Speare Code Editor
The Small PHP IDE for PHP Development

Copyright (C) 2020 Sevenuc Consulting
Version 0.0.1
Update: 4 Mar 2020

Free, Lightweight, Open Source, Extendable Flexibility

Speare (http://www.sevenuc.com/en/speare.html) is an ultra lightweight code editor and a small IDE that has an efficient code navigation and call routines tracing ability, and has integrated debugging environment for C, C++, PHP and all kinds of PHP web frameworks. It was originally developed to providing a native scripting language debugging environment that seamlessly integrated with C and C++. It not only has very concise user interface but also has a very flexible architecture to easily add a new programming language in it.

For general information of debugging C and C++ extension for PHP with Speare code editor, please refer this document:
http://www.sevenuc.com/download/Small_IDE_for_C_and_C++_development.pdf

Debug PHP Projects

1. Show the debug toolbar

Click main menu "Navigate" → "Toggle Output".

2. Debug toolbar

From left to right: Start, Stop, Step Into, Step Out, Run To/Continue, Step Over/Step Next, Show Watches. The command

"Run To" tell the debugger run to meet a breakpoint or an exception occurred or the program exited. On the rightmost there are three other icons, they are, search items in the stackview, siding stackview, and clean debug output.

Search in the debug output

Click in the output area to let it got focus and use the shortcut key "Command + F" to do the searching.

3. Socket Port

You can set the socket communication port number both used by Debug Server and the editor. Open the Preferences of Speare and select the "Extra" panel and then input your number.

Note: Please remember to empty the port number when you switched to debugging with the default built-in programming languages with default port number.

4. Watches

Watch List: click debug toolbar "Show Watches" to manage watched variables and expressions, batches of watched items can be removed by multiple selection.

a. Whenever values of watched variables changed, debug session will pause at that point.
b. The values of evaluated expressions will be sent as a *"virtual stack node"* that paired the expression and its value together and representing as JSON key and value.

When watched variables and expressions display in stackview, their nodes normally has a different icon and text colour, and always placed on the top of stackview.

Add Watches:

a. Click debug toolbar "Show Watches".
b. Right click stackview and select menu item "Watch Variable".

Remove Watches:

a. Click debug toolbar "Show Watches".
b. Right click stackview and select menu item "Remove Watch".

5. Run Extra Commands

a. Click main menu "Debug" → "Send Debug Command".

b. Right click stackview and select menu item "Send Command".

When send an extra debug command, the input box will prompt in stackview, so please click stackview to let it got focus before select menu item.

6. Show Stackview Item Values

Right click stackview and select menu item "View Value AS", variables can be configured to display as "Hex", "Decimal", "Octal", "UTF-8" and "Unicode" sequences.

Caution:

Please ensure all source files have been dragged in the left side Treeview (Workspace Explorer) before start a debug session, because macOS app can't be allowed to access files outside of its sandbox.

Startup Debugging Session

1. Launch a Terminal.app window or tab, and start Debug Server.
Note: *The content directly printed by the debugging program will be displayed in the terminal instead of in the debug "output view".*

2. Click start button on the "Debug toolbar" to start debugging session.

3. Continually click "Step Over" button several times on the "Debug toolbar" to ignore some initialising steps in debugging session until it reached the main entry point.

The PHP Debugger

The PHP debugger of Speare code editor supports all kinds debugging of PHP applications and any version of PHP interpreter that has Xdebug support from PHP 5.x to PHP 7.x. Different with Lua, Ruby and Python debugging, this time Speare acts as debug server and Xdebug as the client.

Setup PHP development environment on macOS

This section described how to compile everything from source without Homebrew and Docker to establish a sample PHP development environment.

1. Toolchain Prepare

Please refer front part of this document:

http://www.sevenuc.com/download/compile and install GCC on macOS.pdf

2. Complie and Install MySQL

boost_1_59_0.tar.gz
mysql-5.7.34.tar.gz

a. Install boost 1.59.0

```
cd boost_1_59_0
./b2 --prefix=/usr/local --layout=system --build-type=complete
./b2 install
```

b. Compile MySQL

```
mkdir mybuild && cd mybuild
cmake .. \
-DCMAKE_INSTALL_PREFIX=/usr/local/mysql-5.7.34 \
-DDOWNLOAD_BOOST=OFF \
-DWITH_BOOST=/usr/local/boost \
-DLOCAL_BOOST_DIR="/usr/local/include/boost" \
-DWITH_SSL="system" \
make
sudo make install
```

c. Add MySQL executables to PATH:

```
echo 'export PATH=/usr/local/bin:$PATH' >> ~/.zshrc
source ~/.zshrc
```

3. Complie and Configure Nginx

pcre-8.43.tar.gz
nginx-1.20.1.tar.gz

a. **Compile Nginx**:

```
cd nginx-1.20.1
./configure \
  --prefix=/usr/local/nginx \
  --with-debug \
  --with-http_ssl_module \
  --with-http_realip_module \
  --with-http_ssl_module \
  --with-http_perl_module \
  --with-http_stub_status_module \
  --with-pcre=/usr/local/pcre-8.43
make
sudo make install
```

b. Configure Nginx:

cd /usr/local/nginx/conf/

sudo cp nginx.conf.default nginx.conf

vi nginx.conf

```
server {
        listen       80;
        server_name  localhost;
        error_log    /var/log/nginx/error.log;
        root         /Users/yeung/Sites;
        index        index.php index.html index.htm;
        try_files    $uri $uri/ /index.php;
        error_page   404      /404.html;

        location ~ \.php$ {
            # root                  html;
            fastcgi_split_path_info ^(.+\.php)(/.+)$;
            fastcgi_pass            127.0.0.1:9000;
            fastcgi_index           index.php;
            try_files               $uri = 404;

# Tell the PHP process that the current script is $document_root
# $fastcgi_scriptname,and PHP will find the script and process it, so
# the location of the script should be right.
# fastcgi_param  SCRIPT_FILENAME /scripts$fastcgi_script_name;

 fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
 include                         fastcgi_params;
        }
}
```

sudo mkdir -p /usr/local/nginx/logs

sudo touch /usr/local/nginx/logs/nginx.pid

4. Complie and Configure PHP

openssl-1.1.1d.tar.gz
libpng-1.6.37.tar
jpegsrc.v9d.tar.gz
libzip-1.7.3.tar.gz
bzip2-1.0.6.tar.gz
curl-7.77.0.tar.gz
libmcrypt-2.5.8.tar.gz
php-7.3.28.tar.gz

a. **Compile PHP**:

```
cd php-7.3.28.tar.gz

./configure \
  --prefix=/usr/local/php-7.3.28 \
  --with-config-file-path=/usr/local/php-7.3.28/etc \
  --enable-bcmath \
  --enable-mbstring \
  --enable-sockets \
  --enable-zip \
  --with-gd \
  --with-imap-ssl \
  --with-mysqli \
  --with-pear \
  --with-pdo-mysql \
  --with-xmlrpc \
  --with-xsl \
  --with-jpeg-dir=/usr/local/libjpeg \
  --with-png-dir=/usr/local/libpng \
  --with-bz2=/usr/local/libbz2 \
   --with-zlib-dir=/usr/local/zlib \
  --with-iconv=/usr/local/libiconv \
  --with-libzip=/usr/local/libzip \
  --with-curl=/usr/local/libcurl \
  --with-openssl=/usr/local/openssl-1.1.1d \
  --with-mysqli=/usr/local/mysql-5.7.34/bin/mysql_config \
  --enable-fpm
```

make
sudo make install

Prepare PHP configuration file

cp php.ini-development php.ini

b. Setup options in php.ini:

```
enable_dl=1
[xdebug]
zend_extension=/usr/local/php-7.3.28/lib/php/extensions/no-debug-non-zts-20180731/xdebug.so
;xdebug.default_enable=1
xdebug.remote_enable=1
xdebug.remote_host="127.0.0.1"
xdebug.remote_port=10000
xdebug.remote_handler="dbgp"
xdebug.remote_mode=req
xdebug.remote_connect_back=1
xdebug.remote_autostart=1
xdebug.remote_log_level=10
;xdebug.show_error_trace=1
;xdebug.show_exception_trace=1
;xdebug.remote_log =
```

c. Setup php-fpm:

1. configure settings:

sudo cp php.ini /usr/local/php-7.3.28/etc/
sudo cp sapi/fpm/php-fpm /usr/local/php-7.3.28/sbin/
sudo chmod +x /usr/local/php-7.3.28/sbin/php-fpm
sudo cp /usr/local/php-7.3.28/etc/php-fpm.conf.default ./php-fpm.conf

2. Setup options in php-fpm.conf:

pid=run/php-fpm.pid
include=/usr/local/php-7.3.28/etc/php-fpm.d/*.conf
sudo cp php-fpm.conf /usr/local/php-7.3.28/etc/

sudo cp /usr/local/php-7.3.28/etc/php-fpm.d/www.conf.default ./www.conf

3. Setup options in www.conf

user = www
group = www
sudo cp www.conf /usr/local/php-7.3.28/etc/php-fpm.d/

create log directory

mkdir -p /usr/local/php-7.3.28/var/log/

5. Complie Xdebug

xdebug-2.9.8.tgz

tar zxvf xdebug-2.9.8.tgz
cd xdebug-2.9.8/xdebug-2.9.8
/usr/local/php-7.3.28/bin/phpize
./configure --enable-xdebug --with-php-config=/usr/local/php-7.3.28/bin/php-config

make
sudo cp modules/xdebug.so /usr/local/php-7.3.28/lib/php/extensions/no-debug-non-zts-20180731/

6. Setup a WordPress website as a sample:

mkdir -p ~/Sites

Put wordpress-5.7.2.tar.gz in ~/Sites

tar zxvf wordpress-5.7.2.tar.gz

setup document root

sudo ln -s ~/Sites/wordpress /usr/local/nginx/html/wordpress
cp wp-config-sample.php wp-config.php

```
define( 'DB_NAME', 'wpdb' );
define( 'DB_USER', 'wpuser' );
define( 'DB_PASSWORD', 'wppass' );
define( 'DB_HOST', 'localhost' );
define( 'DB_CHARSET', 'utf8' );
```

7. Setup MySQL for the WordPress Website Sample

```
# setup MySQL database location
sudo mkdir -p /usr/local/var/mysql
mysqld --initialize --basedir=/usr/local/mysql-5.7.34
–datadir=/usr/local/var/mysql

# A temporary password is generated for root@localhost: xxxx
mysql -u root -p
# input the xxxx
# reset password for root user
ALTER USER 'root'@'localhost' IDENTIFIED BY 'newpass';
# shutdown MySQL server
mysqladmin --user=root --password shutdown


# re-start MySQL server
sudo mkdir -p /usr/local/var/log
mysqld_safe --datadir=/usr/local/var/mysql
--log-error=/usr/local/var/log/mysql.log &
mysql -u root -p
# input the new password to login

# create database 'wpdb' for WordPress
# create user 'wpuser' for WordPress with password 'wppass'

CREATE DATABASE IF NOT EXISTS wpdb DEFAULT CHARACTER SET utf8 COLLATE
utf8_bin;
USE wpdb;
CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'wppass';
GRANT ALL PRIVILEGES ON wpdb.* TO 'wpuser'@'localhost';
FLUSH PRIVILEGES;
quit;
```

Initialise the WordPress Sample Website

a. Shutdown system's Apache web server which taken port 80

```
sudo /usr/sbin/apachectl stop
```

b. Launch Nginx:

```
sudo /usr/local/nginx/sbin/nginx
```

```
# use the following command to re-launch when nginx.conf changed
```

```
sudo /usr/local/nginx/sbin/nginx -s reload
```

c. Launch php-fpm:

```
sudo /usr/local/php-7.3.28/sbin/php-fpm -c /usr/local/php-7.3.28/etc/php.ini
```

close all php-fpm processes

```
ps aux | grep "php"
```

```
sudo kill -9 `sudo ps -ef | grep php-fpm | grep -v grep | awk '{print $2}'`
```

d. Launch Safari and goto the url:
http://localhost/wordpress/wp-admin/install.php

```
# Welcome!
```

```
# Welcome to the famous five-minute WordPress installation process!
```

setup admin account for WordPress until a message appeared

```
# Success!
```

Begin a debug session with WordPress sample

Steps of PHP Debugging with Speare:

1. Drag entire ~/Sites/wordpress folder into Speare code editor
open wordpress/wp-content/themes/twentytwentyone/functions.php
add a breakpoint at line 39 in function twenty_twenty_one_setup()

2. Click main menu "Debug" → "Start"
the folowing message should printed in debug output view

```
PHP Debug Server listening on port: 10000
```

3. Login WordPress:
Input the following url in address field:
http://localhost/wordpress/wp-admin/customize.php?theme=twentytwentyone&return=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fthemes.php

4. Click debug toolbar

Add breakpoint, step in, step out, step next, watch stack trace ...

Switch PHP Interpreter

You can directly switch any version of PHP interpreter, it does not affect the server side Speare. PHP is an excellent programming language to develop Web Applications based on frameworks such as Drupal, Zend Framework, CodeIgniter, Symfony framework etc, but not limited that, in fact PHP is also very suitable to develop command line applications.

Extend Speare Code Editor for PHP development

Speare Code Editor can be easily extended to support any type of PHP development including web applications that based on frameworks such as Drupal, Zend Framework, CodeIgniter, Symfony and Yii framework and all kinds of command line applications.

To add scripts to better support debugging PHP in Speare code editor, please download the guide document and following the description in it.

© 2020 Sevenuc Consulting
All rights reserved