

Нововведения объектной модели PHP5

Возможность	Описание
Конструкторы/деструкторы	<code>__construct()/__destruct()</code> (последний без параметров) При перезагрузке метода вызов методов класса-предка осуществляется через <code>parent::__construct()</code> и <code>parent::__destruct()</code>
Передача объектов по ссылке	<code>stdClass \$class1 = \$class2;</code> - передача по ссылке <code>stdClass \$class1 = clone \$class2;</code> - копирование всего класса возможно изменить алгоритм клонирования, с помощью специального метода с именем <code>__clone()</code> (новый объект - <code>\$this</code> , уже существующий (копия которого создается) - <code>\$that</code>)
Доступ к атрибутам и методам	<code>private</code> - можно обратиться только изнутри класса <code>protected</code> - можно обращаться к переменной или методу из классов-потомков <code>public</code> - можно обращаться из любого места программы
Константы в классах	<code>class MyClass { const Property = 5; }</code> Доступ по <code>MyClass::Property</code> (Обращения через <code>\$this</code> не поддерживаются)
Статические атрибуты и методы	Статические методы можно вызывать без создания объекта этого класса <code>class MyClass { static function Display() {} }</code> Вызов <code>MyClass::Display();</code>
Абстрактные классы и методы	Если метод определяется как <code>abstract</code> , он должен быть переопределен в классе-потомке. При этом параметры переопределенного метода должны совпадать с параметрами абстрактного метода. Модификатор уровня доступа для абстрактных методов не учитывается. Уровень доступа определяется методом, переопределяющим абстрактный. <code>abstract class BaseClass { abstract function Display(); }</code> <code>class ExtClass extends BaseClass { public function Display() {} }</code>
Интерфейсы	Использование интерфейсов позволяет использовать множественное наследование. Таким образом, класс может реализовывать несколько интерфейсов одновременно, а не расширять только один абстрактный класс. <code>interface Displayable { function Display(); }</code> <code>interface Storeable { function Store(); }</code> <code>class Figure implements Displayable, Storeable {</code> <code>function Display() {}</code> <code>function Store() {}</code> <code>}</code>
Финальные классы и методы	Метод класса, объявленный как финальный, невозможно переопределить в классе-потомке. Финальный класс невозможно использовать для создания классов-наследников. <code>class FinalClass { final function Display() {} }</code>
Преобразование ссылок на объекты, возвращенные из методов	Можно вызывать цепочку методов в одном операторе. <code>\$Object2->Process(\$Object1->Process(\$Object1);</code> (метод <code>Process</code> возвращает объект)
Итератор по атрибутам класса	Все переменные класса, доступные в текущем контексте, могут быть перебраны циклом <code>foreach</code> . <code>foreach (\$Object as \$PropName => \$PropValue) {}</code>
Указание типов объектов при вызове методов	<code>public function Process(MyClass \$Object) {}</code>
Проверка объекта на принадлежность классу	<code>if (\$Object instanceof MyClass) {}</code>
Обработка обращений к несуществующим атрибутам, несуществующим методам	возможно перехватывать обращение к несуществующим или закрытым свойствам класса, методам. Для этого введены <code>__get()</code> , <code>__set()</code> и <code>__call()</code> <code>function __set(\$Name, \$Value) {}</code> <code>function __get(\$Name) {}</code> <code>function __call(\$Name, \$Args) {}</code>
Автоматическая загрузка классов	<code>function __autoload(\$ClassName) {}</code>
<code>__METHOD__</code> с именем класса и метода	Вывод: <code>MyClass::MyMethod</code>
Обработка исключений	Блоки <code>try</code> , <code>throw</code> и <code>catch</code> : <code>throw new Exception("Cannot divide by zero");</code> <code>try {} catch (Exception \$MyException) {}</code> <code>try { throw new NewException("Error"); } // свой класс-обработчик</code> <code>catch (NewException \$MyException) {} // class NewException extends Exception {}</code> <code>catch (Exception \$MyException) {}</code>
Работа <code>foreach</code> со ссылками	<code>foreach (\$Array as &\$Value) { \$Value = 5; }</code>
Значения по умолчанию, передаваемые по ссылке	<code>function MyFunction(&\$Argument = null) {}</code>
XML и Web Services	<code>SimpleXML</code> , <code>SOAP</code> , <code>XSLT</code> , <code>MySQLi</code> , <code>SQLite</code> , <code>Tidy</code>