VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

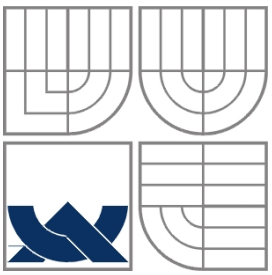
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# AUTOMATIZOVANÁ PRÁCE S WEBOVÝMI FORMULÁŘI

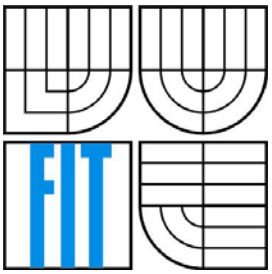
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PETR ZDRÁHAL
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# AUTOMATIZOVANÁ PRÁCE S WEBOVÝMI FORMULÁŘI

AUTOMATIC WEB FORM PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR ZDRÁHAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2007

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav informačních systémů Akademický rok 2006/2007

# Zadání diplomové práce

Řešitel: **Zdráhal Petr, Bc.**
Obor: Informační systémy
Téma: **Automatizovaná práce s webovými formuláři**
Kategorie: Web

Pokyny:

1. Seznamte se s jazyky HTML/XHTML, zejména s prostředky pro tvorbu formulářů
2. Prostudujte protokol HTTP a možné způsoby přenosu formulářových dat
3. Navrhněte způsob popisu formulářových polí a jejich možných hodnot pomocí XML
4. Vytvořte nástroj pro automatické získání tohoto popisu z HTML dokumentu
5. Implementujte nástroj pro vyplnění a odeslání zadaných hodnot do daného formuláře z příkazové řádky

Literatura:

- Mikle, P.: XDHTML - Referenční příručka, Zoner Press, Brno 2004
- Fielding, R.: Hypertext Transfer Protocol - HTTP/1.1, RFC 2616, The Internet Society, 1999

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Burget Radek, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 28. února 2006
Datum odevzdání: 22. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2
L.S.



doc. Ing. Jaroslav Zendulka, CSc.
*vedoucí ústavu*

**LICENČNÍ SMLOUVA**
**POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO**

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Bc. Petr Zdráhal**
Id studenta: 49409
Bytem: Josefa Janáčka 949, 530 12 Pardubice
Narozen: 18. 03. 1983, Benešov
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

**Článek 1**
**Specifikace školního díla**

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Automatizovaná práce s webovými formuláři
Vedoucí/školitel VŠKP: Burget Radek, Ing., Ph.D.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

## Článek 2
## Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

## Článek 3
## Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

| | |
|---|---|
| ..... | ..... |
| Nabyvatel | Autor |

# Abstrakt

Diplomová práce se zabývá problematikou automatizace webových formulářů. Obsahuje stručné zasvěcení do problematiky značkovacích jazyků HTML a XHTML a popis prostředků pro tvorbu formulářů. Dále je v práci uvedena stručná charakteristika protokolu HTTP a značkovacího jazyka XML a je navržen způsob popisu formulářových dat pomocí XML dokumentu. Poté následuje popis implementace nástroje na vytvoření XML popisu z HTML dokumentu a popis implementace nástroje na automatizované odesílání dat z CSV souboru za použití XML popisu formuláře. V práci je uveden popis nejdůležitějších algoritmů použitých pro implementaci. V závěru jsou shrnuty dosažené výsledky a možná vylepšení do budoucna.



# Klíčová slova

HTML, XHTML, formuláře, HTTP, metoda GET, metoda POST, XML, DTD, C++, HTML parser, XML parser, CSV, CSV parser, Automatizace



# Abstract

This thesis deals with the web form automatization. It contains a short introduction to the HTML and XHTML markup languages and their facilities for the definition of forms. Next, there is a short overview of the HTTP protocol and XML markup language and a method of the form description using XML is proposed. The thesis contains a description of tool for creating XML file from HTML document and a description of a data sender tool using a CSV data file and an XML form description. Next, there is a description of the algorithms used in the implementation. The conclusion includes the achieved results and the improvements for the future.



# Keywords

HTML, XHTML, forms, HTTP, method GET, method POST, XML, DTD, C++, HTML parser, XML parser, CSV, CSV Parser, Automatization


# Citace

Petr Zdráhal: Automatizovaná práce s webovými formuláři, diplomová práce, Brno, FIT VUT v Brně, 2007

# Automatizovaná práce s webovými formuláři

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením
Ing. Radka Burgeta, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

## Poděkování

Velmi rád bych poděkoval vedoucímu mé diplomové práce Ing. Radku Burgetovi, Ph.D. za podnětné připomínky, trpělivost a odborné rady. Poděkování si jistě zaslouží i moje rodina, přítelkyně a přátelé za jejich trpělivost a podporu. Děkuji vám všem.

# Obsah

# Úvod

S postupným vývojem internetu jako celosvětové sítě a rozšiřováním této služby do všech domácností, dochází také k vývoji nástrojů a jazyků pro vytváření různých webových aplikací, které mohou sloužit nejenom k prezentaci určitých informací, ale mnohem častěji se začíná webových aplikací využívat k podnikání (různé druhy elektronických obchodů) a také pro tvorbu informačních systémů. S příchodem informačních systémů vyvstává problém s hromadným zpracováním formulářových dat.

Tímto problémem se zabývá diplomová práce s názvem „Automatizovaná práce s webovými formuláři“. Práce se zabývá problémem implementace nástroje pro automatické odeslání stejně strukturovaných dat pomocí daného webového formuláře. Jednoduše řečeno, výsledná aplikace bude automaticky čerpat předem daná data a ty pomocí stanoveného formuláře, který je součástí webové aplikace, odesílat protokolem HTTP místo nás. Odpadne starost s několikanásobným vyplňováním stejného formuláře.

Několik úvodních kapitol diplomové práce slouží jako teoretický úvod do dané problematiky. V první kapitole jsou nastíněny dva základní značkovací jazyky, které jsou v současné době nejvíce využívané pro tvorbu webových aplikací a s nimi souvisejícími formuláři. Jedná se o jazyky HTML a XHTML.

Druhá kapitola se zabývá problematikou protokolu HTTP, stavbou požadavků a odpovědí při komunikaci pomocí tohoto protokolu a především popisuje možné způsoby přenosu formulářových dat, které budou následně využity pro automatické odesílání dat.

Třetí kapitola popisuje moderní značkovací jazyk XML, který se v současné době dostává do popředí v oblasti sdílení a výměny dokumentů, a definuje možný popis formulářových dat pomocí XML dokumentu. Tento popis bude důležitou součástí pro implementaci obou nástrojů potřebných pro automatizaci formulářů. DTD jednoznačně definující strukturu XML dokumentu je součástí přílohy.

Po teoretickém úvodu již následují praktické kapitoly, které se zabývají návrhem, implementací a ovládáním obou nástrojů. Čtvrtá kapitola tedy obsahuje již zmíněný návrh a implementaci aplikace. V této kapitole jsou jednak navrhnuty základní moduly a třídy, ale také jsou zde rozebrány nejdůležitější algoritmy, stavové automaty a principy jak pro získání samotného XML popisu, tak také pro odesílání dat zpět na server.

Pátá kapitola obsahuje stručné seznámení s ovládáním aplikace, s používanými parametry na příkazové řádce a nechybí zde ani popis jednoduchého nastavená pro výchozí stav aplikace. Podrobnější manuál k programu je součástí přílohy.

Šestá a poslední kapitola obsahuje způsob testování aplikace a popis jednotlivých testovacích příkladů. Také zde jsou dosažené výsledky a omezení aplikace. Tato kapitola tedy shrnuje funkční možnosti implementované aplikace.

Diplomová práce navazuje na výsledky semestrální práce, ze které byly převzaty teoretické analýzy použitých technologií a navržená XML struktura. Dále se diplomová práce zabývá zejména implementací nástrojů, které automatizují časté vyplňování stejných formulářů.

# 1 Jazyky HTML a XHTML

HTML a XHTML jsou značkovací jazyky, které vychází ze SGML. Oba jsou si velice podobné, neboť XHTML vychází z HTML za dodržení pravidel vyplývajících z XML. Oba také slouží k publikaci dokumentů v celosvětové síti označované jako WWW (World Wide Web). Za specifikacemi HTML i XHTML stojí konsorcium W3C, které se snaží v poslední době prosadit XHTML jako jediný standard pro tvorbu webových stránek. HTML je již ukončená vývojová větev a dále se nerozvíjí.

V dalším textu je očekávána základní znalost syntaxe jazyků HTML a XHTML.

## 1.1 Historie

1989 – zahájení projektu WWW ve výzkumných laboratořích CERN (Švýcarsko)

1991 – zveřejněna první neformální specifikace jazyka HTML

1992 – první dostupná verze prohlížeče (řádkový režim prohlížení)

1993 – na světě pracuje kolem 50 serverů
– dokončen první grafický prohlížeč dokumentů (NCSA Mosaic)
– zveřejněn návrh specifikace HTML 2.0

1994 – Marc Andressen (autor prohlížeče Mosaic) zakládá společnost Mosaic Communications Corp. (uvádí nový prohlížeč pod názvem Netscape)
– založena nekomerční organizace WWWC (World Wide Web Consorcium)
– francouzský institut INRIA (Institut National de Recherche en Informatique et en Automatique) přebírá vývoj WWW

1995 – vyšla oficiální specifikace HTML 2.0
– Netscape neoficiálně rozšiřuje jazyk HTML 2.0 o tabulky a rámce (často označované jako HTML 3.0)

1996 – předběžná specifikace HTML 3.2, která ke specifikaci HTML 2.0 přidává tabulky (je tedy chudší než neoficiální HTML 3.0)
– uveden jazyk pro definici stylu CSS level 1

1997 – vydána oficiální specifikace HTML 3.2
– Microsoft uveřejnil neoficiální prvky dynamického HTML
– objevila se předběžná specifikace HTML 4.0

1998 – vydána specifikace XML 1.0
– zveřejněna oficiální specifikace HTML 4.0
– uvedena druhá verze jazyka pro definici stylu CSS level 2

1999 – vyšla nová revize HTML 4.0 označovaná jako HTML 4.01

2000 – uveřejněn jazyk XHTML 1.0 (Specifikace HTML 4.01 s integrovanými pravidly XML)
– XHTML Basic (XHTML upravené pro přenosná zařízení)

2001 – modularizovaná verze XHTML (možnost tvorby XHTML profilů pro různá zařízení) označovaná jako XHTML 1.1

2002 – první veřejný návrh XHTML 2.0

## 1.2 HTML

HyperText Markup Language neboli značkovací jazyk pro hypertext. Používá se pro publikování stránek na internetu pomocí systému WWW (World Wide Web). Tento jazyk vychází z rozsáhlejšího značkovacího jazyka SGML (Standard Generalized Markup Language). Jazyk HTML prošel dlouhým vývojem, kdy bylo zveřejněno několik oficiálních specifikací z nichž nejnovější je specifikace HTML 4.01. Touto specifikací byl vývoj standardu HTML zastaven a veškeré úsilí se nyní soustřeďuje na XHTML.

### 1.2.1 Verze HTML

- **HTML 2.0** – Standard vyvinula komunita IETF (Internet Engeneering Task Force). První návrh se objevil v roce 1993 a oficiální specifikace byla vydána v listopadu roku 1995. Je to první verze, která odpovídá syntaxi SGML. Kromě základních textových prvků přidává i interaktivní formuláře.
- **HTML 3.0** – Neoficiální verze specifikace, kterou přinesl Netscape v roce 1995. Tato specifikace rozšiřuje původní specifikaci HTML 2.0 o tabulky a rámce. Pro svoji složitost nebyla tato verze nikdy přijata jako standard.
- **HTML 3.2** – Standard již přineslo konsorcium W3C, které zaštiťuje podporu až doposud. Předběžná specifikace byla vydána v květnu 1996 a jako oficiální specifikace byla prezentována v lednu 1997. Ke specifikaci HTML 2.0 přidává tabulky, zarovnání textu a stylové elementy pro ovlivňování vzhledu.
- **HTML 4.0** – Předběžná specifikace vydána v prosinci roku 1997. K původní specifikaci HTML 3.2 byly přidány nové prvky pro tvorbu tabulek a formulářů, nově byly specifikovány rámce, plovoucí rámce, kaskádové styly, skriptování, objekty a internacionalizace. Tato verze se snaží dosáhnout původního účelu ve smyslu, že jednotlivé prvky by měly určovat sémantiku (význam) jednotlivých částí dokumentu a vzhled by měl být ovlivňován připojovanými styly. Některé prezentační elementy byly zavrženy.
- **HTML 4.01** – Oficiální revize HTML 4.0 byla vydána v prosinci 1999. Tato revize přináší řadu detailních upřesnění jazyka, úpravy v definici tabulky, rozšířené formuláře a kaskádové

styly. Je to poslední verze jazyka, která se již dále nevyvíjí. V současnosti je nahrazována jazykem XHTML, jehož základem je právě zmiňovaná specifikace HTML 4.01.

**Poznámka:**

HTML 4.x obsahuje tři varianty:

- Strict – zahrnuje všechny prvky a atributy, které nejsou deprecated (nedoporučené) a nejsou součástí rámců.
- Transitional – zahrnuje vše jako varianta Strict plus všechny značky a atributy, které jsou deprecated.
- Frameset – identická jako Transitional s tím, že obsahuje značky související s rámci.

### 1.2.2 Struktura HTML dokumentu

HTML dokument má předepsanou strukturu. Následující popis vystihuje základní schéma HTML dokumentu (některé prvky nejsou povinné, ale přesto je doporučeno dodržet danou kostru) :

1. Deklarace DTD – pro každou verzi HTML existuje definice pravidel pomocí DTD (Document Type Definition). Od verze 4.01 musí být odkaz na deklaraci DTD v dokumentu uveden pomocí klíčového slova DOCTYPE. DTD definuje pro danou verzi elementy, které je možné používat a s jakými atributy.
2. Kořenový element – element „html“ obaluje celý dokument. Je nepovinný ale je doporučeno ho používat.
3. Hlavička dokumentu – hlavičku obaluje element „head“ a jsou to metadata, která se vztahují k celému dokumentu. Definují název dokumentu, kódování, klíčová slova, nadpis, jazyk, použitý styl zobrazení, popis a jiné.
4. Tělo dokumentu – vymezeno elementem „body“ a obsahuje vlastní text dokumentu.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
       "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title> ... Titulek ... </title>
    ... Ostatní prvky hlavičky ...
  </head>
  <body>
    ... Tělo dokumentu ...
  </body>
</html>
```

**Poznámka:**

- zápis značek v HTML není citlivý na malá/velká písmena, tudíž můžete napsat jak <html> tak <HTML>; do budoucnosti je ale určitě lepší psát všechno malými písmeny, protože XHTML již je citlivé na velká/malá písmena

# 1.3 XHTML

eXtensible HyperText Markup Language. Rozšiřitelný značkovací jazyk pro publikování dokumentů v systému WWW vyvinutý konsorciem W3C. Vychází z poslední specifikace HTML 4.01, na kterou aplikuje pravidla z jazyka XML. V současné době je to hlavní standard prosazovaný konsorciem W3C a v brzké budoucnosti by měl plně nahradit HTML.

## 1.3.1 Verze XHTML

- **XHTML 1.0** – Oficiální specifikace byla uveřejněna v roce 2000. Jedná se o jazyk vycházející z HTML 4.01, kdy specifikaci HTML 4.01 spojuje s pravidly XML (především to, že každá značka je párová a hodnoty atributů musí být v uvozovkách). Je tedy přísnější a čistější než HTML 4.01.
- **XHTML 1.1** – Specifikace byla prezentována v roce 2001. Je postavena na standardu XHTML 1.0 Strict, z něhož odstraňuje všechny konstrukce, které byly označeny jako deprecated (překonané). V této specifikaci je již uplatněn princip modularizace, kdy jsou jednotlivé elementy rozděleny dle svých vlastností do fiktivních modulů a celý standard sestává z kombinací těchto modulů. To nám dává možnost definovat různé standardy pro různá zobrazovací zařízení (PC, PDA, telefon atd.)
- **XHTML 2.0** – První veřejný návrh je z roku 2002 a v současné době ještě stále není oficiální verze. I přesto, že vychází z HTML 4.01, XHTML 1.0 a XHTML 1.1, není zpětně kompatibilní s těmito specifikacemi.

**Poznámka:**

XHTML 1.0 obsahuje tři varianty:

- Strict – popisuje strukturální části XHTML 1.0, ale neposkytuje žádné značky týkající se grafické úpravy a rámců.
- Transitional – zahrnuje vše jako varianta Strict, plus značky grafického formátování
- Frameset – identická jako Transitional s tím, že obsahuje značky související s rámci

### 1.3.2 Hlavní rozdíly oproti HTML

- Dokument musí být well-formed – jedná se především o korektní ukončování značek a korektní vnořování značek.
- Názvy značek elementů a jejich atributů musí být zapsány malými písmeny.
- Hodnota atributu musí být vždy v uvozovkách.
- Atributy, které šlo zapisovat ve zkrácené formě, se v XHTML musí zapisovat v plné syntaktické formě (např. místo pouze checked je nutné psát checked="checked").
- Prázdné elementy musí být zakončeny ukončovací značkou (např. <br /> nebo <br></br>).
- Používání atributu id jako jednoznačného identifikátoru v celém dokumentu.
- Používání pouze blokových elementů ve formuláři.

### 1.3.3 Modularizace

Modularizace je rozdělení XHTML prvků do fiktivních modulů podle jejich specifikace. Tento princip vzniknul jako reakce na požadavky prohlížení obsahu WWW i na jiných zařízení než je klasické PC. Modularizace nám tudíž dává možnost slučování různých modulů a vytváření tak různých skupin, které můžeme nazývat jako XHTML profil. Na tomto principu je odvozena například specifikace jazyka XHTML Basic (profil odvozený pro malá přenosná zařízení skládajícího se jen ze základních modulů). XHTML 1.1 je již plně modularizovaná specifikace jazyka.

### 1.3.4 Struktura XHTML dokumentu

XHTML dokument má předepsanou strukturu. Následující popis vystihuje základní schéma XHTML dokumentu (některé prvky nejsou povinné, ale přesto je doporučeno dodržet danou kostru) :

1. Oficiální identifikace XHTML dokumentu – znakové kódování dokumentu a kódování jazyka. Identifikace se uvádí v elementu <?xml …… ?>
2. Deklarace DTD – pro každou verzi XHTML existuje definice pravidel pomocí DTD (Document Type Definition). Odkaz na deklaraci DTD v dokumentu je uveden pomocí klíčového slova DOCTYPE. DTD definuje pro danou verzi elementy, které je možné používat a s jakými atributy.
3. Kořenový element – element „html" obaluje celý dokument.
4. Hlavička dokumentu – hlavičku obaluje element „head" a jsou to metadata, které se vztahují k celému dokumentu. Definují název dokumentu, kódování, klíčová slova, nadpis, jazyk, použitý styl zobrazení, popis a jiné.
5. Tělo dokumentu – vymezeno elementem „body" a obsahuje vlastní text dokumentu.

```
<?xml version=”1.0” encoding=”iso-8859-2”?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns=”http://www.w3.org/1999/xhtml” xml:lang=”cs” lang=”cs”>
  <head>
    <meta http-equiv=”content-type”
          content=”text/html; charset=iso-8859-2” />
    <meta http-equiv=”content-language” content=”cs” />
    <title>...Titulek...</title>
  </head>
  <body>
    ...Tělo dokumentu...
  </body>
</html>
```

**Poznámka:**

- povinný zápis značek malými písmeny
- je-li uvedeno DTD, musí být uvedeno naprosto přesné DTD
- dobře formátovaný XHTML dokument by měl obsahovat všechny výše uvedené náležitosti

# 1.4 Formuláře v HTML/XHTML

Jednou z předností webových aplikací v systému WWW je možnost interakce s uživatelem. Ta je primárně aplikována pomocí hypertextových odkazů. Taková interakce by ovšem nebyla dostačující. Proto (X)HTML obsahuje prvky, pomocí nichž lze vytvářet interaktivní formuláře nebo vkládat ovládací prvky do dokumentu. Informace získané pomocí interaktivního formuláře jsou předány nějakému skriptu na straně serveru, který je dále zpracuje. Formulář má proto dvě hlavní části – viditelnou část v (X)HTML stránce a neviditelnou část na straně serveru (skript). Vzhledem k charakteru a zadání diplomové práce nebudeme uvažovat možnost zpracování dat z formuláře skriptem na straně klienta.

## 1.4.1 Základní principy a konstrukce formuláře

Celý formulář je vždy ohraničen elementem *form*. Uvnitř tohoto kontejneru se vyskytují veškeré ovládací prvky (tlačítka, textová pole, zaškrtávací pole, rolovací seznamy atd.) potřebné pro daný formulář. Vzhled těchto ovládacích prvků závisí z velké části na použitém prohlížeči, ale funkce je vždy stejná. Kromě ovládacích prvků se uvnitř formuláře mohou vyskytovat veškeré ostatní prvky běžně používané v těle (X)HTML dokumentu. Hlavní účel elementu *form* je tedy ve sdružení

ovládacích prvků do jednoho bloku, aby se s nimi dalo pracovat. Formulářů se může v jednom HTML dokumentu vyskytovat libovolný počet, ale nesmějí se vnořovat do sebe.

```
<form[ atributy]>
  ... Prvky formuláře (select, input, textarea) ...
  ... Další různé (X)HTML prvky ...
  ...
</form>
```

### 1.4.2 Formulářové značky

V této kapitole budou vysvětleny všechny značky, které jsou podstatné z hlediska našeho zpracování formuláře. U jednotlivých značek jsou popsány i všechny atributy podstatné pro další zpracování.

**<form> -** párová značka, která vymezuje formulář v daném dokumentu

```
<form      id="identifikátor"
           name="unikátní jméno formuláře"
           action="odkaz na script pro zpracování formuláře"
           method="GET|POST"
           accept="seznam MIME typů obsahu formuláře"
           enctype="typ kódování odesílaných dat" >
    ... ovládací prvky ...
</form>
```

- **id** - unikátní identifikátor prvku, atribut „id“, sdílí totožný prostor názvů s atributy „name“ (name má přednost před id, jsou-li uvedeny oba, jinak může být použit jeden nebo druhý)
- **name** - unikátní jméno formuláře
- **action** - atribut určuje URI adresu skriptu, který formulář zpracuje; může to být URI s protokolem http nebo mailto; chování pro jinou než URI hodnotu není definováno
- **method** - atribut specifikuje http metodu, která bude použita pro odeslání dat formuláře
  - GET – implicitní metoda; data jsou odesílána pomocí URI (URI?jméno=hodnota&jméno=hodnota atd.); hodnoty formuláře jsou redukovány na ASCII hodnoty
  - POST – metoda odesílá data v těle http požadavku
- **accept** - seznam MIME typů (oddělených čárkou), jehož prostřednictvím server zpracovávající tento formulář jej bude moci korektně ovládat

- **enctype** - použití v případě, že metoda odesílání je POST; implicitní hodnota je "application/x-www-form-urlencoded"; je li v odesílaných datech obsažen soubor nebo odesíláme ne-ASCII znaky, musí být jako hodnota enctype zadána "multipart/form-data"

**<input> -** nepárová značka, která slouží k definici jednoho ovládacího prvku typu vstupní pole

```
<input     id="identifikátor"
           type="typ prvku"
           name="jméno prvku"
           value="hodnota prvku"
           maxlength="maximální počet znaků"
           accept="typ souboru"
           alt="alternativní text"
           checked="zaškrtnutí prvku"
           readonly="prvek pouze pro čtení"
           disabled="prvek vypnutý" >
```

- **id** - unikátní identifikátor prvku, atribut „id", sdílí totožný prostor názvů s atributy „name" (name má přednost před id, jsou-li uvedeny oba, jinak může být použit jeden nebo druhý)
- **type –** typ prvku
    - text – implicitní typ; jednořádkové textové pole; atribut value specifikuje počáteční hodnotu
    - password – podobné jako text; vpisovaná hodnota je zastírána např. hvězdičkami
    - checkbox – zaškrtávací políčko; atribut value specifikuje hodnotu vracenou v případě zaškrtnutí políčka; při odesílání formuláře se odesílají pouze zaškrtnuté ovládací prvky checkbox
    - radio – volba právě jedné možnosti; prvky typu radio se pomocí stejného jména sdružují do skupin (je-li jeden zapnut, tak se ostatní nastaví jako vypnuto); ve skupině prvků typu radio je implicitně nastaven ten, který má nastaven atribut checked (pokud takto žádný nastaven není, pak se nastavuje první prvek ve skupině); atribut value specifikuje hodnotu vracenou pro danou volbu
    - submit – tlačítko pro odesílání dat na server; atribut value specifikuje popis tlačítka
    - reset – tlačítko pro obnovení implicitních hodnot; atribut value specifikuje popis tlačítka
    - hidden – skryté pole; hodnota je odesílána s formulářem; atributem value lze definovat pevnou hodnotu nezadávanou uživatelem
    - image – tlačítko s obrázkem; atribut src specifikuje URI obrázku; tlačítko odesílá data z formuláře a souřadnice odpovídající kliknutí na obrázek; hodnoty jsou měřeny

v pixelech od levého a horního okraje; souřadnice x a y se odesílají jako hodnoty „name.x=x“ a „name.y=y“, kde name je hodnota atributu name
    - file – pole pro vybrání souboru; umožňuje spolu s formulářem odeslat na server jeden nebo více lokálních souborů; atributem accept lze vymezit přípustné typy souborů
    - button – tlačítko bez implicitní činnosti; činnost je definovaná pomocí skriptu
- **name** – jméno prvku; musí být v rámci formuláře jednoznačné; ve všech prvcích kromě typu „submit“ a „reset“ je tento atribut vyžadován
- **value** – počáteční hodnota prvku dle typu
    - text, password, hidden – počáteční hodnota prvku
    - checkbox, radio – požadován; definuje výsledek, je-li prvek vybrán kliknutím
    - submit, reset, button – text na tlačítku
    - image – symbolický výsledek přenášený na skript
    - file – nemůže být použit s tímto typem
- **maxlength –** maximální počet znaků akceptovatelných u prvků „text“ a „password“
- **accept** – seznam MIME typů přípustných pro pole typu „file“, které je ochoten skript zpracovávající formulář přijímat
- **alt** – alternativní text
- **checked** – implicitní zaškrtnutí pole typu „checkbox“ nebo implicitní volba pole typu „radio“
- **readonly** – pole je pouze pro čtení; atribut lze použit pouze s typem „text“
- **disabled** – znepřístupnění ovládacího prvku

**<textarea> -** párová značka, která se používá pro definici víceřádkového vstupního pole formuláře. Smí se vyskytnout pouze v prvku „form“ a nesmí být vnořována. Text uzavřený mezi značkami je implicitní zobrazovaný text. Zalomení řádku je zachováno a délka textu není omezena.

```
<textarea  id="identifikátor"
           name="jméno prvku"
           readonly="prvek pouze pro čtení"
           disabled="prvek vypnutý" >
      ... Text ...
</textarea>
```

- **id** - unikátní identifikátor prvku, atribut „id“, sdílí totožný prostor názvů s atributy „name“ (name má přednost před id, jsou-li uvedeny oba, jinak může být použit jeden nebo druhý)
- **name** – jméno prvku; musí být v rámci formuláře jednoznačné (odesílá se na server)
- **readonly** – pole je pouze pro čtení, není možná modifikace
- **disabled** – znepřístupnění ovládacího prvku (nelze zapsat text nebo ovládací prvek vybrat)

**<select> -** párová značka, pomocí které lze vytvořit rozbalovací seznam a slouží ke specifikaci nabídky s řadou volitelných vstupů definovaných pomocí prvku „option“. V prvku „select“ se předpokládá použití alespoň jednoho prvku „option“. Prvek „select“ může obsahovat přednastavené volby. Chování UA (user agent – prohlížeč) při nestandardním nastavení není nijak specifikováno a závisí to čistě na prohlížeči.

```
<select    id="identifikátor"
           name="jméno prvku"
           multiple="výběr více položek"
           disabled="prvek vypnutý" >
      <optgroup>...
           <option> ... </option>
           ...
</select>
```

- **id** - unikátní identifikátor prvku, atribut „id“, sdílí totožný prostor názvů s atributy „name“ (name má přednost před id, jsou-li uvedeny oba, jinak může být použit jeden nebo druhý)
- **name** – jméno prvku; musí být v rámci formuláře jednoznačné (odesílá se na server)
- **multiple** – povoluje možnost výběru více než jedné položky současně
- **disabled** – znepřístupnění ovládacího prvku (ovládací prvek nelze vybrat)

**<optgroup> -** párová značka, která slouží k definici skupiny nabídek v prvku „select“. Značka se smí vyskytnout pouze v prvku „select“ a musí obsahovat alespoň jeden prvek „option“.

```
<optgroup  id="identifikátor"
           label="návěstí pro skupinu voleb"
           disabled="prvek vypnutý" >
      <option> ... </option>
      ...
</optgroup>
```

- **id** - unikátní identifikátor prvku
- **label** – nadpis/návěstí skupiny nabídek
- **disabled** – znepřístupnění skupiny nabídek (nelze provádět volby)

**<option> -** párová značka, pomocí které se definuje jedna položka nabídky uvnitř prvku „select“. Značka se může vyskytovat pouze v prvku „select“ nebo „optgtoup“ a nelze ji vnořovat.

```
<option    id="identifikátor"
           value="hodnota"
           selected="výběr více položek"
           disabled="prvek vypnutý" >
      ... text nabídky ...
</option>
```

- **id** - unikátní identifikátor prvku
- **value** – hodnota prvku pro odeslání v případě výběru; pokud není uvedena, tak se odešle obsah nabídky
- **selected** – specifikuje implicitně vybranou nabídku
- **disabled** – znepřístupnění nabídky (volba nelze vybrat)

**<button>** - párová značka, která slouží k definici tlačítka v dokumentu

```
<button    id="identifikátor"
           type="typ prvku"
           name="jméno tlačítka"
           value="hodnota prvku"
           disabled="prvek vypnutý" >
      ... nápis / obrázek tlačítka ...
</button>
```

- **id** - unikátní identifikátor prvku, atribut „id“, sdílí totožný prostor názvů s atributy „name“ (name má přednost před id, jsou-li uvedeny oba, jinak může být použit jeden nebo druhý)
- **type –** typ tlačítka
    - button – jednoduché tlačítko (ovládání skriptem)
    - submit – tlačítko pro odesílání formuláře na server (implicitní hodnota)
    - reset – tlačítko pro obnovení implicitních hodnot formuláře
- **name** – jméno tlačítka
- **value** – definuje výsledek pro tlačítko typu „button“
- **disabled** – znepřístupnění tlačítka (není možnost ovládání)

# 2 Protokol HTTP

HyperText Transfer Protocol. HTTP je internetový protokol aplikační vrstvy určený původně pro výměnu hypertextových dokumentů ve formátu HTML. V současné době je používán i pro přenos dalších informací. Pomocí rozšíření MIME umí přenášet jakýkoliv soubor (podobné jako email), používá se společně s XML pro tzv. webové služby a pomocí aplikačních bran zpřístupňuje i další protokoly, jako např. FTP nebo SMTP. K protokolu HTTP existuje také jeho bezpečnější verze nazvaná jako HTTPS.

Protokol HTTP se používá v systému WWW od roku 1990. První verze, označovaná jako HTTP/0.9, byla jednoduchým protokolem pro přenos surových dat přes internet. HTTP/1.0, který byl definován v RFC 1945, vylepšil původní specifikaci protokolu povolením formátu MIME pro jednotlivé zprávy, obsahujícím metainformace o přenášených datech. Nicméně HTTP/1.0 neuspokojilo požadavky pro hierarchické proxy servery, kešování, persistentní spojení nebo virtuální hosty. Na základě této nekompletní implementace byl protokol HTTP/1.0 přepracován, aby dvě komunikující strany mohly specifikovat svoje správné možnosti. Tento přepracovaný protokol je nazýván jako HTTP/1.1.

HTTP protokol neřeší transport dat mezi klientem a serverem a nestará se o detekci přenosových chyb. Tyto vlastnosti řeší protokol transportní vrstvy nazývaný TCP. Protokol HTTP je typu klient-server, pracuje v modelu požadavek-odpověď (mezi jednotlivými spojeními není uchovávána žádná informace) a jedná se o bezstavový protokol. Dvojice požadavek-odpověď tvoří ucelenou jednotku komunikace a mají shodnou strukturu.

Komunikace mezi klientem a serverem probíhá obvykle pomocí protokolu TCP/IP a to především na portu 80. Tento port je v protokolu TCP implicitním portem pro aplikační protokol HTTP.

## 2.1 Struktura požadavku/odpovědi HTTP

Vzhledem k architektuře klient-server protokolu HTTP máme dva druhy zpráv. Požadavky na zprávy HTTP odesílá klient a odpovědi odesílá server.

- Request – požadavek klienta
- Response – odpověď serveru

Požadavek i odpověď mají předem daný formát a veškerá komunikace probíhá v textové podobě. Obecně můžeme strukturu požadavku i odpovědi definovat následovně. Každý řádek je oddělen dvojící CRLF (šestnáctkově 0D0A). Jako první je stavový řádek (podrobnější popis je uveden v následujících kapitolách), následuje seznam hlaviček zprávy, povinná mezera, která odděluje hlavičky od těla zprávy, a nakonec tělo zprávy

```
Stavový řádek CRLF
Hlavičky CRLF
CRLF (povinná mezera)
Tělo
```

### 2.1.1 Požadavek

směr klient->server (žádost o nějaký zdroj definovaný pomocí URI)

```
Request      = Request-Line
               (( general-header
               | request-header
               | entity-header ) CRLF)*
               CRLF
               [ message-body ]
```

Požadavek je uveden v řádku Request-Line, za ním následují hlavičky oddělené CRLF, prázdný řádek a na konci může být uvedeno tělo.

```
Request-Line      = Method SP Request-URI SP HTTP-Version CRLF
```

V první řádku požadavku je uvedena metoda HTTP protokolu, mezera, URI zdroje na který směřujeme požadavek, mezera a verze HTTP protokolu.

Metody mohou být následující – OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT.

### 2.1.2 Odpověď

směr server->klient (odpověď na klientův požadavek)

```
Response     = Status-Line
               (( general-header
               | response-header
               | entity-header ) CRLF)*
               CRLF
               [ message-body ]
```

Odpověď je uvedena v řádku Status-Line, za ním následují hlavičky oddělené CRLF, prázdný řádek a na konci může být uvedeno tělo.

```
Status-Line        = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

V první řádku odpovědi je uvedena verze HTTP protokolu, mezera, číselný kód stavu a nakonec textový doplněk stavu.

### 2.1.3 Hlavičky

Zprávy protokolu HTTP mohou obsahovat různé hlavičky. Některé se používají pouze v požadavcích a jiné pouze v odpovědích. Proto dělíme hlavičky do čtyř základních skupin

1. General-header – obecná pole, mohou se vyskytovat jak v požadavku tak v odpovědi
2. Request-header – hlavičky požadavků
3. Response-header – hlavičky odpovědí
4. Entity-header – hlavičky popisující předávanou entitu

V následujícím seznamu jsou vyjmenované hlavičky pro jednotlivé skupiny.

1. Cache-Control, Connection, Date, Pragma, Trailer, Transfer-Encoding, Upgrade, Via, Warning
2. Accept, Accept-Charset, Accept-Encoding, Accept-Language, Authorization, Expect, From, Host, If-Match, If-Modified-Since, If-None-Match, If-Range, If-Unmodified-Since, Max-Forwards, Proxy-Authorization, Range, Referer, TE, User-Agent
3. Accept-Ranges, Age, ETag, Location, Proxy-Authenticate, Retry-After, Server, Vary, WWW-Authenticate
4. Allow, Content-Encoding, Content-Language, Content-Length, Content-Location, Content-MD5, Content-Range, Content-Type, Expires, Last-Modified, extension-header

## 2.2 Způsoby přenosu formulářových dat

Formulářová data se přenášejí pomocí HTTP metod GET a POST. Jejich použití závisí nejen na charakteru zpracovávaného formuláře, ale především na omezujících možnostech použití metod. Hlavním rozdílem je, že metoda GET přenáší formulářová data v hlavičce požadavku (pomocí URI), zatímco metoda POST přenáší formulářová data v těle HTTP požadavku. Při návrhu formuláře je nutné zvážit vhodnost použití jednotlivých metod.

### 2.2.1 Metoda GET

Metoda GET odesílá formulářová data prostřednictvím URI. Z toho plyne jedno důležité omezení, na které je třeba si dát pozor. Tímto omezením je, že délka URI je obvykle omezena na 2048 bytů.

Metoda GET je tedy nevhodná pro odesílání velkého množství dat, zejména pro odesílání binárních dat, jako jsou soubory atd.

Zakódování formulářových dat je stanoveno následovně. Jména ovládacích prvků a jejich hodnot jsou escapovány (mezery a ne-ASCII znaky jsou převedeny na únikové sekvence) takto: mezery jsou nahrazeny znakem '+', rezervované znaky jsou nahrazeny zápisem '%HH', kde HH je hexadecimální hodnota znaku a řádkové zlomy jsou interpretovány jako pár CRLF ('%0D%0A'). Ovládací prvky se v seznamu skládají podle pořadí výskytu v dokumentu ve tvaru: „jméno_prvku=hodnota" a páry jsou navzájem oddělené znakem '&'.

Na následujícím příkladu je možno vidět všechna pravidla zakódování formulářových dat stanovená v předešlém odstavci. Jedná se o odeslání jednoduchého formuláře, který obsahuje prvek „input" typu „text" (Name), „radio" (Main), „submit" (Send), „reset" (Reset), prvek „textarea" (Text) a prvek „select" (Type). Data jsou zakódována v URI (na výpisu zvýrazněno modře). Požadavek neobsahuje žádné tělo.


Obrázek 1: Testovací formulář pro odesílání dat pomocí metody GET

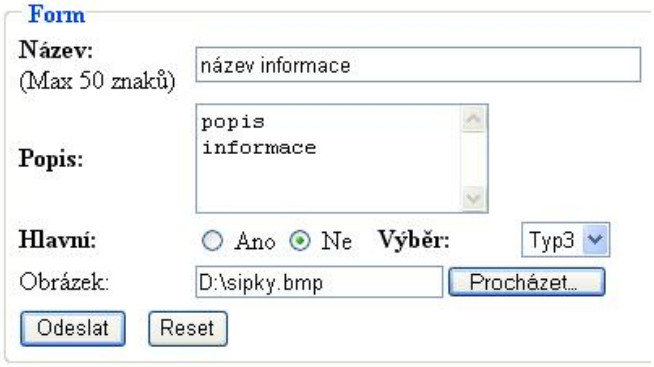
```
GET /~xzdrah03/pokus2.php?Name=n%E1zev+informace&
    Text=popis%0D%0Ainformace%0D%0A&Main=n&Type=3&Send=Odeslat HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/
    x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-
    powerpoint, application/msword, application/x-icq, */*
Referer: http://www.stud.fit.vutbr.cz/~xzdrah03/pokus2.php
Accept-Language: cs
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
     Embedded Web Browser from: http://bsalsa.com/; .NET CLR 1.1.4322)
Host: www.stud.fit.vutbr.cz
Connection: Keep-Alive
```

### 2.2.2 Metoda POST

Metoda POST odesílá formulářová data v těle HTTP požadavku, které nemá žádná omezení co do velikosti obsahu. Tato metoda je tedy vhodná pro přenos rozsáhlých dat a především souborů.

Ve standardních formulářích (X)HTML je možné použít metodu POST dvěma způsoby. Záleží na nastavení atributu „enctype“ elementu „form“. V případě “application/x-www-form-urlencoded” jsou data escapována stejným způsobem jako je popsán v metodě GET a vložena do těla dokumentu. Pokud bude nastaveno “multipart/form-data” jsou data přenášena ve svém původním tvaru. Druhý způsob je vhodný zejména pro přenos ne-ASCII znaků a binárních dat.

Na následujícím příkladu je možno vidět přenos formulářových dat v těle HTTP požadavku. Jedná se o odeslání jednoduchého formuláře, který obsahuje prvek „input“ typu „text“ (Name), „radio“ (Main), „file“ (Picture), „submit“ (Send), „reset“ (Reset), prvek „textarea“ (Text) a prvek „select“ (Type). Data z formulářových polí i binární data obrázku (pro zkrácení je uvedeno jen část binárních dat) jsou v těle požadavku (na výpisu zvýrazněné modře).




Obrázek 2: Testovací formulář pro odesílání dat pomocí metody POST

1. **“application/x-www-form-urlencoded”**

```
POST /~xzdrah03/pokus3.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/
    x-shockwave-flash, application/vnd.ms-excel,application/vnd.ms-
    powerpoint, application/msword, application/x-icq, */*
Referer: http://www.stud.fit.vutbr.cz/~xzdrah03/pokus3.php
Accept-Language: cs
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
    Embedded Web Browser from: http://bsalsa.com/; .NET CLR 1.1.4322)
Host: www.stud.fit.vutbr.cz
Content-Length: 81
```

```
Connection: Keep-Alive
Cache-Control: no-cache

Name=n%E1zev+informace&Text=popis%0D%0Ainformace%0D%0A&Main=n&Type=3&Send=
Odeslat
```

2. **"multipart/form-data"**

```
POST /~xzdrah03/pokus1.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/
      x-shockwave-flash, application/vnd.ms-excel, application/
      vnd.mspowerpoint, application/msword, application/x-icq, */*
Referer: http://www.stud.fit.vutbr.cz/~xzdrah03/pokus1.php
Accept-Language: cs
Content-Type: multipart/form-data; boundary=---------------------------
      7d61f414210616
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
      Embedded Web Browser from: http://bsalsa.com/; .NET CLR 1.1.4322)
Host: www.stud.fit.vutbr.cz
Content-Length: 2184
Connection: Keep-Alive
Cache-Control: no-cache

-----------------------------7d61f414210616
Content-Disposition: form-data; name="Name"

n.zev informace
-----------------------------7d61f414210616
Content-Disposition: form-data; name="Text"

popis
informace

-----------------------------7d61f414210616
Content-Disposition: form-data; name="Main"

n
-----------------------------7d61f414210616
Content-Disposition: form-data; name="Type"

3
```

```
-----------------------------7d61f414210616
Content-Disposition: form-data; name="Picture"; filename="D:\sipky.bmp"
Content-Type: image/bmp

<BINÁRNÍ OBSAH SOUBORU>

-----------------------------7d61f414210616
Content-Disposition: form-data; name="Send"

Odeslat
-----------------------------7d61f414210616--
```

# 3 Popis formulářových dat a jejich hodnot pomocí XML

## 3.1 XML

XML (eXtensible Markup Language – rozšiřitelný značkovací jazyk) je obecný značkovací jazyk standardizovaný konsorciem W3C. Jedná se o velmi flexibilní textový formát odvozený ze SGML. Jeho použití je především pro ryze strukturované dokumenty, tj. dokumenty obsahující strukturované nebo strukturovatelné informace. XML není určen jen pro texty, ale poradí si s jakýmikoliv daty i databázovými.

Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jazyk umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se potom definuje připojeným stylem. Další možností je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML.

Původní jazyk pro publikování HTML již přestal vyhovovat především pro svou složitost, která vznikla jeho postupným rozšiřováním. Jazyk XML nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a také jeho syntaxe je podstatně přísnější, než HTML.

### 3.1.1 Verze XML

V současné době jsou k dispozici dvě verze XML a to 1.0 a 1.1.

- **Verze 1.0** – Původně definovaná v roce 1998. Od té doby prošla menšími revizemi bez vydání nové verze. V současné době existuje 4. revize vydaná 16.srpna 2006. Je široce používaná a stále doporučená pro hlavní použití.
- **Verze 1.1** – Poprvé zveřejněna 4.února 2004, ve stejný den jako XML 1.0 ve své třetí revizi. V současné době existuje druhé vydání, které bylo zveřejněno 16.srpna 2006. Obsahuje nové rysy (některé jsou problematické), které mají zjednodušit XML pro použití v určitých třídách uživatelů. XML 1.1 není příliš používáno a je doporučeno jenom v případě, že je potřeba použít jejích nových rysů.

Obě verze se liší v požadavcích na použité znaky v názvech elementů a atributů. Verze 1.0 dovolovala pouze užívání znaků platných ve verzi Unicode 2.0, která obsahuje většinu světových

písem, ale neobsahuje později přidané sady jako je Mongolština a jiné. Verze XML 1.1 zakazuje pouze řídící znaky, což znamená, že mohou být použity jakékoli jiné znaky.

Je třeba poznamenat, že omezení ve verzi 1.0 se vztahuje pouze na názvy elementů a atributů. Jinak obě verze dovolují v obsahu dokumentu jakékoli znaky. Verze 1.1 je tedy nutná, pokud potřebujeme psát názvy elementů v jazyku, který byl přidán do Unicode později.

### 3.1.2 Struktura XML dokumentu

Struktura XML dokumentu je velice jednoduchá, skládá se pouze ze značek a obsahu. Existuje několik základních druhů značek, které se mohou v dokumentu vyskytnout :

- Zpracovatelské instrukce (Pis – Processing Instructions)
- Deklarace definice dokumentu (DTD)
- Prvky (elements)
- Komentáře
- Entity
- CDATA sekce
- Bílé znaky

Minimální XML dokument:

```
<?xml version=”1.0” encoding=”iso-8859-2”?>
<dokument>
  ... Tělo dokumentu ...
</dokument>
```

### 3.1.3 DTD

DTD (Document Type Definition) je definice syntaxe konkrétního značkovacího jazyka zapsaná prostřednictvím SGML. DTD obsahuje definici všech prvků tohoto konkrétního značkovacího jazyka a jejich atributů, které lze v dokumentu použít. Prostřednictvím SGML je tato definice zapsána ve standardizované podobě a je snadno zpracovatelná.

Deklarace může být externí, tzn. že dokument obsahuje odkaz na externí soubor, který tuto deklaraci obsahuje, nebo interní, tzn. že všechny deklarace jsou uvedeny přímo v dokumentu. Oba dva případy lze navzájem kombinovat, a tímto způsobem pomocí lokálních definic a parametrů upravit DTD v externím souboru. Definice obsažené v dokumentu jsou nadřazené definicím v externím souboru.

## 3.2 Návrh XML struktury

V této kapitole je popsán návrh XML struktury, která bude použita pro automatizaci webových formulářů. XML struktura vychází z elementů klasických formulářových prvků, které zjednodušuje a extrahuje jenom podstatné atributy. Odpovídající definice dokumentu v podobě DTD je připojena ke zprávě jako Příloha 1.

### 3.2.1 XML elementy

**<form> -** párová značka, která vymezuje formulář, je velice podobná stejně pojmenovanému elementu v HTML, avšak zahrnuje pouze podstatné atributy

```
<form      id="identifikátor"
           action="odkaz na script pro zpracování formuláře"
           method="GET|POST"
           accept="seznam MIME typů obsahu formuláře"
           enctype="typ kódování odesílaných dat" >
      ... elementy ...
</form>
```

- **id** - unikátní identifikátor formuláře – odvozen z atributu name nebo id formuláře
- **action** - atribut určuje URI adresu skriptu, který formulář zpracuje; může to být URI s protokolem HTTP nebo mailto
- **method** - atribut specifikuje HTTP metodu, která bude použita pro odeslání dat formuláře
    - GET – data jsou odesílána pomocí URI
    - POST – metoda odesílá data v těle HTTP požadavku
- **accept** - seznam MIME typů (oddělených čárkou)
- **enctype** – způsob kódování
    - “application/x-www-form-urlencoded”
    - “multipart/form-data”

**<element> -** párová značka, která určuje jeden konkrétní ovládací prvek formuláře; tato značka nahrazuje prvek input typu text, password, submit, reset, hidden, image, file

```
<element   id="identifikátor"
           maxlength="maximální počet znaků"
           accept="typ souboru"
           readonly="prvek pouze pro čtení"
           column="sloupec v csv souboru" >
      <label> popisek prvku </label>
```

```
    <type> typ prvku </type>
    <value> hodnota prvku </value>
    <alt> náhradní hodnota </alt>
</element>
```

- **id** - unikátní identifikátor prvku – odvozen z atributu name nebo id prvku
- **maxlength** – maximální počet znaků zejména pro prvky text a password
- **accept** - MIME typ souboru akceptovaného prvkem file
- **readonly –** označuje prvek pouze pro čtení
- **column –** tento atribut musí nastavit uživatel, aby bylo jasné mapování na daný sloupec v csv souboru s daty
- **label** – popisek prvku, slouží jen pro lepší vizuální orientaci
- **type** – označuje typ prvku, ze kterého byl tento element vytvořen
- **value** – hodnota prvku, která se bude odesílat
- **alt** – alternativní popisek pro prvek typu image

**<selection> -** párová značka, která nahrazuje prvek radio a select (jedná se o stejné výběrové prvky s rozdílem grafického znázornění, které nás nezajímá)

```
<selection      id="identifikátor"
                multiple="mnohonásobný výběr"
                column="sloupec v csv souboru" >
    ... prvky itemsel ...
</selection>
```

- **id** - unikátní identifikátor prvku – odvozen z atributu name nebo id prvku
- **multiple** – označuje, zda je možný výběr více hodnot
- **column** – tento atribut musí nastavit uživatel, aby bylo jasné mapování na daný sloupec v csv souboru s daty; pokud je „selection“ typu „multiple“, bude atribut „column“ přítomen u všech vnitřních elementů „itemsel“, aby bylo možné nastavit mapování na sloupce u každého zvlášť

**<itemsel> -** párová značka, která nahrazuje prvek checkbox, použitím v prvku selection slouží jako seznam hodnot výběru

```
<itemsel   id="identifikátor"
           selected="zda je vybráno"
           readonly="prvek pouze pro čtení"
           column="sloupec v csv souboru"
```

```
            csvvalue="hodnota hledaná v csv souboru" >
        <value> hodnota prvku </value>
        <description> popis možnosti </description>
</itemsel>
```

- **id** - unikátní identifikátor prvku – odvozen z atributu name nebo id prvku
- **selected** – označuje, zda je možnost vybrána
- **readonly –** označuje prvek pouze pro čtení
- **column –** tento atribut musí nastavit uživatel, aby bylo jasné mapování na daný sloupec v csv souboru s daty
- **csvvalue** – tento atribut může nastavit uživatel a označuje hodnotu, která bude hledaná v csv souboru v případě, že se jedná o typ „checkbox" nebo je tento element součásti selection s atributem „multiple". Pokud je tento atribut prázdný, bude se v případě typu „checkbox" hledat v csv souboru hodnota „value" a v případě, že jde o výběr v selection, tak hodnota „description". Pokud je v daném sloupci csv souboru nalezena daná hodnota, je prvek zaškrtnut (checkbox), nebo vybrán (select).
- **value –** hodnota prvku, která se bude odesílat
- **description –** popisek dané možnosti pro výběr (informativní charakter)

**<textarea> -** párová značka, která nahrazuje prvek textarea

```
<textarea  id="identifikátor"
           readonly="prvek pouze pro čtení"
           column="soupec v csv souboru" >
        <label> popisek prvku </label>
        <text> text, který obsahuje </text>
</textarea>
```

- **id** - unikátní identifikátor prvku – odvozen z atributu name nebo id prvku
- **readonly –** označuje prvek pouze pro čtení
- **label –** popisek prvku (informativní charakter)
- **text –** text samotného elementu, který zachovává veškeré formátování

# 4 Implementace

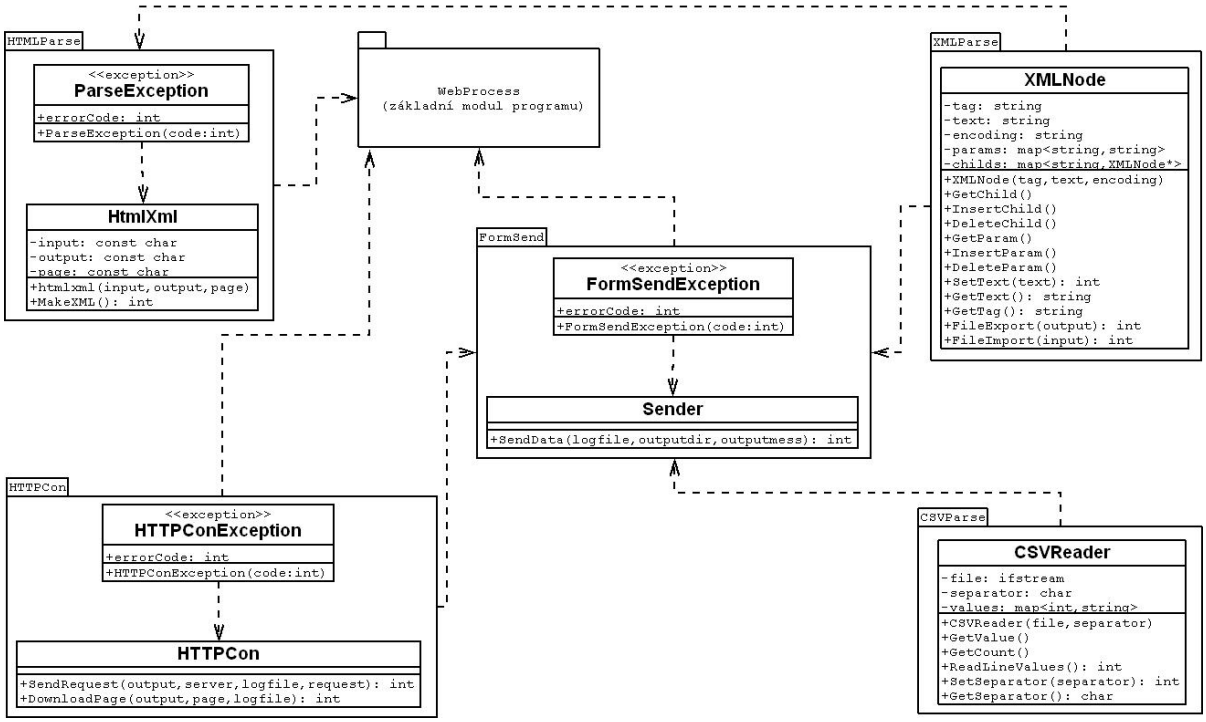
Aplikace jsou implementovány pomocí objektově orientovaného jazyka C++. Výhody objektového přístupu byly více či méně využity v jednotlivých modulech aplikace. Aplikace byla vyvíjena pod operačním systémem Windows XP, ale díky využití vývojového prostředí DevC++, které využívá překladač MinGW, je aplikace přenositelná jak pro systémy Microsoft Windows, tak pro systémy založené na unixovém jádře. Pro překlad jsou tedy využity standardní GNU C++ překladače za pomocí programu make. Součástí aplikace jsou dva vytvořené makefile soubory, jeden pro operační systémy Microsoft Windows a druhý pro Linux. Musí existovat dva odlišné makefile soubory, protože se program liší v jednom modulu pro určitý operační systém. Jedná se o modul s jednoduchým TCP klientem, neboť operační systémy mají odlišnou práci se sokety.

Nástroj pro automatické získání XML popisu z HTML dokumentu a nástroj pro vyplnění a odeslání zadaných hodnot do daného formuláře z příkazové řádky jsou součástí jedné aplikace, kde se pomocí argumentů příkazové řádky určuje, která funkce bude použita a s jakými parametry. Základní myšlenkou tedy je, že první nástroj stáhne HTML stránku na disk, dále pomocí stavového automatu najde formulář a jednotlivé elementy uvnitř formuláře a nakonec vytvoří XML popis definovaný v kapitole 3.2. Takto vytvořený XML soubor je potřeba uživatelsky editovat a doplnit potřebné hodnoty atributů pro mapování elementů na dané sloupce v CSV souboru. Doplněný XML popis již může být použit jako vstup pro druhý nástroj, který spolu s daty CSV souboru vytváří jednotlivé HTTP požadavky a pomocí nich odesílá data na webový server pro daný formulář.

Aplikace se tedy skládá z několika samostatných modulů, kde každý modul se stará o jiný typ činnosti. Jedná se o pět základních modulů. Modul pro připojení k HTTP serveru a odesílání požadavků (tento modul existuje ve dvou verzích pro operační systémy windows a linux), modul pro zpracování HTML souboru a extrakci formulářových dat, modul pro práci s XML souborem, modul pro práci se souborem CSV a modul pro automatické odesílání dat do vybraného formuláře. Vazby mezi jednotlivými moduly jsou probrány v kapitole 4.1. Podrobnější popis jednotlivých modulů poté v následujících kapitolách. Nad všemi těmito moduly potom stojí hlavní aplikace, která zajišťuje uživatelské rozhranní a spolupráci jednotlivých modulů. Z výše uvedeného vyplývá, že moduly jsou použitelné i pro jakoukoliv jinou aplikaci. Například k nim jde naprogramovat grafické uživatelské rozhranní pro určitý systém.

## 4.1 Návrh základních modulů

Na následujícím obrázku (obrázek 3) jsou znázorněny všechny použité moduly včetně jejich vzájemných vazeb.




Obrázek 3: Diagram modulů včetně vazeb

Základní modul programu „WebProcess“, který obsahuje zejména uživatelské rozhranní, využívá moduly „HTTPCon“ – pro stažení zvolené stránky z HTTP serveru, „HTMLParse“ – pro zpracování a extrakci formuláře ze stažené HTML stránky, a „FormSend“ – pro automatické odesílání dat do vybraného formuláře.

Modul „HTMLParse“, který se stará o parsování webové stránky a extrakci formuláře, využívá modul „XMLParse“, pomocí něhož vytváří dynamicky XML strom daného formuláře, který je potom možné exportovat do souboru.

Modul „XMLParse“ je samostatný a stará se pouze o zpracování XML souborů. Stejně tak je samostatný i modul „CSVParse“, který má na starost zpracování CSV souboru. Modul „HTTPCon“ zapouzdřuje funkce pro odesílání a příjem dat od HTTP serveru.

Poslední důležitý modul „FormSend“ obsahuje třídu pro analýzu formuláře popsaného pomocí XML souboru a odesílání dat do tohoto formuláře umístěného na webu. K tomu využívá modul

„XMLParse“, který zpřístupňuje XML soubor, modul „CSVParse“ pro přístup k datům uloženým v CSV souboru a modul „HTTPCon“ pro odesílání požadavků webovému serveru.

## 4.2 TCP klient

Jednoduchý TCP klient je implementován v modulu „HTTPCon“, který zabezpečuje komunikaci s webovým serverem. Jedná se o jednoduchou třídu, která obsahuje vlastnosti a metody potřebné pro úspěšné připojení k serveru na portu 80 a pro úspěšné odesílání dat (HTTP request) a přijímání odpovědí (HTTP response). Modul obsahuje také třídu pro odchytávání výjimek, které by mohly vzniknout během činnosti HTTP modulu.

Modul „HTTPCon“ neposkytuje rozhranní třídy pro ostatní aplikace, ale poskytuje pouze dvě základní funkce, které používají tuto třídu.První funkce slouží pro stažení požadované HTML stránky z webu, kde hlavním parametrem je URL adresa. Stažená stránka je poté uložena do definovaného souboru. Druhá funkce je pro odesílání HTTP požadavků na daný webový server. Veškeré odpovědi od webového serveru jsou také uloženy do souborů pro další případnou analýzu. Další možností obou funkcí je aktivace zaznamenávání veškeré komunikace do definovaných souborů.

Z důvodu odlišné práce se sokety v operačních systémech windows a unix, obsahuje aplikace dva velmi podobné moduly a to „HTTPCon“ a „HTTPConWin“. Jejich kompilace a přilinkování do programu se liší použitým Makefile souborem. Hlavičkový soubor je pro oba moduly totožný.

Ve všech hlavičkových souborech jsou také definovány jednotlivé chybové kódy, které je modul schopen vracet v určitých situacích, a proměnná obsahující popis chyby v případě jejího výskytu.

## 4.3 HTML Parser

Analýza HTML souboru se provádí pomocí modulu HTMLParse. Tento modul obsahuje třídu, která zpracovává HTML soubor, snaží se najít formulář, a pokud nějaký najde, tak dynamicky vytváří strom XML popisu tohoto formuláře. Princip extrakce tohoto formuláře s podrobným popisem stavového automatu je popsán v kapitole 4.3.1. Aby mohl tento modul vytvářet XML popis, je nutná spolupráce s modulem XMLParse, který nabízí rozhranní pro udržování a tvorbu XML popisu v paměti počítače. Modul obsahuje také třídu pro odchytávání výjimek, které by mohly vzniknout během parsování HTML stránky.

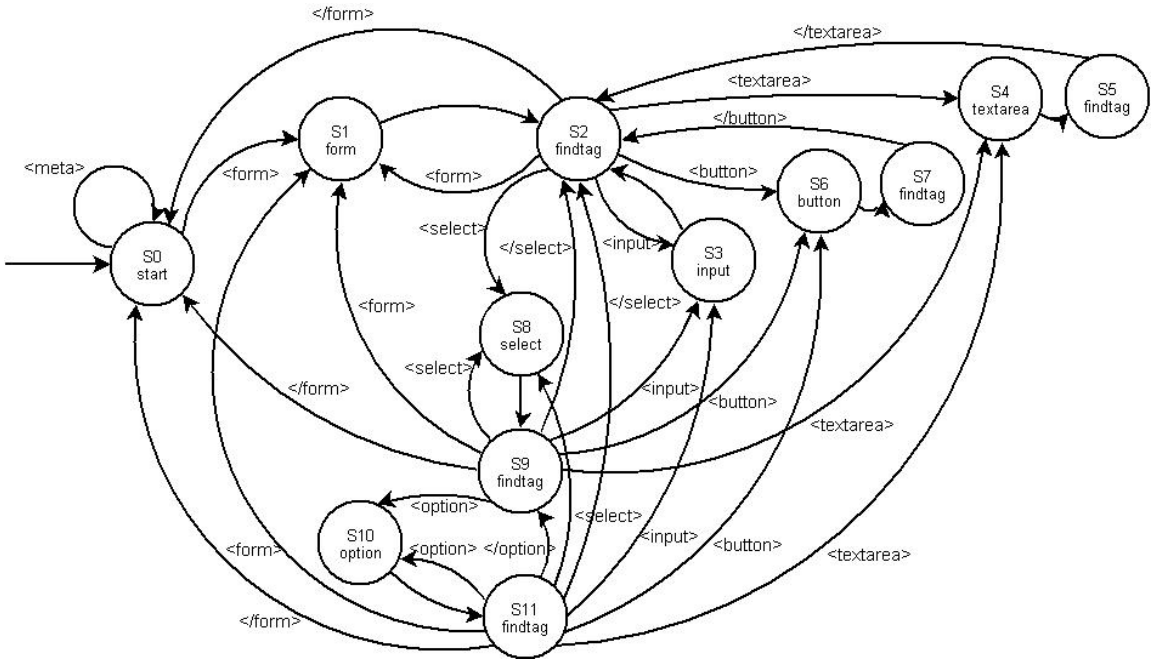
Modul „HTMLParse“ neposkytuje v hlavičkovém souboru rozhranní třídy pro ostatní aplikace, ale poskytuje pouze základní funkci, která používá tuto třídu. Jedná se o funkci „Parse“, která přebírá všechny potřebné parametry, kterými jsou: jméno a cesta k HTML souboru, definování výstupního XML souboru a URL adresa, ze které byla daná stránka stažena. Poslední parametr je potřebný z důvodu sestavení kompletní URL cesty v atributu „action“ elementu „form“, protože v dalším zpracování se bude používat již pouze XML soubor. V hlavičkovém souboru jsou také definovány jednotlivé chybové kódy, které je modul schopen vracet v určitých situacích, a proměnná obsahující popis chyby v případě jejího výskytu.

### 4.3.1 Získání formuláře

K analýze HTML souboru a zejména jeho formuláře se používá stavový automat, který nám zajistí správné načtení všech elementů, ze kterých se během načítání extrahuje pouze formulář s jeho elementy. Pokud se v HTML souboru nachází více formulářů, tak jsou postupně načteny všechny tyto formuláře s tím rozdílem, že je pro každý formulář vytvořen samostatný výstupní XML soubor. Například pokud je nastaven jako výstupní soubor pro XML popis „output.xml“, tak druhý nalezený formulář bude uložen v souboru „output.xml2“ atd. Tímto je uživateli dána možnost najít a modifikovat správný formulář, který pak použije k dalšímu zpracování.

Na obrázku 4 je znázorněn kompletní automat, kterým se zpracovává HTML soubor. Všechny stavy automatu jsou zároveň i koncovými stavy, protože průběh zpracování končí až po nalezení konce souboru. Celé zpracování používá funkci „Findtag“, která pokračuje v aktuálním stavu čtení vstupního souboru a vrací následující nejbližší element i s řetězcem, co všechno mu předcházelo. Tato funkce je podrobně popsána v kapitole 4.3.1.1.

Chování stavového automatu v určitých situacích bylo analyzováno jednak pomocí oficiální specifikace HTML, tak pomocí testování chování prohlížečů (Internet Explorer, Mozzila Firefox) v těchto situacích. Jedná se například o přítomnost formátovacích elementů mezi elementy „<button>” a „</button>“, specifické chování určitých elementů v případě neuzavření elementu jeho ukončujícím protějškem a jiné.




Obrázek 4: Stavový automat pro získání formuláře z HTML souboru

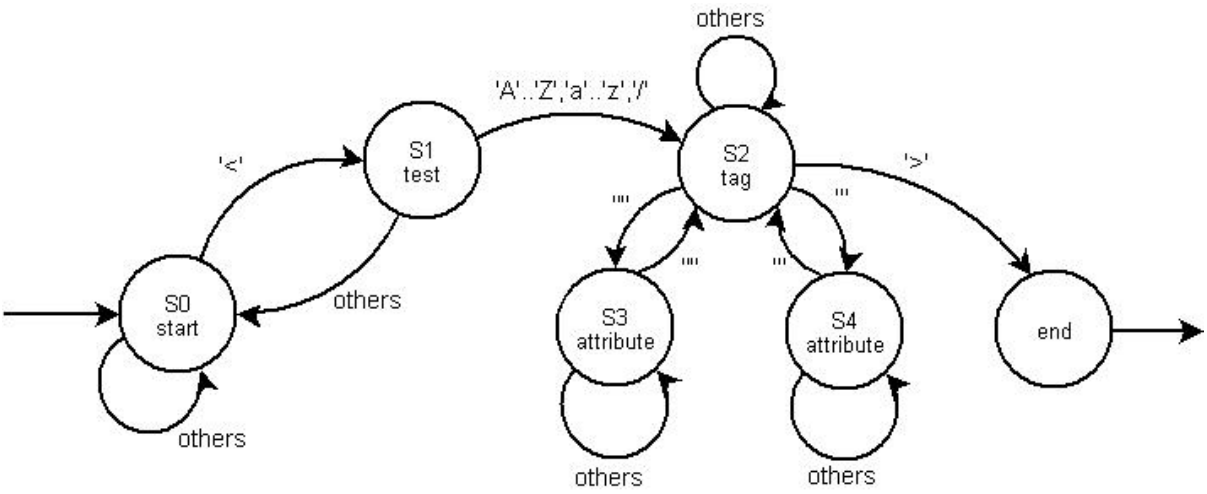
**Stručný popis jednotlivých stavů:**

- S0 – počáteční stav, přeskakujeme všechny elementy mimo elementu „meta“, ve kterém se snažíme najít kódování daného HTML souboru; jakmile narazíme na element „form“ přesuneme se do stavu S1
- S1 – tento stav je pro zpracování nalezeného elementu „form“; dojde také k inicializaci všech výstupních souborů a je vytvořen základ stromu XML popisu v paměti; po úspěšném zpracování se přesouváme do stavu S2
- S2 – v tomto stavu načteme další element a podle typu elementu budeme pokračovat do dalších stavů; jedná se o takový rozcestník elementů
- S3 – dojde ke zpracování elementu „input“; poté návrat do stavu S2
- S4 – zpracování počátečního elementu „textarea“; poté se přesuneme do stavu S5
- S5 – nalézáme se v elementu textarea; budeme čekat dokud nenajdeme ukončující element; vše co načteme tak si budeme pamatovat jako obsah elementu „textarea“; po ukončení se přesuneme do stavu S2
- S6 – zpracování elementu „button“;poté se přesuneme do stavu S7
- S7 – čekáme na ukončující element; vše co jsme zatím načetli, kromě elementů, si budeme pamatovat; po nalezení ukončujícího elementu se přesuneme do stavu S2
- S8 – zpracujeme element „select“; přesuneme se do stavu S9
- S9 – čekáme na vhodné elementy; stav funguje jako druhý rozcestník, neboť element „select“ může být přerušen jakýmkoliv jiným formulářovým elementem; nejvíce nás zajímá element „option“

- S10 – byl nalezen element „option“ uvnitř elementu „select“; zpracujeme tento element a přejdeme do stavu S11
- S11 – poslední stav, který funguje trochu jako rozcestník, protože element „option“ může být ukončen jakýmkoliv formulářovým elementem a tím končen i „select“, ve kterém se nalézá; nejvíce nás zajímá ukončující element pro element „option“ nebo „select“

#### 4.3.1.1 Hledání elementů

Funkce pro hledání následujícího elementu je stěžejní pro úspěšnou analýzu HTML souboru pomocí automatu popsaného v předcházející kapitole. Tato funkce opět vychází ze stavového automatu, pomocí kterého je schopna identifikovat správný element. Element se vyznačuje tím, že je zapsán mezi otevírací „<“ a uzavírací „>“ závorku. Platí pravidlo, že ihned po otevírací závorce následuje znak ze skupiny „A“ .. „Z“ nebo „a“ .. „z“, případně znak „/“, který označuje ukončující závorku. Pokud ihned za otevírací závorkou najdeme cokoliv jiného, tak se nejedná o element, nebo se nejedná o správně zapsaný element a nebudeme pokračovat v jeho zpracování. Budeme se tedy snažit najít další element, který bude vyhovovat daným pravidlům. Stavový automat také hlídá situaci v případě atributů, kdy hodnota atributu uzavřená v uvozovkách nebo apostrofech může také obsahovat znak „>”. Automat tedy končí po nalezení uzavírací závorky. Mimo samotného elementu automat vrací také řetězec s obsahem všech dat, která se načetla před samotným elementem. Toto je vhodné například v situaci, kdy se nalézáme uvnitř elementu „<textarea>“ (S4) a stavový automat pro zpracování HTML souboru čeká na ukončující element „</textarea>“ (S5). V případě nalezení elementu „</textarea>” potřebujeme také řetězec, který se nalézal před samotným ukončujícím elementem.



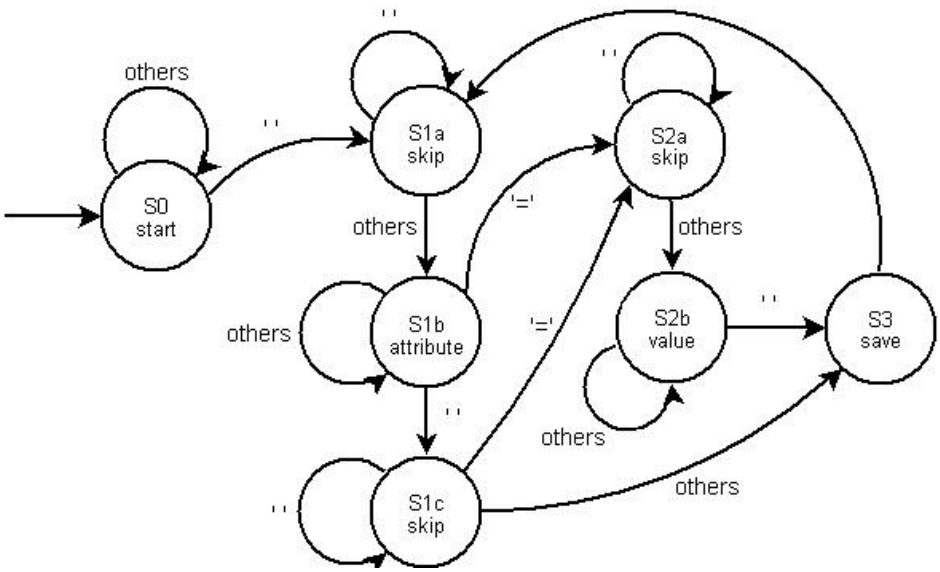

Obrázek 5: Stavový automat pro nalezení nejbližšího elementu

**Stručný popis jednotlivých stavů:**

- S0 – začátek hledání, budeme číst, dokud nenajdeme znak „<“
- S1 – testujeme, zda se jedná o tag; pokud další znak není „A“..„Z“, „a“..„z“ nebo „/“, tak se o tag nejedná a vrátíme se do S0, jinak S2
- S2 – budeme číst vše co je uvnitř tagu; dokud nenajdeme znak „>“, dáváme pozor na vnoření do uvozovek nebo apostrofů (hodnoty atributů)
- S3 – čteme hodnotu atributu v uvozovkách; čteme vše dokud uvozovky neskončí
- S4 – čteme hodnotu atributu v apostrofech; čteme vše dokud apostrofy neskončí

#### 4.3.1.2 Analýza atributů daného elementu

Pokud je nalezen požadovaný element a je potřeba ho zpracovat, potřebujeme nejdříve analyzovat všechny jeho atributy. Zpracováním se myslí vytvoření odpovídajícího prvku v XML popisu. Pro vytvoření tohoto prvku potřebujeme znát atributy, aby takto vytvořený prvek mohl být správně nastaven. K analýze atributů u požadovaného elementu nám slouží stavový automat znázorněný na obrázku 6. Jedná se o jednoduchý automat, který znak po znaku zpracovává řetězec s elementem a snaží se najít atributy a jejich odpovídající hodnoty. Koncový stav automatu je po vyčerpání celé délky řetězce. Koncovým stavem tedy může být každý stav automatu. Automat je vytvořen tak, aby byl schopen reagovat na větší počty mezer, než jenom jednu, na hodnotu elementu v uvozovkách či apostrofech, nebo bez nich. Výsledné atributy se ukládají do asociativního pole, kde se dají podle názvu atributu dobře vyhledávat a zpracovávat.




Obrázek 6: Stavový automat pro analýzu atributů daného elementu

**Stručný popis jednotlivých stavů:**

- S0 – začátek hledání; budeme číst, dokud nenajdeme mezeru; přeskočíme tím jméno elementu
- S1a – přeskočíme všechny mezery dokud nenajdeme jiný znak
- S1b – načteme jméno atributu; pokud najdeme mezeru, přesuneme se do stavu S1c nebo pokud najdeme znak „=“ tak se přesuneme do stavu S2a
- S1c – přeskočíme všechny mezery; pokud najdeme znak „=“, tak přejdeme do stavu S2a; pokud najdeme cokoliv jiného, tak jsme načetli atribut bez hodnoty a přeskočíme do stavu S3
- S2a – přeskočíme všechny mezery, které se nachází za znakem „=“
- S2b – načítáme hodnotu atributu dokud nenarazíme na mezeru
- S3 – uložíme načtený atribut do asociativního pole a pokračujeme ve čtení

## 4.4 XML Parser

Hlavním problémem pro oba implementované nástroje je potřeba pracovat s XML popisem. Na internetu existuje několik dobrých XML parserů, ale buď jsou příliš rozsáhlé a velké, nebo poskytují zbytečně mnoho funkcí, které pro tuto aplikaci nevyužijeme. To byl hlavní důvod, proč došlo k implementaci vlastní jednoduché knihovny, která bude obsahovat jen to nejpodstatnější pro práci s XML popisem formuláře. Vzhledem k tomu, že si XML soubor aplikace vytváří sama, není podstatné kontrolovat důsledně DTD a celou syntaxi XML souboru.

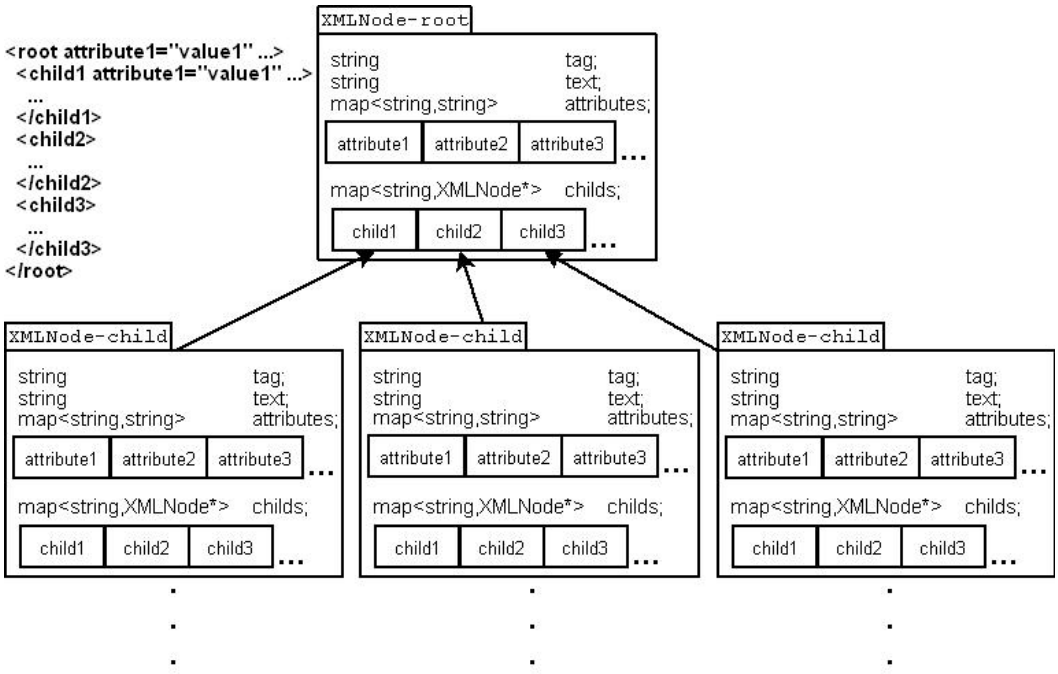
Vzhledem ke struktuře XML souborů, vyplývající z jednoho hlavního elementu, který obaluje celou XML strukturu, se jako nejvýhodnější řešení jeví stromová struktura. Pomocí této struktury lze vhodně uchovávat celý XML popis v paměti a procházet jednotlivé prvky stromu (elementy XML popisu). Knihovna tedy poskytuje rozhranní, pomocí kterého se dá udržovat a dynamicky měnit XML popis, který je pomocí stromové struktury uložen v paměti.

Modul „XMLParse“ obsahuje jednu základní třídu, která představuje jeden uzel (element) XML stromu. Pomocí objektů této třídy lze vhodně vytvářet kompletní strukturu XML popisu. V kapitole 4.4.1 je podrobně tato struktura popsána. Třída také obsahuje vlastnosti a metody potřebné pro účely obou aplikací.

### 4.4.1 Stromová struktura

Formulář se ukládá v paměti pomocí stromové struktury, která je vytvářena pomocí třídy „XMLNode“. Jeden objekt této třídy představuje jeden uzel, tedy jeden element formuláře. Tento

objekt obsahuje základní vlastnosti, kterými jsou název elementu (tag), text elementu (text), seznam atributů (attributes) a seznam potomků (childs). Atributy elementu jsou ukládány pomocí STL knihoven, konkrétně se jedná o asociativní pole (map). Toto pole nám dovoluje vyhledávat parametry podle jména. Seznam potomků využívá také asociativní pole, ve kterém je každý potomek určen svým jménem a jeho hodnota je ukazatel na další objekt typu „XMLNode“. Takováto struktura nám dává dobrou možnost dynamicky vytvářet celý XML popis formuláře. Graficky je tato struktura znázorněna na malém příkladu na obrázku 7.




Obrázek 7: Nastínění stromové struktury XML popisu

Mimo základních vlastností nám také třída „XMLNode“ nabízí spoustu metod, které nám ulehčují práci s jednotlivými objekty. Jedná se samozřejmě o konstruktor a destruktor (destruktor také zajistí řádné uvolnění z paměti), ale zejména o metody pro práci s polem potomků a atributů. Na jednotlivých metodách si můžeme všimnout polymorfismu, kdy pod jedním názvem existují různé implementace, které se liší počtem a typem předávaných parametrů. Mezi základní metody také patří export XML stromu do souboru, a to od daného uzlu níže a také import XML souboru. Pomocí metody import lze zpětně vytvořit XML strukturu v paměti ze vstupního souboru. Podrobnosti o vlastnostech a metodách třídy „XMLNode“ se nacházejí v příloze, případně v projektové dokumentaci.

### 4.4.2 Export XML

Export XML popisu z paměti do souboru je základní vlastností třídy „XMLNode“. Vzhledem k tomu, že metodu export obsahuje každý uzel stromu, tak je možné vyexportovat jenom tu část stromu,

kde se kořenovým uzlem stane uzel, ze kterého byla zavolána metoda export. Parametry jsou pouze dva, výstupní stream, do kterého se XML ukládá, a booleovská proměnná, která specifikuje, zda je potřeba nejdříve vložit hlavičku XML souboru. Hlavička se tedy vkládá pouze u prvního „kořenového“ uzlu.

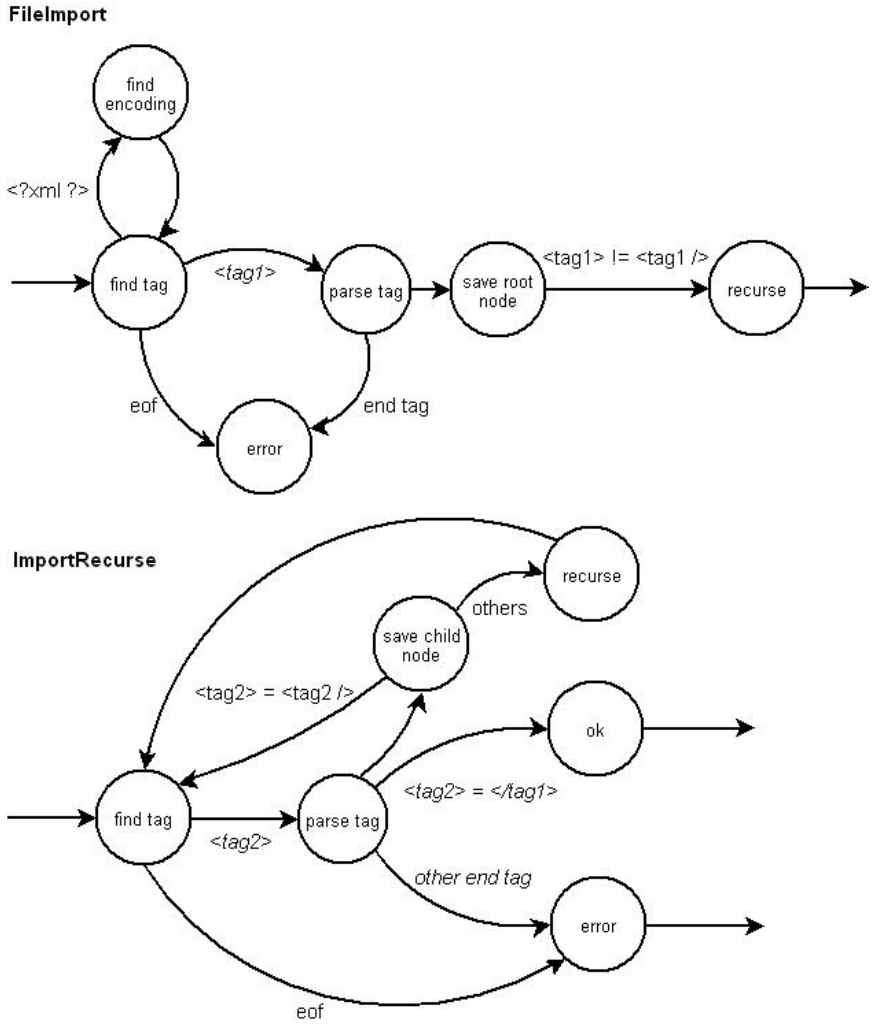
Algoritmus exportu je vcelku jednoduchý. Po zavolání metody export se do výstupního souboru uloží otevírací tag pro daný element s tím, že se do něj vloží veškeré atributy z pole attributes. Poté následuje postupné volání všech potomků a jejich metod export. Jakmile se vrátí řízení zpět do původní metody, tak se ještě do výstupního souboru zapíše ukončovací tag daného elementu. Lze konstatovat, že jde o jistou formu „rekurzivního“ volání.

### 4.4.3 Import XML

Pro potřeby odesílání dat je nutné modifikovaný XML soubor načíst do paměti. K tomu nám poslouží metoda „FileImport“, která má jako parametr vstupní soubor. Stačí nám tedy v paměti alokovat kořenový uzel, tedy první objekt třídy „XMLNode“ a poté zavolat metodu pro import souboru. Ostatní uzly se již v paměti alokují automaticky. Import už je na rozdíl od exportu trochu složitější záležitost. Pro analýzu obsahu XML souboru se využívají podobné automaty jako pro parsování HTML. Jedná se především o automaty pro nalezení nejbližšího elementu a nalezení všech atributů elementu. Na tyto automaty se podíváme podrobněji v kapitolách 4.4.3.1 a 4.4.3.2.

Obrázek 8 znázorňuje postup načtení XML souboru. Nejde přímo o stavový automat, spíše byla jeho grafická jednoduchost a přehlednost použita pro znázornění postupu importování XML struktury do paměti s vytvořením XML stromu pomocí třídy „XMLNode“. Celý postup importování se skládá ze dvou hlavních funkcí. První zajistí inicializaci a načtení kořenového elementu a druhá se již volá rekurzivně a postupným načítáním elementů se vytváří XML strom.

V první funkci „FileImport“ se tedy nejdříve zjistí kódování, pokud v XML souboru nějaké je, a poté se načte kořenový element. Je to nejvyšší element, který obaluje celou XML strukturu. Z nalezeného elementu se poté extrahují atributy a všechny podstatné vlastnosti se uloží do kořenového objektu třídy „XMLNode“. Pak se již zavolá rekurzivní funkce, které se předá ukazatel na vstupní soubor, ukazatel na aktuální uzel stromu a řetězec, který obsahuje ukončovací element právě vloženého elementu. Tím zajistíme, že rekurzivní funkce skončí úspěšně pouze tehdy, pokud najde element shodný s ukončovacím elementem, který byl aktuální při volání rekurze. Vše ostatní končí chybou.

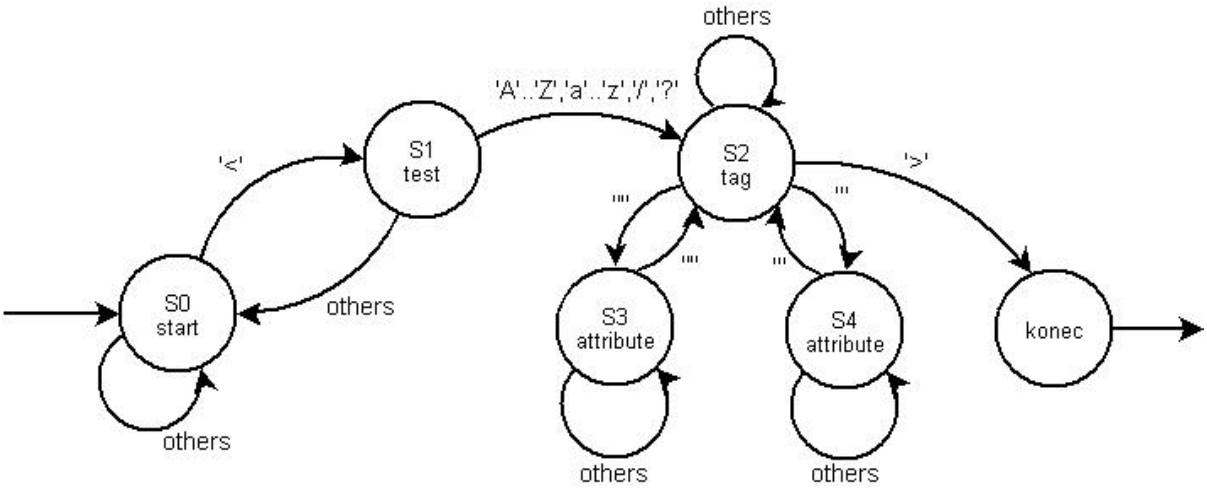



Obrázek 8: Postup načtení XML souboru

Rekurzivní funkce už tedy jenom najde další element, rozparsuje na atributy a pokud se nejedná o ukončující element (správný nebo chybný), tak se daný element uloží jako potomek k předanému aktuálnímu uzlu a zavolá se rekurzivně tato funkce. Jako aktuální uzel se jí ale předá právě vložený potomek. Rekurzivní funkce hledá elementy tak dlouho, dokud nenajde správný ukončující element, nebo není konec souboru.

#### 4.4.3.1 Hledání elementů

Stavový automat pro hledání elementů je téměř shodný s automatem pro hledání elementů v případě HTML parseru. Jediný a hlavní rozdíl je v tom, že jako počáteční znak tagu zde může být i „?“. Z důvodu vysoké podobnosti je zde uveden pouze obrázek automatu. Podrobnější popis lze nalézt v kapitole 4.3.1.1.

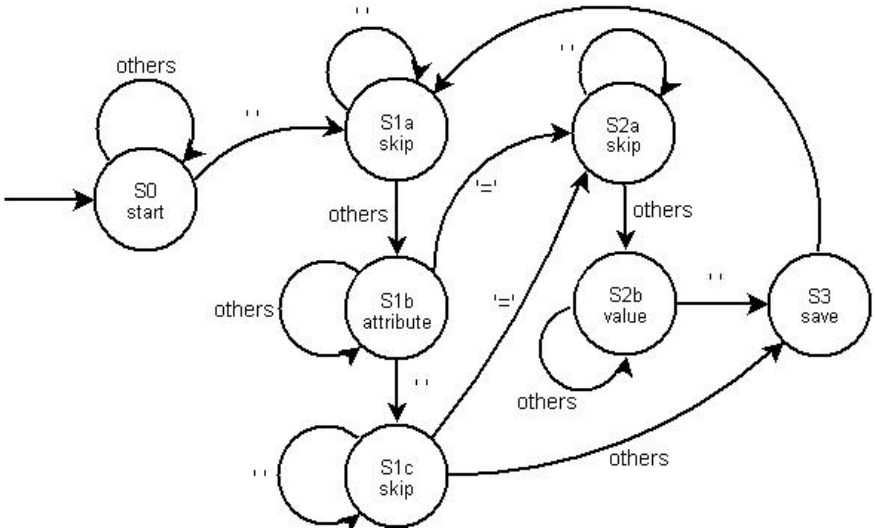



Obrázek 9: Stavový automat pro nalezení nejbližšího elementu

#### 4.4.3.2 Analýza atributů daného elementu

Pro analýzu načteného elementu se používá stavový automat zobrazený na obrázku 10. Jedná se o téměř totožný automat jako v případě HTML parseru. Tento automat nevrací pouze načtené atributy, ale vrací i název elementu (tag), který je potřeba pro uložení elementu do stromové struktury.

Načtené atributy se ukládají do asociativního pole, které se pak vrací společně s názvem elementu. Vzhledem k použití stejného typu asociativního pole, jako je použit v objektu pro uzel XML stromu, není potřeba žádná další analýza atributů. Ty se jednoduše uloží k danému uzlu XML popisu.




Obrázek 10: Stavový automat pro analýzu atributů elementu

## 4.5 CSV Parser

Pro potřeby čtení dat z CSV souboru vznikla knihovna „CSVParse“, která nám práci s CSV souborem zapouzdřuje. Tato knihovna obsahuje třídu „CSVReader“, která zajišťuje korektní čtení hodnot z CSV souboru. Celé rozhranní třídy je poskytováno v hlavičkovém souboru.

### 4.5.1 CSV (Comma-Separated Values)

Comma-Separated Values, neboli hodnoty oddělené čárkami, je jednoduchý formát pro výměnu tabulkových dat. Soubor CSV je složený z řádků, ve kterých jsou jednotlivé položky odděleny čárkou a jednotlivé řádky jsou odděleny znaky CR a LF. Je důležité upozornit zejména na to, že česká lokalizace Microsoft Excel používá jako oddělovač středník místo čárky. Je s podivem, že i přesto název CSV zůstal stejný.

Pokud hodnota uložená v CSV souboru obsahuje znaky použité pro formátování, jakými jsou čárka a znaky CR a LF, je celá hodnota uzavřena mezi uvozovky. Pokud ovšem hodnota obsahuje uvozovku, je potřeba escapovat tuto uvozovku ještě jednou uvozovkou před ní. Dochází ke zdvojování uvozovek. Následující pravidla si ukážeme na příkladu. Příklad obsahuje smyšlená data o třech knižních vydání. Pod tabulkou číslo 1 je ekvivalentní zápis v CSV souboru.

| | | | |
|---|---|---|---|
| 2000 | PHP 4 | Jiří Bráza | 356 |
| 2004 | Javascript | Aleš Novák, Petr Zapletal | |
| 2001 | C++ „učebnice základů jazyka” | Jan Nový<br>Oldřich Navrátil | 180 |

Tabulka 1: Příklad pro CSV soubor

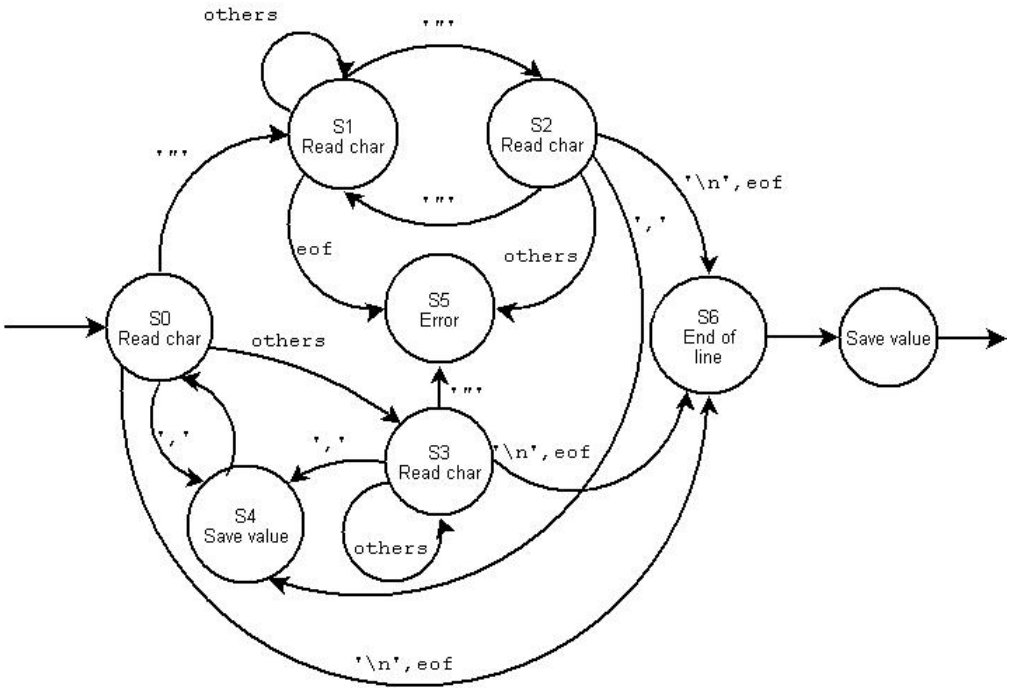
```
2000,PHP 4,Jiří Bráza,356
2004,Javascript,”Aleš Novák, Petr Zapletal”,,
2001,”C++ ””učebnice základů jazyka”””,”Jan Nový
Oldřich Navrátil”,180
```

### 4.5.2 Čtení CSV souboru

Modul „CSVParse“ se nám tedy postará o korektní čtení hodnot z CSV souboru. Modul obsahuje ve svém hlavičkovém souboru třídu „CSVReader“, která zapouzdřuje operace spojené se čtením CSV souboru. Hlavičkový soubor obsahuje také chybové kódy a proměnnou, ve které lze nalézt v případě chyby její podrobnější popis.

Třída obsahuje pár základních vlastností a metod. Vlastnosti jsou přístupné pouze pomocí metod. Jedná se o proměnnou se souborem, asociativní pole s hodnotami jednoho řádku a proměnná se znakem, který je použit jako oddělovač. Tato vlastnost vznikla zejména kvůli MS Excel, který jako oddělovač používá středník. Dává to možnost definovat si vlastní libovolný oddělovač hodnot. Podrobný popis všech vlastností a metod se v projektové dokumentaci.

Čtení hodnot ve třídě probíhá pomocí metody „ReadLineValues“, která načte hodnoty z jednoho řádku (tedy jednoho záznamu tabulky) a tyto hodnoty uloží do asociativního pole. Pomocí dalších metod je možno toto pole procházet nebo přímo přistupovat k určitým prvkům podle indexu. Na obrázku 11 je nakreslen stavový automat, který se stará o správné načtení všech hodnot na jednom řádku do pole. Ve stavovém automatu je jako oddělovač použita čárka, v programu je oddělovač samozřejmě volitelný parametr.




Obrázek 11: Stavový automat pro čtení CSV souboru

**Stručný popis jednotlivých stavů:**

- S0 – načteme znak; pokud se jedná o uvozovky, tak se přesuneme do stavu S1; pokud najdeme oddělovač, tak se přesuneme do stavu S4; pokud najdeme konec řádku nebo souboru, tak se přesuneme do stavu S6, jinak se přesuneme do stavu S3
- S1 – hodnota je v uvozovkách; čteme vše dokud nenarazíme na uvozovky; konec souboru znamená chybu, protože není ukončená hodnota
- S2 – testujeme zda nepřijde druhá uvozovka; pokud ano, tak se jednalo o zdvojené uvozovky a vrátíme se do stavu S1; pokud najdeme konec řádku nebo souboru, tak

ukončíme čtení řádku a uložíme poslední načtenou hodnotu; pokud načteme oddělovač, tak přejdeme do stavu S4; jakýkoliv jiný znak je chyba syntaxe CSV souboru

- S3 – čteme hodnotu v klasickém režimu (bez uvozovek); pokud najdeme čárku tak se přesuneme do stavu S4; pokud najdeme konec řádku nebo souboru, tak se přesuneme do stavu S6 a uložíme poslední načtenou hodnotu; v případě načtení uvozovek přejdeme do chybného stavu (uvozovky můžou být jen v hodnotě uzavřené do uvozovek)
- S4 – tento stav ukládá hodnotu do asociativního pole a poté se přesune do stavu S0 pro načtení další hodnoty
- S5 – chybový stav; končíme čtení hodnot s chybou syntaxe
- S6 – konec řádku nebo souboru; uložíme do pole poslední načtenou hodnotu a končíme čtení

## 4.6 Získání XML popisu z HTML

Jako první věc se musí nejdříve stáhnout požadovaný HTML dokument z webového serveru pomocí modulu „HTTPCon“. Poté může následovat jeho analýza a tvorba XML popisu. Získání XML popisu formuláře z HTML dokumentu je jednoprůchodový proces, kdy se během analýzy HTML dokumentu zpracovávají postupně všechny elementy formuláře a ihned se vytváří strom s XML popisem. K analýze HTML dokumentu se používá výše popsaný modul „HTMLParse“, který využívá modul „XMLParse“. Základní algoritmy z obou těchto modulů jsou popsány v předcházejících kapitolách. V této kapitole je popsáno pár pravidel pro převod HMTL elementů na odpovídající XML elementy. Výsledné XML elementy vycházejí z navržené XML struktury v kapitole 3.

Element „form“ se tedy jednoduše mapuje na odpovídající element v XML. Jednoduše se pouze najdou potřebné atributy a ty se uloží do výsledného elementu. Za zmínku stojí snad jenom funkce analyzující atribut „action“, který nemusí obsahovat kompletní URL odkaz na zpracovávající skript. V tomto případě funkce, dle pravidel relativních odkazů a pomocí znalosti kompletního URL stránky se zdrojovým formulářem, sestaví kompletní URL odkaz na zpracovávající skript. Tato maličkost nám později umožní použití XML struktury jako plnohodnotného popisu pro odeslání dat.

Element „input“ typu „radio“ je jeden z problematičtějších elementů, protože v XML popisu vytváří prvek „selection“. Je to dáno zejména jeho vlastností, kdy stejně pojmenované prvky „radio“ vytvářejí skupinu. Můžeme je tedy považovat za jakýsi výběrový seznam. Z toho důvodu musíme nejdříve v XML stromu zkusit najít selection, který se jmenuje stejně jako zpracovávaný prvek. Pokud takový prvek existuje, tak k němu pouze přidáme nového potomka, element „itemsel“, a pokud

neexistuje, tak musíme vytvořit nový prvek „selection“ pro případ, že by se ve formuláři nalézal další takový prvek ve skupině. Pokud vytváříme prvek „selection“, tak nesmíme zapomenou vložit nový atribut „column“, který nám zajistí mapování elementu na sloupec v CSV souboru. U jednotlivých prvků „itemsel“ zase přibude atribut „csvvalue“, pomocí kterého může uživatel vyplnit hodnotu odpovídající hodnotě v CSV souboru.

Dalším zpracovávaným elementem je opět „input“ ale tentokrát typu „checkbox“. Pro tento prvek se vytváří element „itemsel“, který není součástí „selection“ ale stojí samostatně. Převádí se do něj všechny atributy popsané pomocí DTD a samozřejmě se musí doplnit atributy „column“ a „csvvalue“ pro mapování do CSV souboru.

Posledním prvkem z elementu „input“ je prvek typu „text“. Tento prvek se pouze analogicky vytvoří v XML popisu jako element „element“ se všemi potřebnými atributy. K tomuto elementu se přidává pouze atribut „column“, který zajistí mapování na sloupec v CSV souboru. Z CSV souboru se obsah daného sloupce celý vkládá jako hodnota elementu a proto není potřeba druhý atribut „csvvalue“. V případě, že je element typu „file“,tak by měla CSV hodnota obsahovat cestu k souboru pro odeslání.

Element „textarea“ se v XML popisu vytváří pod stejným jménem s tím, že se použijí jen potřebné atributy. Důležité je uložit celý obsah tohoto elementu, pokud tam nějaký byl, jako CDATA. K elementu přidáme pouze prvek „column“ ze stejného důvodu jako u elementu „input“ typu „text“.

Další zpracovávaný element „button“ již obsahuje jednoduché pravidlo pro zpracování. Jedná se o jeho typ, kdy se zpracovávají elementy pouze typu „submit“. Ostatní nemají pro funkci formuláře žádný podstatný smysl. Jinak se opět zpracovávají jen podstatné atributy definované navrženým XML popisem, obsah prvku se podobně, jako v případě elementu „textarea“, uzavírá do sekce CDATA a vzhledem k faktu, že jde pouze o tlačítko, tak se přidává navíc pouze atribut „column“.

Element „select“ pouze iniciuje vytvoření podobného elementu v XML popisu s názvem „selection“. Přidávají se jen podstatné atributy a navíc atribut „column“ pro mapování na CSV soubor. Další obsah elementu „selection“ závisí na dalším zpracování HTML dokumentu.

Posledním zpracovávaným elementem je „option“. Tento element je součástí elementu „select“ a tudíž se v XML popisu přidává k elementu „selection“. Musíme se nejprve dostat na poslední vytvořený prvek „selection“ a v něm vytvořit nového potomka. Element „option“ má stejně jako prvek „input“ typu „radio“ odpovídající element s názvem „itemsel“. Do vytvořeného elementu se přidávají všechny potřebné atributy společně s novým atributem „column“. Pokud je tento element

součástí „selection“ s platným atributem „multiple“, musíme přidat ještě atribut „csvvalue“, který nám pomůže při mapování odpovídající hodnoty v CSV souboru.

Atributy „column“ a „csvvalue“ musí uživatel vyplnit dle pravidel popsaných v kapitole 4.8.

## 4.7 Odeslání hodnot do formuláře

Pro odesílání dat do formuláře slouží modul „FormSend“. Tento modul obsahuje třídu pro odchytávání výjimek, které by mohly vzniknout během zpracování, a dále třídu „Sender“, která zapouzdřuje veškeré funkce potřebné pro sestavování správných HTTP požadavků. Tyto sestavené požadavky jsou poté pomocí modulu „HTTPCon“ odeslány na webový server. Dále modul spolupracuje s modulem „XMLParse“, pro přístup k XML popisu formuláře, a s modulem „CSVParse“, pro zpřístupnění hodnot CSV souboru.

Samotné rozhranní tříd není hlavičkovým souborem poskytováno. Je zde pouze funkce, která dané třídy využívá. Třída „Sender“ byla použita zejména kvůli zapouzdření a lepší přehlednosti kódu. Dále jsou v hlavičkovém souboru chybové kódy pro různé situace a proměnná, která v případě chyby obsahuje její podrobnější popis. Detailnější informace o jednotlivých vlastnostech a metodách obou tříd je možné najit v projektové dokumentaci.

Před samotným odesíláním dat dochází nejprve k inicializace XML popisu, kdy je vstupní XML soubor importován pomocí třídy „XMLNode“ do paměti. Dále se pomocí třídy „CSVReader“ inicializuje CSV soubor, který by měl obsahovat veškerá data, která potřebujeme odeslat. Poté již následuje cyklus odesílání dat. Jedním cyklem je myšleno načtení jednoho řádku v CSV souboru, neboť tento řádek by měl obsahovat veškerá data pro vyplnění formuláře. Pokud máme načten řádek, dochází k průchodu XML stromem a sestavováním HTTP požadavku podle typu elementu a jeho hodnoty.

Kořenový element XML popisu obsahuje všechny důležité atributy k tomu, abychom věděli na jaké URL bude směřovat HTTP požadavek, jestli budeme sestavovat požadavek typu GET nebo POST a v případě typu POST, jestli budeme kódovat přenášená data pomocí metody „application/x-www-form-urlencoded“ nebo pomocí „multipart/form-data“. Princip obou typů kódování je popsán v kapitole 2.2. Dále se analyzují jednotlivé elementy a jejich atributy individuálně dle typu elementu.

**Pravidla pro analýzu elementu „element“:**

- Pokud je element typu „hidden“, „readonly“ nebo není zadána hodnota atributu „column“ , tak se odesílá jeho výchozí hodnota

- Pokud je element typu „text“ a je nastaven atribut „maxlength“, tak se délka odesílaného řetězce zkracuje na velikost danou tímto atributem
- Pokud je element typu „submit“ a je nastavena jakákoliv hodnota u atributu „column“, tak se element odesílá jako stisknuté tlačítko

**Pravidla pro analýzu elementu „textarea“:**

- Pokud má element nastaven atribut „readonly“ nebo není zadaná žádná hodnota v atributu „column“, tak se odesílá jeho výchozí hodnota

**Pravidla pro analýzu elementu „selection“:**

- Pokud má element nastaven atribut „readonly“ nebo není zadána hodnota u atributu „column“, tak se odesílá hodnota elementu „itemsel“, který má nastaven atribut „selected“
- Pokud je nastaven atribut „multiple“, tak se odesílají všechny hodnoty, které vyhovují stanoveným podmínkám a hodnota atributu „column“ se načítá u každého potomka typu „itemsel“ zvlášť (pokud není nastaven tak používáme globální „column“ u „selection“)
- Pokud není nastaven u potomka typu „itemsel“ atribut „csvvalue“, tak se s hodnotou v CSV souboru porovnává obsah tagu „description“

**Pravidla pro analýzu elementu „itemsel“:**

- Pokud má element nastaven atribut „readonly“ nebo není zadána hodnota u atributu „column“, tak se odesílá hodnota elementu „itemsel“, pokud má nastaven atribut „selected“
- Pokud není nastaven atribut „csvvalue“, tak se s hodnotou v CSV souboru porovnává obsah tagu „value“

Podrobněji jsou atributy „column“ a „csvvalue“ vysvětleny v následující kapitole 4.8.

## 4.8 Pravidla pro mapování XML do CSV

Mapování elementů XML popisu na hodnoty v odpovídajícím sloupci CSV souboru je velice důležitou součástí procesu odesílání dat. Bohužel tato část potřebuje zásah uživatele, který potřebné mapovací údaje musí zanést do XML popisu formuláře. Je proto potřeba určitá znalost struktury XML souboru a jeho syntaxe. Dále je také potřeba dodržet pár zásadních pravidel, které se týkají samotného doplnění hodnot. Tato pravidla jsou podrobněji popsána v této kapitole.

Pro potřeby mapování se k elementům v XML popisu přidávají atributy „column“ a „csvvalue“. Atribut „column“ se nachází u každého elementu, který je potřeba nějakým způsobem vyplňovat nebo identifikovat pomocí dat z CSV souboru. Do tohoto atributu tedy zadáme číslo, které specifikuje pořadí sloupce, který bude pro vyplnění použit. Sloupce v CSV souboru jsou číslovány od 1 postupně s přírůstkem 1. Pokud zadáme neplatnou hodnotu sloupce, nedojde k jeho načtení a bude do formuláře vyplňována prázdná hodnota. Pokud necháme atribut prázdný, tedy nebude obsahovat žádnou hodnotu, bude program považovat element, jako kdyby byl pouze pro čtení a bude s každým záznamem odesílat jeho implicitní hodnoty.

Atribut „csvvalue“ můžeme najít pouze u elementů „itemsel“ a to jak těch, co jsou samostatné, tak i těch, které jsou součástí elementu „selection“. Tento atribut vzniknul zejména kvůli nutnosti správně identifikovat, zda má být prvek odeslán, případně, která položka z výběrového elementu „selection“ byla vybrána. Hodnota tohoto prvku tedy řeší nejednoznačnost dat mezi CSV souborem a XML popisem. Hodnota tohoto atributu je obecný text, který se porovnává s hodnotou daného sloupce v CSV souboru a pokud se tyto hodnoty rovnají, tak se bere prvek jako vybraný a bude odeslán spolu s ostatními daty formuláře. Pokud je atribut „csvvalue“ prázdný, tak se v případě samostatně stojícího elementu „itemsel“ porovnává s CSV souborem jeho hodnota „value“ a v případě elementu „selection“ se pro porovnávání aplikuje hodnota „description.

Trochu problém je v tomto případě element „selection“ s atributem „multiple“, protože nám tento způsob nedává možnost vybrat více elementů. V tomto případě je tedy atribut „column“ přidán do každého elementu „itemsel“ nacházejícího se v elementu „selection“. To dává možnost mapovat každou hodnotu ve výběru na jeden sloupec CSV souboru a tím vybrat více hodnot k odeslání.

Tato pravidla jsou názorně ukázána na příkladech v manuálu k programu, který je součástí této dokumentace jako příloha 2.

## 4.9 Použití modulů v jiné aplikace

Díky návrhu jednotlivých funkcí jako samostatných modulů je možné tyto moduly použít pro jakoukoliv jinou aplikaci. Tím je myšlen zejména případ, že by někdo potřeboval udělat podobný program s jiným uživatelským rozhranním (ať už grafické nebo textové). Samozřejmě, že modul pro tvorbu XML stromu, pro připojení k HTTP serveru a modul pro čtení CSV dat jsou naprosto samostatné a tím využitelné i pro zcela jiné aplikace.

# 5 Ovládání programu

Program pracuje v textovém režimu a vše se nastavuje pomocí příkazové řádky. Konzolové aplikace bylo využito jednak z důvodu, že je to součást zadání, ale také z důvodu nezávislosti textového režimu na operačním systému a z toho vyplívající jeho přenositelnost mezi různými operačními systémy. Veškeré volby vstupních a výstupních souborů, zapínání logování komunikace s webovým serverem a volba funkce aplikace jako takové, se řeší pomocí parametrů na příkazové řádce. Jednotlivé parametry, jejich funkce a význam jsou vysvětleny v kapitole 5.1. Aplikace také umožňuje specifikovat výchozí nastavení, díky kterému nám odpadne nutnost zadávat stále ty samé parametry na příkazové řádce. Možnosti takového nastavení jsou popsány v kapitole 5.2.

## 5.1 Parametry příkazové řádky

Pomocí parametrů se zapínají jednotlivé funkce aplikace a nastavují jiné vstupní a výstupní soubory, než jaké jsou definovány jako výchozí. Syntaxe parametrů na příkazovém řádku je následující:

```
webprocess [[-g url]|[-s]] [-h] [-d char] [-o file] [-x file] [-l file] [-c file]
```

Charakter jednotlivých přepínačů se může lišit podle funkce aplikace. Ta může provádět dvě základní funkce. První funkcí je stáhnutí webové stránky a vytvoření XML popisu formuláře a druhá je automatické odesílání dat do formuláře.

1. **Stáhnutí webové stránky a vytvoření XML popisu**
   - `-g url` … tento parametr určuje, že se bude stahovat webová stránky a vytvářet XML popis, „url“ určuje adresu webové stránky ke stažení (povinný)
   - `-o file` … „file“ obsahuje novou cestu k souboru, do kterého se bude ukládat stažená webová stránka (nepovinný)
   - `-x file` … „file“ obsahuje novou cestu k souboru, do kterého se uloží vytvořený XML popis formuláře (nepovinný)
   - `-l file` … přepínač „-l“ zapne logování připojení k webovému serveru do souboru, parametrem „file“ můžeme specifikovat nový soubor pro ukládání záznamů
   - `-h` … vypíše nápovědu
2. **Automatické odesílání dat do formuláře**
   - `-s` … tento parametr určuje, že se budou automaticky odesílat data do formuláře (povinný)

- `-c file` … „file“ obsahuje cestu k CSV souboru, který obsahuje data pro odeslání (nepovinný)
- `-x file` … „file“ obsahuje cestu k XML souboru, který obsahuje XML popis formuláře (nepovinný)
- `-o directory` … „direcotry“ obsahuje cestu k adresáři, do kterého se budou ukládat odpovědi z webového serveru na jednotlivá odesílaná data (nepovinný)
- `-l file` … přepínač „-l“ zapne logování připojení k webovému serveru do souboru, parametrem „file“ můžeme specifikovat nový soubor pro ukládání záznamů
- `-d char` … „char“ obsahuje znak, který bude použit jako oddělovač sloupců v CSV souboru
- `-h` … vypíše nápovědu

## 5.2 Výchozí nastavení

Výchozí nastavení aplikace je výhodné zejména v případě, kdy často používáme stejné cesty k souborům a adresářům a je pro nás nepohodlné je stále znovu zapisovat pomocí parametrů na příkazové řádce. Toto nastavení můžeme upravovat v souboru „setting.ini“, který se nalézá ve stejné složce jako samotný program. Nejedná se o plnohodnotný „ini“ soubor, ale pro takto jednoduché nastavení plně postačuje. Jako výchozí nastavení můžeme definovat následující parametry.

```
OutputPage=./=html=/download.html
XmlFile=./=xml=/output.xml
CsvFile=./=csv=/file.csv
CsvDelimiter=;
OutputDirectory=./=html=/
GetPageLog=./=log=/HTTPDownload.log
SendDataLog=./=log=/HTTPSend.log
```

Obrázek 12: Ukázka souboru setting.ini

Význam je následující:

- OutputPage … soubor, do kterého se ukládá stažená webová stránka
- XmlFile … soubor, do kterého se ukládá, nebo ze kterého se čte XML popis formuláře
- CsvFile … soubor, ve kterém jsou uložena data pro odesílání
- CsvDelimiter … znak, který bude použit jako oddělovač v CSV souboru
- OutputDirectory … adresář pro ukládání výstupních souborů, které jsou odpověďmi od webového serveru na posílaná data

- GetPageLog … soubor, do kterého se bude ukládat záznam komunikace s webovým serverem při stahování webové stránky
- SendPageLog … soubor, do kterého se bude ukládat záznam komunikace s webovým serverem při odesílání dat

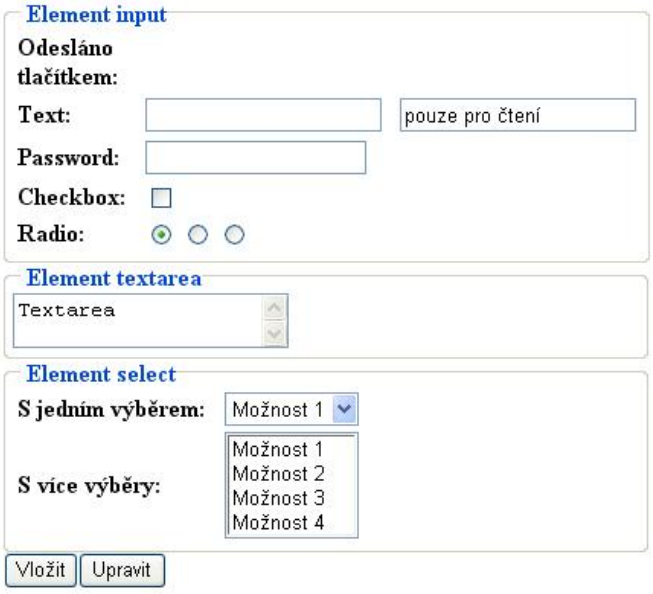
# 6 Testování aplikace

Aplikace byla testována pomocí formulářů vytvořených v jazyce PHP především pro tyto účely, ale byla také testována na existujících webových serverech na internetu. Testovací formuláře běží pod webovým serverem Apache s podporou jazyka PHP.

## 6.1 Testování stažení stránky a vytvoření XML

Pomocí aplikace byly stahovány různé webové stránky, ať už obsahovaly formulář nebo ne. Například úvodní webová stránka serveru www.seznam.cz obsahuje celkem 8 různých formulářů. Pro všechny tyto formuláře byly vytvořeny XML popisy v samostatných souborech. Také byl vytvořen jeden speciální, i když syntakticky zcela nesprávný, soubor, který má za účel otestovat celý proces parsování HTML. Testována byla tedy benevolentnost k nepřesné syntaxy HTML souborů a chování v extrémních situací. Chování HTML parseru bylo porovnáváno s chováním prohlížečů Microsoft Explorer a Mozzila Firefox. Na obrázku 13 je vidět hlavní testovací formulář, který obsahuje většinu formulářových elementů. V kapitole 6.2 je testována schopnost odesílat data do tohoto formuláře.

Testovací formulář je vyhotoven ve třech variantách. Všechny varianty vypadají stejně, liší se pouze ve způsobu odesílání dat. Jeden tedy používá metodu GET, druhý metodu POST a třetí také metodu POST, ale pomocí kódování „multipart/form-data“. Psolední jmenovaný navíc obsahuje element pro odesílání souborů. Formulář je naprogramován tak, že odeslaná data vyplní do formuláře, takže je hned vidět co všechno se vyplnilo. Na přiloženém datovém nosiči jsou k nahlédnutí stažené formuláře a vytvořené XML soubory.




Obrázek 13. Testovací formulář

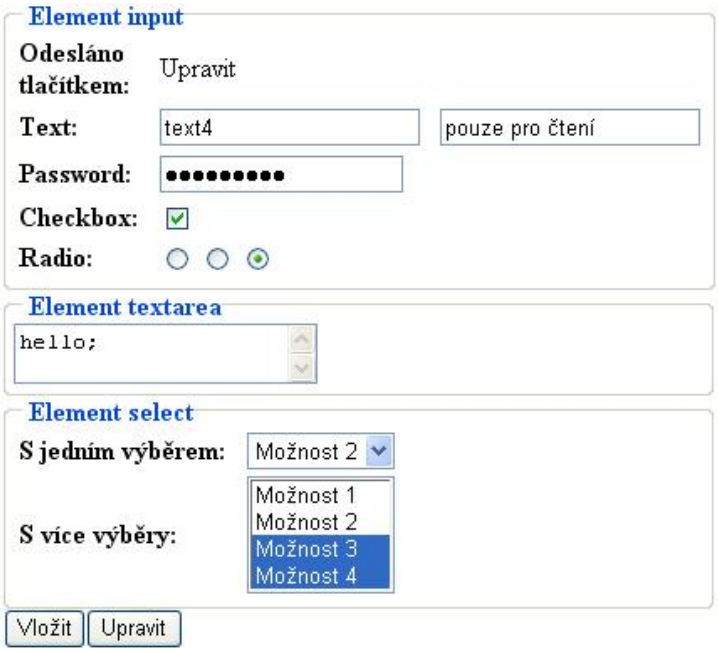
## 6.2 Testování odesílání dat

Pro testování odesílání dat byly použity XML popisy vytvořené z testovacích formulářů popsaných v kapitole 6.1. Dále byl vytvořen CSV soubor, který obsahuje testovací data použitá pro odesílání. Ve spolupráci se strukturou CSV souboru byla doplněna všechna mapovací data do XML popisu. Postup a pravidla pro mapování jsou popsána v manuálu k programu, který je součástí přílohy.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| text1 | Password1 | 1 | Radio1 | content of element textarea | opt2 | 1 | 0 | 0 | 1 | Test.jpg |
| text2 | Password2 | 0 | Radio3 | | opt1 | 0 | 1 | 0 | 0 | |
| text3 | Password3 | 0 | Radio2 | something<br>next line | opt3 | 0 | 1 | 0 | 1 | |
| text4 | Password4 | 1 | Radio3 | hello; | opt2 | 0 | 0 | 1 | 1 | Test.zip |

Tabulka 2: Testovací CSV soubor

Program je tedy schopen odesílat data do formuláře pomocí všech tří známých metod (GET a POST s oběma typy kódování). Výsledky vyplněných formulářů jsou k nahlédnutí na přiloženém datovém nosiči. Jsou tam ve členěné adresářové struktuře všechny vstupy a výstupy uplatněné při testování. Na obrázku 14 je znázorněn příklad jednoho formuláře, který vrátil webový server v podobě HTML stránky po obdržení požadavku s odesílanými daty.




Obrázek 14. Vyplněný testovací formulář

# Závěr

Cílem této práce bylo popsat teoretický základ pro jednotlivé technologie, které byly dále využity pro implementaci programu umožňujícího automatizovat webové formuláře, a samozřejmě popsat způsob implementace tohoto programu.

Byly popsány nejpoužívanější značkovací jazyky HTML a XHTML a zejména prostředky pro tvorbu formulářů. Dále byl uveden stručný úvod do HTTP protokolu, jež je nezbytnou součástí internetu jako takového, a především byly prezentovány způsoby, jakými se mohou pomocí HTTP protokolu přenášet formulářová data. Jedná se především o metody GET a POST a způsob kódování jednotlivých informací z formuláře. V poslední části teoretického úvodu byl nastíněn nový značkovací jazyk XML, který se v poslední době dostává hodně do popředí, zejména kvůli jeho univerzálnosti. Důležitou částí této kapitoly je návrh XML struktury, která popisuje vybraný formulář ve zjednodušené formě, příhodnější pro programové zpracování. Součástí návrhu XML struktury je i DTD, které specifikuje povolenou syntaxi XML dokumentu. Toto DTD je uvedeno v příloze zprávy.

V druhé části diplomové práce byla popsána implementace nástroje, který se v podstatě skládá ze dvou hlavních aplikací. První aplikace umožňuje automaticky generovat navrženou XML strukturu z webového formuláře, který definujeme jako vstup této aplikace. Vygenerovaný XML popis slouží pro automatické vyplňování formuláře a odesílání dat pomocí protokolu HTTP. Druhá aplikace dostává na vstup jednak XML strukturu vygenerovanou pomocí první aplikace a dále soubor s daty, které je třeba odeslat. Jako soubor s daty byl použit typ souboru CSV, který nám zaručí přenositelnost mezi operačními systémy. Mezikrokem, který musí uživatel udělat, je doplnění určitých údajů do XML popisu formuláře, které slouží jako mapovací informace. Jedná se o mapování určitého elementu formuláře na určitý sloupec s daty v CSV souboru. Pomocí těchto informací již aplikace bude schopna cyklicky odeslat veškerá stanovená data pomocí protokolu HTTP stejně, jako kdybychom je vyplnili na internetu sami.

Program je schopen stáhnout jakoukoliv HTML stránku z webu a je schopen z ní analyzovat všechny formuláře a vygenerovat jejich XML popis. Po namapování elementů formuláře na příslušná data CSV souboru je aplikace také schopna odeslat data do formuláře. Aplikace je schopna generovat jak požadavky typu GET, tak požadavky typu POST v obou způsobech kódování. Program je tedy schopen odesílat i soubory. Omezením aplikací je zejména neschopnost pracovat se stránkami, které jsou dostupné pouze po přihlášení v rámci webových stránek na serveru. Dále program neumí pracovat se zabezpečeným protokolem HTTPS.

V dohledné době by bylo vhodné program rozšířit o několik nových funkcí. Klient pro připojení k HTTP serveru by měl být schopen automaticky zažádat o novou stránku v případě, že server vrátí chybový kód 300, tedy že stránka byla přesunuta. Dále by bylo vhodné připravit grafické uživatelské rozhranní, jednak pro pohodlnější práci, ale zejména také pro lepší názornost funkcí programu. Určitě by také bylo potřeba zamyslet se nad možnými způsoby práce s formuláři, ke kterým je nutné se přihlásit, a také nad možným způsobem práce s HTTPS protokolem.

# Literatura


[1] Dudek, J.: Vývoj a standardy XHTML, 7. 6. 2002. Dokument dostupný na URL http://interval.cz/clanky/vyvoj-a-standardy-xhtml/

[2] Fielding, R.: Hypertext Transfer Protocol - HTTP/1.1, RFC 2616, The Internet Society, 1999

[3] Harold, E., R., Means, W., S.: XML v kostce – Pohotová referenční příručka, Computer Press, Praha 2002

[4] Kosek, J.: Protokol HTTP, 23.11.2006. Dokument dostupný na URL http://www.kosek.cz/vyuka/izi228/prednasky/http/frames.html

[5] Mikle, P.: XDHTML - Referenční příručka, Zoner Press, Brno 2004

[6] Schafranovich, Y: Common Format and MIME Type for Comma-Separated Values (CSV) Files, RFC 4180, SolidMatrix Technologies, Inc., 2005. Dokument dostupný na URL http://tools.ietf.org/html/rfc4180

[7] Snížek, M.: XHTML – vývoj a jeho možnosti, 22. 7. 2002. Dokument dostupný na URL http://interval.cz/clanky/xhtml-vyvoj-x-html-a-jeho-moznosti/

[8] Snížek, M.: XHTML – formuláře, 6. 5. 2003. Dokument dostupný na URL http://interval.cz/clanky/xhtml-formulare/

[9] The Cplusplus Resources Network. Dokument dostupný na URL http://www.cplusplus.com

[10] Výborný, O.: WWW,HTTP servery. Dokument dostupný na URL http://www.fi.muni.cz/~kas/p090/referaty/2003-podzim/skupina10/xvyborny_www.html

[11] Wikipedia, CS: HTML. Dokument dostupný na URL http://cs.wikipedia.org/wiki/Html

[12] Wikipedia, CS: XHTML. Dokument dostupný na URL http://cs.wikipedia.org/wiki/XHTML

[13] Wikipedia, CS: Hyper Text Transfer Protocol. Dokument dostupný na URL http://cs.wikipedia.org/wiki/HTTP

[14] Wikipedia, CS: XML. Dokument dostupný na URL http://cs.wikipedia.org/wiki/XML

[15] Wikipedia, EN: XML. Dokument dostupný na URL http://en.wikipedia.org/wiki/XML

[16] Wikipedia, CS: CSV. Dokument dostupný na URL http://cs.wikipedia.org/wiki/CSV

[17] Zapletal, L.: Protokol HTTP 1.1 pod lupou, 27. 3. 2001. Dokument dostupný na URL http://www.root.cz/clanky/protokol-http-1-1-pod-lupou/

# Seznam příloh

# Příloha 1 – DTD navržené XML struktury

Pro navržený XML popis byla vytvořena následující definice DTD, která jednoznačně určuje možnou syntaxi dokumentu.

```
<!ELEMENT form (element|itemsel|selection|textarea)*>
<!ATTLIST form    id ID #REQUIRED
                  action CDATA #REQUIRED
                  method (GET,POST) #REQUIRED
                  accept CDATA #IMPLIED
                  enctype CDATA #REQUIRED>
<!ELEMENT element (label?,type,value,alt?)>
<!ATTLIST element id ID #REQUIRED
                  maxlength CDATA #IMPLIED
                  accept CDATA #IMPLIED
                  readonly (y,n) #IMPLIED
                  column CDATA #REQUIRED>
<!ELEMENT selection (itemsel)+>
<!ATTLIST selection     id ID #REQUIRED
                        multiple (y,n) #IMPLIED
                        column CDATA #REQUIRED>
<!ELEMENT itemsel (value?,description?)>
<!ATTLIST itemsel id ID #REQUIRED
                  selected (y,n) #IMPLIED
                  readonly (y,n) #IMPLIED
                  column CDATA #REQUIRED
                  csvvalue CDATA #IMPLIED>
<!ELEMENT textarea (label?,text)>
<!ATTLIST textarea      id ID #REQUIRED
                        readonly (y,n) #IMPLIED
                        column CDATA #REQUIRED>
<!ELEMENT label (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT alt (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT text (#PCDATA)>
```

# Příloha 2 – Manuál programu

Tento manuál k programu je podrobnějším popisem ovládání a různého nastavení. Proto obsahuje podobné informace jako kapitola 5 diplomové práce. Zde ovšem nalezneme podrobnější popis i s příklady, který není vhodné zahrnout do samotné práce.

WebProcess je konzolová aplikace, která je schopná vytvořit XML popis Vámi zadaného formuláře a po malých úpravách vygenerovaného XML souboru je schopna automaticky odesílat všechna data z připraveného souboru do formuláře. Všechny funkce a možnosti se zapínají a nastavují z příkazové řádky, existuje také jednoduchý soubor s výchozím nastavením, který usnadní časté zadávání stejných údajů. Aplikace je platformově nezávislá a může být tedy provozována na jakémkoliv operačním systému.

**Přeložení programu**

Program je přeložitelný jak pod systémem linux, tak pod operačním systémem z rodiny Microsoft Windows.

Pod operačním systémem linux (unix) stačí pouze zadat příkaz „make“ a vytvořený soubor „Makefile“ se již postará o správné přeložení a nalinkování všech součástí programu. K překladu slouží GNU C++ překladače, které již bývají standardní součástí těchto operačních systémů. Po úspěšném překladu bude ve stejném adresáři vytvořen soubor „webprocess“, který představuje samotnou aplikaci.

Vzhledem k faktu, že byl tento software vyvíjen pod operačním systémem Microsoft Windows, máme dvě možnosti překladu. V adresáři aplikace se nachází soubor s kompletním projektem patřící k vývojovému prostředí DevC++. Pokud toto prostředí máme, stačí pouze otevřít projekt a nechat přeložit. DevC++ využívá překladače MinGW, které vycházejí z GNU C++ překladačů a jsou upraveny pod operační systémy Windows. Pokud není k dispozici vývojové prostředí DevC++, ale je k dispozici GNU C++ překladače pod Windows a program „make“ pro překlad, stačí nám přepsat název souboru „Makefile.win“ na „Makefile“ a pustit příkaz „make“. Pokud vše funguje v pořádku, dojde k přeložení všech souborů a nalinkování do jednoho programu. Výsledkem bude existující soubor „webprocess.exe“, který by již měl být plně spustitelný.

**Syntaxe volání programu**

```
webprocess [[-g url]|[-s]] [-h] [-d char] [-o file] [-x file] [-l file] [-c file]
```

Charakter jednotlivých přepínačů se může lišit podle způsobu použití aplikace. Tedy jestli budeme stahovat HTML stránku a vytvářet XML popis, nebo jestli se již chystáme automaticky odesílat data.

Proto budou jednotlivé parametry a přepínače vysvětleny v následujících kapitolách samostatně dle typu použití. Jediným společným přepínačem je –h, který nám zajistí výpis nápovědy, ve které nalezneme syntaxy volání programu a stručnou informaci k jednotlivým přepínačům.

**Stáhnutí HTML stránky a vytvoření XML popisu**

Prvním krokem k automatizaci webového formuláře je stáhnutí dané HTML stránky a vytvoření XML popisu jejího formuláře. Tento krok dělá program automaticky a definujeme ho přepínačem –g, za kterým musí následovat URL dané HTML stránky. Program se pak již pokusí připojit k serveru a poslat požadavek na HTML stránku. Poté co přijde odpověď, uloží se přijatá HTML stránka do souboru a je postoupena dalšímu zpracování. Ihned proběhne i analýza HTML stránky a vytvoření XML popisu formulářů, které byly nalezeny v dané HTML stránce. Pro každý formulář je vytvořen zvlášť XML soubor s jeho popisem. První formulář je uložen do souboru specifikovaném v nastavení nebo pomocí příkazové řádky a ostatní formuláře se ukládají do podobného souboru. Pokud je výstupní soubor „output.xml“, bude další formulář v souboru „output.xml2“ atd.

Pomocí parametrů na příkazové řádce můžete nastavit následující vlastnosti:

- `-o file` … „file“ obsahuje novou cestu k souboru, do kterého se bude ukládat stažená webová stránka
- `-x file` … „file“ obsahuje novou cestu k souboru, do kterého se uloží vytvořený XML popis formuláře
- `-l file` … přepínač „-l“ zapne logování připojení k webovému serveru, parametrem „file“ můžeme specifikovat nový soubor pro ukládání záznamů

Pokud používáte stále stejné výstupní soubory pro XML popis, logové soubory a jiné, je vhodné správně nastavit soubor „setting.ini“ pro zjednodušení práce. O možnostech tohoto souboru si povíme v další části nápovědy.

**Úprava XML popisu**

Před dalším zpracování pomocí programu je potřeba upravit vygenerovaný XML soubor. Jelikož by aplikace nepoznala, která data v CSV souboru patří ke které položce formuláře, je zapotřebí doplnit nezbytné informace do XML popisu. Jedná se především o dvě informace. První z nich je „column“ neboli sloupec. Tato informace nám u každého elementu XML popisu jednoznačně určí příslušnost určité informace z CSV souboru k určité položce formuláře. Druhou důležitou informaci je „csvvalue“, neboli hodnota v CSV souboru. Tato hodnota se doplňuje jenom u výběrových prvků formuláře, jakou jsou seznamy, zaškrtávací políčka a jiné. Vzhledem ke struktuře těchto polí by nebylo jasné, podle čeho poznat, která položka má nebo nemá být vybrána, Je tedy potřeba pomocí hodnoty „csvvalue“ specifikovat, která hodnota nalezená v příslušném sloupci CSV souboru značí, že

daná položka bude vybrána, případně zaškrtnuta. Prakticky je tedy potřeba doplnit hodnoty do všech prvků, které obsahují hodnoty „column“ nebo „csvvalue“.

Samozřejmě existuje pár pravidel, kdy tyto hodnoty není potřeba doplnit. Pokud například potřebujeme, aby se vždy odesílala výchozí hodnota u daného prvku, tak necháme „column“ prázdné a bude se odesílat jeho výchozí hodnota. Druhá výjimka nastává u tlačítek. Pokud máme prvek s označením „element“ a jeho vnitřní prvek „type“ obsahuje text „submit“, tak se jedná o tlačítko. Pokud takových tlačítek je ve formuláři více, tak je určitě potřeba vybrat to správné, pomocí kterého jakoby formulář odešleme. Toto tlačítko vybereme tak, že zadáme do atributu „column“ jakoukoliv hodnotu. Pokud tuto hodnotu necháme prázdnou, tak se tlačítko vůbec neodešle. Dává to tedy možnost vybrat si tlačítko, pomocí kterého formulář odešleme. Tato možnost je potřeba, protože řada webových skriptů rozlišuje při zpracování druh tlačítka, které bylo stisknuto, a podle toho následují odlišné funkce zpracování.

**Následuje pár příkladů správného vyplnění:**

- **Klasický textový element** – na obrázku 1 vidíme jak takový prvek může vypadat, zde stačí pouze doplnit číslo sloupce v CSV souboru do „column“; pokud necháme hodnotu prázdnou, tak se odešle text v elementu „value“

```
- <element column="1" id="Text">
    <type>text</type>
    <value />
  </element>
```

Obrázek 1. Textový element v XML

- **Výběrová pole –** výběrové pole se může vyskytovat jako jeden samostatný zaškrtávací prvek, nebo jako skupina souvisejících voleb v rolovací nabídce. V obou typech je potřeba nastavit hodnotu „column“ a „csvvalue“ podobně jako na obrázku 2.

```
- <itemsel column="3" csvvalue="1" id="Checkbox">
    <value>y</value>
  </itemsel>
- <selection column="4" id="Radio">
  - <itemsel csvvalue="Radio1" id="Radio" selected="selected">
      <value>r1</value>
    </itemsel>
  - <itemsel csvvalue="Radio2" id="Radio">
      <value>r2</value>
    </itemsel>
  - <itemsel csvvalue="Radio3" id="Radio">
      <value>r3</value>
    </itemsel>
  </selection>
```

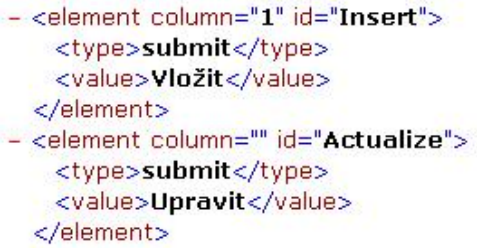
Obrázek 2. Výběrová pole v XML

- **Pole s mnohonásobným výběrem –** prvek s mnohonásobným výběrem je podobný jako v předcházejícím bodě, ale abychom měli možnost definovat více voleb v CSV

souboru, tak pro každou volbu musí existovat samostatný sloupec. Proto je nutné vyplnit hodnoty „column“ a „csvvalue“ u každé možnosti, jako je to na obrázku 3.


Obrázek 3. Pole pro vícenásobný výběr v XML

- **Tlačítka –** na obrázku 4 je vidět, že se formulář odešle pomocí tlačítka s hodnotou „Vložit“ v prvku „value“




Obrázek 4. Tlačítka v XML

**Odeslání dat do formuláře**

Posledním krokem k automatizaci webového formuláře je odeslání připravených dat z CSV souboru do formuláře pomocí upraveného XML popisu. Tento krok dělá program automaticky a definujeme ho přepínačem –s. Program se pak již pokusí připojit k serveru a cyklickým průchodem přes XML popis poslat požadavky na HTML stránku obsahující námi odesílaná data. Poté, co přijde odpověď, uloží se přijatá HTML stránka do souboru jako výsledek odeslání dat. Pro každý vyplněný formulář (jeden cyklus odeslání dat) je vytvořen samostatný soubor. Tyto soubory se ukládají do adresáře stanoveného přepínačem –o.

Pomocí parametrů na příkazové řádce můžete nastavit následující vlastnosti:

- `-c file` … „file“ obsahuje cestu k CSV souboru, který obsahuje data pro odeslání

- `-x file` … „file“ obsahuje cestu k XML souboru, který obsahuje XML popis formuláře
- `-o directory` … „direcotry“ obsahuje cestu k adresáři, do kterého se budou ukládat odpovědi z webového serveru na jednotlivá odesílaná data
- `-l file` … přepínač „-l“ zapne logování připojení k webovému serveru, parametrem „file“ můžeme specifikovat nový soubor pro ukládání záznamů
- `-d char` … „char“ obsahuje znak, který bude použit jako oddělovač sloupců v CSV souboru

Pokud používáte stále stejné výstupní soubory pro XML popis, logové soubory a jiné, je vhodné správně nastavit soubor „setting.ini“ pro zjednodušení práce. O možnostech tohoto souboru si povíme v další části nápovědy.

**Výchozí nastavení**

Výchozí nastavení aplikace je výhodné zejména v případě, kdy často používáme stejné cesty k souborům a adresářům a je pro nás nepohodlné je stále znovu zapisovat pomocí parametrů na příkazové řádce. Toto nastavení můžeme upravovat v souboru „setting.ini“, který se nalézá ve stejné složce jako samotný program. Nejedná se o plnohodnotný „ini“ soubor, ale pro takto jednoduché nastavení plně postačuje. Jako výchozí nastavení můžeme definovat následující parametry.

```
OutputPage=./=html=/download.html
XmlFile=./=xml=/output.xml
CsvFile=./=csv=/file.csv
CsvDelimiter=;
OutputDirectory=./=html=/
GetPageLog=./=log=/HTTPDownload.log
SendDataLog=./=log=/HTTPSend.log
```

Obrázek 5: Ukázka souboru setting.ini

Význam je následující:

- OutputPage … soubor, do kterého se ukládá stažená webová stránka
- XmlFile … soubor do kterého se ukládá, nebo ze kterého se čte XML popis formuláře
- CsvFile … soubor, ve kterém jsou uložena data pro odesílání
- CsvDelimiter … znak, který bude použit jako oddělovač v CSV souboru
- OutputDirectory … adresář pro ukládání výstupních souborů, které jsou odpovědmi od webového serveru na posílaná data
- GetPageLog … soubor, do kterého se bude ukládat záznam komunikace s webovým serverem při stahování webové stránky
- SendPageLog … soubor, do kterého se bude ukládat záznam komunikace s webovým serverem při odesílání dat