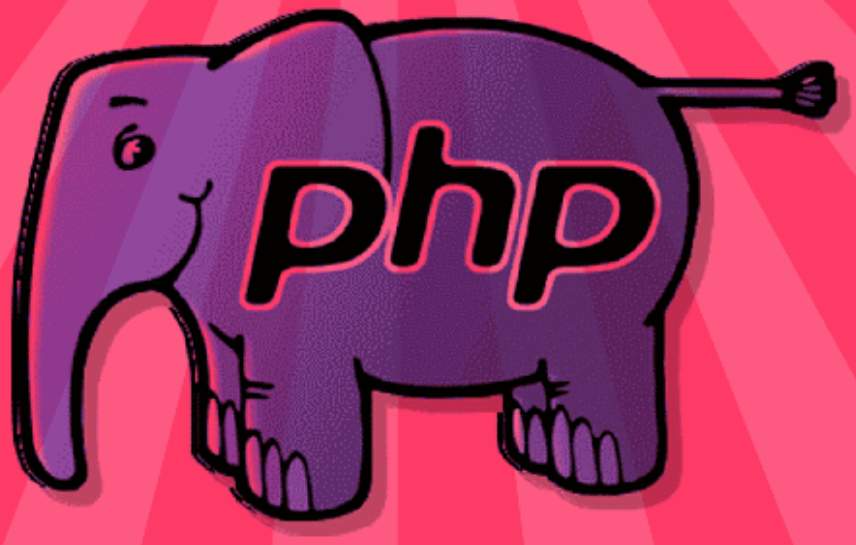
المختصر السريع فى

البرمجة غرضية التوجيه فى PHP

Object Oriented Programming



المؤلف والمبرمج

أحمد فتحى

البريد الالكترونى

ahmed_fa@hotmail.com

طبعة

2012

الموقع الداعم

موقع زيروجيت

http://0gate.com

محتوى الكتاب

هذا الكتاب مجانى
لا تنسونا صالح الدعاء

✓ مقدمة:-

البرمجة الموجهة بالأهداف(Object Oriented Programming) هي عبارة عن كتابة الكود بطريقة معينة والتي تسمح للمطورين بتجميع عدة مهام فى مجموعات تسمى كلاس Classes .
وهذا يساعد على عدم تكرار الكود أكثر من مرة في عدة أماكن، ويكون سهل الحفظ أيضا .
ومن الأشياء المفيدة أيضا أنة إذا كنت تريد تطوير الكود الموجود في الكلاس يمكنك ذلك بسهولة بدون التعديل على كود التطبيق كلة ، فيمكنك تطوير الجزء الذي تريده ولن يحدث أي مشاكل في التطبيق بشكل عام وهذا ما يميز البرمجة الموجهة بالأهداف .
البرمجة الموجهة بالأهداف تقوم بتخويف المطورين من الدخول فيها والتعامل معها وذلك لأنها تقدم الكثير من الجمل البرمجة الجديدة والتي تجعل بعض المطورين تعتقد أنها صعبة ، ولكن العكس هو الصحيح البرمجة باستخدام البرمجة الموجهة بالأهداف سهلة وبسيطة وتساعد على تطوير عملك بسهولة وبمجهود أقل .

✓ ما الفرق بين الكلاس Class و الكائن Object :-

كمثال بسيط الفرق بينهم هو مثل الفرق بين التخطيط لبناء مباني ، وبناء المباني نفسها، أي أن الكلاس هو التخطيط والذي توضح فيه كيف ستقوم ببناء التطبيق الذي تريده، أما الكائن فهو ما يقوم بتنفيذ ما قمت بتخطيطه في الكلاس وهو المسئول عن التنفيذ .

✓ بنية الكلاس وكيفية إنشائها:-

لكى تقوم بإنشاء هيكل الكلاس الرئيسي تمهيدا بعد ذلك لكتابة محتوياته ، فإنه دائما نستخدم كلمة محجوزة في اللغة تسمى `class` هذه الكلمة هي التي تقوم بإنشاء الكلاس، ثم نستخدم أيضا أقواس { } واحد في بداية الكلاس وواحد في نهاية الكلاس والذي يدل على أنة ما بداخل القوسين هو جسم الكلاس .
وفى ما يلي مثال على إنشاء البنية الأساسية للكلاس .

```
<?php

   class MyClass
   {
    // هنا يتم كتابة محتوى الكلاس
   }

?>
```

بعد إنشاء الكلاس يأتي دور التنفيذ ويمكن فعل ذلك بإنشاء كائن من هذا الكلاس ، هذا الكلاس هو المسئول عن تنفيذ الكلاس الذي قمنا بإنشائه ، والذي سوف نستخدمه أيضا بعد ذلك للتعامل مع هذا الكلاس .
وطريقة كثابتة هي أولا نقوم بإنشاء متغير لكى نخزن الكائن في هذا المتغير ثم نقوم بإسناد الكائن لهذا المتغير وذلك باستخدام الكلمة المحجوزة `new` هذه الكلمة معناها أنك تريد عمل نسخة من الكلاس لكى تقوم بالتعامل معها ، مع العلم أنك يمكنك عمل أكثر من نسخة من الكلاس وذلك بإنشاء أكثر من كائن لنفس الكلاس .

```
<?php

 $obj = new MyClass;

?>
```

ويمكننا عمل كود بسيط يقوم بطباعة محتوى الكلاس وذلك كما يلي .

```
<?php

  var_dump($obj);

?>
```

والكود النهائي سيكون بهذا الشكل .

```
<?php

  class MyClass
  {
   // هنا يتم كتابة محتوى الكلاس
  }

   $obj = new MyClass;
   var_dump($obj)

?>
```

وبعد تنفيذ الكود السابق وتنفيذة فى صفحة الويب سوف يكون الناتج بالشكل التالي .

```
object(myclss)#1 (0) { }
```

✓ تعريف خصائص الكلاس - Properties :-

المتغيرات داخل جسم الكلاس الرئيسي تسمى خصائص أو Properties بالانجليزية ، أي خصائص الكلاس ، وهذه الخصائص مثلها مثل المتغيرات التي نستخدمها بالطريقة المعتادة ، لكنها تختلف في شي واحد فقط وهو أنة لا يمكن الوصول أو التعامل مع هذه المتغيرات إلا عن طريق الكائن الذي قمنا بإنشائه والذي هو تابع للكلاس الذي بداخلة هذه الخاصية . وفى ما يلي نوضح طريقة كتابة خصائص الكلاس .

```
<?php

  class MyClass
  {
     public $prop = "الخاصية الأولى";
  }

   $obj = new MyClass;
   var_dump($obj)

?>
```

عندما شاهدنا الكود السابق بالتأكيد كلمة جديدة لم نأخذها سابقا وهى الكلمة المحجوزة public والتى نحدد مدى رؤية هذه الخاصية والتي سوف نشرحها بعد ذلك لكن كل شي بوقته .
نأتي الآن، لشي أخر وهو كيف يمكننا قراءه هذه الخاصية وطباعة قيمتها على الشاشة ؟

```
<?php

  echo $obj->prop;

?>
```

كما ذكرنا سابقا أنة نستطيع الوصول إلى الخصائص المتواجدة داخل الكلاس ، وذلك باستخدام الكائن التي قمنا بإنشائه سابقا والذي هو تابع الكلاس الخاصة بنا ، وقمنا بعمل ذلك عن طريق كتابة أسم الكائن أولا والذي يسمى في المثال السابق $obj ، ثم أستخدمنا السهم (->) والذي يستخدم للوصول إلى أعضاء الكلاس ، ثم بعد ذلك أسم الخاصية وهو prop . هذه هي طريقة الوصول لأي عضو بالكلاس ، والكود التالي هو الكود النهائي .

```
<?php
```

```php
  class MyClass
  {
     public $prop = "الخاصية الأولى";
  }

   $obj = new MyClass;
   echo $obj->prop;  // طباعة قيمة الخاصية

?>
```

✓ تعريف دوال الكلاس – Methods :-

الدوال داخل جسم الكلاس الرئيسي تسمى الطرق أو الأساليب أو Methods بالانجليزية، وهى تستخدم لأداء عمل محدد. وتستطيع الكلاس أن تحتوى على أكثر من دالة داخلها ، أي أنك قادر على إضافة أي عدد من الدوال داخل الكلاس . وفى ما يلي سوف نعرض عليكم مثال يوضح كيفية أستخدام الدوال .

```php
<?php

  class MyClass
  {
      public $prop = "خاصية الكلاس";
      public function setProperty($new_val)
      {
          $this->prop = $new_val;
      }

      public function getProperty()
      {
          return $this->prop . "<br />";
      }
  }

  $obj = new MyClass;
  echo $obj->getProperty(); // طباعة قيمة الخاصية
  $obj->setProperty("أنا خاصية جديدة"); //i قمنا بتغيير قيمة الخاصية
  echo $obj->getProperty();  // قمنا بطباعة قيمة الخاصية مرة أخرى بعد تعديلها

?>
```

في البداية نلاحظ بوجود كلمة جديدة في الكود السابق وهى $this , ونستخدمها عندما نريد الإشارة إلى دالة أو خاصية تابعة لنفس الكلاس ، ففي الكود السابق كتبنا مثلا $this->prop ، المقصود من $this هنا هو أننا نريد الإشارة إلى الخاصية prop هذه الخاصية هي عضو تابع للكلاس ، أي كأننا نقول له الخاصية التابعة لك أيها الكلاس قم بإسناد المتغير $new_val لها، وهذا المتغير ليس عضو في الكلاس وإنما هو متغير عادى نستخدمه كوسيط للدالة .
نرجع إلى موضوعنا وهو شرح الكود السابق ، قمنا بعمل دالتين داخل الكلاس هذه الدالتين أصبحو أعضاء تابعين لهذا الكلاس، الدالة الأولى setProperty تستخدم لتغيير قيمة الخاصية $prop التابعة للكلاس ، والدالة الأخرى getProperty تقوم بطباعة قيمة هذه الخاصية .

وكما ذكرنا سابقا أنة يمكننا إنشاء أكثر من كائن أو Object تابع لكلاس واحد، بحيث أنة يمكنك أن تضيف بيانات تابعة لهذا الكائن لا تؤثر على بيانات الكائن الأخر ، هيا بنا نشاهد مثال يوضح هذه الميزة .

```php
<?php

  class MyClass
  {
      public $prop = "قيمة الخاصية في الكلاس";
      public function setProperty($newval)
      {
```

```
        $this->prop = $newval;
    }

    public function getProperty()
    {
        return $this->prop . "<br />";
    }
}

// إنشاء كائنين
$obj = new MyClass;
$obj2 = new MyClass;

// جلب قيمة الخاصية من الكائنين
echo $obj->getProperty();
echo $obj2->getProperty();

// تعديل قيمة الخاصية فى الكائنين
$obj->setProperty("قيمة الخاصية الجديدة في الكائن الأول");
$obj2->setProperty("قيمة الخاصية الجديدة في الكائن الثاني");

// طباعة ناتج قيمة الخاصية فى الكائنين
echo $obj->getProperty();
echo $obj2->getProperty();

?>
```

✓ أستخدام دوال البناء والهدم - Constructor & Destructor :-

دالة البناء أو Constructor بالانجليزية، تستخدم إذا كنت تريد تنفيذ كود محدد عند إنشاء كائن جديد، فمن المعروف أن دالة البناء هي أول دالة يتم تنفيذها بعد إنشاء كائن، وغالبا تستخدم عندما يوجد كود مهم لابد من تنفيذه قبل تنفيذ أي دالة أخرى في الكلاس ، ولتوضيح ذلك تابع معنا فى المثال التالي .

```
<?php

  class MyClass
  {
      public $prop = "أنا خاصية";
      public function __construct()
      {
          echo 'تم تهيئة الكلاس "',__CLASS__,'" بنجاح <br />';
      }

      public function setProperty($newval)
      {
          $this->prop = $newval;
      }

      public function getProperty()
      {
          return $this->prop . "<br />";
      }
  }

  // إنشاء كائن جديد
  $obj = new MyClass;
```

```
  // جلب قيمة الخاصية
  echo $obj->getProperty();

  // طباعة رسالة فى نهاية الملف
  echo "نهاية الملف.<br />";

?>
```

بعد تطبيق المثال السابق ستجد أن أول شي تم تنفيذه في الكلاس هي دالة البناء __construct ، وأيضا يوجد شئ جديد فى الكود تم أستخدامة وهو الثابت __CLASS__ والذي يستخدم لجلب أسم الكلاس الذي نستخدمه .

دالة الهدم أو Destructor بالانجليزية ، وطريقة عملها هي عكس طريقة عمل دالة البناء ، أي إذا كانت دالة البناء يتم تنفيذها أول دالة بعد إنشاء الكائن ، فدالة الهدم هي أخر دالة يتم تنفيذها قبل تدمير الكائن ، وأكثر الاستخدامات شيوعا لها هي تنظيف الكلاس من البقايا ، أي يمكننا إزالة المتغيرات وتفريغ الذاكرة منها أو إغلاق الاتصال بقاعدة البيانات ، وغيرها من الاستخدامات الأخرى ، وفى ما يلي مثال يوضح أستخدام دالة الهدم .

```
<?php

  class MyClass
  {
      public $prop = "أنا خاصية";
      public function __construct()
      {
          echo 'تم تهيئة الكلاس "',__CLASS__,'" بنجاح <br />';
      }

      public function __destruct()
      {
          echo 'لقد تم تدمير الكلاس "',__CLASS__,'" بنجاح.<br />';
      }

      public function setProperty($newval)
      {
          $this->prop = $newval;
      }

      public function getProperty()
      {
          return $this->prop . "<br />";
      }
  }

  // إنشاء كائن جديد
  $obj = new MyClass;

  // جلب قيمة الخاصية
  echo $obj->getProperty();

  // طباعة رسالة فى نهاية الملف
  echo "نهاية الملف.<br />";

?>
```

فى المثال السابق قمنا باستخدام دالة الهدم ، وقمنا بطباعة جملة بداخلها ، بعد تطبيق هذا المثال سوف تجد أنة طباعة الجملة التي بداخل دالة الهدم أخر واحدة ، وهذا يدل أن دالة الهدم يتم تنفيذها كأخر دالة في الكلاس .

ويمكننا أيضا تدمير الكائن الخاص بنا بعد الانتهاء من استخدامه وذلك كما في الشكل التالي .

```
<?php
```

```php
  class MyClass
  {
      public $prop = "أنا خاصية";
      public function __construct()
      {
          echo 'تم تهيئة الكلاس "',__CLASS__,'" بنجاح <br />';
      }

      public function __destruct()
      {
          echo 'لقد تم تدمير الكلاس "',__CLASS__,'" بنجاح.<br />';
      }

      public function setProperty($newval)
      {
          $this->prop = $newval;
      }

      public function getProperty()
      {
          return $this->prop . "<br />";
      }
  }

  // إنشاء كائن جديد
  $obj = new MyClass;

  // جلب قيمة الخاصية
  echo $obj->getProperty();

  unset($obj); // هنا قمنا بتدمير الكائن

  // طباعة رسالة في نهاية الملف
  echo "نهاية الملف.<br />";

?>
```

✓ الوراثة - Inheritance :-

الكلاس يمكنها أن ترث خصائص ودوال كلاس أخرى وذلك باستخدام الكلمة المحجوزة extends ، ولكى نقوم بعمل وراثة بين كلاس وأخرى ، يمكنك رؤية المثال التالي .

```php
<?php

  class MyClass
  {
      public $prop = "أنا خاصية";
      public function __construct()
      {
          echo 'تم تهيئة الكلاس "',__CLASS__,'" بنجاح <br />';
      }

      public function __destruct()
      {
          echo 'لقد تم تدمير الكلاس "',__CLASS__,'" بنجاح.<br />';
      }
```

```
    public function setProperty($newval)
    {
        $this->prop = $newval;
    }

    public function getProperty()
    {
        return $this->prop . "<br />";
    }
}

class MyOtherClass extends MyClass
{
    public function newMethod()
    {
        echo "الكلاس الابن هو " . __CLASS__ . ".<br />";
    }
}

// إنشاء كائن جديد تابع للكلاس الابن
$newobj = new MyOtherClass;

// طباعة ما بداخل الدالة التابعة للكلاس الابن
echo $newobj->newMethod();

// جلب قيمة الخاصية من الكلاس الأم من خلال الكائن الذي يرث من الكلاس الابن
echo $newobj->getProperty();

?>
```

طيب ماذا لو نريد التعديل على دالة موروثة من الكلاس الأم أي أننا لا نريد أن نتقيد بالكود المكتوب بداخلها في الكلاس الأم ونريد تعديله ، فهل يمكن التعديل عليها وإضافة سلوك جديد للدالة بدلا من السلوك القديم لها ، تابع معنا في المثال التالي والذي يوضح ذلك .

```
<?php

class MyClass
{
    public $prop = "أنا خاصية";
    public function __construct()
    {
        echo 'تم تهيئة الكلاس "',__CLASS__,'" بنجاح <br />';
    }

    public function __destruct()
    {
        echo 'لقد تم تدمير الكلاس "',__CLASS__,'" بنجاح.<br />';
    }

    public function setProperty($newval)
    {
        $this->prop = $newval;
    }

    public function getProperty()
    {
        return $this->prop . "<br />";
    }
}
```

```
  class MyOtherClass extends MyClass
  {
      public function __construct()
      {
          echo "الباني الجديد التابع للكلاس " . __CLASS__ . "الابن.<br />";
      }
      public function newMethod()
      {
          echo "الكلاس الابن هو " . __CLASS__ . ".<br />";
      }
  }

  // إنشاء كائن جديد تابع للكلاس الابن
  $newobj = new MyOtherClass;

  // طباعة ما بداخل الدالة التابعة للكلاس الابن
  echo $newobj->newMethod();

  // جلب قيمة الخاصية من الكلاس الأم من خلال الكائن الذي يرث من الكلاس الابن
  echo $newobj->getProperty();

?>
```

في الكود السابق قمنا بإضافة دالة الباني أو Constructor في الكلاس الابن ، لكن الآن أصبحت الكلاس الأم لديها دالة الباني ، وأيضا كلاس الابن يحتوى على كلاس الباني ، لكن أي من الاثنين سيتم تنفيذة علما بأن الكلاس الابن MyOtherClass قام بالوراثة من الكلاس الأم MyClass ، والمفروض في هذه الحالة يقوم بتنفيذ دالة الباني في الكلاس الأم ، لكن أنت نسيت أننا قمنا بإضافة دالة الباني في الكلاس الابن أيضا ، أي أننا قمنا بتعديل دالة الباني في الكلاس الأم ، وأصبح دالة الباني في الكلاس الابن بدلا من دالة الباني في الكلاس الأم أى سيتم تنفيذ الدالة المتواجدة فى كلاس الابن .

طيب هناك شي أخر ، ماذا لو كنا نريد تعديل محتوى الدالة الموروثة من الكلاس الأم + إضافة محتوى الكلاس القديم أيضا الموجود في الدالة في الكلاس الأم ، أي نريد إضافة المحتوى القديم + إضافة محتوى جديد ، ويظهر الاثنين مع بعض ، يمكننا عمل ذلك باستخدام الكلمة parent ومعناها الكلاس الأم ، ثم إضافة المعامل :: ثم أسم الدالة فى الكلاس الأم. وفى المثال التالي نوضح طريقة الاستخدام .

```
<?php

  class MyClass
  {
      public $prop = "أنا خاصية";
      public function __construct()
      {
          echo 'تم تهيئة الكلاس "',__CLASS__,'" بنجاح <br />';
      }

      public function __destruct()
      {
          echo 'لقد تم تدمير الكلاس "',__CLASS__,'" بنجاح.<br />';
      }

      public function setProperty($newval)
      {
          $this->prop = $newval;
      }

      public function getProperty()
      {
          return $this->prop . "<br />";
```

```
    }
}

class MyOtherClass extends MyClass
{
    public function __construct()
    {
        // قمنا هنا باستدعاء دالة الباني التابعة للكلاس الأم
        parent::__construct();
        echo "الباني الجديد التابع للكلاس " . __CLASS__ . "الابن.<br />";
    }
    public function newMethod()
    {
        echo "الكلاس الابن هو " . __CLASS__ . ".<br />";
    }
}

// إنشاء كائن جديد تابع للكلاس الابن
$newobj = new MyOtherClass;

// طباعة ما بداخل الدالة التابعة للكلاس الابن
echo $newobj->newMethod();

// جلب قيمة الخاصية من الكلاس الأم من خلال الكائن الذي يرث من الكلاس الابن
echo $newobj->getProperty();

?>
```

في الكود السابق قمنا باستدعاء دالة الباني أو Constructor التابعة للكلاس الأم ، من داخل دالة الباني في الكلاس الابن هكذا `parent::__construct()` , وعند التطبيق سنجد أنة تم طباعة محتوى دالة الباني في الكلاس الام ودالة الباني في الكلاس الابن أيضا .

✓ معدلات الوصول - Access Modifiers :-

هى عبارة كلمة يتم إضافتها للدالة Method أو الخاصية Property ، والغرض من إضافة هذه الكلمة لعمل قيود على إمكانية الوصول إلى بيانات الدالة أو بيانات الخاصية ، على سبيل المثال تخيل أن لدينا ثلاثة منازل (منزل 1, منزل 2, منزل 3) . المنزل "رقم 1" عام أو Public أي يستطيع أي شخص الدخول فيه وأخذ ما بداخلة بدون أي مشاكل ، والمنزل "رقم 2 " هو منزل خاص أو Private لا يستطيع أي شخص أن يدخله إلى أهل بيته فقط أي أصحاب المنزل نفسة أو الأسرة نفسها هى فقط التي تدخل هذا المنزل ، أما المنزل "رقم 3 " فهو منزل محمى أو Protected فيستطيع أن يدخله الأقارب التابعين لهذه الأسرة أو السلالة العائلية فقط . أتمنى أن تكون الفكرة اقتربت إلى ذهنكم ولو قليل .
نرجع مرة أخرى لموضوعنا ، هناك ثلاثة معدلات وصول Access Modifiers للدوال والخصائص التابعة لأي كلاس .

- **Public**
  تعنى أن البيانات التي داخل الدالة أو الخاصية مرئية من الجميع ، ويمكن الوصول إليها من أي مكان سواء من داخل الكلاس أو من خارج الكلاس

- **Private**
  تعنى أنة لا يمكن الوصول إلى البيانات التي تحتويها الدالة أو الخاصية إلا من داخل الكلاس الخاص بها فقط ولا يمكن توريثها لكلاس أخرى .

- **Protected**
  تعنى أنة يمكن الوصول إلى البيانات التي تحتويها الدالة أو الخاصية من داخل الكلاس التابعة لها ، وتقبل أيضا التوريث .

وفيما يلي نقوم باستعراض مثال يوضح كل معدلات الوصول .

```php
<?php

  /**
   * تعريف الكلاس
   */
  class MyClass
  {
      public $public = 'Public';
      protected $protected = 'Protected';
      private $private = 'Private';

      function printHello()
      {
          echo $this->public;
          echo $this->protected;
          echo $this->private;
      }
  }

  $obj = new MyClass();
  echo $obj->public; // يعمل
  echo $obj->protected; // يحدث خطأ - Fatal Error
  echo $obj->private; // يحدث خطأ - Fatal Error
  $obj->printHello();


  /**
   * 2 تعريف الكلاس
   */
  class MyClass2 extends MyClass
  {
      // يمكننا إعادة الإعلان عن الدوال التي تستخدم Public, Protected
      protected $protected = 'Protected2';

      function printHello()
      {
          echo $this->public;
          echo $this->protected;
          echo $this->private;
      }
  }

  $obj2 = new MyClass2();
  echo $obj2->public; // يعمل
  echo $obj2->private; // غير معرف - Undefined
  echo $obj2->protected; // يحدث خطأ - Fatal Error
  $obj2->printHello();

?>
```

في الكود السابق قمنا بإنشاء 2 كلاس ، الأولى MyClass وهى الأم ، و MyClass2 وهى الابن ، في بداية الكلاس الأم قمنا بتعريف ثلاث خصائص (public, protected, private) وهى بنفس أسماء معدلات الوصول ، وقمنا بإضافة قيمة لكل خاصية ، ثم بعد ذلك قمنا بتعريف دالة باسم printHello وهى تقوم بطباعة قيم الخصائص التي قمنا بإنشائها، بعد ذلك قمنا بإنشاء كائن جديد باسم obj تابع للكلاس الأم ، ثم قمنا بالوصول بشكل مباشر إلى الثلاث خصائص ،وكما نعلم أننا يمكننا الوصول بشكل مباشر إلى الدوال والخصائص التي تستخدم معدل الوصول public لأنة يمكن الوصول إليها من داخل الكلاس وخارجها أيضا لأنها عامة ، أما بالنسبة إلى protected و private فلا يمكن الوصول إليهم بشكل مباشر من الخارج . أما في الكلاس الثاني MyClass2 قمنا بإعادة تعريف الخاصية protected وأضفنا لها قيمة جديدة مع العلم بأن هذه الخاصية موجودة في الكلاس الأم ، لكن هذا لا يحدث مشاكل لان الخصائص والدوال التي تستخدم معدل الوصول protected يمكن

لأي كلاس أخر يرث إلى الخاصية protected الموجودة في الكلاس الأم ، وهكذا مع بقية الكود .

طيب ماذا عن الكلمة Static ، هذة الكلمة تشبه أيضا معدلات الوصول الأخرى لكن لها طريقه عمل مختلفة ، فإذا قامت أى خاصية أو دالة استخدمها ، فأعلم أنة يمكن الوصول إلى هذه الخاصية أو الدالة التى تستخدم هذه الكلمة بشكل مباشر من الكلاس وليس من كائن ، أي أننا كما تعلم إذا كنا نريد الوصول إلى دوال أو خصائص فى كلاس ما لابد من إنشاء كائن أولا ثم من خلال هذا الكائن نصل إلى دوال وخصائص الكلاس هكذا .

```php
<?php

 class test {
  public function Mgs() {
   echo "<script type='text/javascript'> alert('السلام عليكم') </script>";
   }
 }

 $obj = new test;
 $obj->Mgs();

?>
```

في الكود السابق نرى الطريقة التي نعتاد عليها للوصول إلى دوال وخصائص أي كلاس ، وذلك عن طريق إنشاء كائن تابع للكلاس ، لكن مع أستخدام الكلمة Static سيكون شكل الكود السابق هكذا .

```php
<?php

 class test {
  static function Mgs() {
   echo "<script type='text/javascript'> alert('السلام عليكم') </script>";
   }
 }
 test::Mgs();
?>
```

فى الكود السابق قمنا بالوصول إلى الدالة Mgs بشكل مباشر من خلال كتابة أسم الكلاس ثم المعامل :: ثم أسم الدالة، لكن لا تنسى شي أننا قمنا بإضافة كلمة static لتدل على أن هذه الدالة يمكن الوصول إليها بشكل مباشر ، وبما أننا سوف نستخدمها بشكل مباشر فإن ناتج الدالة سيكون ثابت فى أى كل الأحوال لأننا نستخدمها بشكل مباشر ولا تتبع أي كائن .

وفى النهاية نتمنى التوفيق للجميع

الموقع الداعم
موقع زيروجيت
http://0gate.com

قام بكتابة هذا الكتاب
:المبرمج:
أحمد فتحي
جمهورية مصر العربية

بريد إلكتروني
ahmed_fa@hotmail.com

المـــــــراجع

موقع tutsplus
موقع php