOM reprezentuje XML dokument jako strom objektů, což umožňuje programátorům jednoduše změnit strukturu dokumentu, obsah nebo styl. JavaScript spolu s XML DOM poskytuje výkonné nástroje pro práci s XML daty na klientově straně, což umožňuje například zobrazování, úpravu, validaci nebo přenos XML dat mezi klientem a serverem bez potřeby obnovovat celou stránku.

7.1 Správa

XML DOM je spravován World Wide Web Consortium (W3C), což je mezinárodní komunita, která vyvíjí otevřené webové standardy, aby zabezpečila dlouhodobý růst webu. W3C udržuje specifikace XML, XML DOM a mnoho dalších technologií, které jsou klíčové pro interoperabilitu mezi webovými technologiemi.

7.2 Struktura XML DOM

XML DOM reprezentuje XML dokumenty jako strom objektů, kde každý uzel v stromu odpovídá určité části dokumentu. Následuje stručný popis hlavních typů uzlů:

- **Element Nodes** - Reprezentují elementy XML a mohou obsahovat další uzly, jako jsou text, další elementy, nebo atributy.
- **Text Nodes** - Obsahují textový obsah elementů.
- **Attribute Nodes** - Uchovávají atributy příslušných elementů, jsou přístupné prostřednictvím elementů, které je obsahují.
- **Comment Nodes** - Umožňují vložení komentářů do XML dokumentů.

7.3 Manipulace s uzly pomocí JavaScriptu

JavaScript umožňuje manipulaci s uzly v XML DOM, což zahrnuje přidávání, mazání a modifikaci uzlů, stejně jako změnu jejich atributů. Toto je základní přístup k dynamické interakci s XML na webových stránkách.

8. Javascript a HTMLdom

Podstat JavaScriptu v HTMLdom je totožná s XMLdom viz. 7

8.1 K čemu je to potřeba

JavaScript společně s HTML DOM se využívá pro vytváření interaktivních a dynamických webových aplikací. Umožňuje reagovat na uživatelské akce, jako jsou kliknutí myši a stisky kláves, a dynamicky aktualizovat obsah bez nutnosti znovunačítat stránku. To zlepšuje uživatelskou zkušenost a efektivitu webových aplikací.

8.2 Kdo to spravuje

HTML DOM je standardizovaný a spravovaný World Wide Web Consortium (W3C), mezinárodní organizací, která udržuje a vyvíjí webové standardy, aby zajistila dlouhodobý růst a srozumitelnost internetu.

8.3 Hierarchie HTML DOM

HTML DOM reprezentuje strukturu HTML dokumentu jako strom uzlů, kde každý uzel může být:

- **Document** - Reprezentuje celý dokument.
- **Element** - Reprezentuje HTML elementy jako jsou , <p>, <a> atd.
- **Text** - Obsahuje text uvnitř elementů.
- **Attribute** - Reprezentuje atributy elementů, jako je 'class', 'id', atd.

8.4 Manipulace s uzly

JavaScript může manipulovat s HTML DOM pomocí různých metod a vlastností, umožňující:

- Přidávat nové uzly.
- Odstraňovat stávající uzly.
- Měnit obsah a atributy uzlů.

9. PHP

PHP je serverový skriptovací jazyk, který se široce používá pro vývoj webových aplikací. PHP je zkratka pro ”PHP: Hypertext Preprocessor”a byl navržen tak, aby snadno integroval skriptování na straně serveru s HTML. Umožňuje vývojářům vytvářet dynamické webové stránky, které mohou interagovat s databázemi a provádět komplexní funkce.

9.1 Základy

PHP kód se píše ve skriptech umístěných mezi tagy <?php a ?>. Kód se spustí na serveru, který generuje HTML, jež je odesíláno do webového prohlížeče. Toto umožňuje vytvářet personalizovaný obsah pro uživatele.

9.2 Struktury

PHP podporuje běžné programovací struktury, včetně podmínek (if, else), smyček (for, while, foreach) a funkcí. Tyto struktury umožňují vývojářům psát dobře organizovaný a opakovaně použitelný kód.

9.3 Formuláře

PHP je obzvláště užitečné pro zpracování dat z HTML formulářů. Data odeslaná formulářem mohou být přijata PHP skriptem a použita pro různé účely, jako je ukládání informací do databáze nebo validace uživatelských vstupů.

9.4 Pole a proměnné

Proměnné v PHP jsou uvozeny znakem $ a mohou obsahovat širokou škálu datových typů, včetně řetězců, celých čísel a polí. Pole mohou být indexovaná nebo asociativní, kde indexovaná pole používají číselné indexy a asociativní používají názvy klíčů.

9.5 Kdo to spravuje

PHP je open-source a je spravováno PHP Group, která zodpovídá za vývoj jazyka. PHP Group zveřejňuje aktualizace a udržuje dokumentaci, která je dostupná na oficiálních webových stránkách PHP.

```php
<?php
// Získání dat z formulare
$jmeno = $_POST['jmeno'];
$email = $_POST['email'];

// Zobrazení dat
echo "Jméno:$jmeno<br>";
echo "Email:$email<br>";
?>
```

Obrázek 9.1: Ukázka php kódu

9.6 PHP Objekty

PHP objekty jsou instance tříd, které jsou základními stavebními kameny objektově orientovaného programování (OOP) v PHP. OOP umožňuje programátorům organizovat kód do modulárních, znovupoužitelných jednotek zvaných třídy, které definují vlastnosti (atributy) a chování (metody) objektů.

9.6.1 Definice a použití

Objekty jsou vytvořeny pomocí klíčového slova `new` a jména třídy. Například:

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

    public function __construct($jmeno, $vek) {
        $this->jmeno = $jmeno;
        $this->vek = $vek;
    }

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

$osoba = new Osoba("Petr", 25);
echo $osoba->pozdrav();
```

Obrázek 9.2: PHP objekt

9.7 PHP Vyjímky

Vyjímky jsou způsob, jakým PHP zpracovává chyby v objektově orientovaném programování. Vyjímky umožňují programu přerušit normální tok vykonávání a vykonat kód určený k ošetření chyby.

9.7.1 Druhy a dělení

Ve výchozím nastavení PHP poskytuje několik základních tříd vyjímek, jako je `Exception` a `ErrorException`. Vývojáři mohou vytvářet vlastní vyjímky odvozením od těchto základních tříd. Vyjímky mohou být rozděleny podle úrovně závažnosti nebo podle komponenty, ve které se chyba objeví.

9.7.2 Použití vyjímek

Pro zachycení vyjímek se používá blok `try-catch`. Během vykonávání bloku `try` může být vyvolána vyjímka, která je následně zachycena v bloku `catch`, kde je možné ji zpracovat.

```
try {
    $cislo = 0;
    if($cislo == 0) {
        throw new Exception("Deleni␣nulou.");
    }
    $vysledek = 100 / $cislo;
} catch (Exception $e) {
    echo "Chyba:" . $e->getMessage();
}
```

Obrázek 9.3: PHP vyjímky

9.8 PHP a XMLdom

PHP umožňuje dynamickou manipulaci s XML pomocí XML DOM. XML DOM v PHP poskytuje rozhraní pro práci s XML dokumenty jako s objektovými stromy.

```php
<?php
$doc = new DOMDocument();
$doc->load('example.xml');
$root = $doc->documentElement;
$elements = $root->getElementsByTagName('element');
foreach ($elements as $el) {
    echo $el->nodeValue;
}
?>
```

Obrázek 9.4: Ukázka xml a php

10. SimpleXML

SimpleXML je rozšíření jazyka PHP navržené pro snadnou manipulaci s XML daty. Umožňuje vývojářům přistupovat k XML dokumentům a manipulovat s nimi s minimálními programovými nároky, převádějíc XML dokumenty přímo na objekty v PHP, které lze snadno procházet a modifikovat.

10.1 K čemu je SimpleXML potřeba

SimpleXML je ideální pro aplikace, kde je třeba rychle a efektivně zpracovávat XML data. Typické použití zahrnuje čtení konfiguračních souborů, zpracování odpovědí z webových API, nebo transformaci XML dat do HTML nebo jiných formátů v PHP.

10.2 Kdo spravuje SimpleXML

SimpleXML, jako součást oficiálního PHP, je spravováno PHP Group. Tato organizace udržuje PHP a jeho rozšíření a zajišťuje, že PHP zůstává aktuální a bezpečné pro vývojáře a uživatele.

10.3 Ukázka kódu

Následující příklad ukazuje, jak lze v PHP pomocí SimpleXML načíst XML soubor a vypsat obsah určitého elementu:

```
<?php
$xml = simplexml_load_file('example.xml');
echo 'Jméno: ' . $xml->name;
?>
```

Seznam použité literatury

1. CHATGPT4. Chat instance [https://chatgpt.com/share/a0565e35-9109-4513-8d07-2ba0308cf54d]. 11. června 2024.