الفصل الخامس
ترتيب الكود البرمجي

تعلمنا في الفصول السابقة أساسيات من أساسيات البرمجة وأعطينا مثال عن الروتين في الحياة اليومية وهو أن تقوم بعمل شيء أكثر من مرة في الحياة اليومية مثل شرب الشاي أو شرب القهوة وغير ذلك، فصلنا هذا يتكلم عن ترتيب الكود ويتكلم تقريباً عن نفس فكرة الروتين اليومي فأنت في حياتك تكرر بعض الأعمال بشكل روتيني

وقد تكون مللت الروتين فأحضرت شيئاً يساعدك على التخفيف من هذا الروتين... فمثلاً عند استخدامك لبرنامج MS Word قد تكون مللت من تنسيق عدة نصوص بطريقة معينة فأنت عند ذلك تقوم بصناعة ماكرو يقوم بفعل العمل الذي كنت تفعله في عدة خطوات بخطوة واحد فقط !!

ولنقل أنك في حياتك اليومية وفي يوم إجازة وقررت أن تقوم بعمل تنظيف شامل (يا إلهي عليك غسيل أطباق الصحون وتنظيف الأثاث وتنظيف الأرضية وترتيب المكتبة وترتيب غرفة النوم و... و... إلخ) عند ذلك فإنك تبحث عن طريقة عملية لكي يتم إنجاز هذه المهمة في أسرع وقت فتقوم بتقسيم هذه المهمة الكبيرة على عده أقسام (التنظيف، الترتيب، الغسيل، .....) ثم تقوم باستدعاء أطفالك وفلذات أكبادك وتقسم على كل واحد منهم مهمة بسيطة يستطيع القيام بها.. هذا التقسيم يسمي في عالم البرمجة بالـfunction (دالة أو وظيفة)

Function

الدالة هي جزء من كود البرنامج يتم تعريفه عن طريق المبرمج ليتم تنفيذ شيء معين بواسطتها، تقوم الدالة بأخذ قيم وتسمى (arguments معطيات) كمدخلات، ثم تقوم بعمل بعض التعديلات على هذه المدخلات وتقوم بإخراج قيمة أخرى في أكثر الأحيان تقوم الدالة بأخذ القيم ووضعها في متغيرات أخرى تسمى بالـ(parameters) لكي يتم إجراء العمليات عليها داخل الدالة وهذه المتغيرات لا

تعمل خارج الدالة أي أنها متغيرات خاصة بالدالة فقط!... في فصولنا السابقة قمنا باستخدام دوال عديدة مثل دوال فرز المصفوفات ودوال إيجاد نوع البيانات، هذه المرة سنقوم ببناء دوالنا الخاصة بنا، ومن صنعنا نقوم بإعطائها المعلومات والبيانات وهي تقوم بإجراء العمليات عليها ومن ثم إخراج الحلول...

تعريف واستدعاء الدوال

لكي تقوم بتعريف دالة فإنك تقوم بكتابة الكلمة function متبوعة باسم الدالة والبارمترات اللازمة والتي سيتم إجراء العمليات عليها بين قوسين ومن ثم تقوم بكتابة الكود اللازم وسط { و }

الصيغه:

```
Function functionname (parameters)
{
function code
}
```

تقوم بكتابه اسم الدالة بدلاً من functionname ثم تقوم بتعريف المتحولات أو المتغيرات parameters ومن ثم تقوم بكتابة الكود الذي سوف يقوم بالمطلوب بين القوسين بدلاً من function code

دعنا الآن نقوم بكتابة دالة من إنشائنا والتي تقوم بإجراء عملية الجمع على متغيرين وسنقوم بتسمية الدالة باسم sumnoraml وهو اسم من تأليفنا ويدل على وظيفة وهدف الدالة ويمكن أن تقوم بتمسية الدالة بأي اسم تريده ولست مجبراً بكتابة اسم معين.

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return $a;
}
?>
```

نقوم في هذه الدالة بإجراء عملية إضافة ١٠٠ على المتغير أو القيمة التي يتم تمريرها.

Return

يجب أن نضعها في نهاية كل دالة، نستخدم هذه الكلمة لكي نقوم بإعلام الدالة أن وظيفتها انتهت وأيضاً نستخدمها إذا كان لدينا أكثر من قيمة ونريد أن نقوم بإخبار الـPHP ما هي القيمة التي سيتم اعتمادها ففي مثالنا هذا أردنا إخبار الـPHP بأن يقوم بأخذ المتغير $a بأنه هو القيمة النهائية مع أنه لو لم نضع المتغير فسيتم اعتباره هو الناتج النهائي لأنه لا يوجد متغير آخر تمت عليه أي عمليات.

الذي أقصده أننا لو كتبنا الكود بالشكل التالي:

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
?>
```

فإنه لاضرر من ذلك لأنه لا يوجد لدينا إلا قيمة واحدة لن يتم اعتماد قيمة غيرها ولكن لو افترضنا أنه لدينا أكثر من قيمة كما في المثال التالي:

```
<?
$b)،Function sul($a
{
$a = $a + 100 ;
$b= $b*100;
return $a ;
}
?>
```

هنا يجب تحديد أي المتغيرين سيكون هو القيمة النهائية للدالة.

شرح الدالة (sumnormal)

تقوم الدالة التي صنعناها بأخذ قيمتين ومن ثم فإنها تقوم بزيادة العدد الذي يتم تمريره ١٠٠

ولكي نقوم بإخراج نتيجة الدالة فإننا ببساطة نستطيع ذلك بإجراء أحد الأمرين echo أو print.

مثال:

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
echo sumnormal(500);
?>
```

لقد قمنا بتمرير رقم بدلاً من المتغير ويمكننا أيضاً تمرير متغير بدلاً من الرقم

مثال:

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
$f=100;
echo sumnormal($f);
?>
```

لاحظ أننا استخدمنا متغير في الدالة (مما يثبت كلامنا في الأعلى أن للدالة متغيرات خاصة بها) وليس معنى ذلك أننا لا نستطيع استخدام متغيرات بنفس الاسم المذكور في الدالة فيمكننا مثلاً كتابة نفس اسم المتغير بدون حصول أي مشاكل كالتالي:

```
<?
Function sumnormal($a)
{
```

```
$a = $a + 100 ;
return ;
}
$a=100;
echo sumnormal($a);
?>
```

يمكننا أيضاً استدعاء دالة بشكل عادي إذا كانت هي تقوم بالطباعة

مثال:

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
print $a;
return ;
}

$a=100;
sumnormal($a);
?>
```

print

يقوم الأمر print بنفس عمل الدالة echo ولا يوجد بينهما اختلاف سوى أن الدالة echo قديمة وهي الأصل أما الدالة print فقد تم إنشاؤها في php4 ولا يوجد أي فرق بينهما إطلاقاً.

مثال:

```
<?
”;أحمدPrint “
?>
```

ويمكننا بها إخراج نتيجة دالة

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
```

```
}
$a=100;

print  sumnormal($a);
?>
```

اين يتم وضع الداله ؟

يمكنك وضع الدالة في أول الكود أو في آخره أي أنه لا فرق بين:

```
<?
//لاحظ أننا قمنا بتعريف الدالة أولاً ثم استدعائها
Function fares($d)
{
print “alfareees@hotmail.com”;
}

; fares($d)
?>
```

وبين:

```
<?
//لاحظ أننا قمنا باستدعاء الدالة أولاً ثم تعريفها
; fares($d)

Function fares($d)
{
print “alfareees@hotmail.com”;
}
?>
```

يمكنك أيضاً عدم وضع متغيرات في الدالة كالتالي:

```
Html_header ()
{
Print “<html><head><title>alfareees</title></head>”;
Return ;
}
```

هذه الدالة تقوم بكتابة الطور الأول من صفحة html لاحظ أننا لم نقم بوضع أي متغيرات أو عوامل أو متحولات (سمها كما شئت).

تمرير القيم إلى الدالة:

هناك نوعين من تمرير القيم

١ - تمرير القيمة مباشرة إلى الدالة (passing by value)

وذلك أن نضع القيمة مباشرة بدون إدراجها في متغيرات.

مثال:

```
<?
Function alfars ($f)
{
$f=$f+$f;
return ;
}
echo alfars(100);
 ?>
```

لاحظ أننا قمنا بإدراج القيمة مباشرة للدالة من غير وضعها في متغيرات.

٢ - تمرير القيمة عن طريق المرجع (passing by reference)

نقصد بهذا أننا نقوم بوضع القمية في متغير أولاً ثم نضع هذا المتغير في الدالة لكي يتم إجراء العمليات عليه مثال:

```
<?
Function alfars ($f)
{
$f=$f+$f;
return ;
}
$r =1000;
echo alfars($r);
 ?>
```

إعداد قيمة افتراضيه للدالة

تستطيع أن تجعل الـPHP4 يقوم بإدراج قيمة افتراضية عند عدم تمرير متغيرات إليه

مثال:

```
<?
Function alfars ($f=40)
{
$f=$f+$f;
return ;
}
echo alfars();
?>
```

إذا لم يتم إعطاء قيمة للدالة فإنها ستفترض أن القيمة هي ٤٠ مباشرة.

أما إذا تم تمرير قيمة أو متغير فإنه سيتم العمل بالقيمة التي تم تمريرها بدلاً من القيمة الافتراضية

مثال:

```
<?
Function alfars ($f=40)
{
$f=$f+$f;
return ;
}
echo alfars(100);
?>
```

مدى المتغيرات [variable scope]

هناك متغيرات محلية (local) ومتغيرات عامة (global)، نقصد بالمتغيرات المحلية التي تكون في داخل الدالة ونقصد بالعامة التي تكون في كود الـPHP بشكل عام.

مثال

```
<?
//هذا متغير عام
$r= “salem”;
function ala($s)
{
//هذا متغير محلي
$s = “progrramer”;
}
echo $r ;
ala($s);
echo $s;
?>
```

مثال:

```
<?
//هذا متغير عام
$r= “salem”;
function ala($s)
{
//هذا متغير محلي
$s = “progrramer”;
}
echo $r ;
$s=10;
echo $s;
?>
```

في المثال الأول استطعنا طباعة المتغير $r ولم نسطتع طباعة المتغير $s لأنه محلى (لا يتم تنفيذه الا داخل الدالة) وعندما نريد طباعته فإننا يجب أن نطبع ناتج الدالة لكي نحصل عليه (أي أننا لا نستطيع طباعته بشكل مباشر)

مثال:

```
<?
//هذا متغير عام
$r = “salem”;
function ala($s)
```

```
{
//هذا متغير محلي
$s = “programmer”;
}
//استطعنا طباعته بشكل مباشر
echo $r ;
ala($s);
//يجب استخدام الداله لكي تتم طباعته
echo ala($s);
?>
```

لاحظ أننا حتى لو قمنا بعملية طباعة المتغير من نفس الدالة فالناتج يكون مختلفاً لأن لكل متغير عالمه الخاص به

لكي نقوم بجعل المتغير الذي بداخل الدالة متغيراً عاماً فيمكننا ذلك بإحدي الطريقتين التاليتين:

الطريقة الأولى:

```
<?
function ala($y)
{
echo $y. “<br>”;
global $s;
$s = "programmer";
return ;
}
$f =10;
ala($f);
echo $s;
?>
```

لاحظ أننا عندما استخدمنا global في داخل الدالة لكي يتم تعريف أن المتغير متغير عام وبعدما قمنا باستخدام الدالة قامت بطباعة المتغير المراد طباعته ومن ثم بعد ذلك قامت بتعريف متغير جديد ($s) وهذا المتغير متغير عام لأننا وضعنا قبله الكلمة global فاستطعنا طباعته بكل سهولة.

الطريقة الثانية:

هي أن نستخدم المصفوفة $GLOBALS التي تستخدم في PHP لتعريف المتغيرات العامة أيضاً

مثال:

```
<?
function ala($y)
        {
echo $y. “<br>”;
$GLOBALS[“s”] ;
$s = "programmer";
return ;
}
$f =10;
ala($f);
echo $s;
?>
```

المتغيرات المستقرة [static variable]

أقصد بالمتغيرات المستقرة هي التي تكون قيمتها ثابتة

مثال:

```
<?
Function  addfares($y)
{
$y;
$y=$y+1 ;
return $y;
}
echo addfares($y);
echo addfares($y);
echo addfares($y);
echo addfares($y);
?>
```

```
<?
Function  addfares($y)
{
static $y;
$y=$y+1 ;
return $y;
}
echo addfares($y);
echo addfares($y);
echo addfares($y);
echo addfares($y);
?>
```

لاحظ عندما عرفنا المتغير بأنه static فإنه يحتفظ بقيمته حتى لو انتهت الدالة.

دوال متداخلة

يمكننا عمل تعشيش للدوال مثلما كنا نفعل مع بناء القرارات والتكرارات

مثال:

```
<?
Function sum($sa)
{
   $sa=$sa-1;
function goadd ($r)
{
$r = $r+$r;
return $r;
}
$sa= goadd ($sa);
return $sa;
}
echo sum (15);
?>
```

في مثالنا هذا لدينا دالتين الدالة الأولى هي sum والدالة الثانية هي goadd

وظيفة الدالة الأولى هي أن تقوم بالإنقاص من العدد الذي يمرر إليها واحد ثم تقوم بتطبيق دالة داخلية فيها هي goadd تقوم بزيادة العدد على نفسه.. ومن ثم قمنا بنداء الدالة الأولى (لأنها هي الأساس الذي يوجد به الدوال الداخلية) وطباعة قيمتها.

اشتمال الملفات (include files)

قد يكون لديك في برنامجك متغير متكرر في أكثر من صفحة أو رسالة خطأ معينة أو تريد إدرج نص كبير الحجم في صفحات متعددة.
هنا يمكنك اشتمال ملفات في داخل ملفات الـPHP. هذه الملفات قد تحتوي على نصوص أو كود html أو كود PHP.

إن الصيغة التي تستخدمها لاشتمال الملفات هي:

```
Include (filename);
```

مثال:

قم بفتح ملف نصي واكتب فيه ما تشاء ثم احفظه باسم a.txt
قم بإنشاء ملف php واكتب فيه ومن ثم احفظه باسم b.php

```
<?
Include (“a.txt”);
?>
```

انقلهما إلى مجلد السيرفر.. شغل ملف الـb.php وانظر النتيجة.
يمكنك أن تقوم بإنشاء ملف PHP وتحتفظ فيه بجميع الـfunction المطلوبة لبرنامجك وعند إرادتك لاستخدام أي واحدة منها تقوم فقط باشتمال الملف ومن ثم استدعائها.

داله تلوين الكود:

هل رأيت مواقع تقوم بتلوين الكود بشكل مذهل مثل موقع zend ؟.... الأمر بسيط كل ما عليك أولاً

قم بوضع الكود في ملف نصي وسمه بأي اسم (مثلاً file.txt) وبعد ذلك قم باستخدام الدالة

Show_source

مثال:

```
<?
show_source ("file.txt");
?>
```