Заметка simple test lexer

Просто привожу быстрый пример, пока экспериментирую с Simple Tests Lexer. Если вам нужен инструмент для разбора какого-нибудь мини-языка в PHP , это хорошее место для поиска.

Одним из побочных продуктов Simple Test является лексер на основе регулярных выражений, который можно найти в CVS здесь. Маркус использует его для построения HTML- парсера для Simple Test, но его можно применять практически к чему угодно. Обратите внимание, что есть отдельная версия лексера, доступная на Marcus SourceForge «code dump» - http://sf.net/projects/lamplib .

Сейчас я вынужден исследовать его из-за необходимости. WACT разрабатывает шаблонное выражение/язык привязки данных, использование которого описано в Руководстве для авторов шаблонов. К сожалению, есть ошибка (эффекты описаны здесь), которая, похоже, предполагает, что синтаксис PCRE в PHP изменился где-то между PHP 4.1.2 и 4.3.2.

Рассматриваемое регулярное выражение:

```
$regex = '/^((?Us).*)'. preg_quote('{$', '/') .
            '(([^"\'}]+|(\'|")(?U).*\4)+)' .
            preg_quote('}', '/') . '((?s).*)$/';
```

Я не нашел никаких очевидных указаний на изменения в синтаксисе в руководстве PHP / CVS (любой вклад будет очень признателен). Это регулярное выражение также заставляет меня ломать голову (Джефф тоже ломает голову над этим), поэтому изучаю возможности более управляемого подхода к разбору языка выражений шаблонов.

В любом случае, если говорить коротко, вот очень простой пример использования Simple Tests lexer в качестве шаблонизатора, который может помочь кому-то начать. Дополнительные примеры лучше всего искать, изучая тестовые случаи парсера.

Если у меня есть такой шаблон;

```
Начальное сообщение — {$Greeting}<br>
Последнее слово по теме — {$Closing}
```

где `{$Greeting}` и `{$Closing}` — ссылки на переменные шаблона, которые я хочу заменить некоторыми значениями, назначенными мной шаблонизатору, синтаксический анализатор, использующий лексический анализатор Simple Tests, может выглядеть так:

```
<?php
// Включить парсер
require_once  'path/to/simpletest/parser.php' ;

класс YetAnotherTemplateParser {

    // Здесь размещается вывод шаблона
    var  $output = '' ;

    // Хэш переменных для замены
```

```
    var  $phpVars = array ( ) ;

    // Регистрация переменной
    function registerVariable ( $name , $value )  {
        $this -> phpVars [ $name ] = $value ;
     }

    // Отображение страницы
    function display ( )  {
        echo  $this -> output ;
     }

    /**
* Функция обратного вызова (или режим/состояние), вызываемая Lexer. Эта
* имеет дело с текстом за пределами ссылки на переменную.
    * @param string совпадающий текст
    * @param int состояние лексера (здесь игнорируется)
    */
    function writePlainText ( $match , $state )  {
        $this -> output .= $match ;
         return  TRUE ;
     }

    /**
* Обратный вызов для ссылок на переменные шаблона.
    * @param string совпадающий текст
    * @param int состояние лексера
    */
    function writeVariable ( $match , $state )  {
        switch  ( $state )  {
            // Ввод ссылки на переменную
            case LEXER_ENTER:
                 // Начало ссылки на переменную — пока ничего не нужно делать
            break ;

            // Содержимое переменной reference
            case LEXER_UNMATCHED:
                 if  (  isset ( $this -> phpVars [ $match ] )  )  {
                    $this -> output .= $this -> phpVars [ $match ] ;
                 }
            break ;

            // Выход из ссылки на переменную
            case LEXER_EXIT:
                 // Конец ссылки на переменную - ничего не нужно делать - завершено
            break ;
         }
        return  TRUE ;
     }
```

```
}

// Создаем парсер шаблона
$Parser = & new YetAnotherTemplateParser ( ) ;

// Регистрируем некоторые переменные шаблона для замены
$Parser -> registerVariable ( 'Greeting' , 'Hello World!' ) ;
 $Parser -> registerVariable ( 'Closing' , 'Goodbye World!' ) ;

//Создаем лексический анализатор
// Второй аргумент: начальное "состояние" или функция обратного вызова
// Третий аргумент: чувствительность к регистру ВКЛ - не имеет значения для этого примера,
// так как шаблоны регулярных выражений не являются буквами, но, чтобы вы знали,
$Lexer = &new SimpleLexer ( $Parser , 'writePlainText' , TRUE ) ;

// Добавить ссылку на переменную, начинающую шаблон регулярного выражения
// Второй аргумент: текущее состояние, к которому применяется этот шаблон -
предотвращает
// синтаксис шаблона, подобный {$my{$var, вызывающий два
// перехода
// Третий аргумент: состояние (функция обратного вызова) для отправки дальнейших вызовов
// после того, как шаблон будет найден
$Lexer -> addEntryPattern ( ' \{ \$ ' , 'writePlainText' , 'writeVariable' )
;

// Добавляем шаблон выхода для ссылок на переменные, возвращая лексер в
// его предыдущее состояние (использует стек состояний)
// Второй аргумент: состояние, в котором применяется этот шаблон
$Lexer -> addExitPattern ( ' \} ' , 'writeVariable' ) ;

$template = 'Вступительное сообщение — {$Greeting}<br>
  Последнее слово по теме — {$Closing}' ;

$Lexer -> разбор ( $template ) ;

$Parser -> дисплей ( ) ;
 ?>
```

Надеюсь, комментарии объяснят, что происходит.

Конечно, это очень нетребовательный язык, но, судя по опыту, Simple Tests Lexer может прекрасно масштабироваться до довольно сложного языка (HTML, для которого Маркус его использует, не так уж и прост в анализе).

Примечание: это не для того, чтобы побудить вас писать еще больше шаблонизаторов! Их более чем достаточно. Например, разбор CSS , Javascript, VBScript (подталкивание подталкивание) или SQL может быть стоящей миссией или для мини-языков, таких как язык выражений шаблонов WACT.

Дополнения и Файлы

Ссылка на оригинальную статью