Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Aplikace v prostředí World Wide Web

Katolický David

© 2012 ČZU v Praze

! ! !

Místo této strany vložíte zadání diplomové práce.
(Do jedné vazby originál a do druhé kopii)

! ! !

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Aplikace v prostředí World Wide Web" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 5.4.2012 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Pavlu Šimkovi za vedení diplomové práce.

Aplikace v prostředí World Wide Web

Application in World Wide Web Environment

Souhrn

V diplomové práci jsou charakterizovány technologie tvorby webových stránek. Práce je zaměřena na skriptovací jazyk PHP především v souvislosti se zpracováním XML souborů. Dále jsou v teoretické části charakterizovány vybrané rozhraní pro komunikaci webových služeb dostupné v jazyce PHP a náležitosti, které mají XML soubory splňovat. V praktické části je analýza nabídky poskytovatelů monitoringových online aplikací a také návrh a vývoj nové aplikace monitorující dostupnost protokolů a služeb. V závěru práce jsou shrnuty výhody a nevýhody nové aplikace a porovnání s již existujícími aplikacemi.

Summary

This thesis describes technologies for creating webpages. It's focused on familiarizing with scripting language PHP, primarily in relation to processing of XML files. Furthermore, theoretical part describes some interfaces for web services communication available in PHP language and requirements that XML files must comply. Practical part contains analysis of suppliers providing with online monitoring applications and also describes development of new application for monitoring the availability of protocols nad services. The conclusion summarizes advantages and disadvantages of new application and its comparison with already existing ones.

Klíčová slova: webová aplikace, monitoring, php, php-cli, skript, protokoly, služby, online

Keywords: web application, monitoring, php, php-cli, script, protocols, services, online

Obsah

1. Úvod

Internet a internetové technologie se stále více rozšiřují. Nejdříve se využívalo pouze statických HTML stránek, které měly za úkol návštěvníky informovat o určitém tématu. Po nějakém čase se stalo pravidlem, že webové stránky neměly již pouze statický obsah, ale obsah byl zobrazován dynamicky za pomoci skriptovacích jazyků.

Pro tvorbu webových prezentací a aplikací v současné době stále převažují dva hlavní jazyky. Prvním z nich je pravděpodobně nejvyužívanější skriptovací jazyk PHP. Skripty jsou prováděny na straně serveru, kde server zpracuje požadavek, vytvoří HTML výstup a následně ho pošle klientskému prohlížeči, který obsah zobrazí. Velikou výhodou je, že PHP je nezávislé na platformě. Používané skripty, až na pár malých výjimek, fungují na mnoha různých operačních systémech. Další velikou výhodou je, že PHP umožňuje využívání mnoha knihoven, které se dají použít pro nejrůznější účely, jako jsou např. práce se soubory, práce s většinou databázových systémů, zpracování textu nebo grafiky atd.

Druhým jazykem je, společností Microsoft vyvíjený, produkt ASP (Active Server Pages). V dnešní době se využívá spíše jeho nástupce ASP.NET, který využívá framework .NET. Avšak na trhu jsou i jiné. Webové aplikace se v dnešní době také programují např. v Pythonu, případně v Ruby on Rails, ale tyto technologie zatím nejsou příliš využívány.

Společnosti také začínají ve velké míře využívat moderní Cloud computing a jiné online formy outsourcingu. Tyto technologie mají zpravidla někde v internetu své servery, na kterých serverové aplikace fungují a klienti se k nim připojují pomocí takzvaných tenkých klientů, které jsou většinou reprezentovány webovým rozhraním. V moderní době je také trend využívat Cloud služby, nejen firmami, ale i obyčejnými lidmi. A to především z hlediska pohodlnosti a dostupnosti těchto služeb odkudkoli na internetu. Těmto serverům provozovatelé běžně garantují určitou dostupnost. Takovými službami jsou například emailová schránka, online datové úložiště, aplikace společnosti Google atd.

2. Cíl práce a metodika

Cílem diplomové práce je návrh a následné vytvoření serverové aplikace s webovým rozhraním pro testování dostupnosti vybraných služeb internetu. Tato webová aplikace by měla být schopna monitorovat služby, které jsou z pohledu uživatele podstatné. Jedná se mimo jiné o odezvu serveru, dostupnost služby v daném čase, pravidelné monitorování dostupnosti v určitém časovém intervalu, ukládání jednotlivých naměřených hodnot do databáze a následné generování statistik a grafů dostupnosti. Dále je pak důležité informovat uživatele, který monitoring provádí, o výpadcích nebo nedostupnosti monitorovaných služeb dle jeho zadaných kritérií, a to minimálně formou emailových zpráv.

Aplikace bude kromě výše uvedených pravidelných služeb monitoringu nabízet také jednorázový test dostupnosti jakékoli služby na určité adrese v internetu a na konkrétním portu. Tato služba bude pouze informovat, zdali je služba v daný okamžik dostupná, či nikoli.

Další cíle této práce jsou pak analýza konkurence a srovnání poskytovaných služeb, výhod a nevýhod a následné porovnání s vytvořenou aplikací.

Metodikou práce je sběr informací z dostupných zdrojů, které byly použity pro teoretický základ diplomové práce.

V první kapitole je popsána stručná charakteristika skriptovacího jazyka PHP formuláře, dále je popsán šablonovací nástroj Smarty, odesílání pošty, autentizace uživatelů a v poslední řadě pak komunikační rozhraní SOAP a XML_RPC.

V dalších kapitolách teoretické časti práce je popsána syntaxe, pravidla XML dokumentů a jednotlivé druhy parserů pro zpracování, respektive vytváření XML dokumentů.

Tyto zdroje byly použity pro získání požadovaných znalostí potřebných pro zhotovení vlastní aplikace. V praktické části diplomové práce je popsán návrh vývoje aplikace v jazyce PHP, databáze pro použití v relační databázi MySQL, vývoj vlastní aplikace, její funkce, popis jednotlivých částí a její uživatelská a administrátorská část. Aplikace využívá primárně operačního systému linux, démona CRON pro automatické spouštění procesů, serveru PHP 5.3 a webového serveru

Apache 2 pro administrační rozhraní. Dále pak využívá zmíněného databázového systému MySQL.

Testování protokolů a služeb probíhá v pravidelných intervalech. Z těchto měření jsou výsledky o dostupnosti ukládány do databáze a následně tvořeny statistiky a grafy dostupnosti v administraci. Ty jsou k dispozici ve webové části aplikace. Pro vyhodnocení dostupnosti monitorovaných služeb je použita statistická veličina aritmetického průměru, která určuje celkovou dostupnost dané služby za určité období.

Na základě syntézy teoretických poznatků a výsledků praktické části práce jsou formulovány závěry diplomové práce.

3. Teoretická východiska

3.1. Skriptovací jazyk PHP

„Označení PHP bylo původně zkratkou anglické fráze „Personal Home Page“. Tuto technologii vytvořil v roce 1994 Rasmus Lerdorf kvůli sledování návštěvníků svých stránek. S postupným nárůstem užitečnosti a možností využití této technologie se ujal nový název „PHP: Hypertext Preprocessor“.“[1]

PHP je vloženým skriptovacím jazykem. Pokud je řečeno, že PHP je vložen do HTML, znamená to, že jej lze interpretovat přímo v kódu HTML, díky čemuž je vývoj dynamických webových prezentací snáze dostupný.

„PHP není programovací jazyk, je pouze skriptovací. Jazyk PHP je navržen, aby vykonal určitou činnost jako reakci na výskyt určité události – například, když uživatel odešle vyplněný formulář nebo přejde na určitou adresu URL.“[1]

Pro jazyk PHP existuje velké množství Frameworků, akcelerátorů, rozšíření, které znatelně programátorům ulehčují práci a řeší za ně automaticky některé činnosti, jako například ověřování uživatelů, zabezpečení proti SQL Inject, snadnou správu vzhledů atp. Mezi frameworky lze zařadit určitě dva nejznámější. Tím jsou bezpochyby ZEND framework a Nette framework. Frameworky jako takové v této práci popisovány z důvodu rozsahu nebudou. Mezi další možnosti rozšíření patří asi nejrozšířenější šablonovací nástroj Smarty, pomůcka pro ověřování formulářů QuickForm, odesílání emailové korespondence a autentizace uživatelů. Dalšími důležitými nástroji v jazyce PHP jsou různé komunikační rozhraní. V práci budou zmíněny dva, v současné době nejpropracovanější. Těmi jsou komunikační rozhraní XML_RPC a rozhraní SOAP.

3.1.1. Funkce v PHP

Funkce v jakémkoli programovacím jazyce se používají ke srozumitelnému rozdělení kódu na menší funkční bloky. *„Každý takový blok vykonává na základě zadaných parametrů určitou činnost, případně vypočítá nějakou hodnotu. Může však také uložit např. údaje do souboru nebo databáze.“*[2] *„Dá se shrnout, že funkce je vlastně blok kódu, který lze jednou nadefinovat a pak jej vyvolávat z ostatních částí programu.“*[3]

[1] ULLMAN, Larry. PHP a MySQL, názorný průvodce tvorbou dynamických WWW stránek, s. 12
[2] LACKO, Luboslav. PHP a MySQL Hotová řešení, s. 51
[3] CASTAGNETTO, Jesus a spol. PHP Programujeme profesionálně, První vydání,s. 117

3.1.2. Formuláře s QuickForm

Formulář lze v PHP, respektive HTML, vytvořit několika způsoby. Buď je možné ho vytvořit přímo v jazyce HTML jeden prvek po druhém, nebo využít nějakého nástroje. Tímto nástrojem je například modul HTML_QuickForm. Funguje tak, že se volají jednotlivé metody, které vytvoří strukturu formuláře. Po vytvoření kompletního formuláře se jediným příkazem vypíše celý formulář najednou. QuickForm zároveň zachovává výchozí hodnoty při odesílání formuláře, zachycuje chyby při vyplňování formuláře, zajistí jednotný vzhled všech formulářů a usnadní také práci se soubory.

Vytvoření a zobrazení formuláře

V modulu QuickForm se formuláře vytvářejí odlišným způsobem než v HTML. Nejdříve se specifikují prvky formuláře v tom pořadí, v jakém se mají objevit, a teprve následně se nechá vypsat celý formulář najednou.

Ověřovací pravidla

Jednotlivým prvkům lze přiřadit ověřovací pravidla. U textového elementu jsou to například ověřování pomocí regulárních výrazů, zdali vyplněná položka je číslo, email, pouze písmena atp. Dále je možnost zvolit u každého elementu, zdali je povinný nebo volitelný a jeho minimální či maximální délka. U elementu file pro upload souborů prostřednictvím formuláře je to pak ještě ověření mimetype nahrávaného souboru, ověření, zdali soubor není větší než zadaná velikost, případně zdali název souboru vyhovuje regulérnímu výrazu.[4]

[4] SKLAR, David. PHP 5 moduly, rozšíření, akcelerátory, s. 73-98

```
$form = new HTML_QuickForm('send_email');
$form->addElement('text',subject','Předmět:',"size:'30'
maxlength:'128'");
//Předmět musí být alespoň 5 znaků dlouhý.
$form->addRule('subject','Zadejte platný předmět','minlength',5);
$form->addElement('submit','send','Odeslat zprávu');
$data = $form->getSubmitValues();
If($data['send']) {
 //provedou se ukony po odeslani formulare
}else{
 //v případě že formulář nemohl být odeslán, nebo se právě zobrazil,
pak se opět vypíše
 $form->display();
}
```

Zdrojový kód 1: Příklad použití formuláře QuickForm

V QuickForm lze definovat i vlastní ověřovací pravidla pomocí metody registerRule(), které se jako parametr předá název vlastní vytvořené funkce. Tu lze poté opět aplikovat na elementy v podobě metody addRule(), kde se jako poslední parametr uvede název registrované funkce.[5]

Filtry

V QuickForm je možnost aplikovat některé předdefinované filtry na formulář ještě před odesláním formuláře. Nejčastěji využívaný filtr je trim, který ořízne mezery před a za zadaným textem do elementu. Toto lze jednoduchým způsobem aplikovat na jednotlivé elementy, případně zadáním řetězce __ALL__, který zajistí aplikaci na všechny elementy. Stejně jako u ověřování, nabízí modul QuickForm vytváření vlastních filtrů. Ty se vytvářejí ve formě funkce, která se pak předává metodě applyFilter() jako druhý parametr. [5]

3.1.3. Šablony Smarty

Jedním ze všeobecných základních principů programování je oddělit prezentační a logickou vrstvu aplikace. V PHP to dlouhou dobu nebylo zvykem a vždy byl promíchán kód aplikace s prezentační vrstvou – vzhledem. Toto promíchání přinášelo problémy zejména při úpravách nebo změnách uživatelského rozhraní aplikace. Tyto dvě vrstvy je tedy zapotřebí oddělit z důvodu, aby kodér pracující s grafikou nemusel

[5] SKLAR, David. PHP 5 moduly, rozšíření, akcelerátory, s. 99 - 102

být programátor. Mimo komplexních frameworků, jako jsou například zmíněné Zend nebo Nette, které plně podporují oddělení vrstev (tzv. MVC – Model-View-Controller), existují ještě další možnosti. Jednou z nich je například Smarty šablonovací nástroj. Zajišťuje oddělení prezentační vrstvy od logické. Kodér tudíž může bez problémů měnit vzhled aplikace a přitom nemusí pracovat se zdrojovým kódem. Další výhodou šablonovacích nástrojů je to, že pokud daná aplikace používá několik grafických rozhraní, je velice jednoduché vyrábět další a další bez jakéhokoli zásahu do zdrojového kódu.

Smarty šablonovací nástroj byl vyvinut autory Andrei Zmievski a Monte Orte. Je to v současnosti nejpopulárnější šablonovací nástroj pro PHP a je nepochybně nejmocnější. Velikou výhodou je, že byl vydán pod licencí LGPL. Uživatelé tak mají zaručen veliký stupeň flexibility, pokud jde o modifikace a redistribuce softwaru. Nespornou výhodou také je, že se může používat zdarma. Mezi hlavní schopnosti Smarty šablon patří tyto:

- **Vyspělá prezentační logika.** Smarty nabízí konstrukce, které umožňují vyhodnocování založené na podmínkách, případně iterační zpracování dat. Přestože je sám o sobě jazykem, syntaxe je taková, že ji designér rychle zvládne, i když nemá předběžné programátorské znalosti.
- **Kompilace Šablony.** Aby se eliminovaly nadměrné režijní náklady při realizaci, převádí Smarty standardně své šablony do srovnatelných skriptů PHP, což vede při následných voláních k mnohem rychlejší realizaci. Smarty je také natolik inteligentní, že překompiluje šablonu, pokud se její obsah změnil.
- **Cachování šablon (caching).** Smarty nabízí volitelnou možnost skladovat šablony v souborech. Liší se od kompilace v tom, že zapnete-li cachování, zabráníte dokonce tomu, aby se daná logika vykonávala; realizuje se jen uskladněný obsah. Je možné vyznačit dobu života uskladněných dokumentů, řekněme na pět minut. Během této doby se tak nebudou provádět databázové dotazy vztahující se k vytvoření obsahu šablony.
- **Vysoce konfigurovatelný a rozšiřitelný**. Objektově orientovaná architektura Smarty umožňuje modifikovat a rozšiřovat jeho výchozí chování. Původním návrhářským cílem bylo, aby byl dobře konfigurovatelný, a tím se uživatelům

nabídla značná flexibilita, především co se týká přizpůsobování chování Smarty. Děje se to prostřednictvím zabudovaných metod a atributů.

- **Zabezpečení**. Smarty nabízí řadu schopností, jejichž účelem je odstínit data serveru i data aplikace před poškozením kódu ze strany designéra, ať už záměrným nebo jiným. [6]

Smarty je mocný nástroj i z pohledu Cachování. Při použití smarty šablonovacího nástroje lze velice jednoduše – za pomoci nastavení jedné proměnné $smarty->caching – zapnout pro danou stránku cachování. Defaultní čas cachování se smarty je jedna hodina. To znamená, že po otevření dané stránky se stránka uloží jako statická a při následném otevření této stránky se stránka znovu nevytváří pomocí PHP, ale pouze se načte již uložená statická forma dané stránky. Tuto dobu lze opět velice jednoduše ovlivnit pomocí nastavení proměnné $smarty->cache_lifetime.[6] Takto lze nastavit dobu cachování jak je pro danou stránku potřeba. Důvod cachovat je především pro ušetření výkonu při běhu aplikace a urychlení distribuce obsahu ze serveru na clienta. Například, pro zpravodajský web, který přidává na své stránky nějaké novinky cca 1x denně, je zbytečné zatěžovat databázi a server sestavením úvodní stránky pro každého návštěvníka, když pravděpodobnost, že zrovna přibyla nějaká novinka, je minimální. Pro tento příklad je typické použít cachování s defaultním nastavením jedné hodiny. Na druhou stranu je třeba u zpravodajských serverů typu idnes.cz nepřípustné čekat na aktualizaci zpráv několik desítek minut. Ale i zde existuje způsob, jak použít cachování. Lze to například dělat tak, že opět bude zapnuté cachování, avšak při přidání nějaké novinky nastane vynucené obnovení cachovaných stránek. Tímto se dosáhne aktuálnosti výpisu novinek při použití cachování z důvodu odlehčení zátěže serveru. Samozřejmě existují případy, kdy se cachování absolutně nehodí a výkon se musí řešit na straně serveru pomocí nástrojů loadbalancing či jiných nástrojů pro rozložení výkonu. Takový případ může nastat například u pokladních systémů, kde cachování z důvodu úspory výkonu není možné.

[6] GILMORE, Jason W. Velká kniha PHP 5 MySQL: kompendium znalostí pro začátečníky i profesionály, s. 408

3.1.4. XML_RPC

„*XML-RPC umožňuje volat proceduru na vzdáleném počítači (RPC je zkratka Remote Procedure Call). V programech PHP můžete být klientem XML-RPC (voláte funkce na jiných počítačích), i serverem XML-RPC (spouštíte funkce jako odpověď na vzdálené požadavky). XML-RPC dociluje vzdálených volání procedur tak, že předává zprávy XML přes připojení HTTP. Klient XML-RPC zabalí do dokumentu XML informace o proceduře, kterou chce zavolat, i argumenty pro proceduru. Pak předá dokument XML nějakému serveru XML-RPC přes normální požadavek POST HTTP. Server XML-RPC se podívá do dokumentu XML, zavolá patřičnou místní funkci, a vrátí jako odpověď na požadavek HTTP výsledky v podobě jiného dokumentu XML.*“[7]

XML_RPC se zaměřuje na to, aby byl relativně jednoduchý na použití a to jak na straně klienta, tak na straně serveru. Není tak jemný ani silný jako SOAP, ale pro menší aplikace je svou jednoduchostí a univerzálností zcela vyhovující. Komunikační rozhraní XML_RPC má svoji velice podrobnou dokumentaci dostupnou na adrese http://www.xmlrpc.com. XML_RPC oproti jiným službám v PHP má předem definované skalární typy. Jsou jimi int, boolean, string, souble, dateTime, iso8601 a base64. V zasílaném XML souboru, který má být zpracován, musí být definováno, o jaký skalární typ se jedná, a to tak, že je daná hodnota navíc obklopena ještě výše uvedenou značkou. [8]

```
<value><int>15</int></value>
<value><string>text</string></value>
```

Zdrojový kód 2: Popis XML souboru pro použití v XML_RPC

Při vytváření objektu XML_RPC_Value se předává této třídě jako první argument řetězec, jako druhy argument typ.

```
$jmeno = new XML_RPC_Value('David','string');
$vek = new XML_RPC_Value(23,'int');
```

Zdrojový kód 3: Metoda vytváření objektu XML_RPC_VALUE

[7] SKLAR, David. PHP 5 moduly, rozšíření, akcelerátory, s.179
[8] CHOW, Shu-Wai. Programujeme Mashup aplikace pro Web 2.0 v PHP, s. 19 - 40
PHP: Hypertext Preprocessor [online]. [cit. 2012-02-21]. Dostupné z: http://www.php.net

V případě, že je potřeba přes toto rozhraní zasílat například obrázky nebo jakákoli jiná binární data, je zapotřebí tato data zaslat zakódovaná do base64 pomocí funkce base64_encode().

Volání procedur XML_RPC na jiném serveru

Pro volání vzdálených procedur slouží dvě třídy. Jsou to třídy XML_RPC_Client a XM_RPC_Message.

„*Objekt XML_RPC_Client obsahuje informace o serveru, který se kontaktuje. Objekt XML_RPC_ Message obsahuje informace o proceduře, která se má volat, a o jejích argumentech. Konstruktor XML_RPC_C1ient přebírá dva argumenty: cestu a název serveru:*“[9]

```
$client = new XML_RPC_Client('/RPC2', 'time.xmlrpc.com');
```

Zdrojový kód 4: Vytvoření nového XML_RPC klienta

„*To znamená, že se požadavky XML-RPC budou odesílat metodou POST na URL http://time.xmlrpc.com/RPC2. Jeli zapotřebí používat jiný port než je výchozí (80), předá se číslo portu do konstruktoru jako třetí argument.*“ [9]

Nyní se musí vytvořit zpráva, která se odešlete na server XML-RPC. Jakmile se zpráva odešle na server XML-RPC, zavolá se procedura currentTime.getCurrentTime():

```
$message = new XML_RPC_Message('currentTime.getCurrentTime');
```

Zdrojový kód 5: Vytvoření XML_RPC zprávy, která se bude zasílat na server

Zpráva se odešle na server tak, že objekt zprávy je předán metodě send() klienta:

```
$response = $client->send($message);
```

Zdrojový kód 6: Odeslání zprávy na server

Pokud se procedura vykonala úspěšně, bude obsahovat návratové hodnoty. Pro získání návratové hodnoty se zavolá metoda value () odpovědi:

```
$return_value = $response->value();
```

Zdrojový kód 7: Získání návratové hodnoty po odeslání zprávy na server

V případě, že se vyskytla nějaká chyba, bude obsahovat $response informace o chybě. [9]

[9] SKLAR, David. PHP 5 moduly, rozšíření, akcelerátory, s. 185

3.1.5. SOAP

Koncept webových služeb umožňuje volat funkce na vzdáleném serveru pomocí protokolu HTTP. Vzdálenému serveru se předávají parametry, ten na základě těchto parametrů vyhodnocuje výsledky, se kterými pak dále pracuje. Je přitom úplně jedno, v jakém programovacím jazyku je webová služba napsaná i z jakého jazyka ji voláme. Rozdíly jsou kompenzovány formátem výměny dat, kterým je obvykle XML. Jakýkoliv program schopný pracovat s XML daty tedy může být klientem i serverem webových služeb.[10]

SOAP protokol je úspěšným nástupcem XML-RPC. Po zaregistrování do služby, kterou je potřeba využívat, je většinou možné ihned začít. Každý SOAP protokol musí obsahovat takzvaný wsdl dokument (Web Services Description Language), kde je definováno, co daná webová služba nabízí, jaké obsahuje funkce, a způsob, jakými se tyto funkce volají a jaké se jí předávají parametry. Je zapsána ve formátu XML. V použitém programovacím jazyce se tedy vytvoří SOAP klient, kterému se přiřadí dané wsdl XML. Tímto SOAP klient ví, jak funkce využívat a na programátorovi je už pouze to, předat tomuto rozhraní potřebné parametry k definovaným funkcím a zpracovávat návratové XML.

3.1.6. Odesílání pošty

Pro odesílání emailové pošty prostřednictvím jazyka PHP lze využít několik možností. V PHP existuje funkce mail(), která zajistí, prostřednictvím webového serveru odeslání emailové zprávy. Takto odesílaná zpráva pak používá syntaxi:

```
mail(komu, předmět, zpráva);
```

Zdrojový kód 8: Syntaxe funkce mail()

Ovšem v současnosti by takto zaslanou zprávu většina provozovatelů e-mailových služeb označila jako spam. Proto se musí do funkce zahrnout i určité informace, které jsou obsaženy v hlavičce e-mailu, aby zaslané e-maily dorazily v pořádku[6] „*Pro tento případ použijeme poslední nepovinný parametr funkce a tím je zaslání speciálních hlaviček. Skript pro odeslání emailu bude vypadat takto:*“[11]

[10] VRÁNA, Jakub. Webové služby v PHP: XML-RPC a SOAP. Root.cz [online]. 10.8.2007[cit. 2012-03-12]. Dostupné z: http://www.root.cz/clanky/webove-sluzby-php-xmlrpc-soap/
[11] MACH, Jakub. PHP pro úplné začátečníky, s. 138

```
<?PHP
$to = “test@seznam.cz“;
$from = “cilova_adresa@spolecnost.cz“;
$subject = “Předmět emailu“;
$message = “Text emailové zprávy“;
$headers = “From: David Katolický <dkatolicky@gmail.com> \n“;
mail($to,$subject,$message,$headers);
?>
```

Zdrojový kód 9: Syntaxe funkce mail () s hlavičkou from

3.1.7. Autentizace uživatelů

„*Ověřování totožnosti uživatelů neboli autentizace je v současných webových aplikacích běžnou praxí. Dělá se nejen kvůli bezpečnosti, ale také proto, aby se daly nabídnout schopnosti, které jsou přizpůsobené preferencím a typu uživatele. Obvykle je uživatel vyzván, aby zadal uživatelské jméno a heslo. Oba údaje uvažované dohromady formují pro daného uživatele jednoznačnou identifikační hodnotu (jeho "doklady", neboli „průkaz totožnosti“).*“[12]

Jak již bylo řečeno, autentizace může probíhat na základě uživatelského jména a hesla. Bezpečnější autentizace (například v bankách) může probíhat na základě nějakého dalšího ověření. V dnešní době je to především prostřednictvím zasílání autorizačních SMS zpráv. Nejbezpečnější metoda autentizace je prostřednictvím certifikátu. Kombinace certifikátu a přístupového hesla se dá považovat za jednu z nejbezpečnějších forem autentizace. Pokud bychom uvedli příklad opět v bankovnictví, kde je bezpečnost transakcí na prvním místě, pak některé banky dokonce pro velké klienty, především firmy, vydávají čipové karty, na kterých je certifikát přímo nahrán. Nedochází tudíž k manipulaci s certifikátem jako souborem v počítači a odpadá další riziko odcizení tohoto certifikátu. Bez této karty není možné provádět žádné operace.[13]

[12] GILMORE, Jason W. Velká kniha PHP 5 MySQL: kompendium znalostí pro začátečníky i profesionály, s. 291

[13] MĚCHÁČEK, Jaroslav. Autentizace na webu. In: UVT MU Zpravodaj [online]. 1999 [cit. 2012-03-13]. Dostupné z: http://www.ics.muni.cz/bulletin/articles/173.html

V PHP je autentizace dostupná pomocí několika způsobů. Jednou z možností je využít přímo jednoho z modulů apache, kdy se přihlašování provádí prostřednictvím souboru .htaccess a .htpassword. Toto ověření uživatele pak probíhá prostřednictvím http hlaviček. Tento způsob je však již delší dobu zastaralý a používá se velmi výjimečně. Běžným způsobem v dnešní době je přihlašování prostřednictvím nějakého formuláře přímo na webových stránkách.[14]

„*Implementovat jednoduchou kontrolu přístupu není těžké. Následující příklad má tři možné výstupy jak se zobrazí. Pokud se soubor načte bez parametrů, zobrazí se HTML formulář, který od uživatele žádá, aby zadal své uživatelské jméno a heslo. Pokud jsou parametry zadány, ale nejsou správné, zobrazí se chybová hláška. Pokud jsou parametry zadány správně, zobrazí se tajný obsah.*“[15]

```
<?PHP
$name = $HTTP_POST_VARS[‘name‘];
$psw = $HTTP_POST_VARS[‘psw‘];

If (empty($name) || empty($psw))
{
?>
<h1>Přihlašte se prosím</h1>
<form Metod=“post“ action=““>
Uživatelské jméno: <input type=“text“ name=“name“><br/>
Heslo: <input type=“password“ name=“psw“><br/>
<input type=“submit“ value=“Přihlaš“>
</form>
<?PHP
}
else if($name==‘user‘ && $psw==‘pass‘)
{
Echo “Přihlášení proběhlo úspěšně“;
}
else
{ Echo “Bylo zadáno špatné uživatelské jméno a heslo.“; }
```

Zdrojový kód 10: Kontrola proměnných zaslaných prostřednictvím formuláře

Tomuto příkladu se však dají v praxi vytknout minimálně dvě důležité věci. Ověřování probíhá pouze na základě jednoho uživatelského jména a heslo se porovnává

[14] KOSEK, Jiří. PHP - tvorba interaktivních internetových aplikací: podrobný průvodce, s. 248
[15] WELLING, Luke, THOMSONOVÁ, Laura. *PHP a MySQL – rozvoj webových aplikací, Druhé vydání.* Str. 343, 345

v textové podobě. První problém by se dal vyřešit prostřednictvím databáze. Bylo by potřeba navázat spolupráci s databází a skript patřičně upravit, aby se uživatelská jména a hesla porovnávala s daty uloženými v databázi. Druhý problém je pak možné vyřešit prostřednictvním hashovacích funkcí (např. MD5, SHA1 atp.), případně použít metodu salted, kdy se před hashováním přidá k heslu nějaký řetězec a hash probíhá s tímto řetězcem.

Po úspěšném přihlášení uživatele v některých případech dochází k uložení dat do takzvaných COOKIES proměnných. Tyto proměnné uchovávají informace vázané k dané webové stránce (URL) a lze s nimi pomocí několika funkcí v PHP pracovat. Nejznámější způsob použití je pro zapamatování přihlášeného uživatele.[16]

[16] GUTMANS, Andi a spol. *Mistrovství v PHP 5, První vydání.* Str. 48

3.2. XML

Název XML vzniklo z anglického jazyka Extensible Markup Language. Jazyk XML byl vyvinut konsorciem W3C a byl sestaven na základě předchozích zkušeností se značkovacími jazyky.[17]

XML je ideální formát pro ukládání strukturovaného, nebo semistrukturovaného textu určeného pro šíření a publikaci na celé řadě médií. Jazyk XML obsahuje specifické instrukce nazývané tagy, elementy a entity. Použití XML dokumentů je velice široké. V dnešní době se používá především pro předávání informací mezi počítačovými programy, podnikovými aplikacemi a může také sloužit pro bezproblémový přenos dat mezi různými programovými platformami.[18]

3.2.1. Syntaxe XML

Pro možnost práce s dokumenty XML, je zapotřebí znát, jak tyto dokumenty vypadají. Je tedy zapotřebí znát syntaxi jazyka XML a jeho další rysy. Dále pro potřeby praktické části diplomové práce je zapotřebí vědět, jak se s ním pracuje v PHP.

3.2.1.1. Elementy a struktura dokumentu

„*Každý XML dokument se skládá z elementů. Elementy se v textu vyznačují pomocí tzv. tagů. Většině elementů odpovídají dva tagy - počáteční a koncový.*“[19]

```
<test>Toto je obsah elementu test.</test>
```

Zdrojový kód 11: Ukázka elementu XML

Výše je ukázán příklad, který obsahuje jeden element test. Jeho obsah je vyznačen pomocí tagů <test> (počáteční tag) a </test> (ukončovací tag). [19]

„*Názvy tagů se tedy zapisují mezi znaky „<“ a „>“. Ukončovací tag má před svým názvem ještě znak „/“ aby se odlišil od počátečního.*“ [19]

Některé elementy nemusejí mít žádný obsah. Můžeme je samozřejmě zapisovat tak, že za počátečním tagem uvedeme hned ten koncový.

```
<test>Toto je obsah elementu test.<br></br> A tohle také.</test>
```

Zdrojový kód 12: Ukázka správného použití elementu v XML

[17] MERCER, Dave. Beginning PHP5, s. 313 - 315
[18] BRADLEY, Neil. XML: kompletní průvodce, s. 18
[19] KOSEK, Jiří. PHP a XML, s. 17

To však není příliš pohodlné, a proto lze v XML použít ještě jednu variantu tagu, která říká, že element nemá žádný obsah. Počáteční tag ukončíme dvojicí znaků „/>“ místo pouhého „>“ a koncový tag vynecháme. [20]

```
<test>Toto je obsah elementu test.<br/> A tohle také.</test>
```

Zdrojový kód 13: Ukázka alternativy správného použití elementu v XML

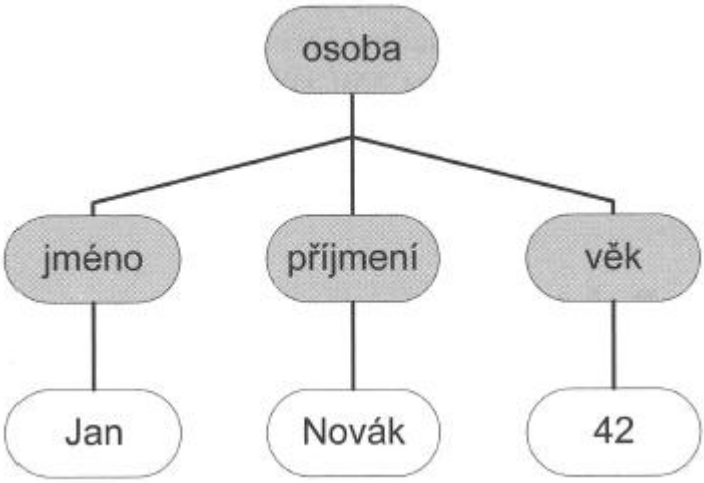
„Každý dokument XML musí obsahovat pro všechny počáteční tagy odpovídající koncový tag, nebo musí být počáteční tag zapsán jako element s prázdným obsahem.“ [20]

3.2.1.2. Datový model dokumentu

„Jak je z příkladů zřejmé, elementy do sebe můžeme zanořovat, takže element může obsahovat další elementy nebo text. Elementy tak vytvářejí hierarchickou stromovou strukturu. Každý dokument XML je možné interpretovat jako strom, jehož jednotlivé uzly odpovídají jednotlivým elementům, případně textu uvnitř elementů.“ [20]

```
<osoba>
      <jméno>Jan</jméno>
      <příjmění>Novák</příjmění>
      <věk>42</věk>
</osoba>
```

Zdrojový kód 14: Datový model XML v kódu



Obrázek 1: Datový model dokumentu XML (Zdroj: KOSEK, Jiří. PHP a XML, s. 19)

[20] KOSEK, Jiří. PHP a XML, s. 18-19

„*Uzly odpovídající textovému obsahu elementů jsou ve stromu vždy na nejnižší úrovni listů a už na ně nemohou být navěšeny žádné další uzly. V případě, že má element tzv. smíšený obsah, jsou jeho dětmi ve stromové reprezentaci jak textové uzly, tak uzly odpovídající elementům.*“[21]

Popsanému stromovému modelu dokumentu XML se říká infoset. Tento abstraktní datový model říká, že dokument XML je stromová struktura složená z jednotlivých uzlů. Uzly se dělí na několik typů (elementy, textové uzly, atributy, komentáře, instrukce pro zpracování, jmenné prostory a další). U každého uzlu pak infoset definuje několik jeho vlastností jako jméno, rodiče, seznam dětí apod. Na infosetu je tak založena většina jazyků a rozhraní, které jsou vystaveny nad XML. Je to pochopitelné, protože při práci s dokumentem XML je většinou podstatná jeho struktura a obsah zachycený v elementech, a infoset nabízí právě tento pohled na dokument XML. [21]

3.2.1.3. Atributy

„*Elementy jsou základním prostředkem pro členění informací uvnitř dokumentu XML. Kromě elementů lze pro zachycení informací využít atributy. Atributy se vždy zapisují k počátečnímu tagu elementu.*“ [21]

```
<odstavec zabezpečení="důvěrné">Nějaká tajná informace.</odstavec>
```

Zdrojový kód 15: Ukázka vložení atributu do elementu XML

„*V uvedené ukázce je atributu zabezpečení přiřazena hodnota důvěrné. Hodnotu atributu je nutné vždy uzavřít do uvozovek nebo do apostrofů. U jednoho tagu lze použít více atributů najednou, stačí je oddělit mezerou.*“ [21]

```
<odstavec zabezpečení="důvěrné" autor="Jan Novák">Nějaká tajná
                    informace.</ostavec>
```

Zdrojový kód 16: Ukázka vložení více atributů do elementu XML

U jednoho elementu přitom nesmí být použity dva atributy se shodným názvem.

[21] KOSEK, Jiří. PHP a XML, s. 18-20

3.2.1.4. Názvy elementů a atributů

Jazyk XML je (na rozdíl například od HTML) citlivé na velikost písmen. Počáteční a koncový tag se proto musí shodovat i ve velikosti písmen. Níže uvedený element je zapsaný chybně, protože jeho počáteční a koncový tag si neodpovídají. [22]

```
<NÁZEV>Tento element je zapsán špatně.</název>
```

Zdrojový kód 17: Ukázka chybně zapsaného elementu

„*Samotná jména elementů a atributů mohou přitom být vytvářena poměrně volně. První znak jména musí být písmeno nebo podtržítko, další znaky mohou navíc obsahovat i čísla, tečku a pomlčku. Písmena přitom mohou být i z jiné než anglické abecedy. Jména elementů a atributů tak můžeme psát klidně česky, nebo třeba rusky v azbuce.*“ [22]

3.2.1.5. Deklarace XML

Každý dokument XML by měl začínat deklarací XML, ve které je určeno, jakou verzi XML používáme a v jakém kódování je soubor uložen. [22]

```
<?xml version="1.0" encoding="utf-8"?>
<osoba>
	<jméno>Jan</jméno>
	<příjmění>Novák</příjmění>
	<věk>42</věk>
</osoba>
```

Zdrojový kód 18: Ukázka kompletního jednoduchého dokumentu XML

„*Každá aplikace, která podporuje XML, musí umět zpracovat soubor uložený v kódování UTF-8 nebo UTF-16. Proto bychom měli dokumenty XML přednostně ukládat a ostatním posílat v jednom z těchto kódování. V praxi se přitom častěji používá UTF-8 kvůli lepší kompatibilitě se staršími aplikacemi. Dokumenty je možné ukládat i v jiných kódováních, ale pak musíme toto kódování povinně určit v deklaraci XML a nemáme jistotu, že tento dokument zvládnou zpracovat všechny aplikace.*“ [22]

V případě, že je zapotřebí do dokumentu vložit nějaký znak, který buď není snadno dostupný na klávesnici, nebo nejde reprezentovat v použitém kódování, je možné do dokumentu vložit odkaz na číselný kód znaku v Unicode. [22] Předchozí ukázku dokumentu XML lze tedy zapsat i takto.

[22] KOSEK, Jiří. PHP a XML, s. 22

```
<?xml version="1.0" encoding="us-ascii"?>
<osoba>
      <jméno>Jan</jméno>
      <příjmění>Nov&#xE1;k</příjmění>
      <věk>42</věk>
</osoba>
```

Zdrojový kód 19: Ukázka kompletního jednoduchého dokumentu XML s použitím vkládání znaku v Unicode

3.2.1.6. Jmenné prostory

„Jedním ze základních cílů jazyka XML je poskytnout aplikacím formát, ve kterém půjde vyměňovat informace po celém světě. Pro dosažení tohoto úkolu je však potřeba zajistit, aby byly elementy používané v dokumentech jednoznačně identifikované a navzájem rozlišitelné. Jinak nebude například možnost rozlišit, zda element název popisuje název knihy v katalogu knihkupectví, nebo obchodní název firmy ve výpisu z obchodního rejstříku, nebo ještě něco úplně jiného. Nutnost jednoznačného rozlišení elementů je důležitá v těch případech, kdy není přesně známo, jaké informace zpracovávaný dokument obsahuje, nebo se zpracovává komponovaný dokument, který obsahuje elementy z několika různých oblastí.“[23]

„Problém jednoznačné identifikace elementů v dokumentech XML řeší jmenné prostory. Používáme-li v dokumentu XML jmenné prostory, není už element jednoznačně identifikován jen svým jménem, ale kombinaci jména a jmenného prostoru. Jmenný prostor má přitom podobu adresy URI, která zajišťuje možnost celosvětově vytvářet nová URI a přitom zachovat jejich unikátnost. Důležité je uvědomit si, že URI adresa v tomto případě slouží jen jako identifikátor, aplikace pracující s XML se nikdy nesnaží z této adresy získat nějaký dokument.

Pro rozlišení dvou dříve zmíněných významů elementu název tak můžeme použít dva různé jmenné prostory. V následující fiktivní syntaxi doplníme před názvy elementů URI jmenného prostoru.“ [23]

```
<{http://knihkupectvi.cz/katalog}název>Čuk a
Gek</{http://knihkupectvi.cz/katalog}název>
<{http://justice.cz/ns/or}název>Grada</{http://justice.cz/ns/or}název>
```

Zdrojový kód 20: Ukázka použití jmenných prostorů v XML

[23] KOSEK, Jiří. PHP a XML, s. 25-26

Je patrné, že kombinace URI jmenného prostoru a název elementu je teď už jednoznačná a je možné odlišit, kdy se o jaký název jedná. Zároveň je však zřetelné, že výše uvedený zápis by byl velmi nepohodlný a zdlouhavý. Ve skutečnosti se ani nejedná o syntaxi, která by fungovala. Jednalo se pouze o demonstraci principu. V dokumentech XML se pro usnadnění zápisu příslušnosti elementu do nějakého jmenného prostoru používá jedna z následujících dvou syntaxí.[24]

- „*První možností je použití implicitního (výchozího) jmenného prostoru. Chceme-li, aby nějaký element a všichni jeho potomci (tj. elementy v něm obsažené) patřily do nějakého jmenného prostoru, stačí, když u tohoto elementu nadeklarujeme požadovaný jmenný prostor jako implicitní pomocí atributu xmlns. Následující ukázka je dokument v jazyce XHTML, kde všechny elementy patři do jmenného prostoru http://www.w3.org/1999/ xhtml.*“[25]

```
<?xml version="l.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
      <head>
            <title>Ukázka XHTML stránky</title>
      </head>
<body>
      <hl>Ukázka XHTML stránky</hl>
      <p>Všechny elementy v tomto dokumentu patří do jmenného prostoru
XHTML.</p>
</body>
</html>
```

Zdrojový kód 21: Použití implicitního jmenného prostoru

- „*Druhá varianta spočívá v deklaraci prefixu, který zastupuje zvolený jmenný prostor. Tento prefix se pak zapisuje před jména všech elementů patřících do jmenného prostoru. Deklarace prefixu jmenného prostoru se provádí pomocí atributu ve speciálním tvaru.*“ [24]

```
xmlns:«prefix»="«URI»"
```

Zdrojový kód 22: Deklarace prefixu

[24] KOSEK, Jiří. PHP a XML, s. 25-27
[25] MARCHAL, Benoit. XML v příkladech, s. 110 - 113

Ukázkový XHTML dokument proto lze zapsat také následujícím způsobem.

```
<?xml version="1.0" encoding="UTF-8"?>
<html:html xmlns:html="http://www.w3.org/1999/xhtml">
<html:head>
      <html:title>Ukázka XHTML stránky</html:title>
</html:head>
<html:body>
      <html:h1>Ukázka XHTML stránky</html:h1>
      <html:p>Všechny elementy v tomto dokumentu patří do jmenného
prostoru XHTML.</html:p>
</html:body>
</html:html>
```

Zdrojový kód 23: Použití jmenného prostoru prostřednictvím prefixu

„Zápisu jména elementu ve tvaru html: title se říká kvalifikované jméno elementu (QName). To se skládá z prefixu (html) a z lokálního jména (title). Kombinace lokálního jména a jmenného prostoru, pro který je prefix deklarován, společně jednoznačně identifikuje element. Prefix ovšem může být libovolný, slouží jen jako pomůcka pro zkrácení zápisu.“ [26]

Atributy se v běžných případech do jmenného prostoru neumísťují, a proto se na ně ani nevztahuje implicitní jmenný prostor. Chápe se to tak, že atribut patří vždy k elementu, u kterého je uveden, a tento element už do nějakého jmenného prostoru patří.

[26] KOSEK, Jiří. PHP a XML, s. 27

3.2.1.7. Speciální atributy

„*Jazyk XML a na něj navazující standardy definuje několik univerzálních atributů, které lze použít na libovolném elementu. Všechny tyto atributy jsou globální a patří do jmenného prostoru http://www.w3.org/XML/1998/namespace. Tento jmenný prostor je speciální a nemusí se pro něj deklarovat prefix. Existuje pro něj předdefinovaný prefix xml.*“[27]

Xml:lang

„*Pomocí atributu xml: lang je možnost pro element určit jazyk, v jakém je zapsán jeho obsah. Tuto informaci pak mohou využívat různé aplikace, např. pro správné dělení slov nebo indexování textu.* “[27]

```
<kapitola xml:lang="cs">
      <para>Celá kapitola je česky.</para>
      <para xml:lang="en">This is the only exception because it is in
English.</para>
      <para>Obsah předchozího elmentu byl anglicky.</para>
</kapitola>
```

Zdrojový kód 24: Určení jazyku dokumentu pomocí xml:lang

Jako hodnota atributu se uvádí kód jazyka podle BCP 475. Jazykový kód českého jazyka je cs, pro slovenštinu je sk a pro angličtinu en. Lze používat i třípísmenné kódy, což je pro více exotické jazyky dokonce nutnost.[28]

Xml:space

„*Mnoho jazyků založených na XML - třeba XHTML nebo XSL-FO - vícenásobné výskyty bílých znaků nahrazují jednou mezerou. Jsou ale případy, kdy se nám to nehodí. XML proto nabízí možnost, jak aplikaci předat informaci o tom, že uvnitř elementu se nemají bílé znaky nijak upravovat. Informace se předává tím, že se k elementu přidá atribut xml:space a nastaví se na hodnotu preserve. Druhou možností je nastavit jej na hodnotu default, kdy se pak uplatní výchozí nastavení aplikace pro práci s bílými znaky.*“[27]

[27] KOSEK, Jiří. PHP a XML, s. 39
[28] CASTAGNETTO, Jesus. Programujeme PHP profesionálně, s. 318-320

Xml:id

„*V mnoha případech se hodí, když můžeme element jednoznačně identifikovat. V XML k tomu slouží atribut xml:id, který elementu přiřadí jednoznačný identifikátor. V jednom dokumentu se přitom nemohou vyskytovat dva elementy se stejným identifikátorem. Atribut xml:id tak má podobnou úlohu jako primární klíč v relačních databázích.*

Hodnota atributu je přitom poměrně omezená. Identifikátor musí začínat písmenem nebo podtržítkem, za kterým následují další písmena, číslice, tečky, podtržítka nebo pomlčky.“[29]

```
<kniha>
<název>Ze života hmyzu</název>
<kapitola xml:id="uvod">
      <název>Úvod</název>
</kapitola>
</kniha>
```

Zdrojový kód 25: Ukázka Použití xml:id

„*Na elementy s takto přiřazeným identifikátorem je pak snadné odvolávat se v různých rozhraních a dotazovacích jazycích. Např. rozhraní DOM nabízí metodu getElementByld (), která vrátí element s daným identifikátorem. Podobně se chová i funkce id() v jazyce XPath. Na elementy s identifikátorem je možnost odvolávat se i v Xlnclude.*“[29]

Xml:base

Pomocí atributu xml: base jde změnit základní URL, s kterým se skládají relativní URL uvedená v dokumentu. Předpokládejme, že dokument uložený na adrese http://example.cz/manualy/instalace.xml má následující obsah. [29]

```
<?xml version="1.0" encoding="UTF-8"?>
<manual xmlns:xi="http://www.w3.org/2001/XInclude">
<nazev>Instalační příručka</nazev>
<xi:include href="kapitoly/uvod.xml"/>
<xi:include href="kapitoly/prvni_instalace.xml"/>
<xi:include href="kapitoly/upgrade.xml"/>
</manual>
```

Zdrojový kód 26: Ukázka bez použití xml:base

[29] KOSEK, Jiří. PHP a XML, s. 40

„*Elementy Xlnclude ukazují na jednotlivé kapitoly. Aby je mohl parser načíst, musí však pro jednotlivé kapitoly znát jejich absolutní URL adresu. Tu získá tak, že relativní adresy z atributu href složí se základní adresou dokumentu http://example.org/manualy/instalace.xml. Po složení adres získáme následující absolutní adresy jednotlivých kapitol:* “[30]

```
http://example.cz/manualy/kapitoly/uvod.xml
http://example.cz/manualy/kapitoly/prvni_instalace.xml
http://example.cz/manualy/kapitoly/upgrade.xml
```

Zdrojový kód 27: Výsledný formát bez použití xml:base

V dokumentu XML bylo zapotřebí u každého relativního odkazu opakovat adresář „kapitoly“, ve kterém byly jednotlivé kapitoly umístěné. Obejít to lze právě pomocí atributu xml:base. Tento atribut umožňuje změnit základní URL pro odkazy uvedené v elementu s tímto atributem. Nové základní URL vznikne složením dosavadního základního URL s adresou uvedenou v xml:base. [30]

```
<?xml version="1.0" encoding="UTF-8"?>
<manual xmlns:xi="http://www.w3.org/2001/XInclude"
xml:base="kapitoly/">
<nazev>Instalační příručka</nazev>
<xi:include href="uvod.xml"/>
<xi:include href="prvni_instalace.xml"/>
<xi:include href="upgrade.xml"/>
</manual>
```

Zdrojový kód 28: Ukázka s použitím xml:base

“*Atribut xml: base nalézá uplatnění nejen ve spojení s elementy Xlnclude, ale obecně s jakýmikoliv elementy, které vytvářejí nějaký druh odkazů - např. XLink, katalogové soubory, apod.*” [30]

[30] KOSEK, Jiří. PHP a XML, s. 40-41

3.2.2. SimpleXML

Knihovna SimpleXML se hodí pro načítání jednoduchých dokumentů XML. Tato knihovna funguje tak, že v paměti vytvoří strukturu objektů, která odpovídá souboru XML. Tato knihovna ovšem má podstatná omezení, a to především v oblasti jmenných prostorů a omezení podpory smíšeného obsahu. Dokumenty lze upravovat jen ve velmi omezené míře. Lze říci, že knihovna SimpleXML se hodí pro jednodušší XML dokumenty. Pokud je zapotřebí zpracovávat složitější XML soubory a následně s nimi pracovat, je zapotřebí využít jiné alternativy – např. DOM. [31]

3.2.2.1. Načtení dokumentu

„*SimpleXML umí dokumenty XML načítat buď ze souboru, nebo z řetězce. Slouží k tomu funkce simplexml_load_file() a simplexml_load_string(). Obě vrací novou instanci třídy SimpleXMLElement, která reprezentuje kořenový element právě načteného dokumentu XML.*

Parsování dat z řetězce se hodí v případech, kdy dokument XML vzniká až během provádění skriptu - jedná se např. o data z formuláře nebo z databáze.

Funkce simplexml_open_file() je však mnohem univerzálnější. Díky nové I/O architektuře PHP lze v jakékoliv funkci původně určené pro práci se soubory používat jakýkoliv jiný vstupní (případně výstupní) proud. Proudy jsou přitom abstrakcí, která definuje jednotné rozhraní pro práci se vstupy a výstupy. PHP podporuje několik druhů proudů - běžné soubory, zdroje dostupné přes protokoly HTTP/HTTPS/FTP/FTPS, komprimované proudy a další. Není tak problém načítat dokumenty XML přímo z webového serveru.“ [31]

Poměrně často jsou dokumenty XML veliké, protože obsahují veliké množství dat. Při přenášení po síti nebo přes internet je zapotřebí přenášená data co nejvíce eliminovat. Proto se dokumenty XML při ukládání komprimují, aby nebyly tak veliké. Například obrázky ve formátu SVG se často ukládají do souborů s příponou .svgz, aby se naznačilo, že se jedná o původní XML zkomprimované metodou zlib. Díky proudům je možno takový dokument přímo načítat pomocí SimpleXML.

[31] KOSEK, Jiří. PHP a XML, s. 87-88

```
$xml = simplexml_load_file("compress.zlib://../data/soustava.svgz");
echo $xml->title; // vypíše řetězec, který je obsažen v elementu title
```

Zdrojový kód 29: Načtení komprimovaného XML dokumentu pomocí SimpleXML

„Proudy nabízejí opravdu bohaté možnosti a navíc si můžeme vytvářet vlastní filtry, které mohou číst dokument například z databáze nebo nějakého systému pro správu obsahu. Fantazii se meze nekladou.„ [32]

3.2.2.2. Čtení hodnot

„Jak bylo již uvedeno, dokument XML je v knihovně SimpleXML reprezentován jako několik objektů typu SimpleXMLElement. Vývojáři PHP se snažili, aby tyto objekty měly co nejjednodušší rozhraní. Kvůli tomu se však objekt někdy chová jako objekt, někdy jako řetězec a někdy jako pole. Je potřeba dát si pozor, jak s danou částí pracovat.“[32]

```
<?xml version="1.0" encoding="utf-8"?>
<faktura cislo="12/2000" vystaveni="2.2.2000" splatnost="16.2.2000">
 <odberatel>
  <nazev>Poučená, a.s.</nazev>
  <adresa>Široká 21, Praha 1, 110 00</adresa>
  <ico>098765432K/ico>
  <dic>CZ0 987 65432 K/dic>
 </odberatel>
 <dodavatel>
  <nazev>XMLCompany, s.r.o.</nazev>
  <adresa>Dlouhá 12, Praha 1, 110 00</adresa>
  <ico>1234567890</ico>
  <dic>CZ1234567890</dic>
 </dodavatel>
 <polozka>
  <popis>Analýza nasazení XML v podnikovém IS</popis>
  <cena dph="5">50000</cena>
 </polozka>
 <polozka>
  <popis>XML Editor - 10 licencí</popis>
  <cena dph="5">128956</cena>
 </polozka>
 <polozka>
  <popis>Notebook microMini</popis>
  <cena dph="19">89500</cena>
```

[32] KOSEK, Jiří. PHP a XML, s. 88-90

```
 </polozka>
</faktura>
```

Zdrojový kód 30: Ukázka XML dokumentu obsahující elementy i atributy

„Budeme nyní předpokládat, že jsme si do paměti pomocí SimpleXML načetli dokument s fakturou a máme jej uložený v proměnné $xml. Tato proměnná teď zastupuje kořenový element dokumentu faktura. Jednotlivé podelementy jsou dostupné jako členské proměnné, které vrací opět instance třídy SimpleXMLElement. Jméno odběratele faktury můžeme snadno zjistit např. pomocí zápisu $xml->odberatel->nazev. Tento zápis sice nevrací řetězec, ale instanci třídy SimpleXMLElement, v mnoha kontextech se však konverze na řetězec provede automaticky. Je tedy možné jméno odběratele vypsat pomocí:“[33]

```
echo $xml->odberatel->nazev;
```

Zdrojový kód 31: Získání názvu odběratele z dokumentu XML

„Tato metoda přístupu k podelementům je jednoduchá, ale má své limity. Prvním je přístup k elementům, jejichž jméno obsahuje znaky nedovolené v identifikátorech jazyka PHP. Takovým znakem je například ‚-‘, který se však v názvech elementů XML používá poměrně často.“ [33] V takovémto případě by výše uvedené volání objektu bylo syntakticky nesprávné, a tudíž by bylo zapotřebí využít možnosti dynamické kontroly identifikátoru za běhu skriptu.

V případě, že je na stejné úrovni více elementů se stejným názvem, tento princip přestává fungovat. Ve výše uvedeném případě se jedná konkrétně o element ‚polozka‘. Při práci $xml->polozka vrátí SimpleXML pouze první prvek. Pokud je potřeba získat informace o druhém prvku, pak je zapotřebí pracovat s objektem $xml->polozka[1]. Ovšem, jak zjistit kolik těchto elementů je v XML definováno? Bohužel zde funkce count pro zjištění počtu nefunguje. Je tedy zapotřebí použít cyklus foreach, který zajistí projití všech elementů.

[33] KOSEK, Jiří. PHP a XML, s. 90-91

```
foreach ($xml->polozka as $polozka)
{
  echo $polozka;
}
```

Zdrojový kód 32: Projití všech elementů „položka“ v XML

„Zatím popsané způsoby čtení dokumentu XML předpokládaly, že jsou známa jména elementů, které se v dokumentu vyskytují. Pokud jména nejsou předem známa, je zde možnost použití metody children (), která vrací všechny podelementy. Ke zpracování nalezených elementů je opět nejlepší použít foreach:“ [34]

```
foreach ($xml->odberatel->children() as $nazev => $element)
{
   echo "Element $nazev obsahuje hodnotu $element.<br>\n";
}
```

Zdrojový kód 33: Projití dokumentu XML a získání všech potomků elementu odběratele

„Kromě podelementů může být u každého elementu uvedeno několik atributů. Ty jsou opět dostupné dvěma způsoby. Jednak je možné s každým objektem SimpleXMLElement pracovat jako s asociativním polem, které je indexované názvy atributů.“ [34]

```
echo "Číslo faktury: " . $xml["cislo"];
echo "Sazba DPH první položky: " . $xml->polozka[0]->cena["dph"];
```

Zdrojový kód 34: Získání atributů z XML

Pokud názvy atributů nejsou dopředu známy, pak je možné využít stejně jako u elementů metody attributes(), pomocí které lze cyklem foreach opět získat všechny atributy.

[34] KOSEK, Jiří. PHP a XML, s. 91

3.2.3. SAX

SAX je rozhraní pro práci s XML soubory, které funguje odlišně než SimpleXML. Hlavní rozdíl v tomto rozhraní je, že čte vstupní dokument XML postupně a aplikaci předává data jako proud událostí. Díky tomu je toto rozhraní rychlé a paměťově nenáročné. Na druhou stranu je použití rozhraní SAX náročnější, než jakékoli jiné, a proto se vyplatí pouze v případě, že jsou zpracovávány velmi velké soubory.

3.2.3.1. Události

„*Při čtení dokumentu XML parser používající rozhraní SAX generuje proud událostí, které odpovídají jednotlivým částem dokumentu XML a které jsou dále předávány aplikaci.*“[35] Existuje několik událostí.

Začátek elementu

„Událost je generována pro každý počáteční tag elementu. Pro obsluhu události musí být definována funkce, která má následující parametry:“ [35]

```
function startElement(«parser», «název», «atributy»)
```

Zdrojový kód 35: Začátek elementu SAX

„*Všechny funkce pro obsluhu události jako první parametr dostanou identifikátor parseru, který se právě používá pro čtení. V parametru «název» je předáno jméno elementu, který právě začal. Parametr «atributy» obsahuje asociativní pole atributů.*

Kvůli starému návrhu SAX rozhraní v PHP je podpora jmenných prostorů velmi špatná.“ [35]

```
<db:title xmlns:db="http://docbook.org/ns/docbook">Úvod</db:title>
```

Zdrojový kód 36: Použití jmenných prostorů v SAX

„*V parametru «název» jako hodnotu dostaneme kvalifikované jméno elementu db:title, což práci příliš neusnadní, protože nevíme, jakému jmennému prostoru prefix db odpovídá.*

[35] KOSEK, Jiří. PHP a XML, s. 105-106

V takových případech je proto potřeba parser vytvořit pomocí funkce xml_parser_create_ns () místo xml_parser_create (). Prefix je pak nahrazen URI adresou jmenného prostoru. O rozebrání tohoto identifikátoru na lokální jméno elementu a adresu jmenného prostoru je potřeba vyřešit vlastními silami. To je pro praktické použití velmi nepohodlné a PHP zde daleko zaostává za knihovnami určenými pro jiné jazyky, jako je například Java.“[36]

Konec elementu

„*Pro každý koncový tag je rovněž generována událost. Jako parametr je obslužné funkci předáno jméno právě končícího elementu:*“ [36]

```
function endElement(«parser», «název»)
```

Zdrojový kód 37: Konec elementu v SAX

„*Pro jméno elementu přitom platí stejná pravidla jako u funkce pro obsluhu začátku elementu.*“ [36]

Znaková data

„*Veškerý text uvnitř elementů (tj. v datovém modelu XML se jedná o textové uzly) je předáván pomocí další události. Její obslužná funkce očekává následující parametry:*“ [36]

```
function characters(«parser», «data»)
```

Zdrojový kód 38: Vytvoření obsahu elementu předáním dat funkci characters

„*V parametru «data» pak lze získat textový obsah elementů. Text je přitom předáván v kódování UTF-8. Stejným způsobem jsou hlášeny i bílé znaky - jako obyčejný text.*“ [36]

Toto jsou jen některé z událostí, které SAX podporuje. Zmíněná metoda je jednou z nejstarších, které existují. Rozsáhlost podporovaných funkcí je relativně vysoká, avšak v PHP ne úplně optimálně řešená. Proto jsou v této práci uvedeny pouze základní události.

[36] KOSEK, Jiří. PHP a XML, s. 106

3.2.3.2. Čtení dat

Jak již bylo uvedeno, rozhraní SAX funguje na proudovém zpracování dokumentu. Tomuto principu se někdy říká push model. Zpracování dat je řízeno parserem, který volá jednotlivé definované funkce předávající data určitým definovaným způsobem cílové aplikaci. Pak již stačí předávat parseru data ve formátu XML a on zařídí vše potřebné.[37]

Spuštění parseru se provádí tak, že se zavolá funkce xml_parse(). Parametry pro tuto funkci jsou data pro zpracování a informace, zdali jde o poslední data nebo ne. V případě, že se parseru sdělí, že je již konec dokumentu, zakončí parsování a ohlásí případné chyby. [37]

[37] KOSEK, Jiří. PHP a XML, s. 110-111

3.2.4. DOM

Rozhraní DOM je možné prohlásit za asi nejlepší pro práci s XML v PHP. Toto rozhraní funguje tak, že celý XML dokument se načte do paměti, zpracuje, a pak lze s tímto souborem pracovat jako s objekty. S nimi pak lze pracovat pomocí různých metod. Například je takto možné zjišťovat nebo měnit obsah.[38]

Toto rozhraní lze zároveň použít i pro parsování jiného obsahu. Stačí, aby splňoval určité podmínky. Ideálně se také hodí například pro zpracování XHTML dokumentu a získávání dat z webových stránek.

3.2.4.1. Objektová reprezentace dokumentu

„*Rozhraní DOM modeluje dokument v paměti také jako stromovou strukturu objektů. Objekty přitom opět odpovídají jednotlivým částem dokumentu XML, jako jsou elementy, atributy, textové uzly apod. Na rozdíl od infosetu, rozhraní DOM věrněji odráží způsob, jakým byl dokument v souboru uložen, takže se například odlišuje normální textový uzel od textového uzlu zapsaného pomocí sekce CDATA. Jednotlivé uzly stromu tvoří instance tříd jako DOMDocument, DOMElement, DOMText nebo DOMAttr. Protože mnoho metod a vlastností těchto objektů je stejných, jsou všechny odvozeny od společného předka DOMNode.*“[38]

3.2.4.2. Načtení dokumentu

Stejně jako v jiných rozhraních, je i v rozhraní DOM zapotřebí nejdříve načíst data XML. Po načtení XML dat vytvoří rozhraní DOM celý strom dokumentu a teprve poté s ním lze pracovat. Nejjednodušší je načíst strom z existujícího dokumentu pomocí metody load.[38]

```
$doc = new DOMDocument();
$doc->load("dokument.xml");
```

Zdrojový kód 39: Načtení dokumentu DOM z XML

[38] KOSEK, Jiří. PHP a XML, s. 123-124

Rozhraní DOM umí zpracovávat i dokument, který není uložen v souboru. Například, potřebujeme-li zpracovat nějaký formulář, lze předat parseru DOM data jako textový řetězec a on je zpracuje v DOM reprezentaci.[39]

```
$doc = new DOMDocument();
$doc->loadXML("
<článek>
 <autor>Pepa</autor>
 <název>Bez názvu</název>
</článek>
")
```

Zdrojový kód 40: Načtení dokumentu DOM z řetězce

3.2.4.3. Čtení dokumentu

Informace o uzlu

„XML dokument je v paměti reprezentován jako DOM strom složený z jednotlivých uzlů. Tyto uzly odpovídají jednotlivým částem dokumentu XML, jako jsou elementy, atributy, textové uzly a komentáře. Uzly jsou v paměti reprezentovány jako objekty. Všechny tyto objekty přitom mají společného předka DOMNode. Metody a vlastnosti této třídy jsou tak dostupné na všech uzlech stromu. Pro některé druhy uzlů pak specifická třída nabízí nějaké metody a vlastnosti navíc.“[39]

[39] KOSEK, Jiří. PHP a XML, s. 125-127

O každém uzlu ve stromu lze zjistit určité podstatné údaje jako je jeho typ, zda je to element, atribut nebo textový řetězec. Tyto údaje lze zjistit pomocí metoty DOMDocument::noteType(). Ta následně vrací jednu z hodnot uvedenou v tabulce.

Konstanta	Hodnota	Popis
XML_ELEMENT_NODE	1	Uzel je element
XML_ATTRIBUTE_NODE	2	Uzel je atribut
XML_TEXT_NODE	3	Uzel je textový
XML_CDATA_SECTION_NODE	4	Uzel s CDATA sekcí
XML_ENTITY_REF_NODE	5	Uzel je odkaz na entitu
XML_ENTITY_NODE	6	Uzel je entita
XML_PI_NODE	7	Uzel je instrukce pro zpracování
XML_COMMENT_NODE	8	Uzel je komentář
XML_DOCUMENT_NODE	9	Uzel je dokument
XML_DOCUMENT_TYPE_NODE	10	Uzel nese informaci o typu dokumentu
XML_DOCUMENT_FRAG_NODE	11	Uzel je fragment dokumentu
XML_NOTATION_NODE	12	Uzel je definice notace

Tabulka 1: Konstanty s typy uzlů (Zdroj: KOSEK, Jiří. PHP a XML, s. *127)*

Výběr elementu na základě jména

V případě, že nás z XML dokumentu zajímají pouze některé elementy, lze použít metodu DOMDocument::getElementByTagName(). Této metodě se jako parametr pošle název elementu, který je potřeba získat. Metoda následně vrátí seznam všech elementů s daným jménem seřazený tak, jak se vyskytují v dokumentu. V případě, že chceme získat všechny elementy, pak lze metodě jako parametr předat „*“ a ta následně vrátí seznam všech elementů v dokumentu. Výsledek je vždy vrácen jako instance třídy DOMNodeList, se kterou se velice intuitivně pracuje. Úplně nejjednodušší způsob, jak procházet získané elementy, je použití v cyklu foreach.[40]

```
foreach ($doc->getElementsByTagName("pozadovana_polozka") as $polozka)
                                        {…;}
```

Zdrojový kód 41: Výběr elementu na základě jména v DOM

Rozhraní DOM má veliké množství tříd, metod a nástrojů. Vyjmenovávat je všechny v této práci přesahuje její rozsah. Stejným způsobem, jako funguje metoda getElementByTagName, funguje například metoda getElementById. Podobně funguje také metoda na zjištění rodiče a potomků jednotlivých elementů.

[40] KOSEK, Jiří. PHP a XML, s. 131

Čtení atributů

„*Každý uzel, který reprezentuje element, má vlastnost attributes. Ta vrací všechny atributy použité u daného elementu. Využívá se přitom třída DOMNamedNodeMap, která umožňuje k jednotlivým atributům přistupovat pomocí jejich jména. Samotné atributy jsou reprezentovány třídou DOMAttr, která je opět potomkem DOMNode.*“[41]

Length	Počet uzlů v seznamu.
Item(«index»)	Přístup k určitému uzlu v seznamu (číslováno od nuly).
getNamedItem(«název»)	Přečtení uzlu s daným jménem.
getNamedItemNS(«URI jmenného prostoru», «lokalni jmeno»)	Přečtení uzlu s daným lokálním jménem patřícího do určitého jmenného prostoru.

Tabulka 2: Metody třídy DOMNamedNodeMap

Každý atribut je reprezentován pomocí třídy DOMAttr. Pomocí vlastnosti nodeName a nodeValue lze získat jméno a hodnotu atributu. [41]

V případě, že jsou známy názvy atributů, které jsou potřeba z dokumentu získat, pak je využití velice snadné. Lze totiž využít metod getAttribute, respektive getAttributeNS, očekávající jako parametr jméno atributu, který je potřeba získat. V případě, že názvy předem známy nejsou, není to nemožné, ale musí se využít jiných metod ke zjištění existujících atributů v jednotlivých uzlech a následně je všechny přečíst pomocí cyklu foreach.

[41] KOSEK, Jiří. PHP a XML, s. 139-140

3.2.4.4. Modifikace dokumentu

V PHP se rozhraní DOM dá využít nejen ke čtení XML dokumentů, ale také k úpravám existujících dokumentů, případně k tvorbě zcela nových dokumentů.[42]

Vytváření nových uzlů

Pokud je zapotřebí přidat nové uzly do již existujícího dokumentu s existující DOM strukturou, je zapotřebí tyto nové uzly vytvořit jako objekty. DOMDocument nabízí několik metod, které toto zajistí.

createAttribute(«jméno atributu»)	Vytvoří nový uzel atributu.
createAttributeNS(«jmenný prostor», «jméno atributu»)	Vytvoří nový uzel atributu, který patří do zadaného jmenného prostoru.
createCDATASection(«text»)	Vytvoří nový uzel se sekcí CDATA.
createComment(«text»)	Vytvoří nový uzel komentáře.
createDocumentFragment()	Vytvoří nový uzel reprezentující fragment dokumentu XML.
createElement(«jméno elementu», «hodnota»)	Vytvoří nový uzel elementu. Pokud uvedeme druhý, nepovinný parametr «hodnota», napojí se na nově vytvořený uzel automaticky i textový uzel s obsahem «hodnota».
createElementNS(«jmenný prostor», «jméno elementu», «hodnota»)	Vytvoří nový uzel elementu patřícího do zvoleného jmenného prostoru. Pokud uvedeme nepovinný parametr «hodnota», napojí se na nově vytvořený uzel automaticky i textový uzel s obsahem «hodnota».
createEntityReference(«jméno entíty»)	Vytvoří nový uzel s odkazem na entitu.
createProcessinglnstruction(«cíl», «data»)	Vytvoří nový uzel s instrukcí pro zpracování. Druhý parametr «data» je nepovinný.
createTextNode(«text»)	Vytvoří nový textový uzel.

Tabulka 3: Metody třídy DOMDocument pro vytváření nových uzlů

[42] KOSEK, Jiří. PHP a XML, s. 141

Připojování, odpojování a klonování uzlů

„*Nově vytvořené uzly můžeme připojovat do existujícího DOM stromu. Důležité je si uvědomit, jak je strom organizován. Skládá se z uzlů a každý uzel může obsahovat další uzly jako děti - ty jsou reprezentovány jako seznam uzlů u svého rodiče. Při provádění veškerých operací s uzly proto musíme znát rodiče, protože v jeho seznamu dětských uzlů se provádí skutečná manipulace s uzly.*“[43]

K připojování a odpojování uzlů se používají metody appendChild() – připojí uzel na konec rodičovského uzlu a removeChild – tato metoda odpojí uzel z dokumentu, který ale v paměti stále zůstává, takže je možné ho připojit například na jiné místo. [43]

Taktéž lze nějaký uzel zkopírovat na jiné místo. Takzvané klonování uzlů se provádí pomocí metody cloneNode(). Lze ji zavolat na jakémkoli uzlu. Tato funkce navrací kopii klonovaného uzlu. Pokud se metoda zavolá prázdná, zkopíruje se pouze daný element. Pokud se zavolá s parametrem cloneNode(true), potom vrátí kopii klonovaného uzlu včetně jeho potomků.

Uložení dokumentu

„*Modifikovaný dokument je většinou vhodné uložit pro další použití nebo alespoň odeslat do prohlížeče. Pro uložení dokumentu DOM do souboru slouží metoda save(). Jako parametr očekává jméno souboru, kam se má uložit obsah celého stromu DOM.*“ [43]

```
$doc->save("vystup.xml");
```

Zdrojový kód 42: Uložení dokumentu do XML

„*Druhá metoda saveXML() vrátí řetězec obsahující DOM strom serializovaný do XML. Takto získané XML můžeme například odeslat rovnou do prohlížeče.*“ [43]

```
echo $doc->saveXML();
```

Zdrojový kód 43: Vložení obsahu DOM dokumentu do řetězce a následné vypsání

[43] KOSEK, Jiří. PHP a XML, s. 142, 153

3.2.4.5. Zpracování HTML

Jak již bylo dříve zmíněno, DOM rozhraní umožňuje číst i jazyk XHTML, protože je zapisován syntaxí XML. V dnešní době je naprostá většina stránek již tvořena standardem XHTML namísto starého HTML, tudíž už tento druh parseru lze většinou použít na jakékoli webové stránky.

V případě, že by parser měl problém stránku přečíst, nebo v případě, že by cílová webová stránka byla i sémanticky chybně napsaná, existuje možnost využití speciálního parseru obsaženého v knihovně libxml2, který umí přečíst HTML kód a vrátí ho ve tvaru XML dokumentu. Tento parser automaticky doplní chybějící tagy, napraví odlišnosti mezi syntaxí XGML a XML a dokáže dokonce ve většině případů správně opravit v dokumentu například překřížené tagy, což je asi nejběžnější chyba.

„*Třída DOMDocument nabízí metody loadHTML() a loadHTMLFile(), které načtou HTML kód z řetězce, resp. ze souboru a po opravě případných chyb jej vrátí jako DOM strom. Se vzniklým DOM stromem lze dále pracovat jako s jakýmkoliv jiným stromem vzniklým načtením dokumentu XML.*“[44]

3.2.4.6. Další možnosti DOM

DOM rozhraní umí mnohem více, než je v této práci zmíněno. Zmíněny jsou pouze základní funkce, vlastnosti a metody tohoto rozhraní. Umí například také dotazování pomocí jazyka XPath, validování dokumentů, získávání informací z XHTML dokumentů atp.[45]

3.2.5. XPath

XPath je dotazovací technologie, která doplňuje samotný jazyk XML. Pomocí XPath lze zapisovat dotazy, které vyberou určité části dokumentu XML, případně provedou nějaké matematické operace z dat uložených v XML. XPath tvoří základ jazyka XSLT, podle kterého se definuje referenční integrita ve W3C XML schématech, umožňuje rychle najít určité uzly v DOM stromu apod.[46]

[44] KOSEK, Jiří. PHP a XML, s. 157
[45] PHP: Hypertext Preprocessor [online]. [cit. 2012-02-21]. Dostupné z: http://www.php.net
[46] SKONNARD, Aaron a Martin GUDGIN. XML - pohotová referenční příručka: referenční příručka programátora k XML, XPath, XSLT, XML Schema, SOAP a dalším, s. 47 - 50

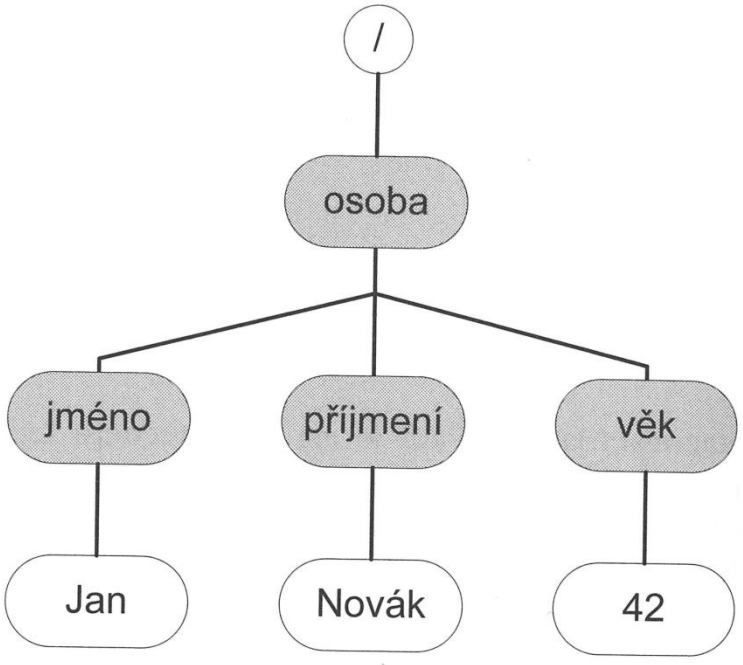
3.2.5.1. Základní struktura výrazu

„V jazyce XPath dotazy zapisujeme jako výrazy, které používají konstrukce jazyka XPath. Nejčastější druh výrazu vybírá z dokumentu určité elementy, případně jiné uzly. Například dotaz „/katalog/polozka“ vybere všechny elementy „polozka“, které jsou uvnitř elementu „katalog“, jenž je kořenovým elementem dokumentu.“[47]

Tomuto výrazu se říká cesta. Funguje tak, že postupně prochází dokument. Jakmile najde element katalog, začne ho prohledávat a hledá element katalog. Jednotlivé části cesty jsou odděleny lomítky.

3.2.5.2. Datový model

Výrazy v jazyce XPath se vyhodnocují nad stromovou reprezentací dokumentu. Ta je velice podobná infosetu – datovému modelu XML. Datový model XPathu však není tak podrobný – například veškeré odkazy na entity se nahradí svým obsahem a tím ztratí informace o případně použité deklaraci typu dokumentu. V datovém modelu XPath jsou jednotlivé části dokumentu XML reprezentovány pomocí uzlů, jak ukazuje i následující obrázek. [47]



Obrázek 2: Datový model XPath (Zdroj: KOSEK, Jiří. PHP a XML, s. 172)

[47] KOSEK, Jiří. PHP a XML, s. 171-172

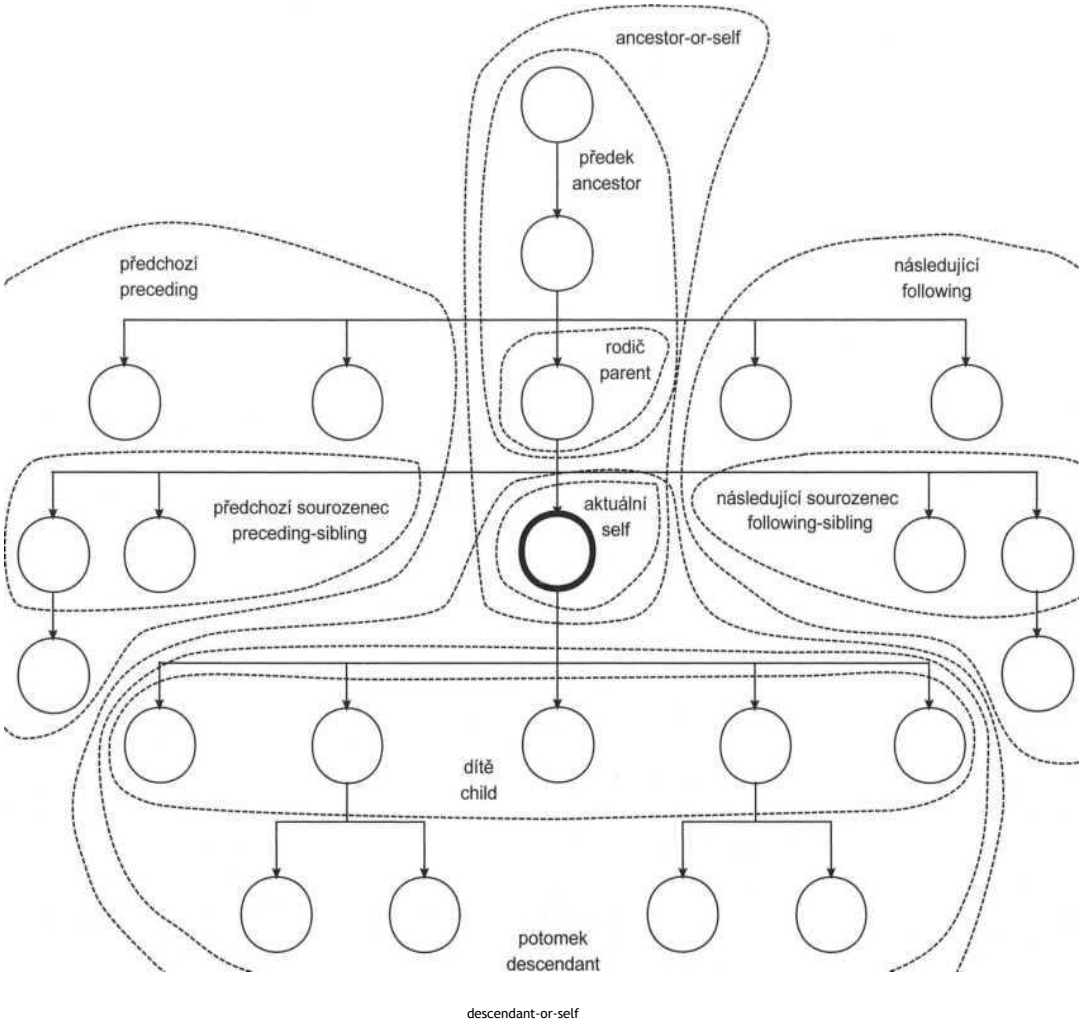
3.2.5.3. Testování výrazů

Testování výrazů se provádí nejsnadněji prostřednictvím testovací utility. Ta je dostupná v Internet Exploreru a v Mozille prostřednictvím JavaScriptové aplikace, kde lze interaktivně testovat XPath výrazy. Výsledek se vždy barevně zobrazí.

V případě, že výraz začíná znakem „/“, vyhodnocuje se od kořenového uzlu. V případě že tímto znakem nezačíná, vyhodnocování začíná v aktuálním uzlu.[48]

3.2.5.4. Identifikátor osy

Každá položka cesty se skládá ze tří částí - identifikátoru osy, testu uzlu a predikátu.



Obrázek 3: Význam nejpoužívanějších os v XPathustr (Zdroj: KOSEK, Jiří. PHP a XML, s. 176)

[48] KOSEK, Jiří. PHP a XML, s. 174

Identifikátor osy určuje, jaká část stromu dokumentu se má prohledávat. V uvedeném dotazu „/katalog/polozka“ je použita zkrácená syntaxe. Ve skutečnosti odpovídá dotazu:[49]

```
/child::katalog/child::polozka
```

Zdrojový kód 44: Popis identifikátoru osy

Zde je identifikátorem osy „child“ a od následujícího uzlu je oddělen znaky „::“. Osa definuje, kde se mají výsledky hledat. V tomto případě se mají prohledávat dětské uzly, které jsou ve stromu dokumentu o jednu úroveň níž. [49]

Lze samozřejmě využívat i jiné osy.

ancestor::	Rodič aktuálního uzlu a všichni jeho další předci až ke kořenu stromu dokumentu. Uzly jsou uspořádány v opačném pořadí než v dokumentu.
ancestor-or-self::	Aktuální uzel a všichni jeho další předci až ke kořenu stromu dokumentu. Uzly jsou uspořádány v opačném pořadí než v dokumentu.
attribute::	Uzly odpovídající všem připojeným atributům, pořadí není nijak pevně definováno.
child::	Všechny děti aktuálního uzlu ve stejném pořadí jako v dokumentu.
descendant::	Všichni potomci aktuálního uzlu ve stejném pořadí jako v dokumentu.
descendant-or-self::	Aktuální uzel a všichni jeho potomci ve stejném pořadí jako v dokumentu.
following::	Všechny uzly následující za koncem aktuálního uzlu ve stejném pořadí jako v dokumentu.
following-sibling::	Všechny následující uzly, které jsou sourozenci aktuálního uzlu, ve stejném pořadí jako v dokumentu.
namespace::	Uzly odpovídající deklarovaným jmenným prostorům, pořadí není nijak pevně definováno.
parent::	Rodič aktuálního uzlu

Tabulka 4: Význam nejpoužívanějších os v XPathu(Zdroj: KOSEK, Jiří. PHP a XML, s. 176)

[49] KOSEK, Jiří. PHP a XML, s. 175

3.2.5.5. Predikáty

„Predikáty umožňují dále filtrovat uzly vybrané pomocí testu uzlu na zvolené ose. Predikát se zapisuje do hranatých závorek a může mít několik podob.

Nejjednodušší variantou predikátu je číslo. To vyjadřuje, kolikátý uzel nás zajímá. Například pro výběr druhé položky katalogu můžeme použít výraz /katalog/polozka[2]. Uzly jsou přitom číslovány od jedničky a pozici posledního uzlu vrací funkce last(). Na poslední položku se tak můžeme odkázat pomocí zápisu /katalog/polozka[last()].

Uzly jsou číslovány v pořadí, v jakém se vyskytují v dokumentu s výjimkou reverzních os (ancestor, ancestor-or-self, preceding a preceding-sibling). Na nich jsou uzly číslovány v opačném pořadí než v dokumentu. Číselná hodnota tak spíše odpovídá pojmu vzdálenost. Například zápis preceding-sibling::polozka[1] vrací nejbližší předcházející položku.

Predikát může obsahovat i jakýkoliv XPath výraz, který vybírá množinu uzlů. Takový výraz se pak testuje pro každý dosud vybraný uzel a tento uzel se stává aktuálním uzlem pro jeho vyhodnocení. Vrací-li výraz alespoň jeden uzel, považuje se predikát za splněný.“[50]

```
//* [cena]
```

Zdrojový kód 45: Použití predikátu v XPath pro výběr elementů, který mají podelement cena – zkrácený tvar

„Výše uvedený dotaz vybere z dokumentu pouze ty elementy, které mají podelement cena. Na tomto místě je užitečné si uvědomit, že v predikátu je uvedený XPath výraz používající zkrácenou syntaxi - v plném tvaru by se jednalo o výraz“ [50]

```
child::cena
```

Zdrojový kód 46: Použití predikátu v XPath pro výběr elementů, který mají podelement cena – plný tvar

„V predikátu se může vyskytovat i libovolná podmínka, tj. výraz, který vrací logickou hodnotu. Do výsledku se pak dostanou pouze ty uzly, pro které je predikát pravdivý. Například pro výběr položek dražších než 10 000 můžeme použít dotaz /katalog/položka[cena > 10000]. Pro výběr položek v kategorii minidisc zase můžeme použít dotaz /katalog/položka[kategorie = 'MiniDisc'].“ [50]

[50] KOSEK, Jiří. PHP a XML, s. 180

4. Analytická část

4.1. Existující poskytovatelé monitoringových služeb

Podobně zaměřených služeb na trhu existuje několik. Zde budou srovnány nabízené služby několika společností zabývajících se monitoringem webových služeb. Tyto společnosti budou analyzovány z hlediska nabízených služeb, poplatků, výhod a nevýhod.

4.1.1. Hyperspin.com

Hyperspin.com je singapurská společnost, která na celosvětovém trhu působí od roku 2002. Jejich klientela se pohybuje v řádech statisíců uživatelů z více než 100 zemí světa. Tato společnost provádí monitorovací služby prostřednictvím jejich šestnácti serverů, které umožňují zjistit, zdali jsou služby dostupné z celého světa. Poskytují své služby ve 33 jazycích, mezi které patří i čeština.

Své služby mají zpoplatněny pevnými měsíčními sazbami a to dle frekvence monitorování

- 1min – 12$ / měsíc
- 2 min – 7$ / měsíc
- 3 min – 5$ / měsíc
- 5 min – 3$ / měsíc
- 10 min – 2,5$ / měsíc
- 15 minut – 2$ / měsíc

Dále tato společnost poskytuje množstevní a dealerské slevy. Dealerská sleva je 15% a množstevní slevy se pohybují od 5% do 30% v závislosti na frekvenci a množství monitorovaných serverů. Další slevy poskytují v případech, že si klient předplatí službu na 3 měsíce (5%), na půl roku (10%) nebo na rok (15%). Dohromady může tedy klient získat slevu až 45%. Navíc při registraci získá nový klient vstupní kredit na poskytování služeb zdarma.

Společnost monitoruje velké množství služeb, a to ping, HTTP, HTTPS, FTP, SSH, SMTP, DNS, POP3, IMAP, MySQL a jakékoliv služby fungující na TCP/IP. Notifikace uživatelů o případném výpadku je prováděna prostřednictvím emailových zpráv, případně SMS.

Výhody

- Dlouhé působení na trhu
- Veliká klientská základna
- Poskytování služby v mnoha jazycích
- Vybudovaná síť serverů, ze kterých se provádí monitoring
- Affilate síť pro prodej služeb s 15% marží

Nevýhody

- Neosobní přístup ke klientům
- Grafické zobrazení historie dostupnosti

4.1.2. Siux.cz

Siux.cz je česká společnost zaměřující se především na monitoring webových stránek. Mezi její služby patří i možnost monitorovat SMTP, PING a jakýkoli socket. V tomto případě probíhá monitoring tak, že skript zjišťuje, zdali na daném socketu naslouchá nějaká služba. Toto je ale spíše okrajová záležitost. Společnost se specializuje na webové stránky takovým způsobem, že monitoruje nejenom dostupnost dané stránky, ale umí monitorovat například, zdali na webové stránce není odkaz na již neexistující web. Testuje také, jestli se na stránce vyskytuje nějaký textový řetězec, či zdali je celý web validní podle validátoru W3C. Umí také ukládat veškeré měření ve formě HTML dokumentu. Zpětně je pak možné si projít jednotlivé HTML výstupy a zjistit, proč se stránky nezobrazovaly. Umí také v případě erroru vytvořit screen aktuální obrazovky tak, jak se zobrazí v prohlížeči.

Tato společnost má ceník řešený dosti rozsáhlým způsobem. Ve zkratce lze říci, že se měření pohybuje od 69,- Kč do tisíců Kč měsíčně.

Zajímavou službou je možnost kontaktovat klienta nejen klasickými metodami jako je email či sms, ale zároveň je zde možnost notifikace i prostřednictvím ICQ. Dalším atypickým parametrem je, že nevyužívají k monitoringu pouze servery v profesionálních serverovnách s páteřním připojením k internetu, ale také servery, které monitorují služby od běžných ISP jako je ADSL od O2, UPC atp.

Výhody

- 16 měřících bodů ve střední Evropě – ČR, SR, Maďarsko
- Různé zdroje měřících bodů (od páteřních linek až po ADSL připojení)
- Různé cenové úrovně
- Možnost volby počtu bodů pro měření dostupnosti
- Veliký výběr možností monitoringu v oblasti webových stránek
- Možnosti komunikace – SMS, mail, ICQ
- Interval měření od 5 sekund
- Jako reference významné portfolio velkých českých firem v čele se společností Seznam.cz

Nevýhody

- Vysoká cena
- Málo měřených služeb
- Příliš složitá konfigurace možností finální služby

4.1.3. Monitis.com

Montis.com je společnost, která nabízí monitoring rozdělený do několika skupin. Nabízí monitoring dostupnosti, doby načítání stránek, Java aplikací, trafiku, zátěže webového serveru atd. Je zaměřena pro široké spektrum uživatelů. Pro developery má možnosti monitoringu například vytíženosti serverů, volných HW prostředků. Pro IT správce má moduly monitoringu eliminujících náklady, rizika, cenu, koupi speciálního softwaru. Pro manažery například univerzální nástroje dostupnosti, počet transakcí, bezpečnost, výskyt problému.

Tato společnost nabízí patnáctidenní zkušební verzi, ve které je zahrnuto dokonce 20 SMS zpráv zdarma. Nabízí i službu zdarma, kde jsou parametry monitoringu 1x za 30 minut, což v jistých případech bohatě postačí. Dále je možno vybírat ze dvou předvolených balíčků, nebo sestavit měřící plán na míru.

Ke každému balíčku je možné přidat notifikace emailem, zpoplatněnými SMS zprávami, ale také prostřednictvím Android aplikace, kde je možnost nahlížet do historie měření pohodlně z mobilního zařízení.

Společnost nabízí, stejně jako společnost Hyperspin, slevy při platbě na delší období. Ceny balíčků jsou ovšem vyšší. Obsahují však určitý počet SMS zpráv, které

jsou v rámci balíčku. Tuto sazbu klient zaplatí i v případě, že nedojde k žádnému výpadku. Tato společnost se tedy vyplatí v případě služby, která je značně nestabilní a má pravidelné výpadky.

Výhody

- Přednastavené balíčky služeb
- Mobilní aplikace
- Testovací doba, ve které je možné vyzkoušet zdarma i zasílání SMS zpráv
- Program zdarma s velice omezenými možnostmi

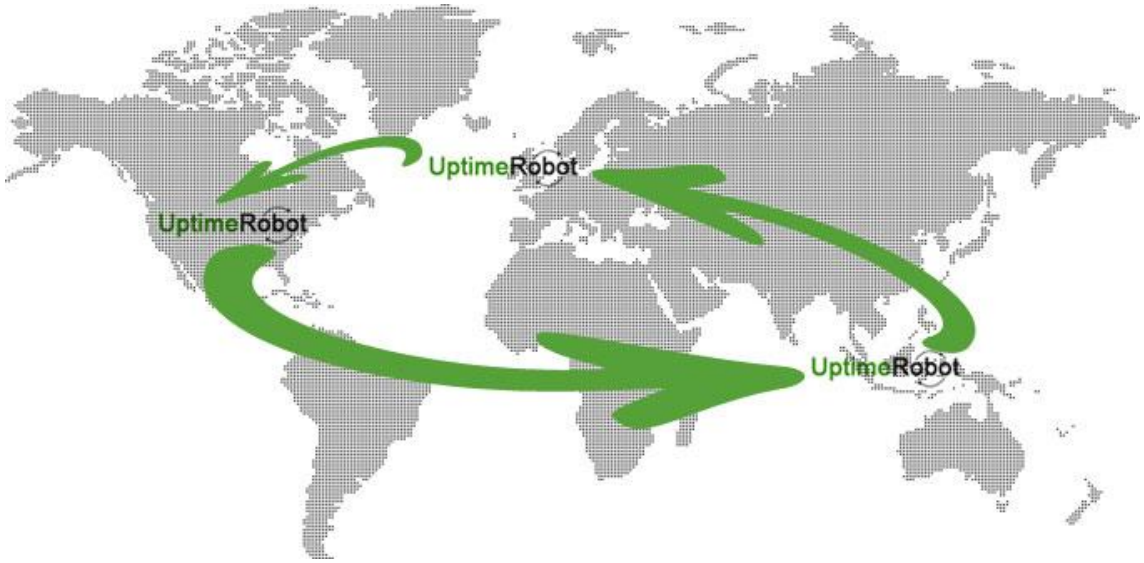
Nevýhody

- U všech alternativ je povinné objednat si minimálně 10 předplacených SMS bez ohledu na jejich využití

4.1.4. Uptimerobot.com

Uptimerobot.com je společnost, která vznikla v lednu 2010 v USA. Jako jedna z mála poskytuje své služby zdarma. Politika této společnosti je poskytování služeb zdarma. Financovat provoz chtějí pouze z reklamy. Na webových stránkách této služby prezentují, že minimálně do 1. června 2012 bude služba určitě zdarma, následně pokud neseženou reklamní sponzory, bude částečně zpoplatněna. Nic méně, v současné době nabízí registraci zdarma a monitoring až 50 webových stránek nebo pingů zdarma. Notifikaci nabízejí formou emailu, Twitteru nebo iOS Push zdarma. Dále nabízejí zpoplatněnou notifikaci SMS zprávami. Další službou, kterou poskytují je API pro ovládání účtu prostřednictvím SOAP protokolu. Podporují zasílání zpráv ve formátu XML, JSON a JSON-p. Prostřednictvím jejich API je možné ovládat kompletně celý účet.

Monitoring provádějí, kromě serveru v severní Americe, také ze vzdálených serverů v Asii (Singapur) a Evropě (Irsko) umístěných v Cloudech společnosti Amazon.



Obrázek 4: Mapa serverů UptimeRobot (Zdroj: http://blog.uptimerobot.com/wp-content/uploads/uptimerobot-locations.jpg)

Výhody

- Služba provozovaná zdarma s možností monitoringu každých 5 minut
- Tři monitorovací místa v USA, Evropě a Asii
- API rozhraní
- Široká možnost notifikací

Nevýhody

- Krátkou dobu na trhu (od roku 2010)
- Pokud neseženou sponzora ve formě reklamy, služba bude zpoplatněna
- Nepohodlné ovládání – veškeré ovládání mají řešené přes JavaScript a AJAX
- Podpora malého množství protokolů a služeb
  - Ping
  - HTTP

4.1.5. Monitoring-serveru.cz

Monitoring-serveru.cz je česká společnost provozující monitoring od roku 2005. Měří, zdali je daná služba dostupná. Podporují 19 různých protokolů a služeb – HTTP, FTP, POP3, IMAP, SMTP, MySQL, Postgre SQL, Ping, LDAP, DNS server, Telnet včetně všech jejich zabezpečených protokolů.

Monitoring-serveru.cz účtuje poplatky za každé měření. To znamená, že nemají žádné paušální poplatky. Pokud je potřeba měřit nějakou službu např. pouze týden, nastaví se v administraci pouze na týden a pouze týden bude zpoplatněn. Do léta 2011 poskytovali měření hlaviček HTTP (dostupnost) v deseti-minutovém intervalu zdarma. Od této doby je i toto měření zpoplatněno.

Pro eliminaci chyb způsobených případnou nedostupností monitorovacích serverů disponují dvěma servery, ze kterých měření probíhá. Samozřejmě tyto dva monitorovací servery se monitorují navzájem.

Při nastavení měření daného protokolu nebo služby lze nastavit interval od 15 vteřin do 10 minut na jedno měření. Záleží tedy na uživateli samotném, jak potřebuje danou službu důkladně měřit. Z hlediska notifikací poskytují možnost zaslání emailem, zpoplatněnou SMS zprávou nebo jako RSS a XML kanál. Ze všech měření jsou generované statistiky.

Obrázek 5: Statistiky monitoringu společnosti monitoring-serveru.cz (zdroj: http://www.monitoring-serveru.cz)

Výhody

- Kvalitní statistiky
- Přehledné administrační rozhraní
- Široká podpora měřených služeb
- Účtování za každé provedené měření, nikoli měsíčním paušálem
- Přijatelná cena měření
- Monitoring ze dvou serverů v ČR
- Rozmezí intervalu měření 15 vteřin až 10 minut

Nevýhody

- Nemají možnost použití zdarma

4.2. Shrnutí

Vesměs všechny společnosti nabízejí monitoring stejných protokolů a služeb. Jediné, v čem se jednotlivé popisované společnosti liší je ve způsobu notifikací, zpoplatnění služeb a intervalů měření.

Existují případy, kdy je využívání externích služeb příliš robustní a zbytečně drahé. Případně může být tento způsob monitoringu nevyhovující např. z hlediska potřeb monitoringu v interní síti. Zde přichází na řadu řešení na míru popisované v další kapitole diplomové práce, které je v jistých oblastech sice omezenější než výše srovnávané služby, avšak plně dostačující z hlediska potřeb. Navíc je použití této možnosti monitoringu pro vlastní účely zcela zdarma, což přináší také jistou výhodu.

5. Výsledky a diskuze

Na základě analytické části, kde byly prozkoumány nabízené služby konkurenčních portálů zabývající se monitoringem, byly vyhodnoceny následující výsledky.

5.1. Návrh, realizace a provoz služby

Navrhovaná aplikace slouží pro monitorování různých protokolů a služeb v online, případně v LAN prostředí. Jejím základním cílem je informovat provozovatele služby, že něco není v pořádku. Měla by sloužit k notifikaci provozovatelů jednotlivých služeb, že jsou nedostupné, aby mohli začít co nejdříve podnikat kroky k nápravě. Tato notifikace by měla být co nejjednodušší, neúčelnější a nejrychlejší. Pokud provozovatel poskytuje nějakou důležitou službu, nemůže si dovolit mít několikahodinový výpadek, aniž by to zjistil do doby, než ho o tom nějaký zákazník uvědomí.

Zároveň ve světě IT se nelze spoléhat na garantované dostupnosti ze strany poskytovatelů hardwaru. Výpadky se prostě dějí a v případě, že o nich víme, můžeme je nějakým způsobem ovlivnit. Pokud o nich ovšem nevíme, nezmůžeme vůbec nic.

Tato služba je tedy určena nejen pro všechny administrátory, správce sítí, správce serverů, ale i obyčejné lidi, případně firmy, které mají své webové stránky. V ideálním případě by se vlastník, provozovatel nebo kterákoli jiná zodpovědná osoba měla o výpadku či nedostupnosti služby dozvědět jako první. Samozřejmě existuje i manuální kontrola, ale ta s velikou pravděpodobností nebude zrovna v inkriminovaný čas provedena. Proto přichází na řadu opět technika, která by tento monitoring měla provádět automaticky sama dle zvolených kritérií.

Soukromá osoba, vlastnící emailovou schránku u nějaké společnosti, poskytující mailhosting zdarma, pravděpodobně nebude potřebovat monitorovat dostupnost této služby. Avšak malý provozovatel hostingové služby bude chtít mít informace o tom, zdali všechny jeho služby fungují. Těmito službami mohou být nejenom dostupnost webových stránek, ale také dostupnost FTP, mail serveru či jiné vlastní služby.

Nová aplikace by tedy měla být navržena tak, aby bylo možné monitorovat pokud možno cokoli.

5.1.1. Navrh nabízených služeb

Jak již bylo uvedeno, nová aplikace by měla poskytovat co nejúčelnější možnost monitorování. To znamená, jak jednoduché možnosti pro monitorování v IT oblasti nezkušenými lidmi, tak i zkušenými administrátory, kteří potřebují na základě nedostupnosti provádět nějaké operace – spouštět skripty, diagnostiky atp.

Nejčastěji monitorované protokoly a služby by tedy mohly být tyto:

- HTTP
  - Zjištěni HTML head.
  - HTTP Request s GET parametry
  - HTTP Request s POST parametry
- HTTPS
  - HTTPS regest s GET parametry
  - HTTPS regest s POST parametry
- FTP
  - Odpověď na portu 21
  - Test přihlášení se zadanými parametry
- IMAP
  - Odpověď na portu 143
  - Test přihlášení se zadanými parametry
- SMTP
  - Odpověď na portu 25
  - Test přihlášení se zadanými parametry
- MySQL
  - Odpověď na portu 3306
  - Test přihlášení k databázi se zadanými parametry
- Ping

U všech zmíněných protokolů a služeb je zapotřebí monitorovat nejen, jejich dostupnost, ale je také důležité monitorovat a zjišťovat další parametry. Těmi mohou být například:

- Timeout, do kterého požadovaná služba musí odpovědět
- Někdy je zapotřebí použít přihlašovací údaje
- Monitorovat svou vlastní službu na určitém portu
- Možnosti notifikace
  - SMS
  - Email
- Vlastní intervaly mezi měřením
- Dalšími užitečnými službami mohou být
  - grafy dostupnosti
  - dlouhodobá historie dostupnosti
  - RSS feed o dostupnosti služby
  - XML informace o dostupnosti
  - XML_RPC kontaktování služby v případě výpadku
  - Spuštění určité url v případě výpadku
  - Přidávání monitorovaných míst a zjišťování aktuálních informací pomocí komunikace XML_RPC

Samozřejmostí je monitoring z více míst pro eliminování chyb způsobených vlastním monitorovacím místem a následné chování vyhodnocení nedostupnosti na základě zjištěných výsledků. Například, v některých případech, pokud dojde k neúspěšnému měření z jednoho místa, je zapotřebí pokusit se provést monitoring z jiného místa. Pokud i tam bude neúspěšné, zaslat report o nedostupnosti. V jiném případě se může hodit již informace o nedostupnosti při prvním zjištění.

5.1.2. Návrh aplikace

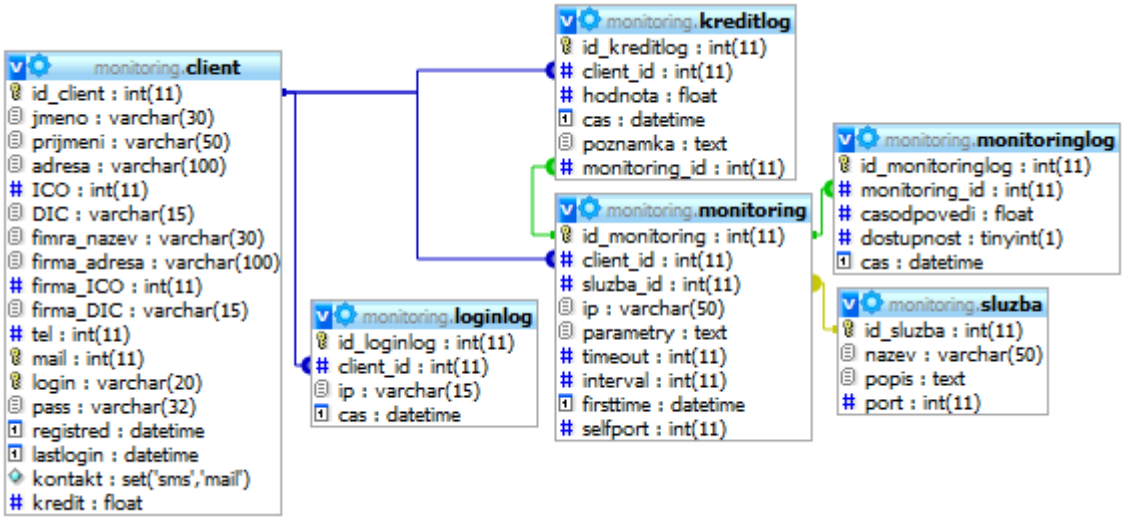
Navrhovaná aplikace je naprogramována v jazyce PHP v kombinaci s databází MySQL. Je programována pro použití v režimu PHP-CLI. To znamená, že se jednotlivé skripty spouští v konzoli operačního systému a nikoli pomocí Apache jako webová stránka. Výhodou je, že tato služba bude moci být provozována na různých operačních systémech, kde lze spouštět PHP skripty. Tudíž vývoj služby není orientován na jednotlivé operační systémy.

Služba bude provozována na Unixu, konkrétně na OS Linux Debian 6.0 za použití PHP5.3 a nejnovější verze MySQL 5.5.20.

Samozřejmostí je využití CRON deamona, který zajistí pravidelné spouštění jednotlivých skriptů tak, jak bude potřeba.

5.1.2.1. Návrh databáze

Pro aplikaci byla zvolena relační databáze MySQL především z důvodu podpory velkého množství operačních systémů, jednoduchosti nasazení a možnosti použití zdarma. Navíc tato databáze je navrhována a vyvíjena pro optimální použití s PHP a povědomí o tomto databázovém systému je velmi široké.



Obrázek 6: Schéma navržené databáze

Prvotní návrh databáze je zobrazen na schématu uvedeném výše. Tabulky splňují databázové normální formy a jsou navrženy v InnoDB. Tato forma úložiště v MySQL se vyznačuje rychlejším zpracováním složených dotazů do databáze nad více tabulkami z důvodu, že podporuje cizí klíče. Je to sice na úkor rychlosti zápisu do databáze (dotazu INSERT INTO), avšak ten aplikaci nikterak neomezuje.

Tabulka „client“ slouží k uchovávání informací o klientech. Jsou tam uvedeny veškeré kontaktní údaje, fakturační údaje, údaje o zbývajícím kreditu, datum registrace, poslední přihlášení a samozřejmě přihlašovací údaje. Přihlašování uživatelů bude možné buď prostřednictvím emailové adresy, případně prostřednictvím zvoleného loginu. Oba tyto údaje jsou samozřejmě na úrovni databáze kontrolovány z hlediska unikátnosti.

Tabulka „sluzba“ slouží k poskytování informací o nabízených službách. Jsou v ní informace o jednotlivých službách, jejich popis a vždy defaultní port těchto služeb. Na základě vybrané služby se bude provádět určitý způsob měření.

Tabulka „monitoring“ slouží k uložení jednotlivých měření. Záznamy jsou zde přes cizí klíč napojeny na klienta a službu. Vazba na tabulku služeb slouží k identifikaci měřené služby – jakým způsobem má být měření prováděno. Zároveň jsou v těchto záznamech uvedeny další podstatné informace k měření. Tyto informace jsou například IP adresy, které se mají měřit. Mohou být zadány v číselné podobě, ale také v podobě URL. Dále pro určitý druh služeb, jako je například FTP, HTTP a další, existuje pole pro uložení parametrů GET či POST či přihlašovacích údajů k dané službě. Dále je uložen požadovaný maximální timeout měření, požadovaný interval a čas, kdy má být služba poprvé změřena. Dále je možné nastavit k dané službě vlastní port, na kterém je služba dostupná.

Další 3 tabulky „kreditlog“, „loginlog“ a „monitoringlog“ slouží, jak názvy napovídají“, k logování určitých akcí. Do tabulky „monitoringlog“ se ukládají jednotlivé výsledky o měřeních – tj. zdali byla v měřený čas služba dostupná, jak dlouho trvalo, než služba odpověděla a kdy byla služba měřena. Z této tabulky se generují grafy o dostupnosti a jednotlivé informační kanály XML, RSS. Tabulka „loginlog“ slouží k logování přihlášených uživatelů. Do této tabulky se ukládají informace, odkud byl uživatel přihlášen. Poslední tabulka „kreditlog“ loguje čerpání kreditu uživatele.

5.1.2.2. Návrh monitorovacích scriptů

Jak již bylo zmíněno na začátku této kapitoly, jednotlivé monitorovací skripty jsou naprogramovány ve skriptovacím jazyce PHP, provozovány na OS Debian 6.0 a spouštěny v režimu PHP-CLI pomocí deamona CRON.

CRON spouští pravidelně každou minutu řídící skript driver.php, který zjistí z databáze, z tabulky monitoring, jaké služby se mají právě monitorovat a pomocí příkazů EXEC spustí jednotlivé monitorovací skripty s danými parametry získanými z databáze.

Každou službu je potřeba monitorovat jiným způsobem, proto má každá svůj monitorovací skript. Skriptu se vždy předávají informace potřebné k měření jako argumenty. Například monitoring služby FTP se spouští skriptem:

```
php ftp.php --port=21 --ip=192.168.12.23 --timeout=30
 --param="login(katolicky,test);" --secondtest=false
```

Zdrojový kód 47: Spouštění skriptu pro monitoring FTP včetně parametrů

V každém monitorovacím skriptu je nejdříve funkce, která zpracuje argumenty do pole, aby s nimi mohl skript dále pracovat. Výše uvedené parametry jsou předány skriptu v asociativním poli takto:

```
array(5) {
  ["port"]=>
  string(2) "21"
  ["ip"]=>
  string(13) "192.168.12.23"
  ["timeout"]=>
  string(2) "30"
  ["param"]=>
  string(22) "login(katolicky,test);"
  ["secondtest"]=>
  string(5) "false"
}
```

Zdrojový kód 48: Vyhodnocení předaných parametrů vstupní funkcí v PHP

Takto zadaný požadavek se bude snažit připojit k FTP serveru na IP 192.168.12.23 na defaultním portu 21 s přihlašovacími údaji login: katolicky, heslo: test. V případě, že se skriptu nepodaří připojit na FTP účet do 30 sekund, bude test vyhodnocen jako nedostupný. V tomto konkrétním případě se rovnou test vyhodnotí jako nedostupný a uloží se do tabulky monitoringlog. V případě, že by

parametr --secondtest byl true, pak skript provede požadavek na druhý monitorovací server, aby zkusil otestovat tuto dostupnost z jiného místa. V případě neúspěchu i zde pak prohlásí server za nedostupný.

Každý uživatel si má možnost ve své administraci zvolit, zdali chce být o výpadcích informován a jakým způsobem. Možností, jak sledovat výpadky, je několik. Buď přímo přes administraci, kde jsou vidět veškeré výpadky a statistika dostupnosti, nebo přes některé rozhraní. Těmi jsou například RSS nebo XML. Samozřejmostí je pak emailová notifikace zdarma.

5.1.2.3. Specifické nastavení serverů

Server v prostředí Linux Debian 6.0 nemá grafické rozhraní. Veškerá konfigurace je nastavována za pomocí příkazové řádky prostřednictvím protokolu SSH. Tímto způsobem je nakonfigurovaný nejen samotný operační systém, ale i veškeré služby, které na serveru běží.

Speciální nastavení vyžaduje především PHP. V defaultním nastavení má PHP nastaveno několik proměnných, které této aplikaci nevyhovují. Jsou to především dvě proměnné. První z nich je memory_limit, která říká, kolik může PHP využít paměti. Defaultně je tato hodnota nastavena na 128Mb, což by při větším počtu monitorovaných služeb nemuselo stačit. Druhou zmiňovanou hodnotou je max_execution_time. Hodnota této proměnné říká, jak dlouho může skript běžet. Defaultně je tato hodnota nastavena na 30 sekund. Pak je skript automaticky ukončen. Tato hodnota musí být nastavena minimálně na takovou hodnotu, kterou bude možno nastavit jako timeout pro monitorování služby.

5.1.2.4. Provoz služeb

Aplikace je vyvíjena a bude zveřejněna jako opensource. Bude tedy možné si celou aplikaci stáhnout a používat ji libovolně pro vlastní potřebu. Taktéž aplikace bude spuštěna, jako zpoplatněná služba, na níže zmíněných, vlastních serverech.

K provozu monitorovacích služeb je zapotřebí virtuálního serveru. Je to potřeba ze dvou důvodů. Zaprvé jsou skripty naprogramovány pro mod PHP-CLI. Zadruhé vyžadují specifické nastavení php.ini, což žádný webhosting nenabízí. Z důvodu, aby se eliminovaly zdroje nedostupnosti přetížením linky nebo chybou vlastního serveru, je zapotřebí pro měření použít minimálně 2 servery. Pro tyto účely byly vybrány Virtuální Privátní Servery (VPS) od společnosti Wedos a od společnosti Hetzner v Německu. Tato odlišná umístění serverů by měla dostatečně zajistit, aby měření byla relevantní.

V první fázi spouštění služby bude služba dostupná pouze pro Českou republiku, a to především z důvodů fakturace služeb.

Ve druhé fázi, jakmile bude vyřešena problematika fakturace DPH, bude služba postupně rozšířena pro země, jako je Polsko, Slovensko, Maďarsko, Rakousko, Německo a další evropské země v rámci Evropské unie.

Ve třetí fázi bude služba rozšířena do ostatních zemí mimo Evropu, především do USA, Austrálie a Asie.

Samozřejmostí expanze do dalších zemí je zajištění provozu serverů v monitorovaných lokalitách, případně v lokalitách, kam vedou přímé páteřní linky, aby měření nebyla zkreslena rychlostí internetového připojení mezi danými peeringovými centry. Určitě není možné monitorovat servery v USA z České republiky a brát výsledné odezvy jako relevantní a směrodatné.

5.1.2.5. Ovládání služby

Služba se skládá z monitorující části, databázové části a ovládací části. Monitorující část se skládá z jednotlivých skriptů, prostřednictvím kterých se provádějí jednotlivé monitorování. Databázová část se skládá z jednotlivých tabulek podrobněji popsaných v kapitole 5.1.2.1 Návrh databáze, do kterých se ukládají veškeré informace, se kterými aplikace pracuje. Poslední, ovládací, část je řešena prostřednictvím webové administrace, přes kterou lze ovládat a nastavovat celou službu. Více o tomto administračním prostředí je popsáno v kapitole 5.1.3 Administrační prostředí.

5.1.2.6. Propagace služeb

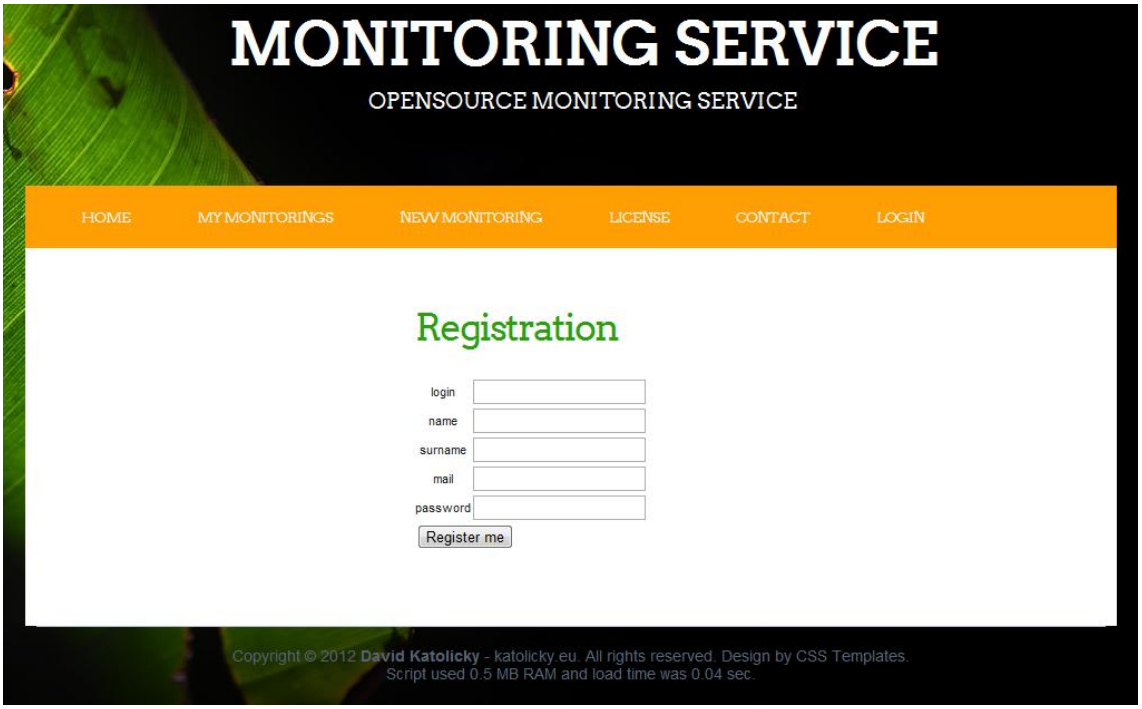
Po úplném dokončení aplikace bude služba propagována pomocí banerové reklamy na tématicky podobných webech. Zároveň bude využito PPC reklamních kampaní služeb AdSence, Etarget, případně jiných. Další propagace bude prováděna prostřednictvím PR článků na webových stránkách zabývajících se touto tématikou, různých fórech, blogách apod. Možná bude využito i direct mailingu a přímého oslovení potencionálních zákazníků z řad firem prostřednictvím telefonu.

5.1.3. Administrační prostředí

Administrační rozhraní je důležité pro ovládání a sledování výsledků měření. Veškeré nastavení těchto měření, konfigurace měření, nastavení akcí v případě, že došlo k výpadku a ostatní nastavení probíhá právě prostřednictvím Administračního prostředí.

5.1.3.1. Autentizace

Aplikace je připravena pro použití více uživateli. To, aby si vzájemně jednotliví uživatelé nemohli měnit nastavení měření, je v aplikaci vyřešena autentizace. Nový uživatel může provést registraci. Ta probíhá prostřednictvím jednoduchého formuláře.



Obrázek 7: Registrační formulář vyvíjené aplikace

Nový uživatel zde zadá uživatelské jméno, prostřednictvím kterého se bude do aplikace přihlašovat, jeho reálné jméno a příjmení, které slouží k identifikaci dané osoby, email, na který mají být prováděny notifikace a samozřejmě heslo. Registrační formulář lze jednoduše rozšířit pro registraci firem. Takto rozšířený formulář by pak ještě obsahoval pole pro název firmy, adresu, IČO atd. Tabulka „clients“ v databázi je pro toto rozšíření již připravena.

Po úspěšné registraci je tedy možné se přihlásit. K tomu slouží přihlašovací formulář po klikutní na volbu „Login“.

Obrázek 8: Přihlašovací formulář vyvíjené aplikace

V případě, že bylo zadáno správné uživatelské jméno a heslo, je uživatel přihlášen a přesměrován na podstránku „My monitorings“.

Údaje o přihlášeném uživateli jsou uloženy v globálních proměnných Sessions. Je tam uložen údaj o tom, který uživatel je momentálně přihlášený a kontrolní hash jeho účtu. Při jakémkoli pohybu po administraci je prováděno vždy ověření, zdali informace o uživateli a hash účtu souhlasí. V případě, že tyto data nesouhlasí, je uživatel okamžitě odhlášen. Toto je prováděné pomocí jednoduché třídy checkuser, která je volaná při každém pohybu na stránce.

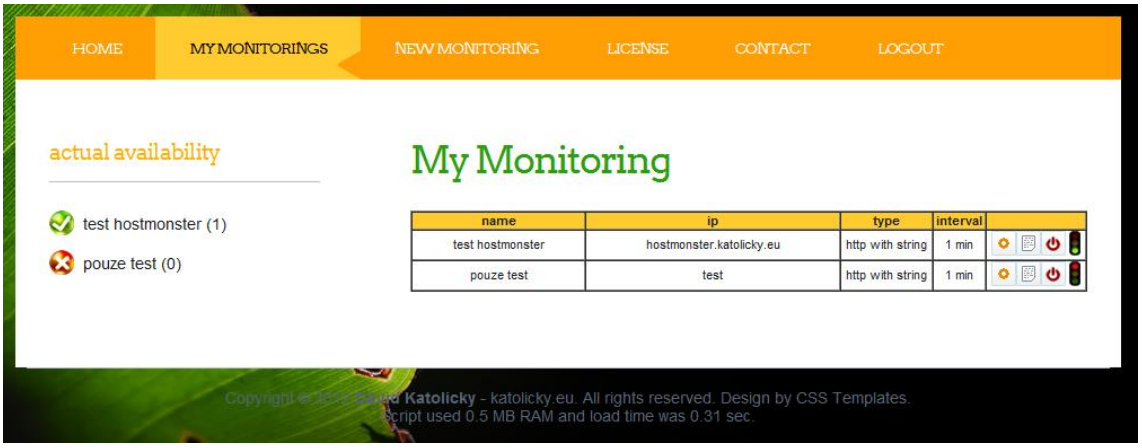
```
class checkuser{
	public function __construct() {
		$db = new db();

		if($db->numRows("SELECT * FROM client WHERE
id_client='$_SESSION[usr]' AND hash='$_SESSION[hash]'")==0)
		{
			unset($_SESSION['usr']);
			unset($_SESSION['hash']);
			session_destroy();
		} } }
```

Zdrojový kód 49: Třída pro kontrolu autentizace

5.1.3.2. Přehled mých měření

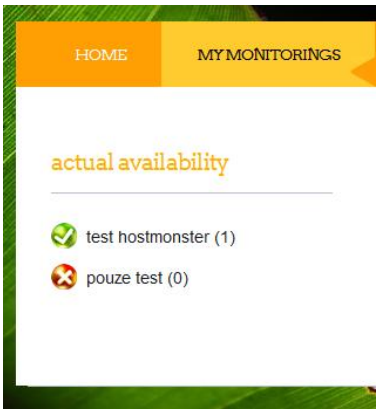
Jak již bylo zmíněno, po úspěšném přihlášení je uživatel automaticky přesměrován do sekce moje měření, neboli „My monitorings“. Zde jsou zobrazeny všechny monitorované služby uživatele a veškeré odkazy pro ovládání a zjišťování informací o dostupnosti.



Obrázek 9: Přehled mých měření vyvíjené aplikace

Na obrázku je vidět daná sekce. Jsou zde 2 měření, z toho je první aktivní a druhé neaktivní. Dále jsou zobrazeny základní informace o měření, tj. název, monitorovaná IP adresa zadaná v číselném formátu, nebo jako URL, dále pak typ a frekvence měření. V posledním sloupci tabulky jsou tlačítka pro zobrazení nastavení, statistik a výpis dat a časů, kdy byl server offline.

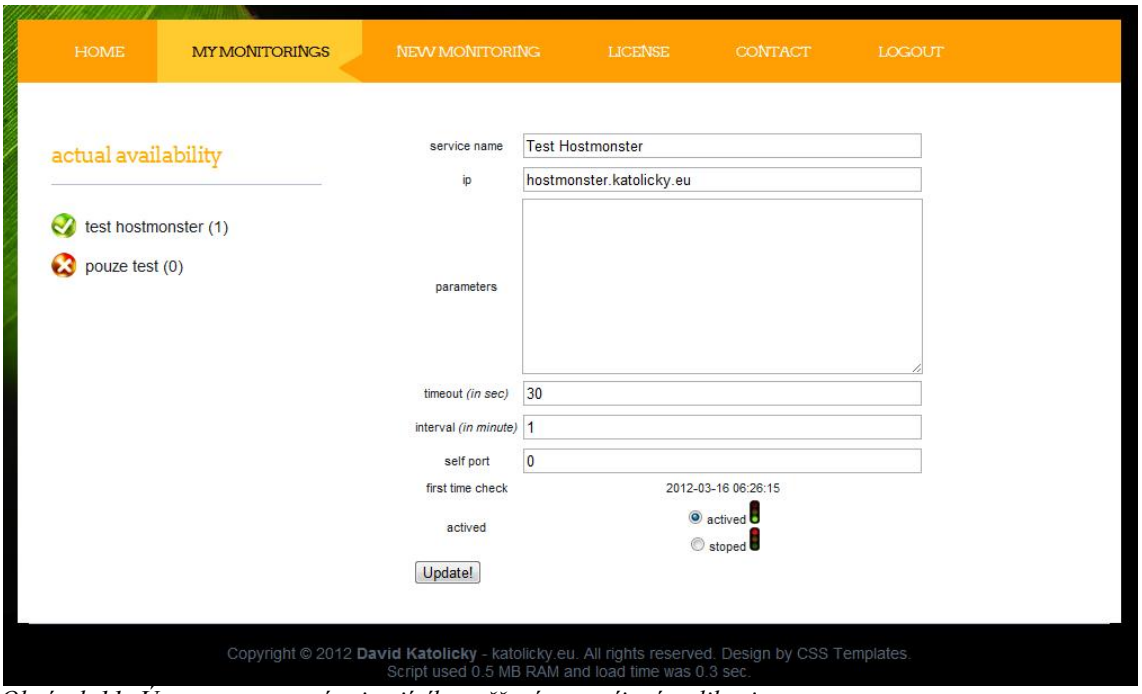
Pro rychlý přehled je vždy v levém sloupci výpis monitorovaných služeb daným uživatelem, obrázkem je naznačeno, zdali poslední měření bylo úspěšné či nikoli a v závorce pak číslem uvedený počet neúspěšných měření za posledních 24 hodin. Po kliknutí na jednotlivé služby je uživatel přesměrován do sekce statistik dané služby.



Obrázek 10: Sidebar s přehledem dostupnosti měření vyvíjené aplikace

5.1.3.3. Úprava a vytvoření nového měření

Při vytváření, respektive úpravě existujícího měření, je potřeba vyplnit, respektive editovat, krátký formulář, kde se uvádí informace důležité pro nastavení měření.

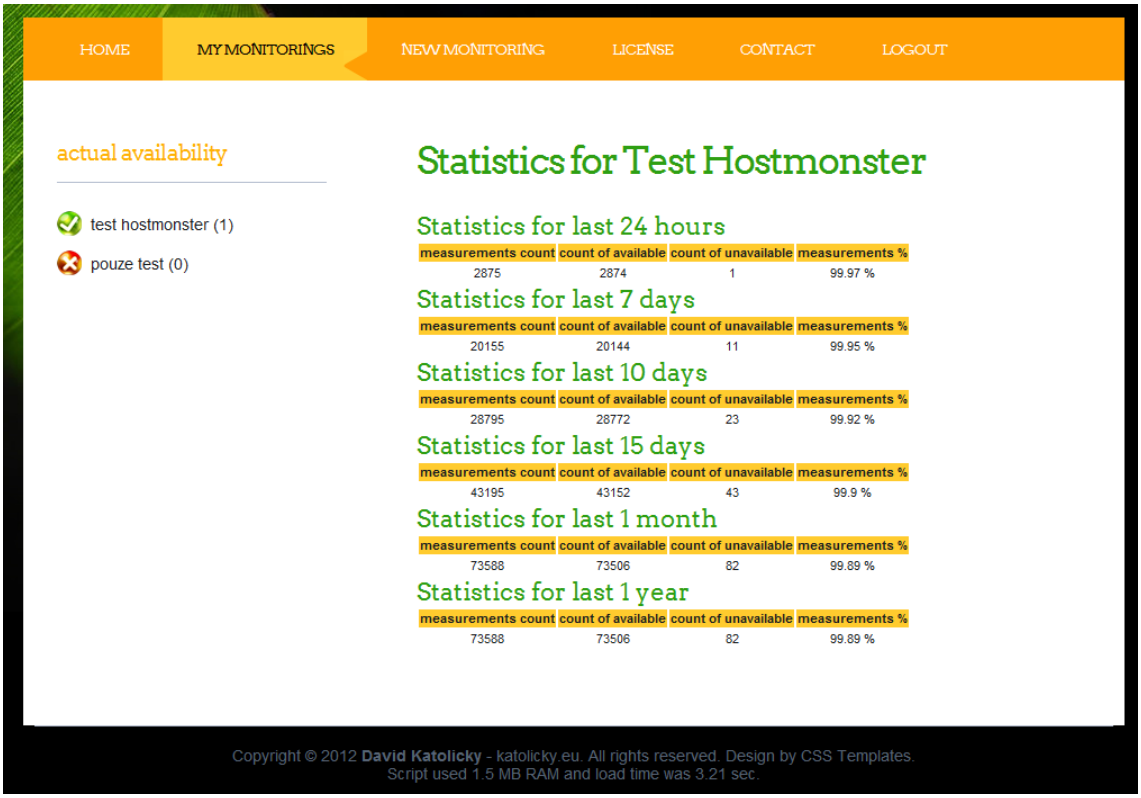


Obrázek 11: Úprava nastavení existujícího měření ve vyvíjené aplikaci

Na uvedeném obrázku je pro příklad uvedena editace existujícího měření „Test Hostmonster“. Jak je z obrázku patrné, lze upravit zásadní položky, jako jsou název služby, IP adresa služby, parametry pro testování, timeout, interval měření, případně vlastní port pro případ, že daná služba nepracuje na defaultním portu. V poslední řadě je možnost dané měření zapnout či vypnout.

5.1.3.4. Statistiky měření

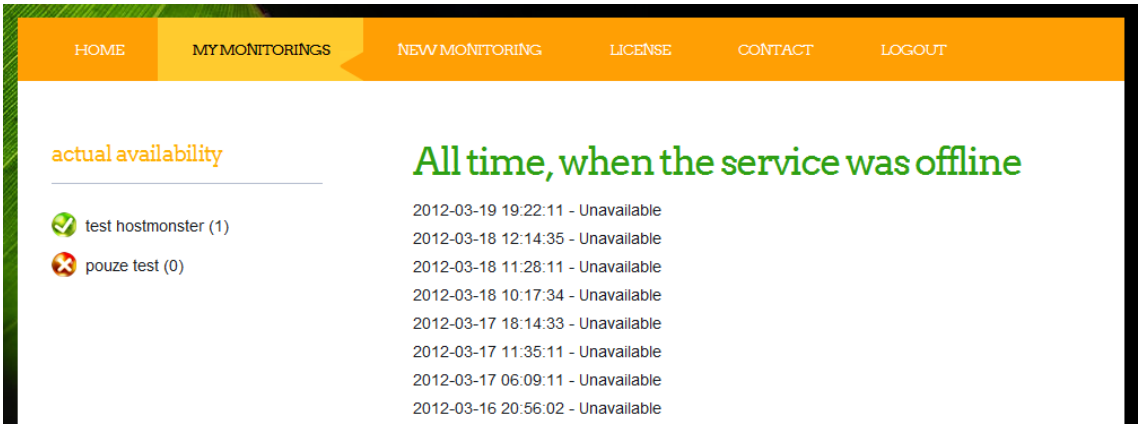
Jednou z nejdůležitějších částí jsou pravděpodobně statistiky měření. Tyto statistiky jsou dostupné po kliknutí na tlačítko  „statistiky“. Na níže uvedeném obrázku jsou uvedeny statistiky měření opět pro službu „Test Hostmonster“. Aplikace ukazuje statistiky za posledních 24 hodin, 7 dní, 10 dní, 15 dní 1 měsíc a 1 rok. Vždy na přelomu roku dochází k přepočítávání databáze a ukládání hodnot měření na celkové hodnoty za poslední rok. Dílčí hodnoty se pak pro rychlost a úsporu místa odstraní. U každého období jsou uvedeny hodnoty celkového počtu měření, úspěšného měření, neúspěšného měření a procentuelní dostupnosti dané služby v daném období.



Obrázek 12: Podrobné statistiky konkrétního měření ve vyvíjené aplikaci

5.1.3.5. Přehled neúspěšných měření

Poslední volbou měření je přehled o neúspěšném měření. Tyto údaje jsou zobrazeny pouze jako výpis dat a časů kdy byla služba nedostupná. Tento přehled není potřeba nijak dál popisovat. Slouží pro konkrétní zobrazení, kdy byla služba nedostupná.



Obrázek 13: Přehled všech neúspěšných pokusů pro konkrétní měření ve vyvíjené aplikaci

5.1.4. Monitorovací skript HTTP se zjištěním výskytu textu

Skript má za úkol zjištění dostupnosti daných HTTP stránek. V případě, že je stránka dostupná, ověří, zdali se na dané stránce vyskytuje konkrétní text. Tento druh monitoringu lze použít pro zjištění případů, kdy webová stránka vrací návratový kód 200, je na první pohled dostupná, ale například nemá funkční komunikaci s databází, a proto se pouze zobrazuje error hláška namísto konkrétního obsahu.

K takovémuto typu monitoringu je použita následující funkce:

```
function check($host, $timeout, $find) {
    $fp = fsockopen($host, 80, $errno, $errstr, $timeout);
    if (!$fp) {
        echo "$errstr ($errno)\n";
    } else {
       $header = "GET / HTTP/1.1\r\n";
       $header .= "Host: $host\r\n";
       $header .= "Connection: close\r\n\r\n";
       fputs($fp, $header);
       while (!feof($fp)) {
           $str .= fgets($fp, 1024);
       }
       fclose($fp);
       return (strpos($str, $find) !== false);
    }
}
```

Zdrojový kód 50: Funkce check pro monitorování http se zjištěním výskytu textu

Funkce pracuje tak, že jí jsou předány parametry $host, $timeout a $find. Funkce se pokusí otevřít socket na zadanou adresu uvedenou v proměnné $host. Pokud se podaří tento socket vytvořit do času uvedeného v proměnné $timeout, provede načtení obsahu webové stránky a v tomto obsahu hledá textový řetězec uvedený v proměnné $find. V případě, že tento řetězec nalezne, vrátí funkce hodnotu TRUE. V případě, že se nepodaří vytvořit socket nebo nebude nalezen textový řetězec, funkce vrátí hodnotu FALSE.

Výsledek takového měření bude podobně jako všechny ostatní ukládán do databáze pro následující přehledy. V případě, že stránka bude nedostupná, bude se

na základě uživatelem definovaných pravidel provádět notifikace o nedostupnosti služby.

5.1.5. Monitorovací skript HTTP HEAD

Skript má za úkol monitorovat definovanou stránku, zdali funguje. Toto měření funguje na základě zjišťování hlaviček stránky. Skript si z dané monitorované stránky vyžádá hlavičky (nikoli celý obsah) a následně je vyhodnotí. Pokud server vrátí v HTTP hlavičce návratový kód 200, stránka je prohlášena za funkční. V případě jiného návratového kódu je stránka vyhodnocena jako nedostupná a provádí se notifikace uživateli.

Skript má nevýhodu, že pokud se na dané webové stránce bude zobrazovat něco, co se tam zobrazovat nemá (například že stránky jsou v režimu úprav), neovlivní to monitoring, protože tato funkce nepracuje s obsahem.

5.1.6. Monitorovací skript FTP

Skript má za úkol monitoring dostupnosti FTP účtu. Dostupnost ale není jediným problémem FTP služby. Výjimečně může dojít k tomu, že služba vypadá jako dostupná (odpovídá na portu 21), ale nelze se k ní připojit - Odmítá příchozí připojení. Proto tento skript má možnost simulovat připojení k FTP. Uživatel má možnost tuto volbu zvolit, zadat své přihlašovací údaje k FTP a skript se následně bude připojovat k FTP. V případě, že k připojení nedojde, vrátí chybu.

5.1.7. Monitorovací skript SSH

Funguje na základě monitorování portu 22. V případě, že služba odpovídá na daném portu, je měření vyhodnoceno jako „dostupné“. V opačném případě jako nedostupné. U tohoto typu měření není z důvodu bezpečnosti možnost testování konkrétního připojení prostřednictvím uživatelského jména a hesla (ke službě SFTP/SCP)

5.1.8. Monitorovací skript POP3 a IMAP

Skript pro monitorování protokolu POP3 má pochopitelný důvod. V případě provozování mail serveru je důležité, aby se také uživatel dostal ke svým mailům prostřednictvím protokolu POP3 nebo IMAP. To, zdali jsou tyto protokoly dostupné je tedy potřeba v určitých případech monitorovat. A přesně k takovému slouží tento skript. Pomocí něj se monitoruje, zdali je POP3, respektive IMAP, dostupné a zdali se k serveru dá prostřednictvím konkrétních přihlašovacích údajů přihlásit.

5.1.9. Monitorovací skript SMTP

Další službou mailového serveru je odesílání pošty. To probíhá prostřednictvím SMTP serveru. V případě, že z nějakého důvodu SMTP server přestane fungovat, neodcházejí žádné emaily, což je většinou velký a těžko vysvětlitelný problém. Proto je nutné službu pravidelně monitorovat, aby nedošlo k delšímu výpadku a předešlo se tak nepříjemnostem vůči uživatelům mailového serveru.

5.2. Další možnosti rozšíření

Jako další rozšíření této služby je připraven kanál XML-RPC, přes který bude možno zasílat autorizované požadavky. Těmito požadavky jsou například:

- Vložení nové služby k monitorování
- Získání informací o monitorování
- Získání seznamu monitorování a jejich nastavení
- Získání informací o osobním účtu, kreditu a čerpání
- Vytvoření uživatele

Další z připravovaných rozhraní je klasický HTTP request s POST proměnnými, ve kterých uživatel definuje požadavek, ten se vyhodnotí a uživateli bude vrácen zpět XML výsledek o provedení požadavku. Přes toto rozhraní bude možné ovládat účet stejným způsobem, jako u výše uvedeného XML-RPC.

Plánované je rozšíření všech provozovaných protokolů a služeb pro monitoring v SSL režimu. Tudíž služeb IMAPs, POP3s, SMTPs, FTPs atd.

Další, do budoucna plánovanou službou, je monitoring linuxových serverů. Data pak budou aplikaci pravidelně předávána přes již zmíněné rozhraní XML-RPC. Monitoring se bude seznam spuštěných procesů, vytížení serveru z pohledu paměti RAM, disku a procesoru. Bude monitorovat počet SQL dotazů do databáze. Ze všech těchto testů budou pravidelně generované grafy. V případě, že nějaká hodnota přesáhne

uživatelem nastavenou horní hranici, bude zaslána informace prostřednictvím emailové zprávy, případně prostřednictvím SMS. Tato služba by měla vyřešit problém s vytížení serveru, místa na disku, dalšími hardwarovými prostředky a upozornit administrátora na případné výjimečné chování serveru.

Další plánovanou, velice užitečnou utilitou, je aplikace pro telefony s OS Android, přes kterou půjde nejen svůj účet obsluhovat, ale bude možné provádět jednorázové testy. Taktéž přes tuto aplikaci bude možné sledovat grafy dostupnosti a všechny důležité výsledky.

6. Závěr

V teoretické části diplomové práce byly uvedeny specifikace a náležitosti jazyka XML. Tento jazyk byl charakterizován v souvislosti s funkcemi pro práci se soubory XML v programovacím jazyce PHP. Taktéž byly řešeny principy komunikačních rozhraní webových služeb SOAP a XML-RPC.

V analytické části praktické části práce jsou charakterizovány vybrané společnosti, které se zabývají monitoringem internetových protokolů a služeb. Byly srovnávány české i zahraniční společnosti, a to jak z hlediska poskytovaných služeb, tak i z hlediska poplatků za monitoring. Závěrem analytického průzkumu bylo zjištění, že téměř všechny společnosti nabízejí stejné možnosti monitoringů. Jednotlivé společnosti se zásadně lišily pouze v cenách a množství monitoringových serverů, odkud měření probíhá. Avšak žádná z porovnávaných společností nenabízela možnost monitorovat služby uvnitř lokální sítě, což může být potřebné např. pro velké firmy. Na tomto nedostatku byl postaven základ návrhu nové aplikace, která je popsána v druhé polovině praktické části diplomové práce v kapitole 5. Výsledky a diskuze.

Zde jsou provedeny jednotlivé kroky důležité pro návrh nové aplikace, která má být distribuována jako opensource aplikace, a tudíž může být instalována podle potřeby, například i ve vnitřní síti, a sloužit tak k monitorování lokálních služeb. Návrh aplikace probíhal v několika krocích. V první fázi byl zvolen programovací jazyk vhodný pro tuto aplikaci. Tímto jazykem je PHP, a to především z důvodů velké podpory v nejrůznějších operačních systémech. V druhé fázi byl vybrán databázový server MySQL, který optimálně spolupracuje s jazykem PHP. Zároveň také splňuje předpoklady jednoduchosti nasazení a rozšířené povědomí o možnostech nastavení tohoto databázového serveru. Z těchto tří důvodů se tedy zdá být pro tento účel vhodný. V další fázi byl zvolen poslední klíčový parametr - operační systém. Jako ideální se zdál být Debian 6.0 z hlediska malé náročnosti samotného OS na hardwarové prostředky serveru a jeho stabilitě. V případě provozu aplikace jako online nástroje pro veřejnost byly zvoleny dvě konkrétní společnosti poskytující VPS.

Dále proběhl návrh a následně vývoj samotné aplikace, která měla splňovat jednoduchost nasazení a intuitivní ovládání. Z jednotlivých analýz byly zjištěny podstatné protokoly a služby k monitoringu. Následně byla navržena vhodná databáze podporující multiuživatelský přístup k aplikaci. Samotné ovládání služby je řešeno prostřednictvím webového rozhraní, kde jsou zobrazovány přehledy měřených služeb, aktuální dostupnost, statistiky dostupnosti, statistiky výpadků atd.

Dalším plánovaným rozšířením této aplikace je např. realizace API, mobilní aplikace pro zařízení s OS Android a iOS, rozšíření o další služby, které lze monitorovat atd.

Výhodou nové aplikace je především distribuce jako opensource řešení s možností využití pro vlastní potřeby zdarma. To znamená, že je možné ji využít nejen k monitoringu online protokolů a služeb, ale i služeb ve vnitřní síti bez připojení k internetu. Jako výhodu lze také považovat, vzhledem k relativně hodně rozšířenému jazyku PHP, možnost jakýchkoli úprav aplikace dle vlastních potřeb, případně doprogramování individuálně potřebných modulů. Jako nevýhodu lze dočasně brát malý rozsah monitorovaných služeb v porovnání s analyzovanými společnostmi a možnost notifikací o výpadcích pouze přes emailové zprávy.

7. Seznam literatury

- KOSEK, Jiří. *PHP a XML*. 1. vyd. Praha: Grada, 2009, 367 s. ISBN 978-80-247-1116-4 (BROž.).
- SKLAR, David. *PHP 5 moduly, rozšíření, akcelerátory*. Vyd. 1. Brno: Zoner Press, 2005, 341 s. ISBN 80-868-1519-6.
- ULLMAN, Larry. *PHP a MySQL, názorný průvodce tvorbou dynamických WWW stránek, první vydání.* Computer Press, Brno 2004. Str. 534. ISBN: 80-251-0063-4
- TANSLEY, David. *PHP a MySQL – Vytváříme Dynamické Webové stránky*. Softpress s.r.o, Praha 2003. Str. 480. ISBN: 80-86497-40-2
- MACH, Jakub. *PHP pro úplné začátečníky, Druhé vydání.* Computer Press, Brno 2003. Str. 167. ISBN: 80-7226-834-1
- BRADLEY, Neil. *XML: kompletní průvodce.* 1. vyd. Praha: Grada, 2000, 537 s. ISBN 80-716-9949-7.
- GILMORE, Jason W. *Velká kniha PHP 5 MySQL: kompendium znalostí pro začátečníky i profesionály.* Vyd. 1. Brno: Zoner Press, 2005, 711 s. ISBN 80-868-1520-X.
- WELLING, Luke, THOMSONOVÁ, Laura. *PHP a MySQL – rozvoj webových aplikací, Druhé vydání.* Softpress s.r.o, Praha 2004. Str. 910. ISBN: 80-86497-60-7
- GUTMANS, Andi a spol. *Mistrovství v PHP 5, První vydání*. CP Books, Brno 2005. Str. 655. ISBN: 80-251-0799-X
- CASTAGNETTO, Jesus a spol. *PHP Programujeme profesionálně, První vydání.* Computer Press, Brno 2001. Str. 656. ISBN: 80-7226-310-2
- CHOW, Shu-Wai. *Programujeme Mashup aplikace pro Web 2.0 v PHP*. Vyd. 1. Brno: Computer Press, 2008, 280 s. ISBN 978-80-251-2057-6.
- MARCHAL, Benoit. *XML v příkladech.* Vyd. 1. Praha: Computer Press, 2000, 447 s. ISBN 80-722-6332-3.
- SKONNARD, Aaron a Martin GUDGIN. *XML - pohotová referenční příručka: referenční příručka programátora ke XML, XPath, XSLT, XML Schema, SOAP a dalším.* 1 vyd. Překlad Lucie Rút Bittnerová. Praha: Grada, 2006, 342 s. ISBN 80-247-0972-4.
- MERCER, Dave. *Beginning PHP5*. Indianapolis, IN: Wiley, c2004, 859 s. ISBN 07-645-5783-1.
- CASTAGNETTO, Jesus. *Programujeme PHP profesionálně*. 2. oprav. a aktualiz. vyd. Překlad Ludvík Roubíček. Brno: Computer Press, 2004, 656 s. ISBN 80-722-6310-2.
- KOSEK, Jiří. *PHP - tvorba interaktivních internetových aplikací*: *podrobný průvodce*. Vyd. 1. Překlad Ludvík Roubíček. Praha: Grada, 1999, 490 s. Průvodce (Grada). ISBN 80-716-9373-1.

Elektronické zdroje

- PHP: Hypertext Preprocessor. *PHP: Hypertext Preprocessor* [online]. [cit. 2012-02-21]. Dostupné z: http://www.php.net
- VRÁNA, Jakub. Webové služby v PHP: XML-RPC a SOAP. Root.cz [online]. 10.8.2007[cit. 2012-03-12]. Dostupné z: http://www.root.cz/clanky/webove-sluzby-php-xmlrpc-soap/
- MĚCHÁČEK, Jaroslav. Autentizace na webu. In: *UVT MU Zpravodaj* [online]. 1999 [cit. 2012-03-13]. Dostupné z: http://www.ics.muni.cz/bulletin/articles/173.html

8. Seznamy

- **Seznam zdrojových kódů**
  - Zdrojový kód 1: Příklad použití formuláře QuickForm - str. 9
  - Zdrojový kód 2: Popis XML souboru pro použití v XML_RPC - str. 12
  - Zdrojový kód 3: Metoda vytváření objektu XML_RPC_VALUE - str. 12
  - Zdrojový kód 4: Vytvoření nového XML_RPC klienta - str. 13
  - Zdrojový kód 5: Vytvoření XML_RPC zprávy, která se bude zasílat na server - str. 13
  - Zdrojový kód 6: Odeslání zprávy na server - str. 13
  - Zdrojový kód 7: Získání návratové hodnoty po odeslání zprávy na server - str. 13
  - Zdrojový kód 8: Syntaxe funkce mail() - str 14
  - Zdrojový kód 9: Syntaxe funkce mail () s hlavičkou from - str 15
  - Zdrojový kód 10: Kontrola proměnných zaslaných prostřednictvím formuláře - str 16
  - Zdrojový kód 11: Ukázka elementu XML - str 18
  - Zdrojový kód 12: Ukázka správného použití elementu v XML - str 18
  - Zdrojový kód 13: Ukázka alternativy správného použití elementu v XML - str 19
  - Zdrojový kód 14: Datový model XML v kódu - str 19
  - Zdrojový kód 15: Ukázka vložení atributu do elementu XML - str 20
  - Zdrojový kód 16: Ukázka vložení více atributů do elementu XML - str 20
  - Zdrojový kód 17: Ukázka chybně zapsaného elementu - str 21
  - Zdrojový kód 18: Ukázka kompletního jednoduchého dokumentu XML - str 21
  - Zdrojový kód 19: Ukázka kompletního jednoduchého dokumentu XML s použitím vkládání znaku v Unicode - str 22
  - Zdrojový kód 20: Ukázka použití jmenných prostorů v XML - str 22
  - Zdrojový kód 21: Použití implicitního jmenného prostoru - str 23
  - Zdrojový kód 22: Deklarace prefixu - str 23
  - Zdrojový kód 23: Použití jmenného prostoru prostřednictvím prefixu - str 24
  - Zdrojový kód 24: Určení jazyku dokumentu pomocí xml:lang - str 25
  - Zdrojový kód 25: Ukázka Použití xml:id - str 26
  - Zdrojový kód 26: Ukázka bez použití xml:base - str 26
  - Zdrojový kód 27: Výsledný formát bez použití xml:base - str 27
  - Zdrojový kód 28: Ukázka s použitím xml:base - str 27
  - Zdrojový kód 29: Načtení komprimovaného XML dokumentu pomocí SimpleXML - str 29
  - Zdrojový kód 30: Ukázka XML dokumentu obsahující elementy i atributy - str 30
  - Zdrojový kód 31: Získání názvu odběratele z dokumentu XML - str 30
  - Zdrojový kód 32: Projití všech elementů „položka“ v XML - str 31
  - Zdrojový kód 33: Projití dokumentu XML a získání všech potomků elementu odběratele - str 31
  - Zdrojový kód 34: Získání atributů z XML - str 31
  - Zdrojový kód 35: Začátek elementu SAX - str 32
  - Zdrojový kód 36: Použití jmenných prostorů v SAX - str 32
  - Zdrojový kód 37: Konec elementu v SAX - str 33
  - Zdrojový kód 38: Vytvoření obsahu elementu předáním dat funkci characters - str 33
  - Zdrojový kód 39: Načtení dokumentu DOM z XML - str 35

9. Přílohy

9.1. SQL pro vytvoření databáze

```
--
-- Struktura tabulky `client`
--


CREATE TABLE IF NOT EXISTS `client` (
  `id_client` int(11) NOT NULL AUTO_INCREMENT,
  `jmeno` varchar(30) COLLATE utf8_czech_ci NOT NULL,
  `prijmeni` varchar(50) COLLATE utf8_czech_ci NOT NULL,
  `adresa` varchar(100) COLLATE utf8_czech_ci NOT NULL COMMENT 've
formatu Ulice Č.P., mesto, psc, zeme',
  `ICO` int(11) NOT NULL,
  `DIC` varchar(15) COLLATE utf8_czech_ci NOT NULL,
  `fimra_nazev` varchar(30) COLLATE utf8_czech_ci NOT NULL,
  `firma_adresa` varchar(100) COLLATE utf8_czech_ci NOT NULL COMMENT
've formatu Ulice Č.P., mesto, psc, zeme',
  `firma_ICO` int(11) NOT NULL,
  `firma_DIC` varchar(15) COLLATE utf8_czech_ci NOT NULL,
  `tel` int(11) NOT NULL,
  `mail` varchar(250) COLLATE utf8_czech_ci NOT NULL,
  `login` varchar(20) COLLATE utf8_czech_ci NOT NULL,
  `pass` varchar(32) COLLATE utf8_czech_ci NOT NULL,
  `registred` datetime NOT NULL,
  `lastlogin` datetime NOT NULL,
  `kontakt` set('sms','mail') COLLATE utf8_czech_ci NOT NULL DEFAULT
'mail',
  `kredit` float NOT NULL DEFAULT '0',
  PRIMARY KEY (`id_client`),
  UNIQUE KEY `mail` (`mail`),
  UNIQUE KEY `login` (`login`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci
AUTO_INCREMENT=1 ;


-- --------------------------------------------------------


--
-- Struktura tabulky `kreditlog`
--


CREATE TABLE IF NOT EXISTS `kreditlog` (
  `id_kreditlog` int(11) NOT NULL AUTO_INCREMENT,
  `client_id` int(11) NOT NULL,
```

```
  `hodnota` float NOT NULL,
  `cas` datetime NOT NULL,
  `poznamka` text COLLATE utf8_czech_ci NOT NULL,
  `monitoring_id` int(11) NOT NULL,
  PRIMARY KEY (`id_kreditlog`),
  KEY `client_id` (`client_id`),
  KEY `monitoring_id` (`monitoring_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci
AUTO_INCREMENT=1 ;


-- --------------------------------------------------------


--
-- Struktura tabulky `loginlog`
--


CREATE TABLE IF NOT EXISTS `loginlog` (
  `id_loginlog` int(11) NOT NULL AUTO_INCREMENT,
  `client_id` int(11) NOT NULL,
  `ip` varchar(15) COLLATE utf8_czech_ci NOT NULL,
  `cas` datetime NOT NULL,
  PRIMARY KEY (`id_loginlog`),
  KEY `client_id` (`client_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci
AUTO_INCREMENT=1 ;


-- --------------------------------------------------------


--
-- Struktura tabulky `monitoring`
--


CREATE TABLE IF NOT EXISTS `monitoring` (
  `id_monitoring` int(11) NOT NULL AUTO_INCREMENT,
  `client_id` int(11) NOT NULL,
  `sluzba_id` int(11) DEFAULT NULL,
  `ip` varchar(50) COLLATE utf8_czech_ci NOT NULL,
  `parametry` text COLLATE utf8_czech_ci,
  `timeout` int(11) NOT NULL DEFAULT '30',
  `interval` int(11) NOT NULL DEFAULT '10' COMMENT 'interval mezi
merenim v minutach',
  `firsttime` datetime NOT NULL,
  `selfport` int(11) DEFAULT NULL,
  PRIMARY KEY (`id_monitoring`),
```

```
  KEY `client_id` (`client_id`),
  KEY `sluzba_id` (`sluzba_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci
AUTO_INCREMENT=1 ;


-- --------------------------------------------------------


--
-- Struktura tabulky `monitoringlog`
--


CREATE TABLE IF NOT EXISTS `monitoringlog` (
  `id_monitoringlog` int(11) NOT NULL AUTO_INCREMENT,
  `monitoring_id` int(11) NOT NULL,
  `casodpovedi` float NOT NULL,
  `dostupnost` tinyint(1) NOT NULL,
  `cas` datetime NOT NULL,
  PRIMARY KEY (`id_monitoringlog`),
  KEY `monitoring_id` (`monitoring_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci
AUTO_INCREMENT=1 ;


-- --------------------------------------------------------


--
-- Struktura tabulky `sluzba`
--


CREATE TABLE IF NOT EXISTS `sluzba` (
  `id_sluzba` int(11) NOT NULL AUTO_INCREMENT,
  `nazev` varchar(50) COLLATE utf8_czech_ci NOT NULL,
  `popis` text COLLATE utf8_czech_ci NOT NULL,
  `port` int(11) NOT NULL,
  PRIMARY KEY (`id_sluzba`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci
AUTO_INCREMENT=1 ;


--
-- Omezení pro exportované tabulky
--


--
-- Omezení pro tabulku `kreditlog`
--
```

```
ALTER TABLE `kreditlog`
  ADD CONSTRAINT `kreditlog_ibfk_1` FOREIGN KEY (`client_id`)
REFERENCES `client` (`id_client`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `kreditlog_ibfk_2` FOREIGN KEY (`monitoring_id`)
REFERENCES `monitoring` (`id_monitoring`) ON DELETE CASCADE ON UPDATE
CASCADE;


--
-- Omezení pro tabulku `loginlog`
--
ALTER TABLE `loginlog`
  ADD CONSTRAINT `loginlog_ibfk_1` FOREIGN KEY (`client_id`)
REFERENCES `client` (`id_client`) ON DELETE CASCADE ON UPDATE CASCADE;


--
-- Omezení pro tabulku `monitoring`
--
ALTER TABLE `monitoring`
  ADD CONSTRAINT `monitoring_ibfk_1` FOREIGN KEY (`client_id`)
REFERENCES `client` (`id_client`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `monitoring_ibfk_2` FOREIGN KEY (`sluzba_id`)
REFERENCES `sluzba` (`id_sluzba`) ON DELETE CASCADE ON UPDATE CASCADE;


--
-- Omezení pro tabulku `monitoringlog`
--
ALTER TABLE `monitoringlog`
  ADD CONSTRAINT `monitoringlog_ibfk_1` FOREIGN KEY (`monitoring_id`)
REFERENCES `monitoring` (`id_monitoring`) ON DELETE CASCADE ON UPDATE
CASCADE;
```