Preberanie dát z Portálu VŠ

Praktické využitie v cms systémoch

FAkulta Ekonomiky a manŽmentu SPU v Nitre

2016

# Úvod

Cieľom tohto dokumentu je čitateľovi priblížiť jednoduchý spôsob získavania a spracovania informácií o fakulte, resp. o študijných programoch fakulty, získaných z Portálu VŠ. Zameraný je na spracovanie informácii pre potreby CMS (Content Management System) systémov, ktoré sú dostupné a používané univerzitami na Slovensku.

# Stručne o Portáli VŠ

Hlavným cieľom portálu je zabezpečenie vzniku systematicky usporiadaných a odborne garantovaných informácií týkajúcich sa vysokých škôl, vytvorenie priestoru na výmenu skúseností a poskytovanie informácií o konferenciách, seminároch a vzdelávacích produktoch. Portál VS umožňuje preberanie obsahu v podobe výmenných formátov XML či JSON.

# Stručne o XML

XML znamená eXtensible Markup Language, v preklade rozšíriteľný značkovací jazyk, ktorý bol vyvinutý a štandardizovaný konzorciom W3C (World Wide Web Consortium) ako pokračovanie jazyka SGML a zovšeobecnenie jazyka HTML. Umožňuje jednoduché vytváranie konkrétnych značkovacích jazykov na rôzne účely a široké spektrum rôznych typov údajov.

Jazyk je určený predovšetkým na výmenu údajov medzi aplikáciami a na zverejňovanie dokumentov. Jazyk umožňuje opísať štruktúru dokumentu z hľadiska vecného obsahu jednotlivých častí a nezaoberá sa sám osebe vzhľadom dokumentu alebo jeho časti.

Jazyk XML nemá žiadne preddefinované značky (tagy, názvy jednotlivých elementov) a tiež jeho syntax je podstatne prísnejšia (a jednoduchšia) ako syntax HTML.

Sila XML je najmä v jeho hierarchickej štruktúre a pomerne jednoduchom spôsobe zápisu. Umožňuje opisovať – označovať ľubovoľné dáta a prenášať ich medzi rôznymi aplikáciami a platformami.

Jazyk XML nám dovoľuje vytvárať štruktúrované dokumenty veľmi flexibilným spôsobom. Dá sa použiť k vytváraniu dokumentov, ktoré sa zdanlivo podobajú dokumentom HTML.

XML a HTML v skutočnosti slúžia veľmi odlišným účelom. HTML obsahuje iba informácie o zobrazení dát a neobsahuje údaje o tom, o aké dáta ide. XML sa pokúša oddeliť obsah a jeho reprezentáciu. Dôležitým rozdielom je fakt, že môžeme definovať svoje vlastné značky, ktoré určujú štruktúru dát.

Každý dokument XML sa skladá z kombinácie dát značiek a znakov. Značky dávajú dokumentu XML štruktúru, zatiaľ čo znaky predstavujú vlastný obsah. Všetky XML dokumenty spĺňajúce špecifikáciu XML musia dodržiavať isté pravidlá – potom sú považované za správne štruktúrované (well-formed).

1. Každý element XML musí mať začiatočnú aj koncovú značku. „Prázdne“ elementy bez koncovej značky (ako napríklad <IMG> alebo <HR> v jazyku HTML) nie sú v XML povolené. Je tu však skrátený zápis, ktorý sa dá použiť v prípade, keď daný element neobsahuje žiadne dáta. Namiesto <empty></empty> môžeme použiť alternatívu <empty />. Tieto dve možnosti sa považujú za zhodné.
2. Dokument XML musí obsahovať jediný pár značiek (skladajúci sa zo začiatočnej a koncovej značky), tzv. koreňový element dokumentu, v ktorom sú všetky ostatné elementy vložené. Na koreni nemôže byť viac elementov.
3. Počiatočné a koncové značky každého elementu musia byť riadne vnorené – vnorený element musí byť celý obsiahnutý vo svojom nadradenom elemente. Inými slovami, počiatočné a koncové značky vložených elementov sa nesmú prekrývať, ako to bolo povolené v HTML. To zaisťuje hierarchickú štruktúru dokumentu XML.

## Parsovanie XML

XML je značkovací jazyk, ktorý je medzi vývojármi obľúbený najmä vďaka jeho prehľadnosti a jednoduchosti. Najčastejšie sa používa na ukladanie konfigurácie programov a v jednoduchších aplikáciách dokáže poskytnúť rozumnú alternatívu k relačným databázovým systémom. XML tvorí aj základ protokolu SOAP a mnohých ďalších technológií ako napríklad AJAX či RSS. Procesu získavania údajov z XML štruktúr sa hovorí “parsovanie” a pre väčšinu programovacích jazykov existuje tzv. XML parser, čiže nástroj na čítanie XML štruktúr. Z XML si bez väčších problémov mali poradiť napr. ANSI C, PERL, JAVA, DELPHI či PHP. Závisí na programátorovi, ktorí jazyk ovláda, respektíve na systéme do ktorého chceme dáta preberať.

# Stručne o JSON

JSON (JavaScript Object Notation) je odľahčený formát na výmenu dát. Je jednoducho čitateľný i zapisovateľný človekom a jednoducho analyzovateľný i generovatelný strojovo. Je založený na podmnožine programovacieho jazyka JavaScript, Standard ECMA-262 3rd Edition - December 1999. JSON je textový, na jazyku nezávislý formát, využívajúci však konvencie dobre známe programátorom jazykov rodiny C (C, C++, C#, Java, JavaScript, Perl, Python a dalších). Vďaka tomu je JSON ideálnym jazykom pre výmenu dát.

Je založený na dvoch typoch štruktúr:

Kolekcie párov názov/hodnota. Tá býva v rozličných jazykoch realizovaná ako objekt, záznam (record), štruktúra (struct), slovník (dictionary), hash tabuľka, kľúčový zoznam (keyed list) alebo asociatívne pole.

Zoradený zoznam hodnôt. Ten je vo väčšine jazykov realizovaný ako pole, vektor, zoznam (list) alebo postupnosť (sequence).

Jedná sa o univerzálne dátové štruktúry a v podstate všetky moderné programovacie jazyky ho v nejakej forme podporujú.

# JSON vs. XML

## JSON

* Plusy:
	+ Jednoduchá syntax, menšie množstvo značkovacích prvkov
	+ Jednoduché použitie s JavaScript, majú s JavaScript rovnaké základné dátove typy
	+ JSON Schema pre popis, dátový typ a validáciu štruktúry
	+ JsonPath pre extrahovanie informácií z hlboko vnorených štruktúr
* Mínusy:
	+ Na úkor jednoduchej syntaxi je podporovaná iba malá množina dátových typov

## XML

* Plusy:
	+ Je možné vytvoriť "dialekty" pre akýkoľvek účel
	+ XML Schema pre dátový typ a validáciu štruktúry. Je tiež možné vytvoriť nové dátové typy
	+ XSLT pre transformáciu do rôznych výstupných formátov
	+ XPath / XQuery pre extrakciu informácií z hlboko vnorených štruktúr
	+ podpora pre namespace
* Mínusy:
	+ V porovnaní s JSON je potreba omnoho viac dát pre rovnaké množstvo informácií.

Nasledujúci výstup je príklad výstupu z Portálu VS, o informáciách o fakulte. Výstup je v XML formáte.

## Príklad výstupu XML z PVS:



Informácie rovnaké ako v predchádzajúcom príklade, ale výstup je vo formáte JSON.

## Príklad výstupu JSON z PVS:



# Preberanie dát z Portálu VŠ – praktické príklady

Príklady sú inšpirované implementáciou na FEM SPU v Nitre, kde takéto preberanie dát funguje v reálnej prevádzke. FEM SPU v Nitre má web postavený na CMS Contao. Je programovaný v jazyku PHP ako väčšina používaných CMS a preto sa aj v tomto dokumente zameriame práve na jazyk PHP. Pomocou knižnice cURL, vieme poslať požiadavku, ktorá nám podľa identifikačného čísla (ďalej len ID), pošle požadované dáta.

Portál VŠ vie na požiadanie poskytnúť generovaný XML alebo JSON súbor pre jednotlivé školy a fakulty na základe číselníkov. Tieto súbory obsahujú informácie o škole, jej fakultách, či jednotlivých študijných odboroch. V našom prípade si ukážeme preberanie dát o študijných programov FEM SPU v Nitre.

## Formát URL adresy

Zaslanie požiadavky na Portál VŠ je v princípe veľmi jednoduché. Potrebujeme len adresu s parametrami pre konkrétnu fakultu a formát výstupu ktorý očakávame.

$request\_url=”https://www.portalvs.sk/sk/informacie-o-vysokych

 skolach?export=json&fakulta=704020000&programy=1“;

## Parametre URL adresy

* export: formát očakávaného výstupu (json, xml)
* fakulta: ID fakulty (dá sa zistiť na Portáli VŠ)
* programy: zahrnúť do výstupu aj informácie o študijných programoch (true, false)

## Formát XML

Keď poznáme URL adresu, na ktorej sa nám vygeneruje náš XML súbor, môžeme pristúpiť k jeho spracovaniu. Obsah XML súboru potrebujeme načítať do premennej *$xml*. V PHP od verzie 5 na parsovanie XML existuje knižnica SimpleXML, ktorá poskytuje veľmi jednoduchú syntax. Použijeme funkciu na parsovanie xml súboru „*simplexml\_load\_file()*“

<?

 $request\_url = "http://...."; // adresa xml súboru

 $xml = simplexml\_load\_file($request\_url) or die("Chyba!");

?>

Ak nám táto možnosť nefunguje, pravdepodobne máme na serveri zakázané načítavanie súborov z iných serverov. Buď teda toto obmedzenie zrušíme, alebo v rámci bezpečnosti prídeme na iný spôsob. Je tu možnosť xml súbor ukladať ručne na vlastný server, prípadne automaticky pomocou CRONu v časových intervaloch. Týmto spôsobom však nemusíme mať vždy preberané aktuálne dáta. Ďalšia z možností je použiť nástroj PHP - „cURL“. cURL je jednoducho povedané nástroj na prenos dát po protokoloch ako http, ftp a podobne. V predchádzajúcich riadkoch sme popísali formát adresy, ktorú v tomto kroku môžeme použiť. Zápis potom vyzerá približne nasledovanie:

$ch = curl\_init($request\_url); //adresa xml

curl\_setopt($ch, CURLOPT\_HEADER, false);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

$xml\_data = curl\_exec($ch);

Týmto zápisom sme docielili, že obsah xml súboru máme v premennej *$xml\_data*, ktorý môžeme následne pomocou SimpleXML parsovať do premennej *$xml*.

$xml = simplexml\_load\_string($xml\_data);

Samozrejme môžeme, a bolo by vhodné, ešte ošetriť, čo v prípade, že sa nepodarí pripojiť na server, kde je umiestený xml súbor. Takže môžeme pridať podmienky pomocou ktorých vyhodnotíme, či premenná *$xml\_data* nie je prázdna, prípadne či neobsahuje napr. hlášku 404. V týchto prípadoch by sme následne načítali do premennej *$xml*, xml súbor zo zálohy na našom serveri pomocou „*simplexml\_load\_file()*“.

Teraz už máme xml súbor sparsovaný v premennej *$xml* a môžeme pristúpiť k samotnému zobrazovaniu jednotlivých údajov.

Ako vypisovať jednotlivé dáta z xml pomocou SimpleXML, je najlepšie si naštudovať na príkladoch na oficiálnej stránke: *http://sk.php.net/manual/en/simplexml.examples-basic.php*

Napríklad základné informácie o fakulte vypíšeme nasledovným spôsobom:

echo $xml->fakulta->nazov; //vypíše názov

echo "<p>Adresa:<br>".

 $xml->fakulta->ulica."<br>".

 $xml->fakulta->psc.", ".

 $xml->fakulta->mesto."</p>";

echo "<p>Tel: ".$xml->fakulta->tel;

 ...

 ...

## Formát JSON

Podobným spôsobom ako pri XML môžeme získať formát JSON. Podmienkou však je zmena parametru *&export* v URL požiadavky.

$ch = curl\_init($request\_url); //adresa json

curl\_setopt($ch, CURLOPT\_HEADER, false);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

$json\_data = curl\_exec($ch);

Týmito krokmi sme získali iba reťazec string. Na jednoduchšiu prácu ešte výsledok prevedieme do poľa.

$obj = json\_decode($json\_data) or die("Chyba!");

## Spracovanie a výpis údajov

V ďalšom príklade vidíme, ako vypíšeme údaje o konkrétnom študijnom programe. Cyklus nám prebehne celý xml a keď narazí na element so správnym ID študijného programu, ktorý potrebujeme, vypíše potrebné údaje:

foreach ($xml->children() as $node) {

 if ($node['kod'] == $idsp) { //idsp= ID Študijného programu

 echo "<p>Študijný program: ".$node->nazov."</p>";

 echo "<p>Akademický rok: ".$node->{'akademicky-rok'}."</p>";

 echo "<p><b>Forma štúdia: </b>".$node->forma."<br>";

 echo "<b>Metóda štúdia: </b>".$node->metoda."<br>";

 if ($node->anotacia!="")

 echo "<h4>Anotácia:</h4><p>".$node->anotacia."</p>";

 }

}

/\*

 Vhodné by bolo overiť, či vôbec daný element má nejaký obsah

\*/

Rovnakým spôsobom vieme zobraziť dáta z JSON formátu:

foreach ($obj->programy as $program) {

 if ($program->programId == &idsp) {

 echo "<p>Študijný program: ".$program->nazov."</p>";

 echo "<p>Akademický rok: ".$program->akademickyRok."</p>";

 echo "<p><b>Forma štúdia: </b>".$program->forma->nazov."<br>";

 echo "<b>Metóda štúdia: </b>".$program->metoda."<br>";

 if ($program->anotacia!="")

 echo "<h4>Anotácia:</h4><p>".$program->anotacia."</p>";

 }

}

/\*

 Podobne by bolo vhodné overenie

\*/

# CMS a praktické príklady

V tejto časti využijeme vyššie uvedené príklady, na implementáciu do najpoužívanejších CMS systémov. Spôsobov je samozrejme viac. Je tu možnosť napísať vlastný modul pre konkrétne CMS, alebo priame vloženie kódu do stránky atď. Cieľom tohto dokumentu ale nie je opisovať jednotlivé CMS systémy. Použijeme preto spôsob ktorý nemusí byť optimálny a bezpečný, ale z tohto pohľadu je najuniverzálnejší a aplikovateľný na väčšinu CMS. Ide o jednoduchú demonštráciu aplikovania dát z portálu VŠ. Prostredníctvom tzv. „Snippet“-ov, vložíme do stránky kus kódu, bez toho aby sme zásadným spôsobom zasahovali do celého systému. Nie všetky CMS majú podporu takýchto snippet-ov priamo a preto je potrebné doinštalovať modul, ktorý nám takýto kód umožní spustiť.

Všeobecne môžeme na vysvetlenie použiť nasledujúcu schému



Obr.1 Schéma využitia stránky a snippet-u

Princíp je všade rovnaký. Spôsob akým chceme dáta získané z Portálu VŠ spracovať, prípadne naformátovať a zobraziť, určuje snippet, ktorý je obsahom stránky, ktorej sa dožadujeme pri volaní odkazu. Jediným parametrom, ktorý nám určuje aký študijný program sa zobrazí je v našom prípade parameter *$idsp*.

## Implementácia

Ako prvý krok vytvoríme snippet, v ktorom použijeme PHP kód ktorý nám vyhovuje. V príklade budeme pracovať s formátom XML. Použijeme už vyššie spomenuté načítanie dát z Portálu VŠ pomocou knižnice cURL, z adresy URL ktorá volá snippet načítame identifikátor štud. programu a podľa tohto ID zobrazíme obsah samotnej stránky. Veľmi jednoduchý kód by mohol vyzerať nejak nasledovne.

<?php

 //nacitanie dat z portalvs.sk

 $ch = curl\_init("https://www.portalvs.sk/sk/informacie-o-vysokych-

 skolach?export=xml&fakulta=704020000&programy=1");

 curl\_setopt($ch, CURLOPT\_HEADER, false);

 curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

 curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, false);

 $xml\_r = curl\_exec($ch);

 if(curl\_error($ch))

 {

 echo 'error:' . curl\_error($ch);

 }

 if ($xml\_r=="") {

 echo "<span style=\"color: red;\"><small><i>Nepodarilo sa spojit z

 www.portalvs.sk - zobrazené údaje nemusia byť

 aktuálne!!!</i></small>

 </span>";

 $xml = simplexml\_load\_file("/tl\_files/fem/pvs/xml/fem-pvs-data.xml");

 } else {

 /\* vsetko ok; \*/

 if (strpos($xml\_r, "404 Stránka sa nenašla") === false){

 //vsetko ok, nacitaj premennu

 $xml = simplexml\_load\_string($xml\_r);

 } else {

 echo "<span style=\"color: red;\"><small><i>Podarilo sa spojit s

 www.portalvs.sk, avsak údaje sa nepodarilo prebrať.

 Zobrazené údaje nemusia byť aktuálne!!!</i></small>

 </span>";

 $xml = simplexml\_load\_file("/tl\_files/fem/pvs/xml/fem-pvs-

 data.xml");

 }

 }

Pokúsili sme sa načítať súbor a v prípade zlyhania načítame lokálne uložený. V ďalšom kroku načítame pomocou metódy *$\_GET* id programu a výsledok zobrazíme ako štandardný výstup.

 $idsp = $\_GET["idp"]; // url+ ?idp=cislo

 foreach ($xml->children() as $node) {

 $arr = $node->attributes();

 //if ((string) $arr == $idsp) {

 if ($arr['kod'] == $idsp) {

 echo "<h3>Štúdijný program: ".$node->nazov."</h3>";

 echo "<h5>Akademický rok: ".$node->{'akademicky-rok'}."</h5>";

 echo "<p><b>Forma štúdia: </b>".$node->forma."<br>";

 echo "<b>Stupeň štúdia: </b>1.<br>";

 if ($node->forma!="denná/externá") {

 echo "<b>Dĺžka štúdia v rokoch: </b>3<br>";

 } else {

 echo "<b>Dĺžka štúdia v rokoch: </b>3 / 3<br>";

 }

 echo "<b>Metóda štúdia: </b>".$node->metoda."<br>";

 echo "<b>Udelený akademický titul po absolvovaní študijného

 programu: </b>Bc.<br>";

 if ($node->{'jazyky-studia'}!="")

 {

 $langs = "";

 echo "<b>Informácie o vyučovacích jazykoch: </b>";

 foreach ($node->{'jazyky-studia'}->{'jazyk-studia'} as $language)

 $langs .= $language.", ";

 }

 echo substr($langs, 0, -2)."<br></p>";

 if ($node->anotacia!="")

 echo "<h4>Anotácia:</h4><p>".$node->anotacia."</p>";

 echo "<h4>Uplatnenie absolventov:</h4><p>".

 $node->uplatnenie."</p>";

 }

 }

?>

Tento kód je z dôvodu jednoduchosti samozrejme skrátený, taktiež sú vynechané štýly a JavaScript. Je však funkčný a použiteľný ako ukážka implementácie. V našom prípade sme rovnaký kód pre snippet použili pre všetky CMS.

Ako spoločný krok pri takomto postupe ešte možno uviesť nastavenie alias-u stránky. Ide o akési unikátne meno stránky. Toto meno je možné použiť pri volaní stránky obsahujúcu snippet. V tomto kroku ale nejde o nevyhnutnosť ale skôr o prehľadnosť. Ako sa už k takémuto aliasu stránky dostať je individuálne a spôsobov je mnoho. Ako z obrázku vyššie vidno, môže to byť linka alebo položka v menu. Príklad externého odkazu:

http://www.stranka-fakulty.sk/portal-vs-reader/?idp=11521

Pozn.: Správanie po zadaní nesprávneho, alebo žiadneho identifikátora je individuálne. Je na autorovi samotnej stránky ako ošetrí takéto situácie.

Ďalšie kroky sú už ale pre každé CMS individuálne. Preto v krátkosti uvádzame postup ako je možné tento kód aplikovať pre každé CMS jednotlivo, spolu s výsledným efektom v prehliadači používateľa.

## Drupal

Drupal umožňuje vytvoriť snippet priamo a nie je potrebné inštalovať žiadne doplnky. Stačí len povoliť modul „PHP Filter“. Po tomto kroku nám pribudne nový typ obsahu, resp. formát „PHP Code“. Ten nám umožní vytvoriť snippet v podobe obyčajnej stránky.

Obsah tejto stránky je vlastne kód ktorý sme vytvorili. Ten sa postará o zobrazenie spracovaných dát.

Na takto vytvorenú stránku už stačí len odkazovať a ako parameter zadať id študijného programu. V CMS Drupal by takýto odkaz, štandardne bez použitia alias-u vyzeral približne takto:

*http://nasa.drupal.stranka/node/2/?idp=11521*



 Obr.2 Ukážka použitia dát z portálu VŠ v CMS Drupal

## Joomla

Joomla nemá možnosť spúšťať snippety priamo, preto je potrebné nájsť vhodný doplnok. My použijeme „Regular Labs Snippets“. V manažmente tohto komponentu vytvoríme nový snippet a ako obsah použijeme náš PHP kód. K samotnému komponentu sa nedá pristupovať priamo, ale je potrebné ho priradiť k stránke. To znamená, že stačí vytvoriť obyčajnú stránku a ako jej obsah nastaviť *{snippet 1}*. Číslo určuje id snippet-u, keďže tých môže byť viac samozrejme.

Následne už stačí vytvorenú stránku so snippet-om otvoriť, samozrejme so zadaným id programu. Pre Joomla by štandardná adresa vyzerala takto:

*http://nasa.joomla.stranka/index.php/portal-vs-reader/?idp=11521*



Obr.3 Ukážka použitia dát z portálu VŠ v CMS Joomla

## WordPress

Podobne ako Joomla ani Wordpress nemá priamu podporu snippetov, čo ale jednoducho vyriešime použitím doplnku „PHP Code For Posts“. Ďalej je postup takmer identický ako v predchádzajúcom prípade. V manažmente doplnku vytvoríme nový snippet s našim PHP kódom a ten vložíme ako obsah stránky, resp. článku.

Potom už len zavoláme odkaz na stránku spolu s id programu:

http://nasa.wordpress.stranka/2016/10/05/portal-vs-reader/?idp=11521



Obr.4 Ukážka použitia dát z portálu VŠ v CMS WordPress

## Contao

Na záver ešte ukážka ako to vyzerá v praxi, na stránke FEM SPU v Nitre. Web fakulty je postavený na CMS Contao a využíva podobné princípy ako sú popísané v tomto dokumente:



Obr.5 Stránka FEM

Obsah

[Úvod 1](#_Toc469905995)

[Stručne o Portáli VŠ 1](#_Toc469905996)

[Stručne o XML 1](#_Toc469905997)

[Parsovanie XML 2](#_Toc469905998)

[Stručne o JSON 2](#_Toc469905999)

[JSON vs. XML 3](#_Toc469906000)

[JSON 3](#_Toc469906001)

[XML 3](#_Toc469906002)

[Príklad výstupu XML z PVS: 4](#_Toc469906003)

[Príklad výstupu JSON z PVS: 5](#_Toc469906004)

[Preberanie dát z Portálu VŠ pre potreby FEM SPU Nitra 6](#_Toc469906005)

[Formát URL adresy 6](#_Toc469906006)

[Parametre URL adresy 6](#_Toc469906007)

[Formát XML 6](#_Toc469906008)

[Formát JSON 7](#_Toc469906009)

[Spracovanie a výpis údajov 8](#_Toc469906010)

[CMS a praktické príklady 9](#_Toc469906011)

[Implementácia 9](#_Toc469906012)

[Drupal 12](#_Toc469906013)

[Joomla 13](#_Toc469906014)

[WordPress 14](#_Toc469906015)

[Contao 15](#_Toc469906016)

[Slovník pojmov 17](#_Toc469906017)

[Literatúra 18](#_Toc469906018)

# Slovník pojmov

**CMS** – Content Management System, resp. redakčný systém

**Cron** – nástroj, plánovač úloh

**HTML** – Hypertext Markup Language

**JSON** – JavaScript Object Notation

**Parsovanie** – v počítačovej terminológii analyzovanie string reťazca a jeho ďalšie spracovanie, ako napr. vytvorenie dátovej štruktúry nad ktorou možno vytvárať operácie

**Snippet** – kus kódu, vykonávajúci určitú funkciu v rámci programu

**URL** – Uniform Resource Locator

**XML** – Extensible Markup Language

# Literatúra

What is PHP? In: php. [online], [cit. 2012-04-20]. Dostupné na Internete: <http://www.php.net/>

XML Tutorial. In: w3schools.com. [online], [cit. 2012-04-20]. Dostupné na Internete: <www.w3schools.com/xml/>

Introducing JSON. In: www.json.org. [online], [cit. 2016-11-30]. Dostupné na internete: <http://www.json.org/>

JSON – Introduction. In: w3schools.com. [online], [cit. 2016-11-30]. Dostupné na internete: <http://www.w3schools.com/js/js\_json\_intro.asp>

ŠUSTÁK, Róbert. 2012. Preberanie dát z Portálu VŠ pomocou XML a PHP : celoškolský odborný seminár s medzinárodnou účasťou Nitra, 23. máj 2012 - Slovenská poľnohospodárska univerzita v Nitre