Require

ကျွန်တော်တို့ အနေနဲ့ PHP Program တွေ ရေးတဲ့ နေရာမှာ အစောပိုင်း ကာလတွေတုန်းက

- require
- require_once
- include
- include_once

Require Vs Include

Require နဲ့ include က အလုပ်လုပ်ပုံ အတူတူပါပဲ။ အဓိကအားဖြင့် error ပြတဲ့ ပုံစံမှာ ကွာသွားပါတယ်။ require က require လုပ်တဲ့ file မတွေ့ဘူးဆိုရင် ရပ်သွားပါတယ်။ include ကတော့ file မတွေ့ရင် warning သာ ပြပြီး script ကို run ပါတယ်။

_once keyword

_once keyword ကတော့ memory ပေါ်မှာ load တစ်ခါ load လုပ်ထားတယ် ဆိုရင် ထပ်ပြီး လုပ်စရာ မလိုတော့ဘူးဆိုတဲ့ အဓိပ္ပါယ်ပါ။

Autoload

Autoload ကတော့ PHP ရဲ့ magic method တစ်ခု ဖြစ်ပါတယ်။ ကျွန်တော်တို့ အနေနဲ့ file တွေ အများကြီးကို require လုပ်မယ်ဆိုရင် အတော် အလုပ်ရှုပ်ပါတယ်။ MVC လို့ ခေါ်တဲ့ Model သပ်သပ်၊ View သပ်သပ်၊ Controller သပ်သပ် ခွဲခြားပြီး ရေးကြတဲ့အခါမှာ folder တွေကို ခွဲခြားပြီး ရေးရပါတယ်။ အဲဒီလို ရေးရတဲ့ အပြင်

developer ရဲ့လိုအပ်ချက်အရ ထပ်ဖြည့်တာတွေလဲ အများကြီး ဖြစ်လာပါတယ်။ အဲဒီလို ဖြစ်လာတဲ့အခါ require နဲ့သာ ရေးမယ်ဆိုရင် အလုပ်အတော် ရှုပ်မှာ ဖြစ်ပါတယ်။ developer အနေနဲ့ Model ဆိုရင် model folder ထဲ Controller ဆိုရင် Controller ထဲ ထည့်လိုက်View ဆိုရင် View ထဲထည့်လိုက်ပြီး အားလုံးကို လိုက်ပြီး require လုပ်နေစရာမလိုမှ အဆင်ပြေပါလိမ့်မယ်။ အဲဒီအတွက် require ဆိုတဲ့ method အစား autoload ဆိုတဲ့ magic method ကို သုံးလာကြပါတယ်။

```
// Cat.php
<?php
class Cat {
      function say() {
            echo "Meow!";
      }
}
// Dog.php
<?php
class Dog {
      function say() {
            echo "Bark!";
      }
}
// index.php
<?php
    function __autoload($classname) {
      $filename = $classname . ".php";
      if(is_readable($filename)) {
            require $filename;
      }
    }
    $cat = new Cat;
```

```
    $cat->say();
    $dog = new Dog;
    $dog->say();
```

အဲဒီလို ရေးမယ်ဆိုရင် __autoload() ကို သုံးပြီး classname နဲ့တူတဲ့ file တွေကို autoload လုပ်ပေးပါလိမ့်မယ်။ require နဲ့ ရေးမယ်ဆိုရင်

```
require "Cat.php";

require "Dog.php";
```

ဆိုပြီး ရေးပေးနေရပါလိမ့်မယ်။ ဒါပေမယ့် အဲဒီ php magic method __autoload မှာ ပြဿနာ ရှိတာက __autoload ဆိုတဲ့ magic method ကို တစ်ခုတည်း သုံးလို့ရပါတယ်။ အဲဒီအတွက် multiple autoload လုပ်ချင်တယ်ဆိုရင် မရတော့ပါဘူး။ PHP အနေနဲ့ နောက်ပိုင်းမှာ __autoload() ကို deprecated လုပ်သွားဖို့ ရှိပါတယ်။

spl_autoload_register()

__autoload() megic method ရဲ့ အားနည်းချက် တစ်ခုဖြစ်တဲ့ တစ်ခုတည်း သုံးလို့ရတဲ့ ပြဿနာကို ဖြေရှင်းဖို့အတွက် spl_autoload_register() ဆိုတာကို PHP က သတ်မှတ်ပေးပါတယ်။ ဥပမာ ရေးကြည့်ရအောင်

controllers/CatController.php

```php
<?php
class CatController {
	function say() {
		return "Meow!";
	}
}
```

controllers/DogController.php

```php
<?php
class DogController {
	public function say() {
		return "Bark!";
	}
}
```

models/DogModel.php

```php
<?php
class DogModel {
	public function count() {
		return 5;
	}
 }
```

models/CatModel.php

```php
<?php
class CatModel {
	public function count() {
		return 3;
	}
 }
```

spl.php

```php
<?php
function autoloadModel($className) {
    $filename = "models/" . $className . ".php";
    if (is_readable($filename)) {
        require $filename;
    }
}

function autoloadController($className) {
    $filename = "controllers/" . $className . ".php";
    if (is_readable($filename)) {
        require $filename;
    }
}

spl_autoload_register("autoloadModel");
spl_autoload_register("autoloadController");

$catctl = new CatController;
$dogctl = new DogController;
$catModel = new CatModel;
$dogModel = new DogModel;

echo "Cat says " . $catctl->say() . ". We have " . $catModel->count() . "
cats.";
echo "<br />";
echo "Dog says " . $dogctl->say() . ". We have " . $dogModel->count() . "
dogs.";
```

spl_autoload_register() အနေနဲ့ MVC Pattern တွေမှာ folder တွေ ကွဲနေတဲ့ အနေအထားအတွက် အများကြီးအထောက် အပံ့ပေးပါတယ်။

PSR-0

PSR-0 ကတော့ FIG လို့ခေါ်တဲ့ Framework Interoperability Group က သတ်မှတ်လိုက်တဲ့ standard တစ်ခု ဖြစ်ပါတယ်။ autoload ကို ပုံစံမျိုးစုံနဲ့ လုပ်ကြတဲ့အခါ တစ်ခုနဲ့ တစ်ခု စနစ်တွေက မတူကြတော့ပါဘူး။ အဲဒီလို မတူတဲ့အတွက် ပေါင်းစပ်ရလဲ ခက်ခဲလာပါတယ်။ အဲဒီအတွက် standard တစ်ခု သတ်မှတ်ဖို့ လိုတယ်ဆိုပြီး PSR-0, PSR-1, PSR2 စသည်ဖြင့် သတ်မှတ်ကြပါတယ်။ ဒီ ဆောင်းပါးနဲ့ Composer ဆိုတဲ့ ဆောင်းပါးကို ပေါင်းပြီး ဖတ်မယ်ဆိုရင် ပြည့်စုံသွားပါလိမ့်မယ်။

Soe Thiha Naung

Myanmar Links