Chyba nie muszę nikogo przekonywać do tego, że warto jest podzielić kod na logiczne moduły, zamiast umieszczać wszystkiego w jednym pliku. Później można za pomocą funkcji require_once / include_once załączyć jedynie potrzebne skrypty i z tych cegiełek budować nasz system. Niestety takie rozwiązanie, prócz niezaprzeczalnych plusów, ma też minusy. W tym artykule postaram się pokazać jak owe minusy zniwelować.

Nim zaczniesz warto

- wiedzieć cokolwiek o PHP 5 obiektowym
- posiadać serwer www (może być lokalny, np. Wamp)
- mieć jakieś 5-10 minut

Przykład "brzydkiego" kodu

{codecitation class='brush: php'}

```
<?php  require_once 'pliki/klasa1.php';  require_once 'pliki/klasa2.php';  require_once 'pliki/klasa3.php';  require_once 'pliki/klasa4.php';  require_once 'pliki/klasa5.php';
```

{/codecitation}

Wady powyższego kodu

1. Nie jest powiedziane, że wszystkie klasy będą w ogóle wykorzystane (jednak lepiej jest wszystkie require trzymać razem, niż rozsiać je po kodzie). A więc część niepotrzebnie zwalnia pracę bezcelowo się ładując
2. Na początku skryptu mamy wiele linii kodu, które tak naprawdę niewiele robią.
3. Dodając następne klasy musimy pamiętać o odpowiednim zaincludowaniu ich [także o

kolejności includowania w przypadku dziedziczenia itp.]

Rozwiązanie

Starczy jedynie użyć ciekawej funckji PHP - autoload . Sprójrz na prosty przykład:

{codecitation class='brush: php'}

<?php function __autoload($className) { echo $className; } $obj = new MyClass1(); // wyswietli "MyClass1" $obj2 = new MyClass2(); // wyswietli "MyClass2" ?>

{/codecitation}

Skoro możemy wyświetlić nazwę klasy, to czemu nie wykorzystać tego do załączenia plików z klasami! Starczy jedynie używac odpowiednich nazw.

{codecitation class='brush: php'}

<?php function __autoload($className) { $path = 'pliki/'.$className. '.php'; require_once $path; } $obj1 = new klasa1(); $obj2 = new klasa2(); $obj3 = new klasa3(); $obj4 = new klasa4(); $obj5 = new klasa5(); ?>

{/codecitation}

Tym sposobem możemy załączyć wszystkie potrzebne pliki. Oczywiście w folderze 'pliki' powinny się znajdować pliki 'klasa1.php', 'klasa2.php', ..., 'klasa5.php'. I w tych plikach powinny znajdować sie deklaracje odpowiednich klas.

Poprawki

Każdy kod na początku nie jest idealny, choć często działa. Warto jednak tu zabezpieczyć sie przed jednym bardzo niebezpiecznym zjawiskiem - możliwością wywołania klasy, która nie istnieje. Warto zastosowac bardzo prostą w użyciu funkcję file_exists .

{codecitation class='brush: php'}

```
<?php function __autoload($className) { $path = 'pliki/'.$className. '.php'; require_once $path; } $obj1 = new klasa1(); ?>
```

{/codecitation}

Załóżmy teraz, że nie będzie pliku 'pliki/klasa1.php'. Co wtedy? Oczywiście powstanie błąd. Warto się przed tym zabezpieczyć:

{codecitation class='brush: php'}

```
<?php function __autoload($className) { $path = 'pliki/'.$className. '.php'; if(file_exists($path)) { require_once $path; } else { // tu mozesz wylac sobie maila o bledzie die('Przykro nam, wystapil niespodziewany blad systemu.'); } } $obj1 = new klasa1(); ?>
```

{/codecitation}

Takie wydawałoby się oczywiste i banalne zabezpieczenie pozwala nam uniknąć sytuacji, które mogą narazić nas na spore straty (od kwestii finansowych, przez zawieszenie systemu, po bezpieczeństwo i prestiż).

Co dalej

Skoro już potrafimy ładować automatycznie pliki, to warto teraz nauczyć się jak odpowiednio je przetrzymywać, aby jeszcze przyjemniej móc zarządzać naszym kodem. Ale to już jest temat na inny artykuł, który mam zamiar niedługo napisać.