
objects in php new \$foo->className(); Class name must be a valid object or a string

Posted by Jack Bates - 2008/05/08 05:14

I am trying to load PHP objects stored in a database, where the class name is stored in a column: `$object = new $resultSet->getString(1);` This fails for the same reason that the following fails: `<?php class Foo { public function className() { return 'Foo'; } } $foo = new Foo; $bar = new $foo->className();` Fatal error: Class name must be a valid object or a string in test.php on line 12 I guess this error is due to the confusion of parsing `()` as the argument list for the `className` function, or the `Foo` constructor... I work around this error by using a temp variable: `$tmp = $foo->className(); $bar = new $tmp;` - however the above reads like hacky code : (When calling dynamically named functions, I generally use `call_user_func()` to avoid awkwardness with `$object->$tmp($arg1, ...)` In other words, I prefer: `call_user_func(array($object, 'get' . $someName), $arg1, ...);` - to: `$tmp = 'get' . $someName; $object->$tmp($arg1, ...);` However there does not appear to be an analog of `call_user_func()` for constructing new instances of dynamically named classes? If I recall correctly, there was also a way to work around calling dynamically named functions (e.g. `$object->$tmp($arg1, ...);`) using curly braces: `$object->{'get' . $someName}($arg1, ...);` - however I cannot recall the exact syntax. Can anyone confirm that there is a curly brace syntax for calling dynamically named functions? Could it be applied to instantiating dynamically named classes? Can anyone recommend a cleaner alternative to: `$tmp = $foo->className(); $bar = new $tmp;` Thanks and best wishes, Jack

=====

objects in php new \$foo->className(); Class name must be a valid object or a string

Posted by Stut - 2008/05/08 05:14

name is stored in a column: `$object = new $resultSet->getString(1);` This fails for the same reason that the following fails: `<?php class Foo { public function className() { return 'Foo'; } } $foo = new Foo; $bar = new $foo->className();` I would rather have a factory method that returns a new instance of the class. There's no need for the outside world to know the class name. `<?php class Foo { public function newInstance() { return new self(); } public function test($a) { echo 'test: ' . $a . n; } } $foo = new Foo; $foo->test('foo'); $bar = new $foo->newInstance(); $bar->test('bar');` ? However, if you insist on doing it your way can I make a small suggestion? It's better to spend your time on functionality rather than finding ways to save some typing. I see no reason to try to combine the two statements - saving typing and a pitiful amount of disk space are the only benefits. -Stut

=====

objects in php new \$foo->className(); Class name must be a valid object or a string

Posted by Casey - 2008/05/08 05:14

name is stored in a column: `$object = new $resultSet->getString(1);` This fails for the same reason that the following fails: `<?php class Foo { public function className() { return 'Foo'; } } $foo = new Foo; $bar = new $foo->className();` Fatal error: Class name must be a valid object or a string in test.php on line 12 I guess this error is due to the confusion of parsing `()` as the argument list for the `className` function, or the `Foo` constructor... I work around this error by using a temp variable: `$tmp = $foo->className(); $bar = new $tmp;` - however the above reads like hacky code : (When calling dynamically named functions, I generally use `call_user_func()` to avoid awkwardness with `$object->$tmp($arg1, ...)` In other words, I prefer: `call_user_func(array($object, 'get' . $someName), $arg1, ...);` - to: `$tmp = 'get' . $someName; $object->$tmp($arg1, ...);` However there does not appear to be an analog of `call_user_func()` for constructing new instances of dynamically named classes? If I recall correctly, there was also a way to work around calling dynamically named functions (e.g. `$object->$tmp($arg1, ...);`) using curly braces: `$object->{'get' . $someName}($arg1, ...);` - however I cannot recall the exact syntax. Can anyone confirm that there is a curly brace syntax for calling dynamically named functions? Could it be applied to instantiating dynamically named classes? Can anyone recommend a cleaner alternative to: `$tmp = $foo->className(); $bar = new $tmp;` Thanks and best wishes, Jack Does... `<?php $bar = new $foo->className();` ? ...work? Otherwise, I'd just do... `<?php $className = $foo->className(); $bar = new $className;` ? ...instead of `$tmp`. - Hide quoted text -- Show quoted text -

=====