Interactive shell

As of PHP 5.1.0, the CLI SAPI provides an interactive shell using the **-a** option if PHP is compiled with the **--with-readline** option.

Using the interactive shell you are able to type PHP code and have it executed directly.

Example #1 Executing code using the interactive shell

```
$ php -a
Interactive shell

php > echo 5+8;
13
php > function addTwo($n)
php > {
php { return $n + 2;
php { }
php > var_dump(addtwo(2));
int(4)
php >
```

The interactive shell also features tab completion for functions, constants, class names, variables, static method calls and class constants.

Example #2 Tab completion

Pressing the tab key twice when there are multiple possible completions will result in a list of these completions:

```
php > strp[TAB][TAB]
strpbrk   strpos    strptime
php > strp
```

When there is only one possible completion, pressing tab once will complete the rest on the same line:

```
php > strpt[TAB]ime(
```

Completion will also work for names that have been defined during the current interactive shell session:

```
php > $fooThisIsAReallyLongVariableName = 42;
php > $foo[TAB]ThisIsAReallyLongVariableName
```

The interactive shell stores your history which can be accessed using the up and down keys. The history is saved in the `~/.php_history` file.

As of PHP 5.4.0, the CLI SAPI provides the `php.ini` settings `cli.pager` and `cli.prompt`. The `cli.pager` setting allows an external program (such as `less`) to act as a pager for the output instead of being displayed directly on the screen. The `cli.prompt` setting makes it possible to change the *php >* prompt.

In PHP 5.4.0 it was also made possible to set `php.ini` settings in the interactive shell using a shorthand notation.

Example #3 Setting *php.ini* settings in the interactive shell

The `cli.prompt` setting:

```
php > #cli.prompt=hello world :>
hello world :>
```

Using backticks it is possible to have PHP code executed in the prompt:

```
php > #cli.prompt=`echo date('H:i:s');` php >
15:49:35 php > echo 'hi';
hi
15:49:43 php > sleep(2);
15:49:45 php >
```

Setting the pager to *less*:

```
php > #cli.pager=less
php > phpinfo();
(output displayed in less)
php >
```

The `cli.prompt` setting supports a few escape sequences:

`cli.prompt` escape sequences

Sequence	Description
\e	Used for adding colors to the prompt. An example could be *\e[032m\v \e[031m\b \e[34m\> \e[0m*
\v	The PHP version.
\b	Indicates which block PHP is in. For instance /* to indicate being inside a multi-line comment. The outer scope is denoted by *php*.
\>	Indicates the prompt character. By default this is >, but changes when the shell is inside an unterminated block or string. Possible characters are: *' " { (>*

Note:
Files included through auto_prepend_file and auto_append_file are parsed in this mode but with some restrictions - e.g. functions have to be defined before called.

Note:
Autoloading is not available if using PHP in CLI interactive mode.

User Contributed Notes 13 notes

add a note

147 Ryan P — 6 years ago

```
Interactive Shell and Interactive Mode are not the same thing, despite the similar names and functionality.

If you type 'php -a' and get a response of 'Interactive Shell' followed by a 'php>' prompt, you have interactive shell available (PHP was compiled with readline support). If instead you get a response of 'Interactive mode enabled', you DO NOT have interactive shell available and this article does not apply to you.

You can also check 'php -m' and see if readline is listed in the output - if not, you don't have interactive shell.

Interactive mode is essentially like running php with stdin as the file input. You just type code, and when you're done (Ctrl-D), php executes whatever you typed as if it were a normal PHP (PHTML) file - hence you start in interactive mode with '<?php' in order to execute code.

Interactive shell evaluates every expression as you complete it (with ; or }), reports errors without terminating execution, and supports standard shell functionality via readline (history, tab completion, etc). It's an enhanced version of interactive mode that is ONLY available if you have the required libraries, and is an actual PHP shell that interprets everything you type as PHP code - using '<?php' will cause a parse error.

Finally, if you're running on Windows, you're probably screwed. From what I'm seeing in other comments here, you don't have readline, and without readline there is no interactive shell.
```

55 spencer at aninternetpresence dot net — 6 years ago

```
In Windows, press Enter after your ending PHP tag and then hit Ctrl-Z to denote the end-of-file:

C:\>php -a
Interactive mode enabled

<?php
echo "Hello, world!";
?>
^Z
Hello, world!

You can use the up and down arrows in interactive mode to recall previous code you ran.
```

5 #linuxmint-es — 7 months ago

```
For use interactive mode enabled on GNU/Linux on distros Debian/Ubuntu/LinuxMint you must install "php*-cli" and "php*-readline" packages from official repository.
Example:
>$sudo aptitude install php5-cli php5-readline

After that you can use interactive mode.
Example:
~ $ php -a
Interactive mode enabled

php >echo "hola mundo!\n";
hola mundo!
php >

I hope somebody help it!
```

14 Anonymous — 7 years ago

```
Just a few more notes to add...

1) Hitting return does literally mean "execute this command".  Semicolon to note end of line is still required.  Meaning, doing the following will produce a parse error:

php > print "test"
php > print "asdf";

Whereas doing the following is just fine:

php > print "test"
php > ."asdf";

2) Fatal errors may eject you from the shell:

name@local:~$ php -a
php > asdf();

Fatal Error: call to undefined function...
name@local:~$

3) User defined functions are not saved in history from shell session to shell session.

4) Should be obvious, but to quit the shell, just type "quit" at the php prompt.

5) In a sense, the shell interaction can be thought of as linearly following a regular php file, except it's live and dynamic.  If you define a function that you've already defined earlier in your current shell, you will receive a fatal "function already defined" error only upon entering that closing bracket.  And, although "including" a toolset of custom functions or a couple of script addon php files is rather handy, should you edit those files and wish to "reinclude" it again, you'll cause a fatal "function x already defined" error.
```

12 Anonymous — 7 years ago

```
It seems the interactive shell cannot be made to work in WIN environments at the moment.

Using "php://stdin", it shouldn't be too difficult to roll your own.  You can partially mimic the shell by calling this simple script (Note: Window's cmd already has an input history calling feature using the up/down keys, and that functionality will still be available during execution here):

<?php

$fp = fopen("php://stdin", "r");
$in = '';
while($in != "quit") {
    echo "php> ";
    $in=trim(fgets($fp));
    eval ($in);
    echo "\n";
    }

?>

Replace 'eval' with code to parse the input string, validate it using is_callable and other variable handling functions, catch fatal errors before they happen, allow line-by-line function defining, etc.  Though Readline is not available in Windows, for more tips and examples for workarounds, see http://www.php.net/manual/en/ref.readline.php
```

3 lee8oi at gmail dot com — 6 years ago

```
I use git-bash in windows to connect to my servers via SSH. When I use the interactive mode via 'php -a' command I have to hit ctrl+d twice to execute the entered code. Example:
(<ctrl+d> denotes hitting ctrl & D)

-bash$ php -a
Interactive mode enabled
<?php
echo 'hello world';
?><br />
<ctrl+d>
<ctrl+d>
hello world<br />
-bash$

Note: this still displays the <br /> tag but without the tag your output would likely be attached to your bash prompt like this:

hello world-bash$
```

1 wheat at wheatdesign dot com — 6 months ago

```
If you're stuck on Widows or any other machine where PHP was not compiled with readline support, one solution is to use a web-based PHP CLI. I use this in training classes, especially the sort where people bring their own laptops and I can't assume they have PHP installed. The best one I've found--partly because of the UX and partly because it's free (no credit card required) and quick to setup, is http://repl.it
```

0 alexmarcxyz at gmail dot com — 3 days ago

```
While configuring php shell script, We need to take care of these commands, Abstract.php, Compiler.php, Indexer.php, Log.php, You can check more details about these commands at, https://www.cloudways.com/blog/php-shell-scripts-magent . Hope it will help your readers as well as I got help from your and this post.
```

0 John — 1 month ago

```
If you delete your "~/.php_history", you MUST re-create the file manually!

Because after I deleted my history file, "php -a" (interactive mode) never saved any history anymore.

It only started working after I ran "touch ~/.php_history" to create an empty file. From then on, PHP is saving history again!

I thought this was a bit unusual. Normally, applications recreate their history files themselves. But just be aware of the fact that PHP works this way instead, guys and girls! :-)
```

0 elijah at elijahlynn dot net — 3 years ago

```
Bug #55496 Interactive mode doesn't force a newline before the prompt =>
https://bugs.php.net/bug.php?id=55496

Fixed on July 24th, 2014 @ http://git.php.net/?p=php-src.git;a=commit;h=71d3a69425449972f4efdf7228c6f7e49e090755

Until then, this will work:

php -dcli.prompt="\nphp> " -a
```

0 alexandrebr at gmail dot com — 6 years ago

```
For those who (just like me) can't get it working, try to press CTRL+D after inserting some commands.

Example:
php
<?php
echo "Hello World!\r\n";
(Hit CTRL+D here)
Hello World!

This is NOT interactive mode, but may help you.

To have the "-i" available, you'll need the following arguments while compiling PHP:
--with-readline e --with-libedit
```

-2 xEviL — 7 years ago

```
When building php on FreeBSD from ports one can add --with-readline option by manually editing the var CONFIGURE_ARGS in Makefile inside the php port directory and proceeding with build as usual.
```

-6 Shane Harter — 4 years ago

```
If you've ever wanted to build your own interactive shell, I released a project recently that makes it insanely easy to build awesome shell apps in PHP. It blends features from Zend2 and Symonfy2 with things like regex routing, state management, etc. Check it out here:

https://github.com/shaneharter/sheldon
```

add a note