


```
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! 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.

From:
<https://wwoss.direct.quickconnect.to/> - **worldwide open-source software**

Permanent link:
https://wwoss.direct.quickconnect.to/doku.php?id=wiki:devel:parser:test:simple_test_lexer_notes&rev=1737037104

Last update: **2025/01/16 17:18**

