|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **XML parser** | | |
| Roman Malček | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2016/2017 | |

***Poděkování***

##### *Chtěl bych poděkovat panu učiteli Ing. Petru Grussmannovi za návrh, konzultace a pomoc s projektem. Dále svým spolužákům Ondřejovi „Toasteru“ Kinštovi, Matějovi Hlouškovi a Jáchymu Bulkovi za zajímavé nápady.*

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2016

*podpis autora práce*

**ANOTACE**

Cílem projektu bylo vytvořit webovou aplikaci pro parsování a následný import dat z XML souboru do databáze, která by fungovala např. pro xml-feedy v Prestashopu. Aplikace byla vyvinuta s použitím jazyka Python ve Flask frameworku a využívá i řady dalších webových technologii. K parsování dat jsem použil knihovnu ElementTree a následnému importu do databáze PyMySQL.

**Klíčová slova**

Webová aplikace, ElementTree, XML soubor, PyMySQL, databáze, Python, Flask

OBSAH

[Úvod 6](#_Toc472354556)

[1 Využité technologie 7](#_Toc472354557)

[1.1 Jádro aplikace 7](#_Toc472354558)

[1.1.1 Python 7](#_Toc472354559)

[1.1.2 Flask 7](#_Toc472354560)

[1.1.3 MySQL 7](#_Toc472354561)

[1.1.1 SQLAlchemy 7](#_Toc472354562)

[1.1.2 Jinja 2 8](#_Toc472354563)

[1.1.3 Bootstrap 8](#_Toc472354564)

[1.1.4 Pycharm 8](#_Toc472354565)

[1.2 XML parser 8](#_Toc472354566)

[1.2.1 PyMySQL 8](#_Toc472354567)

[1.2.2 ElementTree 9](#_Toc472354568)

[2 řešení a použité postupy 10](#_Toc472354569)

[2.1 Prostředí aplikace 10](#_Toc472354570)

[2.1.1 Základ aplikace 10](#_Toc472354571)

[2.1.2 XML parser 10](#_Toc472354572)

[2.2 Adresářová struktura 10](#_Toc472354573)

[2.3 Dynamický objekt 11](#_Toc472354574)

[2.4 Parsování dat 11](#_Toc472354575)

[2.5 Import do databáze 13](#_Toc472354576)

[2.6 Řešené problémy 14](#_Toc472354577)

[2.6.1 Nahrávání souborů 14](#_Toc472354578)

[2.6.2 SQL injection 14](#_Toc472354579)

[3 Výsledky a výstupy 15](#_Toc472354580)

[Závěr 16](#_Toc472354581)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 17](#_Toc472354582)

# Úvod

Projekt jsem si vybral na základě zkušeností z odborné praxe, kde jsem se často setkával s problémem importování většího množství dat do databáze Prestashopu. Na praxi jsem se setkal s řešením Prestashop modulu v PHP, což nebylo ideální, protože se import musel krokovat, aby nedošlo k odpojení od databáze. Z toho důvodu jsem na základě rady učitele Ing. Petra Grussmanna zvolil framework Flask.

Framework je použitelný v mnoha situacích a lze jej rozšířit o spoustu dalších funkcí. Proto bylo nutné vytvořit v frameworku Flask všechny základní funkce, které by měla taková webová aplikace obsahovat. Pro správnou funkci aplikace je nutné, aby jednotlivé technologie správně spolupracovaly a aby jednotlivé části aplikace byly dále rozšířitelné podle uvážení.

Podrobněji se budu v této dokumentaci věnovat technologiím využitých ke tvorbě jádra  
a poté těch využitých v části samotného XML parseru. Následně vás uvedu do problematiky  
a přiblížím, jak jsem postupoval.

# Využité technologie

K programování celé aplikace jsem použil jazyk Python. A jako základ aplikace framework Flask, který je velice přehledný a využívá spoustu technologií, které jednoduchým způsobem řeší potřebné věci, jako jsou databáze, formuláře či následné převedení informací do formy zobrazitelné v internetovém prohlížeči.

## Jádro aplikace

### Python

Python je široce používaný, interpretovaný, dynamický, skriptovací a všestranný programovací jazyk. Kód programu v pythonu je ve srovnání s jinými jazyky krátký a dobře čitelný. K význačným vlastnostem patří jeho jednoduchost z hlediska učení. Je vhodný pro začátečníky díky své jednoduchosti a čistotě syntaxe.

### Flask

Jedná se o „micro web framework“ napsaný v Pythonu. Díky rozsáhlé dokumentaci a tomu že je v Pythonu, bylo jednoduché se v frameworku zorientovat. Je založený na Werkzeug toolkitu a Jinja2 templatovacím enginu.

Pro začátek jsem použil projekt Flask-Skeleton, což je již rozepsaný základ webové aplikace ve Flasku volně dostupný na internetu. Jedná se o jakýsi základový kámen celého projektu, který spojuje všechny použité technologie.

### MySQL

MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.

### SQLAlchemy

Pythonový SQL toolkit a objektově relační mapper. SQLAlchemy se chová k SQL databázím jako k objektům, což zlepšuje výkon databáze a ulehčuje nám s ní práci.

### Jinja 2

Moderní šablonovací systém pro Python. Mezi hlavní výhody patří přehlednost, rozšířitelnost pomocí filtrů a pluginů, snadnost použití a vysoká rychlost.

### Bootstrap

Bootstrap je volně stažitelná sada nástrojů pro tvorbu webu a webových aplikací.

Obsahuje návrhářské šablony založené na HTML a CSS, sloužící pro úpravu typografie, formulářů, tlačítek, navigace a dalších komponent rozhraní, stejně jako další volitelná rozšíření JavaScriptu.

Bootstrap je kompatibilní s poslední verzí všech hlavních prohlížečů a elegantně se přizpůsobuje použití na starších prohlížečích jako je Internet Explorer 8. Od verze 2.0 také podporuje responzivní design. To znamená, že se rozložení stránky dynamicky přizpůsobuje  
s ohledem na používané zařízení (stolní PC, tablet, mobilní telefon). Bootstrap má otevřenou licenci a je dostupný na GitHub.

### Pycharm

Integrované vývojové prostředí (IDE) specifické pro Python. Obsahuje grafický debugger  
a je cross-platformní. Velice ulehčuje vytváření aplikace a je (s jistými omezeními) zadarmo pro studenty.

## XML parser

### PyMySQL

Knihovna pro komunikaci s MySQL databázemi v Pythonu, jejíž účelem bylo nahradit knihovnu MySQLdb, která nemá podporu pro novější verze Pythonu.

### ElementTree

Knihovna pro parsování dat ve struktuře XML souboru.

XML je ve své podstatě hierarchický datový formát a nejpřirozenější způsob, jak ho reprezentovat je stromem. ElementTree má dvě třídy pro tento účel - ElementTree reprezentuje celý XML dokument jako strom, a element představuje jeden uzel v tomto stromu. Interakce s celým dokumentem se obvykle provádí na úrovni ElementTree, interakce s jediným elementem a jeho dílčími prvky na úrovni elementu.

Každý element má řadu vlastností, které jsou s ním spojeny:

* Značka, která je string identifikující jaký druh dat tento prvek představuje.
* Atributy, uložené v Python slovníku.
* Textový řetězec.
* Volitelný tail string.
* Řada podřízených prvků, uložených v Python sekvenci.

# řešení a použité postupy

## Prostředí aplikace

### Základ aplikace

Jako základ aplikace jsem použil Flask-Skeleton, jedná se o předem připravenou „kostru“ aplikace. Abych jej mohl využít a rozšířit o potřebné funkce, bylo nutné nainstalovat veškeré požadavky projektu a seznámit se s jeho strukturou.

Dále bylo nutné vytvořit části, které by byly použitelné při tvoření téměř jakékoliv webové aplikace (jako např. registrování uživatelů). Všechny tyto části se dají vytvořit jednoduše ve Flask frameworku a některé jsou již předem připravené v projektu Flask-Skeleton.

Aplikace je realizovaná formou moderních webových stránek s responzivním designem.

### XML parser

Celá část XML parseru je tvořená z HTML formulářů, pomocí kterých se nejprve nahraje XML soubor na server do dočastných souborů a následně nastaví import. O výsledku jsme nakonec informováni prostřednictvím flash message.

## Adresářová struktura

project

│

├───client - grafická část aplikace (šablony, styly,…)

│ │

│ ├───static - statické části jako jQuery, css,…

│ │

│ └───templates - šablony částí, které jsou na všech stránkách

│ │

│ ├───errors - šablony, zobrazující se v případě chyby

│ │

│ ├───main - šablony pro všechny uživatele

│ │

│ └───user - šablonypouze pro přihlášené uživatele

│

└───server - logická část aplikace (program v pythonu)

│

├───main - funkce dostupné všem uživatelům

│

└───user - funkce dostupné pouze přihlášeným uživatelům

## Dynamický objekt

Aby bylo možné ukládat data z libovolného XML souboru, potřeboval jsem vytvořit dynamický objekt, který by se měnil v závislosti na jeho obsahu. Proto jsem použil pole slovníků, které nejsou nijak omezené a přistupuje se k nim pomocí klíčů, které jsou uloženy v samostatném poli.

**class** Data():

**# vytvoření pole slovníků #** pole = []  
  
 **def** \_\_init\_\_(self, pocetRadku):  
 self.pole = [{} **for** x **in** range(pocetRadku)]

**# metoda pro ukádání dat #** **def** setData(self, radek, key, value):  
 self.pole[radek][key] = value

**# metoda pro čtení dat #** **def** getData(self, radek, key):  
 **return** self.pole[radek][key]  
  
 **def** \_\_repr\_\_(self):  
 **return '<Data>'**

## Parsování dat

Samotné parsování je zajištěno knihovnou ElementTree. Problém se ale schovával v kombinaci tagů a atributů ve struktuře XML souborů. Musel jsem zajistit, aby aplikace snímala na každém řádku tagy i atributy a následně je ukládala do dynamického objektu.

**# uložení kopie souboru a její otevření #**

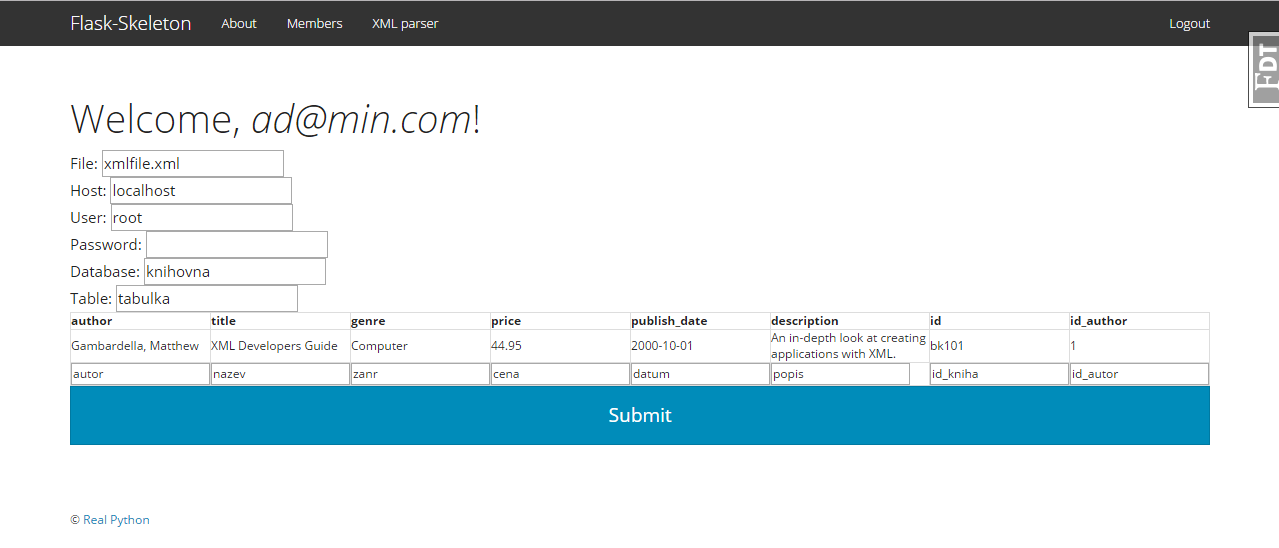
filename = file.filename  
target = tempfile.gettempdir()  
destination = **"/"**.join([target, filename])  
file.save(destination)  
file = open(destination,**"r"**)

**# parsování dat pomocí knihovny ElementTree #**  
tree = ET.parse(file)  
root = tree.getroot()

**# uložení dat do dynamického objektu a nastavení důležitých proměnných #**  
tagy = []  
radky = 0  
**for** child **in** root:  
 sloupce = 0  
 **for** tag **in** child:  
 **if** root[radky][sloupce].tag **not in** tagy:  
 tagy.append(root[radky][sloupce].tag)  
 sloupce += 1  
 radky += 1  
  
data = Data(radky)  
  
radkyX = 0  
attributy1 = 0  
attributy2 = 0  
**for** child **in** root:  
 sloupceY = 0  
 **if** child.attrib != **"{}"**:  
 **for** key **in** child.attrib:  
 **if not** key **in** tagy:  
 tagy.append(key)  
 attributy1 += 1  
 data.setData(radkyX, key, child.attrib[key])  
 **for** tag **in** child:  
 data.setData(radkyX, root[radkyX][sloupceY].tag, root[radkyX][sloupceY].text)  
 **if** tag.attrib != **"{}"**:  
 **for** key **in** tag.attrib:  
 **if not** key **in** tagy:  
 tagy.append(key)  
 attributy2 += 1  
 data.setData(radkyX, key, tag.attrib[key])  
 sloupceY += 1  
 radkyX += 1  
 sloupce = sloupceY + attributy1 + attributy2

## Import do databáze

Pro nastavení importu jsem použil HTML formulář. Z uvedených dat se sestaví SQL dotaz   
a pomocí knihovny PyMySQL se aplikace připojí k databázi a následně dotaz provede.



Obrázek č. 1 - Nastavení importu

**# sestavení SQL dotazu #**

sql = **'INSERT INTO `'** + request.form[**'table'**] + **'` ('  
for** tag **in** tagy:  
 **if** request.form[tag] != **""**:  
 **if** sql == **'INSERT INTO `'** + request.form[**'table'**] + **'` ('**:  
 sql += request.form[tag]  
 **else**:  
 sql += **','** + request.form[tag]  
sql += **') VALUES '  
for** x **in** range(0,radky):  
 prvni = **True  
 if** x == 0:  
 sql += **"("  
 else**:  
 sql += **",("  
 for** tag **in** tagy:  
 **if** request.form[tag] != **""**:  
 **if** prvni == **True**:  
 sql += **"'"** + data.getData(x,tag) + **"'"** prvni = **False  
 else**:  
 sql += **",'"** + data.getData(x,tag) + **"'"** sql += **")"**sql += **";"  
  
# připojení k databázi a následný import dat #**

conn = pymysql.connect(host=request.form[**'host'**], user=request.form[**'user'**], password=request.form[**'password'**],  
db=request.form[**'database'**], cursorclass=pymysql.cursors.DictCursor)

conn.cursor().execute(sql)

## Řešené problémy

### Nahrávání souborů

Původně jsem chtěl soubory nahrávat do mezipaměti serveru, to byl problém, protože se mezipaměť při přechodu na další krok vymazala. Z tohoto důvodu jsem začal experimentovat s Ajaxem, který je schopný odesílat data z formuláře bez obnovení stránky. Toto řešení se mi nakonec nepodařilo realizovat.

V konečné řešení se soubor ukládá do dočasných souborů na serveru. Aplikace pracuje s touto kopií souboru a po dokončení importu se soubor ze serveru odstraní.

### SQL injection

Při testování aplikace jsem se setkal s problémem, kdy se SQL dotaz neprovedl úspěšně, protože se při jeho tvorbě narušil string (řetězec znaků) uvozovkami v datech XML souboru. To mě vedlo k problému s SQL injection, který ale není tak závažný, protože se uživatel připojuje k vlastní databázi.

Aby se nenarušil string uvozovkami, použil jsem funkci pro převedení některých znaků na jejich escape sekvence.

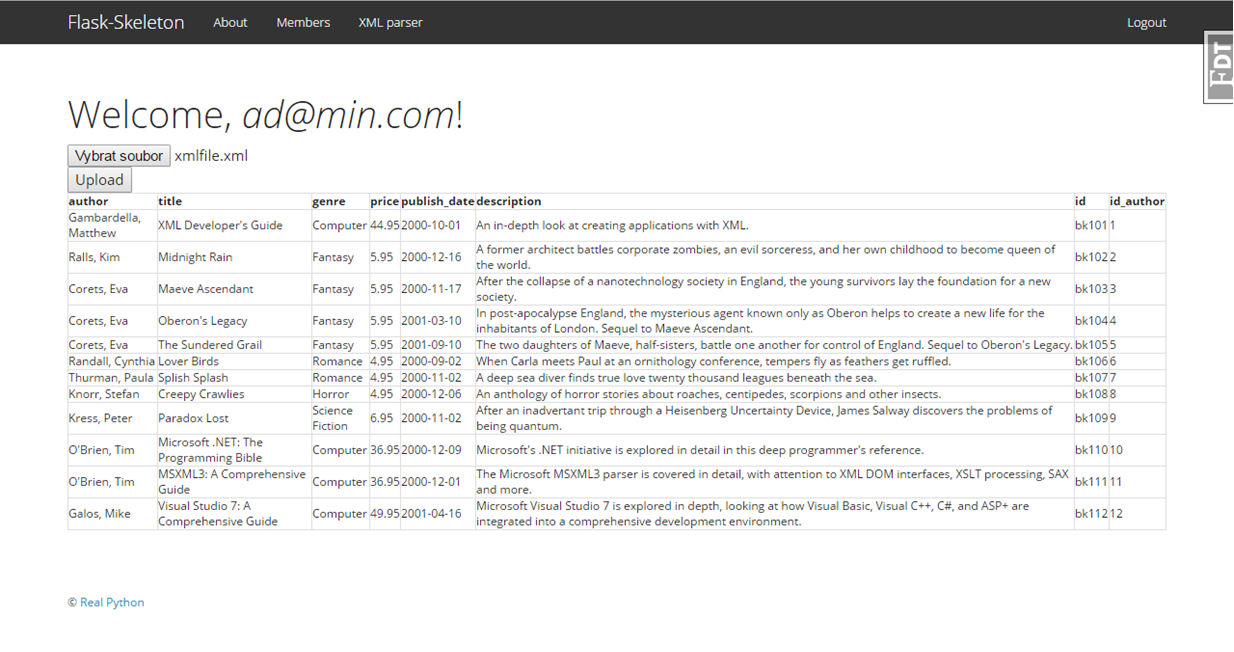
# Výsledky a výstupy

**Splněné cíle:**

* Zajištění funkčnosti šablony.
* Parsování dat.
* Uložení dat do dynamického objektu.
* Vypsání dat do tabulky pro následné nastavení importu.
* Import dat do databáze.

**Nesplněné cíle:**

* Offline processign.
* Import do více tabulek najednou.



Obrázek č. 2 - výpis dat v průběhu programování

# **Závěr**

Cílem projektu XML parser bylo vytvořit aplikaci, která by dokázala parsovat libovolné XML soubory a následně jejich obsah importovat do databáze. Projekt byl realizován jako internetová aplikace v jazyce Python s využitím Flask Frameworku s podporou dalších technologií jako JQuery, Bootstrap atd. Základem aplikace je část XML parseru, přístupná pouze pro oprávněné uživatele. Aplikace je řešena formou moderních webových stránek s responzivním designem.

Do budoucna plánuju změnu nahrávání souboru na JQuery uploader, který je atraktivnější  
a ukazuje aktuální stav uploadu, implementování offline processingu a vytvoření více uživatelsky přívětivého prostředí.

Projekt je volně přístupný na internetu na adrese https://github.com/kangusrm/XML-parser [cit. 2016-12-29].

Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] *Github Flask-skeleton* [online], poslední aktualizace: 13. 12. 2016,  
<<https://github.com/realpython/flask-skeleton>>

[2] *Flask dokumentace* [online], ©2016, [cit. 2016-12-29],  
<http://flask.pocoo.org/docs/0.12/>

[3] *Flask SQLAlchemy dokumentace* [online], ©2014, [cit. 2016-12-29],<http://flask-sqlalchemy.pocoo.org/2.1/>

[4] *Python dokumentace* [online], ©2016, [cit. 2016-12-29],  
<<https://docs.python.org>>

[5] *ElementTree dokumentace* [online], ©2016, [cit. 2016-12-29],<https://docs.python.org/2/library/xml.etree.elementtree.html>

[6] *W3Schools Bootstrap Tutorial* [online], ©2016, [cit. 2016-12-29],<http://www.w3schools.com/bootstrap/>

[7] *Stack Overflow* [online], ©2016,<http://stackoverflow.com>