WebDAO - платформа для создания web приложений

Разработка веб приложений в WebDAO

Александр Загацкий

WebDAO - платформа для создания web приложений : Разработка веб приложений в WebDAO

Александр Загацкий

Аннотация

Документация по использованию WebDAO в разработке web приложений. Включает описание процесса установки и настройки web окружения, примеры кодов и описание модулей.

Содержание

Список таблиц

Об этой книге

Книга о платформе WebDAO.

Александр Загацкий

Глава 1. WebDAO - платформа для web приложений

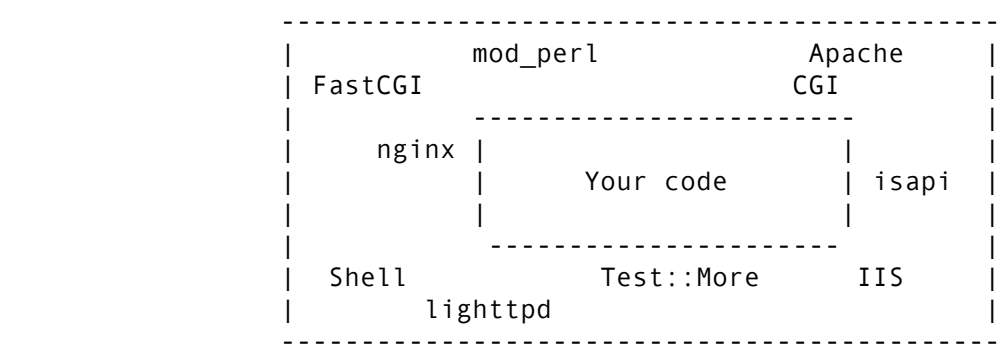
WebDAO - объектно-ориентированная система для создания производительных web приложений, состоящая из набора модулей на языке *perl*.

Основные принципы

- * Абстракция среды выполнения кода приложения
- * Динамическая структура доменной логики
- * Адресация объектов по URL
- * Встроенная поддержка сессионных параметров

Абстракция среды выполнения кода приложения

Существует большое количество окружений, в которых работают web приложений.



WebDAO проектировалась с целью избавить разработчика от деталей среды работы приложения, снизить затраты при смене окружения, а также упростить отладку и тестирование приложений. Важной целью является упрощение и повышение скорости web разработки.

Динамическая структура доменной логики

Структура объектов доменной логики формируется на основе *XML*(*HTML*) файла. Имя файла может быть производным. Для примера, его имя пусть будет привычным для web разработчиков : *index.xhtml*.

```
<body>
<div>
 <wd>
   <object class="MyTest" id="page"/>
 </wd>
</div>
</body>
```

В этом тексте, кроме XHML тэгов, используются дополнительные, например: *wd* и *object*. Тэг *wd* является признаком начала специальной (*интерпретируемой*) области. Пока отсутствует поддержка пространств XML имен, но со временем, я обещаю, она появится.

Тэги *wd* обрамляют область, в которой располагаются определения объектов и другие интерпретируемые тэги. В приведенном примере, с помощью *command* создается экземпляр класса *MyTest* и идентификатором *page*. Именно идентификатор объекта используется в URL. Например:

```
http://example.org/page
http://example.org/page/Method?par1=1&par2=2
```

В состав пакета *WebDAO* входит лексический анализатор, который обрабатывает этот файл и на основе *xml* тэгов строит структуру объектов.

Создадим файл `MyTest.pm` со следующим содержимым:

```
package MyTest;
use WebDAO;
use base 'WebDAO::Component';

sub fetch {
    "Hello Web X.0!";
}
1;
```

Каждый из объектов доменной структуры, участвующих в формировании результатов, отображает себя сам. Поэтому, в приведенном примере, результирующий XHTML будет выглядеть следующим образом:

```
<body>
<div>
   Hello Web X.0!
</div>
</body>
```

Адресация объектов по URL

Одна из основных идей *WebDAO* - отображение URL на доменную структуру объектов.

Например, для адреса:

```
http:://example.com/test/Method?param=1&param2
```

Будет вызван метод *Method* у объекта *test*. Имена публичных методов, которые доступны для обращений из-вне начинаются с большой буквы. Имена объектов с любой. В случае, если имя метода не указано используется *index_x*. Если этот метод отсутствует у объекта, возвращается статус "404: Not found". Таким образом адреса:

```
http:://example.com/test/?param=1&param2
http:://example.com/test/
```

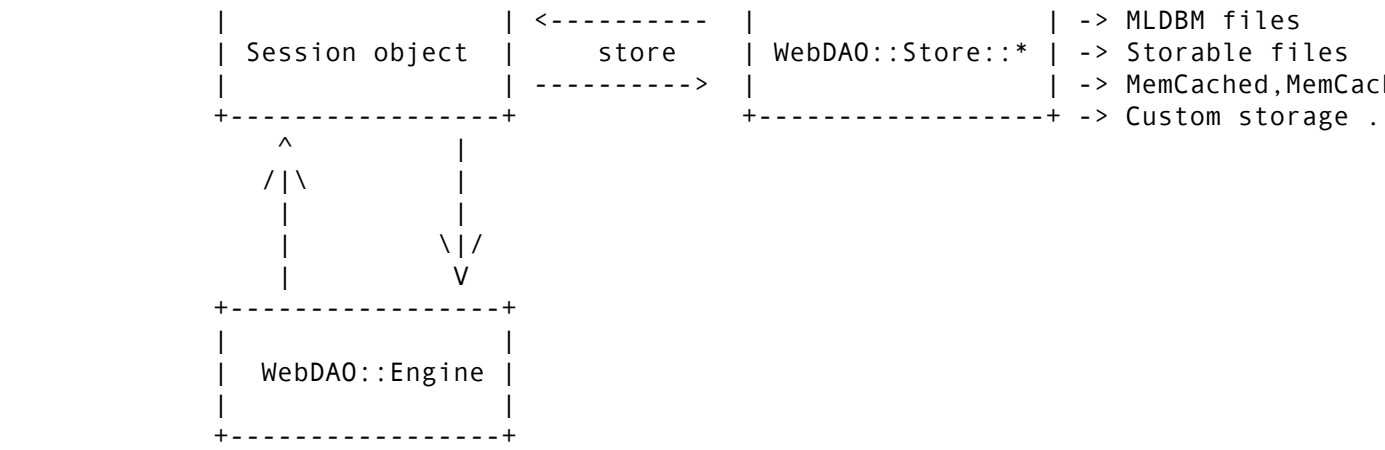
Эквивалентны вызовам

```
http:://example.com/test/index_x?param=1&param2
http:://example.com/test/index_x
```

Встроенная поддержка сессионных параметров

В *WebDAO* встроена поддержка сессионных параметров. Схематически ее можно представить следующей диаграммой объектов:

```
      +-----------------+       load     +------------------+   Storages:
```



Для этого достаточно выбрать источник хранения в конфигурации web сервера и указать атрибуты объекта в наследуемом классе.

Пример конфигурации (*Apache* web server):

```
<VirtualHost *>
    ...
    #set Storable storage
    SetEnv wdStore WebDAO::Store::Storable
    #path for store
    SetEnv wdStorePar path=/tmp/sessions

    #Track session via cookies
    SetEnv wdSession WebDAO::Sessionco
    ...
</VirtualHost>
```

В тексте модуля также необходимо создать атрибуты:

```
package MySess;
use WebDAO::Component;
use base 'WebDAO::Component';

# Определение  сессионных атрибутов и
# их значений по умалчанию

__PACKAGE__->mk_sess_attr( attr1 => undef, _attr2 => undef );

sub UseAttr {
    my $self = shift;
    # read
    my $val = $self->attr1;
    ...
    $self->attr1('test_value');
}
```

Глава 2. Установка Webdao

Если есть уже установленный модуль CPAN, то достаточно выполнить следующую каманду:

```
% perl -MCPAN -e 'install WebDAO'
```

Иначе, потребуется выполнить следующие шаги:

- * скачать последнюю версию CPAN по адресу:

```
http://search.cpan.org/dist/WebDAO/
```

- * распаковать и установить

Для этого потребуется выполнить следующие команды (или их эквиваленты для конектретной системы):

```
tar xzf WebDAO-*.tar.gz
cd WebDAO-*
perl Makefile.PL
make test && sudo make install
```

Глава 3. Конфигурирование Webdao

Конфигурирование производится с помощью переменных окружения. Предопределены следующие переменные:

Таблица 3.1. Конфигурационные параметры

Имя	описание	пример
wdIndexFile	" index.html - наименование файла, который будет обрабатываться при поступлении запросов. В качестве значения возможны: полный путь, или путь относительно `DOCUMENT_ROOT`. По умолчанию: `$ENV{DOCUMENT_ROOT}/index.xhtml` "	index.html
wdEngine	наименование пакета основного модуля. Этот модуль обслуживает все запросы поступающие к */*. По умолчанию: *WebDAO::Engine*	ShowPrice
wdEnginePar	" параметры инициализации при создании основного модуля. Значение - строка, содержащая пары *ключ=значние*. Пары отделяются друг от друга *;*. По умолчанию: *undef* "	config=/home/zag/showprice.ini
wdSession	" имя пакета модуля, обслуживающего сессионность. Этот модуль используется для идентификации web сессии. По умолчанию: *WebDAO::Session* "	WebDAO::Sessionco
wdStore	" имя модуля, обеспечивающего хранение сессионных параметров пользователя. По умолчанию: *WebDAO::Store::Abstract*"	WebDAO::Store::MLDBM
wdStorePar	параметры инициализации для модуля хранения сессионных параметров. По умолчанию: *undef*	path=/home/zag/Work/Recomendator/bin/tmp
wdFCGIreq	количество запросов на *FastCGI* процесс. Параметр используется при работе в режиме *FastCGi*. По умолчанию: -1 - unlimited	1000

Для сервера lighttpd используются имена соответственно: WD_INDEXFILE, WD_ENGINE, WD_ENGINE_PAR, WD_SESSION, WD_STORE, WD_STORE_PAR, WD_DEBUG.

Конфигурирование Web сервера

Поддерживаются все распространенные Web сервера: IIS (isapi_fcgi.dll), nginx, lighttpd, apache.

WebDAO поддерживает режимы *cgi*, *FastCGi*, *mod_perl*. Наиболее производительным является режим *FastCGI*.

В описании используются следующие исходные условия.

Таблица 3.2. Исходные данные

Наименование параметра	Значение
Web root	/usr/zag/www
Временный каталог	/tmp
Каталог для записи логов	/var/log/
Наименование домена	example.org
Расположение файлового сокета для FastCGI сервера	/tmp/myapp.socket

Настройка Standalone сервера FastCGI

В состав пакета *WebDAO* входит скрипт `wd_fcgi.fpl` (`/usr/local/bin/wd_fcgi.fpl` for example). Для запуска самостоятельного сервера используется следующая строка :

```
#!/bin/sh
/usr/local/bin/wd_fcgi.fpl -d -l /tmp/myapp.socket -n 5 -maxreq 1000
```

Для получения справки используется параметр *--help*:

```
/usr/local/bin/wd_fcgi.fpl --help
```

Вывод:

```
Usage:
wd_fcgi.fpl [options]

    -d -daemon     Daemonize the server.
    -p -pidfile    Write a pidfile with the pid of the process manager.
    -l -listen     Listen on a socket path, hostname:port, or :port.
    -n -nproc      The number of processes started to handle requests.
    -m -maxreq     Number of request before process will be restarted
                   -1 - unlimited. (defailt: -1)
```

nginx (standalone FastCGI)

- Простой пример

```
server {
    listen       80;
    server_name  example.org;

    charset utf-8;

    access_log  /var/log/nginx/example.org-access.log ;
```

```
 error_log  /var/log/nginx/example.org-error.log  debug;
        root   /home/zag/www/;
        location ~ / {
            include        fastcgi_params;
            fastcgi_pass   unix:/tmp/webdao.sock;
            fastcgi_param wdSession WebDAO::Sessionco;
            fastcgi_param wdIndexFile index.xhtm;
        }

    }
```

- Пример с использованием собственного пакета основного модуля

Для примера пусть используется в качестве имени модуля: *MySite*. Конструктор этого класса в качестве параметра принимает *config* - путь к конфигурационному файлу.

```
    server {
        listen       80;
        server_name  example.org;

        charset utf-8;

        access_log  /var/log/nginx/example.org-access.log;
 error_log  /var/log/nginx/example.org-error.log  debug;
        root   /home/zag/www/;
        #sample for static data
        #location ~* ^/(js|imag|img|data|data2|css|static|images)/ {
        #}
        location ~ / {
            include        fastcgi_params;
            fastcgi_pass   unix:/tmp/webdao.sock;
            fastcgi_param wdSession WebDAO::Sessionco;
            fastcgi_param wdIndexFile index.xhtm;
         fastcgi_param wdEngine MySite;
     fastcgi_param wdEnginePar config=/home/zag/www/mysite.ini;
        }

    }
```

apache (static + standalone FastCGI)

Возможны два режима. Когда используется собственный менеджер *FCGI*(*FastCgiServer*) и кода подключение производится через FCGI сокет.

Требуется установка модуля *mod_fastcgi*:

```
        mod_fastcgi-2.4.2
```

В глобальной части `httpd.conf` требуется добавить одну из необходимых секций:

- 1 Static (FastCgiServer)

```
     <LoadModule fastcgi_module /usr/lib/apache2/modules/mod_fastcgi.so
        <IfModule mod_fastcgi.c>
        AddHandler fastcgi-script fpl fcgi
        FastCgiServer /usr/local/bin/wd_fcgi.fpl \
            -idle-timeout 3000 -flush -restart-delay 5 \
            -initial-env wdFCGIreq=1000 -processes 4 \
    </IfModule>
```

- 2 Standalone (FastCgiExternalServer)

```
 <LoadModule fastcgi_module /usr/lib/apache2/modules/mod_fastcgi.so
  <IfModule mod_fastcgi.c>
  # Connect via net socket
  # FastCgiExternalServer /usr/local/bin/wd_fcgi.fpl -host localhost:60000
   FastCgiExternalServer /usr/local/bin/wd_fcgi.fpl -socket /tmp/myapp.sock
  </IfModule>
```

Настройки секции VirtualHost:

```
<VirtualHost>
   DocumentRoot /usr/zag/www
   ServerName example.org
   ErrorLog /var/log/example.org-error_log
   CustomLog /var/log/example.org-access_log common
   SetEnv wdEngine WedDAO::Kern
   SetEnv wdIndexFile index.xhtml
   SetEnv wdSession WebDAO::Sessionco

   #for use external storage
   #SetEnv wdStore WebDAO::Store::MLDBM
   #SetEnv wdStorePar path=/tmp

   RewriteEngine on
   AddDefaultCharset UTF-8
   RewriteCond     %{HTTP:Authorization}   ^(.*)$ [NC]
   RewriteRule     /.*             -       [E=HTTP_AUTHORIZATION:%1]
   <IfModule mod_fastcgi.c>
       RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-f
       RewriteRule ^/(.*) /usr/local/bin/wd_fcgi.fpl?$1 [QSA]
   </IfModule>
</VirtualHost>
```

lighttpd (standalone FastCGI)

```
   var.engine = "ZagSite"
   var.defaults = (
       "WD_SESSION"=>"WebDAO::Sessionco",
       "WD_INDEXFILE"=>"index.xhtm"
   )
   $HTTP["host"] == "example.org" {
   server.document-root    = "/home/zag/www/"
   setenv.add-environment =  var.defaults
   }
   #use custom root class - MySite
   $HTTP["host"] == "example.com" {
   server.document-root    = "/home/zag/www/"
   setenv.add-environment =  var.defaults + (
       "WD_ENGINE" => "MySite",
       "WD_ENGINE_PAR"=>"config=/home/zag/www/mysite.ini"
       )
   }
   #skip static
   $HTTP["url"] !~ "^/(js|imag|img|css|static)" {
      fastcgi.server = (
       "" => (
           "" => (
               "socket"      => "/tmp/webdao.sock",
```

```
                "check-local" => "disable"
                    )
                )
            )

    }
```

Настройка cgi сервера

Для работы *WebDAO* как *CGI* приложения используется скрипт *wd_cgi.pl*.

apache (CGI)

```
<VirtualHost *>
   DocumentRoot /usr/zag/www
   ServerName example.org
   ErrorLog /var/log/example.org-error_log
   CustomLog /var/log/example.org-access_log common

   SetEnv wdIndexFile index.xhtm
   SetEnv wdEngine WebDAO::Engine
   SetEnv wdSession WebDAO::Sessionco

   RewriteEngine on
   RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-f
   RewriteRule ^/(.*) /usr/local/bin/wd_cgi.pl?$1 [QSA]
   <Directory "/usr/local/bin/wd_cgi.pl">
     AddHandler cgi-script cgi pl
     Options Indexes FollowSymLinks ExecCGI
   </Directory>
</VirtualHost>
```

Запуск из ***shell***

Для запуска из командной строки используется скрипт *wd_shell.pl*. В процессе выполнения используются переменные окружения.

```
Usage:
    wd_shell.pl [options] file.pl

     options:

      -help  - print help message
      -man   - print man page
      -f file    - set root [x]html file

Options:

  -help   Print a brief help message and exits
  -man    Prints manual page and exits
  -f filename
          Set filename set root [x]html file for load domain
```

Для написания тестов используется `WebDAO::Test`.