Mozilla

platforma pro vývoj aplikací

Verze 1.0.1 – 21.8.2003

David Majda

2. ročník MFF UK

david.majda@seznam.cz

http://dmajda.jinak.cz/

Obsah

Úvod

Většina lidí, zabývajících se alespoň trochu počítači, internetem a tvorbou webových stránek či aplikací pravděpodobně zná prohlížeč Mozilla[1]. Možná ho mají i nainstalován na počítači a dokonce ho rutinně používají. Málokdo už ale ví, co všechno se za tímto projektem skrývá a na základě jakých technologií je vystavěn.

Tato práce si klade za cíl čtenáře seznámit s projektem Mozilla, osvětlit technologie použité při jeho vývoji, ukázat jejich použití a v neposlední řadě také naznačit, jakým směrem se bude projekt ubírat v nejbližších měsících dál.

U čtenáře předpokládám zběžnou orientaci v problematice internetu a alespoň částečnou znalost HTML, XML, CSS, JavaScriptu a C++.

[1] http://www.mozilla.org

Historie projektu Mozilla

Pokud chceme poznat historii Mozilly, musíme do doby před deseti lety, kdy v americkém Národním centru pro supervýpočetní aplikace (NCSA, National Center for Supercomputing Applications[2]) vznikl první webový prohlížeč Mosaic[3]. Jeho tvůrci Marc Andreesen a Jim Clark poté založili firmu Mosaic Communications, později přejmenovanou na Netscape Communications[4]. Tam dále pokračovali ve tvorbě[5] svého webového prohlížeče, tentokrát již pod názvem Netscape Navigator. První betaverze spatřila světlo světa[6] 13. října 1994, brzy poté vyšla i verze 1.0 a další verze následovaly v rychlém sledu.

V polovině 90. let se do hry vložil Microsoft, který pochopil, že ve webovém prohlížeči leží klíč k ovládnutí celého internetu. Proto zakoupil licenci na původní Mosaic a na jeho základě začal vyvíjet vlastní prohlížeč Internet Explorer[7]. První verze srovnatelná s Navigatorem byla 3.0, a verze 4.0 ho pak ve své době výrazně technologicky předstihla (zejména v podpoře technologie DHTML a CSS).

Microsoft distribuoval svůj prohlížeč zdarma ve všech verzích svého operačního systému Windows (mohl si dovolit jeho vývoj dotovat z jiných projektů), a tak si zajistil jeho postupné rozšíření, protože většina uživatelů necítila potřebu prohlížeč měnit. Navíc byl ve své době Internet Explorer i technologicky nejlepší (není divu – v jednu chvíli bylo v jeho vývojovém týmu přes 500 lidí!). Netscape byl pro komerční uživatele placený a postupně začínal ztrácet pozici dominantního prohlížeče. Jeho poslední verze 4.0 byla technologickým paskvilem (trpěl zastaralým renderovacím jádrem, nevyhovující byla i síťová knihovna a další komponenty).

Přišel čas na radikální krok. Tím bylo zveřejnění[8] zdrojového kódu[9] rozpracované verze 5.0. Idea byla taková, že programátoři z celého světa začnou do programu dopisovat nové funkce, a vývojáři Netscapu pak jednotlivé kusy poskládají dohromady. Správou nad otevřenou verzí zdrojového kódu byla pověřena organizace mozilla.org, který byla nezávislá na samotné firmě Netscape.

Bohužel, radikální postup nefungoval podle původních představ. Kód byl složitý a nepřehledný a do vývoje se zapojilo jen několik málo externích programátorů. Nakonec padlo rozhodnutí takřka celý kód úplně přepsat (až na nejnižší vrstvy a JavaScriptovou knihovnu). To vývoj dále pozdrželo, z Netscapu (tou dobou už pohlceného[10] gigantem AOL[11]) začali odcházet klíčoví vývojáři[12]. Z dnešního hlediska

---

[2] http://www.ncsa.uiuc.edu/

[3] http://archive.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html

[4] http://www.netscape.com/

[5] http://www.jwz.org/gruntle/nscpdorm.html

[6] http://www.jwz.org/gruntle/newsrelease1.html

[7] http://www.microsoft.com/windows/ie/default.asp

[8] http://wp.netscape.com/newsref/pr/newsrelease577.html

[9] http://lxr.mozilla.org/seamonkey/

[10] http://news.com.com/2100-1023-218360.html

[11] http://www.aol.com/

se rozhodnutí o přepsání kódu jeví jako minimálně sporné, ale v době učinění tohoto rozhodnutí se asi situace jevila jinak. Přepsání umožnilo některé věci navrhnout čistěji a důsledněji a navíc se pustit i do velkých experimentů s novými technologiemi (některé si v této práci představíme). Díky tomu se Mozilla postupně stala nejen „dalším internetovým prohlížečem“, ale též platformou použitelnou pro vývoj zejména internetových aplikací.

Prohlížeč Mozilla začal postupně nabývat tvar a v dubnu 2002 – po více jak čtyřech letech vývoje – dospěl k verzi 1.0[13]. Z kódu Mozilly byly odvozeny i Netscape Communicator 6.x a 7.x. V době psaní článku probíhají práce na verzi Mozilla 1.5.

Většina zdrojového kódu Mozilly je licencován zároveň jako GPL[14] (GNU General Public License), LGPL[15] (GNU Lesser General Public License) a MPL[16] (Mozilla Public License) a uživatel zdrojového kódu si může vybrat, pod jakou licencí s ním hodlá pracovat. Poslední jmenovaná licence umožňuje začlenit části zdrojového kódu i do komerčních aplikací, čímž se Mozilla stala atraktivní i pro firmy jako Sun[17], IBM[18] nebo třeba RedHat[19] (který například pro své zájmy potřeboval funkční prohlížeč na platformě Linux). Vývojáři z těchto i dalších firem aktivně přispívají k vývoji Mozilly – většinou v těch oblastech, které sami považují za důležité. V tom se projevuje výhoda open source – práce, která je společném zájmu všech, může být díky otevřenosti kódu udělána jen jednou a nemusí se s ní zabývat každá firma zvlášť.

---

[12] http://www.jwz.org/gruntle/nomo.html

[13] http://www.mozilla.org/press/mozilla1.0.html

[14] http://www.gnu.org/licenses/gpl.html

[15] http://www.gnu.org/licenses/lgpl.html

[16] http://www.mozilla.org/MPL/MPL-1.1.html

[17] http://www.sun.com/

[18] http://www.ibm.com/

[19] http://www.redhat.com/

Představení programu

Mozilla se skládá ze tří základních komponent: **Navigator**, **Mail & Newsgroups** a **Composer**. Kromě nich obsahuje některé další menší součásti: **Address Book**, **ChatZilla**, **JavaScript Debugger** a **DOM Inspector**. Jednotlivé komponenty si nyní představíme podrobněji.

Navigator

Navigator je internetový prohlížeč. Z hlediska ovládání vychází z Netscape Navigatoru 4.x a nabízí víceméně obdobné možnosti, přidává ovšem navíc „tabbed browsing“ (otvírání víc stránek v jednom okně najednou), blokování automaticky otvíraných oken a mnoho dalších funkcí. Jejich podrobný popis by zabral minimálně stejný rozsah, jako má celá tato práce.

Mail & Newsgroups

Mozilla obsahuje poměrně chytrý mailový/news klient, jehož základní vlastností je podpora několika nezávislých POP3, IMAP nebo news účtů. Je víceméně ekvivalentní náhražkou programu Microsoft Outlook Express. Umožňuje ukládat maily do složek, automaticky je filtrovat podle různých kritérií a v novějších verzích Mozilly implementuje i automatické filtrování spamu (pomocí adaptivního algoritmu založeném na sledování výskytů různých slov v mailech).

Composer

Composer je nenáročný editor určený k tvorbě jednoduchých HTML dokumentů. Umožňuje základní formátování, vkládání tabulek, obrázků a odkazů. K vykreslování editovaného dokumentu využívá stejné renderovací jádro jako samotná Mozilla. Jako komponenta je editor vestavěn i v Mozilla Mailu, kde se používá pro editaci mailů ve formátu HTML.

Address Book

Jednoduchý adresář kontaktů. Je dobře integrován s Mozilla Mailem.

ChatZilla

ChatZilla je jednoduchý IRC klient. Upřímně řečeno je to nejzbytečnější program z celého balíku, protože na všech platformách existují lepší.

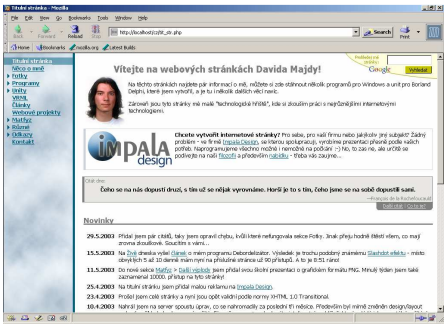
JavaScript Debugger

Velice užitečná pomůcka pro ladění skriptů na webových stránkách, nebo skriptů používaných v Mozille samotné. Umožňuje krokování, nastavování watches a další funkce nezbytné pro komfortní ladění, jak je známe z moderních vývojových prostředí (Microsoft Visual C++, Borland Delphi, apod.).

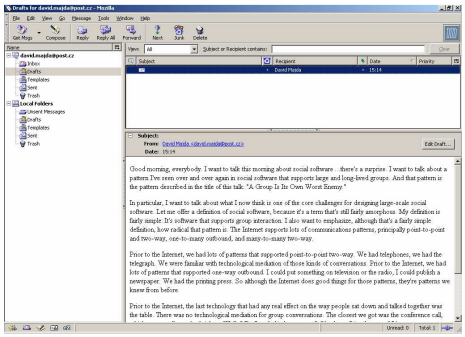
DOM Inspector

Ještě užitečnější nástroj než JavaScript debugger je DOM Inspector. Dokáže zobrazit strom elementů načtené stránky (či XML dokumentu), ukázat pro každou

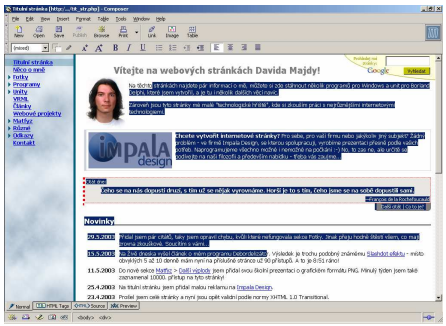
jeho větev např. použitá pravidla CSS nebo jak je reprezentována v DOM jako JavaScriptový objekt. Při ladění složitějších stránek je to velice praktická pomůcka.



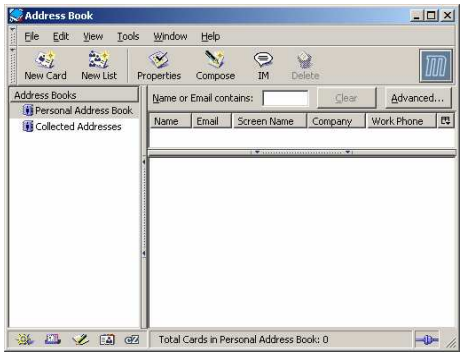
Navigator



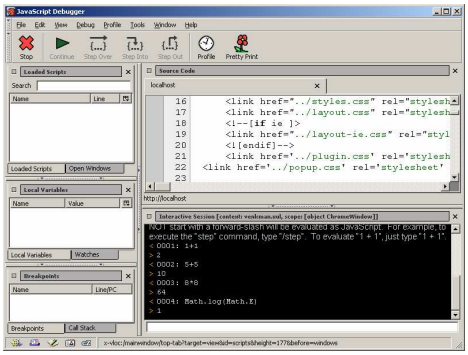
Mail & Newsgroups



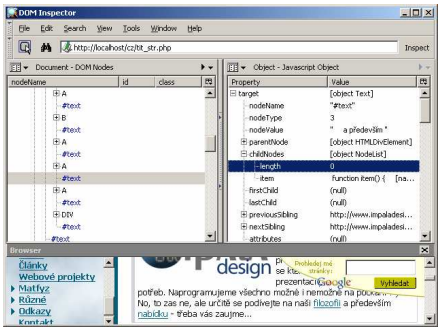
Composer



Address Book



JavaScript Debugger



DOM Inspector

Architektura

Mozilla je velmi rozsáhlý program (zdrojový kód zabírá přes 200 MB) a její architektura není úplně jednoduchá. Dá se říct, že Mozilla se z programátorského hlediska skládá z několika základních komponent či technologií:

- NSPR (Netscape Portable Runtime)
- JavaScript Engine
- XPCOM (Cross Platform Component Object Model)
- XPConnect
- Gecko (renderovací jádro)
- XUL (XML User Interface Language)

Při vývoji Mozilly je dáván velký důraz na její multiplatformnost – tomu je podřízeno takřka vše, včetně pravidel pro psaní kódu[20]. Výsledkem je, že Mozilla se dá zkompilovat a běží na následujících operačních systémech/plaformách:

- Windows
- Linux (i386, PPC)
- MacOS
- MacOS X
- Solaris (2.7/SPARC, 2.6/SPARC, 2.6/x86)
- FreeBSD
- Irix
- BeOS
- HPUX
- OS/2
- ...a další

Jako hlavní vývojový jazyk je použito C++, ale vyšší úrovně kódu jsou z větší části psané v XML a JavaScriptu. O tom všem ale až později.

Nyní si představíme jednotlivé vrstvy kódu trochu podrobněji.

NSPR[21] (Netscape Portable Runtime)

Knihovna NSPR je nejnižší vrstva kódu v Mozille. Je napsána v jazyku C a je to jakási nadstavba standardní knihovny tohoto jazyka. Hlavním důvodem její existence je abstrakce nad jednotlivé funkce konkrétních OS. Knihovna se dá zkompilovat úplně nezávisle na zbytku zdrojového kódu a použít i v jiných programech.

[20] http://www.mozilla.org/hacking/mozilla-style-guide.html

[21] http://www.mozilla.org/projects/nspr/

Momentálně knihovna funguje na platformách Win32, Macintosh a na víc než 20 různých UNIXech.

NSPR zajišťuje především tyto funkce:

- definice portabilních datových typů a maker pro zacházení s nimi
- vlákna
- základní synchronizace
- I/O
- základní síťové funkce
- funkce pro práci s datem a časem
- správa paměti
- načítání dynamických knihoven

Na ukázku způsobu, jakým se NSPR vyrovnává s rozdíly mezi platformami a překladači jazyka C, uvádím kompletní definici datových typů `PRInt64` a `PRUInt64`:

```
/************************************************************************
** TYPES:       PRUint64
**              PRInt64
** DESCRIPTION:
**  The int64 types are known to be 64 bits each. Care must be used when
**  declaring variables of type PRUint64 or PRInt64. Different hardware
**  architectures and even different compilers have varying support for
**  64 bit values. The only guaranteed portability requires the use of
**  the LL_ macros (see prlong.h).
************************************************************************/
#ifdef HAVE_LONG_LONG
#if PR_BYTES_PER_LONG == 8
typedef long PRInt64;
typedef unsigned long PRUint64;
#elif defined(WIN16)
typedef __int64 PRInt64;
typedef unsigned __int64 PRUint64;
#elif defined(WIN32) && !defined(__GNUC__)
typedef __int64  PRInt64;
typedef unsigned __int64 PRUint64;
#else
typedef long long PRInt64;
typedef unsigned long long PRUint64;
#endif /* PR_BYTES_PER_LONG == 8 */
#else  /* !HAVE_LONG_LONG */
typedef struct {
#ifdef IS_LITTLE_ENDIAN
    PRUint32 lo, hi;
#else
    PRUint32 hi, lo;
#endif
} PRInt64;
typedef PRInt64 PRUint64;
#endif /* !HAVE_LONG_LONG */
```

Makra `HAVE_LONG_LONG`, `PR_BYTES_PER_LONG` a další ve výše uvedeném úseku kódu jsou definovány v závislosti na cílové platformě. Zbytek kódu NSPR vypadá hodně podobně a je v něm značné množství platformně závislých pasáží.

Zajímavostí je, že knihovna původně vznikla době, kdy do Netscape Navigatoru byla implementována podpora Javy. Protože ta je sama o sobě multiplatformní, bylo logické udělat multiplatformní i její podpůrnou vrstvu – tou byla právě knihovna NSPR. Časem byla knihovna zobecněna, upravena a tvoří dnes „podklad“ takřka veškerého zdrojového kódu Mozilly.

JavaScript Engine[22]

JavaScriptová komponenta Mozilly trochu vybočuje z její architektury, ale stojí za to se o ní zmínit. Je to kompletní interpreter JavaScriptu, který je ve verzi 1.5 nadmnožinou standardu ECMA-262 Edition 3. Celý interpreter je víceméně dílem jednoho člověka – Brendana Eiche (tvůrce jazyka JavaScript) a byl napsán také ještě v dobách slávy Netscape Navigatoru. Díky své kvalitě přežil ve zdrojovém kódu bez velkých změn až doteď. Skripty jsou enginem před zpracováním předkompilovány do binární reprezentace, a tak jsou později spouštěny poměrně rychle.

Celá komponenta je napsána v jazyce C a je obdobně jako NSPR nezávislá na zbytku kódu Mozilly. Dá se také zkompilovat samostatně jako dynamická knihovna s poměrně přímočarým API. Toho se dá využit například pokud chcete do svého programu přidat podporu skriptování. (Tímto způsobem vyřešila podporu skriptování například firmy Adobe v produktu Adobe Acrobat.)

Zde je jednoduchá ukázka, jak vypadá začátek kódu, který knihovnu využívá:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "jsapi.h"

int main(int argc, char **argv) {
  int c, i;

  JSVersion version;
  JSRuntime *rt;
  JSContext *cx;
  JSObject  *glob, *it;
  JSBool builtins;

  /* inicialiujeme run-time */
  if (!rt = JS_NewRuntime(8L * 1024L * 1024L)
    return 1;

  /* vytvoříme nový kontext a asociujeme ho s JS run-time */
  if ((cx = JS_NewContext(rt, 8192)) == NULL)
    return 1;

  /* vytvoříme nový globální objekt */
  glob = JS_NewObject(cx, clasp, NULL, NULL);

  /* inicializujeme vestavěné objekty JS a globální objekty */
  builtins = JS_InitStandardClasses(cx, glob);

  .
  .
  .
```

[22] http://www.mozilla.org/js/index.html

```
  return 0;
}
```

XPCOM[23] (Cross Platform Component Object Model)

Ve velké části zdrojového kódu využívá Mozilla systém XPCOM, což je komponentový objektový model vystavěný podle modelu COM firmy Microsoft. XPCOM umožňuje obdobně jako COM definovat rozhraní komponenty ve speciálním popisovacím jazyku XPIDL[24] (dialekt jazyka IDL) a rozhraní pak implementovat. To zajišťuje poměrně vysokou úroveň modulárnosti kódu.

Nejtypičtějším způsobem implementace je třída v jazyku C++. Přitom se využívá XPCOM kompilátoru, který z XPIDL souboru vygeneruje hlavičkový soubor a také šablonu `cpp` souboru, do které se jen dopíše samotná implementace.

Jinou možností je rozhraní implementovat v jazyku JavaScript, což je možné díky technologii XPConnect (viz níže).

Objekty v XPCOM podporují počítání referencí se všemi jeho výhodami a nevýhodami. Pro pohodlnou práci je definována řada maker, která skrývají komplexitu kódu. I tak ale platí, že programování v XPCOM není zrovna věc vhodná pro začátečníky v C++.

Všechna rozhraní jsou odvozena z jediného předka `nsISupports` (analogie `IUnknown` v COM), který je v XPIDL definován takto:

```
[scriptable, uuid(00000000-0000-0000-c000-000000000046)]
interface nsISupports {
  void QueryInterface(in nsIIDRef uuid,
                      [iid_is(uuid),retval] out nsQIResult result);
  [noscript, notxpcom] nsrefcnt AddRef();
  [noscript, notxpcom] nsrefcnt Release();
};
```

Nejjednodušší implementace rozhraní jako třídy v C++ pak vypadá následovně:

```
/**
 * Basic component object model interface. Objects which implement
 * this interface support runtime interface discovery (QueryInterface)
 * and a reference counted memory model (AddRef/Release). This is
 * modelled after the win32 IUnknown API.
 */
class NS_NO_VTABLE nsISupports {
public:
  NS_IMETHOD QueryInterface(REFNSIID aIID, void** aInstancePtr) = 0;
  NS_IMETHOD_(nsrefcnt) AddRef(void) = 0;

  NS_IMETHOD_(nsrefcnt) Release(void) = 0;
};
```

[23] http://www.mozilla.org/projects/xpcom/

[24] http://www.mozilla.org/scriptable/xpidl/

Oproti Microsoft COM je má technologie XPCOM podstatně užší pole záběru – neexistuje podpora pro komunikaci komponent nacházejících se v různých procesech (tj. v různých adresných prostorech) a už vůbec ne na různých počítačích (tj. není podporováno RPC). Vzhledem k účelu, k jakému se XPCOM v Mozile používá však tyto nedostatky vůbec nevadí. A konec konců jsme ve světě open-source, takže pokud by někdo tyto vlastnosti potřeboval, může si je dopsat sám.

Jedním z příjemných důsledků podobnosti XPCOM a COM bylo například snadné vytvoření ActiveX komponenty[25] s renderovacím jádrem Mozilly, která je 100% kompatibilní s analogickou komponentou Internet Exploreru. Využití spočívá ve vestavění této komponenty do aplikace, která potřebuje někde zobrazovat HTML kód – typickou ukázkou je funkce „náhled“ u HTML editorů. Na rozdíl od dřívějška mají nyní vývojáři možnost si vybrat, kterou komponentu do své aplikace použijí, případně můžou použít obě.

XPConnect[26]

Objektový model XPCOM není omezen jen na jazyk C++, ale dá se plně využívat i z JavaScriptu. Technologie XPConnect zajišťuje transparentní propojení obou „světů“. Má zejména tyto úkoly:

- zpřístupnění hierarchie objektů Mozilly v JavaScriptu
- ošetřit volání metod objektů implementovaných v různých jazycích s ohledem na volací konvence a typy parametrů
- synchronizovat garbage collector používaný v JS engine s počítáním referencí v XPCOM
- umožnit tvorbu nových komponent v JavaScriptu, včetně komponent odvozených z již existujících

Asi nejčastější využití XPConnect spočívá v tom, že kód vyšších vrstev napsaný v JavaScriptu (typicky se jedná o kód uživatelského rozhraní) využívá ke své práci komponenty z vrstev nižších, které jsou napsány v C++.

Gecko[27]

Nejdůležitější součást každého internetového prohlížeče je jeho renderovací jádro. V případě Mozilly je jím Gecko. Engine Gecko podporuje téměř úplně standardy HTML 4.01[28], XHTML 1.0[29], CSS 1[30] (částečně CSS 2 a 3), DOM 1[31] (částečně DOM 2), XML 1.0[32], RDF[33], JavaScript 1.5 (kompatibilní s ECMA-262 Edition 3), SSL,

---

[25] http://www.iol.ie/~locka/mozilla/mozilla.htm

[26] http://www.mozilla.org/scriptable/index.html

[27] http://www.mozilla.org/newlayout/

[28] http://www.w3.org/TR/1999/REC-html401-19991224/

[29] http://www.w3.org/TR/2000/REC-xhtml1-20000126/

[30] http://www.w3.org/TR/REC-CSS1

[31] http://www.w3.org/TR/REC-DOM-Level-1/

[32] http://www.w3.org/TR/REC-xml

[33] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Unicode, HTTP 1.1 (včetně perzistentních připojení a gzip komprese), PNG[34], GIF a JPEG. V podpoře těchto standardů je ve velké většině Gecko nejlepší na světě.

Při vývoji byla snaha udělat Gecko co nejrychlejší a nejmenší. To se ale spíše nepodařilo, protože Gecko na sebe postupně přebíralo kromě renderování i další úkoly (síťová knihovna, cache,...) a celá technologie se stala poněkud nabobtnalou.

Technologicky je jádro Gecko vystavěno nad XPCOM, což mu zajišťuje modularitu, ale také komplikuje jeho zdrojový kód. Gecko se skládá z následujících komponent:

- parser (samostatný pro HTML, XML a jiné dokumenty)
- layout engine
- subsystém pro práci se styly (CSS)
- knihovna pro práci s obrázky
- síťová knihovna (zvaná Necko)
- subsystém pro vykreslování prvků uživatelského rozhraní (platformě závislý)
- knihovna pro manipulaci u uživatelským nastavením
- API pro plug-iny
- Open Java Interface (OJI)
- subsystém pro práci s formátem RDF
- knihovna pro práci s fonty
- bezpečnostní knihovna
- systém diskové a paměťové cache

Ačkoliv je engine Gecko momentálně pravděpodobně nejlepší na světě z hlediska podpory standardů a kompatibility, existuje i několik jeho srovnatelných konkurentů, o kterých bych se rád stručně zmínil.

Trident (jádro Internet Exploreru na platformě Win32)

Jeho podpora standardů je pravděpodobně nejhorší ze všech zde zmiňovaných renderovacích enginů, bohužel ale jeho zhruba 90% zastoupení mezi uživateli nutí vývojáře omezit se při vývoji webových aplikací na funkce, které toto jádro podporuje. Krom toho obsahuje také hodně hloupých chyb, což při tvorbě stránek vadí ještě víc. Po technologické stránce je založeno na komponentové technologii Microsoft COM.

Renderovací jádro Opery

Prohlížeč Opera je v úrovni podpory standardů Gecku velice blízko a s každou novou verzí se zlepšuje. Kód jeho jádra je sice uzavřený, ale obecně je známo, že je jádro vystavěno multiplatformě a bylo značně přepsáno při vývoji verze Opery 7.0. Nejvyšší prioritou jádra Opery je rychlost vykreslování a nenáročnost, což se mu na rozdíl od Gecka daří dodržovat. Proto je také nasazováno do přístrojů s omezeným množstvím zdrojů (hlavně paměti a výpočetní síly), jako jsou různé elektronické diáře, PDA, apod.

---

[34] www.libpng.org/pub/png/

KHTML

KHTML je jádro používané především v prohlížeči Konqueror. Donedávna zaostávalo v některých oblastech (zejména pokročilejší CSS), ale nyní do jeho vývoje intenzivně přispívá firma Apple, která se rozhodla použít ho do svého nového prohlížeče Safari, a tak se situace rychle napravuje. Jádro je malé a rychlé, kód (v C++) je oproti Gecku velice přehledný – a to je také hlavní důvod, proč ho Apple použil namísto Gecka.

XUL (XML User Interface Language) [35]

Jedním z problémů, se kterým se programátoři Netscapu potýkali při vývoji starého Navigatoru, byla nutnost programovat uživatelské rozhraní zvlášť pro každou platformu. Během vývoje Mozilly padlo rozhodnutí tuto nutnost odstranit a tak byl vyvinut jazyk XUL. Tento jazyk slouží k popisu uživatelského rozhraní aplikace. Je založen na XML – jednotlivé elementy popisují ovládací prvky a jejich atributy vlastnosti těchto prvků. Celý soubor je pak vykreslován pomocí jádra Gecko. Díky tomu je v popisu rozhraní možné používat i běžné HTML tagy.

Funkčnost celému sytému dodává JavaScript, kterým jsou prvky pospojovány. Model je podobný jako u technologie DHTML – jednotlivé ovládací prvky mohou vyvolávat události, jejichž ošetření je řešeno v JavaScriptu. Díky technologii XPCOM může tento JavaScriptový kód využívat kompletní objektový model Mozilly se všemi dostupnými komponentami.

Vzhled ovládacích prvků může být upraven pomocí CSS. To dává bohaté možnosti skinování celé aplikace – se samotnou Mozillou se dovdávají skiny dva: klasický, připomínající starý Netscape Navigator, a nový (zvaný Modern), připomínající StarTrek :-)

Jazyk XUL je velice snadné se naučit (tvorba aplikací v XUL není o nic těžší než psaní webových stránek) a hodí se nejen na vývoj Mozilly, ale i nejrůznějších aplikací. Díky modulárnosti Mozilly je možné i vyvíjet v XUL samostatné aplikace, na Mozille úplně nezávislé.

XUL ale rozhodně není lék na všechno – má i své chyby. Dříve byla problémem především rychlost a náročnost celé technologie – přeci jen parsování XUL souborů a interpretování JavaScriptů něco stojí. Pro urychlení bylo v průběhu vývoje implementováno předkompilování XUL souborů (XUL Fastload[36]) a podobná věc je v plánu i pro JavaScript. Krom toho v kódu Gecka probíhá stále spousta optimalizací, takže Mozilla má tendenci se verzi od verze zrychlovat (to není jev u programů příliš obvyklý!). Dnešní počítače jsou ovšem také podstatně rychlejší než v začátcích vývoje, a tak náročnost přestává pomalu vadit.

V poslední době začal spíš u technologie XUL vadit nedostatek uživatelského komfortu – především nenativní vzhled celé aplikace (stejný na všech platformách). Největší problém to byl na Macintoshích, kde je uživatelské rozhraní značně odlišné od Windows a UNIXů a uživatelé na něj byli tradičně citliví. Na Macích se vše nakonec vyřešilo samostatným projektem Camino[37], který z Mozilly využívá jen renderovací jádro a uživatelské rozhraní je vytvořeno jako nativní. Na platformě

---

[35] http://www.mozilla.org/xpfe/xptoolkit/xulintro.html

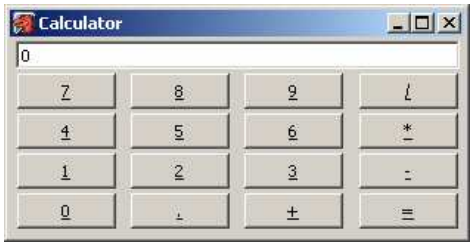
[36] http://bugzilla.mozilla.org/show_bug.cgi?id=134576

[37] http://www.mozilla.org/projects/camino/

Windows a UNIX (přesněji GTK) se nakonec vývojáři rozhodli implementovat podporu nativních uživatelských rozhraní přímo do technologie XUL (třída `nsNativeTheme`), takže ovládací prvky XUL vypadají na pohled i chováním stejně jako standardní ovládací prvky daného operačního systému.

Příklad

Jako ukázku použití jazyka XUL jsem zvolil jednoduchou kalkulačku. Zdrojový kód se skládá ze dvou souborů: `calc.xul` se specifikací rozhraní a `calc.js` s výkonným kódem. Myslím, že zdrojový kód je takřka samodokumentující:




```
<!-- calc.cul -->

<?xml version="1.0"?>

<!-- globální styl -->
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<!-- kořenový element - okno -->
<window id="calc-window" title="Calculator"
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <!-- import skriptu -->
  <script type="application/x-javascript" src="calc.js" />
  <vbox> <!-- vbox řadí při zobrazení své potomky pod sebe -->
    <!-- textové pole -->
    <textbox id="result" readonly="readonly" value="0" />
    <hbox> <!-- hbox řadí při zobrazení své potomky vedle sebe  -->
      <button label="7" accesskey="7" oncommand="num('7');" />
      <button label="8" accesskey="8" oncommand="num('8');" />
      <button label="9" accesskey="9" oncommand="num('9');" />
      <button label="/" accesskey="/" oncommand="op('/');" />
    </hbox>
    <hbox>
      <button label="4" accesskey="4" oncommand="num('4');" />
      <button label="5" accesskey="5" oncommand="num('5');" />
      <button label="6" accesskey="6" oncommand="num('6');" />
      <button label="*" accesskey="*" oncommand="op('*');" />
    </hbox>
    <hbox>
      <button label="1" accesskey="1" oncommand="num('1');" />
      <button label="2" accesskey="2" oncommand="num('2');" />
      <button label="3" accesskey="3" oncommand="num('3');" />
      <button label="-" accesskey="-" oncommand="op('-');" />
    </hbox>
    <hbox>
      <button label="0" accesskey="0" oncommand="num('0');" />
      <button label="." accesskey="." oncommand="num('.');" />
      <button label="+" accesskey="+" oncommand="op('+');" />
      <button label="=" accesskey="=" oncommand="eq();" />
    </hbox>
  </vbox>
</window>
```

```
/* calc.js */

num1 = "";    // první operand
num2 = "";    // druhý operand
oper = "";    // operátor
begin = true; // začít po stisku číslice psát nové číslo?
```

```
/* Vyvolá se po stisku číslice nebo ".". */
function num(n) {
  if (begin) num2 = "";
  num2 += n;
  document.getElementById("result").value = num2;
  begin = false;
}

/* Pomocná funkce na spočítání operace oper nad num1 a num2 a zobrazení
   výsledku. */
function do_op() {
  switch(oper) {
    case "+":
      num1 = parseFloat(num1) + parseFloat(num2);
      break;
    case "-":
      num1 = parseFloat(num1) - parseFloat(num2);
      break;
    case "*":
      num1 = parseFloat(num1) * parseFloat(num2);
      break;
    case "/":
      num1 = parseFloat(num1) / parseFloat(num2);
      break;
  }
  document.getElementById("result").value = num1;
}

/* Vyvolá se po stisku tlačítka operátoru. */
function op(o) {
  if (num1 == "")
    num1 = num2;
  else
    if (!begin)
      do_op();
  oper = o;
  begin = true;
}

/* Vyvolá se po stisku tlačítka ".". */
function eq() {
  if (!num1 == "") do_op();
  begin = true;
}
```

Nástroje na podporu vývoje

Při vývoji Mozilly bylo potřeba vytvořit i nejnutnější infrastrukturu pro koordinaci vývoje, který probíhá značně distribuovaně. Základ poskytuje program CVS[38], který umožňuje koordinovat příspěvky (patche) a jejich integraci do hlavního vývojového stromu. Krom CVS ale vývojáři používají i další nástroje, většinou vyvinuté firmou Netscape a nebo komunitou samotnou.

Bugzilla[39]

Nejdůležitějším pomocníkem při vývoji je Bugzilla, což je systém pro evidenci chyb a požadavků na další rozšíření (RFE, Request For Enhacement) Mozilly – dohromady se tomu v komunitě říká „bugs“, i když to striktně vzato vždy chyby nejsou. Technicky je Bugzilla poměrně jednoduchá webová aplikace napsaná v Perlu napojená na databázi. U každého bugu se eviduje vývojář, kterému je bug přiřazen, cílová verze, priorita a další položky. Systém umožňuje přidávání příloh k jednotlivým bugům, což jsou nejčastěji patche. Krom toho je u každého bugu i diskuze, jejíž příspěvky jsou zainteresovaným vývojářům zasálány automaticky na e-mail. Bugzillu dnes využívají i jiné organizace/projekty – např. NASA, RedHat, Gnome, FreeDOS a další.

Bonsai[40]

Bonsai je jednoduchý systém, umožňující sledovat příspěvky vývojářů (tzv. checkins) do vývojového stromu. Opět je to webová aplikace, propojená s CVS.

LXR[41]

LXR je interaktivní prohlížeč zdrojového kódu Mozilly. Je možné procházet jednotlivé adresáře, vyhledávat a především zobrazovat zdrojové soubory. Všechny identifikátory proměnných, názvy tříd a datových typů jsou zobrazeny jako hypertextové odkazy, které vedou na definice příslušných prvků. Díky tomu je procházení zdrojového kódu podstatně pohodlnější než při prostém prohlížení v nějakém editoru.

Tinderboxy[42]

V areálu firmy Netscape běží 24hodin denně několik strojů (zvaných „tinderboxy“), které dělají stále jednu věc – sestavují aktuální verzi Mozilly pro jednotlivé platformy a provádějí na ní základní testy. V případě, že se do vývojového stromu dostane kód, který způsobí chybu v sestavovacím procesu, projeví se to do několika desítek minut na některém z tinderboxů a vývojáři mohou vše rychle opravit. Faktem je, že narušení sestavení je i přes poměrně přísná pravidla pro přidávání a úpravy kódu vcelku častá věc a dochází k němu běžně několikrát týdně.

---

[38] http://www.cvshome.org/

[39] http://www.bugzilla.org/

[40] http://bonsai.mozilla.org/

[41] http://lxr.mozilla.org/

[42] http://tinderbox.mozilla.org/showbuilds.cgi

Budoucnost projektu Mozilla

Jaká je budoucnost projektu Mozila je nyní poměrně těžké odhadnout. Celý projekt byl z velké části tažen firmou Netscape, což nyní je (nebo spíš byla) divize AOL. Původně se odhadovalo, že Netscape/AOL projekt Mozilla zajímá zejména z důvodu integrace jádra Gecko do AOL klienta (software používaný v USA zákazníky AOL na práci s internetem). O to větším překvapením pak bylo zveřejnění dohody[43] AOL s Microsoftem, z níž mimo jiné plynulo, že do AOL klienta bude integrováno jádro Internet Exporeru. V polovině července 2003 pak došlo k přerušení[44] podpory vývoje Mozilly ze strany Netscape/AOL. Zaměstnanci pracující na Mozille byli propuštěni nebo přeřazeni a v následujících měsících se postupně hlavním tahounem celého projektu stane nově založená nezisková organizace Mozilla Foundation[45].

Faktem je, že vývoj se pravděpodobně úplně nezastaví, ale každopádně o něco zpomalí a zkomplikuje. Netscape/AOL totiž kromě programátorů poskytoval i své technické vybavení, na jejich serverech běží i Bugzilla a veškerá další podpůrná infrastruktura projektu. Naštěstí Netscape/AOL přislíbil technickou infrastrukturu (a dokonce i veškerá práva s Mozillou související) přenechat nadaci Mozilla Foundation.

Na druhou stranu zajímavě působí oznámení Microsoft, že již nebude vyvíjet samostatné verze Internet Explorer na žádné platformě. Podle zpráv prosáklých z Microsoftu čítá nyní vývojová skupina Internet Exploreru jen několik vývojářů, kteří převážně opravují zjištěné bezpečnostní chyby.

Pokud Microsoft neblafuje, pak je zde šance, že se prohlížeče založené na jádru Gecko postupně prosadí aspoň do té míry, aby to donutilo tvůrce stránek vzdát se technik „Internet Explorer only“ a držet se internetových standardů. Není snad třeba říkat, že to internetu jakožto multiplatformnímu prostředí jen prospěje.

Změna celkové koncepce aplikace

Celá Mozilla podědila po Netscape 4.x monolitickou koncepci – všechny aplikace (Navigator, Mail,...) jsou všechny v jednom balíku a propojené, dokonce jsou fyzicky jeden spustitelný soubor. Tato koncepce měla své výhody, například méně zabrané paměti pokud měl uživatel spuštěných víc aplikací najednou, jenže vedla k takovým absurditám, jako že se dá z menu Tools v Mozilla Mail spustit třeba JavaScript Debugger.

K tomu je třeba připočíst nejasné zaměření celého balíku. Všechny verze Mozilly jsou totiž oficiálně vydávané jen k testovacím účelům a nikoliv pro koncové uživatele. Předpokládalo se, že koncový uživatel si stáhne příslušnou odvozenou verzi Netscape. Proto byly do programu integrovány i nástroje typu DOM Inspector, které jsou sice vhodné pro webdesignera, ale nikoliv už pro řadového uživatele.

---

[43] http://www.washtimes.com/business/20030529-113757-3865r.htm

[44] http://slashdot.org/article.pl?sid=03/07/16/1359253

[45] http://www.mozillafoundation.org/

Časem se ukázalo, že Mozillu stahují i koncoví uživatelé, kteří navíc typicky nechtějí celý balík aplikací, ale jen jednu konkrétní (většinou jen prohlížeč). Postupně se tedy přistoupilo k „reformě[46]“.

V průběhu jara 2003 bylo rozhodnuto rozdělit původně monolitickou aplikaci na víc samostatných částí. Ty budou určeny přímo pro koncové uživatele a podle toho budou také vypadat. Pokročilejší funkce si bude možné stáhnout ve formě „extenstions“ – jakýchsi pluginů.

Tato koncepce se setkala s kladným ohlasem v celé komunitě Mozilly a nyní je Mozilla ve stadiu „přerodu“. Pracuje se zejména na dvou nejdůležitějších komponentách, kódově pojmenovaných Mozilla Firebird (prohlížeč, momentálně ve verzi 0.6.1) a Mozilla Thunderbird (mailový klient, ve verzi 0.1). Zatím poslední vydaná verze klasické Mozilly – 1.4 – by měla poskytovat stabilní základnu pro vývoj aplikací založených na Mozille do té doby, než bude přechodová fáze dokončena a aplikace budou považovány za stabilní.

Příspěvky od komunity

Vzhledem k tomu, že tvorba rozšíření Mozilly není díky technologii XUL nic těžkého, vznikla velmi rychle spousta satelitních projektů, které se snaží do Mozilly dodat vlastnosti, které jí podle názoru různých skupin uživatelů chybí. Za všechny jmenujme lepší podporu tabbed browsingu[47] nebo integraci kontroly pravopisu[48].

Většina projektů je soustředěna na serveru MozDev, který jim poskytuje i nutné technologické zázemí.

V nové architektuře Mozilly budou mít příspěvky od komunity mnohem větší roli než dosud, protože dojde k redukci nadbytečných vlastností.

---

[46] http://www.mozilla.org/roadmap.html

[47] http://multizilla.mozdev.org/

[48] http://spellchecker.mozdev.org/

Závěr

Tato práce měla za úkol seznámit čtenáře s architekturou Mozilly, s vrstvami jejího zdrojového kódu a způsobem, jakým je Mozilla vyvíjena. Snad se to podařilo. Díky rozsahu práce se bohužel nedostalo na spoustu dalších zajímavých témat – za všechny jmenujme:

- podrobnější pohled na organizaci vývoje
- důkladnější popis technologie XPCOM
- více příkladů, co lze dělat s XUL
- pravidla pro psaní zdrojového kódu včetně jejich zdůvodnění
- bližší pohled do komunity vývojářů

Snad někdy příště.

Zdroje informací

Při psaní práce jsem vycházel 100% ze zdrojů na internetu, protože v Čechách není o Mozille prakticky žádná literatura dostupná. Poměrně dost odkazů na dokumenty, ze kterých jsem čerpal, je ve formě poznámek pod čarou uvedeno přímo v práci. Zájemci o Mozillu by rozhodně neměli minout následující webové servery:

- http://www.mozilla.org/
- http://www.mozillazine.org/
- http://www.mozdev.org/
- http://www.w3c.org/
- http://devedge.netscape.com

O tvorbě aplikací v Mozille vyšly v USA i dvě knihy: *Creating Applications with Mozilla* a *Creating XPCOM Components*. Druhá z nich je dostupná i v elektronické formě na adrese http://www.mozilla.org/projects/xpcom/book/cxc/.