Semestrální projekt

KP, FEL, ČVUT
Jan Pospíšil, pospij5@fel.cvut.cz
14. února 2008

PHPEditor

Editor zdrojového kódu jazyka PHP

psaný v jazyce PHP s rozšířením PHP-GTK
http://www.vpp-net.com/fel/pjp/phpeditor

Obsah

Úvod

Pro začátek se podíváme na některé pojmy, které by bylo dobré znát.

Co je to GTK+[1][3]

GTK+ (The GIMP ToolKit) je soubor knihoven určených pro tvoru grafického rozhraní programů. Z počátku byly určeny jen pro grafický editor GIMP (pro něj byly původně vyvíjeny), ale postupem času se tyto knihovny rozšiřovaly, vylepšovali a staly se tak použitelnými pro široké spektrum projektů[2]. Nyní se knihovny GTK+ kromě jiného pyšní multiplatformností (jsou vydávány zdrojové kódy, vývojové balíčky i předkompilované binární knihovny pro operační systémy GNU/Linux and Unix, Windows i Mac OSX), širokou podporou jazyků (a to jak jejich zobrazováním, tak překladem komponent knihoven) či rozsáhlým výběrem API pro nejrůznější programovací jazyky.

Základ balíku GTK+ tvoří čtyři knihovny:

GLib – knihovna nejnižší úrovně, která zprostředkovává datové struktury, správu smyčky událostí, zajišťuje bezpečné využití vícevláknových vlastností operačních systémů, dynamické načítání modulů a jiné. To vše v závislosti na operačním systému, pro který je daná verze určená – zajišťuje tím tak mezisystémovou přenositelnost programů psaných s využitím těchto knihoven.

Pango – knihovna která se stará o zobrazování textu. Klade důraz na podporu nejen obvyklých jazyků využívajících latinku, ale přináší podporu i pro jiné abecedy, pro vícebajtové kódování či pro obousměrný text. Zároveň obstarává správu fontů.

Cairo – tato knihovna se stará o 2D grafiku, o samotné vykreslování jednotlivých prvků grafického uživatelského rozhraní. Přináší jak podporu pro různé grafické systémy (X Window systém, Win32, …), tak maximální využití hardwarové akcelerace, která je na daném zařízení k dispozici.

ATK – poslední ze čtveřice základních knihoven se stará o přístupnost aplikací – zajišťuje podporu pro čtečky obrazovek, lupy otevřených oken či zařízení alternativního uživatelského vstupu.

Jak se tedy GTK+ „používá“? Programátor napíše aplikaci za použití vývojového balíčku. Tato je úměrně menší o obecné procedury pro realizaci grafického uživatelského rozhraní, ale nedokáže běžet bez takzvaných běhových knihoven. Uživatel si musí nainstalovat běhové knihovny (v objemu několika megabytů) a pak již může spouštět aplikace napsané pro běh v tomto prostředí. Příkladem rozsáhlejšího využití knihoven GTK+ je GNOME, grafické prostředí operačního systému Linux, které je celé postavené na GTK+.

Co je to PHP[7]

PHP je skriptovací jazyk, který se vyvíjí[9] již od roku 1995. Využívá se především jako skriptovací jazyk na webových serverech, kde umožňuje dynamické generování stránek, zpracovávání informací získaných od uživatelů, komunikaci s databázemi a mnoho dalšího. Jeho výkonné možnosti lze ale využít i v „desktopových“ aplikacích, tedy aplikacích běžících na uživatelském počítači.

Základem interpretru PHP (použitelného pro „desktopové“ aplikace v OS Windows) je jeden výkonný .EXE soubor a jedna dynamická knihovna (oba soubory o velikosti necelých 6 megabytů). Tento základ lze rozšiřovat o další funkce prostřednictvím zásuvných modulů (dynamických knihoven) produkovaných vývojovou skupinou projektu PHP i třetími stranami.

Co je to PHP-GTK[10]

Jedním z pluginů, kterými jde PHP rozšířit, je právě PHP-GTK. Aplikacím psaným v jazyce PHP zpřístupňuje většinu základních funkcí, které nabízí soubor knihoven GTK+. Při použití trojice běhové knihovny GTK+ – PHP – PHP-GTK jdou vyvíjet interpretované aplikace s grafickým uživatelským rozhraním. Příkladem takovéto aplikace je právě PHPEditor, o kterém pojednává tato práce.

Co je to lexikální analýza

Lexikální analýza je jeden z prvních kroků analyzátorů zdrojového kódu (například součást překladačů), který zajišťuje zpracování vstupních dat a rozlišení základních lexikálních elementů (jako je například znak „=“, proměnná, či klíčové slovo „`if`“). Lexikální analyzátor vlastně zjišťuje správné „pořadí písmen a znaků“ a zpracovává zdrojový kód na vyšší logické celky.

Co je to syntaktická analýza

Dalším krokem analýzy zdrojového kódu je syntaktická analýza. Pracuje již s lexikálními elementy a za pomoci syntaktických pravidel zjišťuje, zda je zdrojový kód „správně“ napsán (zda jsou dodržena pravidla jednotlivých jazykových struktur). Výstupem syntaktické analýzy může být jednoduché konstatování ano, zdrojový kód je správně napsán / ne, ve zdrojovém kódu je chyba. Pokud se ale syntaktická pravidla rozšíří o systém výstupu, lze syntaktickou analýzou generovat jinou formu zdrojového kódu. Tímto způsobem je realizován například překlad programu do binární podoby.

PHPEditor

Jako cíl této práce jsem si stanovil udělat textový editor, který by umožňoval zvýrazňování syntaxe a syntaktickou kontrolu pro zdrojové kódy (v testovací fázi vývoje jsem si jako příklad vybral jazyk PHP). Spíše než na vytvoření plnohodnotného editoru zdrojového kódu se hodlám soustředit na tvorbu co nejuniverzálnějšího analyzátoru, jak lexikálního, tak syntaktického. V samotném kódu aplikace chci realizovat jen řídící struktury konečných automatů pro oba analyzátory a konkrétní „nastavení“ (možné konfigurace) těchto automatů umístit do konfiguračních souborů. Změnou konfiguračního souboru poté bude možné rozšířit již napsaný editor o jiné programovací jazyky, případně doplnit definice již existujících jazyků (a to bez zásahu do zdrojových kódů samotné aplikace).

Realizace

Práci jsem se rozhodl psát v interpretovaném programovacím jazyku PHP s rozšířením PHP-GTK zpřístupňujícím do tohoto jazyka prvky grafického uživatelského rozhraní. Editor je psán jako „desktopová aplikace“, je tedy určen ke spouštění na uživatelském počítači. Ke svému běhu vyžaduje přítomnost GTK+ knihoven ve verzi alespoň 2 a PHP-GTK frameworku.

Z funkcí, která má správný editor zdrojového kódu mít, jsem implementoval pouze základní Nový/Otevřít/Uložit. Dále jsem se soustředil jen na funkci analýzy a obarvení editovaného textu.

Analýza se spouští automaticky při každé změně editovaného textu, ale lze ji spouštět také ručně (tlačítkem Spustit). V případě ručního spuštění se v pří výskytu chyby zobrazí dialogové okno s oznámením o chybě, případně s očekávaným opravením této chyby. Analýza se vždy (i při automatickém spuštění) provádí na celém zdrojovém kódu znovu. Tím je způsobeno jisté omezení velikosti editovatelného souboru (do velikosti přibližně 10kB lze ještě editor použít).

Asi nejzásadnějším omezením celého editoru je nemožnost zotavení se z chyby – jakákoli chyba (v lexikální či syntaktické analýze) okamžitě znamená přerušení celé analýzy. Další omezení již plynou z (ne)kompletnosti konfiguračního souboru. V testovacím konfiguračním souboru jsou zanesena pouze pravidla potřebná k analýze vzorového zdrojového kódu (viz Seznam 2).

Lexikální analýza

Lexikální analýzu zajišťuje samostatná třída LexAn implementující interface ILexAn. Lexikální analýza zpracovává vstupní data znak po znaku a podle pravidel získaných z XML dokumentu je skládá do vyšších celků – lexikálních tokenů. Ty jsou následně předány třídě, která se stará o grafické rozlišení jednotlivých tokenů v textu (obarvení), a v podobě terminálních symbolů třídě obstarávající syntaktickou analýzu (tokeny rozpoznatelné syntaktickým analyzátorem viz Tabulka 1). Pokud zjistí lexikální analyzátor ve vstupních datech situaci odporující daným pravidlům, ukončí celou analýzu chybou s informací kde a k jaké chybě došlo.

Syntaktická analýza

Syntaktickou analýzu zajišťuje samostatná třída SyntaxAn implementující interface ISyntaxAn. Syntaktická analýza je realizována zásobníkovým automatem se dvěma zásobníky – jeden pro vstupní terminální symboly (tokeny získané z lexikálního analyzátoru) a druhý pro terminální i neterminální symboly (získané ze syntaktických pravidel načtených z konfiguračního souboru). Stejně jako lexikální analyzátor i syntaktický analyzátor reaguje na neřešitelnou situaci vstupních terminálních symbolů informativní chybou. Pokud jsou celá vstupní data zpracována korektně, nic se nestane. Zde je k dispozici prostor pro generování syntaktických stromů pro zpracování nápovědy při psaní, vyhodnocování výrazů atp. Používané neterminální symboly viz Tabulka 2, používaná syntaktická pravidla viz Seznam 1.

Konfigurace analyzátorů

Tabulky stavů, vstupy i výstupy pro konečné automaty obou analyzátorů jsou uloženy v definičním XML dokumentu pro konkrétní jazyk. Z tohoto jsou při inicializaci aplikace všechna nastavení načtena do paměti, odkud jsou následně používána při každé analýze.

Konfigurační XML dokument obsahuje nastavení pro:

Lexikální analyzátor – obsahuje definici množin znaků a tabulku stavů, přechodů a výstupů pro konečný automat lexikálního analyzátoru.

Obarvovací systém – obsahuje informace pro všechny elementy lexikální analýzy, které se mají obarvit.

Syntaktický analyzátor – obsahuje definici startovního symbolu, epsilon symbolu, dále všech symbolů, které se v syntaktické analýze vyskytují a symbolů, které má syntaktická analýza přeskočit (například komentáře). Na závěr obsahuje definici syntaktických pravidel, podle kterých se vyhodnocuje zdrojový kód.

Další nastavení – konfigurační dokument také obsahuje informace o známých klíčových slovech a konstantách, které se vyhledávají v seznamu lexikálních elementů mezi lexikální a syntaktickou analýzou. (Lexikální analýza obojí rozlišuje jen jako neuvozený řetězec, ale pro syntaktickou analýzu je znalost hlavně klíčových slov nezbytná.)

Strukturu konfiguračního XML dokumenty popisuje soubor ve formátu XML Schema, který je součástí aplikace.

Instalace

Instalace v OS Windows je možná v zásadě dvěma způsoby. Pokud chceme používat pouze PHPEditor a na nic jiného nevyužijeme ani knihovny GTK+, ani PHP, je nejjednodušší stáhnout archiv all-in-one[14], rozbalit na libovolné místě a spustit souborem `PHPEditor.cmd`. Tento archiv obsahuje běhové knihovny GTK+, PHP s rozšířením PHP-GTK i samotnou aplikaci PHPEditor.

Druhý způsob je vhodný v případě, že některou z komponent uvedenou v minulém odstavci využíváme i pro jiný program (například GIMP, Inkscape, či jiná aplikace, psané například v PHP-GTK frameworku). V tomto případě je lepší postupně nainstalovat běhové knihovny GTK +[4][5][6] (například s programem GIMP jsou již nainstalované) a ručně si nastavit PHP[8] s PHP-GTK pluginem (zjednodušená instalace: viz Příloha A). Pro správný běh PHPEditoru jsou třeba knihovny GTK+ verze 2 (doporučuji alespoň verzi 2.10), PHP verze 5 a plugin PHP-GTK verze 2.

V OS Linux je asi jedinou volnou druhý zmíněný postup, tedy nainstalovat (případně zkompilovat) postupně všechny zmíněné části. Pro zjednodušení lze využít například tento[11] instalátor.

Závěr

Realizace lexikální a syntaktické analýzy v jazyce PHP přináší klady i zápory. Za klady lze považovat například existenci asociativních polí (které bez nutnosti deklarace zastupují struktury známe z STL jazyka C++ – například mapa, množina, aj.) či jednoduchého rozhraní DOM pro práci s XML dokumenty. Mezi zápory se řadí hlavně rychlost provádění aplikace zpomalovaná faktem, že se jedná o interpretovaný jazyk s velkým množstvím interních kontrol používaných struktur (viz například zmíněné asociativní pole).

Tabulka 1 – tokeny rozpoznatelné syntaktickým analyzátorem (terminální symboly)

token	význam	token	význam
-	-	T_CONSTANT_ENCAPSED_STRING	*uvozený řetězec*
!	!	T_DEC	--
%	%	T_DIE	die
(	(	T_DOUBLE_ARROW	=>
)	)	T_ECHO	echo
*	*	T_ELSE	else
,	,	T_EPSILON	*prázdný vstup*
.	.	T_FOR	for
/	/	T_FOREACH	foreach
;	;	T_FUNCTION	function
[	[	T_IF	if
]	]	T_INC	++
{	{	T_INCLUDE	include
}	}	T_IS_EQUAL	
+	+	T_ISSET	isset
=	=	T_LNUMBER	*číslo*
T_ARRAY	array	T_NULL	null
T_AS	as	T_OPEN_TAG	<?php
T_BOOLEAN_AND	&&	T_PLUS_EQUAL	+=
T_BOOLEAN_OR	||	T_REQUIRE	require
T_CLOSE_TAG	?>	T_STRING	*řetezec znaků*
T_CONCAT_EQUAL	.=	T_UNSET	unset
T_CONST_STRING	*interní konstanta*	T_VARIABLE	*proměnná*

Většina tokenů byla převzata z oficiální stránky projektu PHP[16].

Tabulka 2 – používané neterminální symboly

S_BLOCK	S_FOREACH_KEY	S_PRIKAZ	S_VYRAZ_LIST
S_BLOCK_INNER	S_FUNC	S_PRIKAZ_PRIKAZ	S_VYRAZ_LIST_END
S_CODE	S_IF	S_PRIKAZ_PRIKAZ_PARAM	S_VYRAZ_LIST_INNER
S_CODE_INNER	S_KONST	S_PRIKAZ_SINGLE	S_VYRAZ_T
S_ELSE	S_PARAM	S_VYRAZ	S_VYRAZ_TT
S_FOR	S_PARAM_END	S_VYRAZ_F	S_VYRAZ_TT_ARRAY
S_FOREACH	S_PARAM_NEXT	S_VYRAZ_FF	

Seznam 1 – používaná syntaktická pravidla

```
S_BLOCK -> { S_BLOCK_INNER }
S_BLOCK_INNER -> S_PRIKAZ S_BLOCK_INNER
S_BLOCK_INNER -> ε
S_CODE -> <?php S_CODE_INNER ?>
S_CODE_INNER -> S_PRIKAZ S_CODE_INNER
S_CODE_INNER -> ε
S_ELSE -> else S_PRIKAZ
S_ELSE -> ε
S_FOR -> for ( S_VYRAZ ; S_VYRAZ ; S_VYRAZ ) S_PRIKAZ
S_FOREACH -> foreach ( $promenna as $promenna S_FOREACH_KEY ) S_PRIKAZ
S_FOREACH_KEY -> => $promenna
S_FOREACH_KEY -> ε
S_FUNC -> function identifikator ( S_PARAM ) S_BLOCK
S_IF -> if ( S_VYRAZ ) S_PRIKAZ S_ELSE
S_KONST -> konstanta
S_KONST -> retezec
S_KONST -> cislo
S_KONST -> null
S_PARAM -> ε
```

```
S_PARAM -> $promenna S_PARAM_END S_PARAM_NEXT
S_PARAM_END -> = S_KONST
S_PARAM_END -> ε
S_PARAM_NEXT -> , $promenna S_PARAM_END S_PARAM_NEXT
S_PARAM_NEXT -> ε
S_PRIKAZ -> S_BLOCK
S_PRIKAZ -> S_FUNC
S_PRIKAZ -> S_PRIKAZ_SINGLE
S_PRIKAZ_PRIKAZ -> die S_PRIKAZ_PRIKAZ_PARAM ;
S_PRIKAZ_PRIKAZ -> echo S_PRIKAZ_PRIKAZ_PARAM ;
S_PRIKAZ_PRIKAZ -> include S_PRIKAZ_PRIKAZ_PARAM ;
S_PRIKAZ_PRIKAZ -> require S_PRIKAZ_PRIKAZ_PARAM ;
S_PRIKAZ_PRIKAZ -> unset S_PRIKAZ_PRIKAZ_PARAM ;
S_PRIKAZ_PRIKAZ_PARAM -> S_VYRAZ
S_PRIKAZ_SINGLE -> S_FOR
S_PRIKAZ_SINGLE -> S_FOREACH
S_PRIKAZ_SINGLE -> S_IF
S_PRIKAZ_SINGLE -> S_PRIKAZ_PRIKAZ
S_PRIKAZ_SINGLE -> S_VYRAZ ;
S_VYRAZ -> S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_F -> ( S_VYRAZ )
S_VYRAZ_F -> S_KONST
S_VYRAZ_F -> array S_VYRAZ_LIST
S_VYRAZ_F -> isset S_VYRAZ_LIST
S_VYRAZ_F -> identifikator S_VYRAZ_LIST
S_VYRAZ_F -> $promenna S_VYRAZ_FF
S_VYRAZ_FF -> = S_VYRAZ
S_VYRAZ_FF -> .= S_VYRAZ
S_VYRAZ_FF -> ε
S_VYRAZ_FF -> += S_VYRAZ
S_VYRAZ_LIST -> ( S_VYRAZ_LIST_INNER )
S_VYRAZ_LIST_END -> , S_VYRAZ S_VYRAZ_LIST_END
S_VYRAZ_LIST_END -> ε
S_VYRAZ_LIST_INNER -> S_VYRAZ S_VYRAZ_LIST_END
S_VYRAZ_LIST_INNER -> ε
S_VYRAZ_T -> - S_VYRAZ_F
S_VYRAZ_T -> ! S_VYRAZ_F
S_VYRAZ_T -> S_VYRAZ_F
S_VYRAZ_T -> -- S_VYRAZ_F
S_VYRAZ_T -> ++ S_VYRAZ_F
S_VYRAZ_TT -> - S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT -> . S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT -> [ S_VYRAZ ] S_VYRAZ_TT_ARRAY S_VYRAZ_TT
S_VYRAZ_TT -> * S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT -> / S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT -> % S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT -> + S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT -> && S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT -> || S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT -> ε
S_VYRAZ_TT -> == S_VYRAZ_T S_VYRAZ_TT
S_VYRAZ_TT_ARRAY -> = S_VYRAZ
S_VYRAZ_TT_ARRAY -> ε
```

Kolize FOLLOW-FIRST v této gramatice jsou řešeny pomocí priority pravidel. Konkrétně se jedná o pravidla pro strukturu `IF – ELSE` a o případ, kdy výrazu s binárním operátorem `[]` přiřazujeme hodnotu (přiřazení do pole – jediná možnost přiřazení, kdy je na levé straně výraz). Priority jsou řešeny podle PHP dokumentace[17].

Seznam 2 – podporované konstrukce

- PHP kód začínající `<?php` a končící `?>`, který může obsahovat:
  - blok příkazů uvozený `{}`
  - komentáře (řádkové i blokové)
  - definice funkcí
  - struktury `for`, `foreach`, `if`
  - příkazy `die`, `echo`, `include`, `require`, `unset`
  - výrazy, které mohou obsahovat:
    - závorky `()`
    - volání funkce
    - proměnné
    - konstanty
    - definice polí
    - operátor `isset`
    - přiřazení (=, `.=`, +=)
    - unární operátory (`-`, `!`, `++`, `--`)
    - binární operátory (`-`, `%`, `*`, `.`, `/`, `[]`, `+`, `&&`, `||`, ==)

Příloha A – postup instalace PHP-GTK frameworku

0. tento postup předpokládá již nainstalované běhové knihovny GTK+
1. ze stránek projektu[14] si stáhněte archiv `php-gtk-frmw.zip`
2. rozbalte tento archiv do složky `C:\Program Files\`
3. přidejte do registrů soubor `C:\Program Files\PHP GTK\php-gtk.reg`
4. nyní již můžete stáhnout PHPEditor (`php-editor.zip`), kamkoliv jej rozbalit a program spustit dvojklikem na soubor `PHPEditor.phpg`

Zdroje

(všechny zdroje byly prohlíženy v noci z 12. na 13. února 2008)

[1] http://www.gtk.org/ – domovská stránka projektu GTK+

[2] http://www.gtk.org/commerce.html – příklady využití GTK+

[3] http://library.gnome.org/devel/gtk/stable/ – dokumentace ke knihovnám GTK+

[4] http://gimp-win.sourceforge.net/old.html – ke stažení zkompilované knihovny GTK+

[5] http://sourceforge.net/projects/gimp-win/ – ke stažení zkompilované knihovny GTK+

[6] http://gladewin32.sourceforge.net/modules/wfdownloads/viewcat.php?cid=15 – ke stažení zkompilované knihovny GTK+

[7] http://www.php.net/ – domovská stránka projektu PHP

[8] http://www.php.net/downloads.php – ke stažení PHP

[9] http://www.php.net/manual/en/history.php – souhrn historie projektu PHP

[10] http://gtk.php.net/ – domovská stránka projektu PHP-GTK – plugin do PHP

[11] http://oops.opsat.net/doc/gtk/unix-installer.html – Unix PHP-GTK instalátor

[12] http://php-gtk.eu/ – komunitní stránka projektu PHP-GTK – plugin do PHP

[13] http://vpp-net.com/ – domovská stránka autora této práce

[14] http://vpp-net.com/fel/pjp/phpeditor/ – domovská stránka PHPEditoru

[15] http://vpp-net.com/fel/api/mmfm/ – jiný projekt psaný ve frameworku PHP-GTK

[16] http://www.php.net/manual/en/tokens.php – seznam lexikálních tokenů jazyka PHP

[17] http://www.php.net/manual/en/language.operators.php – operátory jazyka PHP