Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Využití webových služeb a práce s nimi

Plzeň, 2013 Jana Nová

Originál zadání

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 7.5.2013 ….....

Jana Nová

Poděkování

Děkuji vedoucímu bakalářské práce RNDr. Mikuláši Gangurovi Ph.D. za cenné rady a připomínky při vedení práce. Dále děkuji své rodině za podporu při studiu.

Abstract

Working with Web Services and their usage

The main goal of this thesis is to describe technology of Web Services and demonstrate their practical usage. This text is focused on three main parts of Web Services – SOAP, WSDL, UDDI and also provides an overview how XML is used in Web Services.

As a practical example of Web Services usage a simple client side application of IS/STAG Web Services was implemented. This application can check full-occupied examination term and alert user when there occure an empty seat at the exam. Alert is sent via email.

Obsah

1. Úvod

V dnešní době, kdy informační systém je součástí téměř každého podniku nebo instituce, vzniká čím dál tím větší potřeba výměny informací mezi jednotlivými systémy. Mohou to být například informační systémy obchodních partnerů, které potřebují sledovat množšství zboží na skladě u dodavatele nebo párovat objednávky s daňovými doklady.
Webové služby jsou účinným nástrojem pro komunikaci vzdálených aplikací v síti. Ke komunikaci využívají především internetový protokol HTTP (případně HTTPS) a otevřený standard XML.
Díky tomu je dosaženo platformové nezávislosti, která představuje významnou výhodu této technologie. Webové služby nejsou doménou pouze v business oblasti. Jejich využití můžeme najít téměř všude, kde je potřeba komunikovat se vzdálenou aplikací. V oblasti B2B (B2G i B2E) je ale jejich role důležitá především proto, že pro podnik představuje větší flexibilita v získávání informací nemalou konkurenční výhodu.

Hlavním cílem této práce je seznámit čtenáře s technologií webových služeb a ukázat jejich využití v praxi. Mezi dílčí cíle práce patří:

- Objasnit základní princip fungování webových služeb
- Objasnit princip použití XML ve webových službách
- Popsat jednotlivé části webových služeb (WSDL, SOAP, UDDI)
- Zpracovat přehled vybraných webových služeb použitých v praxi
- Navrhnout a implementovat klientskou aplikaci
- Zpracovat průvodce vytvořením nové webové služby
- Zhodnotit aktuální využití webových služeb

V kapitole č.2 jsou objasněny principy fungování webových služeb a uvedena jejich stručná historie.

Téměř každá definice webových služeb hovoří o třech základních částech a to: WSDL, SOAP a UDDI. Právě o těchto třech hlavních pojmech webových služeb podrobněji pojednává třetí kapitola.

Čtvrtá kapitola seznamuje čtenáře s otevřeným standardem XML. Je zde zejména uvedeno, kde se XML ve webových službách používá. Pro snazší představu jsou přiloženy ukázky kódu.

Konkrétní příklady využití webových služeb v různých oblastech lze nalézt v kapitole č. 5. Tato kapitola popisuje využití webových služeb nad IS/STAG, ale také například možnosti webových služeb, které poskytuje Google. Jsou ukázany i další příklady použití z praxe.

Šestá kapitola je stručným průvodcem vytvoření nové webové služby. Uvedeny jsou i některé nástroje, které nám s tvorbou webové služby mohou pomoci.

V sedmé kapitole je možné nalézt informace o různých využitích webových služeb. Tyto informace byly získány prostřednictvím dotazníkového šetření. Seznam otázek a možností odpovědí lze nalézt v příloze A.

Kapitola č.8 popisuje vývoj klientské aplikace využívající webové služby IS/STAG. Aplikace kontroluje obsazenost míst na daném termínu a v případě, že se z plně obsazeného termínu odhlásí nějaký student, zašle aplikace uživateli upozornění na email. Klientská aplikace je napsána v jazyce PHP. Tato kapitola obsahuje základní informace a postup tvorby aplikace. V příloze této práce lze nalézt také uživatelskou dokumentaci klientské aplikace.

V závěru práce je uvedeno krátké zhodnocení využití webových služeb v současnosti.

2. Úvod do webových služeb

2.1 Co jsou webové služby

Jak již bylo zmíněno v úvodu této práce, webové služby (anglicky Web Services, někdy také Web API) se používají pro komunikaci mezi aplikacemi v síti. Každá aplikace může být napsána v jiném programovacím jazyce a být spuštěna pod jiným operačním systémem. Komunikace je zajištěna předáváním XML zpráv a právě díky využití otevřeného standardu XML není třeba řešit rozdíly v odlišných implementacích jednotlivých aplikací. Pro přenos zpráv je využíván především HTTP protokol, případně zabezpečený HTTPS [1].

Mezinárodní konsorcium W3C definuje webové služby takto:
„A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards“ [2].
Dle této definice je tedy webová služba softwarový systém, který umožňuje vzájemnou interakci strojů v síti. Můžeme také říci, že se jedná o technologii umožňující platformově nezávislou komunikaci vzdálených aplikací v síti. Dále dle definice je její rozhraní popsáno strojově zpracovatelným formátem (WSDL) a s ostatními systémy komunikuje převážně přes HTTP zasíláním SOAP zpráv.

Aplikace může webovou službu poskytovat (např.informace o aktuálním kurzu měn), v tomto případě hovoříme o tzv. Poskytovateli služby nebo být v roli klienta (někdy také označován jako Žadatel služeb) a zpracovávat informace z webových služeb od jednoho nebo více poskytovatelů. Takto získané informace pak klientská aplikace zpracovává a výsledky zobrazuje ve webovém prohlížeči, přes který lze také klientskou aplikaci ovládat [3].

Pojem „webové služby“ je často chápán jako souhrn služeb, které poskytují výrobci webových stránek, e-shopů apod. V této práci však bude tento termín používán výhradně ve významu technologie pro komunikaci vzdálených aplikací v síti.

2.2 Jak webové služby fungují

Nyní si stručně představíme tři základní části webových služeb:

- SOAP (Simple Object Access Protocol)
- WSDL (Web Services Description Language)
- UDDI (Universal Description Discovery and Integration)

Podrobnější popis každé části lze nalézt v kapitole č.3.

2.2.1 SOAP

Zmínili jsme se, že aplikace spolu komunikují pomocí XML zpráv, které obsahují dotazy žadatele (klienta) a odpovědi poskytovatele. Jak má komunikace vypadat nám určuje protokol SOAP, který si můžeme představit jako obálku, která má dvě části. První část je hlavička (header), ve které jsou uloženy řídíce informace přenosu (např. způsob kodování dat, informace o lokalizaci apod.). Druhou části je tělo (body), kde je umístěna vlastní zpráva s dotazem nebo odpovědí. SOAP zpráva je také popsána v XML formátu [1].

2.2.2 WSDL

WSDL (Web Services Description Language), v překladu jazyk pro popis webových služeb, je další důležitou částí celého procesu komunikace vzdálených aplikací. Abychom mohli pracovat s metodami vzdálených aplikací, je nutné znát jejich název, parametry, návrátové hodnoty a v neposlední řadě také umístění webové služby. WSDL dokument nám tedy definuje možnosti komunikace s danou službou. Dokument je vytvořen ve strojově zpracovatelném formátu XML [4].

2.2.3 UDDI

UDDI (Universal Description Discovery and Integration) je adresář, ve kterém můžeme webové služby registrovat a vyhledávat na základě jejich popisu. UDDI registr může obsahovat také informace o podniku, který webové služby nabízí (geografická poloha, oblast působnosti atd.) [5].

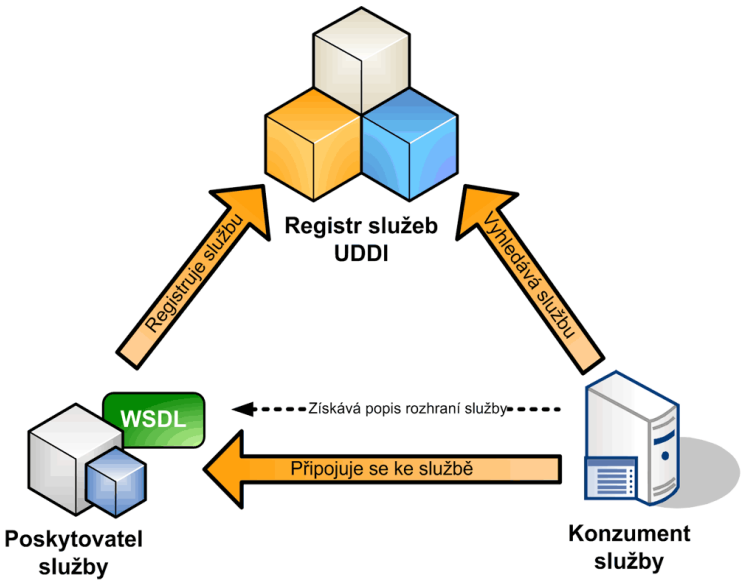
Registry mohou být veřejné nebo privátní (pouze v rámci firmy). Pokud tedy potřebuje vývojář z podniku A postavit klientskou aplikaci, která bude využívat některé webové služby podniku B, UDDI registr mu pomůže zorientovat se v možnostech, které webové

služby podniku B nabízí.

2.3 Popis komunikace

Jak spolu tři základní části webových služeb souvisí znázorňuje obrázek 2.1

- Poskytovatel zveřejní informace o webové službě v registru UDDI (není podmínkou).
- Uživatel, který chce službu využívat (Konzument), prohledává UDDI registr, nalezne požadovanou službu.
- Získá WSDL dokument s popisem rozhraní služby.
- Na základě WSDL již lze vygenerovat SOAP požadavek, kterým se přenáší XML zprávy od poskytovatele ke konzumentovi a naopak.



obrázek. 2.1 Vztah tří základních technologií (SOAP, WSDL a UDDI)

Zdroj: Webové služby a XML. In: [online]. [cit. 2013-01-21]. Dostupné z: http://www.systemonline.cz/sprava-it/webove-sluzby-a-xml.htm

2.4 Historie webových služeb

Kolem roku 1998 začala firma Microsoft pracovat na protokolu, který byl nazván SOAP a navazoval na starší XML-RPC. Postupně se přidávaly další firmy včetně IBM.

Konsorcium W3C (mezinárodní konsorcium vyvíjející webové standardy) ustavilo v roce 2000 pracovní skupinu nazvanou XML Protocol Working Group (součást uskupení Web Services Activity), která měla vývoj protokolu na starosti [4].

V současné době (leden 2013) je protokol SOAP ve verzi 1.2 doporučovaným standardem podle W3C z 27.dubna 2007 [6].

Jazyk pro popis webových služeb WSDL vznikl sloučením jazyků NASSL, SCL a SDL firem IBM, Microsoft a Ariba. Pracovní skupina pod jejímž dohledem se vyvíjel jazyk WSDL se jmenovala Web Services Description Working Group. Aktuální verze doporučovaná W3C je WSDL 2.0 [4].

Specifikace pro registr služeb UDDI byla vyvíjena organizací uddi.org, následně se připojila i společnost OASIS. V současné době (leden 2013) je aktuální verze UDDI v3 označovaná jako OASIS Standard z 3.února 2005 [7].

3. Hlavní části webových služeb

3.1. SOAP (Simple Object Access Protocol)

SOAP (Simple Object Access Protocol) představuje nejdůležitější část webových služeb. Jedná se o protokol, který nám určuje strukturu XML zpráv posílaných mezi klientem a poskytovatelem. Jak má SOAP zpráva vypadat určuje XML schema (více o využití XML ve webových službách je uvedeno v kapitole číslo 4) zpracované organizací W3C, lze ho nalézt na adrese: http://www.w3.org/2003/05/soap-envelope

SOAP zprávy jsou především zasílány přímo v HTTP požadavcích (metodou POST), což zajišťuje průchod zprávy přes firewally, jelikož v drtivé většině případů nebývá komunikace na TCP portu 80 zakázána. Komunikaci lze realizovat i prostřednictvím protokolu SMTP [5].

3.1.1 Struktura SOAP zpráv

SOAP zpráva je složena z následujících částí:

- obálka (envelope)
- hlavička (header)
- tělo (body)

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
   <n:priority>1</n:priority>
   <n:expires>2001-06-22T14:00:00-05:00</n:expires>
  </n:alertcontrol>
 </env:Header>
 <env:Body>
  <m:alert xmlns:m="http://example.org/alert">
   <m:msg>Pick up Mary at school at 2pm</m:msg>
  </m:alert>
 </env:Body>
</env:Envelope>
```

Příkad č. 3.1: Ukázka SOAP zprávy [8]

Výše uvedený kód (příklad 3.1) reprezentuje SOAP zprávu upomínky, která oznamuje příjemci, že má vyzvednout Mary ve dvě hodiny odpoledne ze školy. Celá zpráva je

uzavřena do tzv. SOAP obálky.

Envelope (obálka)

V ukázce můžeme vidět definovaný kořenový element Envelope, který patří do jmenného prostoru (jmenné prostory přiřazují jednoznačný identifikátor pro XML značky, více informací lze nalézt v kapitole č.4): http://www.w3.org/2003/05/soap-envelope/

Pokud není tento jmenný prostor zadán, SOAP zpráva nebude zpracována.

Element Envelope uzavírá všechna přenášená data SOAP zprávy (kromě příloh) do určité obálky. Obsahuje další elementy: Header a Body (první počáteční písmeno v názvu elementu je důležité, protože XML je jazyk case sensitive, tedy záleží, zda je psáno malé nebo velké písmeno). Jak můžeme vidět ze specifikace XML schema pro SOAP obálku [10], element Header není povinný a nemusí být součástí SOAP zprávy. Element Body povinný je.

Element Envelope může používat také atribut encodingStyle odkazující na další jmenný prostor pro kodování dat a to: http://schemas.xmlsoap.org/soap/encoding/

Tento atribut může používat jakýkoliv element, přitom platí, že hodnota atributu u potomka překrývá hodnotu u předka [9].

Header (hlavička)

Element Header ve zprávě SOAP je potomkem elementu Envelope a jak již bylo dříve zmíněno, SOAP zpráva nemusí element Header obsahovat. Pokud jej však obsahuje, musí být první potomek elementu Envelope.

Do této části SOAP zprávy jsou vkládány rozšiřující informace o přenášené zprávě jako například informace o lokalizaci, autentizační informace apod.

Připomeňme si část kódu z příkladu 3.1.

```
<env:Header>
 <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
  <n:priority>1</n:priority>
  <n:expires>2001-06-22T14:00:00-05:00</n:expires>
 </n:alertcontrol>
</env:Header>
```

Jako uzavírající element zde můžeme vidět element `<alertcontrol>` , ve kterém je opět definován jmenný prostor použitých elementů (více v kapitole č.4.).

Další elementy této konkrétní hlavičky obsahují informace o prioritě a době vypršení upomínky.

Potomci elementu Header mohou obsahovat atributy mustUnderstand a actor případně encodingStyle [9].

- **mustUnderstand** říká příjemci SOAP zprávy, zda musí povinně zpracovat element. Pokud má element nastaven tento atribut na hodnotu 1, je zpracování povinné. Pokud má hodnotu 0, zpracování je nepovinné.
- **actor** obsahuje odkaz na koncový bod, pro který je element určen ke zpracování. Ne vždy jsou všechny části určeny ke zpracování až konečným příjemcem.
- **encodingStyle** obsahuje odkaz na definici datových typů

Body (tělo)

Element Body je potomkem elementu Envelope a jeho výskyt v SOAP zprávě je povinný.

Tento element obsahuje vlastní XML zprávu s požadavkem nebo odpovědí konkrétní webové služby.

K popisu možného obsahu elementu Body použijeme opět část kódu z příkladu 3.1.

```
<env:Body>
  <m:alert xmlns:m="http://example.org/alert">
   <m:msg>Pick up Mary at school at 2pm</m:msg>
  </m:alert>
</env:Body>
```

Tato jednoduchá zpráva obsahuje dva elementy. Element `alert`, ve kterém je definován jmenný prostor a element `msg`, který obsahuje vlastní zprávu s odpovědí. Této SOAP zprávě mohl předcházet požadavek s dotazem na aktuální upozornění na určitý den.

Reakcí na požadavek může být také hlášení o chybě. Toto hlášení nám představuje element Fault, který je součástí elementu Body, takže struktura elementu Body s hlášením o chybě by vypadala následovně:

```
<env:Body>
...
<env:Fault>
...
</env:Fault>
```

```
</env:Body>
```
Element Fault obsahuje následující elementy v uvedeném pořadí:

- **<faultcode>** - chybový kód a jeho kategorie (*VerzionMismatch* – příjemce nerozeznal jmenný prostor elementu Envelope, *MustUnderstand* – příjemce nerozpoznal element z hlavičky s atributem mustUnderstand="1", *Client* – chyba na straně klienta, např. chybí element Body nebo jiný element, *Server* – selhalo zpracování na straně Serveru)
- **<faultstrng>** – textový popis chyby
- **<faultactor> -** obsahuje URI zdroje chyby. Tento element je nepovinný v případě, že chyba nastala u konečného příjemce zprávy. Pokud chyba nastala na cestě mezi odesílatelem a konečným příjemcem, je tento element povinný
- **<detail> -** může obsahovat libovolné atributy z libovolných jmenných prostorů, slouží k přenosu informací o chybách [9]

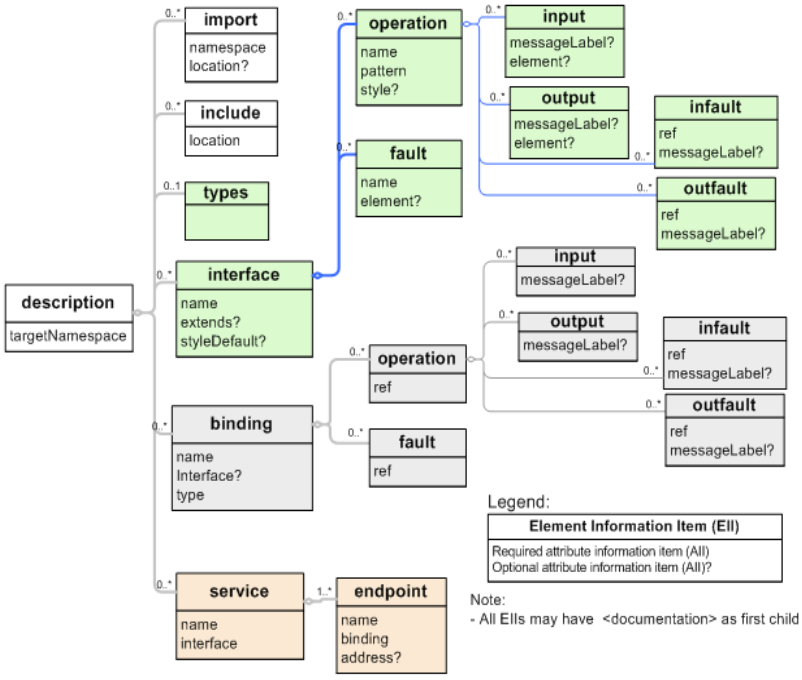
3.2 WSDL (Web Services Description Language)

WSDL je XML dokument, ve kterém jsou popsány metody, datové typy a umístění jednotlivých webových služeb. WSDL verze 2.0 je doporučení W3C z 26.června 2007 [11].

Strukturu WSDL určují především následující elementy:

- types – popisuje druh zpráv, které služba posílá a přijímá, datové typy
- interface – popisuje abstraktní operace, které služba poskytuje
- binding – popsiuje, jak službu zpřístupnit
- service – popisuje, kde webovou službu zpřístupnit [11]

Na následujícím obrázku 3.1 lze vidět kompletní schéma WSDL 2.0.



obrázek 3.1. struktura WSDL 2.0

Zdroj: W3C. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer [online]. [cit. 2013-01-13]. Dostupné z: http://www.w3.org/TR/wsdl20-primer/

Kořenový element je ve verzi 2.0 element description. Ve verzi 1.1 to byl element definitions. Verze 1.1. obsahovala mimo jiné elementy message a portType. Element message ani portType verze 2.0 již neobsahuje.

Velmi zjednodušený kód struktury WSDL vypadá takto:

<description>

<types>...definice datových typů...</types>

<interface>....definice rozhraní (operací, které lze vykonávat)...</interface>

<binding>...definice spojení (vazby) např. na http protokol ….</binding>

<service>...definice rozhraní ve spojení s koncovými body (endpoints)... </service>

</description>

V dnešní době již existuje velké množství nástrojů pro generování WSDL přímo z kódu aplikace. Lze nalézt i mnoho aplikací pro generování SOAP požadavků. Více o těchto možnostech bude řečeno v kapitole 6.

3.3 UDDI (Universal Description Discovery and Integration)

Registr UDDI si můžeme představit jako katalog webových služeb, do kterého ukládají informace o službách jejich poskytovatelé. Ukládají se především informace o umístění webové služby, jejich popis (odkaz na WSDL nebo na popis v jiném formátu), dále lze v registru nalézt také informace o majiteli služby, případně geografické informace a oblast působnosti podniku, která webové služby nabízí [5].

Základní smysl UDDI je webové služby třídit dle různých parametrů a umožnit tak snadné nalezení požadovaných informací o konkrétní službě. Většina velkých softwarových firem má svůj UDDI registr. S velikostí podniku stoupá význam používání UDDI. V případě problémů lze přes registr snadno dohledat oddělení nebo majitele zodpovědného za danou službu. Díky UDDI také nevznikají duplicitní služby. Pokud někdo potřebuje určitou službu, předtím než začne s jejím vytvářením, podívá se do UDDI registru, zda již nebyla dříve vytvořena [12].

UDDI nalezne své využití nejen v rámci jednoho podniku, ale také mezi podniky obchodních partnerů, kdy může výrazně usnadnit propojení podnikových informačních systémů.

Aktuálně uznávaný standard je UDDI v3. Za tímto standardem však nestojí organizace W3C jako tomu bylo v případě SOAP a WSDL, ale organizace OASIS (www.oasis-open.org), která navazuje na práci uddi.org [7].

UDDI v3 standard podporují firmy jako IBM, Microsoft, Novell, Oracle a mnohé další. Navíc verze 3 již nabízí podporu pro digitální podpisy [13].

3.4 REST (Representational State Transfer)

REST představuje alternativu k protokolu SOAP. REST byl poprvé představen v roce 2000 Royem Fieldingem (spoluzakladatelem protokolu HTTP). REST je architektonický styl navržený tak, aby používal přímo HTTP operace – GET, POST, PUT a DELETE [14].

REST je zaměřený spíše na data než operace. K datům se přistupuje přes konkrétní URL adresu. Tedy například URL, které vrátí seznam doporučené literatury pro předmět PPA1.https://stag-ws.zcu.cz/ws/services/rest/predmety/getLiteraturaPredmetu?

katedra=KIV&zkratka=PPA1

getLiteraturaPredmetu v tomto případě představuje konkrétní metodu, která poskytuje požadovaná data. Za znakem otazník následují parametry metody oddělené znakem ampersand (&). Jako další parametr můžeme nastavit i formát výstupu. Tím může být například XML, JSON, CSV nebo XLS [15].

Výhodou přístupu REST oproti SOAP je to, že k vyžádání operace od určité služby není třeba XML zpráva, ale pouze URL. Odpověď na požadavek může obsahovat XML zprávu, ale bez obálky, která je nutná u SOAP protokolu. REST je jednodušší způsob, jak získat požadovaná data od webové služby. Využitím protokolu REST se ušetří množství přenesených dat. Často využívaný výstupní formát REST služeb bývá JSON.

4. Webové služby a XML

V současné době stále vzrůstá potřeba výměny informací mezi různorodými systémy. XML představuje universální formát, díky kterému může být komunikace mezi systémy jednotná, otevřená a efektivní.

Výměna informací mezi systémy byla možná i bez XML. Dříve používaný formát EDI (Electronic Data Interchange), byl ale složitý a jeho implementace byla přílis nákladná [16].

XML (Extensible markup language), v překladu rozšiřitelný značkovací jazyk je webový standard od konsorcia W3C. Dokumenty XML jsou založeny na obyčejném textu – díky tomu je celý dokument snadno čitelný. Jednoduchost je však vyvážena větší datovou velikostí celého souboru.

Dokumenty XML představují držitele obsahu. Pokud chceme dokument naformátovat (například pro zobrazení v prohlížeči), můžeme použít kaskádové styly (CSS) nebo pokročilejší XSL (eXtensible Stylesheet Language).

V této kapitole si představíme, kde se XML ve webových službách používá. Blíže si popíšeme jmenné prostory a xml schémata.

4.1. XML schema

Aby byla komunikace jednoznačná a efektivní, je třeba definovat, jak má XML dokument vypadat, jaké elementy a atributy může obsahovat, v jakém pořadí atp. K definici těchto pravidel XML používá dvě metody:

- DTD (Definice typu dokumentu)
- XML schema

Pokud má dokument definována tato pravidla, a parser – zpracovávající komponenta, která je obsažena například v každém prohlížeči, narazí na chybu, dokument nebude správně zpracován.

Bez ohledu na to, zda je XML dokument vázán na nějaký definiční soubor, měl by být *správně strukturovaný* (well-formed) tzn. vyhovující základním pravidlům pro tvorbu XML dokumentů. Aby byl dokument *platný* (valid), musí vyhovět všem pravidlům

definovaným právě v DTD nebo XML schema [16].

DTD na rozdíl od XML schema používá vlastní syntaxi pro definování pravidel a ve webových službách se příliš nepoužívá, proto se budeme dále věnovat jen XML schema.

XML schema je také jeden ze standardu W3C. V příkladu 4.1 lze vidět ukázku, jak takové jednoduché XML schema vypadá.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="kniha">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="nazev" type="xs:string"/>
   <xs:element name="autor" type="xs:string"/>
   <xs:element name="stranek" type="xs:positiveInteger"/>
   <xs:element name="cena" type="xs:decimal"/>
  </xs:sequence>
<xs:attribute name="naSklade" type="xs:boolean" use="xs:required"/>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Příklad 4.1
Zdroj: upraveno dle [17]

Z ukázky je patrné, že XML schema je opět XML dokument tvořený značkami a definicemi.

4.1.1 Kořenový element

Kořenový element (element, který obaluje všechny osatní elementy) se jmenuje *schema* a patří do jmenného prostoru `http://www.w3.org/2001/XMLSchema`. Prefix označení je v tomto případě xs.

4.1.2 ComplexType a SimpleType

Následuje definice elementu kniha. To, že bude element obsahovat další elementy značí zápis `<xs:complexType>...</xs:complexType>`, do kterého jsou uzavřeny jednotlivé další části.

Naopak tzv. Simple Types – jednoduché typy jsou takové, které již neobsahují žádný další element. V našem případě by to byly elementy *nazev, autor, stranek* i *cena*.

4.1.3 Sequence

Pokud se podíváme na příklad 4.1 vidíme definici elementu `<sequence>`. To znamená, že veškeré elementy, které následují v této definici musí být v takto zachovaném pořadí definovány i ve výsledném XML dokumentu.

4.1.4 Datové typy

XML schema dovoluje definovat také datový typ. Například v ukázce je uvedena definice elementu nazev takto:

```
<xs:element name="nazev" type="xs:string"/>
```

Znamená to, že v elementu název se může vyskytovat jakýkoliv řetězec znaků.
XML schema dovoluje definovat mnoho datových typů jako například:

- string – řetězec znaků
- boolean – logická hodnota true nebo false (0 nebo 1)
- decimal – číslo s desetinou čárkou
- positiveInteger – kladné celé číslo (bez 0)
- date – datum ve tvaru RRRR-MM-DD [17]

a další včetně možnosti definice vlastních typů, které jsou odvozené od existujících. [16]

4.1.5 Atributy

V ukázce lze vidět také definovaný atribut typu boolean „naSklade“, který náleží elementu kniha:

```
<xs:attribute name="naSklade" type="xs:boolean" use="xs:required"/>
```

Deklarace use=“required“ znamená, že vyplnení atributu je vyžadováno.

4.1.6 Výsledný dokument podle ukázkového schématu

Výsledný XML dokument, který by splňoval definici tohoto XML schema může vypadat následovně:

```
<?xml version="1.0"?>
<book naSklade="true">
    <nazev>Název knihy</nazev>
    <autor>Jméno autora</autor>
    <stranek>156</stranek>
    <cena>200</cena>
</book>
```

Možnosti XML schema jsou mnohem rozsáhlejší a převyšují rozsah této práce.

4.2 XML schema ve webových službách

XML schema webové služby hojně využívají například v samotném SOAP požadavku nebo odpovědi, kdy schema definuje mj. celou strukturu obálky, hlavičky i těla. Toto XML schema lze nalézt na adrese: http://schemas.xmlsoap.org/soap/envelope/.

Povolenou strukturu WSDL dokumentu definuje také samostatné XML schema. Definici lze prohlédnout zde: http://schemas.xmlsoap.org/wsdl/.

V neposlední řadě lze XML schema použít také na definici parametrů a návratových hodnot, které se budou následně používat v konkrétní aplikaci a které můžeme přenášet v SOAP zprávách.

4.3 XML jmenné prostory (Namespaces)

Pokud potřebujeme spojit dva různé XML dokumenty, může se stát, že v každém z nich se nachází element se stejným jménem, ale zcela jiným významem. V takovém případě by došlo ke konfliktu a nebylo by zřejmé, který element bylo zamýšleno použít. Například jeden dokument může obsahovat element <cena> vztahující se k jednomu výrobku a druhý dokument může obsahovat také element <cena>, který se ale bude vztahovat ke zcela jinému výrobku. Pokud by byly dokumenty používány samostatně, nic by se nestalo, ale problém by mohl nastal v případě, kdy by bylo potřeba sjednotit dokumenty, tzn. používat značky z obou dokumentů [17 str. 89].

K rozlišení jednotlivých elementů a atributů nám slouží právě jmenné prostory. Na následující upravené ukázce si vše ukážeme a popíšeme.

```
<COLLECTION
      xmlns:book = „http://www.example.com/books“
      xmlns:cd = „http://www.example.com/cd“>
      <book: ITEM >
            <book: TITLE> Nazev knihy</book:TITLE>
            <book: PRICE> 250 Kč</book:PRICE>
      </book:ITEM>
      <cd:ITEM>
            <cd:TITLE>Název cd</cd:TITLE>
```

```
            <cd:PRICE>Název cd</cd:PRICE>
      </cd:ITEM>
</COLLECTION>
```

Příklad 4.2 upraveno dle [17 str. 91]

Jmenný prostor je zde deklarován v počáteční značce pomocí atributu xmlns: . Konkrétně tedy xmlns:book = „http://www.example.com/books“, kde místo book může být jakýkoliv jiný prefix (předpona) s výjimkou vyhrazených začínajících na „xml“. Za znakem rovná se následuje jednoznačný identifikátor jmenného prostoru [17].

Jakmile byl jmenný prostor definován, můžeme ho již používat u konkrétních elementů nebo atributů. V příkladu 4.2 lze vidět definici dvou jmenných prostorů s prefixem book a cd. Následně jsou také jednotlivé prostory přiřazeny ke konkrétním elementům a ačkoliv se elementy jmenují stejně (ITEM, TITLE, PRICE), jmenné prostory přesně rozliší jejich význam (souvislost).

Definovat lze také výchozí jmenný prostor, který se vztahuje ke všem elementům, které obsahuje a které nejsou zahrnuty do jiného jmenného prostoru. Definice výchozího prostoru vypadá takto:

```
<COLLECTION xmlns = „http://www.example.com/books“
      xmlns:cd = „http://www.example.com/cd“>
```

tzn. u výchozího jmenného prostoru neuvádíme prefix. Nutno říci, že jmenné prostory lze přepisovat definováním jiného prostoru v konkrétním elementu [17].

Jmenné prostory používá v definicích WSDL i SOAP.

5. Webové služby v praxi

Na webu lze najít velké množství webových služeb, které poskytují například funkce pro zjištění počasí v konkrétní lokalitě, službu pro převod měn nebo také informaci o tom, zda je požadované doménové jméno volné či nikoliv.

V dnešní době, kdy je kladen důraz na zefektivňování podnikových procesů a snadné získávání informací, představují webové služby účinný nástroj, jak rychle tyto potřebné informace získat. Proto je velmi užitečné, když firmy nabízejí služby pro své obchodní partnery, které mohou poskytovat informace o sortimentu zboží jako jsou aktuální ceny, termíny dostupnosti apod. Žadatel těchto služeb může volání služeb integrovat do vlastního informačního systému, takže v případě potřeby uživatel žádající o informace pouze klikne a zobrazí se mu požadovaná informace. Kdyby podnik služby neposkytoval nebo obchodní partner neměl implementovánu klientskou část, musely by se informace o dostupnosti získávat jinými prostředky jako například telefonem, emailem apod. Tento způsob by byl ale značně neefektivní a docházelo by i ke zpoždění procesů s touto informací souvisejících.

5.1 Webové služby nad IS/STAG

IS/STAG je studijní informační systém, který obsahuje informace o předmětech, studentech, místnostech a jejich obsazení, termínu zkoušek a mnoho dalšího. Nad IS/STAG je definováno několik služeb, které lze využívat. K využití některých služeb je vyžadována autentizace uživatele.

Některé stránky kateder požadují zobrazení informací ze systému IS/STAG na svých webových stránkách. Webové služby nad IS/STAG jsou na tyto požadavky připraveny a je tedy možné stáhnout aktuální informace a zobrazovat je na stránkách kateder [18].

Přístup ke službě je možný přes protokol SOAP a pro snadnější přítup byly služby implementovány také jako REST.

Prozatím není zprovozněn žádný registr UDDI, který by webové služby shromažďoval a třídil. K dispozici je stránka, kde je uveden přehled služeb a základní informace o jejich použití.

Seznam služeb IS/STAG lze nalézt zde: https://stag-ws.zcu.cz/ws/help/list.

5.1.1 Formáty výstupů

Kromě standardního výstupu zprávy v XML byl v přístupu REST implementován také export do různých formátů dat. Konkrétně je možné exportovat mj. do následujících formátů: [15]

- XLS (formát pro MS Excel)
- ICS (formát pro uložení kalendářových dat)
- JSON (javascriptový objektový zápis)

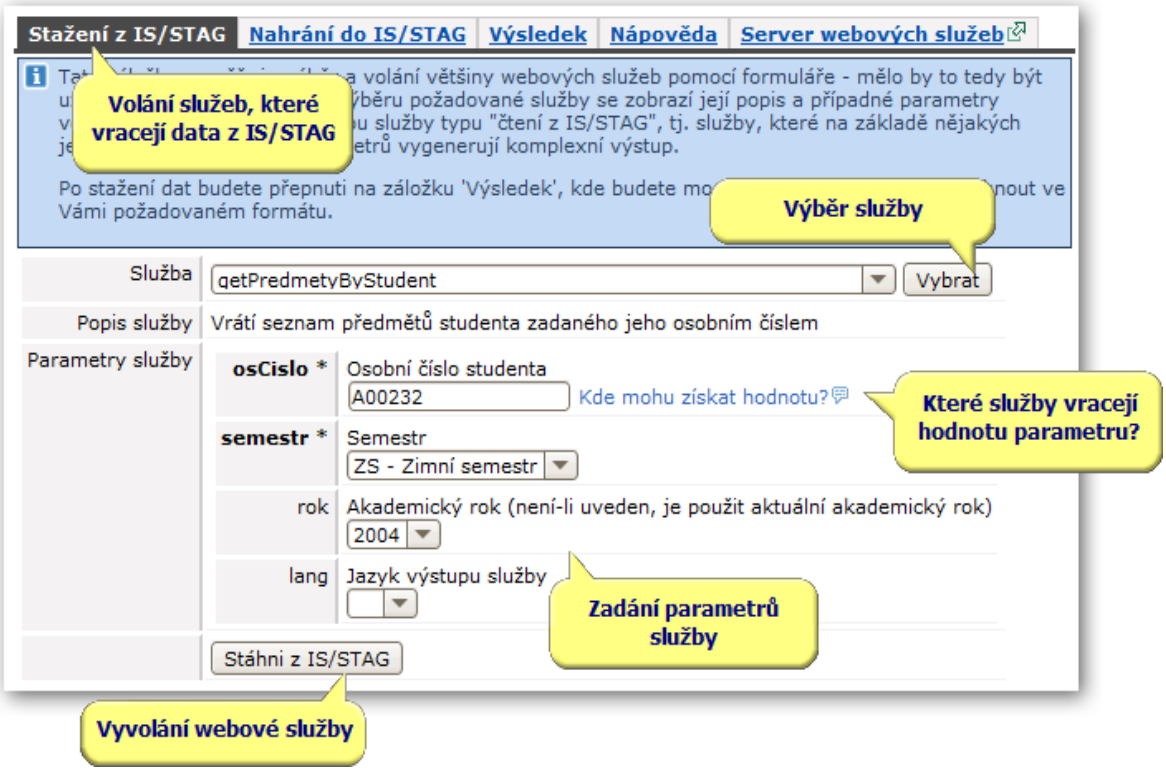
Formát lze zvolit přidáním parametru outputFormat do URL. Tedy například takto:

```
/ws/services/rest/kalendar/getHarmonogramRoku?
rok=2007&outputFormat=XLS
```
[15]

Tato služba poskytne harmonogram roku 2007 ve formátu XLS.

5.1.2 Uživatelské rozhraní

Pro uživatelské pohodlí jsou připravené formuláře (obrázek 5.1), ve kterých uživatel může po přihlášení vyhledat z rozbalovacího seznamu požadovanou službu a pomocí dalších formulářových prvků zadat požadované parametry. Výstupem je odpověď webové služby v několika výše uvedených formátech. Kromě stahování dat ze systému je také možné jejich nahrávání. To se hodí například při hromadném zapisování zápočtu do systému. Vstupem bývá XML nebo CSV dokument s daty, které mají být zapsány [19].



obrázek. 5.1 Ukázka formuláře pro volání webové služby

Zdroj: Webové rozhraní pro vybrané služby nad IS/STAG. In: [online]. [cit. 2013-01-28]. Dostupné z: https://stag-ws.zcu.cz/webaccess/main?page=help

5.1.3 Zabezpečení

K přístupu k některým webovým službám IS/STAG je třeba autentizace uživatele. Aby nebyl uživatel nucen zadávat své přihlašovací údaje do různých (potenciálně nedůvěryhodných) míst, probíhá proces autentizace na důvěryhodném serveru velmi podobném, který zajišťuje klasické přihlášení do systému. Uživateli se zobrazí formulář pro zadání jména a hesla. Po vyplnění a úspěšném ověření je mu vrácen výsledek služby [15].

5.2 Webové služby Google

Google je společnost známá především pro její vyhledávací služby. Lze říci, že samotné vyhledávání, tedy poskytování odkazů na webové stránky s informacemi, které uživatel hledá je jedna velká webová služba. Google řadí výsledky vyhledávání dle svého

tajného algoritmu a snaží se o poskytování co nejrelevantnějších výsledků k danému dotazu. Uživatel tedy napíše klíčové slovo, které hledá, odešle dotaz, Google prohledá své databáze, dle algoritmu výsledky seřadí a výsledek zobrazí uživateli ve webovém prohlížeči.

K webovým službám Google je třeba se přihlásit a získat licenční klíč. Licence opravňuje držitele k 1000 bezplatným vyhledáváním denně. Tímto opatřením chce Google předejít přetížení jeho strojů díky cizím aplikacím [20].

Od roku 2006 Google nedoporučuje používat pro vyhledávací API protokol SOAP [21].

5.2.1 Google Maps

Jako příklad webových služeb poskytovaných společností Google si nyní představíme webové služby Google Maps.

Jako přístup ke službám je používán REST tzn. HTTP požadavek, kde přímo v URL jsou předávány parametry dotazu. Formát odpovědi na požadavek je buď XML nebo JSON [22].

Nyní si uvedeme příklad z Google Geocoding API. Geocoding je proces, kdy ze zadané adresy dokážeme získat zeměpisné souřadnice jako zeměpisnou šířku a výšku. S takto získanými daty můžeme dále pracovat. Požadavek na XML data vypadá následovně [23]:

http://maps.googleapis.com/maps/api/geocode/xml?
address=8+Univerzitní+Plzeň&sensor=false

(pro snadnější demonstraci není URL kódováno). Reakcí na tento požadavek je vrácení XML souboru s údaji viz zkrácený příklad 5.1.

```
<GeocodeResponse>
<status>OK</status>
<result>
<address_component>
<long_name>301 00</long_name>
<short_name>301 00</short_name>
<type>postal_code</type>
</address_component>
<address_component>
```

```
<long_name>Plzeň 1</long_name>
<short_name>Plzeň 1</short_name>
<type>postal_town</type>
</address_component>
<geometry>
<location>
<lat>49.7236682</lat>
<lng>13.3516017</lng>
</location>
<location_type>ROOFTOP</location_type>
</geometry>
</result>
</GeocodeResponse>
```

Příklad 5.1

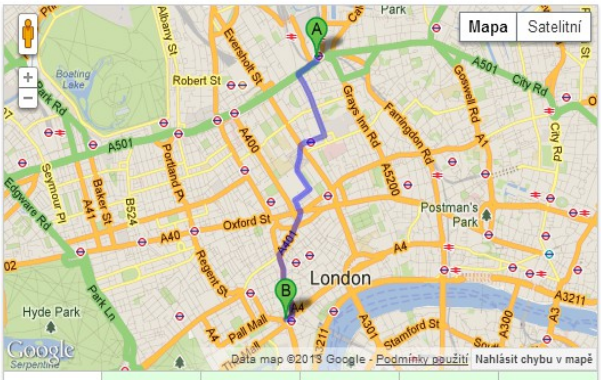
Zdroj: upraveno dle [23]

Výstupní formát dat lze velmi jednoduše změnit na JSON. Pouze se v URL místo „xml“ zapíše „json“ viz. následující odkaz: [23]

http://maps.googleapis.com/maps/api/geocode/json?address=8+Univerzitní+Plzeň&sensor=false

Google Maps poskytují velice propracované aplikační rozhraní k získání geografických dat jako vzdálenost, směr, nadmořská výška apod. Dokumentaci lze nalézt na adrese: https://developers.google.com/maps/documentation/webservices/. S těmito údaji lze budovat aplikace, které dokáží například počítat ujetou vzdálenost na kole a zobrazovat ji na mapě. Využití webových služeb Google Maps v konečných aplikacích může být široké a velmi zajímavé.

Na obrázku 5.2 lze vidět ukázkovou aplikaci, která spočítala vzdálenost místa A od místa B, ukázala dané body na mapě a spočítala přibližný čas za jak dlouho se z bodu A do bodu B člověk dostane.



	Buckingham Palace	Houses of Parliament	Tower Bridge	Trafalgar Square	Westminster Abbey
Euston	2,3 mil 47 min	2,2 mil 44 min	3,5 mil 1 hodin, 11 minut	1,7 mil 34 min	2,3 mil 47 min
Kings Cross	2,7 mil 54 min	2,5 mil 51 min	3,2 mil 1 hodin, 4 minut	2,0 mil 41 min	2,6 mil 54 min
Liverpool St	3,1 mil 1 hodin, 3 minut	2,7 mil 55 min	1,2 mil 24 min	2,4 mil 49 min	2,8 mil 57 min
Paddington	2,1 mil 41 min	3,0 mil 59 min	5,2 mil 1 hodin, 46 minut	2,7 mil 54 min	2,9 mil 58 min
St. Pancras	2,6 mil 51 min	2,4 mil 49 min	3,2 mil 1 hodin, 4 minut	1,9 mil 38 min	2,5 mil 51 min
Victoria	0,5 mil 10 min	0,9 mil 18 min	3,4 mil 1 hodin, 10	1,2 mil 25 min	0,7 mil 14 min

obrázek. 5.2 Google Maps API

Zdroj: GOOGLE INC. Google Maps API [online]. [cit. 2013-04-13]. Dostupné z: https://developers.google.com/maps/location-based-apps

5.3 Twitter

Sociální síť Twitter poskytuje aplikační rozhraní, které také jako Google mapy používá protokol REST. Nejčastější formát dat, který je službou vrácen je JSON [26].

Služby jsou rozděleny do několika skupin jako například:

- tweety (příspěvky uživatelů)
- timeline (strana patřící uživateli, kde se zobrazují tweety)
- trendy
- hledání
- uživatelé

Služby poskytují například vrácení soukromých zpráv poslaných konrétnímu uživateli (po přihlášení uživatele). Parametr může být například počet vrácených zpráv.

Další služba dokáže vrátit konkrétní soukromou zprávu uživatele následujícím voláním:

https://api.twitter.com/1.1/direct_messages/show.json?id=87424932, kde jako parametr je zadáno id konkrétní zprávy. Výstupní formát je opět JSON. Většina služeb dostupných přes REST API v1.1 vyžaduje autentizaci uživatele [26].

Kompletní přehled služeb lze nalézt na tomto odkazu: https://dev.twitter.com/docs/api/1.1

5.4 Amazon

Amazon web services je souhrný název pro cloudové služby poskytované společností Amazon. Nabídka služeb je široká a pokrývá služby od oblasti e-commerce, datových uložišť až po mobilní aplikace. Rozdíl oproti doposud uvedeným službám je ten, že většina z Amazon služeb není zdarma.

Jako příklad jedné z mnoha Amazon služeb si uvedeme datové uložitšě S3 (Amazon Simple Storage Service). Tato služba nabízí přístup přes SOAP i REST. Kromě HTTP zde existuje podpora také pro stahování souborů pomocí BitTorrentu. To může být užitečné pokud potřebujeme distribuovat větší množství dat. Data jsou také šifrována a přístup k nim může uživatel řídit [27].

5.5 Heureka.cz

Heureka.cz je cenový porovnávač, který shromažďuje informace o produktech, které jsou do její databáze exportovány elektronickými obchody. Export zboží probíhá pomocí XML. Popis specifikace exportovaného XML souboru lze nalézt na této adrese: http://sluzby.heureka.cz/napoveda/xml-feed/

Exportované zboží je poté roztříděno podle různých parametrů (především ceny) a zobrazeno uživateli, který si tak může objednat zboží v nejlevnějším obchodě. Heureka také shromažďuje hodnocení od nakupujících zákazníků obchodu a podle tohoto hodnocení také řadí výsledky svého hledání.

Komunikace mezi Heurekou a elektronickým obchodem funguje oboustranně. Obchod dovoluje Heurece získávat informace o dostupnosti, možností dopravy apod. Naproti tomu Heureka může od obchodu požadovat informace zda byla například připsána platba za objednávku [25].

Aplikační rozhraní Heureka je založené na architektuře REST a jako formát výstupu používá JSON. Je také vyžadována komunikace zabezpečeným protokolem

HTTPS [25].

Příklad volání aplikačního rozhraní obchodu pro zjištění dostupnosti zboží vypadá takto:

https://www.exaple.com/api/1/products/availability, kde www.example.com je adresa obchodu. Jako parametry lze uvést pole produktů, případně jejich id nebo počet objednaných kusů [25].

Kompletní popis aplikačního rozhraní lze nalézt na adrese:

http://sluzby.heureka.cz/napoveda/kosik-api/.

6. Průvodce vytvořením nové webové služby

6.1 Vývoj a nástroje

První fáze vývoje webové služby je napsání kódu aplikace, která bude následně volána jako webová služba. Připomeňme si největší výhodu webových služeb - platformovou nezávislost. Díky tomu není třeba držet se konkrétního programovacího jazyka a je tak zcela na tvůrci služby, jaký jazyk zvolí.

Kód aplikace lze napsat i v poznámkovém bloku, ale v dnešní době již existuje velké množství propracovaných vývojových prostředí s mnoha rozšířeními a podporou webových služeb, které budou jistě vhodnější volbou. Takovým vývojovým prostředím, které je zdarma může být například NetBeans nebo Eclipse.

S vývojem webových služeb může pomoci Apache CXF. Je to open source framework, který dokáže například generovat WSDL z Java třídy a naopak. Na Internetu lze najít velké množství online generátorů WSDL z kódu aplikace pro různé programovací jazyky.

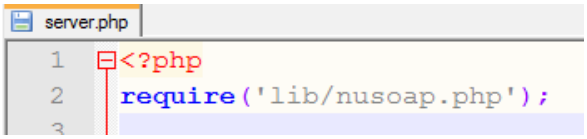
K běhu webové služby je třeba server, na kterém je webová služba nasazena a je schopna přijímat a zpracovávat požadavky od různých klientů.

6.2 Vytvoření webové služby v PHP

Nyní si ukážeme, jak lze vytvořit jednoduchou webovou službu v jazyce PHP. Použijeme k tomu zdarma dostupnou knihovnu NuSOAP, která umí mj. dynamicky generovat WSDL. [28]

Otevřeme si textový editor jako například Notepad++ nebo jakékoliv IDE (Eclipse, NetBeans atd.), a nový soubor uložíme jako server.php

Nejdříve je třeba knihovnu NuSOAP stáhnout (lze z adresy: http://sourceforge.net/projects/nusoap/) a vložit do našeho kódu. To lze provést příkazem require viz obrázek 6.1.



obrázek 6.1 Vytvoření jednoduché PHP webové služby

zdroj: upraveno dle [28]

Obrázek 6.2 ukazuje inicializaci jmenného prostoru (řádek 5), vytvoření nového objektu soap_server a následnou konfiguraci WSDL přiřazením jména služby (AplikacePozdrav) a jmenného prostoru.

```
4  /*jmenny prostor*/
5  $ns="http:/localhost/_SKOLA/web_service/";
6  $server = new soap_server();
7
8  /*nazev sluzby, jmenny prostor*/
9  $server->configureWSDL('AplikacePozdrav',$ns);
10 $server->wsdl->schemaTargetNamespace=$ns;
```

obrázek 6.2 Vytvoření jednoduché PHP webové služby

zdroj: upraveno dle [28]

Dalším krokem je registrace funkce a jejích vstupních a výstupních parametrů (řádek 12 – 14). Na obrázku 6.3 můžeme vidět funkci pozdrav, která má vstupní parametr jmeno, typu string a výstupem této funkce je také string, konkrétně pozdrav zadaného jména. Následuje zapnutí naslouchání služby pro přijimání požadavků (řádek 22) [28].

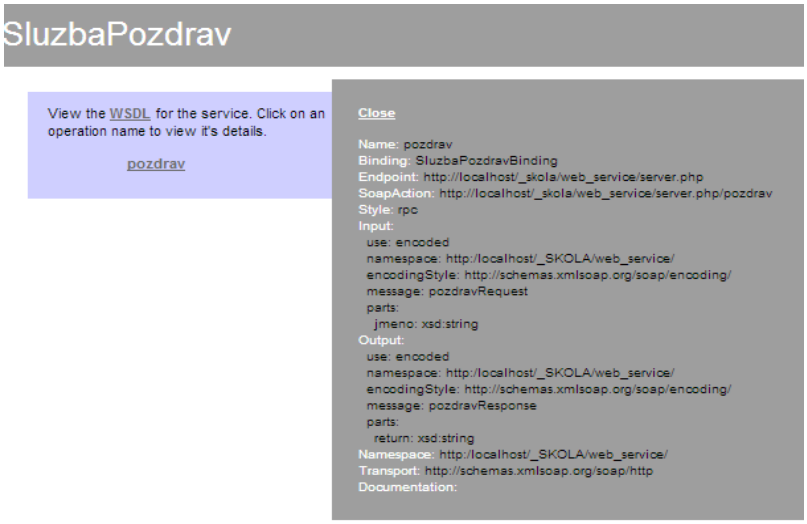
```
12 $server->register('pozdrav',
13 array('jmeno' => 'xsd:string'),
14 array('return' => 'xsd:string'),
15 $ns);
16
17 function pozdrav($jmeno){
18 $vysl='Hello '.$jmeno;
19 return new soapval('return','string',$vysl);
20 }
21
22 $server->service($HTTP_RAW_POST_DATA);
```

obrázek 6.3 Vytvoření jednoduché PHP webové služby

zdroj: upraveno dle [28]

Celý dokument byl uložen jako server.php. Tímto skončila část psaní kódu serveru neboli poskytovatele webové služby. Aby bylo možné službu zavolat, musíme ji umístit na webový server. V této ukázce použijeme server lokální a všechny soubory uložíme do následujícího umístění: http://localhost/_skola/web_service/server.php. Po

napsání této URL do adresního řádku prohlížeče, dostaneme výsledek, který lze vidět na obrázku 6.4. Jedná se o jednoduchý popis webové služby spolu s odkazem na WSDL dokument.



Obrázek 6.4 Vytvoření jednoduché PHP webové služby

zdroj: upraveno dle [28]

Výhodou knihovny NuSOAP je automatické generování WSDL. Stačí tedy přidat ?wsdl za název souboru (http://localhost/_skola/web_service/server.php?wsdl) a dostaneme kompletní WSDL dokument s popisem služby [28].

V této ukázce jsme si předvedli vytvoření a nasazení jednoduché webové služby v jazyce PHP. Postup je podobný i pro jiné programovací jazyky. Využívají se však jiné knihovny, nástroje a servery.

7. Zhodnocení využití webových služeb v současnosti

Příklady některých služeb jsme si uvedli v kapitole č. 5. Abychom získali další informace o využití webových služeb v současnosti, byl vytvořen a rozeslán dotazník.

7.1 Cíl dotazníku

Cílem dotazníku bylo zjistit konkrétní využití technologie webových služeb v praxi. Mezi dílčí cíle tohoto dotazování patří zjistit:

- jak často je po firmách, vyvíjející zakázkový software, požadována implementace webových služeb
- velikost firem, které nejčastěji vyžadují implementaci webových služeb
- zda je více používán SOAP nebo REST
- jak často je využíván registr služeb UDDI
- jaké výstupní formáty dat převládají

7.2 Výběr respondentů a způsob dotazování

Celkem bylo osloveno prostřednictvím emailu 300 firem z celé ČR, které mají v popisu své činnosti vývoj zakázkového softwaru. Dotazník byl směrován na vývojové oddělení firmy.

Celkem 57 firem vyplnilo dotazník (19%). Celý dotazník byl anonymní – vyplněné údaje nebyly spojovány s konkrétní firmou.

Technicky byl sběr dat a prezentace výsledků realizován prostřednictvím formuláře v Google Disku.

Při tvorbě dotazníku byl kladen důraz na krátký čas vyplnění. Dotazník obsahuje celkem 7 otázek (otázky a možnosti odpovědí lze nalézt v Příloze A). Odhad doby vyplnění je přibližně 5 minut.

7.2.1 Zvolené škály

Pokud bylo na nějaké škále třeba vyjádřit, jak často je používán registr UDDI nebo jak často je implementace webových služeb vyžadována, byla použita následující pěti

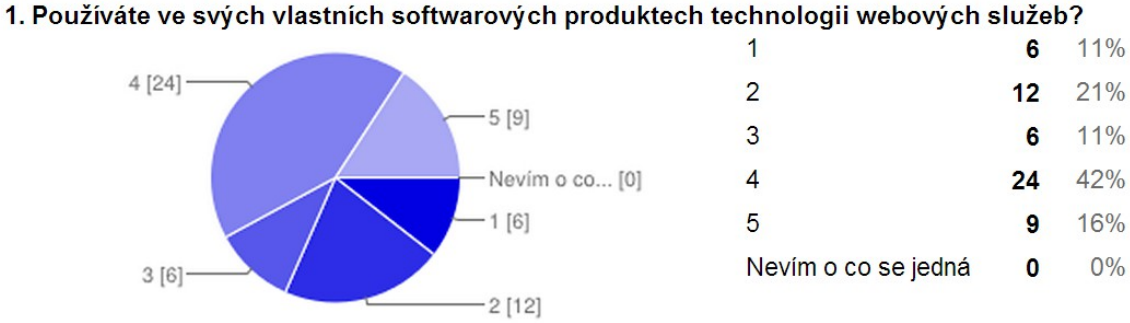
stupňová škála spolu s možností „nevím, o co se jedná“.

- 1 – nikdy (v 0% případů)
- 2 – (v 25% případů)
- 3 – (v 50% případů)
- 4 – (v 75% případů)
- 5 – vždy (v 100% případů)

7.3 Výsledky dotazníku

U každého následujícího obrázku v této kapitole jsou v levém sloupci možnosti odpovědí, uprostřed absolutní četnost dané odpovědi a vpravo její procentuální vyjádření. Otázky č.3, 5, a 7 byly nepovinné a u otázek č. 3, 4 a 6 bylo možné vybrat více odpovědí.

Na obrázku 7.1 lze vidět, že softwarové firmy jsou s technologií webových služeb dobře seznámeni. Nikdo nezvolil odpověď *„nevím o co se jedná“*. Celkem 42 % respondentů uvádí, že webové služby používají v 75 % případů v některých svých softwarových produktech.

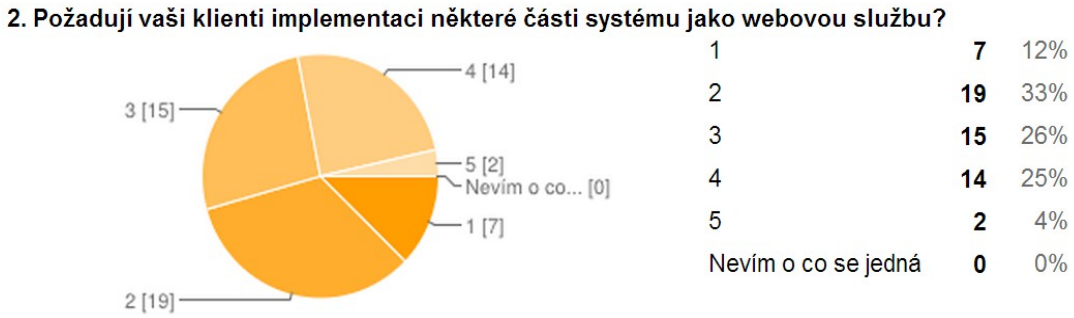


obrázek 7.1

Zdroj: Vlastní dotazníkové šetření, zpracováno službou Google Disk

Situace z pohledu klientů softwarových firem není už tak jednoznačná (obrázek 7.2). Firmy znají tuto technologii, vyžadují ji spíše méně. U této otázky nelze udělat

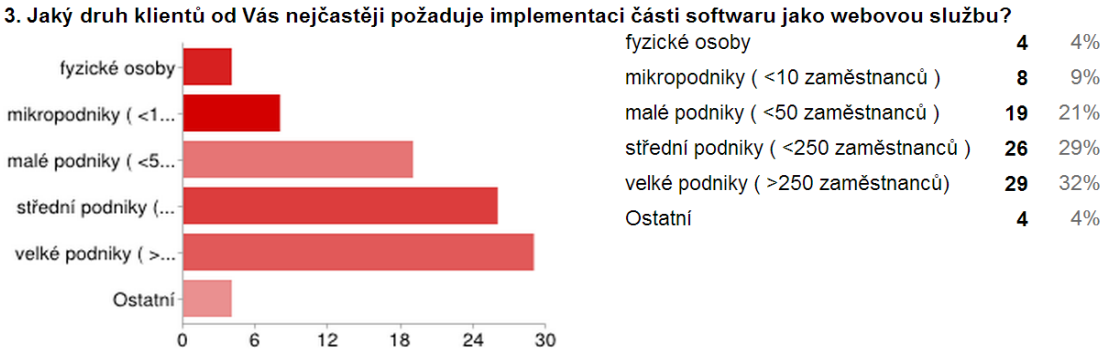
jednoznačný závěr, protože u některých softwarových produktů nemusí být webové služby potřeba.



obrázek 7.2

Zdroj: Vlastní dotazníkové šetření, zpracováno službou Google Disk

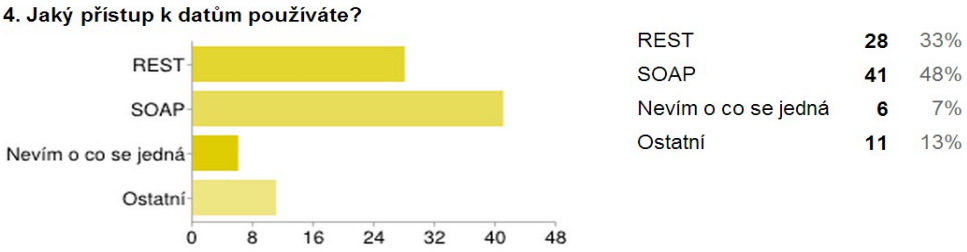
Výsledky z otázky č.3 zobrazené na obrázku 7.3 ukazují, že čím větší podnik je (dle počtu zaměstnanců), tím větší má potřebu využít technologii webových služeb. Může to být dáno hlavně tím, že větší podniky potřebují organizovat větší množství svých zdrojů, dodavatelů, odběratelů apod. Je zřejmé, že velký podnik bude mít mnohem větší nároky na spolupráci různých částí informačních systémů než malý podnik nebo dokonce živnostník, jehož informační systém může tvořit pouze tabulka v Excelu. Jako odpověď *„ostatní“* bylo uvedeno také *„školy“* a *„státní správa“*. Tato otázka dovolovala vybrat více odpovědí.



obrázek 7.3

Zdroj: Vlastní dotazníkové šetření, zpracováno službou Google Disk

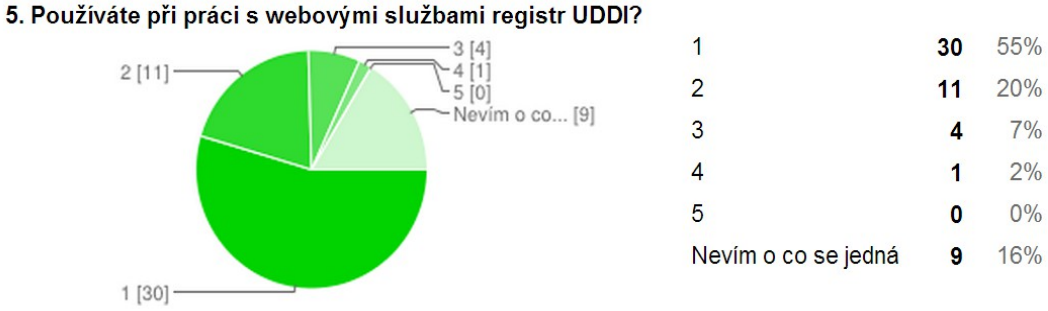
Vzhledem k výsledkům z otázky č.3 bylo možné očekávat, že firmy budou preferovat implementaci webových služeb pomocí SOAP. Může to být dáno tím, že k propojení větších systémů je vyžadován propracovaný standard podle WC3. Webové služby podle REST ale díky své jednoduchosti mají také poměrně vysoké zastoupení viz obrázek 7.4. Jako odpověď *„ostatní“* bylo uvedeno například: WCF (Windows Communication Foundation), XML-RPC, AJAX nebo CORBA. U této otázky bylo možné zaškrtnout více odpovědí.



obrázek 7.4

Zdroj: Vlastní dotazníkové šetření, zpracováno službou Google Disk

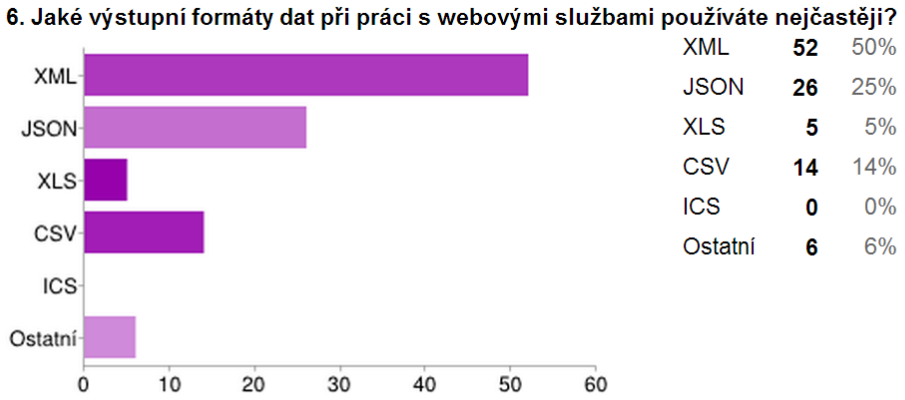
Zajímavou informaci dokládá obrázek 7.5. Celkem 55 % respondentů nikdy nevyužilo registr UDDI. Potvrzuje to skutečnost, že UDDI registry se příliš neujaly. Veřejných UDDI registrů, které jsou aktuální, je na Internetu velmi málo. Využití mohou najít uvnitř větších podniků.



obrázek 7.5

Zdroj: Vlastní dotazníkové šetření, zpracováno službou Google Disk

Obrázek 7.6 potvrzuje, že nejpoužívanějším formátem dat je XML. U této otázky bylo možné zaškrtnout více formátů. Celkem 52 z 57 respondentů uvedlo jako nejpoužívanější formát dat právě XML. Následuje JSON (javascriptový objektový zápis), který uvedlo celkem 26 respondentů.



obrázek 7.6

Zdroj: Vlastní dotazníkové šetření, zpracováno službou Google Disk

Poslední otázka v dotazníku byla otevřená a snažila se zjistit konkrétní příklad použití webových služeb v praxi. Zde si uvedeme pouze deset uvedených příkladů.

- Export objednávek do účetního systému
- Kontrola placení poplatků
- Komunikace mezi čtečkou čárových kódů a ERP systémem
- Zobrazení dat z měřících systémů v mobilních zařízeních
- Odesílání informací o balících na PPL
- Import zboží dodavatele
- Zjištění aktuální teploty na sjezdovce
- Napojení na elektronické bankovnictví České spořitelny
- Předávání dopravních dat mezi heterogeními systémy
- Reporting vedení

Dotazníkové šetření ukázalo, že softwarové firmy ve svých produktech často využívají technologii webových služeb. Ukázalo se, že implementaci některé části systému jako webovou službu požadují především velké a střední firmy, které mají větší nároky na organizaci svých zdrojů a obchodních partnerů. Z šetření vyplynulo, že k implementaci webových služeb je více využíván protokol SOAP. Služby založené na REST jsou ale také využívány velmi často. Potvrdilo se, že registr služeb UDDI je využíván velmi málo. Více než polovina respondentů ho nevyužívá vůbec. Nejpoužívanější výstupní formát dat je XML, následuje JSON a CSV.

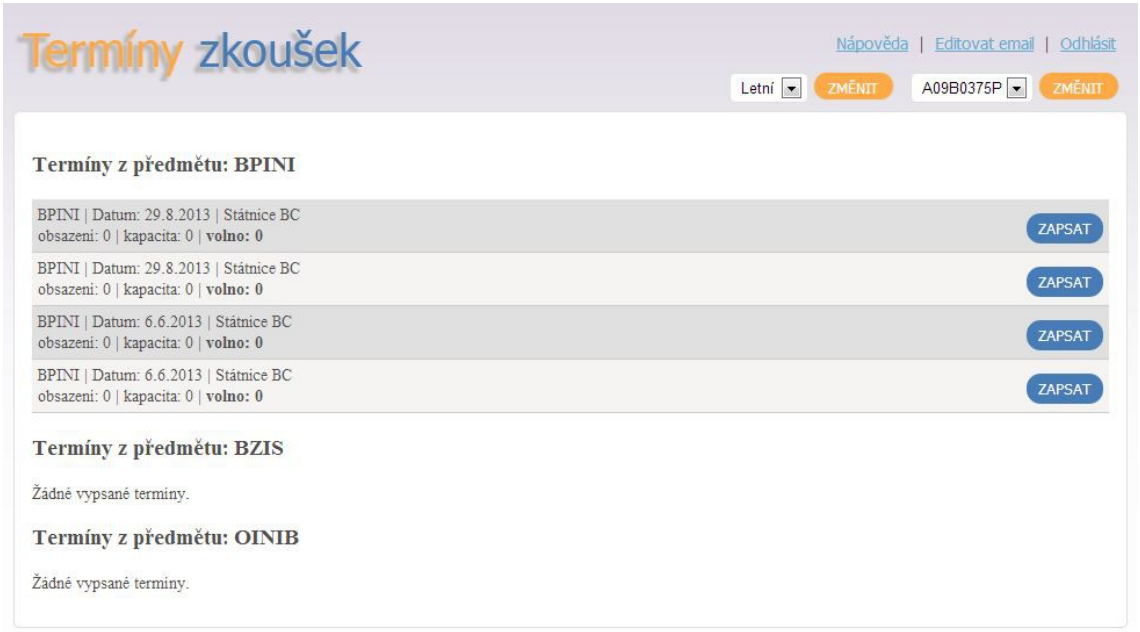
8.Klientská aplikace

V rámci této práce byla vytvořena aplikace, která pomocí webových služeb dokáže upozornit studenta na uvolněné místo na zkouškovém nebo zápočtovém termínu. V aplikaci lze nastavit sledování i více termínů. Upozornění probíhá prostřednictvím emailu. Aplikace byla napsána v jazyce PHP.

8.1 Jak aplikace funguje

Po spuštění aplikace na určité URL se uživateli objeví obrazovka s odkazem na přihlášení. K aplikaci je nutné se přihlásit svojí ORION identitou. Většina použitých webových služeb je dostupná i bez přihlášení, ale právě povinnost přihlašování k určité webové službě se může kdykoliv změnit.

Pokud bylo přihlášení úspěšné, uvidí uživatel výpis termínů roztříděných dle předmětů, které má zapsané (obrázek 8.1). Lze přepínat mezi semestry i osobními čísly.



obrázek 8.1

Zdroj: vlastní

Aby bylo možné odesílat emailové upozornění, musí uživatel nejdříve nastavit emailovou adresu a uložit ji.

Pokud je nějaký termín plně obsazen, tzn. kapacita termínu se rovná aktuální

obsazenosti, objeví se u termínu tlačítko sledovat (obrázek 8.2). Pokud jej stiskneme, uloží se daný termín ke sledování.

obrázek 8.2

Zdroj: vlastní

Aplikace předpokládá nasazení na nějaký druh automatického plánovače (CRON). Díky tomu je možné periodicky kontrolovat obsazenost na termínu.

Podrobný návod, jak aplikace funguje, lze nalézt v Příloze B.

8.2 Popis implementace

Skriptovací část aplikace byla napsána v jazyce PHP. K uložení dat byly zvoleny čtyři .txt soubor (viz kapitola 8.2.1). Prezentační část byla vytvořena v XHTML a CSS 3. Popis všech použitých funkcí lze nalézt v Příloze C.

8.2.1 Uložení dat

Při implementaci byl kladen důraz na jednoduché nasazení a tak k uložení dat byl zvolen .txt soubor. Uložení dat by zajistila také relační databáze MySQL nebo XML soubor. Vzhledem k potřebě uložení pouze malého množství dat, bylo uložení v .txt souboru prozatím dostačující. V případě rozšíření aplikace k více uživatelům by bylo vhodnější zvolit ukládání dat do jediného XML souboru. Tento XML soubor může mít následující strukturu:

```
<student>
<osCislo>A09B0375P</osCislo>
<email>nova.jana@students.zcu.cz</email>
<terminy>
<terminid>55884</terminid>
</terminy>
</student>
```

Současná verze využívá k ukládání dat .txt soubory:

- **mail.txt** - zadaný email, na který bude chodit upozornění
- **osc.txt –** načtené osobní číslo
- **sem.txt** – vybraný semestr
- **terminy.txt** - id jednotlivých termínů, oddělených čárkou (je možné sledovat více termínů)

8.2.2 Použité webové služby

Všechny následující webové služby lze najít na této adrese:

https://stag-ws.zcu.cz/ws/help/list

Byly využity webové služby podle REST především kvůli jednoduchosti jejich volání. Stačí znát adresu služby a pak již jen zadat do URL potřebné parametry. Výsledkem volání služby je XML soubor s požadovanými daty. Pokud bychom chtěli vrátit výsledky v jiném formátu, přidáme jako parametr outputFormat=format, kde „format“ může být například xls, csv, json a u určitých služeb také ics. Pro naše účely je však XML nejlepší volba, především kvůli dalšímu jednoduchému zpracování v PHP.

Celá adresa služby, tak, jak je volána v PHP aplikaci ($stagUserName je PHP proměnná), vypadá následovně:

https://stag-ws.zcu.cz/ws/services/rest/orion/getOsobniCislaByOrionLogin?login=$stagUserName

Dále budeme uvádět pouze část, která se nachází za rest/.

- **orion/getOsobniCislaByOrionLogin?login=$stagUserName**

  Získá podle zadaného loginu osobní čísla a vrátí je jako XML.

- **predmety/getPredmetyByStudent?osCislo=$osCislo&semestr=$semestr**

  Získá seznam předmětů studenta podle osobního čísla a semestru, oba parametry jsou povinné. Tato služba se používá ve výpisu termínů dle předmětů.

- **terminy/getTerminyZkousek?zkratka=$predmet&katedra=$katedra&zobrazitProsle=false**

Vrátí seznam termínů podle zadaného předmětu a katedry, prošlé termíny nebudou zobrazeny. V našem případě musí být parametry zkratka a katedra uvedeny. Tato služba je použita ve výpisu termínu.

- **terminy/getTerminyZkousek?termIdno=$idTerminu**

  Vrátí jeden konkrétní termín. Parametr je termIdno – jednoznačný identifikátor termínu. Tato služba je použita při kontrole konkrétního sledovaného termínu.

8.2.3 Postup realizace

Vzhledem k poměrně malému rozsahu, byla aplikace napsána procedurálně. Skriptovací část aplikace tvoří tyto soubory:

- **funkce.php**

  Soustřeďuje v sobě všechny funkce použité v aplikaci. Tento soubor je dále vkládán příkazem „include“ do dalších částí aplikace. Jsou v něm také definované konstanty se jmény souborů a stavy aplikace.

- **akce.php**

  Zajišťuje obsluhu akcí uživatele. Po zpracování požadavku znovu přesměruje na požadovanou stránku. Je zde realizována obsluha přihlášení, odhlášení, sledování termínů, zrušení sledování, zadání emailu a další akce. Na tento soubor je metodou GET směrován vždy příslušný formulář s nastaveným parametrem „action“. Dle tohoto parametru se poté provede příslušný kód v určité větvi příkazu switch.

- **zadat-email.php**

  Obsahuje formulář pro přidání a editaci emailu. Pokud je email již zadán, je přečten ze souboru mail.txt a vložen do inputu jako výchozí hodnota.

- **index.php**

  Jedná se o úvodní obrazovku jejíž obsah se řídí podle toho, zda je už uživatel přihlášen či nikoliv. Pokud přihlášen není, obsahuje úvodní obrazovka odkaz na přihlášení. Pokud je přihlášen, uvidí výpis konkrétních termínů roztříděných dle předmětů.

- **planovac.php**

  Planovac.php je skript, který je automaticky volán nějakým nastaveným plánovačem (CRON) na webovém serveru. Záleží už jen na uživateli, jakou periodu volání (kontroly) si nastaví. V tomto souboru je jednoduchý skript, který získá id sledovaných termínů a každý termín pomocí webové služby zkontroluje, zda se neuvolnilo místo. Pokud je na termínu obsazení menší než kapacita, zašle uživateli emailové upozornění.

Po stisknutí tlačítka přihlásit je přihlašovacímu serveru metodou GET zasláno URL, na které se má po úspěšném přihlášení zpět přesměrovat. Pokud je přihlášení úspěšné, je uživatel přesměrován zpět a metodou GET je také vrácen uživatelův přihlašovací tiket spolu s jeho loginem. Ten je při úspěšném přihlášení uložen do session. Právě přítomnost loginu v session rozhoduje, zda zobrazí přihlašovací obrazovku nebo výpis termínů.

Pokud je uživatel přihlášen, volá se funkce `vypisZahlavi($stagUserName)`, která pomocí webové služby zjistí ze zadaného uživatelského jména jeho osobní čísla a uloží je do HTML prvku select, aby bylo možné mezi osobními čísly přepínat. V záhlaví lze přepínat i mezi semestry. Údaj o aktuálním zobrazeném semestru se zapíše do pomocného .txt souboru. Volána je také funkce `vypisPredmetyByStudent($osCislo, $semestr)`, která dle zadaného osobního čísla a semestru vypíše termíny rozdělené podle předmětů uživatele. Funkce využívá také dvě webové služby `predmety/getPredmetyByStudent` a `terminy/getTerminyZkousek`. Data z webových služeb jsou vrácena jako XML a zpracována následujícím příkazem: `$xmlUrl = simplexml_load_file(rawurlencode($urlRest))`. Příkazem foreach je pak procházen celý objekt, kde jsou vždy získány údaje o kapacitě a obsazení každého termínu. Část zdrojového kódu funkce `vypisPredmetyByStudent($osCislo, $semestr)` lze nalézt v příloze D.

Pokud je kapacita rovna obsazení, zobrazí se tlačítko pro sledování termínu. Pokud je kapacita větší než aktuální obsazení, zobrazí tlačítko (odkaz) na portál, kde se může student zapsat. Může nastat situace, kdy bude obsazení větší než aktuální kapacita, v tom případě bude také zobrazen odkaz na portál (počítá se s tím, že se jedná o termín, kde není kapacita určena).

Chce-li uživatel sledovat obsazený termín, stiskne tlačítko „sledovat“, tím je metodou GET předán parametr, podle kterého se v souboru akce.php vybere správná větev příkazu switch a zapíše se id termínu do pomocného .txt souboru. Je možné sledovat více termínů, v .txt souboru se oddělují čárkou. Jestliže už nechce uživatel termín sledovat, stiskne tlačítko „zrušit sledování“ a pomocí funkce `array_search($idTerminu, $terminy)` najde v poli zadané id na určité pozici a funkcí `unset($terminy[$key])`id z pole odstraní. Nové pole termínu se poté díky funkci `implode()` zapíše do .txt souboru a přepíše pole staré.

Sledování je pak realizováno ve skriptu planovac.php, kde se funkcí `checkTermin($idTerm)` volá webová služba, která podle id termínu zjistí aktuální obsazení a pokud je volné místo, zavolá funkci `posliMail()`, která se již postará o vlastní odeslání upozornění.

Pomocí session „zprava_systemu“ je uživatel informován o výsledku provedených akcí.

8.2.4 Technické požadavky a omezení aplikace

Verze PHP

Ke spuštění aplikace je třeba webový server s podporou PHP5. V aplikaci je využívána funkce simplexml_load_file, která je dostupná v PHP5, proto by aplikace na starších verzích PHP nefungovala.

Konfigurace serveru

Aby bylo možné načítat data z URL, je třeba v konfiguračním souboru webového serveru php.ini nastavit direktivu `allow_url_fopen = On`. Nastavení je možné provést i v souboru .htaccess takto: `php_value allow_url_fopen On` avšak u některých poskytovatelů hostingu nemusí být tato změna povolena.

Vlastní složka

Aplikace předpokládá, že každý uživatel bude mít svoji klientskou aplikaci ve své složce. Je tedy možné mít na jednom hostingu nebo lokálním serveru více aplikací (pro více uživatelů), ale každý uživatel musí mít soubory aplikace ve vlastní složce. Kdyby se více uživatelů přihlašovalo k aplikaci umístěné ve stejné složce, přepisovali by si navzájem emailové adresy a id termínů (uložené v .txt souboru). Tento problém by bylo

možné odstranit například ukládáním dat do databáze, ukládáním do unikátních .txt souborů nebo jednoho XML souboru.

Nastavení plánovače

Na webovém serveru je třeba nastavit automatické spouštění skriptů, konkrétně soubor planovac.php. Tím bude zajištěna kontrola obsazenosti termínů. Nastavení plánovače a jeho periody už je zcela individuální. Bez automatické kontroly by ale aplikace neměla smysl a sloužila by pouze jako výpis termínů.

Změna nutnosti přihlašování ke službě

Webovou službu `rest/terminy/getTerminyZkousek`, je nyní možné volat i bez ověření uživatele. Právě tuto službu využívá planovac.php k automatické kontrole obsazenosti na termínu. Pokud by se nastavení této služby změnilo a bylo by nutné ověřování, muselo by se v aplikaci toto automatické ověřování při volání služby přidat.

Nejjednodušší způsob, jak získat přístup ke službě bez nutnosti zadávat přihlašovací údaje do formuláře, je uvést údaje přímo v požadavku, například takto: `https://novajana:heslo@stag-ws.zcu.cz/ws/services/rest/student/getStudentInfo?osCislo=A09B0375P`. Přihlašovací údaje by ale musely být uloženy v aplikaci a ideálně také zašifrovány.

9. Závěr

Tato práce popisuje fungování technologie webových služeb a ukazuje některá jejich použití v praxi. Vysvětleno je základní použití XML ve webových službách. Pozornost je věnována především XML schema a jmenným prostorům.

Podrobněji je popsána každá z hlavních částí webových služeb – SOAP, WSDL a UDDI. Krátce je také zmíněna architektura REST, kterou, jak se nakonec ukázalo, využívají často i velké společnosti jako Google, Twitter apod. Právě Google využívá REST například v API produktu Google Maps. Uvedeny jsou také ukázky využití webových služeb, které poskytují Google, Twitter, Amazon, Heureka.cz a IS/STAG.

Součástí práce je i ukázka implementace jednoduché klientské aplikace, která využívá webové služby IS/STAG. Tato aplikace, napsaná v jazyce PHP, dokáže kontrolovat obsazenost míst na zkouškovém nebo zápočtovém termínu. V případě uvolnění místa z plně obsazeného termínu zašle uživateli emailové upozornění. Tato aplikace by se dala v budoucnu rozšířit a vylepšit zejména:

- možností konfigurace při jakém počtu obsazení bude upozornění zasíláno,
- změnou způsobu ukládání dat (XML soubor),
- oddělení prezentace dat od aplikační logiky využitím Smarty šablon.

Práce obsahuje také shrnutí základního postupu, jak vytvořit novou webovou službu. Uvedena je jednoduchá ukázka vytvoření webové služby v jazyce PHP.

V rámci této práce bylo provedeno dotazníkové šetření, které ukázalo, že registr služeb UDDI se již téměř nepoužívá. Průzkum potvrdil, že technologii webových služeb využívají především větší firmy. Z šetření také vyplynulo, že SOAP je stále nejpoužívanější protokol při implementaci webových služeb v zakázkovém softwaru a XML nejpoužívanější výstupní formát.

Příloha A - Otázky a možnosti z dotazníku

1. Používáte ve svých vlastních softwarových produktech technologii webových služeb?
(povinná otázka, pouze jedna možnost)
Na stupnici 1 - 5 vyjádřete, jak často webové služby používáte. 1 = nikdy, 5 = vždy

- 1
- 2
- 3
- 4
- 5
- Nevím, o co se jedná

2. Požadují vaši klienti implementaci některé části systému jako webovou službu?
(povinná otázka, pouze jedna možnost)
Na stupnici 1 - 5 vyjádřete, jak často je implementace webových služeb vyžadována. 1 = nikdy, 5 = vždy

- 1
- 2
- 3
- 4
- 5
- Nevím, o co se jedná

3. Jaký druh klientů od Vás nejčastěji požaduje implementaci části softwaru jako webovou službu?(nepovinná otázka, více možností)

- fyzické osoby
- mikropodniky (<10 zaměstnanců)
- malé podniky (<50 zaměstnanců)
- střední podniky (<250 zaměstnanců)
- velké podniky (>250 zaměstnanců)
- Jiné:

4. Jaký přístup k datům používáte?
(povinná otázka, více možností)

- REST
- SOAP
- Nevím, o co se jedná
- Jiné:

5. Používáte při práci s webovými službami registr UDDI?

(nepovinná otázka, pouze jedna možnost)

Na stupnici 1 - 5 vyjádřete, jak často používáte UDDI. 1 = nikdy, 5 = vždy

- 1
- 2
- 3
- 4
- 5
- Nevím, o co se jedná

6. Jaké výstupní formáty dat při práci s webovými službami používáte nejčastěji?
(povinná otázka, vice možností)

- XML
- JSON
- XLS
- CSV
- ICS
- Jiné:

7. Příklad použití webových služeb v systému
(nepovinná otázka)

Příloha B - Uživatelská dokumentace

K čemu tato aplikace slouží?

Tato aplikace dokáže kontrolovat obsazenost míst na zkouškovém a zápočtovém termínu. Pokud se z plně obsazeného termínu někdo odepíše, aplikace zašle uživateli upozornění na email.

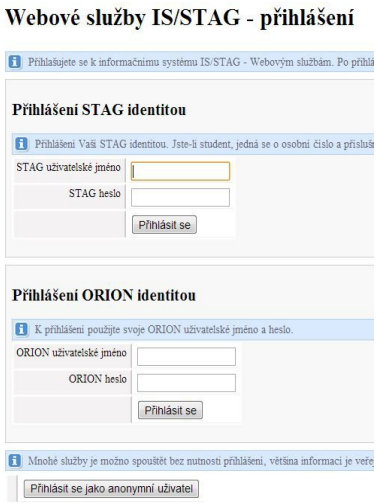
Jak nastavit sledování obsazenosti míst?

Jako první krok je třeba zkopírovat soubory aplikace do umístění na webový server.

Pokud se nacházíte na přihlašovací stránce (obr. b. 1), stiskněte tlačítko přihlásit - budete přesměrováni na přihlašovací obrazovku (obr. b. 2), kde zadáte své přihlašovací údaje - orion login a heslo. Pokud proběhlo ověření v pořádku, budete přesměrování zpět do aplikace a uvidíte výpis všech vašich aktuálních zkouškových termínů. (obr. b. 3).

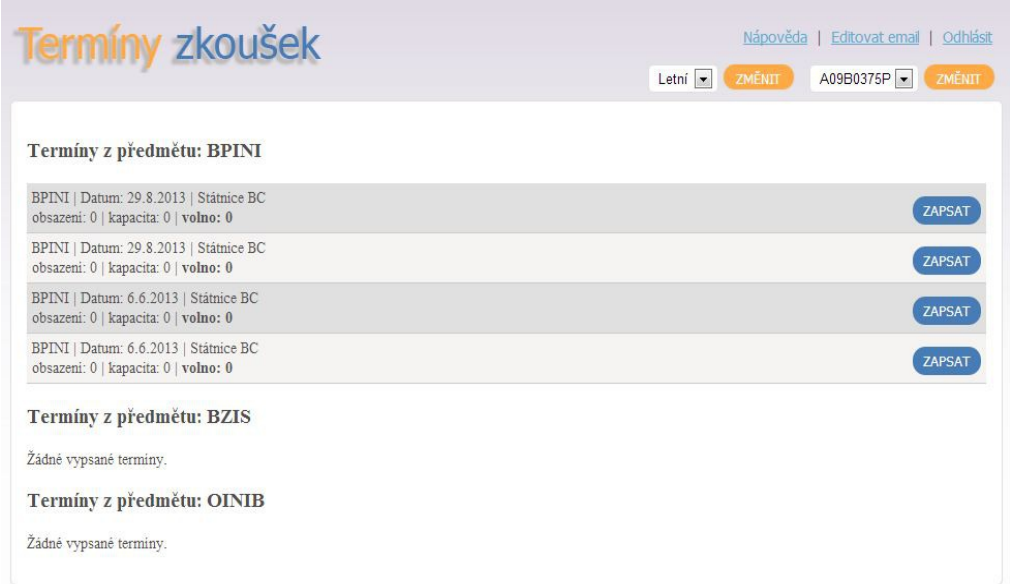
obr. b. 1 Klientská aplikace

Zdroj: vlastní



obr. b. 2 Klientská aplikace

Zdroj: vlastní



obr. b. 3 Klientská aplikace

Zdroj: vlastní

Nastavte Vaši emailovou adresu, na kterou budou chodit upozornění, pokud se ze sledované zkoušky někdo odepíše. Klikněte na „editovat email“. Zadejte Vaši emailovou adresu a stiskněte ok (obr. b.4).

obr. b. 4 Klientská aplikace

Zdroj: vlastní

Na obrazovce nyní vidíte výpis všech termínů a jejich aktuální obsazenost. Pokud se kapacita termínu rovná aktuální obsazenosti, u termínu je vidět tlačítko "sledovat" (obr. b. 5). Po stisknutí je termín zaznamenán a připraven ke sledování. Pokud byste toto tlačítko stiskli před zadáním emailové adresy, budete k jejímu zadání vyzváni. Až když je emailová adresa uložena, je možné nastavit sledování.

obr. b. 5 Klientská aplikace

Zdroj: vlastní

Pokud chcete sledování zrušit, stiskněte tlačítko „zrušit sledování“ (obr. b. 6), termín přestane být sledován.

obr. b. 6 Klientská aplikace

Zdroj: vlastní

Obecné poznámky

- Je možné sledovat více termínů z více osobních čísel jednoho uživatele (jeden orion login).
- Přepínat mezi jednotlivými semestry a osobními čísly je možné výběrem z rozbalovacího seznamu a stisknutím tlačítka „změnit“.
- Kliknutím na tlačítko „zapsat“ budete přesměrování na stránky portálu, kde se můžete na termín zapsat.

Příloha C - Přehled funkcí klientské aplikace

- `zapisOsobniCislo(string stagUserName)`

  Zjistí osobní čísla podle uživatelského jména a zapíše je do souboru.

- `vypisZahlavi(string UserName)`

  Vypíše záhlaví s vybranými údaji – osobní číslo a semestr + odkazy na odhlášení, nápovědu a editaci emailu. Vrátí html obsah.

- `vypisPredmetyByStudent(string osCislo, string semestr = 'ZS')`

  Pomocí webových služeb zjistí předměty studenta a podle nich vypíše termíny. Defaultní semestr je zimní. Vrátí html obsah.

- `getMail()`

  Vrátí email uložený v .txt souboru.

- `getOsobniCislo()`

  Vrátí osobní číslo aktuálně uložené v .txt souboru

- `getTerminyId()`

  Vrátí pole aktuálně sledovaných termínů.

- `jeSledovan(int idTerminu)`

  Test zda je termín již sledován. Vrátí boolean.

- `jeMailOK(string mail)`

  Jednoduchý test formátu zadaného emailu. Vrátí boolean

- `jeZadanMail()`

  Test zda je zadán email, na který bude chodit upozornění

- `checkTermin(int idTerminu)`

  Použije webovou službu ke kontrole určitého sledovaného termínu. Pokud je nějaké místo volné, pošle email s oznámením.

- `posliMail(string predmet, string datum, int obsazeni, int kapacita)`

  Funkce zajišťující vlastní odesílání emailového upozornění.

Příloha D – Část zdrojového kódu funkce vypisPredmetyByStudent()

```
function vypisPredmetyByStudent($osCislo, $semestr = 'ZS'){
$obsah = '';
$css  = '';
$xmlPredmety = "https://stag-ws.zcu.cz/ws/services/rest/predmety/getPredmetyByStudent?osCislo=$osCislo&semestr=
$semestr";
$xmlLoad = simplexml_load_file(rawurlencode($xmlPredmety));

    $cislo=0;
    //mam predmety studenta
    foreach($xmlLoad->predmetyStudenta->predmetStudenta as $pred){
    $cislo++;
    $predmet = $pred->zkratka;
    $katedra = $pred->katedra;

        // zobrazim terminy zkousek z predmetu
        $urlRest = "https://stag-ws.zcu.cz/ws/services/rest/terminy/getTerminyZkousek?zkratka=$predmet&katedra=
        $katedra&zobrazitProsle=false";
        $xmlUrl = simplexml_load_file(rawurlencode($urlRest));

        $obsah .= '<h3>Termíny z předmětu: '.$predmet.'</h3>';

        //vypisu vsechny terminy studenta a jejich obsazeni
        if(!$xmlUrl->terminy->termin){
        $obsah .= '<p>Žádné vypsané termíny.</p>';
        }

        $i=0;
        foreach($xmlUrl->terminy->termin as $ter){

            $kapacita = (int)$ter->limit;
            $obsazeni = (int)$ter->obsazeni;
            $rozdil = $kapacita - $obsazeni;
```

Seznam zkratek

- AJAX - Asynchronous JavaScript and XML
- B2B - Business-to-business
- B2E - Business-to-employee
- B2G - Business-to-government
- CORBA - Common Object Request Broker Architecture
- CSS - Cascading Style Sheets, styly pro formátování html, xhtml a xml stránek
- CSV - Comma-Separated Values, formát souboru
- DTD - Document Type Definition, definice typu dokumentu
- EDI - Electronic Data Interchange, elektronická výměn dat, většinou u obchodních dokumentů
- ERP - Enterprise resource planning, informační systém
- HTTP - Hypertext Transfer Protocol, internetový protokol
- HTTPS - Hypertext Transfer Protocol Secure, zabezpečný HTTP
- ICS – iCal, Formát pro uložení kalendářových dat
- IDE - Integrated development environment, vývojové prostředí
- JSON - JavaScript Object Notation, datový formát
- MySQL – open sourcová relační databáze
- NASSL – jeden z jazyků, ze kterého vzniklo WSDL
- PHP - Hypertext Preprocessor, skriptovací jazyk
- REST - Representational State Transfer, softwarová architektura pro distribuované prostředí
- SCL - jeden z jazyků, ze kterého vzniklo WSDL
- SDL - jeden z jazyků, ze kterého vzniklo WSDL
- SMTP - Simple Mail Transfer Protocol, internetový protokol pro emaily
- SOAP - Simple Object Access Protocol, protokol pro výměnu XML zpráv pomocí HTTP
- TCP - Transmission Control Protocol, ínternetový protokol na transportní vrstvě
- UDDI - Universal Description Discovery and Integration, registr webových služeb
- W3C - World Wide Web Consortium, konsorcium zabývající se webovými standardy

- WCF - Windows Communication Foundation
- WSDL - Web Services Description Language, popis webových služeb
- XHTML - Extensible HyperText Markup Language, značkovací jazyk pro hypertextové dokumenty
- XLS – přípona souboru Microsoft Excel
- XSL - eXtensible Stylesheet Language
- XML -eXtensible Markup Language, značkovací jazyk
- XML-RPC - eXtensible Markup Language - Remote Procedure Call protocol, protokol pro vzdálené volání procedur

Seznam použitých zdrojů

[1] KOSEK, Jiří. PHP a XML. 1. vyd. Praha: Grada, 2009, 367 s. ISBN 978-80-247-1116-4.

[2] Web Services Architecture. W3C. [online]. [cit. 2013-01-07]. Dostupné z: http://www.w3.org/TR/ws-arch/#whatis

[3] Jak fungují webové služby. In: [online]. [cit. 2012-12-28]. Dostupné z: http://interval.cz/clanky/jak-funguji-webove-sluzby/

[4] M. Kuba. Web Services. Zpravodaj ÚVT MU. ISSN 1212-0901, 2003, roč. XIII, č. 3, s. 9-14.

[5] KOSEK, Jiří. Využití webových služeb a protokolu SOAP při komunikaci. In: [online]. [cit. 2013-01-08]. Dostupné z: http://www.kosek.cz/diplomka/html/websluzby.html

[6] Latest SOAP versions. [online]. [cit. 2013-01-08]. Dostupné z: http://www.w3.org/TR/soap/

[7] ERL, Thomas. *SOA: servisně orientovaná architektura : kompletní průvodce*. Vyd. 1. Překlad Ondřej Baše, Lukáš Krejčí. Brno: Computer Press, 2009, 671 s. ISBN 978-80-251-1886-3.

[8] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). [online]. [cit. 2013-01-9]. Dostupné z: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

[9] 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.

[10] Schema defined in the SOAP Version 1.2 Part 1 specification. [online]. [cit. 2013-01-11]. Dostupné z: http://www.w3.org/2003/05/soap-envelope/

[11] *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer* [online]. [cit. 2013-01-12]. Dostupné z: http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/

[12] ROSANOVA, Dan. Why UDDI Is Important. In: [online]. [cit. 2013-01-21]. Dostupné z: http://advice.cio.com/dan_rosanova/why_uddi_is_important

[13] UDDI v3: The Registry Standard for SOA. [online]. [cit. 2013-01-21]. Dostupné z: https://www.oasis-open.org/presentations/uddi_v3_webcast_20050222.pdf

[14] RODRIGUEZ, Alex. RESTful Web services: The basics. In: [online]. 2008 [cit. 2013-01-22]. Dostupné z: https://www.ibm.com/developerworks/webservices/library/ws-restful/

[15] Webové služby nad IS/STAG. In: [online]. [cit. 2013-01-22]. Dostupné z: https://stag-ws.zcu.cz/ws/help?page=tech

[16] KOSEK, Jiří. *XML pro každého: podrobný průvodce*. 1. vyd. Praha: Grada, 2000, 163 s. Průvodce (Grada). ISBN 80-716-9860-1.

[17] YOUNG, Michael J. XML: krok za krokem. Vyd. 2. Brno: Computer Press, 2006, 471 s. ISBN 80-251-1070-2.

[18] Webové služby nad IS/STAG. [online]. [cit. 2013-01-28]. Dostupné z: https://stag-ws.zcu.cz/ws/help/

[19] Webové rozhraní pro vybrané služby nad IS/STAG. In: [online]. [cit. 2013-01-28]. Dostupné z:

https://stag-ws.zcu.cz/webaccess/main?page=help

[20] KLÁN, Petr. Získávání informace s použitím webových služeb. In: *Automa* [online]. [cit. 2013-01-28]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=30711

[21] A well earned retirement for the SOAP Search API. GOOGLE. [online]. [cit. 2013-01-29]. Dostupné z: http://googlecode.blogspot.cz/2009/08/well-earned-retirement-for-soap-search.html

[22] Google Maps API Web Services. GOOGLE. [online]. [cit. 2013-01-29]. Dostupné z: https://developers.google.com/maps/documentation/webservices/

[23] GOOGLE INC. *The Google Geocoding API* [online]. [cit. 2013-04-12]. Dostupné z: https://developers.google.com/maps/documentation/geocoding/

[24] GOOGLE INC. *Google Maps API* [online]. [cit. 2013-04-13]. Dostupné z: https://developers.google.com/maps/location-based-apps

[25] Heureka Košík - API. [online]. [cit. 2013-04-15]. Dostupné z: http://sluzby.heureka.cz/napoveda/kosik-api/

[26] REST API v1.1 Resources. In: [online]. [cit. 2013-01-29]. Dostupné z: https://dev.twitter.com/docs/api/1.1

[27] Amazon Simple Storage Service (Amazon S3). [online]. [cit. 2013-01-29]. Dostupné z: http://aws.amazon.com/s3/

[28] Creating and Consuming Web Services With PHP. In: [online]. [cit. 2013-04-5]. Dostupné z: http://www.xml.com/pub/a/ws/2004/03/24/phpws.html?page=2