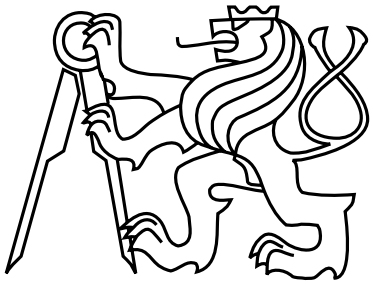
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Rozšíření ExtBrain Communicatoru o další komunikační protokoly

Yun Ruan

Vedoucí práce: Ing. Tomáš Novotný

Studijní program: Elektrotechnika a informatika, strukturovaný, Bakalářský

Obor: Výpočetní technika

28. května 2010

Poděkování

Chtěl bych tímto poděkovat vedoucímu své bakalářské práce Ing. Tomáši Novotnému za cenné připomínky a rady při vypracovávání práce, své rodině za podporu, Luboru Petrovi za překlad a korekturu a samozřejmě také Florian Quèze, hlavnímu vývojáři Instantbirdu.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 28. 5. 2010 .....

Abstract

The aim of this project is to bring protocol libpurple from Instantbird to mail client Mozilla Thunderbird 3 and demostrate the functionality that Instantbird offers.

Abstrakt

Cílem projektu je přenést protokol libpurple od Instantbirdu do poštovního klienta Mozilla Thunderbird 3 a demonstrovat tak funkcionality, které Instantbird nabízí.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

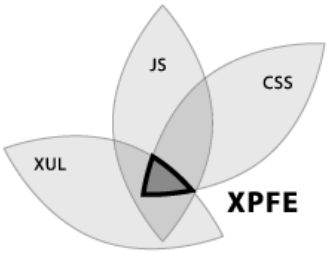
V dnešním době většina lidí vlastní několik emailových účtů od různých poskytovatelů. Kvůli tomu se musí dennodenně vícekrát přihlašovat k serveru svého poskytovatele a musí používat webové emailové rozhraní. Pokud čteme emaily jednou za čas, jistě to nevadí, avšak pokud čteme několik desítek emailů denně, brzy zjistíte, že to není zrovna nejlepší řešení. Proto jsou zde poštovní tlustí klienti, kteří nám slouží k přijímání a odesílání elektronické pošty. Jeden z nich je Mozilla Thunderbird.

Většina z nás také vlastní alespoň jeden IM účet, přes který komunikujeme se svými přáteli, kteří jsou právě připojeni. Lze jim posílat zprávy, přeposílají soubory atd.

Nicméně zatím žádný klient nenabízí služby jako je propojení emailu a IM služeb. Pokud chceme komunikovat pomocí IM služeb, musíme si většinou nainstalovat speciálního klienta. Nabízí se otázka: Proč bychom nemohli sloučit služby jednotlivých klientů do do jednoho programu? Přineslo by nám to komfort v tom, že při čtení emailů bychom mohli být zároveň v kontaktu s přáteli a nemusíme přepínat mezi těmito programy.

1.1 Mozilla application framework

Mozilla application framework[7] je multiplatformní sada softwarových komponent pro vývoj Mozilla aplikací. Původní název XPFE(viz 1.1), zkratka pro Cross Platform Front End, též známý pod názvem XPToolkit.



Obrázek 1.1: XPFE

Komponenty Mozilla application framework jsou:

- **Gecko** - renderovací jádro, navrženo pro výkon a přenositelnost.
- **XML-based User-interface Language (XUL)** - základ uživatelského rozhraní. Základem je jazyk XML[1], pomocí něhož mapujeme různé prvky uživatelského rozhraní jako je widget, ovládací prvky, šablony a další. Syntaxe je velice podobná HTML[2].
- **Necko** - poskytuje rozšiřitelné, platformě nezávislé API[3] pro komunikaci na síti.
- **eXtensible Binding Language (XBL)** - dovoluje definovat vlastní widget pro použití v XUL.
- **Cross Platform Component Object Model (XPCOM)** - multiplatformní komponentový model. Je podobný Microsoft COM[4], podrobně bude popsán v následujících kapitolách.
- **XPConnect** - spojuje JavaScript s XPCOM, umožňuje použít XPCOM komponenty z javascriptového kódu a interakci objektů zevnitř komponent XPCOM.
- a další ...

Mozilla klade velký důraz na multiplatformnost. Výsledkem je, že funguje na skoro všech operačních systémech: Windows, Linux, MacOS, FreeBSD, a další. Její hlavní vývojový jazyk je C++ ale i tak je větší část napsána v JavaScriptu.

[1] XML (**Extensible Markup Language** - obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C(World Wide Web Consortium).

[2] HTML (**HyperText Markup Language**) - další značkovací jazyk používaný především v internetových aplikacích

[3] API (**Application Programming Interface**) - rozhraní pro programování aplikací.

[4] COM (**Component Object Model**) - technologie programových komponent od firmy Microsoft.

1.2 Mozilla Thunderbird

Mozilla Thunderbird[12] je svobodný, open source, multiplatformní poštovní klient, distribuovaný pod licencí MPL/GPL/LGPL[5]. Původně byl vyvíjen společností Mozilla, nyní jeho vývoj převzala dceřiná společnost Mozilla Messaging.

Program je k dispozici hned v několika jazycích s jednoduchým ovládáním i nastavením. Navíc můžeme doinstalovat různé rozšíření dle potřeby, které jsou dostupné na oficiálních stránkách[11].

Thunderbird nabízí tyto standardní funkcionality:

- **Message management** - správa více e-mail, news feed atd.
- **Junk filtering** - obsahuje bayesovský antispamový filtr.
- **Extensions** - umožňuje přidávání funkcí pomocí instalace modulů XPInstall.
- **Themes** - podporuje skinovatelnost uživatelského rozhraní.
- **Standards support** - standardně podporuje komunikační protokoly POP[6] a IMAP[7]

1.3 Instantbird

Instantbird[5] je IM[8] klient využívající knihovnu libpurple[9], podporující veliký počet komunikačních protokolů jako ICQ, AIM, Jabber a další.

Je to svobodný a open source software a šířený pod licencí MPL/GPL/LGPL jako Mozilla Thunderbird.

[5] MPL (**Mozilla Public License**), GPL (**GNU General Public License**), LGPL (**GNU Lesser General Public License**) - jsou to licence pro svobodný software.

[6] POP (**Post Office Protocol**) - internetový protokol, který se používá pro stahování emailových zpráv ze vzdáleného serveru na klienta.

[7] IMAP (**Internet Message Access Protocol**) - internetový protokol pro vzdálený přístup k e-mailové schránce.

[8] IM (**Instant messaging**) - internetová komunikace v reálném čase.

[9] libpurple - Knihovna, která je používaná jako jádro IM programů, např. Pidgin, Adium, Instantbird.

Kapitola 2

Specifikace cíle, použité nástroje

2.1 Cíl projektu

Cílem tohoto projektu je rozšířit projekt ExtBrain communicator o další komunikační protokoly, které podporuje knihovna libpurple. Nejznámější z nich jsou například: MSN, Jabber a ICQ.

Během zkoumání fungování interních modulů Instantbirdu, jsem zjistil, že pro můj projekt je výhodné použít framework **xpcompurple**. Ten v sobě zapouzdřuje knihovnu libpurple pomocí frameworku XPCOM. Xpcompurple vychází z Pidgin libpurple a je připraven k použití v Mozilla aplikací.

2.1.1 Porovnání existujících implementací

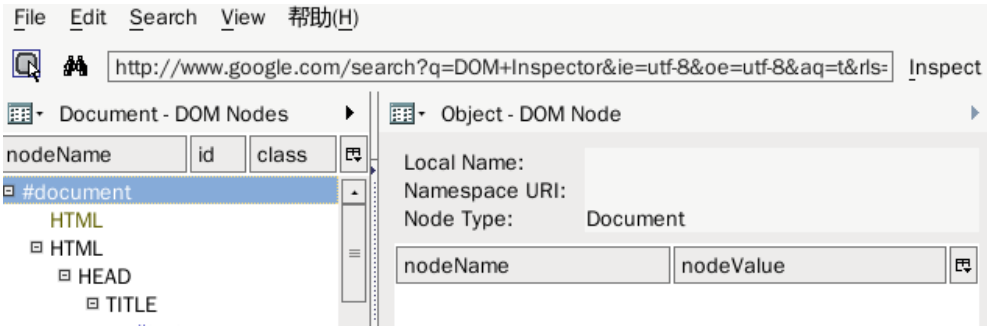
Existující řešení jsem prozatím na stránkách Thunderbird add-ons[11] nenašel. Nemůžu tedy porovnat mojí verzi aplikace s existujícími řešeními.

2.2 Použité nástroje

2.2.1 DOM Inspector

DOM inspector[1] je nástroj, který může být použit ke kontrole a úpravě DOM[1] v XUL nebo v jiných webových dokumentech. V DOM inspektoru je možné zobrazit stromovou strukturu jednotlivých XML elementů. Označením konkrétního uzlu můžeme zobrazit obsah daného elementu. Zobrazení elementů ve stromové struktuře je velice praktické při ladění složitějších stránek, screenshot programu 2.1.

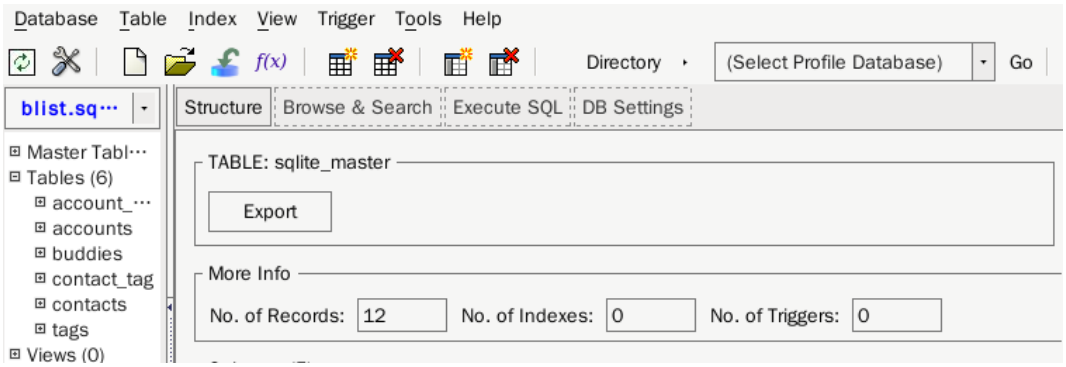
[1] DOM (**Document Object Model**) - je objektově orientovaná reprezentace XML nebo HTML dokumentu.



Obrázek 2.1: DOM inspector

2.2.2 SQLite Manager

SQLite Manager[10] je jednoduchý a užitečný nástroj pro správu SQLite[2] databází, umožňující procházení a editaci databází, spouštění SQL[3] příkazů a intuitivní zobrazování databázových objektů ve stromové hierarchii, screenshot programu 2.2.



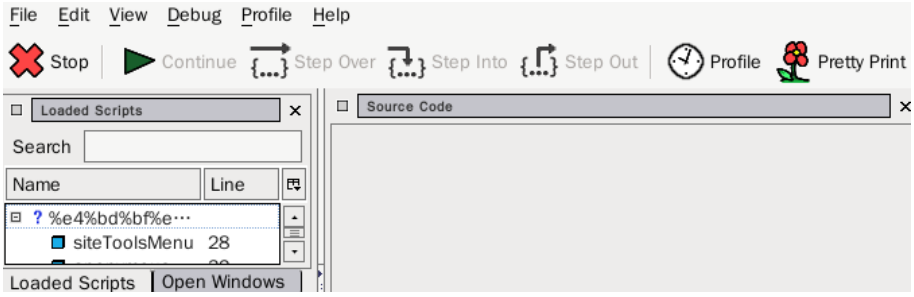
Obrázek 2.2: SQLite manager

2.2.3 Venkman

Venkman[13], známý též jako JavaScriptový debbuger pro Mozilla aplikace, je užitečný nástroj, který umožňuje krokování, nastavování watches a další funkce nezbytné pro komfortní ladění vlastního kódu v JavaScriptu. Vekman je navržen pro Thunderbird, Firefox a další Mozilla aplikace, screenshot programu 2.3.

[2] SQLite - relační databázový systém obsažený v relativně malé knihovně napsané v C.

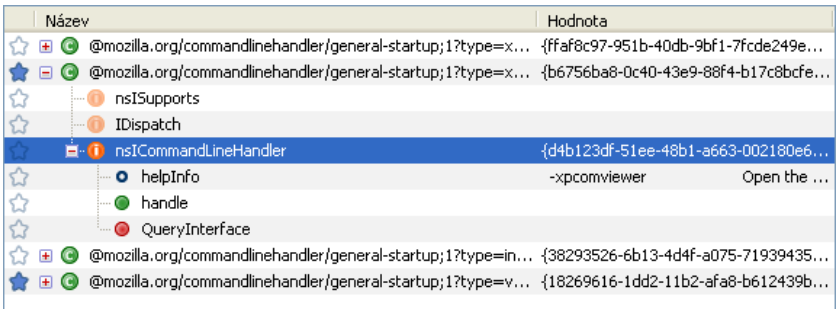
[3] SQL (**Structured Query Language**) - standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.



Obrázek 2.3: Venkman

2.2.4 XPCOM Viewer

XPCOM viewer[17] je nástroj pro zkoumání XPCOM komponent. Nabízí snadné a rychlé vyhledávání XPCOM tříd, rozhraní a aplikačních logů, které jsou součástí dané Mozilla aplikace. Dalšími nabízenými funkcemi je např. vyhledávací filtr, procházení vytvořených objektů s možností zobrazení jejich atributů, export zobrazených dat do HTML, XML nebo textu, screenshot programu 2.4.



Obrázek 2.4: XPCOM viewer

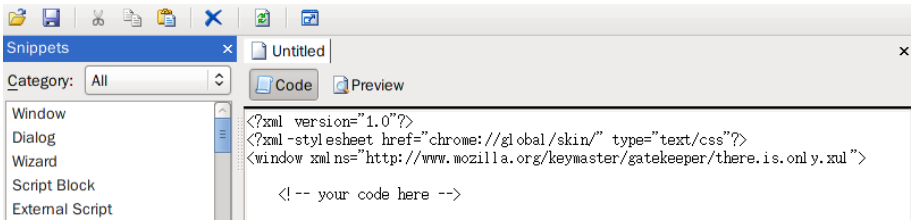
2.2.5 XUL explorer

XUL explorer[19] poskytuje snadný způsob, jak experimentovat s XUL. Je to jednoduchý editor, kterým lze zobrazovat XUL jak inline tak v samostatném okně.

Pro zrychlení práce v sobě obsahuje panel se seznamem **Code snippets**[4], které mohou být rychle vloženy do editoru. Programátor tak nemusí psát velmi často se opakující části kódu a pouhým poklikáním vloží do rozepsané části hotový fragment XUL kódu.

XUL explorer v sobě obsahuje nástroje jako XUL validator, DOM inspector, Error console, JavaScript debugger, které jsou k dispozici pro ladění chyb. Kromě toho také nabízí přístup k XUL informacím na Mozilla Developer Center, screenshot programu 2.5.

[4] **Code snippets** - malé fragmenty XUL nebo JavaScript



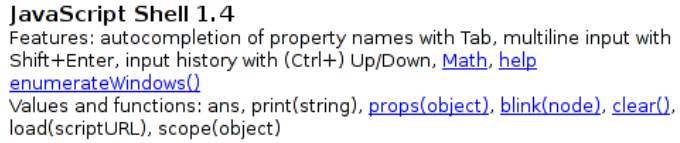
Obrázek 2.5: XUL explorer

2.2.6 Extension Developer's Extension

Extension Developer's Extension[2] je nástroj, který usnadňuje vývojáři psaní rozšíření pro Mozilla Firefox, Thunderbid atd. Nabízí mnoho funkcí, zde popíši jen ty, které jsem při vývoji použil.

2.2.6.1 Javascript Shell

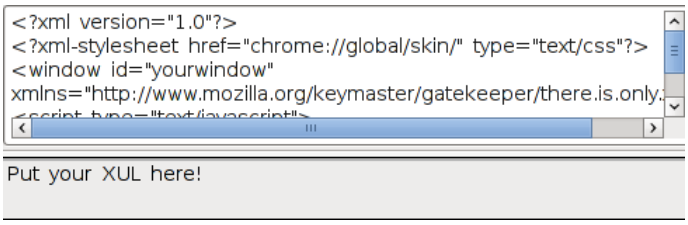
Javascript Shell 2.6 umožňuje spouštět JavaScript interaktivně, experimentovat s XPCOM v shellu. Můžeme spouštět JavaScript, jako by byl umístěn přímo v daném dokumentu.



Obrázek 2.6: Javascript Shell

2.2.6.2 XUL editor

XUL editor nám opět umožňuje editovat XUL a zobrazovat výsledný vzhled dokumentu v reálném čase. XUL editor je zjednodušená varianta XUL exploreru, nenabízí tedy takové množství funkcionalit jako XUL explorer, screenshot programu 2.7.



Obrázek 2.7: XUL editor

Kapitola 3

Skriptování v Mozille

Pokud programujeme aplikace založené na Mozille je třeba se seznámit s XPFE[9]. Tento framework poskytuje řadu již implementovaných funkcionalit jako například autentizaci, volání XPCOM objektů, volání obsluhy událostí, atd. Usnadňuje tak programátorovi integraci nových funkcionalit do Mozilly.

Pro skriptování pod frameworkem XPFE je používán JavaScript. Má bohužel nálepku nesofistikovaného programovacího jazyka, který je především určen pro nasazení na webových stránkách. Tento jazyk má však mnohem širší použití. Disponuje mnoha vlastnostmi podobnými vyšším programovacím jazykům jako je například: modularita, odchytávání výjimek a událostí, podpora regulárních výrazů.

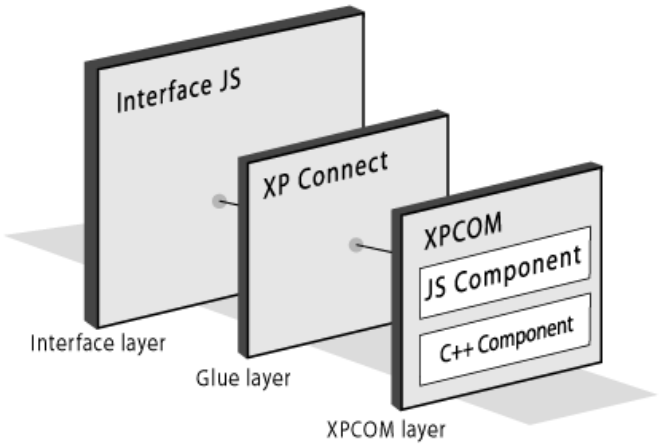
3.1 Popis úrovní skriptování v Mozille

Úrovně skriptování v Mozille[9] můžeme rozdělit do tří vrstev. První vrstva (Interface JS) se stará o manipulaci s objekty pomocí DOM. Druhá vrstva (XPConnect) obsluhuje volání služeb poskytovaných XPCOMem a poslední třetí vrstva (XPCOM) se stará o vytváření XPCOM component. Hierarchii jednotlivých vrstev popisuje obrázek 4.1.

Nyní detailně popíšeme jednotlivé úrovně skriptování.

3.1.1 Interface JS vrstva

Nejvyšší úrovní skriptování je Interface JS. S jeho pomocí můžeme rozšiřovat chování jednotlivých částí aplikace, vytvářet nové widgety, které můžeme navzájem propojovat do jednotného celku. Referenci na atributy jednotlivých komponent získáváme pomocí volání metod založených na DOM. Pokud například chceme pomocí tlačítka editovat vlastnosti textboxu, přidáme do obsluhy události tlačítka volání funkce *getElementByID()* a následně vybranému objektu mohu změnit požadované vlastnosti.



Obrázek 3.1: Úrovně skriptování v Mozille

3.1.2 XPConnect

XPConnect (kapitola 3.3) leží v hierarchii na druhé úrovni, spojuje uživatelské rozhraní s XPCOMem, to znamená že komponenty zabalené ve frameworku XPCOM jsou zpřístupněny pro programování v JavaScriptu.

Pokud JavaScript zavolá nějakou metodu a následně dostane nějaká data od komponenty nižší vrstvy, volání a následné vrácení výsledku je vždy prováděno přes tuto vrstvu.

3.1.3 XPCOM

Jak již bylo několikrát zmíněno, na třetí vrstvě v hierarchii leží framework XPCOM. Jsou v něm zapouzdřeny všechny hlavní třídy a funkce sloužící pro bezproblémový chod vytvářené aplikace. Jinými slovy můžeme říci, že se jedná o jádro Mozilla aplikací. Podrobně se budu XPCOMu věnovat v následujících kapitolách (kapitola 4).

3.2 JavaScript a DOM

Na aplikační úrovni je malý rozdíl mezi internetovou stránkou a běžným grafickým uživatelským rozhraním. HTML DOM a XUL DOM jsou si velice podobné. V obou případech jsou změny stavů a události propagovány přes volání DOM[6].

3.2.1 Co je DOM?

DOM je API používané k přístupu k HTML a XML dokumentům. Pro vývojáře internetových aplikací nabízí dvě hlavní funkcionality. Poskytuje strukturovanou reprezentaci

daného dokumentu a definuje jakým způsobem může být k této struktuře přistupováno z JavaScriptu. Tyto funkcionality dovolují programátorovi manipulovat s uživatelským rozhraním jako se stromem komponent. Dovoluje vytváření nového rozhraní či z již hotového rozhraní odebírat nebo přidávat další elementy.

DOM však není určen jen pro přístup k HTML, XML nebo XUL dokumentům ale můžeme ho použít i pro MathML[1], SVG[2] nebo pro jiné XML dokumentům podobné formáty.

Pokud JavaScriptem vytváříme nový HTML element nebo měníme atribut nějakého XUL objektu, máme k těmto nově vytvořeným prvkům vždy přístup přes objektový model DOMu, který tyto struktury zapouzdřuje.

3.2.2 DOM standardy

Dodržování specifikace DOM je v Mozilla aplikacích rozděláváno do tří úrovní podle stupně jejího dodržování. Každá z úrovní poskytuje vlastní množinu funkcionalit, některé z nich však nejsou dosud Mozillou podporovány.

3.2.3 Metody DOM

Metody v DOMu dovolují programátorovi manipulovat s prvky, které jsou obsaženy v uživatelském rozhraní nebo ve webových stránkách. Umožňují nastavení atributů, vytváření prvků, skrývání prvků a další. DOM zprostředkovává veškerou interakci mezi skripty a rozhraním, např. chceme-li změnit obrázek, když uživatel stiskne tlačítko, DOM zaregistruje událost, vyvolanou stisknutím tlačítka a nastaví požadováné atributy.

Nyní si ukážeme některé metody v DOMu.

3.2.3.1 Výstup na STDOUT pomocí DUMP()

Metoda *dump()* se používá pro tisk dat na STDOUT[3]. Tato metoda je používána hlavně při ladění kódu.

```
Ukázka použití dump():
var assoc = {
    "val"  : "New",
    "number" : 8
};
alert(dump(assoc));
```

```
Výstup:

'val' => "New"
'number' => "8"
```

[1] MathML (**Mathematical Markup Language**) - matematický značkovací jazyk, součástí dokumentů konsorcia W3C (World Wide Web Consortium) jako podmnožina jazyka XML pro zápis matematických a příbuzných vzorců.

[2] SVG (**Scalable Vector Graphics**) - značkovací jazyk a formát souboru, který popisuje dvojrozměrnou vektorovou grafiku pomocí XML.

[3] STDOUT (**Standard output**) - česky: *Standardní výstup*, je to virtuální zařízení, na které program zapisuje výstupní data.

3.2.3.2 getElementById

getElementById(aId) je snad nejčastěji používanou DOM metodu. Je to nejpohodlnější způsob, jak získat referenci na konkrétní prvek s použitím jeho ID jako parametr funkce, kde ID je jedinečný identifikátor tohoto elementu.

```
Ukázka použití getElementById(aID):
<box id="my-id" />

var boxEl = document.getElementById('my-id');
dump("boxEl="+boxEl+"\n");

Výstup:
boxEl=[object XULElement]
```

Hodnota, kterou *getElementById* vrací, je reference na specifikovaný objekt typu **XULElement**. Jakmile je *boxEl* k dispozici, je možné pomocí jiné DOM metody jako *getAttribute* a *setAttribute* změnit pozici a další vlastnosti.

3.2.3.3 getAttribute

Atributy jsou vlastnosti, definované přímo v elementech. XUL prvky mají například atributy: disable, height, style, orient, label a další.

```
<box id="my-id" foo="hello 1" bar="hello 2" />
```

Výše uvedený kód ukazuje element s různými parametry. Řetězce "my-id", "hello 1" a "hello 2" jsou hodnoty jednotlivých atributů. Pokud objekt disponuje nějakými atributy, jejich hodnotu snadno získáme pomocí metody *getAttribute*. Jejím vstupním parametrem je název hledaného atributu. Následující příklad ukazuje její použití.

```
<box id="my-id" foo="this is the foo attribute" />
<script>
  var boxEl = document.getElementById('my-id');
  var foo   = boxEl.getAttribute('foo');
  dump(foo+'\n');
</script>
```

Metoda dump vypíše řetězec "this is the foo attribute", což je hodnota uložené ve *foo*.

3.2.3.4 setAttribute

Jako poslední si ukážeme metodu *setAttribute*. Je určena ke změně hodnoty atributů. Jako např. size, visibility, position, style atd., má dva parametry: název atributu a jeho nová hodnota.

```
<box id="my-id" foo="this is the foo attribute" />
<script>
  boxEl=document.getElementById('my-id');
  boxEl.setAttribute('foo', 'this is the foo attribute changed');
  var foo = boxEl.getAttribute('foo');
  dump(foo+'\n');
</script>
```

Výše uvedený skript změní hodnotu *foo* na "this is the foo attribute changed" a vypíše pomocí *dump()*. Metoda *setAttribute* lze použít i k vytváření nových atributů.

```
<box id="my-id" />
<script>
  boxEl=document.getElementById('my-id');
  boxEl.setAttribute('bar', 'this is the new attribute bar');
</script>

výsledek:
<box id="my-id" bar="this is the new attribute bar" />
```

3.3 XPConnect a skriptovatelné komponenty

XPConnect[18] propojuje JavaScript a uživatelské rozhraní s jádrem aplikací. Zde si ukážeme, jak najít komponenty, které jsou k dispozici programátorovi přes XPConnect, vytváření JavaScriptový objektů a jejich použití v našich aplikacích.

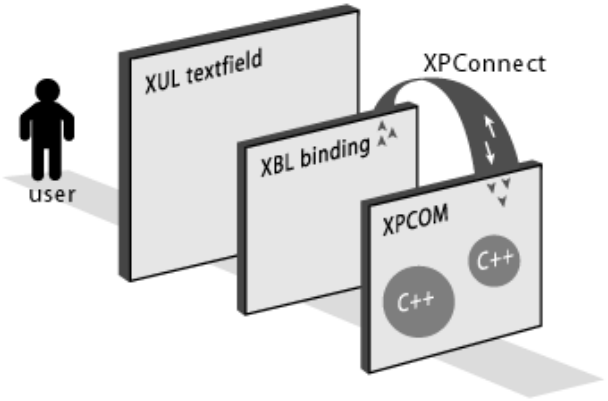
3.3.1 Co je to XPConnect?

Zatím jsme si vystačili se psaním skriptů využívající DOM pro manipulaci s prvky v uživatelském rozhraní. To je však jen začátek. Skriptování v Mozille nabízí mnohem víc. Pravdou je, že funkcionality aplikace jsou pro většinu uživatelů důležitější než vzhled visuálního rozhraní. Kvůli tomu je tady XPConnect a XPCOM. Funkce jako je prohlížení internetových stránek, čtení emailů jsou služby implementované na aplikačním vrstvě. Jsou tedy součástí nižší úrovně. Jsou obvykle napsány v jazyce C++ a uspořádány do modulů, známých jako XPCOM komponenty. Vztah komponent a služeb popisuje obrázek 3.2.

XPConnect propojuje JavaScript a XPCOM komponenty a přibaluje k nim objekty JavaScriptu. Pomocí JavaScriptu a XPConnectu lze vytvářet instance komponent a používat jejich metody a vlastnosti jako když pracujeme s objektem z JavaScriptu.

3.3.1.1 Vytvaření XPCOM objektů ve skriptu

Následující kód demonstruje tvorbu a použití XPCOM komponent v JavaScriptu. Skript vytvoří objekt *filepicker* a používá ho k zobrazování souboru v dialogu filepicker.



Obrázek 3.2: XPConnect

Na prvních dvou řádcích je vidět jak se dialog *fp filepicker* vytváří. Nejdříve se vytvoří instance *nsIFilepicker* a následně se zavolá metoda *createInstance()*, kde *nsIFilepicker* určuje, která komponenta se má vytvořit.

```
// chooseApp:  Open file picker and prompt user for application.
chooseApp: function( ) {
  var nsIFilePicker = Components.interfaces.nsIFilePicker;
  var fp =
     Components.classes["@mozilla.org/filepicker;1"].
         createInstance( nsIFilePicker );
  fp.init( this.mDialog,
     this.getString( "chooseAppFilePickerTitle" ),
     nsIFilePicker.modeOpen );
  fp.appendFilters( nsIFilePicker.filterAll );
  if ( fp.show( ) == nsIFilePicker.returnOK && fp.file ) {
  this.choseApp   = true;
  this.chosenApp  = fp.file;
  // Update dialog.
  this.updateApplicationName(this.chosenApp.unicodePath);
}
```

3.3.1.2 Vyhledávání komponenty a rozhraní

Hledání komponent a rozhraní, pochopení jejich funkce je velice užitečné pro psání Mozilla aplikací. Pro zjednodušení práce existuje webová stránka **Cross-References**(http://mxr.mozilla.org/), sloužící k zobrazení dokumentace a zdrojových kódu Mozilly.

3.3.1.3 Výběr rozhraní z komponent

Ve skriptech můžeme získávat objekty pomocí speciálního objektu *classes* a metody *createInstance()*. Následně můžeme používat metody, které nabízí.

```
var connection = Components.classes
      ["@mozilla.org/network/ldap-connection;1"].
      createInstance(Components.interfaces.nsILDAPConnection);
connection.init(queryURL.host, queryURL.port, null,
              generateGetTargetsBoundCallback( ));
```

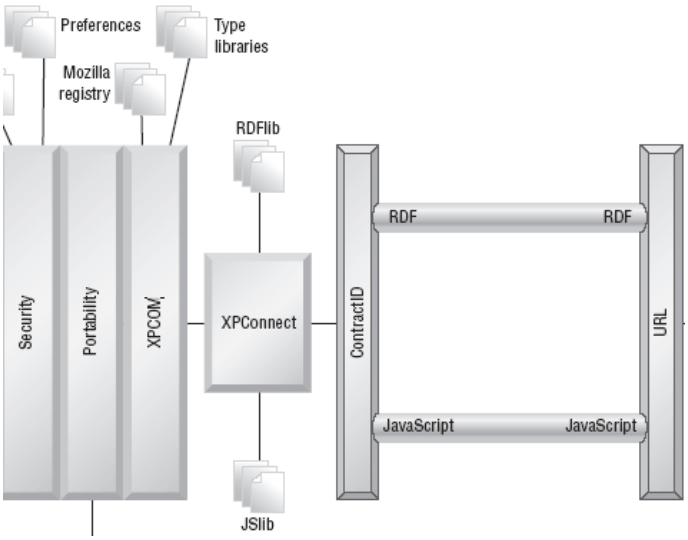
Nejdřív si vezme komponentu, která má CID[4] *ldap-connection;1* a rozhraní *nsILDAPConnection*. Tím vytvoříme nový objekt a můžeme volat jeho metody.

[4]CID (**Contract ID**) - jeden z XPCOM identifikátorů, bude vysvětlen v následujících kapitolách.

Kapitola 4

Úvod do XPCOM

XPCOM[15, 16] je zkratka pro **Cross-platform component object model**. Tento framework je podobný technologii COM od společnosti Microsoft. Podporuje mnoho jazyků, jako např.: JavaScript, Python, Java. Rozhraní XPCOM jsou definována v dialektu IDL[1] nazývané XPIDL[2]. XPCOM poskytuje sadu základních komponent a tříd. Většina propracovanějších komponent v něm však není zahrnuta. Proto musíme využít služeb některých jiných knihoven (např. Gecko, Necko, jiné aplikace, rozšíření).



Obrázek 4.1: Část Mozilla application framework[8]

[1] IDL (**Interface definition language**) - speciální jazyk používaný k popisování rozhraní softwarových komponent.

[2] XPIDL (**Cross Platform Interface Description Language**) - speciální IDL pro XPCOM rozhraní.

4.1 Řešení pomocí XPCOM

XPCOM je framework, který umožňuje vývojářům rozdělit softwarové projekty na menší modulární kousky. Tyto kousky jsou známé jako komponenty, které jsou pak smontované dohromady za běhu programu.

Cílem XPCOM je, aby tyto komponenty mohli být vyvíjeny nezávislé na sobě. XPCOM odděluje implementaci komponent od rozhraní. Pomocí XPCOM vývojáři mohou vytvářet komponenty, které mohou být opakovaně použity v různých aplikacích nebo nahrazovat funkcionality existujících aplikací.

XPCOM nejen podporuje vývoj softwaru, ale také poskytuje řadu pokročilých funkcí, jako například: spravování paměti, správu komponent, abstrakci souborů atd.

4.2 Komponenty XPCOM

Jak již bylo řečeno, XPCOM umožňuje vývojářům rozdělit softwarové projekty na menší kousky. Tyto kousky jsou obvykle malé, opětovně použitelné binární knihovny (DLL[3] ve Windowsu, SO[4] v Linuxu). Mohou obsahovat jednu či více komponent. Když je v knihovně jedena nebo více souvisejících komponent, stává se z knihovny modul.

Rozdělení software do různých komponent ulehčuje jeho vývoj a údržbu.

Výhody rozdělování SW do komponent 4.1:

Benefit	**Popis**
Reuse	Kód je znovupoužitelný v dalších aplikacích.
Updates	Aktualizace komponent bez nutnosti rekompilace celé aplikace
Performance	Moduly, které nejsou nezbytné, mohou být "lazy loaded[5]", nebo nemusí být vůbec načteny, což zvyšuje výkon aplikace.
Maintenance	Usnadňuje hledání a údržbu částí aplikace.

Tabulka 4.1: Výhody použití komponent

4.3 Interface a IDL

Všechna XPCOM rozhraní jsou definována pomocí IDL. K popisování veřejných metod a vlastností komponent pomocí IDL není definován konkretní jazyk. Mozilla ve skutečnosti používá modifikovanou verze IDL, který se jmenuje XPIDL.

[3] DLL (**Dynamic-link library**) - Programové moduly, které mohou být sdíleny více programy, používáno v Microsoft Windows.

[4] SO (**Shared object**) - Linuxová varianta DLL.

[5] Lazy loaded - je postup, kdy kód nebo modul nenahráváme ihned, ale až když jsou potřeba.

Klíčovým rozdílem mezi COM programováním je, že XPCOM odděluje rozhraní a implementaci. Pokud je API odděleno od implementačního jazyka, má uživatel API garanci toho, že rozhraní bude neměnné. COM byl v podstatě vytvořen tak, aby udržoval kompatibilitu na binární úrovni. IDL interface jsou zkompilované do tzv. Type libraries (Typové knihovny).

4.3.1 Interface versus Komponenty

Většina XPCOM komponent implementuje alespoň dvě rozhraní. Každá komponenta potřebuje funkce *QueryInterface*, *AddRef* a *Release*, aby se stala platným XPCOM objektem. Tyto metody jsou zděděné od základního rozhraní *nsISupports* 4.2.

Název	**Typ**	**Popis**	**Parametry/ Návratova hodnota**
AddRef	NS_IMETHOD_(nsrefcnt) AddRef(void)	Zvyšuje čítač referencí rozhraní, přiřazená instance nebude smazána, dokud počet referencí není 0.	vrací: počet referencí
QueryInterface	NS_IMETHOD QueryInterface (REFNSIID aIID, void** aInstancePtr)	Mechanismus pro zjištění rozhraní	Parametry: aIID[in], který je požadované interface IID. aInstancePtr [out], ukazatel na ukazatel rozhraní, který obdrží výsledek. Vrací: NS_OK pokud je rozhraní podporován a NS_NO_INTERFACE když není. NS_ERROR_INVALID_POINTER - aInstancePtr je null.
Release	NS_IMETHOD_(nsrefcnt) Release(void) = 0;	snižuje čítač referencí, pokud je roven 0, tak smaže instance	vrací: počet referencí

Tabulka 4.2: Interface nsISupports

4.3.2 Root interface

QueryInterface, *AddRef* a *Release* jsou implementovány ve všech komponentách. Pokaždé definovat tyto tři metody není zrovna moc efektivní. Proto jsou definovány v *nsISupports* a ostatní rozhraní od něho dědí.

XPIDL podporuje preprocesor jazyka C, používá direktivu #include k vložení jiného IDL souboru, ne jako u Microsoft COM, který používá import.

U všech IDL souborů je nutné v horní části uvést *nsISupports*:

```
#include "nsISupports.idl"
interface nsISimple : nsISupports {
  readonly attribute string value;
};
```

4.3.3 Kompilátor XPIDL

IDL kompilátor je nástroj, který vytvoří binární distribuční soubor s názvem typové knihovny ze zdrojového souboru. XPIDL kompilátor generuje informace o XPCOM rozhraní, hlavičky XPCOM objektů a typové knihovny XPT, z nichž mohou objekty přistupovat přes XPConnect dynamicky. Je schopen vygenerovat i HTML soubor pro dokumentaci, testovací Java class kód a testovací C++ kód.

XPIDL kompilátor je umístěn na: mozilla/xpcom/typelib/xpidl/, lze stáhnout pomocí Mercurial[5] nebo na stránkach Mozilla.org.

Jeho použití je velmi jednoduché. Použití syntaxe a jiné informace o kompilátoru:

```
$ ./xpidl --help
Usage: xpidl [-m mode] [-w] [-v] [-I path] [-o basename] filename.idl
       -a emit annotations to typelib
       -w turn on warnings (recommended)
       -v verbose mode (NYI)
       -I add entry to start of include path for ‘‘#include "nsIThing.idl"’’
       -o use basename (e.g. ‘‘/tmp/nsIThing’’) for output
       -m specify output mode:
          header        Generate C++ header            (.h)
          typelib       Generate XPConnect typelib     (.xpt)
          doc           Generate HTML documentation    (.html)
          java          Generate Java interface        (.java)
```

4.4 Typové knihovny

Základ komponentové architektury XPCOMu je, že jsou soubory nezávislé na jazyku a programovácím prostředí. Tyto soubory jsou zkompilovány do .xpt pomocí XPIDL kompilátoru.

[5] Mercurial - multiplatformní, distribuovaný systém pro správu verzí.

4.4.1 Vytvaření typové knihovny

K vytvoření .xpt(typelib soubor) se používá příznak flag -m typelib se zapnutým warning (-W) a verbose (-v). Aby se soubor přeložil korektně, je nutné uvést kde *nsISupports* leží.

```
# include path to nsISupports.idl
$ $XPIDL_INC = /usr/src/mozilla/xpcom/base
#compile nsISimple.idl
$ xpidl -m typelib -w -v -I $XPIDL_INC \
> -o nsISimple nsISimple.idl
```

4.5 Identifikátory

XPCOM používá dva velmi důležité identifikátory pro rozeznávání komponent a tříd.

4.5.1 Identifikátor tříd

Třída *nsIID* je vlastně typedef pro *nsID* třídu. *nsIID* poskytuje metody jako Equals pro porovnání identifikátorů v kódů.

4.5.2 CID

CID je 128-bitové číslo, které jednoznačně identifikuje třídu nebo komponentu stejným způsobem jako IID identifikuje rozhraní. CID pro *nsISupports* vypadá takto:

```
00000000-0000-0000-c000-000000000046
```

Zápis a délka CID může být nepřehledná a nešikovná s ohledem na kód, proto je velmi často definován následovně:

```
 #define SAMPLE_CID \
{ 0x777f7150, 0x4a2b, 0x4301, \
{ 0xad, 0x10, 0x5e, 0xab, 0x25, 0xb3, 0x22, 0xaa}}
```

Je také možné používat NS_DEFINE_CID. Jednoduché makro pro deklaraci konstant s hodnotou CID:

```
static NS_DEFINE_CID(kWebShellCID, NS_WEB_SHELL_CID);
```

CID je někdy označován jako identifikátor třídy. Pokud třída, na které CID odkazuje, implementuje více než jeden interface, tak CID garantuje, že třída implementuje celou množinu interfaců, když bude Published nebo Frozen.

4.5.3 Contract ID

Contract ID je řetězec používaný k přístupu ke komponentám. Je čitelný a snadno zapamatovatelný. CID nebo Contract ID může být použito k získaní komponenty od Správce komponent. Takhle například vypadá Contract ID pro *LDAP operation component*:

```
"@mozilla.org/network/ldap-operation;1"
```

Formát Contract ID je: doména komponent, modul, název komponenty, číslo verze oddělené lomítky.

Stejně jako CID, Contract ID odkazuje na implementaci než na rozhraní, ne jako IID (interface ID). Ale Contract ID není omezen žádnou specifickou implementací, jako CID, je tedy obecnější. Contract ID specifikuje pouze danou množinu interfaců, kterou chce implementovat a číslo jiné CID mohou být doplněná. To je rozdíl mezi Contract ID a CID, umožňuje překrytí komponent.

4.5.4 Generování identifikátorů

V Unixu/Linuxu, můžete použít program *uuidgen*, spustitelný z console, k vygenerování UUID[6]:

```
$ uuidgen
ce32e3ff-36f8-425f-94be-d85b26e634ee
```

Pro Microsoft Windows existuje program *guidgen.exe*. Můžeme také použít speciálního robota na IRC[7] v *irc.mozilla.org* pomocí příkazu `/msg mozbot uuid`.

4.6 Typy

V XPCOMu existuje mnoho deklaračních typů. Většina z těchto typů má jednoduché mapování. Popišme zde ty nejběžnějších.

4.6.1 Typy metod 4.3

Zajišťují správnost konvence a návratové typy.

4.6.2 Počítání referencí 4.4

Tyto makra správuje počítání referencí.

[6] UUID (**Universally Unique Identifier**) - obecně unikátní identifikátor, kterým mohou být jednoznačně označovány libovolné objekty.

[7] IRC (**Internet Relay Chat**) - jedna z prvních možností komunikace v reálném čase po internetu.

NS_IMETHOD	Deklarace návratového typy.
NS_IMETHODIMP	Implementace návratového typu.
NS_IMETHODIMP_(type)	Speciální případ implementace návratového typu.
NS_IMPORT	Metoda je vyřešena externí knihovnou.
NS_EXPORT	Metoda je řešena za pomoci externí knihovny.

Tabulka 4.3: Typy metod

NS_ADDREF	Volá AddRef na nsISupports objekt.
NS_IF_ADDREF	Stejně jako NS_ADDREF s tím, že zkontroluje null před voláním AddRef.
NS_RELEASE	Speciální případ implementace návratového typu.
NS_IF_RELEASE	Metoda je vyřešena externí knihovnou.

Tabulka 4.4: Počítání referencí

4.6.3 Stavové kódy 4.5

NS_FAILED	Vrací true, když byl stavový kód failure.
NS_SUCCEEDED	Vrací true, když byl stavový kód success.

Tabulka 4.5: Stavové kódy

4.6.4 Variabilní mapování 4.6

4.6.5 Společná error kódů 4.7

nsrefcnt	Výchozí typ pro referenční počet, 32-bitový integer.
nsresult	Výchozí typ pro chyby, 32-bitový integer.
nsnull	Výchozí hodnota null.

Tabulka 4.6: Variabilní mapování

NS_ERROR_NOT_INITIALIZED	Instance není inicializována.
NS_ERROR_ALREADY_INITIALIZED	Instance je už inicializována.
NS_ERROR_NOT_IMPLEMENTED	Metoda není implementována.
NS_ERROR_NO_INTERFACE	Rozhraní není podporováno.
NS_ERROR_NULL_POINTER	Platný ukazatel je nsnull.
NS_ERROR_FAILURE	Metoda selže, generická chyba.
NS_ERROR_UNEXPECTED	Neočekávaná chyba.
NS_ERROR_OUT_OF_MEMORY	Chyba při alokaci paměti.
NS_ERROR_FACTORY_NOT_REGISTERED	Požadováná třída není zaregistrována.

Tabulka 4.7: Společná error kódů

Kapitola 5

Realizace

V této kapitole se budu věnovat popisu realizace projektu a problémům při jeho realizaci.

5.1 Celkový přehled

Tento program by měl být multiplatformní, to znamená, že by měl fungovat pod operačními systémy Microsoft Windows, Linux a Mac OS. Protože však nevlastním MacBook a nesehnal jsem člověka, který by byl ochoten mi s tím pomoci, byl program testován jen pod Microsoft Windows a Debian GNU/Linux.

Grafické rozhraní je definováno v souboru `main.xul` a funkce jsou v souboru `main.js`. O přidávání položky "ICQ test" do popup menu se stará soubor `thunderbidOverlay.xul` a v souboru `overlay.js` je zase funkce, které zobrazí uživatelské rozhraní programu pomocí `window.open("chrome://icqtest/content/main.xul", "Blist","chrome,resizable=yes");`.

5.2 Vývojové prostředí

Při vývoji aplikace jsem používal textový editor GEdit, který je součástí Debian GNU/Linux, podporuje zvýrazňování syntaxe, dále jsem používal open-source nástroj Netbeans IDE, který v sobě obsahuje nástroje, jako je Subversion, Git, Mercurial. Pomocí těchto nástrojů můžeme zálohovat a obnovovat zdrojové soubory projektu.

5.3 Problémy při implementaci

Práce navazuje na předchozí projekt ExtBrain communicator, který funguje pouze pro Jabber. Po dohodě s vedoucím práce mi bylo doporučeno použít xpcompurple od Instantbirdu, který je už připraven k použití v aplikaci Mozilla.

Nejdříve jsem začal s tím, že jsem si stáhl zdrojové kódy Instantbirdu[4] verze 0.1.3.1 z jeho internetových stránek. Ty jsem zkompiloval podle návodu na Instantbird wiki[3]. Návod na kompilaci je popsán v příloze A.

Protože jsem nevěděl, které ze zkompilovaných souborů budu potřebovat, konzultoval jsem to raději na IRC kanálu, kde má tým vývojářů místnost s názvem `#instantbird`. Bylo mi sděleno, že budu potřebovat libpurple.so, libpurplexpcom.so a všechny soubory končící na .xpt z adresáře `obj-instantbird-dbg/purple/purplexpcom/public/_xpidlgen`.

Vytvořil jsem vlastní projekt s názvem "ICQ test" pomocí Firefox/Thunderbird Extension Wizard[1][14]. Stačí jednoduše zadat název, ID, verzi projektu, nastavit možnosti, cílové aplikace a popřípadě jednu ze tří nabízených licencí.

Poté jsem zkopíroval knihovny a soubory (libpurple.so, libpurplexpcom.so a všechny .xpt soubory) do složky "components". Nainstaloval je do Mozilla Thunderbirdu 2. Pomocí programu XPCOM viewer jsem chtěl zjistit, zda jsou všechny třídy a rozhraní načteny. Ovšem nebylo tomu tak. V XPCOM vieweru bylo zobrazeno jen rozhraní *purpleIAccount* atd., třídy jako *@instantbird.org/purple/core;1* nebyly nalezeny.

Následně jsem zkompiloval a vyzkoušel ostatní verze, které má Instantbird uloženy v Mercurialu. Žádná z nich nešla použít. Vždy to dopadlo tak, že v XPCOM vieweru nebyly vidět třídy od Instantbirdu.

Po další diskuzi s vývojáři jsme dospěli k závěru, že se jedná o chybu, konkrétně problém s *nsSocketTransportService*. Problém je v tom, že spojení pomocí soketů není v XPCOM možné použít, podrobnosti najdete na https://bugzilla.mozilla.org/show_bug.cgi?id=418535. Bylo mi řečeno, že je tento problém opraven v xulrunner od verze 1.9b4, zatímco Mozilla Thunderbird 2 běží na 1.8.*.

Z tohoto důvodu je mnou vyvíjený modul použitelný až pro Mozilla Thunderbird 3, který běží již na xulrunner 1.9.*. Problémem je to, že projekt ExtBrain communicator ještě nefunguje pod Thunderbirdem 3. Po konzultaci s vedoucím jsme se rozhodli, že zprovozním Instantbird na Mozille Thunderbidr 3 a předvedu jeho funkcionality.

Stáhl jsem si zdrojové kódy Comm-central z Mercurial, zkompiloval program a nainstaloval jsem balíček (návod na jeho instalaci je popsán v příloze A). Poté se v seznamu objevily XPCOM třídy, které jsem potřeboval.

Když byl tento problém vyřešen, začal zkoušet základní příkazy, které jsou používány pro testování. Např.: připojování k účtu, vypsání protokolů atd.

```
var pcs = Components.classes["@instantbird.org/purple/core;1"]
                    .getService(Ci.purpleICoreService);
pcs.init();
var proto = pcs.getProtocolById(aProto);
if (!proto)
  throw "Couldn't get protocol " + aProto;
var acc = pcs.createAccount(aName, proto);
...
```

[1] Firefox/Thunderbird Extension Wizard - nástroj pro jednoduché a rychlé vytváření .xpi balíčků.

Když jsem postupně napsal výše uvedený kód do JavaScript shellu, tak byl objekt pcs vytvořen bez problému. Když jsem však chtěl pcs zinicializovat celý Thunderbird zamrzl. Nic nešlo udělat, jedině ho vypnout pomocí příkazu *kill* přes konzoli.

Snažil jsem se tedy dopátrat proč se tak děje. Postupně jsem prošel konfigurační soubory, zdrojové kódy, ověřil jsem funkce *init()*. Ze zprávy v konzoli jsem zjistil, že v adresáři Thunderbirdu chybí soubor libpurple.so, tak jsem pro jistotu do tohoto adresáře nahrál jak libpurple.so, tak i libpurplexpcom.so, ale problém se mi tím vyřešit nepodařilo. Nechal jsem si tedy poradit. Řešení bylo poměrně jednoduché, stačilo si stáhnout soubor a udělat *patch*[2], aby knihovny od Instantbirdu mohly fungovat pro xulrunner 1.9.1, jsou totiž zkompilované pomocí xulrunner 1.9.2 a je potřeba udělat patch.

Ukázka kódu z patch souboru:

```
diff --git a/instantbird/content/addbuddy.js b/instantbird/content/addbuddy.js
--- a/instantbird/content/addbuddy.js
+++ b/instantbird/content/addbuddy.js
@@ -58,17 +58,17 @@ var addBuddy = {
...
-    let tags = this.pcs.getTags();
+    let tags = this.pcs.getTags({});
...
```

Po této úpravě je inicializace hotová a Thunderbird nadále běží. Což byla dobrá zpráva do té doby, než se objevil další problém. Tentokrát to byla špatná inicializace. Snažil jsem se najít řešení pomocí programu `strace`[3] ale neúspěšně. Na internetovém fóru jsem se dozvěděl, že mám zkopírovat dva soubory all-instantbird.js a purple-prefs.js do svého projektu a zakomentovat první dva řádky v souboru all-instantbird.js.

Při vytváření projektu jsem narazil na další problém. Funkce *getIter()*, která má vracet seznamy protokolů, účtů atd., při zavolání skončila chybovým hlášením, že chybí ;(konec příkazu). Bylo to způsobeno tím, že klíčové slovo `yield` je nově přidané do JavaScriptu 1.7 a v dřívějších verzích nebylo. Aktualizoval jsem si tedy JavaScript na verzi 1.7.

```
function getIter(aEnumerator)
{
  while (aEnumerator.hasMoreElements())
    yield aEnumerator.getNext();
}
```

Při programování jsem narazil ještě na další problém, který způsobil tento řádek

```
Components.utils.import("resource://app/modules/imWindows.jsm");
```

[2] patch - nástroj, kterým může být soupis změn aplikován na staré soubory tak, že získáme nové.

[3] strace - je nástroj, který sleduje a zaznamenává systémové volání. Je užitečný pro stopování a určování přesné příčiny systémových chyb, nebo podezřelého chování aplikací.

Nešlo soubor `imWindows.jsm` naimportovat, je to způsobeno tím, že alias `app` se odkazuje na soubory Firefoxu, neodkazuje se však tam, kde leží rozšíření. Je proto potřeba nadefinovat vlastní alias a uvést to v souboru `chrome.manifest`. To může vypadat například následovně:

```
resource privatiser modules/ #definice vlastního alias
#ted' se tam odkazujeme pomocí definovaného alias
Components.utils.import("resource://privatiser/imWindows.jsm");
```

Soubor `imWindows.jsm` nebyl nijak důležitý, neovlivňuje chod programu, ale chyba v consoli nevypadá dobře. Program byl nakonec dokončen, ale při jeho implementaci do nových verzí ExtBrain communicatoru je zapotřebí vylepšit například grafické uživatelské rozhraní, způsob zobrazení atd.

Kapitola 6

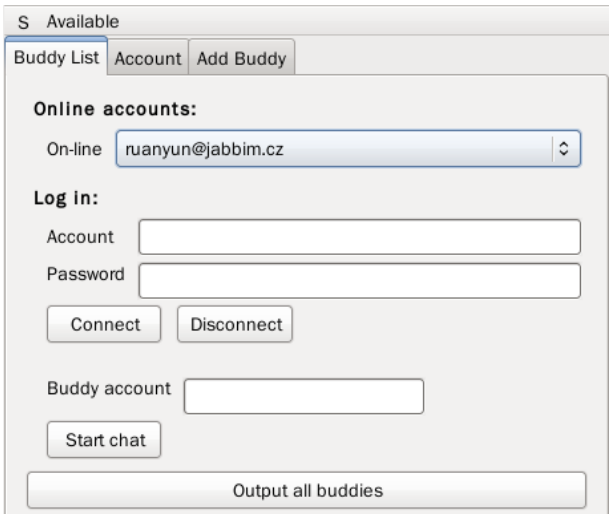
Testování

V této kapitole se budu věnovat popisu testování funkčnosti tohoto projektu.

6.1 Realné nasazení

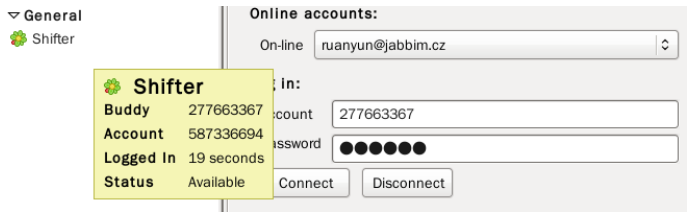
Funkčnost tohoto programu jsem testoval na Debian GNU/Linuxu. V programu je možné se přihásit na několik účtů najednou. Vyzkoušel jsem pouze práci s Jabber a ICQ, protože mám jen tyto dva účty.

Pro spouštění programu stačí provést následující kroky: Tools → ICQ test. Poté se otevře nové okno s přehledem aktuálně přihlášených účtů. Po přihlášení na jiný účet je zapotřebí tento seznam ručně aktualizovat kliknutím na combobox. Status lze změnit kliknutím na tlačítko "S" a vybrat jednu ze tří možností nebo napsat vlastní zprávu.



Obrázek 6.1: Program po spouštění

Připojení a odpojení účtu probíhá bez problémů. Je však nutné číslo účtu a heslo zadávat při každém připojení. U protokolů jako je Jabber, MSN atd., je potřeba zadat i název serveru. Po spuštění se program automaticky připojí na server.



Obrázek 6.2: Přihlášování

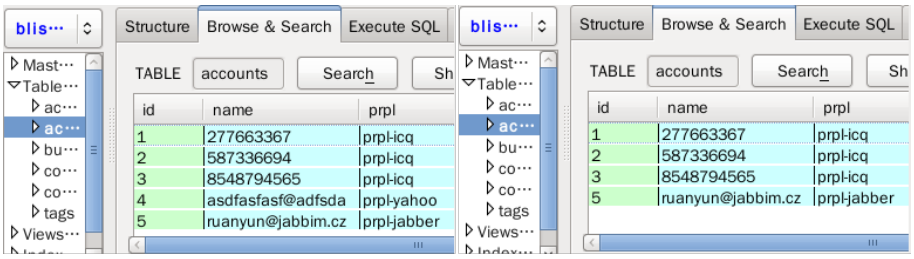
Seznam přátel a skupiny je možné zobrazit po kliknutí na tlačítko "Output all buddies". Při dvojkliku na vybranou položku v seznamu se v textboxu "Buddy account" zobrazí alias nebo ICQ id daného účtu.

Funkce pro vytváření a mazání účtu jsem vyzkoušel tak, že jsem vytvořil několik účtů a postupně jsem všechny smazal, výsledky jsem si ověřil pomocí SQLite manager.

```
//tento řádek vytvoří účet a uloží do databáze.
var cre_acc = pcs.createAccount(username, userproto.value).password=userpass;

//tento zase smaže účet podle ID v databázi.
pcs.deleteAccount(account.id);
```

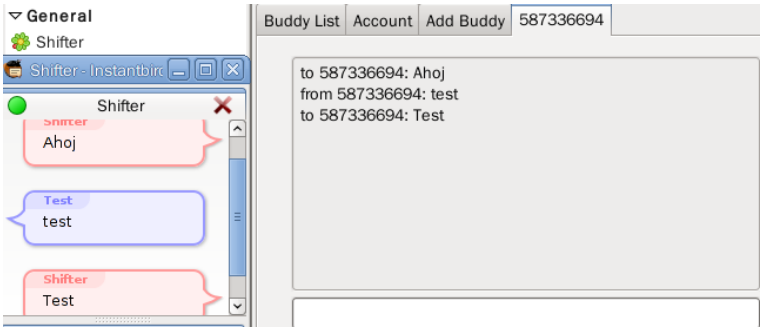
Výsledek podle SQLite manager ukazuje obrázek 6.3:



Obrázek 6.3: Porovnání po provedení příkazu

Pro přidání nového kontaktu do seznamu je potřeba nejdřív vybrat jeden z aktuálně přihlášených účtů, zadat jméno, email nebo ICQ id a vybrat si skupinu, poté potvrdit tlačítkem "Create".

Pro vytváření nové konverzace je potřeba si nejdřív vybrat z online seznamu jeden účet, poté ze seznamu přátel cílový účet. Dvojkliknutím na jméno cílového účtu se objeví v "Buddy account", následně po stisknutí tlačítka "Start chat" můžete už komunikovat. Tato funkcionalita však není zcela dokončena. Můžete komunikovat jenom s jedním člověkem a pokud



Obrázek 6.4: Chatování

chcete mluvit s jiným, musíte nejdřív zavřít program a znovu ho spustit. Což je to potřeba vylepšit v dalších verzích.

Domnívám se, že toto dostatečně demonstruje funkčnost **xpcompurple** pod Mozilla Thunderbird 3.

6.2 Srovnávání s existujícími řešeními

Protože v Mozilla Thunderbirdu zatím neexistuje podobný program, tak není možné porovnání.

Kapitola 7

Závěr

7.1 Zhodnocení

Díky této práci jsem se naučil jak se programují rozšíření do Mozilla aplikací, jak funguje technologie XPCOM, XPConnect a XUL. Poznal jsem výhody a nevýhody programů tohoto typu. Myslím si, že se zkušenostmi, které jsem nabyl při realizaci tohoto projektu, bych mohl použít i jinde, nejen při programování aplikací tohoto typu. Při vypracování tohoto projektu jsem také narazil na mnoho problémů, které jsem ovšem všechny zdárně vyřešil.

Možností dalšího pokračování této práce je spousta, protože **xpcompurple** není sám o sobě dokončený. Tento projekt se stále vylepšuje a zdokonaluje. Hlavním cílem je mít hotový projekt ExtBrain communicator, který bude fungovat pod Mozilla Thunderbirdem 3 a implementovat do něj `xpcompurple`. Poté se zaměřit na jiné detaily, jako je grafické rozhraní atd.

Závěrem bych se chtěl omluvit za mou češtinu a gramatiku. Doufám, že jsem neudělal mnoho chyb.

Literatura

[1] DOM Inspector [online].
https://addons.mozilla.org/cs/thunderbird/addon/1806, stav ze 16. 4. 2010.

[2] Extension Developer [online].
https://addons.mozilla.org/cs/thunderbird/addon/7434, stav ze 16. 4. 2010.

[3] Instantbird kompilace [online].
https://wiki.instantbird.org/Instantbird:Compiling, stav ze 25. 5. 2010.

[4] Instantbird download [online].
http://instantbird.com/download.html, stav ze 25. 5. 2010.

[5] Instantbird [online].
http://instantbird.com/, stav ze 16. 4. 2010.

[6] JavaScript and the DOM [online].
http://www.csie.ntu.edu.tw/~piaip/docs/CreateMozApp/mozilla-chp-5-sect-2.html, stav ze 16. 5. 2010.

[7] Mozilla application framework [online].
https://developer.mozilla.org/en/Mozilla_Application_Framework_in_Detail, stav ze 16. 4. 2010.

[8] Rapid Application Development with Mozilla [online].
http://mb.eschew.org/1, stav ze 16. 5. 2010.

[9] Scripting Mozilla [online].
http://www.csie.ntu.edu.tw/~piaip/docs/CreateMozApp/mozilla-chp-5.html, stav ze 16. 5. 2010.

[10] SQLite Manager [online].
https://addons.mozilla.org/cs/thunderbird/addon/5817, stav ze 16. 4. 2010.

[11] Add-on pro Thunderbird [online].
https://addons.mozilla.org/en-US/thunderbird/, stav ze 16. 4. 2010.

[12] Thunderbird [online].
http://www.mozillamessaging.com/thunderbird/, stav ze 16. 4. 2010.

[13] Venkman [online].
https://addons.mozilla.org/cs/thunderbird/addon/216, stav ze 16. 4. 2010.

[14] Firefox/Thunderbird Extension Wizard [online].
http://ted.mielczarek.org/code/mozilla/extensionwiz/, stav ze 25. 5. 2010.

[15] XPCOM [online].
https://developer.mozilla.org/en/Creating_XPCOM_Components/An_Overview_of_XPCOM, stav ze 16. 5. 2010.

[16] XPCOM [online].
http://www.csie.ntu.edu.tw/~piaip/docs/CreateMozApp/mozilla-chp-8.html, stav ze 16. 5. 2010.

[17] XPCOM Viewer [online].
https://addons.mozilla.org/cs/thunderbird/addon/7979, stav ze 16. 4. 2010.

[18] XPConnect and Scriptable Components [online].
http://www.csie.ntu.edu.tw/~piaip/docs/CreateMozApp/mozilla-chp-5-sect-4.html, stav ze 16. 5. 2010.

[19] XUL Explorer [online].
https://developer.mozilla.org/en/XUL_Explorer, stav ze 16. 4. 2010.

Dodatek A

Instalační a uživatelská příručka

A.1 Instalace Mozilla Thunderbird a Instantbird

Mozilla Thunderbird je k dispozici na webových stránkach Mozilla messaging (http://www.mozillamessaging.com/), aktuální verze je: 3.0.4.

Pokud chcete Mozilla Thunderbird zkompilovat na svém počítači, můžete si zdrojové kódy stáhnout přes CVS, Mercurial nebo FTP/HTTP. Postup je následující (Detailní postup je uveden na webu[1] a je také popsán v A.1.1): Zaprvé musíte mít nainstalované požadované nástroje buď pro Microsoft Windows[2] nebo pro Linux[3], podle toho jaký operační systém používáte. Zadruhé je potřeba stáhnout zdrojové kódy buď z Mercurial nebo přes FTP/HTTP. Poté nakonfigurovat[4] (např. debug mód) a nakonec to zkompilovat pomocí příkazu *make*[5].

A.1.1 Postup instalace

V Linuxu je nutné nainstalovat některé knihovny a nástroje. Zde uvedu potřebné propriety pro Debian GNU/Linux:

```
$ apt-get build-dep thunderbird
$ apt-get install mercurial libasound2-dev libcurl4-openssl-dev libnotify-dev \
libxt-dev libiw-dev mesa-common-dev autoconf2.13
```

Pro Microsoft Windows je potřeba stáhnout: `Microsoft Visual C++ Tools` - obvykle je to `Visual Studio`, `Microsoft Windows SDK(s)`[6], `MozillaBuild`[7]. Podle toho jakou verzi

[1] Postup instalace je na https://developer.mozilla.org/en/Simple_Thunderbird_build

[2] Prerekvizity pro Microsoft Windows: https://developer.mozilla.org/En/Developer_Guide/Build_Instructions/Windows_Prerequisites

[3] Prerekvizity pro Linux: https://developer.mozilla.org/En/Developer_Guide/Build_Instructions/Linux_Prerequisites

[4] Konfigurace: https://developer.mozilla.org/en/Configuring_Build_Options

[5] Kompilace: https://developer.mozilla.org/en/Build_and_Install

[6] Windows SDK je ke stažení na https://developer.mozilla.org/En/Windows_SDK_versions

[7] MozillaBuild je ke stažení na http://ftp.mozilla.org/pub/mozilla.org/mozilla/libraries/win32/MozillaBuildSetup-Latest.exe

Windows používáte.

Jakmile máte tyto nástroje a knihovny nainstalované, můžete stáhnout zdrojové kódy z Mercurialu. V Linuxu stačí příkaz:

```
$ hg clone http://hg.mozilla.org/comm-central/  #pro Mozilla thunderbird
$ hg clone http://hg.instantbird.org/instantbird/ #pro Instantbird
```

Ve Windows je zapotřebí nejdříve spustit dávkový soubor z `MozillaBuild` ve Windows commanderu podle toho, jakou verzi Visual studia máte nainstalovanou, a teprve potom zadat příkazy uvedené výše.

```
C:/MozillaBuild/start-msvc9.bat # pro Visual studio 2008
```

Po stažení kódů je potřeba otevřít adresář, kde jsou kódy uloženy a provést aktualizace:

```
$ cd comm-central
$ python client.py checkout

$ cd instantbird
$ python client.py checkout
```

poté vytvořit soubor `mozconfig` pro Thunderbird i pro Instantbird, ve kterém budou například tyto hodnoty:

```
# Setup a basic mozconfig file
#let's build Thunderbird
echo 'ac_add_options --enable-application=mail' > mozconfig
#in this directory
echo 'mk_add_options MOZ_OBJDIR=@TOPSRCDIR@/objdir-tb-debug' >> mozconfig
#quickly
echo 'mk_add_options MOZ_MAKE_FLAGS="-j4"' >> mozconfig
#debug mode
echo 'ac_add_options --enable-debug' >> mozconfig
#no optimize
echo 'ac_add_options --disable-optimize' >> mozconfig
```

Pro Instantbird je to podobné, ale místo `--enable-application=mail a MOZ_OBJDIR= @TOPSRCDIR@/objdir-tb-debug` bude `--enable-application=instantbird a MOZ_OBJDIR= @TOPSRCDIR@/objdir-instantbird-debug`. To zmámená, že název této aplikace bude instantbird, zatímco Thunderbird používá mail. `MOZ_OBJDIR` je místo, kde má být aplikace nainstalována.

Samotná kompilace se na konec provede pomocí příkazu

```
#v adresaří instantbird a comm-central
$ make -f client.mk build
```

Program po kompilaci najdete v adresáři `MOZ_OBJDIR/mozilla/dir/dist/bin/`.

Pro pozdější aktualizace kódu stačí provést:

```
$ cd instantbird
$ python client.py checkout
$ make -f client.mk build

$ cd comm-central
$ python client.py checkout
$ make -f client.mk build
```

Před kompilací Instantbirdu je potřeba nainstalovat záplatu. Je velice důležitá, protože jinak zkompilované knihovny pod Thunderbirdem 3 nebudou fungovat. Záplata s názvem `instantbird.patch` je uložena spolu se zdrojovým kódem v adresáři `instantbird`.

Záplatu nainstalujete následně:

```
$ cd instantbird
$ patch -p1 instantbird.patch
```

A.2 Instalace projektu

Postup instalace programu je v podstatě stejný jako instalace jiného rozšíření aplikací do Mozilla. V Thunderbirdu si vyberete menu Tools → Add–ons → Extensions → Install, pak si najdete soubor icqtest.xpi a kliknutím na tlačítko OK započne instalace.

Nakonec je nutné ještě zkopírovat knihovnu libpurple.so (purple.dll pod Windows) do adresáře Thunderbirdu.

Po instalaci se objeví nová položka pod menu Tools s názvem “ICQ test”, kliknutím na položku program spustíte.

Dodatek B

Obsah přiloženého CD

- Mozilla Thunderbird zkompilovaný ze zdrojových kódů. Verze pro Microsoft Windows 7 a pro Debian GNU/linux 5.0 lenny.
- Instantbird zkompilovaný ze zdrojových kódů. Verze pro Microsoft Windows 7 a pro Debian GNU/linux 5.0 lenny.
- Profil Mozilla Thunderbird, který obsahuje veškeré nástroje a nastavení.
- Projekt "ICQ test".
  - Dokumentace
  - Zdrojové kódy
  - Navod - Návod ke spuštění programu po instalaci.
  - build.sh - Bash soubor pro vytváření .xpi balíčků, po spuštění vytvoří icqtest.xpi, který můžete nainstalovat do Mozilla Thunderbirdu, předtím je ještě potřeba provést copy_lib.sh.
  - config_build.sh - Konfigurační soubor, jsou zde potřebné odkazy pro build.sh.
  - soft_install.sh - Příkazy pro stažení Instantbird z repositáře a následné instalace.
  - soft_install1.sh - Příkazy pro stažení Mozilla thunderbird 3 z repositáře a následnou instalaci.
  - copy_lib.sh - Zkopíruje knihovnu libpurple.so do Mozilla Thunderbirdu.
  - aktualizace.sh - provede aktualizaci a reinstalaci Instantbirdu nebo Mozilla Thunderbirdu.