WEBL – WEB LANGUAGE

Martin Molhanec

ČVUT, Elektrotechnická fakulta v Praze, Katedra elektrotechnologie
Technická 2, 166 27 PRAHA 6, Dejvice, Česká republika
Telefon: ++420(2)24352118, mailto://molhanec@fel.cvut.cz, http://martin.feld.cvut.cz/~mmm

Abstrakt

The goal of this paper is to present object oriented language WebL, or Web Language. WebL is a programming language designed specifically for the purpose of automating processing tasks over the documents on the Web. It incorporates an implementation of service combinators and markup algebra. The WebL is interpreter written in well-known Internet Java language. The WebL is owned by COMPAQ but grants you the right to use Compaq's Web Language including the sources and modifications to the sources, on any number of computers for your use only.

1. Úvod

Účelem tohoto příspěvku je seznámit širokou odbornou veřejnost se zajímavým objektovým programovacím jazykem a nástrojem – Web Language (zkráceně *WebL*) firmy *Compaq*, který je speciálně určen pro získávání informací z HTML stránek. Tento jazyk implementuje speciální vlastnosti *service combinators* a *markup algebra*. Přestože je tento jazyk vlastněn firmou *Compaq,* je ho možné zcela zdarma využívat včetně možnosti modifikace jeho zdrojových textů. Text tohoto příspěvku rozšiřuje a doplňuje text obdobného příspěvku předneseného na konferenci *Objekty'2000*.

2. Vlastnosti

Pod pojmem WebL je nutné rozumět jednak samotný programovací jazyk a dále interpreter tohoto jazyka napsaný v jazyce Java. WebL byl navržen ve výzkumném centru firmy *Compaq*, dříve výzkumném centru firmy *Digital* pány *Thomasem Kistlerem* a *Hannesem Maraisem* v letech 1998 až 1999. Na celém projektu však pracovala celá řada dalších výzkumných pracovníků, kteří se podíleli na jeho různých částech.

Bohužel, přestože je již první rok třetího tisíciletí, zůstává vývoj tohoto nesmírně zajímavého jazyka na mrtvém bodě. Stránky firmy *Compaq*1 zůstávají beze změn již od roku 1999! Poslední verze *Webl 3.0h* je ze dne *8 June 1999* a tak zůstává jedině živá webová konference, která na stránkách firmy *Compaq* stále funguje. Podobně se zájemce o tento nástroj musí vyrovnat se skutečností, že stažený interpreter musí pozměnit, aby bez chyby fungoval pod Java SDK 1.3.

Jazyk *Webl* vznikl za účelem zpracování dokumentů uložených na WWW serverech sítě Internet. Pro tyto účely byl navržen a je pro ně nanejvýš vhodným nástrojem. Je vhodný zejména pro automatizaci takovýchto úkolů a umožňuje snadnou komunikaci prostřednictvím webových protokolů HTTP a FTP a umí také zpracovávat dokumenty ve formátu prostého textu (*plain text*), HTML i XML.

2.1 Základní vlastnosti

- Jazyk *WebL* je interpreter implementovaný pomocí jazyka *Java*.
- Jazyk *WebL* je objektově orientovaný, obsahuje speciální datové typy (list, set, aj.) a podporuje zpracování výjimek.
- Jazyk *WebL* podporuje vytváření *modulů*, umožňuje volat funkce napsané v jazyce *Java* a naopak.
- Jazyk *WebL* podporuje multitasking v úrovni javovských vláken.
- Pomocí jazyka *WebL* je možné jednoduše stahovat dokumenty uložené na WWW serverech.
- Tyto stažené dokumenty jsou automaticky podrobeny analýze a celý stažený dokument je uložen jako kolekce speciálních objektů.
- Jazyk *WebL* poskytuje speciálně vytvořenou algebru pro práci s uloženým dokumentem, která umožňuje uživateli přesně najít danou informaci v daném dokumentu.
- Jazyk *WebL* umožňuje na základě stažených stránek vytvářet stránky nové, včetně jejich doplnění nebo zkrácení.
- Je ho také možné použít na inteligentní stahovaní objektů vložených do HTML stránek, například obrázků.
- Programy napsané v jazyce *WebL* je možné používat jako *servlety* vložené do HTML stránek.
- Jazyk *WebL* umožňuje vytvoření jednoduchého webového serveru. Potom se jazyk *WebL* stává *server side* skriptovacím jazykem.
- Jazyk *WebL* je také velmi vhodný pro rychlé prototypování webových aplikací, zejména s ohledem na jejich automatizaci.
- WebL je vysoce flexibilní nástroj a poskytuje vysokou úroveň abstrakce, spíše nežli výkonnost, a proto se hodí více jako nástroj pro rychlé vytváření prototypů, než jako prostředek pro vytváření výkonných cílových aplikací.

2.2 Programovací jazyk

- WebL je programovací jazyk vysoké úrovně, objektový, imperativní, interpretovaný, s dynamickými typy a s podporou vláken
- Standardní typy jazyka WebL jsou: boolean, character, integer (64-bit), double precision floats, Unicode strings, lists (seznam), sets (množina), associative arrays (objects), functions, and methods.
- WebL jazyk obsahuje možnost vytvářet objekty pomocí prototypů
- WebL jazyk obsahuje typy množina a seznam
- Součástí jazyka WebL jsou speciální předdefinované typy pro zpracování HTML/XML stránek a jejich částí (page, pieces, piece sets, tags).
- WebL obsahuje standardní řídící struktury: if-then-else, while-do, repeat-until, try-catch, atd.
- Syntaxe jazyka WebL je postavena na syntaxi jazyků C a Modula.
- Jazyk WebL podporuje ošetření výjimek a paralelní zpracování.

2.3 Práce s protokoly

- WebL podporuje ty samé protokoly co jazyk Java, tj. HTTP, FTP, atd.
- WebL dokáže HTML stránky nejen stahovat, ale dokáže též odesílat vyplněné formuláře a provádět navigaci mezi stránkami.
- WebL podporuje práci s HTTP cookies.

- WebL jazyk podporuje práci s HTTP hlavičkami, jak při jejich odesílání, tak při zpracování HTTP hlaviček při jejich příjmu.
- WebL podporuje práci s PROXY a základními typy HTTP autorizace.

2.4 Markup algebra

- WebL rozpoznává prostý text (*plain text*), HTML a XML mime typy.
- WebL užívá rozšiřitelný DTD-based HTML parser, podporující HTML 2:0, 3.2 a 4.0.
- WebL obsahuje speciální markup algebru pro extrahování elementů a textů ze stránek HTML a speciální funkce pro manipulaci s obsahem těchto stránek. Funkce pro extrakci podporují PERL 5 regulární výrazy.
- Elementy stránek jsou mapovány do objektů a jejich vlastnosti do atributů těchto objektů, jsou tedy snadno dostupné přímo z jazyka WebL.
- Pomocí markup algebry je umožněno snadné vytváření složitých dotazů, například: „najdi všechny obrázky ve třetí řádce tabulky, která obsahuje slovo ‚ABC'".
- WebL může pracovat i s překrývajícími se elementy, protože nevyužívá obvyklou stromovou reprezentaci stránky HTML.
- Manipulace se stránkou zahrnuje modifikaci atributů, mazání elementů, kopírování elementů a jejich záměnu.

2.5 Modularita

Jazyk WebL zahrnuje celou řadu standardních modulů (knihoven) pro následující funkce.
- Možnost ukládání stažených stránek na disk.
- Zobrazení stránek v prohlížeči (pouze ve Windows)
- Podpora paralelního zpracování úloh
- Základní manipulace s řetězci, včetně PERL 5 regulárních výrazů.
- Práce s URL řetězci.
- Jednoduchý WEB crawler s podporou úloh.
- Jednoduchý WEB server s podporou úloh a jazyka WebL.
- Java servlet podporu.

2.6 Podpora jazyka Java

- Interpreter jazyka WebL je napsán kompletně v jazyce Java. Díky této skutečnosti je kompletně přenositelný mezi různými platformami (Windows, UNIX, atd)
- Je možné jazyk WebL využívat pomocí volání z jazyka Java.
- Je velmi snadné z jazyka WebL volat funkce napsané v jazyce Java a přistupovat k objektům vytvořeným v jazyce Java.

3. Kombinované služby (Service Combinators)

V jazyce *WebL* se pojmem *service* rozumí *služba*, která umožňuje přístup ke zdrojům na Internetu. Pojmem *service combinators* se rozumí možnost tyto služby speciálním způsobem kombinovat, zejména s ohledem na nespolehlivost Internetu jako přenosového prostředí.

3.1 Základní služba

Základní službou je služba stažení dokumentu z Internetu implementovaná pomocí funkce *getURL* a *postURL* s následující syntaxí.

```
GetURL(url, [. param1=val1, param2=val2, ... .])

PostURL(url, [. param1=val1, param2=val2, ... .])
```

Všimněme si, že druhý parametr je objekt, který tak snadno umožňuje předávat předem neznámý počet parametrů! Jednoduché použití si hned ukážeme.

```
// This program looks up the word "java" on the
// AltaVista search engine.
page = GetURL(
"http://www.altavista.digital.com/cgi-bin/query",
[. pg="q", what="web", q="java" .])
```

3.2 Sekvenční kombinace služeb

Význam sekvenční kombinace je následující. Systém se nejprve pokusí vykonat první službu, a pokud tato služba selže, pokusí se vykonat druhou službu. Operátor sekvenční kombinace je znak *„**?**“*.

```
// This program first attempts to connect to AltaVista
// in California, and in the case of failure,
// attempts to connect to a mirror in Australia.
page = GetURL("http://www.altavista.digital.com") ?
GetURL("http://www.altavista.yellowpages.com.au")
```

3.3 Paralelní kombinace služeb

Při paralelní kombinaci systém spustí obě služby současně, a ta která bude dříve hotova vrací hodnotu. Operátorem paralelní kombinace je znak *„|“*.

```
// This program attempts to fetch a page from one
// of the two alternate sites. Both sites are
// attempted concurrently, and the
// result is that from whichever site
// successfully completes first.
page = GetURL("http://www.altavista.digital.com") |
GetURL("http://www.altavista.yellowpages.com.au")
```

Dalšími kombinátory jsou *Timeout*, *Retry* a *Stall*. Nicméně pro potřeby tohoto článku nepovažuji za nutné se jimi zabývat.

4. Práce s HTML stránkami

Nyní obrátíme naší pozornost ke způsobu s jakým WebL pracuje se stránkami HTML. Uvedeme si opět jednoduchý příklad, který si vysvětlíme:

```
var P = GetURL("http://www.nowhere.com");      // 1 radka
var images = Elem(P, "img");                   // 2 radka
every image in images do                       // 3 radka
   PrintLn(image.src)                          // 4 radka
End                                            // 5 radka
```

První řádka nám ilustruje použití jedné ze základních funkcí WebL, která nám umožní stáhnout danou HTML stránku z WWW serveru. Jedná se o funkci *GetURL*, jejímž parametrem je URL adresa a která vrací objekt *Page*. Úplná dokumentace parametrů funkce GetURL je v dokumentaci WebL. Řádka 2 ukazuje použití funkce Elem, jejíž první parametr je objekt stránka a druhý je řetězec, který udává jaký typ elementů chceme ze stránky vyfiltrovat. Funkce vrací množinu elementů (*pieces set*), které jsou v našem případě typu *image*. Řádek 3 nám uvozuje příkaz cyklu (který je ukončen na řádku 5) typu *every item in set do ... end*. Tento konstrukt nám umožňuje do proměnné *image* získat postupně jednotlivé objekty typu *piece* z množiny *images.* Na řádce 4 posléze tiskneme na standardní výstup hodnotu atributu *src* objektu *image*.

5. Algebra značek (Markup Algebra)

Algebra značek je snad absolutně nejlepším přínosem jazyka *WebL* pro zpracování dokumentů stažených z Internetu. Stažený dokument napsaný v jazyce HTML se skládá ze značek. Příklad takové jednoduché značky je například značka *HEADING1*.

```
<h1 ID="heading 1">Hello World !!!</h1>
```

V jazyce *WebL* je každá taková značka objektem, který je součástí kolekce stažené stránky. Abychom si objasnili práci se značkami v jazyce *WebL*, uvažujme následující příklad.

```
// zakladni pokus se značkami v jazyce WebL                //1
var Page = GetURL("file:///c:/Webl/pokusy/pokus1.htm");     //2
PrintLn(Page);                                             //3
var Headings1 = Elem(Page,"h1");                           //4
every Tag in Headings1 do                                  //5
                    PrintLn(Tag);                          //6
                    PrintLn(Tag.id);                       //7
                    PrintLn(Name(Tag));                    //8
                    PrintLn(Text(Tag));                    //9
end;                                                       //10
```

Pomocí řádky číslo 2 přečteme stránku, v našem případě lokální na počítači. Řádkou číslo 3 ji vytiskneme. Řádkou číslo 4 získáme pomocí funkce *Elem* kolekci *Headings1*, která obsahuje všechny značky typu *H1 (Titulek úrovně 1)*. Řádky 5 a 10 nám ohraničují cyklus typu *every ...in ... do ... end* pomocí kterého do dočasné proměnné *Tag* získáváme postupně všechny prvky kolekce *Headings1*. Řádkou číslo 6 vytiskneme značku. Řádka číslo 7 nám vytiskne hodnotu atributu *id* naší značky. Řádkou číslo 8 vytiskneme název značky a řádkou číslo 9 vytiskneme obsah značky (vše co je umístěno mezi počáteční a koncovou značkou). Výsledek, pokud náš prográmek uplatníme na výše uvedený HTML zdroj, je následující.

```
[. "URL" = "file:/j:/Work/Webl/pokusy/pokus1.htm",     //1
"content-length" = "188" .]                            //2
[. "id" = "heading 1" .]                               //3
heading 1                                              //4
h1                                                     //5
Hello World !!!                                        //6
```

Na řádce číslo 1 a 2 je vypsán obsah naší stránky v interním formátu jazyka *WebL*. Je vidět, že obsahem výpisu je objekt se dvěma atributy: *URL* a *content-length*. Na řádce číslo 3 je vypsán obsah naší značky v interním formátu jazyka *WebL*. Je vidět, že obsahem výpisu je objekt s atributem *id*. Na řádce číslo 4 je vypsána hodnota tohoto atributu, na řádce 5 je název naší značky a na řádce 6 je vypsán obsah naší značky.

Jazyk WebL však dokáže s HTML stránkami daleko složitější manipulace nežli bylo ukázáno na výše uvedené ukázce. Je to možné zejména díky zabudované speciální algebře vybudované nad množinami elementů HTML stránek. Uvedeme si opět několik ukázek, které nám předvedou bohaté možnosti, které nám jazyk WebL při práci s obsahem HTML stránek poskytuje:

```
// Extract the 2'nd row of the 3'rd table.
Elem(Elem(X, "table")[3], "tr")[2]


// Extract the 2'nd row of the table containing the
// word WebL
var t = Elem(X, "table") contain Pat(X, "Webl");
(Elem(X, "tr") inside t)[2]


// Retrieve all the H2's before the appendix
// (assuming only a single appendix is present).
Elem(X, "h2") before
(Elem(X, "h1") contain Pat(X, "Appendix"))


// Retrieve all the headings before Chapter 4
// inclusive.
Elem(X, "h1") !after
(Elem(X, "h1") contain Pat(X, "Chapter 4"))


// Retrieve all the italic elements not in a table.
Elem(X, "i") !inside Elem(X, "table")

// Locate the Parent element of the second table.
Parent(Elem(P, "table")[1])

// Insert an image at the beginning of each h1 tag.
var p = NewPiece("<img scr=a.gif>", "text/html");
every x in Elem(P, "h1") do
```

```
        InsertAfter(BeginTag(x), p)
    end
```

6. Implementace objektů v jazyce WebL

Jazyk WebL je objektově orientovaný jazyk, který umožňuje pracovat s objekty. Na rozdíl od klasických objektově orientovaných jazyku, jako jsou například jazyky C++ nebo Java, se podobá spíše jazyku Javascript, protože též neobsahuje možnost definování tzv. tříd.

Pro vytvoření nového objektu se využívá konstruktor [. .]. Objekt může obsahovat položky různých typů, které mohou uchovávat proměnné, funkce nebo metody uvnitř objektu. Pro specifikování dané položky v objektu je používáno dvou způsobů notace:

```
O.X    specifikuje položku X objektu O
O[E]   vyhodnotí výraz E na hodnotu X
```

Druhý způsob je vlastně práce s asociativním polem. Objekty v jazyce WebL a asociativní pole jsou vlastně jedno a to samé. O.X a O[“X”] odkazují na tu samou položku. Přístup k neexistující položce vyvolá výjimku. Uvedeme si nyní několik příkladů práce s objekty:

```
[. .]                       // The empty object
[. x = 1, y = 1 + 1 .]      // Object with x & y field
var o = [. x = 1 .]         // Object o initialized
o.x                         // Field x of o
o["x"]                      // The same field again
o.y := "hello"              // Defines field y of o
o[1+2] := 42                // Defines field 3 of o
o[4-1]                      // Accesses field 3 of o
Size(ToList(o))             // # fields of o
DeleteField(o, "x")         // Remove field "x"
```

Objektově orientované programování ve WebL je poměrné snadné. V následujícím příkladu si vytvoříme objekt mujucet s jednou položkou (zustatek) a dvěma metodami (uloz a vyber).

```
var mujucet    = [.
      zustatek = 0,
      uloz     =  meth(self, mnozstvi)
                    self.zustatek = self.zustatek + mnozstvi;
      end,
      vyber    =  meth(self, mnozstvi)
                    self.zustatek = self.zustatek - mnozstvi;
      end
.];

mujucet.uloz(100); // uloz 100Kc
mujucet.vyber(50); // vyber 50Kc
```

Pokud se někomu snad může zdát, že se neobejde bez tříd je snadné vytvářet nové objekty pomocí funkce, která nám třídy do určité míry nahrazuje. Přepíšeme si předchozí příklad s využitím funkce nahrazující nám třídu.

```
var mujucetTrida = fun()
      [.
         zustatek = 0,
         uloz     =  meth(self, mnozstvi)
                     self.zustatek = self.zustatek + mnozstvi;
         end,
         vyber    =  meth(self, mnozstvi)
                     self.zustatek = self.zustatek - mnozstvi;
         end
      .]
end;

var mujucet = mujucetTrida();

mujucet.uloz(100); // uloz 100Kc
mujucet.vyber(50); // vyber 50Kc
```

7. Příklady

Na závěr si uvedeme dva jednoduché příklady, které představují kompletní WebL programy plnící však poměrně složité funkce. První příklad demonstruje možnost vytvoření jednoduchého prohledavače, který prochází dané WWW sídlo.

```
// A demonstration web crawler that restricts its
// crawl to a specific site.
//

import Str, WebCrawler;       // zde se importuji standardni
                              // knihovni moduly

var MyCrawler = Clone(WebCrawler_Crawler,
  [.
    Visit = meth(s, page)
      var title = Text(Elem(page, "title")[0]) ? notitle";
      PrintLn(page.URL, " title=", title);
    end,

    ShouldVisit = meth(s, url)
      Str_StartsWith(url, `http://.*[.]yahoo[.]com`)
       and
       Str_EndsWith(url, "(/)|(.html?)")
    end,
  .]);
```

```
// vyse uvedeny prikaz CLONE vytvari pomoci dedicnosti ze // standardni tridy
WebCrawler_Crawler naší tridu MyCrawler
// s predefinovanymi virtualnimi metodami Visit a
// ShouldVisit. Metoda Visit nam vypisuje, ktere stranky
// nas program navstivi a metoda ShouldVisit urcuje, ktere
// stranky nas program muze navstivit


// uzivej 3 vlakna při sve praci
MyCrawler.Start(3);
// definice ktere WWW sidlo se bude prohledavat
MyCrawler.Enqueue("http://www.yahoo.com/");

stall()  // hlavni program usne, zatimco hledac pracuje
         // a vypisuje kudy prochazi
```

Druhý příklad je trochu jiný. Náš program zadá dotaz vyhledávacímu stroji HotBot a vytiskne první vrácenou stránku.

```
// Script to query the HotBot search engine
// and print the first page of results.
//

var P = GetURL("http://www.hotbot.com/default.asp",
                          [. MT="java" .]);
// zadani dotazu pro HotBot, vsimnte si zadani query

var H = Seq(P, "b br # br");
// funkce Seq vraci elementy urcene vyrazem "b br # br"
// tento vyraz vraci text uzavreny mezi znacky <br>

// nasledujici cyklus prochazi vsechny elementy
// a tiskne udaje Title a Abstract
every h in H do
    var a = Elem(h[0], "a")[0];
    PrintLn("Title: ", Text(a));
    PrintLn("Abstract: ", Text(h[2]));
    PrintLn();
    end;
```

Součástí dokumentace WebL je mnoho dalších příkladů, které ukazují jeho nepřeberné možnosti pro práci s obsahem WWW serveru a jsou nesmírně inspirativní.

8. Závěr

Účelem tohoto příspěvku je seznámit odbornou veřejnost s programovacím jazykem WebL. Tento jazyk je speciálně určený na dolování informací z HTML stránek. Významným rysem tohoto jazyka je jednak jeho objektová orientace a potom speciální algebra vybudovaná nad HTML stránkami, která umožňuje velice silné operace nad elementy tvořící HTML stránky. Přestože je WebL nástroj pro výše uvedené účely speciálně určený, je vhodný spíše pro

prototypování takovýchto aplikací, protože jeho implementace jako interpreter napsaný v jazyce Java nevyniká maximální výkonností. Nicméně umožňuje i v takovýchto případech odladit s výhodou nejprve algoritmus aplikace, který je pak možné implementovat například v jazyce C.

Je nesmírná škoda, že firma *Compaq* na další vývoj jazyka *WebL* nevěnuje žádnou energii. Na druhé straně je však velice potěšující, že je tento skvělý nástroj stále zdarma. Přestože se jedná o poměrně neznámý nástroj, lze doufat, že se bude postupně počet jeho uživatelů zvětšovat a že se opět obnoví jeho vývoj.

9. Zdroje

1. http://www.research.digital.com/SRC/WebL