VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AUTOMATIZOVANÁ NAVIGACE NA PRIVÁTNÍCH STRÁNKÁCH

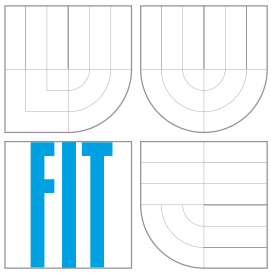
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADEK KLIMENT

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AUTOMATIZOVANÁ NAVIGACE NA PRIVÁTNÍCH STRÁNKÁCH

AUTOMATIC NAVIGATION ON PRIVATE WEBSITES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADEK KLIMENT

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZBYNĚK KŘIVKA, Ph.D.

BRNO 2012

Abstrakt

Tato diplomová práce se zabývá technologiemi souvisejícími s webovými stránkami a popisuje navigaci na nich, včetně přihlašování do privátních částí a udržování kontextu uživatele. Je zde rozebrán návrh mechanismu pro automatizovanou navigaci zahrnujícího skriptovací jazyk i prostředky pro vizuální popis. Uveden je také návrh aplikace, která mechanismus využívá, a implementace jejích jednotlivých částí. V poslední kapitole je popsáno testování na různých webových stránkách a jsou shrnuty z něj získané poznatky.

Abstract

This thesis deals with technologies related to web pages and describes the navigation across them including the authentication to access their private sections and the user context management. It introduces the design of the mechanism for the automated navigation including new scripting language and tools for the visual description. The work also contains the design of the application using the mechanism and the implementation of its parts. The last chapter sums up the knowledge acquired by testing on various websites.

Klíčová slova

navigace, navigační aktivita, webová stránka, extrakce informací, automatické vyplňování formulářů, Jython, platforma NetBeans

Keywords

navigation, navigation activity, web page, information extraction, automatic form filling, Jython, NetBeans platform

Citace

Radek Kliment: Automatizovaná navigace na privátních stránkách, diplomová práce, Brno, FIT VUT v Brně, 2012

Automatizovaná navigace na privátních stránkách

Prohlášení

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

.....
Radek Kliment
23. května 2012

Poděkování

Chtěl bych poděkovat Ing. Zbyňku Křivkovi, Ph.D., za jeho ochotu, konzultace a rady, které mi v průběhu tvorby této práce poskytl.

Obsah

Kapitola 1

Úvod

Internet v dnešní době dosáhl obrovského rozmachu. Spousta informací je tak dostupná skrze webové stránky. Nelze se tedy divit, že pro spoustu lidí je procházení stránek a vyhledávání i zadávání nejrůznějších informací běžnou součástí každého dne. K tomu velmi dobře poslouží internetový prohlížeč.

Čas od času se ale stává, že chcete provádět stejnou nebo velmi podobnou činnost opakovaně, ale nechce se vám vždy znovu procházet potřebnou sekvencí stránek, aby jste požadovanou informaci zadali nebo nalezli. Platí to obzvláště v případech, kdy je k dosažení cíle potřeba projít větším množstvím stránek a narůstá tak časová náročnost. Taková situace může nastávat v pravidelných intervalech, ale i nepravidelně dle aktuálních potřeb. Dalším případem může být provedení série aktivit, přičemž každá se liší pouze jedním nebo více parametry, například zadáním jiného textu do pole ve formuláři. V těchto případech by řada lidí ocenila, kdyby měli možnost takovéto činnosti provádět automatizovaně a ušetřilo jim to tak spoustu cenného času.

Každý web má ale svoji specifickou strukturu, takže i provádění podobných aktivit na různých stránkách se může skládat z naprosto odlišných kroků. Je tedy nutné je definovat zvlášť pro každou činnost a konkrétní web. Proto je důležité, aby popis kroků zajišťujících provedení dané aktivity bylo možné zadávat jednoduchým způsobem a nebylo tedy příliš časově náročné.

Cílem této práce je návrh mechanismu, který umožní automatizovanou navigaci webovými stránkami, a to i včetně přihlašování a procházení jejich privátních částí. Poskytne také prostředky pro zadávání dat načtených ze souborů a extrakci informací ze stránky. Důraz je kladen na uživatelskou přívětivost a efektivitu zadávání popisu navigace.

Motivací pro vytvoření této práce je také skutečnost, že jsem již několikrát potřeboval na určitých stránkách programově realizovat nějakou aktivitu. Většinou jsem volil programovací jazyk Java a aplikaci realizující danou aktivitu vytvořil. Dostupné prostředky pro realizaci jsou však vcelku nízkoúrovňové a strukturu stránky jsem musel odděleně zkoumat s využitím prohlížeče a jeho pluginů. Chyběl mi ucelený nástroj, který by umožňoval navigační aktivitu jednoduchým způsobem popsat a tvorbu popisu co nejvíce usnadnil. Vytvořením právě takovéhoto nástroje se tato diplomová práce zabývá.

1.1 Struktura práce

V kapitole 2 jsou popsány technologie týkající se webových stránek a navigace na nich. Kapitola 3 se zabývá navigací na webu, metodami pro přihlašování do privátních částí stránek

a udržováním kontextu přihlášeného uživatele. Obsahuje i popis existujících nástrojů pro extrakci dat, z nichž některé umožňují i navigaci. V kapitole 4 je uveden návrh mechanismu pro navigaci webovými stránkami, včetně skriptovacího jazyka pro její popis a prostředků pro její grafické zadávání. Kapitola 5 obsahuje návrh uživatelského rozhraní aplikace a popis několika způsobů, jimiž lze vytváření popisu usnadnit. Je v ní také zmíněn návrh dialogů pro konfiguraci uzlů při vytváření vizuálního popisu a možnosti, jakými lze popsané navigační aktivity spouštět. Kapitola 6 popisuje platformu NetBeans, na které je aplikace postavena, použité knihovny a implementaci navrženého mechanismu a jednotlivých částí aplikace. Nakonec jsou v kapitole 7 shrnuty poznatky z testování implementovaného mechanismu a aplikace.

Tato práce navazuje na Semestrální projekt. Byly z něj převzaty kapitoly 2 a 3. Kapitola 4 byla převzata s jistými úpravami. Poslední převzatou kapitolou je 5, která však byla značně rozšířena.

Kapitola 2

Související technologie

V této kapitole jsou popsány technologie, které souvisí s webovými stránkami a jejich přenosem. Je zde zmíněn formát XML pro ukládání strukturovaných textů a objektový model pro zpracování dokumentů v aplikacích.

2.1 World Wide Web

World Wide Web, zkráceně WWW, je množina internetových protokolů a dalších souvisejících technologií, které slouží *k prezentaci hypertextových dokumentů* [1]. Využívá internetu jakožto sítě tvořené počítači po celém světě.

Hypertextové dokumenty obsahují kromě různých informací také odkazy ukazující na další dokumenty, čímž umožňují jejich rychlé a efektivní prohlížení. Dokumenty tak tvoří orientovaný graf.

S WWW souvisí další technologie, jako je například URI pro identifikování zdrojů nebo HTTP pro přenos dat. Jejich popis je součástí této podkapitoly. Pro další technologie, jako například HTML, jsou vyhrazeny zvláštní podkapitoly 2.2 a další.

2.1.1 URI

Universal Resource Identifier umožňuje identifikovat zdroj v síti. Existují dva typy identifikátorů. *URL* je identifikátor popisující umístění zdroje v síti. Identifikátor *URN* pojmenovává zdroj nezávisle na jeho umístění, jméno musí být unikátní. V souvislosti s adresací zdroje se nejčastěji používá pojem URL [1].

URL má tvar `<schéma>://<doména>:<port>/<cesta ke zdroji>?<parametry>`. Některé části však nejsou povinné.

2.1.2 HTTP protokol

HTTP je v prostředí WWW nejčastěji používaný protokol pro přenos dokumentů. Umožňuje přenos jakýchkoliv dat, přičemž jejich typ je specifikován rozšířením *MIME*. HTTP je bezstavový protokol pracující na principu požadavek – odpověď. Protokol samotný tedy neumožňuje určit, zda spolu dva požadavky souvisí či nikoliv [1]. Pro udržování kontextu je tak potřeba využít dalších mechanismů, které jsou popsány v části 3.3.

Požadavek

HTTP požadavek obsahuje na prvním řádku typ požadavku, cestu ke zdroji a verzi protokolu. Následují položky hlavičky ve tvaru `jmeno: hodnota`. Každá je uvedena na zvláštním řádku a jejich počet není omezen. Za poslední položkou je prázdný řádek. Dále mohou následovat data, jejichž podoba závisí na typu MIME. Příklad hlavičky HTTP požadavku je v ukázce 2.1.

```
GET / HTTP/1.1
Host: www.fit.vutbr.cz
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: cs,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Ukázka 2.1: Příklad hlavičky HTTP požadavku

Protokol definuje několik typů požadavků, z nich nejdůležitější jsou `GET` a `POST`. Obě metody umožňují odeslat na server data.

V případě metody GET však data musí být součástí URL adresy na prvním řádku požadavku. Její délka však je omezena. Metoda se nejčastěji používá pro získání dokumentu na základě URL, lze ji použít ale i pro odeslání formuláře s malým množstvím dat.

V případě metody POST nejsou data součástí URL, ale následují za hlavičkou požadavku. Díky tomu, že data nejsou viditelná v URL, se často používá pro odesílání dat z přihlašovacích formulářů. Další běžné použití je pro odesílání většího množství dat, např. souborů.

Odpověď

Odpověď na požadavek obsahuje na prvním řádku stavový kód odpovědi a jeho popis. Na dalších řádcích následují položky hlavičky stejně jako u požadavku. Za hlavičkou mohou opět následovat data ve formátu daném typem MIME. Příklad hlavičky HTTP odpovědi je v ukázce 2.2.

```
HTTP/1.1 200 OK
Date: Sun, 29 Apr 2012 07:12:01 GMT
Server: Apache
Content-Location: index.php.cz
Pragma: no-cache
Keep-Alive: timeout=60, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-2
Content-Language: cs
```

Ukázka 2.2: Příklad hlavičky HTTP odpovědi

Položky hlavičky mohou obsahovat délku posílaných dat, typ obsahu, použité kódování a další informace.

Stavové kódy lze rozdělit do několika kategorií na základě jejich čísla. Pro každou kategorii jsou uvedeny příklady. Úplný seznam stavových kódů a jejich popis je dostupný v RFC 2616 [2] v části 10.

- `1xx` pro informativní zprávy,
  - 100 Continue,
- `2xx` pro úspěšné požadavky,
  - 200 OK,
  - 201 Created,
  - 204 No Content,
- `3xx` pro přesměrování,
  - 301 Moved Permanently,
  - 303 See Other,
  - 304 Not Modified,
- `4xx` pro chybné požadavky ze strany klienta,
  - 400 Bad Request,
  - 401 Unauthorized,
  - 403 Forbidden,
  - 404 Not Found,

  `5xx` pro chyby na serveru,
  - 500 Internal Server Error,
  - 503 Service Unavailable

HTTPS

HTTPS využívá pro přenos dat zabezpečení pomocí SSL či TLS, jinak jsou principy totožné s protokolem HTTP. Místo portu 80 se standardně používá port 443. Užití tohoto protokolu lze poznat ze začátku URL adresy, která obsahuje schéma `https`. Je vhodné ho použít, pokud komunikace obsahuje citlivé údaje a je potřeba zamezit jejich odposlechu.

2.2 HTML

HTML je značkovací jazyk určený pro vytváření webových stránek. Standard definuje rozložení elementů na stránce, tedy jakým způsobem mohou být vzájemně zanořeny. Jednotlivým elementům je přiřazen význam, není však standardizována přesná interpretace prohlížečem.

Pro HTML je typické, že některé elementy nemusí obsahovat ukončující značky, jelikož se ukončení elementu dá odvodit z jeho kontextu. Hodnoty atributů nemusí být uzavřeny v uvozovkách, hodnota dokonce ani nemusí být specifikována. Umožňuje to jednodušší zápis, ale na druhou stranu způsobuje komplikace při zpracování dokumentu v aplikacích.

Detailní informace o aktuální verzi HTML 4.01 lze nalézt ve W3C specifikaci [3].

2.2.1 Elementy podílející se na navigaci

Odkaz Základem pro navigaci je odkaz tvořený elementem `<a>`. Jeho obsah tvoří text odkazu a cíl je v podobě URL adresy uveden v atributu `href`.

Formulář Pro navigaci jsou také podstatné formuláře reprezentované elementem `<form>`. URL pro zpracování formuláře je uvedena v atributu `action` a použitou HTTP metodu definuje atribut `method`. Tento element obsahuje další elementy určující kontrolní prvky, z nichž se formulář skládá.

Může být tvořen elementy `<input>`, jejichž typ je různý v závislosti na hodnotě atributu `type`. Možné hodnoty jsou: [3]

- `text` – vytvoření textového pole,
- `password` – vytvoření pole pro zadání hesla, kdy jsou napsané znaky nahrazeny zástupným symbolem (např. `*`) a prohlížeč neukládá historii zadaných hodnot,
- `checkbox` – vytvoření zatrhávacího tlačítka,
- `radio` – vytvoření skupiny tlačítek, z nichž je možné vybrat jednu volbu,
- `submit` – vytvoření tlačítka pro odeslání formuláře,
- `reset` – vytvoření tlačítka pro obnovení výchozích hodnot,
- `hidden` – vytvoření skrytého pole,
- `file` – pro vložení souboru

Další možností je vložení elementu `<select>` reprezentujícího roletové menu umožňující výběr jedné hodnoty ze seznamu. Položky seznamu jsou určeny podelementy `<option>` elementu `<select>`. Element `<textarea>` definuje textové pole pro zadání většího množství textu.

Příklad HTML formuláře s několika prvky je uveden v ukázce 2.3.

```
<form id="addPerson" name="addPerson" method="get"
      action="persons_add.php">
  <label for="name">jméno</label>
  <input type="text" name="name" id="name"/><br />
  <label for="gender">pohlaví</label>
  <select name="gender" id="gender">
    <option value="muž">muž</option>
    <option value="žena">žena</option>
  </select><br />
  <input type="radio" name="status" id="married"
         value="manželství" checked="checked"/>
  <label for="married">manželství</label>
  <input type="radio" name="status" id="single"
         value="svobodný"/>
  <label for="single">svobodný</label><br />
  <input type="submit" name="addPerson" id="addPerson"
         value="Přidat osobu"/>
</form>
```

Ukázka 2.3: Příklad HTML formuláře

2.3 XML

Extensible Markup Language je značkovací jazyk standardizovaný a vyvíjený konsorciem W3C. Vznikl zjednodušením jazyka SGML v důsledku jeho velké složitosti a náročnosti zpracování v aplikacích. Podnětem pro vznik bylo také omezení HTML, které obsahuje pouze předdefinované značky a je určeno zejména pro prezentaci informací.

XML umožňuje uživateli nadefinovat si vlastní značky v závislosti na potřebách konkrétní aplikace. Značky určují význam obsažených informací, neříkají ale nic o jejich vzhledu. Prezentace informací závisí na aplikaci, která dokument zpracovává. Formát XML je určen především pro platformně nezávislý přenos dat

Jazyk musí ale dodržovat přísnou syntaxi. Elementy musí obsahovat otevírací i ukončovací značku. V tomto se syntaxe liší od HTML, kde lze v některých případech ukončující značku vynechat. Atributy jsou součástí otevírací značky, v rámci elementu musí být unikátní, jejich hodnotu nelze vynechat a musí být uvedena v uvozovkách. To je další rozdíl oproti HTML, kde hodnoty atributů nemusí být v uvozovkách, dokonce nemusí být uvedeny vůbec. Díky dodržování striktní syntaxe je zpracování XML dokumentů v aplikacích jednodušší.

Příklad XML dokumentu je uveden v ukázce 2.4 a je využit u příkladů XPath výrazů v tabulce 2.1 v následující části.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

Ukázka 2.4: Příklad XML dokumentu [4]

Více informací o jazyku XML lze nalézt v [5] a [6], včetně jeho aplikací a souvisejících technologií. Z těchto knih tato část čerpá.

2.4 XPath

XPath je jazyk umožňující adresovat různé části XML dokumentu. Uzly lze vybírat na základě jejich absolutní či relativní pozice, jejich obsahu, typu, hodnot atributů a dalších kritérií. Využívá se hierarchické struktury XML dokumentu. XPath výraz popisuje cestu k jednomu nebo více uzlům v dokumentu. Cesta se skládá z několika kroků. Zápis může připomínat cestu k souboru v adresářové struktuře.

Syntaxe kroku je `jméno_osy::název_uzlu[podmínky]`. Je určen jménem osy, názvem uzlu a dodatečnými podmínkami. Osa určuje směr procházení hierarchie uzlů, například

přímý potomek, následník, rodič, předchůdce, sourozenecké uzly, atributy a další. Za výchozí osu je považována osa přímých potomků. Místo uvedení osy atributů lze názvu atributu předřadit znak `@`. Uzly lze vybrat na základě jejich jména nebo použít zástupný znak `*` pro libovolné jméno. Množinu vybraných uzlů lze dále zúžit podmínkami, například jestli obsahují určité elementy, atributy s požadovanými hodnotami a podobně. Je možné využít i vestavěných operátorů a funkcí.

Specifikace jednotlivých kroků jsou ve výrazu odděleny znakem lomítka. Zápis nemusí být kompletní. Lze například využít výchozí osy přímých následníků nebo zkráceného zápisu pomocí dvojice lomítek pro výběr uzlu nezávisle na umístění v aktuálním uzlu.

Příklady XPath výrazů spolu s popisem jejich významu jsou uvedeny v tabulce 2.1.

`/bookstore`	Vybere kořenový element `bookstore`.
`bookstore/book`	Vybere elementy `book`, které jsou synovskými elementy elementu `bookstore`.
`bookstore//book`	Vybere elementy `book`, které jsou následníky elementu `bookstore`, přičemž nezávisí, kde přesně se uvnitř něj nacházejí.
`/bookstore/book[1]/title/@lang`	Vybere atribut `lang` elementu `title`, který je synovským elementem prvního elementu `book`. Místo zkráceného zápisu `@lang` lze využít i zápisu s názvem osy `attribute::lang`.
`//title[@lang='eng']`	Vybere všechny elementy `title`, které obsahují atribut `lang` s hodnotou `eng`.
`/bookstore/book[position()<3]`	Vybere první dva elementy `book`, které jsou synovské elementy kořenového elementu `bookstore`.
`/bookstore/*`	Vybere všechny synovské elementy kořenového elementu `bookstore`.

Tabulka 2.1: Příklady XPath výrazů [4]

XPath výrazy nejsou omezeny pouze na XML, ale lze je využít i pro výběr uzlů v dokumentech HTML.

Detailnější informace o XPath lze nalézt v knihách [5] a [6], pro rychlé seznámení je vhodný tutoriál [4]. Z těchto zdrojů bylo při vytváření této části čerpáno.

2.5 DOM

Document Object Model je aplikační rozhraní nezávislé na použité platformně a programovacím jazyce, které umožňuje programům a skriptům dynamicky přistupovat k dokumentu, včetně změny jeho obsahu, struktury a vzhledu [7]. Objektový model je určen pro reprezentaci dokumentů s hierarchickou strukturou, například XML nebo HTML. Specifikace [7] byla vytvořena konsorciem W3C a zahrnuje několik úrovní.

Standard vznikl v důsledku potřeby sjednotit rozhraní pro přístup k dokumentu používané v různých prohlížečích. V současné době většina prohlížečů tento standard z větší části implementuje, v některých případech lze ale v implementaci najít rozdíly [1].

2.6 Shrnutí

V této kapitole byly popsány technologie týkající se webových stránek. Byl představen jazyk URI pro identifikaci zdrojů na internetu, protokol HTTP jakožto základní kámen pro přenos dat a technologie HTML pro vytváření stránek, kde je pozornost zaměřena na prvky zajišťující navigaci na webu. Zmíněn byl formát XML pro ukládání strukturovaných dat, včetně související technologie XPath pro adresaci částí nejen XML, ale i HTML dokumentů. Na závěr byl popsán objektový model využívaný pro usnadnění aplikačního přístupu k jednotlivým částem dokumentu a jejich modifikaci.

Mnohé weby vyžadují pro přístup k jejich určitým částem přihlášení uživatele, přičemž existuje více možností, jak ho lze realizovat. Souvisí s tím i potřeba udržování kontextu, jelikož HTTP protokol samotný to neumožňuje. Těmito záležitostmi se zabývá následující kapitola. V její druhé polovině jsou zmíněny i existující nástroje, které jsou sice zaměřeny na extrakci dat z webových stránek, ale některé z nich obsahují i podporu pro popis navigace na požadovanou stránku.

Kapitola 3

Navigace na webových stránkách

V této kapitole jsou nejprve popsány různé typy kroků, které se při procházení webu využívají. Následuje popis možností využívaných pro přihlášení do privátních částí stránek a mechanismů umožňujících udržování kontextu přihlášeného uživatele. Jsou zde zmíněny i nástroje umožňující navigaci na požadovanou stránku a extrakci informací, které obsahuje.

3.1 Typy navigačních kroků

Navigace na webu se obvykle skládá z několika kroků potřebných k dosažení stránky, jejíž obsah uživatele zajímá nebo prostřednictvím ní potřebuje zadat nějaké údaje. Krokem se rozumí přechod z aktuální stránky na následující a může být proveden několika způsoby.

- *Načtení stránky s požadovanou URL*, což je typicky první krok navigace. Uživatel zadá v prohlížeči adresu stránky, případně může využít uložených záložek.
- *Následování odkazu*, kdy uživatel na aktuální stránce vybere některý z odkazů vedoucí na další stránku.
- *Odeslání formuláře*, v tomto případě uživatel vyplní položky ve formuláři a kliknutím na příslušné tlačítko provede jeho odeslání.

3.2 Přihlašování do privátních částí

V této části jsou popsány metody využívané k autentizaci uživatele pro přístup do privátních částí webových stránek. Tyto metody jsou podrobně rozebírány v [8].

3.2.1 Autentizace s využitím protokolu HTTP

Schéma HTTP autentizace je specifikováno v RFC 2617 [9]. Server si vynucuje zaslání údajů odesláním HTTP odpovědi se stavovým kódem `401 Unauthorized` a hlavičkou s položkou `WWW-Authenticate`, která obsahuje informace o požadovaném způsobu autentizace. Existují dva a jsou rovněž popsány v uvedeném RFC.

První z nich je tzv. *Basic Access*, kdy se uživatelské jméno a heslo odesílá oddělené dvojtečkou a zakódované metodou *base64*. Tato metoda však není určena pro šifrování a údaje tak může kdokoliv dekódovat. Proto se tento způsob z důvodu bezpečnosti často využívá v kombinaci se spojením zabezpečeným pomocí SSL.

Při využití metody zvané *Digest Access* server v odpovědi specifikuje metodu použitou pro výpočet kontrolního součtu a hodnotu *nonce* unikátní pro dané spojení. Uživatelské údaje se na server zašlou jako kontrolní součet uživatelského jména a hesla, navíc spolu s kontrolním součtem URL požadavku a typem HTTP metody. Na konec se připojí ještě obdržená hodnota *nonce*.

Prohlížeč po obdržení odpovědi se stavovým kódem `401 Unauthorized` zobrazí dialog pro zadání uživatelského jména a hesla. Po potvrzení údajů opakuje původní požadavek, do hlavičky však přidá položku `Authorization` obsahující údaje zakódované dle použité metody. Jelikož je tento způsob autentizace bezstavový stejně jako protokol HTTP, musí se zadané údaje zasílat při každém požadavku. Prohlížeč si je proto uloží a odesílá automaticky, aby je uživatel nemusel zadávat opakovaně.

Kromě výhody jednoduchého použití má tento způsob ale i určité nevýhody. První z nich je ta, že se uživatelské jméno a heslo musí zasílat při každém požadavku na server, což zvyšuje nebezpečí jejich odchycení. Tato nevýhoda je ale potlačena při použití Digest Access metody nebo využití SSL spojení. Druhou nevýhodou je, že si server nemůže žádným spolehlivým způsobem vynutit odhlášení uživatele, tedy donutit prohlížeč, aby uživatelské údaje smazal z paměti a nadále neposílal.

3.2.2 Autentizace pomocí formulářů

Tento způsob je v současné době nejrozšířenějším. Autentizace je plně v režii webové aplikace, která v případě potřeby zobrazí formulář pro přihlášení. Tohoto může být docíleno například automatickým přesměrováním na stránku s formulářem. Ten se obvykle skládá z pole typu `text` pro zadání uživatelského jména, pole typu `password` pro zadání hesla a tlačítka pro potvrzení. Po jeho odeslání aplikace ověří zadané údaje a zobrazí stránku s chráněným obsahem. Pro přihlášení je však nutné udržovat kontext uživatele, aby při příštím přístupu na stejnou či související stránku nemusel údaje znovu zadávat. Používané mechanismy jsou popsány v následující kapitole 3.3.

Pro odeslání dat z přihlašovacího formuláře by se měla vždy použít HTTP metoda POST. V případě použití GET požadavku by se jednak zobrazily přístupové údaje v adresním řádku prohlížeče, ale uložily by se i do historie.

3.2.3 Další metody

Autentizace uživatele může být realizována pomocí Java appletu nebo flash objektu vloženého do webové stránky. Tuto možnost lze využít v případě, že je nutné vytvořit složitější interaktivní rozhraní pro přihlášení. Další možností je přihlašování do bezpečnostně kritických aplikací, například pro manipulaci s bankovními účty, kdy může být vyžadováno ověření uživatelského certifikátu.

3.3 Udržování kontextu

Potřeba udržování kontextu vyplývá z použití bezstavového protokolu HTTP. Pro rozpoznání, že příchozí požadavky jsou od jednoho a téhož uživatele, je proto nutné použít jiné mechanismy.

V případě HTTP autentizace není udržování kontextu nutné, jelikož jsou uživatelské údaje zasílány při každém požadavku, jak bylo popsáno v části 3.2.1.

V případě formulářového přihlašování je ale žádoucí, aby se přihlašovací údaje zaslaly jen při prvním požadavku. Řešením je vygenerování unikátního a náhodného identifikátoru, který je po přihlášení klientovi zaslán. Na serveru se vytvoří asociace mezi vygenerovaným identifikátorem a uživatelem. Ten se jím při dalších požadavcích prokazuje a není potřeba opakovaně zasílat uživatelské jméno a heslo. V případě, že chce server vynutit odhlášení uživatele, jednoduše se zruší asociaci identifikátoru a uživatele. Dá se to využít například po určité době nečinnosti nebo při obdržení podezřelého požadavku pro automatické odhlášení uživatele. V případě dalšího požadavku se klient musí znovu autentizovat přihlašovacími údaji.

Existuje několik způsobů, jakým si server a klient mohou jedinečný identifikátor uživatele předávat [10].

Cookies Tento přístup se v současné době používá nejčastěji. Cookie je dvojice název – hodnota zaslaná serverem prohlížeči v hlavičce odpovědi, např. po úspěšném přihlášení. Název může být libovolný, hodnota je tvořena vygenerovaným jedinečným identifikátorem. Prohlížeč si tuto informaci uloží a posílá spolu s každým dalším požadavkem, server podle ní uživatele identifikuje. Nevýhodou je, že uživatel může podporu cookies zakázat, v tom případě tento mechanismus selhává.

Skrytá formulářová pole Do formuláře na webové stránce se vloží skryté pole, jehož hodnota je tvořena identifikátorem uživatele. Pole není na stránce viditelné, lze ho však najít ve zdrojovém kódu stránky. Po odeslání formuláře server podle hodnoty tohoto pole identifikuje uživatele. Nevýhodou je, že je nutné, aby stránka obsahovala dynamicky vytvořený formulář s vloženým skrytým polem. V případě statických stránek tuto metodu nelze využít.

Využití URL Identifikátor uživatele se připojí jako parametr přímo k URL. Tento způsob má ale tu nevýhodu, že je identifikátor přímo viditelný v adresním řádku prohlížeče a může být snadno změněn. Navíc se jako součást URL adresy ukládá do historie.

Session API Pro udržování kontextu uživatele se využívá aplikačního rozhraní technologie použité na serveru. Tento přístup je postaven na předchozích metodách a lze ho považovat za nejlepší, jelikož programátorovi usnadňuje práci a napomáhá předejít chybám při implementaci.

3.4 Existující nástroje

V této části jsou popsány existující nástroje pro extrakci dat, které ale umožňují i navigaci na požadovanou stránku. Zmíněn je komerční produkt Lixto, diplomová práce zabývající se extrakcí informací z dynamických webových stránek a dvě bakalářské práce.

3.4.1 Lixto

Principy využívané nástrojem Lixto byly prezentovány v článku *Visual Web Information Extraction with Lixto* [11]. Lixto Visual Developer je komerční nástroj určený pro extrakci dat z webových stránek. Kromě toho ale poskytuje i možnost popsat navigaci nutnou pro dosažení dané stránky. Schopnosti nástroje jsou demonstrovány v článku *Web information acquisition with Lixto Suite: a demonstration* [12]. Z uvedených zdrojů tato část čerpá.

Navigace na cílovou stránku

Navigační kroky potřebné k získání cílové webové stránky se vytváří velmi interaktivním způsobem. Nástroj integruje komponentu zobrazující stránku a uživatel s jejím využitím může procházet web stejným způsobem, jako by používal běžný prohlížeč. Lze provádět klikání na odkazy, zadávání informací do formulářů, jejich odesílání a podobně. Tyto kroky jsou zaznamenávány a následně se provedou pro získání požadované stránky, ze které se data extrahují.

Extrakce dat

Pro extrakci dat se využívá tzv. *wrapperů*, pomocí kterých je možné požadovaná data z HTML stránky uložit jako XML soubor, který pak lze externě dále zpracovávat.

Wrapper je tvořen pojmenovanými vzory, které mohou být hierarchicky uspořádány. Každý vytvořený vzor reprezentuje určitý typ informace na webové stránce. Vzor je popsán pomocí filtrů, které umožňují identifikovat odpovídající uzly ve stránce. Aby informace odpovídala vzoru, musí odpovídat alespoň jednomu jeho filtru. Přidáním filtru lze tedy množinu odpovídajících informací rozšířit.

Filtr je tvořen podmínkami, které určují, jaké vlastnosti musí informace na stránce splňovat, aby filtru vyhovovala. Je nutné splnit všechny podmínky, které filtr obsahuje. Přidání podmínky tak množinu odpovídajících informací zužuje. Pomocí podmínek lze specifikovat, který element musí a nesmí cílovému elementu předcházet nebo ho následovat, které elementy musí či nesmí obsahovat a nebo které atributy musí být u elementu přítomny. Lze také omezit rozsah nalezených elementů, například na první nalezený.

Při extrakci jsou na stránce nalezeny instance vytvořených vzorů, které jsou uloženy jako XML soubor. Každý vzor je tvořen elementem pojmenovaným podle názvu vzoru, jeho obsah pak tvoří elementy stránky, které mu odpovídají.

Pro extrakci jsou dostupné dva mechanismy. První je založen na stromech. Elementy popsány pomocí cesty ve stromě stránky a případně vlastnostmi elementů. Příklad takové cesty může být `*:table:*:tr`. Hvězdička nahrazuje jeden nebo více elementů, pokud na nich nezáleží.

Druhým případem je mechanismus založený na řetězcích. Pracuje se s textem obsaženým v elementu, případně s HTML kódem, pokud obsahuje další zanořené elementy. Obsah elementu se popisuje pomocí regulárního výrazu.

Vytvoření vzoru

Pro vytváření vzorů nástroj poskytuje opět interaktivní rozhraní. Uživatel zadá jméno vzoru a označí pomocí myši část stránky (element) s požadovanou informací. Je vygenerován základní filtr popisující daný element. Zároveň jsou na stránce zvýrazněny elementy, které filtru odpovídají. Pokud vybrané elementy neodpovídají uživatelově představě, může k filtru přidat další podmínky a tím výběr omezit, nebo naopak přidat filtr, což výběr rozšíří.

Interní reprezentace

Stránka je reprezentována stromem elementů. Každý element je určen množinou atributů, které ho popisují. Kromě atributů přítomných ve zdrojovém kódu stránky obsahuje množina ještě speciální atribut `name` pro název elementu a `elementtext` obsahující text daného elementu, případně HTML kód v případě existence dalších zanořených elementů.

Každý vytvořený filtr je interně reprezentován pravidly (predikáty) popsanými pomocí speciálního deklarativního jazyka *Elog*. Tyto pravidla definují elementy extrahované ze stránky. Uživatel tomuto jazyku však nemusí porozumět, jelikož s ním nepřichází do styku. Vždy si vystačí s interaktivním uživatelským rozhraním.

3.4.2 Extrakce dat z dynamických WWW stránek

Další nástroj pro extrakci informací z webových stránek vznikl jako diplomová práce [13]. Poskytuje nejen možnost extrahovat informace z webové stránky, ale umožňuje provést i sekvenci několika kroků pro její získání. Aplikace je implementována jako webová a při vytváření popisu navigačních kroků a dat pro extrakci využívá Java appletu.

Získání požadované stránky

Pro popis navigace na požadovanou stránku aplikace využívá běžného webového prohlížeče v kombinaci s Java appletem. Aktuální stránka je upravena, protože pro korektní zobrazení v prohlížeči je nutné všechny relativní odkazy nahradit absolutními. Dále je doplněn kód v jazyce JavaScript, který zajistí, že jsou požadavky odesílány skrze aplikaci. Může si tak zaznamenat provedený krok, například na který odkaz bylo kliknuto nebo který formulář a s jakými daty byl odeslán. Požadavek je pak předán původnímu serveru. V okně appletu je zobrazen seznam zaznamenaných kroků.

Prohlížeč se ale využívá jen pro definici navigace, při samotné extrakci informací jsou navigační kroky reprodukovány pomocí knihovny pro HTTP komunikaci. To s sebou ovšem přináší nevýhodu, že provedení zaznamenaných kroků nemusí odpovídat jejich provedení v prohlížeči.

Definice extrahovaných dat

Pro určení extrahovaných dat aplikace používá určitou šablonu v podobě podstromu, který se na stránce vyhledává. Pro vytvoření šablony se opět využívá Java applet v kombinaci s prohlížečem. Na stránce se klikne na požadovaný element a díky doplněnému kódu v JavaScriptu lze v appletu toto kliknutí detekovat. Applet zobrazuje strom elementů stránky, přičemž se vybere ten, na který bylo na stránce kliknuto. Strom lze upravovat odebíráním nepodstatných elementů, specifikováním atributů, které musí elementy obsahovat, a určením obsahu textových uzlů. Textové uzly lze také označit jako proměnné. Jejich obsah je pak při nalezení šablony ve stránce exportován do XML souboru.

3.4.3 Online analýza dat z webových stránek

Nástroj pro online analýzu vybraných dat z webových stránek vznikl jako součást bakalářské práce [14]. Aplikace umožňuje přihlášení do privátní části stránek a práci s daty obsaženými v tabulkách nacházejících se ve stránce. Stránka je načtena na základě URL adresy. Uživatel může v její struktuře vybrat tabulky, které budou sledovány. Nad získanými daty je možné provádět aritmetické operace. Aplikace však neumožňuje dostat se na stránku postupnou navigací, například pomocí následování odkazu nebo odeslání formuláře.

Podobnou funkcionalitu nabízí uživateli i *Aplikace pro získávání a zpracování dat z tabulek na webových stránkách* [15], která vznikla rovněž jako bakalářská práce.

3.5 Shrnutí

Tato kapitola se zabývala záležitostmi souvisejícími s navigací na webových stránkách. Kromě popisu různých typů přechodů mezi stránkami byly popsány i možnosti autentizace pro přístup do privátních částí stránek a udržování kontextu po přihlášení. Byly také rozebrány principy využívané existujícími nástroji. Ty jsou však zaměřeny spíše na extrakci informací ze stránky než na navigaci na webu. Neumožňují například provádět opakovanou navigaci na základě vstupních dat nebo využít data nalezená na stránce pro další navigační kroky. První dva uvedené nástroje také příliš neřeší přihlašování uživatele. Tyto nedostatky se snaží vyřešit mechanismus, jehož návrh popisuje následující kapitola.

Kapitola 4

Návrh mechanismu pro navigaci

V této kapitole je popsán návrh mechanismu pro navigaci na webových stránkách. Nejprve jsou uvedeny požadavky na funkcionalitu, kterou by měl mechanismus umožňovat a návrh jejich řešení. Následuje návrh skriptovacího jazyka pro popis navigace. Je zde také uveden návrh vizuálního popisu navigace.

4.1 Požadavky na funkcionalitu

Mechanismus umožňuje jednak navigaci mezi stránkami a přihlašování do privátních částí, ale také disponuje prostředky pro kontrolu obsahu stránek, možnostmi dávkového provádění navigačních kroků a extrakcí dat ze stránky. V některých případech je nastíněno, jakým způsobem se budou řešit.

4.1.1 Navigace

Navigace se skládá z jednotlivých kroků. Může jít o načtení stránky na základě adresy, následování odkazu na aktuální stránce nebo vyplnění a odeslání formuláře.

Načtení stránky

Prvním krokem při navigaci je načtení webové stránky na základě URL adresy. Nemusí to být ale jediný případ. Lze toho využít i pokud je potřeba přejít na stránku, která s tou aktuální třeba příliš nesouvisí, a není možné se na ni dostat pomocí odkazu nebo jiným způsobem.

Navigace pomocí odkazů

Práce s odkazy je jedním ze základních požadavků. Odkaz může být specifikován textem, který obsahuje. Lze ho zadat přímo nebo popsat regulárním výrazem. Odkaz lze také určit pomocí XPath výrazu. Pokud specifikaci odpovídá více odkazů, lze je rozlišit zadáním pořadí.

Navigace s využitím formulářů

Na webové stránce se může nacházet více formulářů, je proto nutné určit, se kterým se bude pracovat. Formulář lze identifikovat více způsoby. Nejjednodušší je jeho pořadí na stránce.

Formuláře často obsahují atribut `name` nebo `id`, takže je lze identifikovat na základě hodnoty těchto atributů. Poslední možností je jeho specifikace pomocí XPath výrazu.

Formulář obsahuje různé prvky, které je před odesláním možné vyplnit. Uvažovány jsou všechny typy textových polí a prvky typu `radio`, `checkbox`, `select` a `file`. Pro textová pole je možné nastavit vložený text, pro prvek `select` vybrat požadovanou hodnotu ze seznamu, u prvků `radio` a `checkbox` nastavit jejich zatrhnutí a v případě prvku typu `file` specifikovat odesílaný soubor.

Prvek, se kterým se bude pracovat, lze určit podle hodnoty atributu `name`. Tento atribut musí být vždy uveden, protože je nutný pro odeslání formulářových dat na server. Identifikace je možná také pomocí atributu `id`, XPath výrazu v rámci formuláře nebo pořadím ve formuláři.

Odeslání formuláře se provede některým z tlačítek typu `submit`. Formulář jich může obsahovat více, proto je nutné jedno z nich určit. Opět lze využít atributu `name`, `id`, XPath výrazu nebo pořadí.

4.1.2 Přihlašování do privátních částí stránek

Způsoby přihlašování do privátních částí stránek byly popsány v části 3.2. Navržený mechanismus podporuje HTTP autentizaci a přihlašování pomocí formulářů.

V případě HTTP autentizace se přihlášení provede specifikováním jména a hesla, které se bude připojovat ke každému následujícímu požadavku.

U formulářového přihlašování se zadání přihlašovacích údajů provede stejným způsobem, jakým se pracuje s jakýmkoliv jiným formulářem. Pro zjednodušení přihlášení ale mechanismus poskytuje funkcionalitu, díky níž se provede přihlášení pouze na základě zadaného jména a hesla. Zde je využito toho, že přihlašovací formulář má ve většině případů stejnou strukturu, která zahrnuje textové pole pro jméno, pole pro zadání hesla a tlačítko pro potvrzení. Takovýto formulář je na stránce automaticky nalezen. Provede se vyplnění zadaných údajů a jeho odeslání.

Přihlašování s využitím Java appletů či flashových objektů vložených do stránky není z důvodu naprosté volnosti a nestandardnosti řešení podporováno. Pro jejich zpracování by byla nutná podpora prohlížeče.

4.1.3 Zpracování odpovědí

Po načtení stránky, ať už zadáním URL, následováním odkazu či odesláním formuláře, je v některých případech potřeba zkontrolovat ji, zda obsahuje požadované informace. Na základě toho se může například rozhodnout o dalším postupu či ukončení navigace.

Mechanismus umožňuje pracovat s aktuální stránkou, tedy tou, která byla načtena v posledním kroku. Informace obsažené ve stránce lze zkontrolovat několika způsoby.

Obsah stránky lze porovnat s regulárním výrazem. Při dalším postupu se může využít informace o nalezení či nenalezení daného výrazu nebo pomocí něj nalézt požadovanou informaci a s tou dále pracovat.

Struktura některých stránek je ale značně komplikovaná, v tom případě může být potřebný regulární výraz složitý a nemusí být jednoduché ho sestavit. Proto je možné pomocí XPath výrazu specifikovat element, ve kterém se informace nachází a až na jeho obsah aplikovat regulární výraz. V některých případech postačí kontrola, zda stránka element daný XPath výrazem obsahuje. Může jít například o element s chybovým hlášením, který se ve stránce při správném provedení požadavku neobjeví.

Kromě kontroly obsahu stránky z ní lze ale i určité informace získat a dále je využít. Je to popsáno v následující části 4.1.4.

4.1.4 Extrakce informací ze stránky

Přestože extrakce dat z webových stránek není hlavním záměrem, navrhovaný mechanismus ji podporuje. To z toho důvodu, že může být velmi užitečné využít informace z načtené webové stránky buď pro další navigaci, nebo je uložit do souboru a poté externě použít pro další potřeby.

Extrahované informace lze specifikovat dvěma způsoby. První je pomocí regulárního výrazu, který se bude vyhledávat ve zdrojovém kódu webové stránky a umožní nalézt požadované informace. Druhý způsob využívá XPath výrazu pro výběr požadovaných uzlů z DOM modelu stránky.

4.1.5 Využití dávkového zpracování

Jedním z požadavků uživatele může být to, že chce provést opakovaně nějakou aktivitu na webu, přičemž každé opakování se liší jen vstupními daty. Mechanismus pro tyto účely poskytuje možnost specifikovat kolekci dat a sekvenci navigačních kroků, která se provádí opakovaně a v každé iteraci se využijí vstupní data dané jedním prvkem z kolekce. Data je možné do kolekce načíst ze souboru. Druhou možností jejího naplnění je využití dat obsažených na webové stránce.

V případě dávkového zpracování je užitečné, pokud lze během jednotlivých iterací sbírat určité informace, například o tom, jestli odeslání formuláře s danými daty proběhlo v pořádku. Mechanismus tuto možnost poskytuje a informaci je možné uložit pod zadaným jménem. Po dokončení dávky je možné s takto ukládanými daty dále pracovat.

4.2 Návrh skriptovacího jazyka

Tato část se zabývá návrhem jazyka, pomocí kterého je navigace popsána. Při návrhu připadaly v úvahu dvě možnosti. Navigační jazyk je možné vytvářet kompletně od začátku. Tato možnost by ale vyžadovala mnoho úsilí, jelikož by bylo pro vytvořený jazyk nutné implementovat vlastní parser a interpret.

Vzhledem k tomu, že navigační jazyk bude disponovat běžnými řídicími konstrukcemi a prací s proměnnými, je velmi výhodné postavit ho na nějakém již existujícím skriptovacím jazyce. Jedním z požadavků pro takový jazyk je, aby existoval parser a interpret umožňující provádět jeho kód v rámci vytvářené aplikace. Dalším požadavkem je, aby ho bylo možné rozšířit o další funkcionalitu.

Oba tyto požadavky bez výjimky splňuje projekt Jython. Jde o interpret pro jazyk Python implementovaný v jazyce Java. V rámci aplikace lze skripty spouštět a díky možnosti provázání s Javou lze jazyk rozšířit o další funkce.

Díky využití Jythonu je možné soustředit větší pozornost na samotnou navigaci a na vytvoření efektivního způsobu pro její popis.

4.2.1 Základní charakteristika

Navigační jazyk poskytuje běžné řídicí struktury, jako jsou podmíněné příkazy a cykly. Pro popis aktivit prováděných během navigace poskytuje řadu vestavěných funkcí, například pro práci s odkazy, formuláři, kontrolu odpovědí apod.

Umožňuje definovat proměnné, které mohou mimo ukládání např. čísel a řetězců sloužit i k uložení požadovaných informací získaných ze stránek. Následně je lze použít pro export nebo další navigaci, například jako parametry vestavěných funkcí nebo pro rozhodování o dalším kroku.

Všechny funkce pracující s informacemi z webové stránky pro jejich získání využívají tu, která je právě načtena. Je to takto realizováno z toho důvodu, že pro provedení dalšího kroku tyto informace postačují a není tudíž potřeba je získávat ze stránek, které byly načteny dříve. Nic však nebrání postupnému sběru informací na různých stránkách a jejich následnému exportu.

4.2.2 Popis navigace

Jak již bylo nastíněno, jazyk poskytuje množství vestavěných funkcí. V této části jsou popsány ty, které se týkají provádění navigačních kroků.

Načtení stránky

Pro načtení stránky slouží funkce, které se předá URL adresa požadované stránky.

```
loadPage("URL")
```

Následování odkazu

V části 4.1.1 byly popsány způsoby specifikace odkazu, odpovídají jim funkce

```
clickLinkByText("text odkazu")
clickLinkByRegex("reg. výraz")
clickLinkByXPath("XPath výraz")
```

Při následování odkazu daného textem se ignoruje velikost písmem i bílé znaky před a za textem. Ve druhém, resp. třetím případě se následuje odkaz s textem odpovídajícím zadanému regulárnímu výrazu, resp. specifikovaný zadaným XPath výrazem.

V případě nalezení více odpovídajících odkazů se použije první nalezený. Existují ale i varianty těchto funkcí, které mají jako druhý parametr číslo udávající, který z nalezených odkazů se použije. Všechny funkce vracejí booleovskou hodnotu indikující, zda byl odkaz nalezen a přechod na příslušnou stránku proveden, či nikoliv.

Práce s formuláři

Nejprve je nutné určit formulář, se kterým se dále bude pracovat. K tomu slouží níže uvedené funkce. Formulář lze vybrat podle hodnoty atributu `name` či `id`, pořadí na stránce nebo XPath výrazu. Všechny funkce vracejí booleovskou hodnotu informující o nalezení či nenalezení formuláře.

```
selectFormByName("jméno")
selectFormById("id")
selectFormByXPath("xpath výraz")
selectFormByPosition(poradi_na_strance)
```

Po vybrání formuláře je možné nastavit hodnoty prvků, které obsahuje. U všech typů prvků je nutné určit, se kterým se bude pracovat, jelikož se jich ve formuláři může vyskytovat více. Prvek lze dle použité varianty funkce specifikovat na základě jeho atributu `name`, `id`, XPath výrazu v kontextu elementu formuláře nebo pořadím ve formuláři.

Následující funkce slouží pro vyplnění textových polí. Nastavený text je dán druhým parametrem.

```
setInputByName("jméno", "text")
setInputById("id", "text")
setInputByXPath("xpath výraz", "text")
setInputByPosition(pozice, "text")
```

Pro práci s prvkem `select` jsou určeny níže uvedené funkce. Druhý parametr určuje výběr požadované volby na základě hodnoty jejího atributu `value` nebo textu, který obsahuje. Pokud je druhým parametrem funkce číslo namísto řetězce, provede se nastavení položky s odpovídajícím indexem v seznamu, indexuje se od 0.

```
setSelectByName("jméno", "vybraná volba")
setSelectById("id", "vybraná volba")
setSelectByXPath("xpath výraz", "vybraná volba")
setSelectByPosition(pozice, "vybraná volba")
```

Při práci s prvkem typu `checkbox` je jeho nastavení dáno booleovskou hodnotou předanou jako druhý parametr následujícím funkcím, `True` pro zaškrtnutí, `False` pro odškrtnutí.

```
setCheckboxByName("jméno", zaskrtnuti)
setCheckboxById("id", zaskrtnuti)
setCheckboxByXPath("xpath výraz", zaskrtnuti)
setCheckboxByXPosition(pozice, zaskrtnuti)
```

V případě prvku typu `radio` je situace s jeho určením trochu jiná. Formulář totiž typicky obsahuje více prvků se stejným atributem `name`, z nichž může být zatrhnut vždy maximálně jeden. Pro rozlišení konkrétního je třeba specifikovat ještě hodnotu atributu `value`. Při určení prvku s využitím atributu `id`, XPath výrazu nebo pozice je situace stejná jako u ostatních prvků. Prvek je tak v rámci formuláře jednoznačně identifikován. Následující funkce provedou zatrhnutí určeného prvku. Ostatní prvky se stejným jménem jsou automaticky odškrtnuty.

```
setSelectedRadioByName("jméno", "hodnota atributu value")
setSelectedRadioById("id")
setSelectedRadioByXPath("xpath výraz")
setSelectedRadioByPosition(pozice)
```

Pro odeslání souboru v rámci formuláře slouží následující funkce. Druhý parametr určuje soubor, který bude odeslán. Umístění souboru může být relativní vzhledem k adresáři projektu, v rámci nějž se skript vytváří, nebo absolutní.

```
setFileInputByName("jméno", "soubor")
setFileInputById("id", "soubor")
setFileInputByXPath("xpath výraz", "soubor")
setFileInputByPosition(pozice, "soubor")
```

Po nastavení všech položek formuláře se provede jeho odeslání. K tomu slouží následující funkce umožňující vybrat požadované tlačítko.

```
submitFormByBtnName("jmeno")
submitFormByBtnId("id")
submitFormByBtnXPath("xpath výraz")
submitFormByBtnPosition(pozice)
```

Přihlašování a odhlašování

Pro přihlášení při použití HTTP autentizace se nastaví jméno a heslo, které se bude používat s každým dalším požadavkem. Slouží k tomu funkce

```
setCredentials("jmeno", "heslo")
```

Pro přihlášení pomocí formuláře lze využít funkci `login`, které se předají přihlašovací údaje. Na stránce je nalezen formulář typický pro přihlášení, tedy takový, který obsahuje pole typu `text` pro jméno, `password` pro heslo a potvrzovací tlačítko typu `submit`. V případě, že stránka obsahuje více takových formulářů, použije se první nalezený. Funkci lze ale i předat jako třetí parametr číslo určující index formuláře na stránce, který se použije. Čísluje se od 0. Pro přihlášení lze ale využít i funkce pro manipulaci s formuláři. Je to nutné v případě, že formulář obsahuje ještě nějaké další prvky, které je potřeba nastavit, například prvek typu `checkbox` nebo `select`.

```
login("jmeno", "heslo")
```

Odhlašování je na každé stránce jiné, proto není k dispozici obecná funkce. Je třeba využít funkcí pro manipulaci s odkazy nebo formuláři v případě tlačítka.

Zpracování odpovědí

Pro kontrolu obsahu stránky, například pro ověření úspěšnosti provedeného požadavku, slouží funkce

```
checkPageContent("regulární výraz")
```

Provede vyhledání zadaného regulárního výrazu na aktuálně načtené stránce a vrátí booleovskou hodnotu udávající, zda byl nalezen či nikoliv.

Kontrolu je možné provést i s využitím testu na přítomnost určitého elementu na stránce. Element je specifikován XPath výrazem. Tímto je například možné ověřit, zda stránka obsahuje prvek s chybovou hláškou, která se může vypsat například po odeslání formuláře s chybnými údaji.

```
checkElementOccurence("xpath")
```

V některých případech může být užitečný nejen test na výskyt elementu, ale i na jeho obsah. K tomu slouží následující funkce.

```
checkElementContent("xpath", "regulární výraz")
```

Předá se jí XPath výraz identifikující element a regulární výraz popisující jeho obsah. Vrací booleovskou hodnotu udávající, zda byl element nalezen a jeho obsah odpovídá zadanému výrazu. Pokud XPath výraz určuje více elementů, musí zadanému regulárnímu výrazu odpovídat obsahy všech.

Ladění

Pro výpis ladících informací lze využít níže uvedené funkce.

```
println(informace)
printlnErr(informace)
```

Druhá uvedená slouží pro výpis chybových hlášení. Funkce akceptují řetězce nebo proměnné, například elementy extrahované ze stránky. V závislosti na typu předané hodnoty je vytvořena její textová reprezentace a vypsána.

Pokud je potřeba skript z ladících důvodů ukončit dříve, než po provedení všech zadaných příkazů, lze použít funkci

```
exit()
```

4.2.3 Způsoby získání dat ze stránky

Při extrakci je nutné specifikovat, které části stránky se použijí. Mechanismus poskytuje dva možné přístupy pro jejich určení, pomocí regulárních výrazů nad zdrojovým kódem stránky nebo XPath výrazu pro určení požadovaných částí stromu elementů. V obou případech se jako zdroj dat použije aktuálně načtená stránka. Výsledek extrakce je možné použít pro další navigaci nebo ho uložit do souboru. Podrobně je to popsáno v části 4.2.5.

Regulární výrazy

S využitím regulárního výrazu lze získat fragment textu webové stránky, který mu odpovídá. A to nejen text odpovídající celému výrazu, ale i jeho částem. To v případě, že se požadovaná část výrazu uzavře do kulatých závorek. Podle pořadí ozávorkované části lze po nalezení výskytu získat jí odpovídající text. Takto lze označit libovolné množství částí, které jsou číslovány od jedničky. Pořadí nula odpovídá výskytu celého výrazu. Vyhledání výrazu lze na stránce opakovat, dokud jsou nacházeny jeho výskyty.

Této vlastnosti lze využít pro získání potřebných dat z webové stránky. Výsledek vyhledání je reprezentován seznamem výskytů regulárního výrazu, z nichž každý může obsahovat ještě nalezené texty odpovídající jeho částem. Výsledek lze reprezentovat jako dvourozměrné pole, kde první dimenze určuje nalezený výskyt výrazu a druhá dimenze texty odpovídající jednotlivým částem výrazu.

Vyhledání informací výše uvedeným způsobem je možné provést pomocí funkce

```
extractAllByRegex("regularni vyraz")
```

vracející dvourozměrné pole s výsledky. Pokud je potřeba vyhledat na stránce jednu konkrétní informaci, je pro jednodušší zpracování dostupná funkce

```
extractTextByRegex("regularni vyraz")
```

vracející nalezený text odpovídající prvnímu výskytu celého regulárního výrazu. Volitelně lze funkci jako druhý parametr předat číslo udávající pořadí části výrazu uzavřené v závorkách, jíž odpovídající text funkce vrátí místo textu odpovídajícímu výskytu celého regulárního výrazu.

XPath výrazy

Výsledkem dotazu tvořeného XPath výrazem je seznam uzlů, které mu odpovídají. Ve většině případů jde o elementy. Dotazem lze ale získat i seznam atributů nebo textových uzlů.

Pro extrakci informací tímto způsobem slouží funkce

```
extractAllByXPath("xpath výraz")
```

vracející seznam uzlů odpovídajících zadanému výrazu. V případě, že je vhodné získat text obsažený v jednom konkrétním elementu, lze využít funkci

```
extractTextByXPath("xpath výraz")
```

Obě uvedené funkce extrahují data z aktuálně načtené stránky. XPath dotaz lze však provádět nejen nad celou stránkou, ale i nad elementem, který byl získán pomocí extrakce. Slouží k tomu následující funkce.

```
queryAllByXPath(element, "xpath výraz")
```

Funkce provede vyhodnocení XPath výrazu v kontextu elementu, který je předán jako první parametr. Podobným způsobem lze získat text obsažený v elementu daném XPath výrazem. Je k tomu určena funkce

```
queryTextByXPath(element, "xpath výraz")
```

Pro získání textu obsaženého v elementu slouží funkce

```
getElementText(element)
```

4.2.4 Sběr dat během navigace

Během vykonávání navigačních kroků může být potřeba si postupně ukládat vybraná data z navštívených stránek. Je to užitečné zejména v případě dávkového provádění navigace na základě vybraných dat. Může například jít o opakované odesílání formuláře. Po každém odeslání formuláře může být vhodné uložit si výsledek.

Pro uvedené účely je k dispozici funkce

```
saveData("jméno", data)
```

která umožňuje získaným datům přiřadit jméno a uložit je. Později je kdykoliv možné k nim přistoupit nebo provést export všech takto uložených dat do XML souboru. Ukládaná data mohou mít různý typ. Lze uložit text, vybraný uzel i seznam uzlů získaných ze stránky. Příklad, jak při exportu vypadá textová reprezentace uložených dat, je v části 4.2.5.

K uloženým datům lze přistoupit na základě jména pomocí funkce

```
getSavedData("jméno")
```

vracející typ, který byl pod daným jménem uložen.

Pokud není třeba všechna ukládaná data exportovat, je samozřejmě možné ukládat je do běžných proměnných.

4.2.5 Využití získaných dat

Informace získané při extrakci lze využít pro další navigaci, například jako parametr pro některou z funkcí nebo pro jejich export.

Další navigace

Lze využít informaci v podobě textového řetězce. Je možné ji získat například zavoláním funkce `extractTextByRegex` či `extractTextByXPath` a uložením navráceného výsledku do proměnné. Poté ji lze použít jako parametr některé z funkcí pro navigaci. Lze ji také předat funkci `saveData` pro uložení dat.

Ze stránky lze získat i kolekci informací. V případě použití funkce `extractAllByRegex` jsou informace dostupné jako dvourozměrné pole obsahující položky reprezentující jednotlivé výskyty vyhledávaného regulárního výrazu. K položkám lze přistupovat pomocí jejich indexu, stejným způsobem k jednotlivým částem v rámci položky, viz ukázka 4.1.

```
#načtení stránky s tabulkou osob
loadPage("www.stud.fit.vutbr.cz/~xklime03/dip/persons.php")
regex = "<tr>\\s*<td>(.*?)</td>\\s*<td>(.*?)</td>"
vysledky = extractAllByRegex(regex)
vysledek3 = vysledky[2]          # 3. nalezený výskyt
celyVyskyt = vysledek3[0]        # 3. výskyt celého reg. výrazu
cast1 = vysledek3[1]             # první položka 3. výskytu
```

Ukázka 4.1: Přístup k datům získaným pomocí regulárního výrazu

Informace získané výše uvedeným způsobem lze opět využít jako parametry různých funkcí zajišťujících provedení navigačních kroků. Pokud je potřeba zpracovat všechny položky kolekce, je možné přes ně iterovat s využitím cyklu, podobným způsobem jako u uzlů v ukázce 4.2.

Další možností je získat ze stránky pomocí XPath výrazu seznam uzlů, které obsahují informace využitelné pro další navigaci. Každý uzel může například obsahovat data pro jedno odeslání formuláře. Zpracování dat je tedy možné provést v cyklu. V jeho těle se provede požadovaná navigační aktivita, třeba načtení stránky a odeslání formuláře. Data pro jednotlivé položky formuláře lze vybrat XPath výrazem v kontextu uzlu v aktuální iteraci. Příklad možného zápisu je v ukázce 4.2. Zdrojový kód využité stránky je uveden v příloze B.

```
# načtení stránky s tabulkou osob se jménem a příjmením
# v prvním a druhém sloupci
loadPage("www.stud.fit.vutbr.cz/~xklime03/dip/persons.php")
radky = extractAllByXPath("//tbody/tr") # extrakce řádků tabulky
for radek in radky:       # pro všechny vybrané uzly
  loadPage("www.stud.fit.vutbr.cz/~xklime03/dip/persons_add.php")
  selectFormById("addPerson") # výběr formuláře

  jmeno = queryTextByXPath(radek,"./td[1]") # získání jména
  setInputByName("name", jmeno)      # nastavení jména
  prijmeni = queryTextByXPath(radek,"./td[2]") # získání příjmení
  setInputByName("surname", prijmeni)      # nastavení příjmení

  submitFormByBtnName("addPerson")         # odeslání formuláře
```

Ukázka 4.2: Příklad zápisu dávkového zpracování seznamu XML uzlů

Nic nebrání v extrakci pouze jednoho uzlu a provedení pouze jedné navigační aktivity na nákladě jeho dat. K položkám kolekce je možné přistoupit přes index, např. `radky[0]`.

Export dat

Při exportu dat závisí výstup na jejich typu. Výsledkem je ale vždy XML soubor. Pro export slouží níže uvedená funkce.

```
exportData(data, "soubor")
```

V případě dat získaných pomocí regulárních výrazů je každý nalezený výskyt reprezentován elementem `result`, který obsahuje elementy `group` představující nalezené části regulárního výrazu v daném výskytu. Možný výsledný výstup je zobrazen v ukázce 4.3.

```
<output>
  <result id="0">
    <group id="0">1. výskyt celého reg. výrazu</group>
    <group id="1">výskyt 1. části reg. výrazu</group>
    <group id="2">výskyt 2. části reg. výrazu</group>
    ...
  </result>
  ...
</output>
```

Ukázka 4.3: XML výstup exportu dat získaných pomocí regulárního výrazu

Při exportu uzlů získaných pomocí XPath výrazu je výsledkem jejich XML reprezentace obalená elementem `output`.

Další možností je export všech dat postupně ukládaných s využitím funkce `saveData`. Slouží k tomu následující funkce.

```
exportAllSavedData("soubor").
```

Odpovídající XML výstup je v ukázce 4.4. Element `item` s atributem `name` odpovídá jedněm datům uloženým pod daným jménem. Tento element obsahuje reprezentaci uložených dat. Může to být prostý text, v případě dat získaných pomocí regulárního výrazu elementy `result` z příkladu v ukázce 4.3 nebo elementy získané pomocí XPath výrazu.

```
<output>
  <item name="jméno">
    reprezentace uložených dat
  </item>
  ...
<output>
```

Ukázka 4.4: XML výstup exportu postupně ukládaných dat

Lze exportovat i celou webovou stránku. Aktuálně načtená stránka se uloží do souboru pomocí funkce

```
exportActualPage("soubor").
```

Všem funkcím pro export lze zadat i pouze název souboru bez cesty. V tom případě se uložení souboru provede do složky, ve které se nachází projekt, v rámci něhož je skript vytvářen.

4.2.6 Dávkové zpracování dat ze vstupních souborů

Data lze načíst i ze vstupních souborů uložených na disku. Mezi podporované formáty patří XML a CSV. Hlavním účelem je využití dat, která se z nich získají, pro další navigaci.

Při práci se XML souborem se po jeho načtení pomocí XPath výrazu vybere seznam uzlů s daty, která se použijí pro navigaci. S takto získanými uzly lze dále pracovat stejným způsobem, jako s uzly získanými ze stránky, jak je ukázáno v ukázce 4.2. K načtení souboru a vybrání požadovaných uzlů na základě XPath výrazu slouží funkce

```
importXml("soubor", "xpath")
```

Při práci s CSV souborem reprezentuje každý jeho řádek data pro provedení jedné navigační aktivity. Jednotlivé řádky lze v cyklu postupně zpracovávat a v každé iteraci provést požadovanou činnost. Při otevírání souboru se jako druhý parametr specifikuje, zda se má první řádek považovat za záhlaví. Třetí parametr určuje kódování souboru. K datům na řádcích lze přistupovat na základě čísla sloupce a pokud soubor obsahuje záhlaví, tak i podle jména sloupce. V ukázce 4.5 je příklad zpracování dat z CSV souboru a provádění odesílání formuláře na základě nich. Zdrojový text využité stránky lze nalézt v příloze B.

```
loadPage("www.stud.fit.vutbr.cz/~xklime03/dip/persons.php")
radky = importCsv("osoby.csv", True, "utf-8") # načtení souboru
for radek in radky:
  loadPage("www.stud.fit.vutbr.cz/~xklime03/dip/persons_add.php")
  selectFormByName("addPerson")   # vybrání formuláře

  jmeno = csvGetColByPosition(radek, 0) # dle čísla sloupce
  prijmeni = csvGetColByName(radek, "prijmeni") # dle jména sl.
  setInputByName("name", jmeno) #vyplnění jména
  setInputByName("surname", prijmeni) # vyplnění příjmení

  submitFormByBtnName("addPerson")    #odeslání
```

Ukázka 4.5: Příklad dávkového zpracování dat z CSV souboru

4.2.7 Uživatelské funkce a proměnné

Pro aktivity, které se při popisu skriptu vícekrát opakují, je možné definovat uživatelské funkce. Příklad definice a volání uživatelské funkce v navrženém skriptovacím jazyce je v ukázce 4.6.

```
def uzivatelska_funkce():
    println("text")

uzivatelska_funkce()

promenna = u"text obsahující diakritiku"
```

Ukázka 4.6: Definice a volání uživatelské funkce, uložení textu s diakritikou do proměnné

Pro ukládání informací lze využít proměnné. Lze do nich ukládat např. čísla a řetězce. Pro uložení řetězce, který obsahuje diakritiku, je nutné před počáteční uvozovku napsat písmeno `u`, aby byl zpracován jako unicode. Příklad definice takovéto proměnné je v ukázce 4.6.

Do proměnných lze však ukládat také složitější datové typy. Jsou jimi například pole s výsledky v případě extrakce informací pomocí regulárních výrazů, seznam elementů při extrakci informací pomocí XPath výrazu nebo seznam řádků při importu CSV souboru. Příklady ukládání složitějších datových typů do proměnných je možné vidět v ukázkách 4.1, 4.2 či 4.5.

4.3 Interaktivní popis navigace

Navigace se skládá z postupného provádění navigačních kroků. Je možné provádět i větvení na základě obsahu stránky nebo iterativní provádění sekvence kroků na základě dat ze stránky či vstupního souboru.

Sled jednotlivých kroků navigace lze popsat pomocí grafu s uzly spojenými orientovanými hranami. Každý uzel představuje provedení nějakého kroku navigace, který je dán typem uzlu. Může jít o načtení stránky, následování odkazu, vyplnění a odeslání formuláře, kontrolu obsahu stránky, iterování přes kolekci získaných informací a podobně.

Hrany reprezentují následnost prováděných kroků. Pokud jsou dva uzly propojeny hranou, znamená to, že po provedení kroku daného výchozím uzlem se provede krok daný cílovým uzlem hrany.

Graf lze interaktivně vytvářet s využitím dostupných typů uzlů a jejich propojování hranami. Každý uzel lze konfigurovat pomocí dialogu, který v závislosti na typu uzlu nabízí různé možnosti nastavení. V případě načtení stránky lze vybrat její URL, při odesílání formuláře vybrat ten, který se použije a vyplnit jeho hodnoty, při extrakci například definovat data, která se extrahují.

Data potřebná pro konfiguraci uzlu však mohou být získána až v průběhu provádění. Proto je možné spustit navigaci pouze po požadovaný uzel, který je potřeba nastavit. Tímto se získá požadovaná stránka a data z ní lze využít pro dokončení konfigurace uzlu.

Pro spuštění vizuálně popsané navigační aktivity je nejprve vygenerován odpovídající zápis ve skriptovacím jazyce a ten je následně interpretován. Vygenerovaný zápis je možné ručně upravit. Provedené změny však nelze reflektovat zpět do vizuální reprezentace. Je to tak z toho důvodu, že nelze určit, které části skriptu odpovídají kterým uzlům vizuálního popisu.

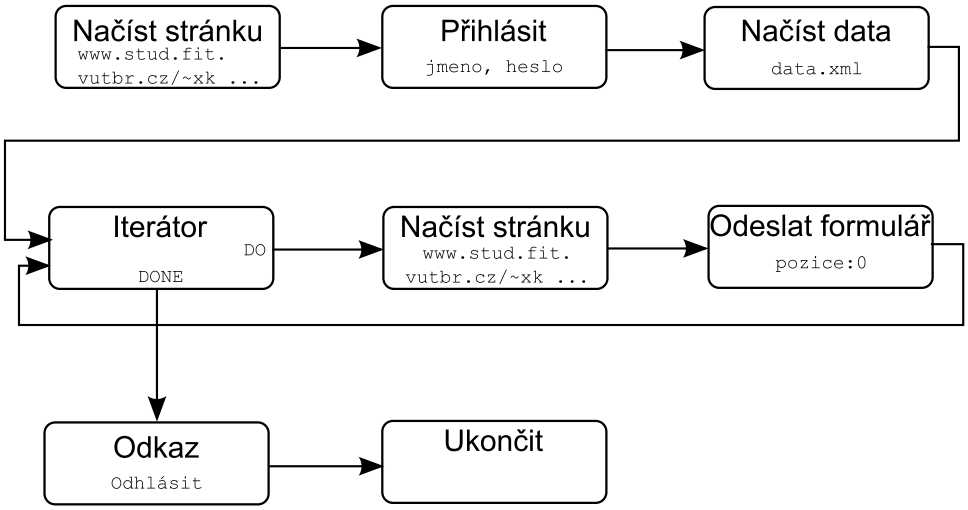
Příklad zápisu navigace ve skriptovacím jazyce je v ukázce 4.7 a jemu odpovídající vizuální popis je uveden na obrázku 4.1.

```
loadPage("www.stud.fit.vutbr.cz/~xklime03/dip/persons.php")
login("jmeno","heslo")

osoby = loadXml("data.xml", "/osoby/osoba") #načtení xml souboru
for osoba in osoby:                # pro všechny vybrané elementy
  loadPage("www.stranka.cz/persons_add.php") # načtení stránky
  selectFormByPosition(0) # výběr formuláře

  jmeno = queryTextByXPath(osoba,"./@name")  # text elementu
  setInputByName("name", jmeno)     # nastavení textového pole
  prijmeni = queryTextByXPath(osoba,"./@surname") #text elementu
  setInputByName("surname", prijmeni)  #nastavení textového pole

  submitFormByBtnName("addPerson")      # odeslání formuláře

clickLinkByText(u"Odhlásit")
```

Ukázka 4.7: Příklad popisu navigace pomocí skriptovacího jazyka

Je využito fiktivní stránky umožňující přidávání osob pomocí formuláře. Její zdrojový kód se nachází v příloze B. Navigační aktivita v ukázkách začíná načtením stránky na základě URL adresy, pokračuje přihlášením, kdy je automaticky nalezen formulář obsahující pole pro jméno a heslo. Následuje načtení externího XML souboru s daty a vybrání uzlů, jejichž data se použijí pro navigaci. Je iterováno přes všechny vybrané uzly. V každé iteraci se provede načtení stránky s formulářem, nastavení použití formuláře dle jeho pozice na stránce, vybrání dat z uzlu v aktuální iteraci, jejich vyplnění do pole ve formuláři a odeslání formuláře. Na konci je provedeno odhlášení kliknutím na odkaz obsahující text „Odhlásit“.

Při vytváření grafického popisu navigace je možné využít některou z předpřipravených šablon a její úpravou dosáhnout potřebné aktivity na webu.



Obrázek 4.1: Vizuální popis navigace odpovídající skriptu v ukázce 4.7

4.4 Shrnutí

V této kapitole byly popsány cíle, které by měl navigační mechanismus splňovat. Byl uveden návrh jazyka pro popis navigace, který umožňuje vytyčené cíle realizovat. Nakonec byl představen princip grafického znázornění navigačních kroků.

S popisem navigace ale souvisí i uživatelské rozhraní vytvářené aplikace a způsoby, pomocí kterých je možné vytváření popisu navigace usnadnit. Právě tímto se zabývá následující kapitola.

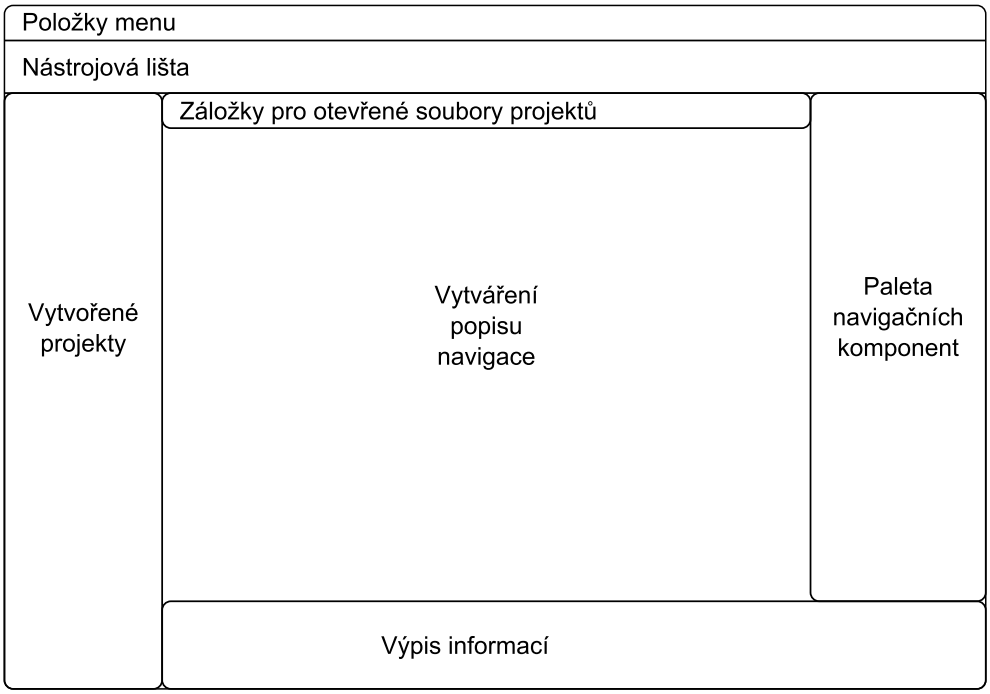
Kapitola 5

Návrh aplikace

Tato kapitola popisuje návrh uživatelského rozhraní aplikace a metody usnadňující vytváření popisu navigace, ať už pomocí zápisu příkazů navrženého skriptovacího jazyka nebo grafického znázornění.

5.1 Uživatelské rozhraní

Návrh uživatelského rozhraní je na obrázku 5.1. Rozložení prvků je typické pro běžné vývojové prostředí. V horní části se nachází položky menu a nástrojová lišta. Uživatel muže vytvořit více projektů, které jsou zobrazeny v seznamu v levé části. Otevřené soubory projektů lze přepínat pomocí záložek pod nástrojovou lištou.



Obrázek 5.1: Návrh uživatelského rozhraní aplikace

Střední část okna slouží pro vytváření popisu navigace. Lze ji popsat s využitím navrženého skriptovacího jazyka, jehož zdrojový kód lze vytvářet pomocí vestavěného editoru.

Druhou možností je využít vizuálního popisu navrženého v části 4.3. V tom případě se zobrazí pracovní plocha umožňující vytvoření grafu popisujícího navigační aktivitu. Lze využít komponenty dostupné v paletě v pravé části okna. Po jejich vložení na plátno je lze propojit hranami, které určují následnost kroků reprezentovaných uzly. Každý uzel lze konfigurovat pomocí dialogu, který je závislý na typu uzlu.

Ve střední části hlavního okna lze ještě zobrazit náhled poslední načtené stránky a její zdrojový kód. V pravé části okna lze kromě palety navigačních uzlů zobrazit i strom elementů stránky.

Ve spodní části okna je k dispozici konzole pro výpis různých informací, například varování o nenalezení požadovaného odkazu či formuláře nebo ladící výpisy vytvořené v rámci skriptu.

5.2 Usnadnění tvorby popisu navigace

Vytváření popisu navigace lze usnadnit několika způsoby. Je pro to ale potřeba mít k dispozici stránku, která bude před právě popisovaným krokem načtena. Při popisu pomocí skriptovacího jazyka lze využít funkce `exit()`, která ukončí provádění skriptu, ale zůstane zachována poslední načtená stránka. Při grafickém popisu navigace ji lze spustit pouze po určitý uzel, který předchází tomu právě konfigurovanému. Získá se tím opět stránka potřebná pro usnadnění popisu dalšího kroku. Provádění se ukončí uzlem, ze kterého nevede žádná hrana.

5.2.1 Zobrazení stromu stránky

Při vytváření XPath výrazu, ať už pro výběr odkazu, formuláře či extrahované informace, je užitečné zobrazení stromu elementů stránky. Velmi to usnadní orientaci v její struktuře a tím i tvorbu výrazu. Po vytvoření výrazu je možné zvýraznit ve stromu uzly, které jsou jím vybrány. Na základě výběru elementu lze také vygenerovat XPath výraz, který je poté možné upravit dle potřeb.

5.2.2 Zobrazení zdrojového kódu stránky

V případě výběru informací ze stránky na základě regulárního výrazu se vychází z jejího zdrojového kódu. Pro sestavení výrazu je nutné mít možnost si zdrojový kód zobrazit. Po vytvoření regulárního výrazu je možné aplikovat ho na stránku a zvýraznit v jejím zdrojovém kódu nalezené výskyty pro ověření, zda odpovídají požadovaným informacím.

Pokud je prováděna konfigurace uzlu, při níž se regulární výraz zadává, je po jeho vyplnění hned možné přepnout se do zobrazení zdrojového kódu a zadaný výraz nechat zvýraznit. Po kontrole jeho správnosti je možné přepnout se zpět ke konfiguraci uzlu.

5.2.3 Zobrazení načtené stránky

Pro ověření, zda vytvořený krok navigace proběhl správně a dle očekávání, lze zobrazit načtenou stránku podobným způsobem jako v prohlížeči. Zobrazenou stránku je možné využít také například pro vygenerování XPath výrazu pro výběr požadovaného formuláře. Při konfiguraci uzlu, v jehož kontextu se výraz zadává, je možné přepnout se do náhledu stránky. Kliknutím na požadovaný element je vygenerován jemu odpovídající XPath výraz, který je vyplněn do pole v dialogu pro konfiguraci uzlu. Následně ho lze upravit dle potřeb.

Další možností je zvýraznění elementů vybraných zadaným XPath výrazem. Elementy jsou v zobrazení stránky odlišeny barvou jejich pozadí a orámováním. Podobným způsobem lze zvýraznit také uzly označené v DOM stromu stránky.

Zobrazení načtené stránky lze také využít pro nalezení požadovaného elementu v DOM stromu. Pokud je v náhledu stránky pomocí pravého tlačítka myši kliknuto na element, je tento element vybrán v jejím DOM stromu.

5.2.4 Nápověda při tvorbě skriptu

Při psaní zdrojového kódu v navrženém jazyce pro navigaci jsou uživateli napovídány dostupné vestavěné funkce. Při zadávání jejich parametrů jsou napovídány různé možnosti na základě aktuálně načtené stránky. Například při výběru formuláře na základě atributu `name` jsou nabídnuty jeho hodnoty, které se u formulářů na stránce vyskytují. Podobně jsou hodnoty napovídány při práci s prvky formuláře. V tom případě je potřeba mít označen aktivní formulář, v rámci kterého se bude pracovat. K tomu je k dispozici vestavěná funkce. Je nutné, aby byla nejprve vykonána v rámci provádění skriptu. Poté je nápověda při práci s položkami formuláře dostupná.

Při vytváření XPath výrazu jsou na základě již napsané části nabídnuty možnosti, jak lze v jeho tvorbě pokračovat. Jedná se o jména elementů a atributů vyskytujících se v podstromu stránky vybraném na základě části výrazu, která byla prozatím vytvořena. Nápověda při vytváření XPath výrazu je však dostupná i při výběru uzlů z XML souboru.

5.2.5 Nápověda při vytváření vizuálního popisu

Nápověda různých možností na základě načtené stránky je dostupná i při vytváření vizuálního popisu navigace. Lze ji využít při konfiguraci jednotlivých uzlů pomocí dostupných dialogů. Nápověda je podrobněji popsána v následující kapitole 5.3, kde je rozebrán návrh zmíněných dialogů.

5.3 Návrh dialogů pro konfiguraci uzlů

V této části je popsán návrh dialogů pro konfiguraci různých uzlů při vytváření vizuálního popisu navigace. Dialog pro konfiguraci uzlu s formulářem je vzhledem k jeho větší složitosti popsán podrobněji.

5.3.1 Načtení stránky a následování odkazu

Při konfiguraci uzlu pro načtení stránky je pouze zadána URL adresa, z níž se stránka získá.

Pro následování odkazu lze vybrat, zda bude určen na základě textu odkazu, regulárního výrazu popisujícího text odkazu či dle XPath výrazu a je zadána příslušná hodnota. Při zadávání hodnoty je k dispozici nápověda možností na základě aktuálně načtené stránky. Lze zadat také pořadí odkazu v případě, že zadanému způsobu odpovídá více odkazů.

5.3.2 Přihlášení

Je nutné vybrat, zda se přihlášení provede s využitím HTTP protokolu nebo pomocí formuláře na stránce. Je zadáno uživatelské jméno a heslo.

5.3.3 Odeslání formuláře

V dialogu pro konfiguraci uzlu pro zpracování formuláře je nejprve nutné vybrat, s jakým formulářem se bude pracovat. Je to možné podle jeho atributu `name`, `id`, podle pozice ve stránce nebo XPath výrazu. Poté lze načíst prvky, které jsou ve formuláři obsaženy. Prvky jsou vypsány v tabulce. Každý řádek odpovídá jednomu prvku a je v něm uveden typ prvku, způsob jeho identifikace a vyplněná či vybraná hodnota. Prvky lze přidávat a odebírat i ručně a je možné nastavovat jejich vlastnosti.

Při vyplňování identifikátoru, podle kterého se prvek určuje, jsou napovídány hodnoty dostupné ve vybraném formuláři. Tedy při výběru podle atributu `name` se napoví všechny hodnoty, které jsou u daného atributu v prvcích formuláře dostupné. Při výběru dle XPath se napovídají možnosti, jak lze ve vytváření XPath výrazu v kontextu vybraného formuláře pokračovat.

Pokud je nastavován prvek `select`, konkrétně jeho vybraná hodnota, jsou napovídány možnosti, které prvek obsahuje. Jde o textové popisky položek, které jsou při zobrazení v prohlížeči dostupné, i o hodnoty atributu `value` dostupných voleb.

Při vyplňování textu do textového pole je možné kromě textové informace zadat i některou z vestavěných funkcí, která bude provedena a navrácený text do pole vložen.

5.3.4 Extrakce dat

Při extrakci dat ze stránky je možné vybrat, zda budou extrahována pomocí regulárního či XPath výrazu. Zadá se příslušný výraz spolu s názvem dat, pod kterým budou dostupná.

V případě extrakce pomocí XPath výrazu je možné přepnout se do zobrazení náhledu stránky a kliknutím na element jemu odpovídající XPath výraz vygenerovat. Výraz je vyplněn do příslušného textového pole. V případě regulárního výrazu je možné přepnout se do zobrazení zdrojového kódu stránky a nechat si zvýraznit nalezené výskyty pro kontrolu správnosti. Poté je možné přepnout se zpět ke konfiguraci uzlu.

5.3.5 Uložení dat

Extrahovaná data lze uložit pod zadaným jménem. Je to užitečné, pokud je potřeba během navigace sbírat data, která pak budou exportována do souboru.

5.3.6 Import dat

V případě importu dat je nastaven typ souboru, ze kterého se data načtou. Dostupné jsou CSV a XML. Poté lze vybrat umístění souboru a název dat, pod kterým budou dále dostupná. V případě dat z CSV souboru lze nastavit, zda soubor obsahuje řádek se záhlavím sloupců a jeho kódování. U XML souboru je zadán XPath výraz popisující uzly, které budou vybrány. Kódování XML souboru je specifikováno přímo v jeho obsahu.

5.3.7 Export dat

Při exportu dat je možné vybrat, zda jsou exportována jen určitá data specifikovaná jejich jménem, všechna data nebo celá stránka. Je nutné nastavit umístění a název souboru, do kterého se data exportují.

5.3.8 Iterátor

Pro iterátor je nutné nastavit jméno kolekce, přes níž se iteruje, a název, pod kterým bude položka kolekce v každé iteraci dostupná.

5.3.9 Větvení

Větvení lze provádět na základě testu na přítomnost elementu ve stránce a testu obsahu elementu nebo celého zdrojového textu stránky pomocí regulárního výrazu. Dle volby je zadán příslušný XPath výraz pro určení elementu a regulární výraz pro kontrolu obsahu.

5.4 Spouštění projektů

Spuštění vytvořeného projektu je v rámci aplikace možné ve dvou režimech. Lze ho spustit v režimu návrhu a ladění nebo naplánovat jeho spuštění. Projekt lze navíc spustit i pomocí konzolové aplikace.

5.4.1 Návrh a ladění

V tomto případě jsou informace vypisovány do konzole v dolní části hlavního okna aplikace. Chybová hlášení jsou zvýrazněna červenou barvou. V případě, že nastane závažná chyba, která znemožňuje pokračování provádění skriptu, je uživatel o chybě informován zobrazením dialogového okna s popisem chyby a provádění je ukončeno.

5.4.2 Plánované spuštění

Druhou možností je naplánování spuštění projektu na určitý čas. Spouštění je také možné v zadaném intervalu opakovat. V tomto případě nejsou informace vypisovány do konzole, ale do souboru, který je k tomu určen. Tento soubor je vytvořen ve složce projektu v adresáři `run` a je pojmenován na základě data a času spuštění. V případě, že nastane závažná chyba, je její popis rovněž zapsán do uvedeného souboru. Pro správu naplánovaných spuštění projektů je vytvořen dialog.

5.4.3 Konzolová aplikace

Spuštění projektu lze provést i s využitím konzolové aplikace. Je určena pro spouštění hotových projektů. Při spuštění je jí předán jako parametr název souboru se skriptem a je provedena interpretace příkazů, které jsou v něm obsaženy. Informace jsou vypisovány na standardní výstup a chybová hlášení včetně závažných chyb na standardní chybový výstup.

5.5 Shrnutí

V této kapitole byl představen koncept uživatelského rozhraní a různé metody využitelné pro usnadnění popisu navigace. Jejich přínos pro uživatele je popsán v kapitole 7 týkající se testování aplikace na různých stránkách. Popsán byl také návrh dialogů pro konfiguraci jednotlivých typů uzlů při vytváření vizuálního popisu navigace a byly uvedeny různé možnosti pro spouštění projektů.

V následující kapitole jsou rozebrány detaily implementace navržené aplikace, včetně navrženého mechanismu pro navigaci.

Kapitola 6

Implementace

V této kapitole je nejprve popsána platforma NetBeans, na níž je implementovaná aplikace postavena, a jsou zde uvedeny použité knihovny. Dále je zde rozebrána implementace jednotlivých částí aplikace.

6.1 Vývojová platforma

Implementovaná aplikace je postavena na platformě NetBeans verze 7.1. Jelikož se jedná o *rich client platformu*, je zde tento pojem vysvětlen. Následuje popis samotné platformy NetBeans. Jednotlivé části této sekce čerpají z [16].

6.1.1 Rich client aplikace

Termín *rich client* se používá pro klienty, kteří provádí zpracování dat. Klient poskytuje uživateli grafické uživatelské rozhraní a většinou je rozšiřitelný pomocí zásuvných modulů.

Rich client je obvykle postaven nad nějakým základem, tzv. *rámcem*. Ten nabízí určitou infrastrukturu, na které může uživatel aplikaci sestavit z logických částí, tzv. *modulů*. Velmi výhodné je, pokud zdánlivě nesouvisející řešení, která mohou být vytvořena různými výrobci, mohou pracovat společně a tvořit navenek jeden celek. Poskytovatelé softwaru mohou aplikace skládat z různých modulů dle potřeb a přání uživatele.

6.1.2 Rich client platforma

Rich client platforma je prostředí pro životní cyklus programu využitelné jako základ pro vytvoření desktopové aplikace.

Většina aplikací má podobné vlastnosti, jako jsou položky menu, toolbary, stavový řádek, ukazatele průběhu zpracování, nastavení, ukládání a načítání uživatelských dat, internacionalizace, nápověda a podobně. Pro tyto typické vlastnosti aplikací poskytuje rich client platforma rámec, pomocí kterého mohou být snadno vytvořeny. Hlavním znakem takového rámce je možnost konfigurace a přizpůsobitelnost. Položky menu lze například deklarativně definovat v XML souboru a s využitím rámce jsou automaticky načteny a vytvořeny.

Nejdůležitějším aspektem rich client platformy je její architektura. Aplikace jsou psány ve formě modulů, které jsou tvořeny logicky souvisejícími částmi aplikace. Moduly jsou navzájem izolované. Jsou popsány deklarativně a jsou platformou automaticky načteny. Nejsou potřeba žádné explicitní vazby mezi zdrojovým kódem a aplikací. Mezi samostatně

fungujícími moduly jsou tak volné vztahy a je zjednodušena dynamická rozšiřitelnost aplikace nebo záměna jejích částí. Takto je snadné z jednotlivých modulů sestavit uživatelsky specifickou aplikaci.

6.1.3 Platforma NetBeans

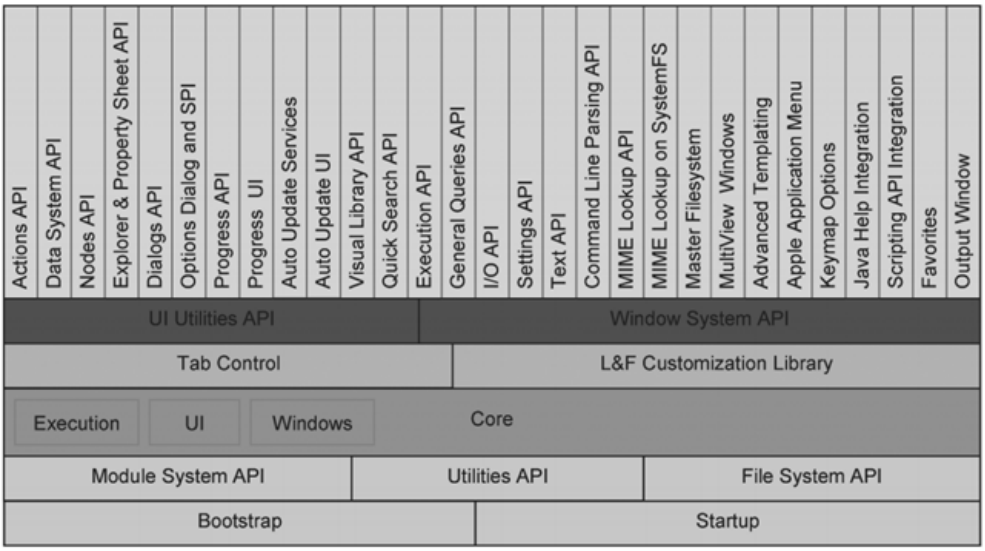
Platforma NetBeans poskytuje kromě obecných vlastností rich client platformy množství API, které mohou být při vytváření aplikace velmi užitečné. Platforma poskytuje programátorovi prostředky pro ukládání stavu, propojování akcí s položkami menu, toolbarů a klávesovými zkratkami, správu oken, editor, podporu pro vytváření uživatelských dialogů a nastavení, internacionalizaci, systém nápovědy a další. Bez využití rich client platformy musí programátor toto vše vytvářet ručně.

Nejznámějším produktem postaveným na platformě NetBeans je vývojové prostředí NetBeans IDE. Existuje ale velké množství dalších aplikací, které jsou na této platformě rovněž postaveny.

Architektura platformy NetBeans

Základním stavebním kamenem platformy je modul. Modul je kolekce tříd, které spolu funkčně souvisí. Obsahuje popis rozhraní, které modul vystavuje, a také popis ostatních modulů, které pro svůj běh potřebuje.

Celá platforma NetBeans, stejně jako aplikace na ní postavené, je rozdělena do modulů. Ty jsou načítány jádrem platformy, tzv. *běhovým kontejnerem NetBeans* (*NetBeans Runtime Container*). Kontejner načítá moduly automaticky a dynamicky a je zodpovědný také za běh aplikace. Aby byla aplikacím poskytnuta vysoká úroveň modularity, poskytuje platforma mechanismy a koncepty, které umožní rozšiřovat i moduly pomocí jiných modulů, ale na druhou stranu spolu moduly mohou komunikovat bez velké závislosti mezi sebou. Tento princip se nazývá slabá vazba (*loose coupling*).



Obrázek 6.1: Architektura platformy Netbeans [16]

Velmi dobrým příkladem modulární aplikace je NetBeans IDE. Funkčnost a charakteristiky vývojového prostředí, jako jsou např. podpora jazyka Java či editor zdrojového

kódu, jsou rozděleny do modulů postavených na platformě NetBeans. Přináší to velkou výhodu, že aplikace může být s využitím přídavných modulů rozšířena, například o podporu dalších jazyků nebo pro uživatelsky specifické potřeby. Některé moduly mohou být také deaktivovány nebo odinstalovány.

Architektura platformy NetBeans je zobrazena na obrázku 6.1. Vlastní platforma je sestavena ze skupiny základních modulů, které jsou nutné pro start aplikace a vytvoření uživatelského prostředí. Platforma však navíc poskytuje množství API a služeb, které značně zjednodušují vývoj.

Jmenovat lze například *Actions API* pro vytváření uživatelských akcí, *Options SPI* pro vytváření dialogů pro nastavení, *Text API* pro vytvoření editoru textových souborů, *MultiView Windows* pro vytváření oken s více pohledy na data, *Keymap Options* pro nastavení klávesových zkratek a *Output Window* pro výpis informací do konzole.

6.2 Použité knihovny

V této části jsou popsány knihovny, které implementovaná aplikace využívá. Zejména se jedná o knihovnu pro interpretaci Jythonu, HTTP klient pro internetovou komunikaci a komponentu pro zobrazování HTML stránky.

6.2.1 CSV soubory

Pro práci s CSV soubory je využita volně dostupná knihovna *JavaCsv*[1]. Umožňuje nastavit oddělovač záznamů na řádku i znak pro jejich uvození. Umí pracovat i s hlavičkou s názvy sloupců. Při načítání jednotlivých záznamů pak umožňuje získávat jejich položky podle čísla i názvu sloupce.

6.2.2 XML soubory

Pro zpracování XML souborů je možné využít standardních tříd z balíčku `org.w3c.dom`. V aplikaci je však pro práci s XML soubory využita knihovna *XOM*[2]. To z toho důvodu, že dle mého názoru poskytuje intuitivnější rozhraní pro tvorbu elementů a jejich atributů. Navíc nejsou svázány s dokumentem, jako v případě standardních knihoven, a lze tedy části dokumentu vytvořit nezávisle na sobě a pak je spojit v jeden celek. Usnadní to ukládání projektů a export dat jako XML soubor.

6.2.3 Jython interpret

Jak bylo zmíněno v kapitole 4.2, navržený skriptovací jazyk je založen na projektu Jython[3]. Jedná se o interpret jazyka Python verze 2.5 implementovaný pro Javu. Tento interpret je v Javě dostupný skrze standardní *Scripting API*. Stačí knihovnu přidat do classpath a pak lze na základě jména skriptovacího jazyka získat třídu umožňující interpretovat příkazy.

[1] http://www.csvreader.com/java_csv.php
[2] http://www.xom.nu
[3] http://www.jython.org

6.2.4 HTTP klient

Pro načítání HTML stránek a práci s nimi je využita knihovna HtmlUnit[4]. Jde o webový prohlížeč bez uživatelského rozhraní určený pro Java programy. Poskytuje model pro HTML dokumenty a API pro načítání stránek a práci s odkazy či formuláři. Do určité míry podporuje i zpracování JavaScriptu. Je určen buď pro testování stránek, nebo pro načítání informací z nich [17].

6.2.5 Zobrazení HTML stránky

Pro zobrazení náhledu HTML stránky je využita knihovna *MozSwing*[5]. Poskytuje komponentu integrovatelnou do uživatelského rozhraní vytvořeného pomocí *Java Swing*. Komponenta integruje webový prohlížeč Mozilla Firefox a využívá nativních knihoven. Ty jsou dostupné pro operační systémy Windows, Linux, Mac OS a Solaris. Díky integraci prohlížeče Firefox poskytuje velmi kvalitní zobrazení HTML stránky.

Stránku je možné načíst na základě URL nebo z řetězce obsahujícího zdrojový kód stránky. Komponenta také umožňuje získat element nacházející se pod kurzorem myši při kliknutí, což je pro potřeby implementované aplikace velmi užitečné.

6.3 Interpret popisu navigace

V této části je popsáno, jak je implementována interpretace navrženého skriptovacího jazyka a jakým způsobem je vyřešeno provádění vizuálního popisu navigace.

6.3.1 Skriptovací jazyk

Jak již bylo zmíněno, navržený skriptovací jazyk je založen na jazyce Python, pro který je pro Javu dostupný interpret, v rámci projektu Jython. Základní programové konstrukce, jako jsou podmínky a cykly, a práce s proměnnými jsou tedy stejné jako v jazyce Python. Tento základ je však nutné rozšířit o funkce pro realizaci jednotlivých navigačních kroků. Interpret pro potřeby rozšíření umožňuje provádět ve skriptu i kód v jazyce Java. V rámci skriptu je tak možné vytvořit objekt třídy definované v jazyce Java a zavolat nějakou jeho metodu. Tohoto mechanismu je pro rozšíření také využito.

Inicializaci interpretu Jythonu a předávání skriptu, který se spouští, zajišťuje třída `WebBrowserEngine`. Dále je vytvořena třída `WebBrowser`, která obsahuje funkce zajišťující provádění navigačních kroků. Před prováděním uživatelem napsaných skriptů pro navigační aktivity je vždy proveden inicializační skript, který zajistí definování vestavěných funkcí, aby je uživatel ve svém skriptu mohl volat.

Ukázka definice několika funkcí je v ukázce 6.1. Součástí definice je i speciální vlastní notace pro zápis komentářů k jednotlivým funkcím. Tyto komentáře jsou z inicializačního souboru načteny a zobrazovány při napovídání funkcí v editoru pro vytváření skriptu.

Proměnná `webBrowser`, jejíž metody jsou v těle definic funkcí volány, je inicializována v rámci objektu třídy `WebBrowserEngine`, a je do interpretu vložena pomocí dostupné metody. Proměnnou lze poté v inicializačním skriptu používat.

Při předávání proměnných jako parametrů pro metody definované v Javě dochází k automatické konverzi typů. Při vytvoření proměnné s řetězcem, která je definována v rámci

[4] http://htmlunit.sourceforge.net
[5] http://sourceforge.net/projects/mozswing

```
#@ Načte stránku s danou URL adresou a nastaví ji jako
aktuálně načtenou.@
def loadPage(url):
  webBrowser.loadPage(url)

#@Následuje odkaz dle textu, který obsahuje. Pokud je
na stránce více odkazů s daným textem, lze je odlišit
pořadím daným parametrem pos.@
def clickLinkByText(text, pos=0):
  webBrowser.clickLinkByText(text, pos)
```

Ukázka 6.1: Ukázka definice vestavěných funkcí v inicializačním skriptu

skriptu a je předána jako parametr, tedy dojde k automatické konverzi na Java typ `String`. Naopak pokud je v jazyce Java vytvořeno pole, je navráceno například jako návratová hodnota a uloženo do proměnné, lze s ním pak v rámci skriptu pracovat jako s běžným polem definovaným v jazyce Python. Lze zavolat například funkci `len` pro zjištění jeho délky nebo přes něj iterovat.

6.3.2 Vizuální popis

Vizuální popis je reprezentován třídami, které představují jednotlivé uzly. Je vytvořena abstraktní třída `NavigationStep`, která slouží jako základ pro jednotlivé typy uzlů. Hrany jsou reprezentovány odkazy mezi objekty těchto tříd.

Takto vytvořený popis není přímo interpretován, ale je na základě něj vygenerován zápis pomocí skriptovacího jazyka, který je prováděn.

Generování skriptu z vizuálního popisu

Při generování skriptu se postupuje od počátečního uzlu a zpracovávají se všechny další uzly, dokud se nenarazí na uzel, z nějž nevede žádná hrana. Každý uzel představuje určitou aktivitu a má zadány nějaké parametry. Na základě typu uzlu a jeho parametrů je vygenerována část skriptu, která uzlu odpovídá. V případě uzlu načtení stránky je například vygenerována funkce `loadPage`, při následování odkazu dle textu funkce `clickLinkByText`.

Pro podmínky je k dispozici uzel větvení. Mohou z něj vést dvě hrany pro větev `if` a větev `else`. Obě větve jsou ukončeny uzlem sloučení. Nejprve je vygenerován příkaz `if` se zápisem podmínky, následně tělo větve `if`, klíčové slovo `else` a tělo větve `else`. Obě větve jsou oproti ostatním příkazům odsazeny. Je to tak z důvodu syntaxe zápisu podmínek.

Složitějším případem je generování skriptu pro iterátor. Jeho tělo je odsazeno o jednu úroveň. Může však obsahovat další iterátor a zanoření tak může být v podstatě libovolné.

Při zpracování uzlu iterátoru je nejprve vygenerován příkaz `for` s názvem proměnné a kolekce, poté je zvýšena úroveň zanoření, uzel iterátoru je uložen na zásobník a je zpracováváno tělo iterátoru. To začíná uzlem hrany `DO`. Pro každý uzel těla jsou vygenerovány příkazy s patřičnou úrovní odsazení. Pokud se v těle iterátoru narazí na další iterátor, je znovu rekurzivně zpracován. Tělo iterátoru je ukončeno tak, že z posledního uzlu jeho těla vede hrana zpět do uzlu iterátoru. Pokud se tedy v těle narazí znovu na uzel iterátoru, jehož tělo se zpracovává, je tělo ukončeno, snížena úroveň odsazení a pokračováno aktivitou po provedení iterátoru, která začíná uzlem hrany `DONE`. Pokud se v těle narazí na iterátor, který je v zásobníku, ale není na jeho vrcholu, je nahlášena chyba, jelikož je takové propojení neplatné.

6.4 Realizace navigace na webových stránkách

Pro načítání stránek a práci s nimi je využita knihovna *HtmlUnit*. Ta však obsahuje API pouze pro základní úkony. Veškerá potřebná funkcionalita spojená s interpretací popisu navigace je implementována ve třídě `WebBrowser`. Poskytuje metody pro načtení stránky, práci s odkazy, formuláři, přihlašování, extrakci dat, import dat ze souborů, export dat do XML souboru a další.

6.4.1 Načtení stránky a práce s odkazy

Načtení stránky je realizováno s využitím API HtmlUnit. Načtená stránka je však navíc uložena jako aktuální, aby se s ní dalo dále pracovat.

Při následování odkazu dle textu nebo regulárního výrazu jsou z aktuální stránky získány všechny odkazy. Ty jsou pak postupně procházeny a je hledán odpovídající odkaz obsahující zadaný text nebo odpovídající regulárnímu výrazu. V prvním případě jsou z textu odkazu ořízнuty bílé znaky na začátku a na konci. Pokud je specifikováno i pořadí i odkazu, je vybrán až i-tý vyhovující odkaz. V případě výběru odkazu dle XPath je nalezen prvek, který mu odpovídá, a je provedena kontrola, zda se jedná o odkaz. Pokud odkaz není nalezen, je vypsáno varovné hlášení. Hledání na základě XPath umožňuje přímo třída `HtmlPage` reprezentující stránku.

6.4.2 Práce s formuláři

Při výběru formuláře dle XPath je nalezen element odpovídající výrazu a je zkontrolován, zda jde o formulář. Při výběru dle atributu `name` nebo `id` je vytvořen XPath výraz a podle něj formulář nalezen. Například při výběru formuláře podle jeho jména je vytvořen výraz `.//form[@name='jmeno']`.

Prvky formuláře jsou hledány obdobným způsobem. Při nastavování vybrané hodnoty u prvku `select` je potřeba nejprve nalézt odpovídající volbu `<option>`. Jsou získány všechny tyto elementy vybraného prvku `select` a mezi nimi je hledána volba odpovídající zadané textové hodnotě. Ta je u každé volby porovnávána s hodnotou jejího atributu `value` nebo textem, který obsahuje. Pokud není odpovídající volba nalezena, je vypsáno varovné hlášení.

Při nastavování hodnoty prvku `text` je využita dostupná funkce knihovny HtmlUnit. Podobná situace nastává při zaškrtávání prvku `checkbox` či `radio`. Ve druhém případě je odškrtnutí ostatních prvků se stejným jménem zajištěno automaticky knihovnou HtmlUnit. Odeslání formuláře je po nalezení požadovaného tlačítka opět zajištěno funkcí knihovny HtmlUnit. Nově načtená stránka je nastavena jako aktuální.

6.4.3 Přihlašování

Při využití HTTP přihlášení se pouze nastaví zadané údaje pro třídu `WebClient` knihovny HtmlUnit. Při přihlášení pomocí formuláře je nutné na stránce najít přihlašovací formulář. Za něj je považován takový, který obsahuje textové pole pro zadání uživatelského jména a pole typu `password` pro heslo. Na ostatních prvcích nezáleží. Takovýto formulář je na stránce nalezen, jsou v něm vyplněny požadované údaje a je provedeno jeho odeslání.

6.4.4 Kontrola obsahu stránky

Při kontrole obsahu stránky pomocí regulárního výrazu je jako text stránky uvažován její zdrojový kód získaný serializací jejího DOM stromu. V tomto textu je vyhledán výskyt zadaného regulárního výrazu. V případě testu stránky na obsah elementu je element na stránce vyhledán pomocí XPath výrazu. Při kontrole obsahu elementu je element vyhledán na základě XPath výrazu a v jeho obsahu je vyhledán zadaný regulární výraz.

6.4.5 Import dat ze souborů

Při importu dat z XML souboru je nejprve celý soubor načten a poté jsou z něj vybrány uzly na základě zadaného XPath výrazu. Tyto uzly jsou navráceny jako pole, se kterým lze pak v rámci skriptu pracovat.

V případě načítání dat z CSV souboru je také celý soubor načten a jeho obsah navrácen jako pole záznamů, které jsou reprezentované třídou `CSVRecord`. V záznamu jsou hodnoty jednotlivých sloupců uloženy podle jejich pozice a názvu. Lze s ním pracovat pomocí vestavěných funkcí `csvGetColByPosition` a `csvGetColByName`.

6.4.6 Extrakce dat

Při extrakci dat pomocí XPath výrazu jsou data ve stránce vyhledána a nalezené uzly navráceny jako pole. Při vyhledání textu pomocí XPath je brán v úvahu první nalezený element odpovídající výrazu, jeho obsah je převeden na text a ten je navrácen.

V případě extrakce dat pomocí regulárního výrazu jsou ve zdrojovém textu stránky vyhledány všechny výskyty zadaného regulárního výrazu. Každý nalezený výskyt je reprezentován jako pole řetězců, kdy na pozici 0 je výskyt celého regulárního výrazu a na dalších pozicích pak řetězce odpovídající ozávorkovaným částem výrazu. Seznam jednotlivých nalezených výskytů je také reprezentován jako pole. Celkově je tedy výsledek extrakce reprezentován jako dvourozměrné pole řetězců. Pokud je pomocí regulárního výrazu extrahován pouze text, je uvažován první nalezený výskyt a je navrácen řetězec odpovídající části výrazu uzavřené v závorkách dané pořadím předaným jako parametr příslušné funkci.

6.4.7 Export dat

Při exportu dat je výstup vytvořen ve formátu, jaký byl navržen v části 4.2.5. V případě extrakce elementů stránky je však potřeba provést kopii dané části DOM. Je to nutné z toho důvodu, že model stránky je daný knihovnou HtmlUnit a my chceme uzly uložit jako XML. Kopie je provedena rekurzivní metodou, která začne od extrahovaného elementu, případně kořenového elementu stránky v případě, že se extrahuje celá. V rámci ní je provedena kopie všech atributů a jsou rekurzivně zkopírovány všechny synovské elementy.

6.4.8 Funkce `exit`

V rámci provádění skriptu lze zavolat funkci `exit` pro ukončení provádění skriptu. Jelikož není v Jythonu možné zavolat funkci `sys.exit()`, která je běžně dostupná v jazyce Python, není tedy možné provádění skriptu tímto způsobem ukončit. Proto je funkce `exit` implementována tak, že je vygenerována výjimka vlastního typu `ExitException`, čímž je v podstatě provádění skriptu ukončeno a výjimka předávána dále. Na úrovni interpretu ji lze odchytit a detekovat její typ. V případě této výjimky tedy není vypsáno žádné chybové hlášení, jelikož se jedná o očekávané chování.

6.5 Editor pro vytváření popisu navigace

V této části je popsána realizace editorů pro editaci skriptu a vytváření vizuálního popisu. Oba editory jsou dostupné v rámci jednoho okna a lze mezi nimi přepínat pomocí záložek. Pro vytvoření tohoto zobrazení poskytuje platforma NetBeans *MultiView Windows*. Pro popis jedné záložky zobrazení slouží rozhraní `MultiViewElementDescription`. Obsahuje metody definující informace, jako jsou například název a ikona záložky. Deklaruje také metodu `createElement`, která je zodpovědná za vytvoření grafické komponenty, která je pod danou záložkou zobrazena. Komponenta musí implementovat rozhraní `MultiViewElement`, které mimo jiné deklaruje metody `getVisualRepresentation` a `getToolbarRepresentation` vracející `JComponent` z balíku `javax.swing` pro komponentu obsaženou pod záložkou a toolbar zobrazený vedle tlačítek pro přepínaní jednotlivých elementů.

6.5.1 Skriptovací jazyk

Pro editaci textových souborů poskytuje platforma NetBeans API pro editor. Pro jeho vytvoření pro vlastní typ souboru je potřeba rozšířit třídu `CloneableEditor`. Pro tyto účely je implementována třída `SourceEditorPanel`, která ještě navíc i implementuje rozhraní `MultiViewElement`, aby mohla být zobrazena jako záložka v okně. Pro její implementaci je však ještě potřeba vytvořit třídu `ScriptEditorSupport`, která rozšiřuje `DataEditorSupport`. Překrytím metod `notifyModified` a `notifyUnmodified` lze dosáhnout aktivace a deaktivace akce pro ukládání. Obsahuje privátní třídu `ScriptEnv`, která implementuje `SaveCookie` a stará se o ukládání editovaného souboru.

Pro vytvoření vlastního typu souboru, který je možné v editoru otevřít, je ještě potřeba vytvořit třídu `NavScriptDataObject` rozšiřující `MultiDataObject`. Pomocí ní je definován MIME typ souboru, na základě něhož lze k editoru připojit automatické doplňování kódu nebo zvýraznění syntaxe v případě, že je v něm soubor daného typu otevřen.

Platforma poskytuje také API pro automatické doplňování kódu. Pomocí něj jsou napovídány uživatelské funkce a jejich parametry. Doplňování kódu je však definováno odděleně a s editorem je svázáno pouze MIME typem editovaného souboru. Tento typ se zadává deklarativně jako anotace třídy `ScriptCompletionProvider`, která rozšiřuje třídu `CompletionProvider` a zajišťuje načítání napovídaných položek. Tato třída přepisuje metodu `createTask` vytvářející úlohu pro načtení položek a může být vykonána asynchronně. Úloha je reprezentována třídou `AsyncCompletionTask`, jejímuž konstruktoru je potřeba předat objekt třídy `AsyncCompletionQuery`. Ta obsahuje metodu `query`, kterou je nutno přepsat pro docílení vytvoření seznamu napovídaných položek. Položky jsou přidávány do `ResultSetu`, který je předán jako parametr metody `query`.

Každá položka je reprezentována třídou rozšiřující `CompletionItem`. Pro položky představující funkce je připravena třída `FunctionCompletionItem`, pro položky parametrů slouží třída `ParamCompletionItem`. Obě třídy definují metodu `defaultAction`, která je zavolána v případě potvrzení napovídané položky. Metoda musí zajistit vložení doplňovaného textu do editovaného dokumentu a uzavření okna s napovídanými položkami. Získání seznamu napovídaných položek je realizováno s využitím třídy `HintHelper`. Tato třída je využita i u vytváření vizuálního popisu a bude ještě popsána.

U každé položky je možné zobrazit její popis. Popisy jednotlivých položek jsou uloženy v inicializačním souboru spolu s definicemi vestavěných funkcí a odtud jsou načteny. Pro zobrazení popisu při nápovědě položek je nutné přepsat metodu `createDocumentationTask` třídy `CompletionItem`, kde je specifikován text popisu.

6.5.2 Vizuální popis

Pro vytváření vizuálního popisu je využito *Visual Library API*. Jedná se o knihovnu pro vizualizaci různých struktur. Zvláště dobře použitelná je pro grafové reprezentace [16]. Jednotlivé uzly jsou reprezentovány potomky třídy `Widget`. Konkrétně se jedná o třídu `IconNodeWidget`, která umožňuje vytvořit uzel s požadovanou ikonou a popiskem pod ní. Pro vytvoření hran mezi uzly je využita třída `ConnectionWidget`, u níž lze nastavit zdrojový a cílový uzel a styl začátku a konce hrany. Lze k ní také přidat popisek.

Widgety a hrany ve scéně jsou svázány s objektovým modelem. Při přidávání widgetu do scény je widget vždy asociován s jemu odpovídajícím objektem. Na základě widgetu pak lze získat příslušný objekt a naopak.

Pro uzly je vytvořena třída `NavigationStep`. Ta obsahuje obecné vlastnosti navigačního kroku, jak je například jeho typ, popisek, ikona, vstupní a výstupní hrana. Tato třída je rozšířena pro jednotlivé typy uzlů a každá vytvořená třída obsahuje informace nutné pro konfiguraci daného typu uzlu. Pro krok iterátoru tedy například obsahuje název kolekce, přes níž se iteruje, a jméno proměnné, přes kterou je aktuální prvek kolekce dostupný. Obsahuje navíc ještě odkaz na výstupní hranu `DONE`, která ukazuje na uzel provedený po dokončení iterací pro všechny prvky kolekce, a na vstupní hranu pro návrat z těla iterátoru. Pro hranu `DO` je využita obecná výstupní hrana ze třídy `NavigationStep`.

Pro hrany je určena třída `NavigationEdge`, která obsahuje odkazy na její zdrojový a cílový uzel.

Aby bylo možné s uzly manipulovat, je nutné jim přiřadit akce. Pro jejich vytváření je možné využít třídu `ActionFactory`. Pomocí ní jsou vytvořeny akce například pro přesun uzlu a možnost jeho výběru.

Důležitá je akce pro možnost propojování widgetů hranami vytvořená s využitím metody `createExtendedConnectAction`. Tato metoda vyžaduje předání implementace rozhraní `ConnectProvider`. Ta umožňuje určit, zda mohou být dva widgety propojeny a také za vytvoření hrany mezi nimi. Zajišťuje také správné nastavení odkazů na hrany v rámci uzlů v objektovém modelu.

Předchozí akce se vztahovaly vždy ke konkrétnímu widgetu. Je však potřeba vytvořit ještě jednu akci, která se váže k celé scéně. Jedná se o akci pro přidávání uzlů z palety s využitím funkcionality *táhni a pusť*. Akce je vytvořena metodou `createAcceptAction` třídy `ActionFactory` a vyžaduje poskytnutí implementace rozhraní `AcceptProvider`. Ta je zodpovědná za informaci, zda lze přijmout typ dat předávaný pomocí tažení, a v případě očekávaného typu také za vytvoření příslušného uzlu.

Uzel v paletě implementuje rozhraní `Transferable` a při jeho přetažení poskytuje objekt třídy `NavigationStep`. Název typu dat je nastaven na řetězec `NavigationNode`. Na tento typ dat při přetažení uzlu do scény její `AcceptProvider` reaguje vytvořením příslušného uzlu pro předaný objekt reprezentující navigační krok.

Vizualizace napovídaných položek

Pro vizualizaci položek napovídaných v dialogu pro konfiguraci uzlu je využito seznamu, který se zobrazuje pod příslušným textovým polem. Okno se seznamem položek je implementované jako `JPopupMenu` a pro zobrazení jednotlivých položek je v něm vložena komponenta `JList`. Pro zobrazení seznamu slouží třída `HintAutoCompleter`. Pro připojení automatického doplňování k textovému poli stačí vytvořit objekt této třídy a jeho konstruktoru předat odkaz na textové pole.

Třída zajistí přiřazení akcí pro zobrazení seznamu položek při stisku CTRL+mezerník, nastavení vybrané položky pomocí šipek nahoru a dolu, potvrzení vybrané položky pomocí klávesy ENTER a skrytí nápovědy při stisku ESC. Při stisku kláves CTRL+ENTER je vybraná položka doplněna do textového pole, ale seznam položek zůstane zobrazen. Lze to využít při postupném vytváření XPath výrazu. Třída obsahuje abstraktní metodu `getItems`, která vrací seznam řetězců představujících napovídané položky. Tato metoda je definována až v konkrétních případech použití a seznam položek je v ní vytvořen.

6.5.3 Získání napovídaných položek

Pro získání seznamu položek napovídaných v různých situacích slouží třída `HintHelper`. Obsahuje například metody pro získání seznamu položek pro nápovědu textů odkazů stránky, jmen a id formulářů, nápovědu související s prvky ve formulářích, nápovědu při vytváření XPath výrazů a podobně. Pro načítání položek se využívá aktuálně načtené stránky, případně aktivního formuláře v případě nápovědy při práci s jeho prvky. Stránka i formulář jsou nastaveny v rámci provádění navigačních aktivit a jsou globálně dostupné.

Metody zmíněné třídy jsou využity při napovídání položek při vytváření skriptu i v dialozích pro konfiguraci uzlů u vizuálního popisu.

6.6 Zobrazení obsahu stránky

Obsah aktuální stránky je vizualizován různými způsoby. V této části jsou popsány implementační detaily těchto způsobů zobrazení. Pro každé zobrazení je vytvořena třída rozšiřující `TopComponent`, která umožňuje zobrazení okna v rámci aplikace. Otevřená okna lze přepínat pomocí záložek nebo je přichytávat okrajům hlavního okna. Lze je však také zobrazit odděleně od hlavního okna.

6.6.1 Náhled stránky

Pro zobrazení aktuálně načtené stránky slouží třída `MozSwingBrowserTopComponent`. Pro zobrazování stránky využívá knihovny *MozSwing*, která již byla popsána v použitých knihovnách.

Načtení stránky do vykreslovací komponenty je realizováno předáním jejího zdrojového kódu jako parametr metodě `load` komponenty. Vygenerování zdrojového kódu je realizováno serializací DOM stránky. Stránka může obsahovat odkazy na obrázky, CSS styly a další objekty. Odkazy však mohou být zadány relativně a komponenta je v tom případě nedokáže stáhnout, jelikož nezná základní URL, protože je jí stránka předávána pouze jako zdrojový kód.

Jsou dvě možnosti, jak tento problém vyřešit. První z nich je stažení všech odkazovaných objektů a jejich uložení do adresářové struktury v dočasné složce, spolu se souborem stránky. Pokud by stránka byla do komponenty načtena na základě URL dočasného souboru, došlo by ke správnému načtení všech odkazovaných objektů. Výhodnější řešení je ovšem převedení všech relativních odkazů na absolutní. V tom případě si komponenta všechny odkazované objekty sama korektně načte. Tato transformace odkazů je provedena s využitím objektového modelu stránky ještě před jeho serializací. Pomocí XPath jsou nalezeny všechny elementy `img` pro obrázky a pokud je jejich atribut `src` relativní, je s vyžitím URL stránky upraven na absolutní.

CSS soubory jsou ke stránce připojeny pomocí elementů `link` s atributem `rel` s hodnotou `stylesheet`. Jejich atributy `href` s odkazy na CSS soubory jsou upraveny podobným způsobem jako u obrázků. Existuje však ještě jedna možnost, jak ke stránce připojit CSS soubor. Lze to realizovat pomocí příkazu `import` v elementu `style` v hlavičce stránky. Tyto případy jsou také brány v úvahu a odkaz na soubor nahrazen za absolutní.

Nakonec je ještě provedena úprava odkazů u elementů, pomocí nichž jsou do stránky vkládány externí javascriptové soubory.

Generování XPath výrazu formuláře

Při výběru formuláře pomocí XPath výrazu v případě tvorby vizuálního popisu je možné nechat výraz vygenerovat na základě kliknutí myši do prostoru formuláře v náhledu stránky. Po kliknutí je získán element, který se pod kurzorem myši nacházel. Typicky je to nějaký prvek obsažený ve formuláři. Od něj se postupuje v hierarchii elementů směrem ke kořeni, dokud se nenarazí na element reprezentující formulář. U tohoto formuláře se zkontroluje, zda obsahuje atribut `name` nebo `id`. Pokud ano, je vytvořen XPath výraz, který na základě daného atributu formulář identifikuje. Pokud ani jeden z uvedených atributů není přítomen, je vygenerován XPath výraz popisující kompletní cestu od kořene stránky k danému formuláři.

6.6.2 Zdrojový kód stránky

Pro zobrazení zdrojového kódu stránky je určena třída `WebPageSourceCodeTopComponent`. Ta využívá komponenty `JTextPane`, která umožňuje zvýrazňování požadovaných částí vloženého textu. Změny pozadí jednotlivých částí lze docílit pomocí objektu třídy `Highlighter`, který je pro textovou komponentu dostupný. Pomocí něj lze nastavit pozadí požadované části textu na určenou barvu. Umožňuje také odstranění všech zvýraznění, což je užitečné při zvýrazňování nového regulárního výrazu.

6.6.3 DOM strom stránky

Pro vizualizaci DOM stromu stránky je vytvořena třída `HtmlTreeTopComponent`. Pro zobrazení stromu je využita komponenta `BeanTreeView`. Pro jednotlivé uzly stromu je potřeba rozšířit třídu `Node`. Pro elementy je takto vytvořena třída `ElementNode`. Uzly pro synovské elementy jsou rekurzivně vytvářeny, čímž je zkonstruována celá struktura stromu. Atributy jsou zobrazeny jako součást názvu uzlu.

Výběr elementu dle místa kliknutí ve stránce

Při zpracování události kliknutí ve stránce lze získat element, na který bylo kliknuto. Je reprezentován standardním rozhraním `Node` z balíku `org.w3c.dom`. Pro výběr odpovídajícího uzlu v zobrazení stromu je potřeba mít možnost získat odkaz na tento uzel. Pro tyto účely je vytvořeno asociativní pole mapující elementy stránky na uzly v zobrazení stromu. Je postupně plněno při vytváření jednotlivých uzlů vizualizace stromu.

Zvýraznění vybraného elementu v náhledu stránky

Element vybraný v zobrazení stromu lze zvýraznit v náhledu stránky. Zvýraznění je realizováno nastavením barvy pozadí a orámování elementu s využitím CSS. Uvedené vlastnosti

jsou nastaveny pomocí DOM modelu stránky. Následně je znovu vygenerován její zdrojový kód, na základě něhož je zobrazení náhledu stránky aktualizováno.

6.7 Práce s projekty

Navigační projekt je reprezentován třídou `NavigationProject`. Ta obsahuje jeho jméno, umístění složky s jeho soubory a odkazy na část projektu tvořenou skriptem a část tvořenou vizuálním popisem.

Pro část projektu se skriptem je vytvořena třída `SourceNavigationProject`. Obsahuje název souboru, ve kterém je uložen skript. Tento soubor je umístěn ve složce projektu.

Pro část s vizuálním popisem slouží třída `VisualNavigationProject`. Třída obsahuje název souboru, do nějž je vizuální popis ukládán, a odkaz na počáteční navigační krok. Zajišťuje také generování skriptu z vizuálního popisu, jak bylo popsáno v části 6.3.2.

6.7.1 Zobrazení projektů

Pro zobrazení seznamu projektů slouží třída `ProjectExplorerTopComponent`. Zobrazení projektů je realizováno pomocí stromu s využitím komponenty `BeanTreeView`. Každý projekt je reprezentován uzlem `NavigationProjectNode`, který obsahuje uzel pro část projektu se skriptem a uzel pro část s vizuálním popisem. Přepsáním metody `getPreferredAction` je uzlu přiřazena akce pro otevření okna s projektem, která je aktivována při dvojkliku na uzel.

Při otevření projektu je s využitím *MultiView Windows* vytvořeno okno pro zobrazení projektu. Odkaz na okno je pro daný projekt uložen do asociativního pole. Pokud je pak projekt otevírán znovu, okno není vytvářeno, ale pouze je aktivováno již existující.

6.7.2 Ukládání a načítání

Pro načítání a ukládání projektů slouží třída `NavigationProjectWorker`. Seznam všech otevřených projektů je uložen v souboru `projects.xml`. Ten pro každý projekt obsahuje element s atributem udávajícím umístění složky projektu.

Každý projekt je umístěn ve složce, která obsahuje soubor `project.xml`. Tento soubor obsahuje název projektu a názvy souborů pro skript a vizuální popis. V souboru se skriptem jsou uloženy příkazy navigačního jazyka stejným způsobem, jako jsou vidět v editoru. Vizuální popis je ukládán jako XML soubor. Pro každý krok slouží element `step`, který obsahuje id kroku a pozici uzlu ve scéně, dále pak potřebné informace v závislosti na typu kroku. Propojení uzlů je uloženo s využitím elementů `connection`, které obsahují id zdrojového a cílového uzlu hrany a její typ pro oba uzly.

6.7.3 Spouštění

Existují celkem tři způsoby, jakými je možné projekt spustit. Tyto možnosti jsou popsány v kapitole 5.4. Ve všech případech pro interpretaci vytvořeného skriptu slouží třída `WebBrowserEngine`, která obsahuje metodu `eval`, která provedení skriptu zajistí. Pro vykonávání vestavěných funkcí tato třída obsahuje objekt třídy `WebBrowser`.

V rámci různých funkcí mohou být vypsány různé informace. Výpis informací je však potřeba v různých způsobech spuštění projektu realizovat různým způsobem. Proto způsob výpisu není pevně definován. Je realizován pomocí objektu implementujícího rozhraní `ScriptExecutionListener`, které deklaruje metody informující o začátku a konci spuštění

projektu a metody určené pro výpis informací. Díky tomu se třída `WebBrowser` nemusí starat o konkrétní způsob výpisu a její kód je stejný pro všechny možnosti spuštění projektu. Při konkrétním způsobu provádění je vytvořen objekt implementující uvedené rozhraní, který výpis realizuje požadovaným způsobem.

Režim návrhu a ladění

Pro výpis při spuštění v ladicím režimu slouží třída `DefaultScriptExecutionListener`. Informace jsou vypisovány do konzole v dolní části aplikace. Pro práci s tímto oknem slouží API *Output Window* platformy NetBeans, které poskytuje metody pro otevření okna a výpis textu do něj. Pro zobrazení informací o závažných chybách je využita třída `JOptionPane` a pomocí ní vytvořené dialogy. Interpretace příkazů skriptu projektu je spuštěna v samostatném vlákně, aby neblokovala uživatelské rozhraní.

Režim plánovaného spuštění

V tomto případě je pro výpis určena třída `ScheduledRunScriptExecutionListener`, která zajistí zapsání informací do souboru v adresáři projektu. Pro každé spuštění je vytvořen samostatný soubor. Každé plánované spuštění projektu je opět realizováno v samostatném vlákně.

Konzolová aplikace

Konzolová aplikace pro spuštění skriptu projektu využívá třídu `WebBrowserEngine`, stejně jako v obou popsaných způsobech spuštění v rámci desktopové aplikace. Ostatní potřebné třídy jsou také použity bez nutných změn.

Pro výpis informací slouží nově vytvořená třída `ConsoleRunScriptExecutionListener`, která informace vypisuje na standardní, případně standardní chybový výstup. Hlavní třídou konzolové aplikace je `WebNavigatorConsole`, která zajistí zpracování parametrů příkazové řádky a spuštění interpretace zadaného souboru se skriptem.

6.8 Shrnutí

V této kapitole byla popsána platforma NetBeans, na které je aplikace postavena, použité knihovny a implementační detaily jednotlivých částí vytvořené aplikace. Důležité je však také testování implementovaného mechanismu a aplikace na webových stránkách s různou strukturou, s různým způsobem přihlašování a prováděním různých navigačních aktivit. Touto problematikou se zabývá následující kapitola.

Kapitola 7

Testování

Tato kapitola se zabývá testováním aplikace na různých webových stránkách. Nejprve jsou popsány navigační aktivity, které jsou na jednotlivých stránkách vykonávány. Je také provedeno zhodnocení uživatelského přínosu jednotlivých způsobů usnadnění tvorby popisu navigace. Nakonec jsou zmíněny i stránky, které aplikaci činily problémy.

Testování bylo prováděno na stroji s procesorem Intel Core 2 Duo s frekvencí 2,8 GHz, 4 GB RAM a operačním systémem Windows 7.

7.1 Popis testovaných webových stránek

Testování implementovaného mechanismu a aplikace probíhalo na různých webových stránkách. Následuje stručný popis navigačních aktivit, které jsou na testovaných stránkách prováděny. Pro každou navigační aktivitu je vytvořen projekt, který lze otevřít prostřednictvím aplikace. Všechny potřebné soubory těchto projektů jsou dostupné na příloze C.

Vyhledání na portálu Seznam

Na této stránce je realizováno vyhledání zadaného slova s využitím formuláře na hlavní stránce. Výsledky vyhledávání, konkrétně titulky nalezených stránek, jsou uloženy do XML souboru.

Email na portálu Seznam

Je provedeno přihlášení do emailové schránky uživatele a ze stránky se seznamem emailů jsou extrahovány jejich předměty a čas jejich doručení. Tyto informace jsou uloženy do XML souboru.

Dále je provedeno odeslání emailu s požadovaným předmětem a tělem na zadanou emailovou adresu. K emailu je také přiložen soubor. Na konci je v obou případech provedeno odhlášení.

Seznam jídel v menze na FIT VUT

Je načtena stránka se seznamem jídel. Názvy dostupných jídel jsou uloženy do XML souboru a také vypsány do konzole. K dispozici je i popis této navigační aktivity s využitím vizuálního popisu.

Informační systém FIT VUT

Ze stránky s výpisem předmětů v aktuálním roce jsou extrahovány názvy předmětů a dosažené hodnocení. Informace jsou vypsány v konzoli. Přihlášení je realizováno s využitím HTTP autentizace.

Vyhledání dopravních spojů na portále IDOS

Do formuláře na výchozí stránce jsou vyplněny požadované informace a je odeslán. Ze získané stránky jsou extrahována nalezená spojení. Pro každé spojení je do konzole vypsána výchozí a cílová stanice spolu s časem odjezdu a příjezdu.

Rezervační systém Student Agency

Nejprve jsou vyplněny údaje do přihlašovacího formuláře a je provedeno přihlášení. Následuje přechod na stránku se seznamem rezervací. Pro každou dostupnou rezervaci spoje je do konzole vypsáno datum a čas, trasa, číslo sedadla a cena. Poté je provedeno odhlášení.

Počasí na portále iDNES

Navigační aktivita zahrnuje načtení stránky s počasím. Z ní jsou extrahovány informace o teplotě pro daný den a textový popis předpovědi. Poté je proveden přechod na stránku s počasím pro následující den, z níž jsou extrahovány tytéž informace. Údaje jsou poté vypsány do konzole.

Videoserver FIT VUT

Pomocí formuláře je provedeno přihlášení na videoserver FIT VUT a ze stránky s aktualitami jsou extrahovány názvy přednášek zveřejněných během posledních 5 dní, včetně data přednášky a data zveřejnění.

Zpravodajský portál Novinky.cz

Ze stránky je extrahován titulek a krátký popis aktuálního hlavního článku a informace jsou vypsány do konzole.

Fiktivní stránka pro správu osob

Pro účely testování aplikace byla vytvořena stránka, která umožňuje spravovat seznam osob. Zdrojový kód souvisejících stránek je uložen na příloze C. Jsou však také dostupné z URL uvedené ve skriptu. Stránka s formulářem pro přidání osoby je navíc uvedena v příloze.

Osoby jsou na stránce vypsány formou jednoduché tabulky. Tato stránka je načtena a informace o osobách jsou z ní extrahovány pomocí regulárního výrazu.

Druhá vytvořená aktivita spojená s touto stránkou je přidání osob na základě dat načtených ze XML souboru. Každá osoba je v souboru reprezentována elementem, který obsahuje potřebné informace. Pro každý element je načtena stránka s formulářem pro přidání osoby, který je na základě získaných dat vyplněn a odeslán.

K dispozici je i varianta vytvořená s využitím vizuálního popisu. Připravena je také ještě stejná aktivita, která však využívá dat z CSV souboru.

Slevomat

Na hlavní stránce je s využitím odkazu proveden přechod na přihlašovací stránku a na ní provedeno přihlášení pomocí formuláře. Pomocí odkazu je dále vybrána stránka se zakoupenými vouchery. Pomocí XPath jsou z ní extrahovány názvy aktuálních a již využitých voucherů, které jsou vypsány do konzole. Tato navigační aktivita je vytvořena s využitím vizuálního popisu.

Zpravodajský portál iDNES.cz

Z hlavní stránky jsou načteny odkazy na dostupné články. Pro každý odkaz je proveden přechod na jím určenou stránku s článkem, ze které je extrahován a uložen název a stručný popis článku. Na konci jsou všechna extrahovaná data uložena do XML souboru.

7.2 Výsledky testů

Testování bylo prováděno jednak za účelem odstranění chyb vzniklých při implementaci, ale také pro zjištění, jaký uživatelský přínos mají navržené způsoby usnadnění tvorby popisu navigace.

7.2.1 Vestavěné funkce

Při testování byly důkladně vyzkoušeny vestavěné funkce, zda se chovají podle očekávaní. V některých případech byly odhaleny drobné chyby. Jednalo se například o práci s odkazy na základě regulárních výrazů popisujících jejich text nebo o funkce využívající XPath výraz při práci s prvky formulářů.

Všechny nalezené nedostatky byly odstraněny. Dokonce byly ještě některé funkce přidány. Jde o například o funkci `displayTrayMessage`, pomocí které je možné zobrazit požadovanou hlášku v systémové liště (vedle hodin). Funkce se ukázala jako užitečná v případě periodické kontroly určité informace na stránce. Zpráva může být zobrazena například v okamžiku, kdy se na stránce požadovaná informace objeví. Další přidanou funkcí je `clickLinkByElement`. Umožňuje přechod na stránku danou odkazem určeným elementem předaným jako její parametr. Element může být získán například pomocí funkcí pro extrakci dat ze stránky.

Ukázalo se, že nabídka vestavěných funkcí je pro vytváření navigačních aktivit dostačující. Ve všech testovaných případech bylo možné požadovanou aktivitu pomocí dostupných funkcí popsat. Pouze v případě vytváření vizuálního popisu nebyl v některých případech dostupný uzel, pomocí něhož by bylo možné požadovaný krok vyjádřit. Nebylo však ani cílem, aby byly všechny možnosti skriptovacího jazyka popsatelné i pomocí vizuálního popisu. Proto je pro tyto případy připraven uzel typu *Vlastní*, pomocí kterého je možné zadat libovolné příkazy ve skriptovacím jazyce, které budou provedeny.

7.2.2 Přínos jednotlivých způsobů usnadnění tvorby popisu

Míra uživatelského přínosu jednotlivých navržených způsobů usnadnění vytváření popisu navigace je různá. Pro každý způsob však existují typické situace, kdy se velmi dobře uplatní.

Náhled stránky a strom jejích elementů

Při vytváření popisu navigace bylo nejvíce využíváno zobrazení náhledu stránky a strom jejích elementů. V souvislosti s tím se velmi často uplatnila funkce vybrání elementu ve stromu stránky na základě kliknutí do náhledu stránky. Dobře využitelné je i zvýraznění elementu ve stránce na základě jeho výběru ve stromu elementů. Díky těmto funkcím se lze rychle zorientovat ve struktuře stránky a prozkoumat požadovanou část jejího stromu.

Popsané funkce pro usnadnění se nejvíce uplatní při extrakci informací ze stránky, konkrétně při vytváření XPath výrazu pro jejich výběr. Pro jeho konstrukci je znalost struktury stránky nutná. Zobrazení stromu elementů stránky je dobře využitelné například i při výběru formuláře a při práci s prvky v něm. Snadno pomocí něj lze dohledat atributy pro identifikaci těchto prvků.

Při výběru formuláře podle XPath se dobře uplatní i možnost nechat si výraz vygenerovat na základě kliknutí do prostoru formuláře v zobrazení stránky. Jedná se o velmi pohodlný způsob jeho identifikace ve stránce.

Své využití našla také možnost nechat ve stromu stránky označit elementy vybrané na základě zadaného XPath výrazu. Spolu s tím byla hodně využita možnost nechat si tyto zvolené elementy zvýraznit v zobrazení náhledu stránky. Tímto způsobem lze snadno zkontrolovat, zda je zadaný výraz správný, tedy jestli vybrané a zvýrazněné elementy odpovídají očekávání.

Zdrojový kód stránky

Toto usnadnění tak velké využití nemá. Je to dáno především tím, že struktura většiny stránek je natolik složitá, že vyhledávání informací v jejich zdrojovém kódu na základě regulárního výrazu je velmi problematické. Pro jejich nalezení je potřeba konstrukce složitých regulárních výrazů. Mnohem jednodušší je proto použít pro jejich extrakci například XPath výraz.

V případě, že je ale struktura stránky poměrně jednoduchá, je možnost zvýraznění výskytů zadaného regulárního výrazu ve zdrojovém kódu stránky dobře použitelná a konstrukci výrazu hodně usnadní. Zejména zjednoduší opravu chybně zadaného výrazu. Tento způsob usnadnění byl využit například u navigační aktivity prováděné na vytvořené fiktivní stránce pro správu seznamu osob.

Nápověda položek při tvorbě popisu

Při vytváření skriptu pro popis navigace je velmi užitečné automatické doplňování názvů vestavěných funkcí. Uživatel nemusí jejich názvy znát přesně, ale stačí napsat prvních pár písmen a požadovanou funkci vybrat z nabídnutého seznamu. Při výběru funkcí je navíc zobrazen i popis chování dané funkce. Uživatel je tak schopen funkci rychle použít, i když třeba dosud nebyl s jejím chováním přesně seznámen.

Napovídat je však možné i parametry funkcí. Značně to usnadňuje tvorbu skriptu zejména při práci s odkazy a formuláři. V případě výběru odkazu dle textu jsou napovídány texty všech odkazů na stránce a uživatel tak může požadovaný odkaz snadno identifikovat, a to bez nutnosti zkoumat strom stránky. Pokud uživatel stránku zná, ví přibližně, jaký text se nachází u požadovaného odkazu a s využitím nabídnutého seznamu odkazů ho lze snadno najít.

Podobná situace nastává v případě práce s formuláři a jejich prvky. Uživateli jsou napovídány jejich atributy `name` a `id`, ale například i možná pořadová čísla, podle kterých je

lze také identifikovat. Pokud mají hodnoty atributů intuitivní hodnoty, je možné vytvořit popis pouze za pomoci této nápovědy. V opačném případě přijde na řadu i zkoumání stromu stránky a zjišťování, jakým způsobem lze požadovaný prvek identifikovat.

Velmi dobře se také uplatnila nápověda dostupných voleb u formulářového prvku typu `select`. Není proto nutné zkoumat, které možnosti prvek obsahuje, stačí vybrat z nabídnutého seznamu. Užitečné také je, že se volba dá identifikovat i podle textu, který obsahuje, a nejen podle atributu `value`. Výběr je tak intuitivnější.

Své využití našla také nápověda při vytváření XPath výrazu. Napovídání možností, jak lze v konstrukci výrazu pokračovat, pomáhá jednak při jeho konstrukci, ale také je možné přímo ověřovat, zda je napsán správně. Pokud jsou totiž napovídány očekávané možnosti, má uživatel jistotu, že je prozatím zadaná část výrazu správná.

Nápovědu různých možností lze využít i při vytváření vizuálního popisu. Položky jsou v tomto případě napovídány u textových polí v příslušných dialozích pro konfiguraci jednotlivých uzlů.

7.2.3 Problematické stránky

Existují stránky, které jsou pro implementovaný mechanismus problematické. Jedná se například o stránku pro vyhledání spoje v rezervačním systému Student Agency. Vyplnění a odeslání formuláře s údaji proběhne ještě v pořádku. Poté je však načtena stránka, v rámci které jsou vyhledané spoje získávány komplikovanými ajaxovými požadavky. S nimi si mechanismus neporadí. Načte se pouze stránka, kde je prostor pro zobrazení spojů, samotné spoje již ale načteny nejsou.

Další problematickou stránkou je například rozhraní pro zobrazení emailů od Google. Přihlášení proběhne v pořádku, poté je však pouze zobrazena stránka s informací, že jsou emaily načítány. Samotné načtení stránky se seznamem emailů však již neproběhne. Je tomu tak ze stejného důvodu, jako v případě první stránky, tedy že je při jejím načítání využito složitějšího javascriptového kódu.

Obecně tedy problémy nastávají u stránek, které obsahují komplexnější kód v jazyce JavaScript, zejména ajaxové požadavky. Pokud je ale kód jednodušší, mechanismus s ním poradí. Záleží však na konkrétní stránce a nelze přesně specifikovat, s jakým kódem si mechanismus ještě poradí, a s jakým už ne.

7.3 Shrnutí

V první části této kapitoly bylo popsáno, na jakých stránkách byla aplikace testována. Druhá část se zabývala zejména mírou uživatelského přínosu jednotlivých způsobů usnadnění vytváření popisu navigace. Ukázalo se, že nejvíce je využíváno zobrazení náhledu stránky a stromu jejích elementů. Často využívaným způsobem usnadnění je také nápověda funkcí a jejich parametrů při vytváření skriptu a nápověda různých možností u textových polí v dialozích pro konfiguraci uzlů v případě vizuálního popisu.

Při testování se také ukázalo, že všechny dostupné vestavěné funkce pracují tak, jak mají, a že pro popis různých navigačních aktivit je tato nabídka funkcí dostačující. Problémy se vyskytly pouze u stránek, které obsahují složitější javascriptový kód. V této fázi s tím však nelze nic udělat, jelikož je zpracování JavaScriptu kompletně v režii knihovny, která je pro práci s webovými stránkami využita.

Kapitola 8

Závěr

Cílem této diplomové práce bylo seznámit se s možnostmi přihlašování na privátní stránky, navrhnout mechanismus pro automatizovanou navigaci na webu, včetně skriptovacího jazyka pro její popis, a navrhnout a implementovat aplikaci, která ho bude využívat.

Jsou zde rozebrány technologie týkající se webových stránek. Je popsána navigace na webu, tedy jakými způsoby lze přecházet mezi jednotlivými stránkami, jakými způsoby se lze přihlašovat a jak se poté udržuje kontext uživatele. Uvedeny jsou existující nástroje s podobným zaměřením.

Na základě získaných poznatků byly vytyčeny cíle, které by měl navigační mechanismus splňovat. Následoval návrh skriptovacího jazyka pro popis navigace a získávání informací ze stránek. Byl rovněž představen princip vizuálního popisu navigačních kroků. Navrženo bylo také uživatelské rozhraní aplikace, která bude navigační mechanismus využívat, a popsány metody, pomocí kterých lze vytváření popisu navigace usnadnit.

Aplikace byla implementována s využitím platformy NetBeans a otestována na množství webových stránek s odlišnou strukturou, s různými způsoby přihlašování i bez něj a prováděny byly různé navigační kroky. Vytvořena byla i konzolová verze, která umožňuje snadné spuštění vytvořených skriptů.

Testování ukázalo, že je navržený skriptovací jazyk pro realizaci různých navigačních aktivit plně dostačují. Dobře využitelná je také možnost vizuálního popisu navigace. Bylo zjištěno, že nejvíce využívaným způsobem usnadňujícím vytváření popisu navigace je zobrazení náhledu stránky a stromu jejích elementů. Je to užitečné při definici extrahovaných dat i pro identifikaci jednotlivých prvků stránky, se kterými se během navigační aktivity pracuje.

Vytváření popisu navigace je také značně usnadněno nápovědou názvů vestavěných funkcí a jejich parametrů při vytváření skriptu a nápovědou možností u textových polí pro konfiguraci uzlů v případě vizuálního popisu navigace. Předejde se tím jednak překlepům, ale navíc pokud mají využívané prvky stránky atributy s intuitivními hodnotami, je možné vytvořit popis navigace jen na základě nabízených možností, bez potřeby zkoumat strukturu stránky.

Při testování se vytvořená aplikace celkově ukázala jako dobře použitelná a bylo možné bez problémů popsat požadované navigační aktivity na testovaných stránkách, ať už pomocí skriptu nebo s využitím vizuálního popisu. Své využití dobře najde také možnost naplánovat provedení jednotlivých projektů, včetně opakování po zadaném čase. Umožňuje to například periodickou kontrolu zadané stránky a provedení požadované aktivity v případě očekávané změny.

8.1 Navrhovaná rozšíření

Aplikaci je možné několika způsoby rozšířit o další funkcionalitu. Na závěr jsou zde tyto možnosti shrnuty.

Pravděpodobně nejpodstatnějším rozšířením je vylepšení podpory JavaScriptu. Aplikace si totiž neporadí s některými stránkami, které obsahují složitější javascriptový kód. Jelikož je však zpracování JavaScriptu plně v režii knihovny použité pro práci s webovými stránkami, není možné ho nějakým způsobem ovlivnit. Řešením je tedy výměna knihovny. Ideální možností je využít pro provádění navigačních aktivit přímo prohlížeče. Problémem však je, že přístup k prohlížeči pomocí rozhraní pro jazyk Java je komplikovaný a neposkytuje všechny potřebné prostředky. Bylo by tedy nutné nepodporované, avšak potřebné, záležitosti nějakým způsobem vyřešit.

Aplikace není přímo zaměřena na extrakci dat, proto by bylo vhodné tyto možnosti rozšířit. Mohlo by se jednat například o podporu XQuery dotazů pro vytváření výstupu. Pomocí něj je možné vybrat požadované uzly a na základě nich s využitím zadané šablony vytvořit výstup požadovaného tvaru. V současném stavu umožňuje mechanismus výběr informací na základě XPath výrazu a jejich export.

Další možností rozšíření extrakce dat je poskytnutí sady funkcí pro vytváření XML elementů. V tom případě by bylo možné s využitím dat získaných ze stránky vytvořit požadovanou strukturu.

Literatura

[1] Hruška, T. *Internetové aplikace I.: Internet a WWW.* 2007. Opora k předmětu WAP na FIT VUT v Brně.

[2] *Hypertext Transfer Protocol – HTTP/1.1, RFC 2616* [online]. June 1999 [cit. 2011-12-22]. Dostupné na: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>.

[3] *HTML 4.01 Specification* [online]. W3C Recommendation 24 December 1999 [cit. 2011-12-22]. Dostupné na: <http://www.w3.org/TR/html401>.

[4] *XPath Tutorial* [online]. © 1999-2010 [cit. 2012-01-02]. Dostupné na: <http://www.w3schools.com/XPath/default.asp>.

[5] Harold, R. *XML v kostce.* 1. vydání. Praha: Computer Press, 2002. 438 s. ISBN 80-7226-712-4.

[6] Marchal, B. *XML v příkladech.* 1. vydání. Praha: Computer Press, 2000. 447 s. ISBN 80-7226-332-3.

[7] *Document Object Model (DOM)* [online]. January 19, 2005 [cit. 2012-01-02]. Dostupné na: <http://www.w3.org/DOM>.

[8] Wolter, J. *A Guide to Web Authentication Alternatives* [online]. Last Updated Oct 2003. [cit. 2011-12-17]. Dostupné na: <http://unixpapa.com/auth/index.html>.

[9] *HTTP Authentication: Basic and Digest Access Authentication, RFC 2617* [online]. June 1999 [cit. 2011-12-17]. Dostupné na: <http://www.ietf.org/rfc/rfc2617.txt>.

[10] *Session Tracking Methods* [online]. 31/05/2008 [cit. 2011-12-18]. Dostupné na: <http://javapapers.com/servlet/explain-the-methods-used-for-session-tracking>.

[11] Baumgartner, R., Flesca, S. a Gottlob, G. Visual Web Information Extraction with Lixto. In *Proceedings of the 27th International Conference on Very Large Data Bases.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. S. 119–128. VLDB '01. ISBN 1-55860-804-4.

[12] Baumgartner, R., Ceresna, M., Gottlob, G. et al. Web information acquisition with Lixto Suite: a demonstration. In *Data Engineering, 2003. Proceedings. 19th International Conference on.* March 2003. S. 747 – 749.

[13] Puna, P. *Extrakce dat z dynamických WWW stránek.* Brno: FIT VUT v Brně, 2009. Diplomová práce.

[14] Zapletal, V. *Nástroj pro online analýzu vybraných dat z webových stránek.* Brno: FIT VUT v Brně, 2010. Bakalářská práce.

[15] Kalus, J. *Aplikace pro získávání a zpracování dat z tabulek na webových stránkách.* Brno: FIT VUT v Brně, 2009. Bakalářská práce.

[16] Böck, H. *The Definitive Guide to NetBeans Platform.* 1st edition. New York: Apress, 2009. 370 s. ISBN 978-1-4302-2417-4.

[17] *HtmlUnit* [online]. © 2002-2011 [cit. 2012-05-02]. Dostupné na: <`http://htmlunit.sourceforge.net/`>.

Seznam příloh

- Příloha A – Uživatelský návod k aplikaci
- Příloha B – Zdrojový kód vytvořené stránky pro přidání osoby
- Příloha C – Obsah přiloženého CD

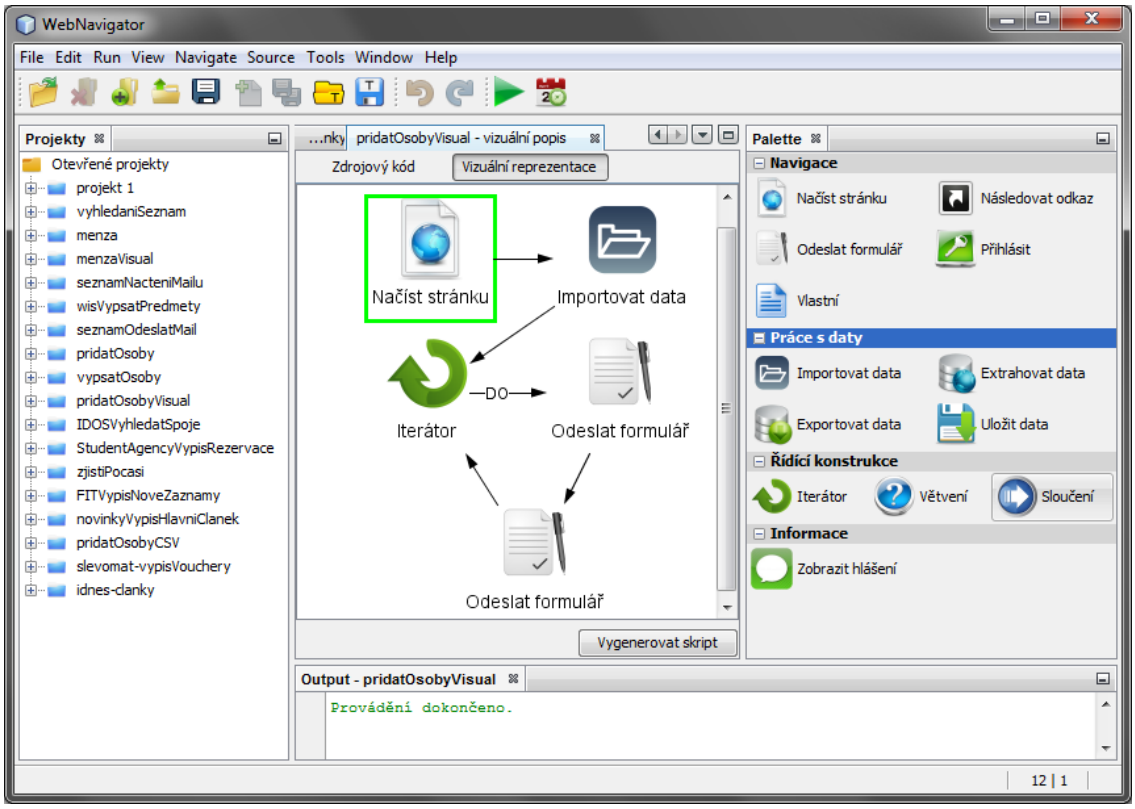
Příloha A

Uživatelský návod k aplikaci

V této příloze je uveden uživatelský návod k aplikaci. Je popsána správa projektů, vytváření skriptu a vizuálního popisu navigace, možnosti usnadnění tvorby popisu a spouštění vytvořených projektů.

A.1 Hlavní okno aplikace

Hlavní okno je zobrazeno na obrázku A.1. Lze ho rozdělit na několik oblastí.



Obrázek A.1: Hlavní okno aplikace

Vlevo je k dispozici seznam vytvořených projektů. V centrální části se nachází editor pro vytváření skriptu a vizuálního popisu navigace. Je zde však také možné zobrazit náhled

načtené stránky či její zdrojový kód. V pravé části se zobrazuje strom elementů načtené stránky. V případě vytváření vizuálního popisu se zde také zobrazí paleta s dostupnými uzly, pomocí kterých lze popis vytvářet. Ve spodní oblasti se nachází konzole pro výpis různých informací, například chybových hlášení nebo informací vypisovaných při provádění skriptů.

A.2 Správa projektů

Projekt je možné přidat vybráním položky *Vytvořit projekt* v menu *File* nebo odpovídajícího tlačítka v nástrojové liště. Je zobrazen dialog pro zadání názvu projektu a vybrání složky pro jeho umístění. Je potřeba ji nejprve vytvořit a poté vybrat. Po potvrzení se nový projekt objeví v seznamu v levé části hlavního okna. Okno pro editaci popisu navigace v rámci projektu lze otevřít poklepáním na něj. Uzel s projektem lze rozbalit, v tom případě se zobrazí položky pro zdrojový kód a vizuální popis. Poklepáním na ně lze přejít k editaci příslušného popisu navigační aktivity.

Dříve vytvořený projekt je možné otevřít pomocí položky *Otevřít projekt* v menu *File*. V zobrazeném dialogu je třeba nalézt složku s požadovaným projektem a vybrat soubor `project.xml`.

Zavření projektu je možné provést jeho vybráním v seznamu a volbou položky *Zavřít projekt* v menu *File*. Lze to rovněž provést přes kontextové menu uzlu projektu. Projekt je tak odstraněn ze seznamu otevřených projektů. Související soubory zůstanou zachovány.

A.3 Vytváření popisu navigace

Navigační aktivitu je možné popsat dvěma způsoby, pomocí skriptu či vizuálního popisu. Pro vytváření obou způsobů popisu jsou dostupné editory. Po otevření projektu lze mezi oběma způsoby přepínat pomocí záložek v horní části okna pro editaci projektu.

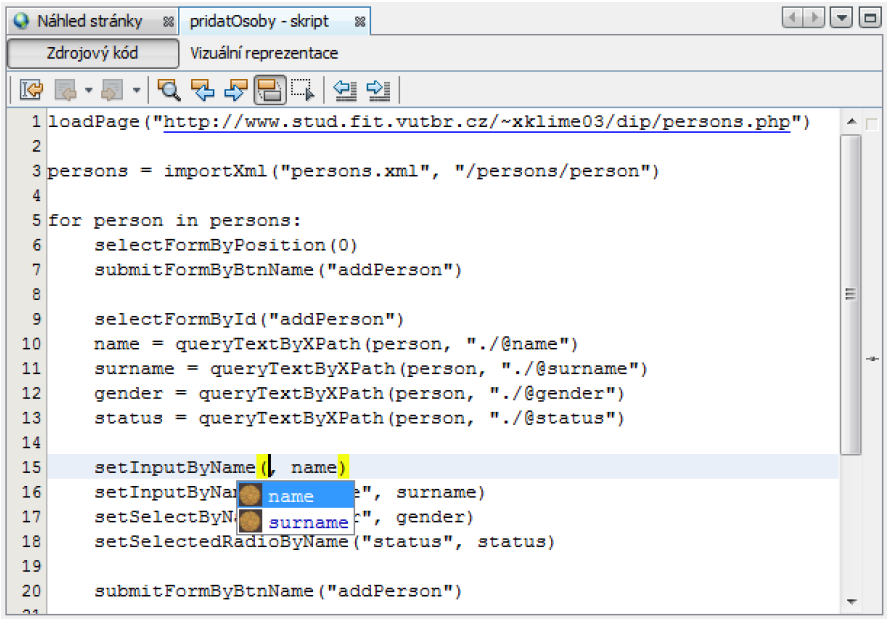
A.3.1 Skript

Pro zadávání příkazů skriptovacího jazyka je dostupný textový editor. Editor podporuje běžné operace, jako jsou zpět a vpřed, kopírování a vkládání, vyhledávání, nahrazování apod. Při zadávání vestavěných funkcí lze pomocí klávesové zkratky CTRL+mezerník vyvolat okno pro automatické doplňování jejich názvů. Jeho ukázka je na obrázku A.2.

U některých funkcí jsou napovídány i jejich parametry. Jedná se například o funkci pro následování odkazu dle textu, kdy jsou napovídány texty všech odkazů obsažených v aktuální stránce. Napovídat lze například i hodnoty atributů prvků stránky při manipulaci s nimi. Při výběru formuláře dle jeho atributu `name` jsou napovídány hodnoty tohoto atributu u všech formulářů vyskytujících se na stránce. Podobně jsou hodnoty napovídány i při práci s prvky formulářů. U prvku typu `select` jsou napovídány i hodnoty, které nabízí pro výběr.

Pro napovídání je však potřeba načíst stránku, na základě níž je nápověda realizována. Při práci s prvky formuláře musí být požadovaný formulář označen jako aktivní. Pro načtení stránky a výběr formuláře slouží vestavěné funkce. Po jejich zadání ve skriptu je ho potřeba spustit. Tím je potřebná stránka načtena nebo formulář zvolen a pak lze zobrazovat napovídané položky při zadávání dalších příkazů.

Při vytváření XPath výrazu, který může být zadán jako parametr některé z vestavěných funkcí, jsou napovídány možnosti, jak v jeho tvorbě pokračovat. Nabízeny jsou elementy



Obrázek A.2: Nápověda atributu `name` u textového pole při vytváření skriptu

nebo atributy, které se nachází v části stromu elementů dané dosud napsanou částí výrazu. Napovídání je prováděno na základě aktuálně načtené stránky nebo XML souboru. Záleží na tom, v jakém kontextu je XPath výraz zadáván.

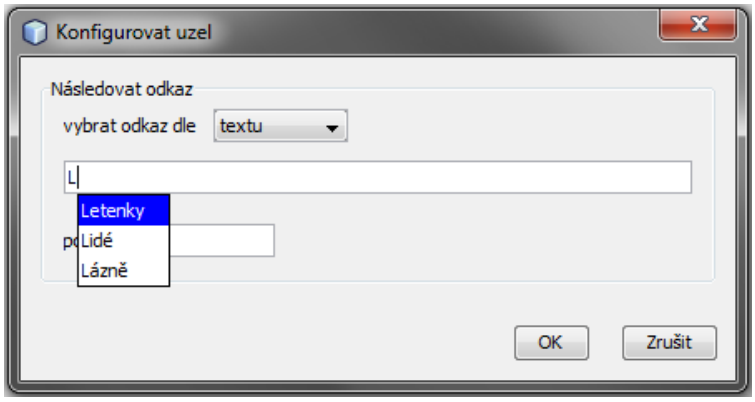
A.3.2 Vizuální popis

V případě vizuálního popisu je k dispozici pracovní plocha pro jeho vytváření. Uzly lze přidávat přetažením z palety dostupných typů uzlů, která je zobrazena v pravé části okna aplikace. Uzly je možno po pracovní ploše posouvat tažením myši. Jejich propojení je možné provést tažením myši při stisknuté klávese CTRL. Uzly reprezentují provedení určitého navigačního kroku, hrany udávají následnost provádění uzlů. Smazání uzlu i hrany je možné pomocí jejich kontextových menu. Přes kontextové menu je také potřeba nastavit počáteční uzel navigační aktivity. Ukázka editoru vizuálního popisu s připravenou navigační aktivitou je vidět na obrázku A.1, kde je zobrazeno hlavní okno aplikace.

Z některých uzlů může vést více hran. V případě iterátoru je to hrana `DO`, která vede směrem k jeho tělu prováděnému v každé iteraci. Na jeho konci se hrana vrací zpět do uzlu iterátoru. Hrana `DONE` udává uzel, kterým se pokračuje po provedení všech iterací. V případě větvení jsou to hrany `TRUE` a `FALSE` směřující k větvím provedeným na základě výsledku podmínky. Obě větve musí být spojeny pomocí uzlu sloučení.

Zobrazení dialogu pro konfiguraci uzlu se provede pomocí výběru odpovídající položky v kontextovém menu uzlu nebo pomocí dvojkliku na něj. Pro lepší přehlednost uzlů ve scéně je možné upravit jejich popisky. Provede se to pomocí dvojkliku na popisek.

Při vytváření vizuálního popisu je rovněž dostupná nápověda různých položek. Je realizována zobrazením okna s nabízenými možnostmi u textových polí v dialozích pro konfiguraci uzlů. Případy, ve kterých je nápověda využitelná, je stejná jako při vytváření skriptu. Na obrázku A.3 je ukázka nápovědy textů odkazů na stránce portálu Seznam.



Obrázek A.3: Nápověda textů odkazů na stránce portálu Seznam

Generování skriptu z vizuálního popisu

Vygenerování skriptu odpovídajícímu vytvořenému vizuálnímu popisu lze provést pomocí tlačítka *Vygenerovat skript* v pravé dolní části editoru. Vygenerovaný skript je přímo vložen do editoru pod záložkou *Zdrojový kód*. Původní obsah editoru je přepsán. Vygenerované příkazy lze libovolným způsobem upravit. Změny však nejsou reflektovány zpět do vizuálního popisu. Pokud je poté upraven i vizuální popis a znovu vygenerován skript, dříve provedené úpravy v něm jsou ztraceny.

Při spuštění se vždy provádí interpretace skriptu. Pro spuštění vizuálního popisu je tedy nejprve nutné nechat odpovídající skript vygenerovat. Generování skriptu při spuštění není prováděno automaticky. Je to tak z toho důvodu, aby bylo možné vygenerovaný skript upravit a spouštět upravenou variantu.

A.3.3 Usnadnění vytváření popisu

Pro usnadnění vytváření popisu je k dispozici náhled načtené stránky, strom jejích elementů a zobrazení jejího zdrojového kódu.

Náhled stránky a strom jejích elementů

Náhled stránky a strom jejích elementů jsou spolu provázány. Při kliknutí pravým tlačítkem myši do zobrazení stránky je odpovídající element vybrán ve stromu elementů. Naopak, pokud je ve stromu elementů nějaký vybrán, lze jeho výskyt zvýraznit v zobrazení stránky změnou jeho pozadí a orámováním. Takto lze vybrat a zvýraznit i více elementů najednou.

Pokud je konfigurován uzel pro práci s formulářem, lze se přepnout do zobrazení stránky a kliknutím pravým tlačítkem do prostoru formuláře nechat vygenerovat jemu odpovídající XPath výraz. Pokud formulář obsahuje atribut `name` nebo `id`, identifikuje XPath výraz formulář na základě tohoto atributu. Jinak je vygenerován výraz popisující kompletní cestu od kořene stránky k danému formuláři.

V okně se zobrazením stromu elementů stránky lze zadat XPath výraz a nechat ve stromu vybrat elementy, které mu odpovídají. Ty je pak případně možné i zvýraznit v zobrazení stránky.

Zobrazení zdrojového kódu stránky

Při zobrazení zdrojového kódu stránky je možné nechat v něm zvýraznit nalezené výskyty zadaného regulárního výrazu. Při konfiguraci uzlu pro extrakci dat se lze při vytváření regulárního výrazu hned přepnout do zobrazení zdrojového kódu stránky, výraz si nechat zvýraznit a případně ho upravit. Poté se lze rychle přepnout zpět ke konfiguraci uzlu.

A.4 Spouštění projektů

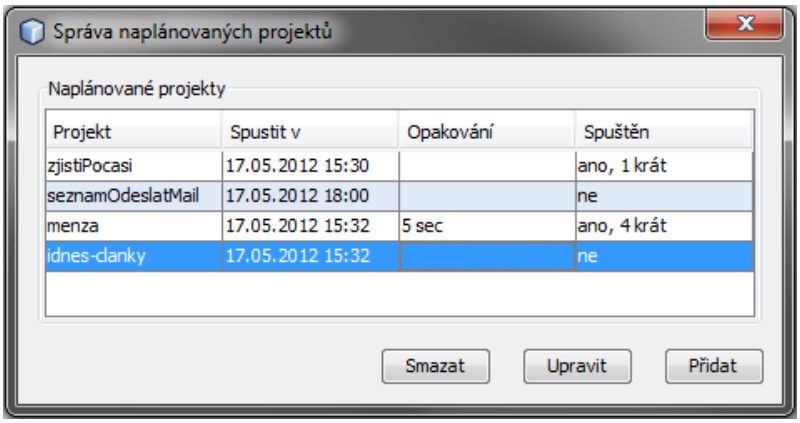
Spuštění vytvořeného projektu je možné třemi způsoby, které budou nyní popsány.

A.4.1 Režim návrhu a ladění

Spuštění aktivního projektu se provede pomocí položky *Spustit projekt* v menu *Run*, odpovídajícím tlačítkem v nástrojové liště nebo pomocí klávesové zkratky F5. Chybová hlášení a informace vypsané v rámci skriptu se při tomto způsobu spuštění zobrazují v konzoli ve spodní části aplikace. Poslední načtená stránka je po dokončení provádění zachována a je použita pro nápovědu při další tvorbě skriptu.

A.4.2 Plánované spuštění

Spuštění projektu je možné naplánovat. K tomu slouží dialog pro správu naplánovaných projektů. Jeho ukázka je na obrázku A.4.



Obrázek A.4: Dialog pro správu naplánovaných spuštění projektů

Přidání naplánovaného spuštění projektu je možné pomocí příslušného dialogu. Ze seznamu je vybrán plánovaný projekt, je zadán čas prvního spuštění a případně také doba, po které se má spuštění opakovat.

Všechny naplánované projekty, včetně již spuštěných, jsou zobrazeny v tabulce. V posledním sloupci je zobrazena informace, zda byl projekt již spuštěn, případně kolikrát v případě opakovaného spouštění.

Při spouštění takto naplánovaných projektů není žádným způsobem ovlivněna aktuální stránka, na základě níž je prováděna nápověda při vytváření skriptu. Chybová hlášení a informace vypisované v rámci skriptu nejsou zobrazovány v konzoli, ale jsou ukládány do souboru, který je vytvořen ve složce `run` v adresáři s projektem. Při každém spuštění je ukládání provedeno do nového souboru. Ten je pojmenován na základě data a času spuštění.

A.4.3 Konzolová aplikace

Vytvořený skript je možné spustit také pomocí konzolové aplikace. Ta je připravena ve formě *Java Archive* souboru. Při spuštění je nutné předat jí jako parametr název souboru se skriptem, který se provede. Aplikace však akceptuje ještě několik dalších parametrů. Jsou uvedeny v tabulce A.1.

`-e kodovani`	kódování interpretovaného souboru
`-js`	vykonávání JavaScriptu při načítání stránek

Tabulka A.1: Parametry konzolové aplikace

Příklad spuštění konzolové aplikace je uveden v ukázce A.1.

```
java -jar WebNavigatorConsole.jar -e utf-8 -js skript.src
```

Ukázka A.1: Příklad spuštění konzolové aplikace

Informace vypisované v rámci prováděného skriptu, resp. chybová hlášení, jsou zapisována na standardní, resp. standardní chybový výstup.

A.5 Instalace a spuštění desktopové aplikace

Aplikaci není potřeba žádným způsobem instalovat. Je již distribuována ve spustitelné podobě. Pouze je vhodné zkopírovat její adresářovou strukturu do umístění, ve kterém je povoleno zapisovat soubory. Spuštění aplikace se v případě operačního systému Windows provede pomocí souboru `webnavigator.exe`, v případě Linux pomocí souboru `webnavigator`. Oba soubory se nacházejí ve složce `bin`.

Příloha B

Zdrojový kód stránky pro přidání osoby

```
<html>
  <head>
    <meta http-equiv="content-type"
          content="text/html; charset=windows-1250">
    <title>Přidat osobu</title>
  </head>
  <body>
    <h1>Přidat osobu</h1>
    <form id="addPerson" name="addPerson" method="get"
          action="persons_add.php">
      <table>
        <tr>
          <td><label for="name">jméno</label></td>
          <td><input type="text" name="name" id="name"/></td>
        </tr>
        <tr>
          <td><label for="surname">příjmení</label></td>
          <td>
            <input type="text" name="surname" id="surname"/>
          </td>
        </tr>
        <tr>
          <td><label for="gender">pohlaví</label></td>
          <td>
            <select name="gender" id="gender">
              <option value="muž">muž</option>
              <option value="žena">žena</option>
            </select>
          </td>
        </tr>
        <tr>
          <td>stav</td>
          <td>
            <input type="radio" name="status" id="married"
                   value="manželství" checked="checked"/>
            <label for="married">manželství</label>
```

```
            <input type="radio" name="status" id="single"
                   value="svobodný"/>
            <label for="single">svobodný</label>
          </td>
        </tr>
        <tr>
          <td></td>
          <td>
            <input type="submit" name="addPerson" id="addPerson"
                   value="Přidat osobu"/>
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

Příloha C

Obsah přiloženého CD

Adresář	**Obsah**
app	spustitelná aplikace
doc/application	programová dokumentace aplikace
doc/pdf	technická zpráva v PDF formátu
doc/src	zdrojové kódy technické zprávy ve formátu LaTeX
examples	soubory připravených projektů
src	zdrojové kódy aplikace jako projekt NetBeans IDE 7.1