Develop web application in WebDAO

The WebDAO Developers Guide

Aliaksandr Zahatski

Develop web application in WebDAO : The WebDAO Developers Guide

Aliaksandr Zahatski

Аннотация

A guide to develop web application using WebDAO. Includes a overview of setup process, including sample code and reference material.

Содержание

Список таблиц

About

WebDAO - object-oriented system for easy creation of high-performance and scalable web applications writen in *perl*.

Aliaksabdr Zahatski

Глава 1. WebDAO - platform for web applications.

WebDAO - object-oriented system for easy creation of high-performance and scalable web applications writen in *perl*.

Key Features

- * Abstraction application code from the environment
- * Dynamic structure of the domain logic
- * Addressing objects by URL
- * Built-in support session parameters

Abstraction application code from the environment

There are many environments in which the web applications work:

```
-------------------------------------------
|           mod_perl             Apache     |
| FastCGI                       CGI         |
|           ------------------------        |
|     nginx |                       |       |
|           |     Your code         | isapi |
|           |                       |       |
|            ----------------------         |
|  Shell             Test::More      IIS    |
|        lighttpd                           |
-------------------------------------------
```

WebDAO designed to save developers from the details of the application environment, reduce costs with a change of environment, and to simplify debugging and testing applications. An important goal is to simplify and increase the speed of web development.

Dynamic structure of the domain logic

The structure of the domain logic is based on *XML*(*HTML*) file. The file name can be derived. For example, let his name be commonplace for web developers: *index.xhtml*.

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

In this text, except for XHTML tags are used more, for example: *wd* and *object*. Tag *wd* is a sign of the special (*interpreted*) field. While there is no support for the XML namespace, but over time, I promise, it will appear.

Tags *wd* frame area in which there are definitions of objects and other interpretable tags. In the above example, using the *command* creates a instance of the class *MyTest* and identifier: *page*. This object identifier is used in the URL. For example:

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

```
In the package I<WebDAO> include lexical analyzer, which processes the file and
```

Create a file `MyTest.pm` with the following content:

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

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

Each of the domain structures involved in the formation of the results, shows himself. Therefore, in this example, the resulting XHTML will look like this:

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

Addressing objects by URL

One of the main ideas *WebDAO* - resolve the URL to the domain structure of objects. For example, for URL:

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

Will be called the method *Method* at object *test*. The names of public methods that are available for applications from outside begin with a capital letter. The names of objects can be arbitrary.If the method is not specified - using the name *index_x*. If this method is not available at the object, returned to the status of "404: Not found". Thus the address below:

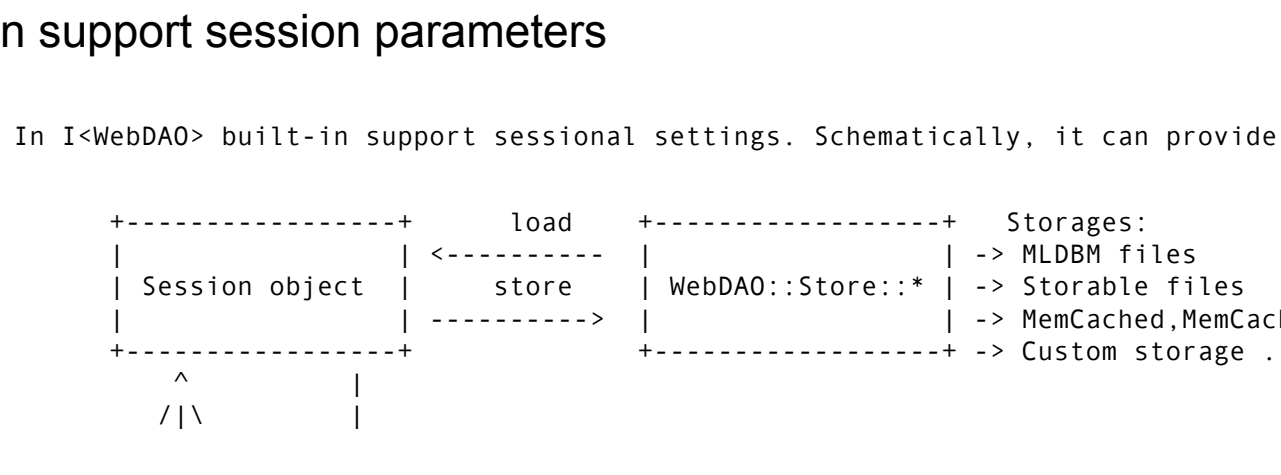
```
http:://example.com/test/?param=1&param2
http:://example.com/test/
```

Equivalent to the following:

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

Built-in support session parameters

```
In I<WebDAO> built-in support sessional settings. Schematically, it can provide
```



To do this, simply select the source storage in the configuration web server, a

Example configuration (*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>
```

The text of the module also need to create attributes:

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

# define list of session attributes

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

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

Глава 2. Install Webdao

You should be able to install this module using the CPAN shell interface:

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

Alternately, you may retrieve this package from CPAN.

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

After downloading the distribution, follow the normal procedure to unpack and install it, using the commands shown below or their local equivalents on your system:

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

Глава 3. Configuring Webdao

Configuration is done via environment variables. The following variables is using:

Таблица 3.1. Configuration parameters

name	description	sample
wdIndexFile	" index.html - name of the file to be processed upon request. Possible values : absolute path or relative to `DOCUMENT_ROOT`. Default: `$ENV{DOCUMENT_ROOT}/index.xhtml` "	index.html
wdEngine	" name of the package core module. This module serves all requests coming to */*. Default: *WebDAO::Engine* "	ShowPrice
wdEnginePar	" initialization parameters when creating the main module. Value a string containing the pairs *key=value*. Pairs are semicolon separated (*;*). Default: *undef* "	config=/home/zag/showprice.ini
wdSession	Name the package module, which serves a sessional. This module is used to identify a web session. Default: *WebDAO::Session*	WebDAO::Sessionco
wdStore	" the name of the module, providing storage of a user session parameters. Defalut: *WebDAO::Store::Abstract*"	WebDAO::Store::Storable
wdStorePar	initialization parameters for session storage module. default: *undef*	path=/home/zag/tmp

For server lighttpd uses names respectively: WD_INDEXFILE, WD_ENGINE, WD_ENGINE_PAR, WD_SESSION, WD_STORE, WD_STORE_PAR, WD_DEBUG.

Configuring Web server

It supports all popular Web servers: IIS (isapi_fcgi.dll), nginx, lighttpd, apache.

The *WebDAO* suuports: *cgi*, *FastCGi*, *mod_perl*. The most productive is the mode *FastCGI*.

In the examples used the following initial conditions.

Таблица 3.2. Baseline data

Parameter	Value
Web root	/usr/zag/www
Temporary directory	/tmp
Directory for logs	/var/log/
Domain name	example.org
Path for file socket for FastCGI	/tmp/myapp.socket

Setup standalone FastCGI

```
In the package I<WebDAO> includes a script C<wd_fcgi.fpl> ( C</usr/local/bin/wd
```

```
To run an independent server using the following command:
```

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

For help use *--help*:

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

Output:

```
    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)

- * Simple example

```
    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;
        }

    }
```

- * An example of using a custom package of basic module

For example, even if used as the name of the module: *MySite*. The constructor of this class as a parameter takes *config* - path to the configuration file.

```
    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)

There are two modes. When use own manager *FCGI* (*FastCgiServer*) and connection is made through FCGI socket.

Requires installation of the module *mod_fastcgi*:

```
        mod_fastcgi-2.4.2
```

As part of the global `httpd.conf` want to add one of the required sections:

- 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>
```

At VirtualHost section:

```
<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"
                   )
               )
           )

   }
```

Work in cgi mode

For work *WebDAO* as *CGI* application use the script *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>
```

Use from command line

To run from the command line using the script *wd_shell.pl*. In the process of being implemented using the environment variables.

```
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
```

See also: .