مكتبة الطـــارق الاكترونية

PHP

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

http://www.t0010.com

كتاب شامل لتعليمك لغة البرمجة العالمية
بي اتش بي من الصفر وحتى الاحتراف

Designer by : aLTar3q

بسم الله الرحمن الرحيم

لغة (PHP)

تتميز لغة PHP بالكثير من الخصائص التي جعلتها الخيار الأمثل لمبرمجي الويب في العالم :

السهولة

تعتبر لغة PHP من أسهل لغات البرمجة تعلما، فهي تريحك من جميع تعقيدات إدارة الذاكرة وتعقيدات معالجة النصوص الموجودة في C من جهة، والكثير من الضعف الموجود في بينية وتصميم لغة البرمجة Perl من جهة أخرى.

تمتلك لغة PHP بنية وقواعدا ثابته وواضحة جدا، معظم قواعد اللغة مأخوذة من كل من C و Java و Perl لصنع لغة برمجة عالية السهولة والسلاسة دون فقدان أي من القوة في اللغة، يفيدك ذلك إذا كنت تعلم أي شيء عن لغات البرمجة الأخرى مثل Visual Basic أو C أو Java حيث ستجد دائما بأنك تفهم مواد الدورة بسرعة، وستكتشف كيف تقوم PHP بتسهيل أصعب الأمور وإذلال العقبات التي تواجه المبرمج حتى يتفرغ تماما للإبداع فقط، كل ما تفكر به تستطيع تنفيذه بلغة PHP.

السرعة

لغة PHP من اللغات المعروفة بسرعتها العالية في تنفيذ البرامج، وخاصة في الإصدارة الرابعة من المترجم، حيث تمت كتابة مترجم PHP من الصفر ليعطي أداءا في منتهى الروعة، كما أن لغة PHP مصممة أصلا كنواة لمترجم، بحيث يمكن أن تضع هذه النواة في عدة قوالب أو أغلفة لتعمل مع التقنيات المختلفة، فيمكنك تشغيل مترجم PHP كبرنامج CGI مثلا، ولكن الأفضل هو إمكانية تركيب مترجم PHP على مزود IIS في صورة وحدة إضافية تضاف إلى المزود عن طريق دوال ISAPI، وتوجد نسخة أخرى منه تركب على مزود Apache أيضا في صورة وحدة خارجية، وتوجد أيضا نسخة مخصصة للدمج مع شفرة مزود Apache بحيث تصبح جزءا من برنامج Apache نفسه، وهي الطريقة الأكثر استخداما الآن في مزودات الويب التي تعمل على أنظمة UNIX وهي الطريقة التي تعطي أفضل أداء لمترجم PHP، حيث يصبح المترجم جزءا من المزود، وبالتالي فإنه سيكون محملا في الذاكرة بانتظار صفحات PHP ليقوم بترجمتها وعرضها للزوار مباشرة دون التأخير الإضافي الذي تتطلبه برامج Perl/CGI مثلا حيث يجب أن يتم تشغيل مترجم Perl مع كل زيارة للصفحة لترجمة الصفحة، ثم يتم إغلاق المترجم، ثم استدعاءه مجددا عند الزيارة الثانية وهكذا، وهذا يشكل فارقا كبيرا في المواقع ذات الضغط العالي بالذات، ويكون استخدام PHP حلا أفضل بكثير.

المزايا

يأتي مترجم PHP لوحده محملا بعدد هائل من الدوال الجاهزة الاستخدام في جميع المجالات، من دوال المعالجة الرياضية والحسابية إلى دوال الوصول إلى قواعد البيانات ومزودات FTP، توفر لك دوال PHP مثلا وصولا إلى مزودات البيانات MySQL و PostgreSQL و MS SQL و Oracle وغيرها من مزودات قواعد البيانات، وهنالك أيضا مجموعة من الدوال لمعالجة ملفات XML، ودوال أخرى لإرسال واستقبال الملفات عن بعد باستخدام بروتوكول FTP، وهنالك مجموعة من الدوال لمعالجة وإنتاج الصور ديناميكيا وملفات Flash ديناميكيا، ناهيك عن جميع الدوال الخاصة بمعالجة النصوص والمصفوفات.

التوافقية

كما قلنا سابقا، فعلى الرغم من أن هنالك الكثير من نسخ PHP التي يعمل كل منها في بيئة مختلفة، إلا أنها جميعا تشترك في النواة الأصلية التي تقوم بالمعالجة الحقيقة لملفات PHP لذا فإن جميع مترجمات PHP تتصرف بنفس الطريقة فيما يتعلق بتنفيذ السكريبتات، فإذا كان السكريبت الذي عملته يعمل على نظام Windows مع مزود IIS فيجب أن يعمل دون الحاجة لأية تغييرات عند نقله إلى مزود Apache، بالطبع تظل بعض الأمور البسيطة جدا التي يوفرها بعض المزودات دون غيرها، ولكن جميع البرامج التي كتبتها منذ أن بدأت تعلمي للغة إلى الآن تعمل على جميع المزودات دون الحاجة لأي تغييرات، إضافة إلى ذلك فإن التغييرات التي حدثت باللغة الأساسية من الإصدارة الثالثة إلى الرابعة قليلة جدا، وأغلب التغييرات كانت في البنية التحتية للمترجم.

الحماية

يوفر PHP الكثير من المزايا المتقدمة، ولكنه يوفر لك الطرق المناسبة لوضع الحدود على هذه المزايا، فيمكنك التحكم بعدد الإتصالات المسموحة بقاعدة البيانات مثلا، أو الحجم الأقصى للملفات التي يمكن إرسالها عبر المتصفح، أو السماح باستخدام بعض الميزات أو إلغاء استخدامها، كل هذا يتم عن طريق ملف إعدادات PHP والذي يتحكم به مدير الموقع.

قابلية التوسع

يمكنك توسعة مترجم PHP بسهولة وإضافة الميزات التي تريدها إليه بلغة C، وحيث أن الشفرة البرمجية للمترجم مفتوحة فإنك تستطيع تغيير ما تريده مباشرة لتحصل على النسخة التي تناسبك من المترجم، ويمكنك أيضا عمل الوحدات الإضافية التي تركب على المترجم لزيادة ميزاته والوظائف المبيته فيه، وفي قد قام فريق تطوير مترجم PHP مسبقا بعمل هذه المهمة وتحويل كمية ضخمة من المكتبات المكتوبة بلغة C إلى مكتبات مخصصة لتضاف إلى المترجم، ومنها حصلنا على جميع الميزات التي تحدثنا عنها مثل الوصول إلى قواعد البيانات ومعالجة ملفات XML.

تاريخ PHP

بدأت PHP كمكتبة من الدوال تضاف على لغة Perl لتسهل عمل برامج CGI بلغة Perl، وبعد أن تلقى Rasmus Lerdof بعض الاقتراحات بتحويلها إلى مترجم بسيط، قام بعمل ذلك المترجم وطرحه على الإنترنت وسماه PHP أو Personal Home Pages أي الصفحات الشخصية، فقد كان عبارة عن نسخة مصغرة من Perl مع بعض الميزات الإضافية للويب، ثم أضاف إليه دعما لنماذج HTML وسماه PHP2/FI، فقام مجموعة من المبرمجين بالعمل على مترجم PHP وأضافوا إليه واجهة تطبيقات برمجية API لتسهيل عملية توسعته فأصبح لدينا PHP 3، بعد فترة من الزمن قامت شركة Zend للتقنيات بعمل مترجمها الخاص للغة والذي سمي zend أيضا، وقد اتصف هذا المترجم بالسرعة العالية وقدراته المحسنة، وجمع مع مكتبات PHP الأخرى لتكوين نواة المترجم PHP، مترجم PHP الآن مقسم على قسمان: المترجم zend ويتم تطويره على مزودات CVS الموجودة في موقع zend والقسم الثاني يسمى PHP وهو عبارة عن المكتبات والدوال الأساسية التي تأتي مع البرنامج، يقوم مترجم zend بقراءة الملفات ومعالجتها والتعامل مع المتغيرات وتنفيذ البرنامج وتوفير واجهة تطوير

للتطبيقات API لتوسعة اللغة، أما PHP فتحتوي الآن على مكتبات مكتوبة بلغة C ومتوافقة مع واجهة التطبيقات التي يوفرها مترجم zend، وبالتالي يعمل القسمان معا لتكوين مترجم PHP، وعندما تزور موقع PHP الرسمي الآن وتحصل على مترجم PHP جاهزا أو تحصل على الشفرة البرمجية الخاصة بك، فإنك تحصل على كل من مترجم zend ومكتبات PHP معا.

تطور PHP تطورا مفاجئا في الفترة الأخيرة، وتشير إحصائيا Net Craft إلى أن مترجم PHP هو أكثر وحدات مزود Apache انتشارات على الإنترنت، كما أن مترجم PHP مركب على حوالي مليوني مزود ويب على الإنترنت.

بنية ملفات PHP

ملفات PHP هي ملفات نصية بسيطة، تشبه في تركيبها ملفات ASP وملفات HTML بشكل عام، يتكون ملف PHP من قسمان، قسم HTML وقسم PHP، الملف بالصورة الطبيعية عبارة عن ملف HTML عادي، ولكنك تستطيع تحديد أجزاء معينة من الملف ليخرج فيها الملف من وضعية HTML إلى وضعية PHP، لإخراج الملف إلى وضعية PHP توجد عدة طرق :

1 - استخدام زوج الوسوم <؟php و ؟> كالتالي :

```
<?php
echo 'This is PHP output!';
?>
```

2 - استخدام زوج المختصر <؟ و ؟> وهو يستخدم بنفس الطريقة السابقة ولكنه يكون بدون الكلمة php في وسم البداية، هذا النوع من الوسوم يحتاج إلى كمية أقل من الكتابة بالطبع، ولكنه يتعارض مع وسوم xml، لذا يقوم البعض بإغلاق ميزة الوسوم القصيرة حتى لا يحصل هذا التعارض (يمكنك اغلاق هذه الميزة بسهولة عن طريق ملف إعدادات PHP).

3 - استخدام زوج الوسوم ASP، وهو من اسمه زوج الوسوم المستخدم في ملفات ASP وهما <% و %>، ميزة وسوم ASP لا تكون فعالة بشكل قياسي ولكنك تستطيع تفعيلها عن طريق ملف إعدادات مترجم PHP.

4 - الطريقة الأخيرة هي استخدام زوج الوسوم التالي :

```
<script language="php" >
    echo 'This is PHP output!';
</script>
```

ولكن هذه الطريقة غير مستخدمة الآن، حيث أنها تصعب عملية التمييز بين شفرات PHP وباقي ملف HTML، وكذلك بالنسبة لبرامج كتابة ملفات HTML التي تعطي تلوينا للشفرة فأغلبها لا يتعرف على هذا النوع من الشفرة ويعتبره جزءا من ملف HTML الاعتيادي.

أفضل الطرق السابقة للتحويل إلى وضعية PHP هو استخدام زوج الوسوم الأول بالطبع، حيث أنه الأكثر استخدامها، ولا يحتوي على أية تعارضات كما أنه يعمل على جميع مترجمات PHP مهما كانت إعداداتها، ولهذا السبب سنستخدمها في جميع الأمثلة التي ستجدها في هذه الدورة.

كتابة ملفات PHP

ملفات PHP هي ملفات نصية بسيطة تماما كما هي ملفات HTML، يمكنك كتابة سكريبت PHP بأي برنامج كتابة نصوص يتيح لك كتابة الملفات النصية البسيطة Plain Text مثل Notepad على النظام ويندوز، ولكن أغلبية مبرمجي PHP يستخدمون أدوات أخرى تسهل عليهم عملية البرمجة عن طريق تلوين الشفرات البرمجية، وتسهل عملية البحث عن الملفات واستبدال المقاطع من عدة ملفات في نفس الوقت، مثل HomeSite، على الرغم من أنك لن تحتاج إلى الكثير من هذه الميزات إلا أن استخدام Notepad في عمل ملفات PHP يعتبر أمرا صعبا جدا وخاصة في الملفات الضخمة حيث أن

Notepad لا تتيح فتح الملفات الكبيرة، والمشكلة الأكبر هي أنها لا توفر ترقيما للأسطر، فإذا ظهرت لك رسالة الخطأ تشير إلى وجود خطأ في السطر 53 فلن تستطيع معرفة السطر المطلوب في Notepad إلا إذا قمت بالعد يدويا من السطر الأول وحتى 53 .. حسنا ماذا لو كان الخطأ في السطر 652، يمكنك البدأ بكتابة سكريبتاتك بالبرنامج المتوفر الآن إلى أن تحصل على برنامج آخر، يمكنك بالطبع فتح ملفاتك بأي محرر نصوص، فإذا كتبتها باستخدام Notepad فهذا لا يعني بأنك ملزم باستخدام Notepad في جميع ملفاتك أو حتى في هذا الملف.

لعمل ملف PHP الآن قم بفتح محرر النصوص الذي اخترته وابدأ بكتابة الصفحة التي تريدها، ولا تنسى إحاطة شفرات PHP بالوسوم الخاصة بها، ثم احفظ الملف في أي مكان في دليل مزود الويب الخاص بك وأعطه الإمتداد المناسب .php أو .php3 حسب إعدادات مزودك، ثم قم بزيارة الصفحة باستخدام المتصفح وستجد الصفحة وقد تمت ترجمتها وعرضها عليك.

تذكر بأنك يجب أن تزور الصفحة مرور بمزود الويب، ولا يمكنك عرض الصفحة عن طريق فتحها كملف خارجي، على سبيل المثال، إذا كان الدليل الجذري لصفحات مزودك هو : `C:\httpd\`

وقمت بعمل صفحة أسميتها test.php في ذلك الدليل، يجب أن تقوم الآن بتشغل مزود الويب وزيارة الصفحة على العنوان http://localhost/test.php، إذا قمت باستخدام الأمر Open من القائمة File في المتصفح لفتح الملف C:\httpd\test.php فلن ترى صفحة PHP مترجمة، وسترى شفرة PHP فقط.

تدريب

قم بتنفيذ ملف PHP التالي :

```
This is the normal html page.<br>
<?php
      echo "This is inside PHP<br>";
      echo "Hello World!<br>";
?>
```

ما الذي تشاهده عند تنفيذ البرنامج السابق؟ من المفترض أن تشاهد الخرج التالي :

```
This is the normal html page.
This is inside PHP
Hello World!
```

ها قد انتهيت من كتابة برنامجك الأول بلغة PHP، لا تقلق إذا لم تفهم أي شيء فيه، سنتعلم الآن كيفية استخدام المتغيرات والعبارات بلغة PHP.

لنكتب سكربتاً بسيطاً (فاتح شهية) :

```
<”html dir = “rtl>
التحية لدي أهل الإسلام هي
؟>
 Echo (”السلام عليكم ورحمة الله وبركاته“)
<؟
<html/>
```

قم بحفظ الملف باسم echo.php
ستعرض علينا عباره مكتوب فيها

التحية لدي أهل الإسلام هي السلام عليكم ورحمة الله وبركاته

شي بسيط أليس كذلك ؟

يتكون كود الـphp من نصوص و كود و علامات ولغة html وقد لاتحتوي على نصوص html .
لكي يعمل الكود يجب أن يكون إمتداد الملف php أو بأي إمتداد من إمتدادات الـphp
مثلاً php3 و phtml
--
عندما تطلب صفحة في الإنترنت فإنك تجري اتصالاً مباشراً مع السيرفر هذه العملية تدعي request للسيرفر (يعني طلبية للسيرفر) يقوم السيرفر بتفسير طلبك والبحث عن الصفحة المطلوبة ويرسل اليك الصفحة المطلوبة كجزء مما يسمي response (استجابة) لمستعرض الانترنت لديك يقوم بعدها المتصفح لديك بأخذ الكود الذى ارجع إليه ويقوم بتجميعه (compile) لكي يصبح صفحة صالحة للعرض هذه العملية التي حصلت تشبه نظرية العميل للخادم(client to server) بحيث أن المتصفح هو العميل والخادم هو السيرفر .
الخادم يقوم بعملية تخزين وترجمة وتوزيع البيانات بينما يقوم العميل (مستعرض الانترنت لديك) بالعبور الى السيرفر واحضار البيانات

بروتوكولات الانترنت :

لانريد هنا أن نذهب إلى التكلم عن تاريخ انترنت العتيق ، النقطة المهمة هي الشبكة المربوطة بنقاط nodes الانترنت صممت لكي تقوم بالحفاظ على المعلومات لكي يتم نقلها من مكان إلى آخر وهي تستخدم مجموعة من البروتوكولات مثل Tcp/Ip لكي يتم نقل البيانات عبر الشبكة .

بروتوكول Tcp/Ip

من مميزات هذا البروتوكول أنه بإستطاعته إعاده تمهيد طريقه للبيانات إذا تم خلل في نقطة أو مكان أثناء نقلها ويتم ذلك بسرعة شديدة.عندما يطلب المستخدم من المستعرض أن يجلب له صفحة من الانترنت فإن المستعرض يجلب هذه الأوامر باستخدام بروتوكول يدعي بروتوكول التحكم في نقل البيانات TCP هذا البروتوكول هو بروتوكول نقل للبيانات وهو يضمن أن البيانات قد تم إرسالها ووصولها بشكل صحيح .

قبل أن يتم إرسال البيانات عبر الشبكة يجب عنونتها والبروتوكول الذي يقوم بعنونة البيانات يدعي HTTP يقوم هذا البروتوكول بوضع عنونة للبيانات لكي يعرف البروتوكول TCP أين سينقل البيانات (فهو لايستطيع نقل البيانات إذا لم يكن لها هدف أو مكان) يستخدم البروتوكول HTTP عن طريق الويب في عملية نقل البيانات من كمبيوتر إلى آخرعندما ترى الصفحة متبوعة بـhttp:// فانك تعلم مباشرة أن الانترنت يستخدم البرتوكول HTTP لإحضار هذه الصفحة يمكنك أن تاخذ صورة بأن الـTCP عبارة عن ساعي بريد الذي يقوم بإيصال رسالة ، هذه الرسالة فيها طابع بريد وعنوان وهو مانسميه بالـHTTP .

يتم تمرير الطلب من المستعرض إلى ملقم أو سيرفر الويب وهو مايعرف بـ HTTP request ويقوم السيرفر برؤية مستودع البيانات لديه لكي يحصل على البيانات المطلوبة فإذا وجد الصفحة في المستودع قام بإرسالها على شكل حزم الى الجهة التي قامت بالطلب باستخدام بروتوكول TCP ويعنون هذه الحزم لمستعرض الانترنت لديك باستخدام بروتوكول http (ننبه دائما الى أنه يرسلها على شكل حزم لكي تعرف السبب عند عدم ظهور صفحة ويب كاملة أن هناك حزمة لم ترسل بشكل جيد) ولكن إذا لم يجد السيرفر الصفحة المطلوبة فانه يقوم بإرسال صفحة تحتوي على رسالة خطا 404 وهذه الصفحة التي أرسلت من ملقم الويب الى المستعرض لديك تسمي HTTP response .

بروتوكول الـ HTTP

رغم ما أخذناه من معلومات كثيرة وقصص كثيرة تشبه قصص ألف ليلة أو حكايات الأطفال إلا أنه رغم ذلك يفوتنا الكثير من التفاصيل في هذا الموضوع لذلك دعنا نغوص قليلاً في التفاصيل عن بروتوكول HTTP بشكل خاص.

عندما تقوم بعملية طلب لصفحة من السيرفر هناك أمور إضافية ترسل مع عملية الطلب http request غير الـURL وهي ترسل كجزء من http request .
نفس الموضوع مع الـhttp response هناك أمور أخرى تصل معه كجزء منه .

الكثير من هذه المعلومات تولد تلقائياً في رسالة الـHTTP ولايقوم المستخدم بالتعامل معها مباشرة ، إذن لايحتاج أن تقلق نفسك بشأن هذه المعلومات إذا أنت لم تنشأها في الأصل ويجب أن تأخذ أيضا في معلوماتك أن هذه المعلومات ترسل كجزء من الـHTTP request والـHTTP response لأن سكربت الـPHP الذي نصنعه يمنحنا تحكماً إضافيا بهذه المعلومات .

كل رسائل الـHTTP تأخذ تنسيقاً معيناً سواء كانت Request أو Response . نستطيع أن نقوم بتقسيم هذا التنسيق إلى ثلاثة أقسام :

1 - Request/response line
2 - Http header
3 - Http body

المحتوي من هذه الأشياء الثلاثة يعتمد على نوع الرسالة إذا كانت HTTp Request أو HTTP response لذلك سنتكلم عنهم بتعمق أكثر .

Http Request

يجب أن يحتوي الـrequest على الأقل الـrequest line (سطر الطلب) والـHOST .
يرسل مستعرض الانترنت طلبية (HTTP request) إلى ملقم الويب تحتوي على التالي :

1- The Request Line
السطر الأول من كل طلبية (http request) هي Request Line الذي يحتوي على ثلاثة أنواع من المعلومات :

أ - أمر HTTP وهو مايعني بـ method .
ب - المسار من السيرفر إلى المصادر المطلوبة (صفحات الانترنت) المطلوبة من قبل العميل (المستعرض)
جـ - إصدارة الـHTTP .

إذن كمثال على الـ Request Line أنظر إلى السطر التالى :
GET /testpage.htm HTTP/1.1

الـmethod يخبر السيرفر كيف يتعامل مع الطلب هناك ثلاثة أنواع شائعه من الـmethod

2- HTTP Header
البت الثاني من المعلومات هو الهيدر HTTP Header .الذي يحتوي على تفاصيل أو وثائق عن العميل مثل نوع المتصفح (نتسكيب أو إكسبلور) الذي قام بطلب الصفحة والوقت والتاريخ والإعدادات العامة
الـHTTP Header يحتوي على معلومات نستطيع تقسيمها الى ثلاث فئات وهي :
أ - عامة GENERAL : تحتوي معلومات إما عن العميل أو السيرفر ولاتخصص إلى فرد أو مجموعة .
ب - شخصية Entity : تحتوي على معلومات عن البيانات التي أرسلت بين المتصفح والسيرفر .
جـ - مطلوبة Request : تحتوي على بيانات عن إعدادات العميل والأنواع المختلفة المقبولة من البيانات .

وهذا مثال :
* / * :Accept

.Accept language: Arabic-KSA
.Connection: Keep –Alive
Host : http://www.arabbuielder.com
Referer: http://www.arabbuielder.com/index.php?something=132
(.....؛User –Agent :Iexploer (win98

مثلما ترى الـ HTTP Header عبارة عن إعداد يتكون من عدة سطور كل سطر يحتوي على قيم معينة .

هناك عدة سطور تشكل الـ HTTP header وأكثرها إختياري ، يقوم الـHTTP بالإخبار عن إنتهاء معلومات الـ header بترك سطر فارغ (وهذا يكون في الـHTTP1.1) .

3- The HTTP Body :
إذا تم استخدام الأمر POST في الـHTTP Request Line عندها يقوم الـ HTTP بطلب المعلومات التي ارسلت في الـbody الى السيرفر .

Http Response

يرسل من السيرفر إلى المستعرض ويحتوي على ثلاثة أشياء :
1- the Response Line
2 - http header
3 - Http Body

1 - The Response Line
الـ response line يحتوي فقط على نوعين من المعلومات :

1 - رقم إصدارة الـHTTP .
2 - شفره أو كود الـhttp request التي تقوم بتحديد إذا كان الـrequest ناجحاً أم فاشل .

مثال :

HTTP/1.1 200 OK

في هذا المثال يقوم الـresponse line بإرجاع القيمه 200 متبوعة بالكلمة OK هذه تشكل وتشير إلى نجاح الـ request ويكون الـresponse يحتوي على الصفحة المطلوبة والبيانات من السيرفر . ومثال آخر هو الشفرة 404 عندما تقوم بطلب صفحة ويفشل السيرفر في الحصول عليها .

2 - HTTP Header

الـ response header يعتبر مشابه request hader الذي ناقشناه في الأعلى .وتنقسم المعلومات التي فيه أيضا إلى ثلاثة أنواع :
أ - عامة GENERAL : معلومات عن الـ client أو السيرفر ولاتخصص إلى واحد منهما .
ب - شخصية Entity : يحتوي على معلومات عن البيانات التي يتم ارسالها بين السيرفر والعميل .
جـ - الإجابة Response : يحتوي معلومات عن السيرفر الذي قام بإرسال الرد وكيفية تعامله ومعاجلته للرد (Response) .

كما قلنا سابقاً ، يتكون من عده سطور ويتم وضع سطر فارغ للإعلام عن إنتهاء الهيدر .

مثال :

```
HTTP/1.1 200 OK -the satus line
Date: Mon; 1st Nov 1999, 16:12:23 GMT -general header
Server : Apache/1.3.12 (Unix) (SUSE/Linux) PHP/4.0.2 -the response
Last-modified: Fri, 29 Oct 1999, 12:08:03 GMT -Entity Header
```

السطر الأول ناقشناه والسطر الثاني مفهوم من غير شرح ، السطر الثالث يقوم بتحديد البرنامج تبع السيرفر ونوعه ونظام التشغيل القائم عليه والسطر الأخير يقوم بتعريف آخر وقت تم فيه تعديل أو تجديد الصفحة .

ملاحظة : قد يحتوي الهيدر على أكثر من هذه المعلومات أو معلومات مختلفة وهذا يعتمد على نوع الشي المطلوب من السيرفر .

3 - Http Body

إذا تم معالجة الطلب بنجاح ، فإن الـHTTP response Body يحتوي على كود الـHTML ويقوم مستعرض الانترنت بتفسيرها وتحويلها إلى الصفحة النهائية التي تراها .

أين سكربت الـPHP من ذلك كله ؟

أصبح الآن لدينا مفهومية جيدة عن طريقة إرسال المستعرض طلب صفحة من السيرفر وكيفية استجابة السيرفر لهذا الطلب .

تكلمنا عن أن سكربت الـphp يتكون من ثلاثة أشياء : نص وكود php وكود html ، لانستطيع وصف الـhtml بأنها لغة برمجة بشكل جيد ونستطيع أن نقول أن الـphp لغة سكربتات Scripting Language لأنها تضيف قدرات html عليها مثل الجداول والفريمات بكود html بداخل كود الـphp هناك لغات تسمي لغات سكربتات قد تكون متآلفاً معها مثل الجافا سكربت والفجول بيسك سكربت بإستثناء أن

الفرق بينها وبين الـphp هو أن الـphp لغة تعتمد على جهة المزود أي السيرفر ويمكنك تخصيص المتصفح الذي يستعرضها .
تجعلنا الـhtml نضمن سكربتات الـphp فيها ضمن قواعد لذلك لكي نستطيع تشغيلها ولكننا لاننسي أن إمتداد الملفات يظل كما هو php أو php3 بدون تغير فيه لكي يتم إرسال السكربت الى مكتبة الترجمة (scripting engine) التي تقوم بترجمة السكربت إلى html (كأنك تترجم من عربي لإنجليزي أو العكس)

مفهوم الـparsing و الـExecution :

ممكن أن نقسم عملية الترجمة الذي يقوم بها سيرفر php إلى قسمين أو عمليتين :
العملية الأولى : هي أن السيرفر يقوم أولا بفحص قواعد اللغة وهذا لايضمن أن السكربت صحيح مائة بالمائة ولكنه تدقيق في الأوامر وقواعد اللغة وهذا مايسمونه بالـParsing
العملية الثانية : هي تنفيذ السكربت بعدها وإخراجه على شكل كود html وهذا مايسمي بالـ Execution .

بقي أن نقول أمراً معروفاً وهو أن السكربتات نوعين :

1 - وهو ماينفذ من جهة المزود
Server –Side scripting
2 - ماينفذ من جهة المستعرض (صفحة انترنت) .

التعليقات

ما رأيك إذا كنت في شركة وكان معك أكثر من مبرمج وأردتم تصميم برنامج ، إإذن قد تحتاجون لتنظيم العمل وتعديله لذا من اللازم أن تقوم بعمل توضيح لفائدة الكود الذي كتبته كي يسهل فهمه عليهم وإضافة تعديلات مناسبة ، إذن التعليقات تستخدم في الإفاده عن شرح الأكواد أو إضافة معلومات لاتستعمل إلا كتوضيح أو أي شي آخر .

يمكنك عمل تعليق من سطر واحد كالتالى :

```
<?
//هذا تعليق لافائدة له له اي معني
?>
```

مثال آخر :

```
<?
//هذه الداله تقوم بطباعه الكلمه تعليق
Echo "تعليق";
?>
```

وأيضا يمكنك استخدام تعليق من أكثر من سطر كالتالي :

```
<?
/* تعليق يتكون من
 اكثر من سطر بعلامة السلاش والنجمه
*/
?>
```

المتغيرات

ماهي المتغيرات ؟

أبسط تعريف يمكن أن نقوله عن المتغير هو أنه مساحة من الذاكرة تستخدم لتخزين المعلومات ويتم التحكم فيها عن طريق المبرمج في الـPHP ، المتغيرات تبدأ بعلامة الـ$ ولكي تقوم بإدخال قيمة في المتغير فإنك تستخدم المعامل (=) إذن لكي تقوم بإنشاء متغير يحتوي على قيمة يمكنك القيام بذلك كالتالى :

```
$alfares = "How Are You Every Body?";
$اسم_المتغير = قيمه ;
```

لاحظ أن السطر السابق يتكون من خمسة أشياء :

1 / المتغير وهو alfares
2 / وقبله علامة الـ$ لكي يعرف مترجم الـPHP أنه متغير
3 / المعامل (=)
4 / الفاصلة المنقوطة (;)
5 / القيمة وهي How Are You Every Body? وهي القيمة الموجودة في المتغير أو التي اقترحناها للمتغير أو التي وضعناها فيه (لأن الذي اقترح القيمه هو أنت (مبرمج الـphp))

ملاحظات :
1- اسماء المتغيرات حساسة لحالة الأحرف إذا كانت كبيرة وصغيرة

```
<?
$Ahmed  = "salem";
$ahmed = "slmoon";
echo $ahmed;
echo  $Ahmed;
?>
```

المتغيرين الذين بالأعلي مختلفين بسبب حالة الأحرف.

2 - يمكنك استخدام المعامل (_)

$First_name

3 - يمكنك استخدام ألف حرف في تسميه المتغيرات (وفي الواقع هي غير محدده) .

علامات التنصيص

وهذه نقطة مهمة وهي لماذا وضعنا علامات التنصيص هذه ؟ فالإجابه تكون هي أن القيمة التي وضعناها حرفية أي تتكون من نصوص وهناك أنواع للمتغيرات وعلى ذلك سنفصل ونقول

هناك انواع للبيانات وهي :

1 - strings (حروف)

```
$Exa = "Just An Example";
$Exa2 = "2.5";
$Exa3 = "2";
```

2 - Integer (ارقام)

```
$Exam = 5;
```

3 - Double (ارقام ذات فواصل)

```
$num= 5.4
```

4 - array
ياتي تفصيلها فيما بعد

5 - objects
تفصيلها في دروس اخري

6 - Unknown .
ياتي تفصيلها في درس اخر .

المتغيرات لايتم تعريف نوعها من قبل المبرمج إنما مترجم الـPHP يقوم بالتعرف عليها لكي يتم إتمام العمليات المختلفه عليها .

البيانات الحرفيه /

في الـPHP أي قيمة تكون بين علامتي تنصيص عادية أو علامة تنصيص مفردة يعتبرها الـPHP قيمة حرفية
أمثلة :
"هذا النص بين علامتي تنصيص عاديه اومزدوجه"
'هذا النص بين علامتي تنصيص مفرده او وحيده'

يجب أن يبدأ النص وينتهي بنفس علامة التنصيص ، وإلا فلن يتعرف الـPHP على القيمة الحرفية أو على النص .

```
<?
$d="غلط '
echo "خطا '
?>
```

لايمكنك أيضاً أن تقوم بوضع علامة تنصيص من نفس النوع التي تستخدمه القيمة الحرفية في وسط العبارة الحرفية أو النص

```
<?
$variable = "هذا النص "خطابسبب وجود علامة في النص من نفس النوع";
?>
```

وتصحيحه

```
<?
$variable  = "هذا النص 'صحيح";
?>
```

وأيضا مثال آخر

```
<?
$r = "This is"BAD"; // خطأ
$t = "This is 'good"; // صحيح
?>
```

أما إذا كنت مصراً على ذلك أو تحتاج إليها في عمليات ضرورية (كما سوف نري فيما بعد حاجتنا إليها في صناعة النماذج) فيمكنك وضع معامل (\) قبل علامة التنصيص .
لكي تعمل معك بكل سهولة .

مثال :

```
<?
$u = "This Only An \" Example\" To Make You Understand Nothing";
?>
```

طيب ما رأيك لو أردنا أن نطبع المعامل (\) بنفسه ؟
الحل هو أن نتبعه بمثله ، وبالمثال يتضح المقال :

```
$file = "c:\windows\system.ini";
echo $file;    // النتيجه c:windowssystem.ini

$file = "c:\\windows\\system.ini";
echo $file;    // النتيجه c:\windows\system.ini
```

يمكنك الجمع بين أكثر قيم المتغيرات في متغير واحد عن طريقة الـ(.)

```
<?
$first = "منتدي" ;
$last = "المطور العربي";
$fullname = $first. $last
Echo $fullname ;
//ولكننا نريد وضع فراغ بين الكلمتين
$fullname=  $first . ' ' . $last  ;
Echo $fullname ;
?>
```

وأيضا يمكننا أن ضيف إلى متغير قيمة متغير آخر :

```
<?
$f="I Love M" ;
$k= "y Country" ;
//إضافه القيمة الى المتغير
$f = $f . $k;
echo $f;
?>
```

```
<?
//تقريباً نفس العملية
 $f="I Love M" ;
$k= "y Country" ;
$f.=$k;
echo $f;
?>
```

الارقام

العدد الفردي والمزدوج
الاختلاف المعروف لدي أنا حتى الآن هو أن الفرق بينهما هو الفاصلة العائمة (والله حتي اعطاءها هذا الاسم يجعل الواحد يشعر بالاحباط والخوف)
لاحظ أننا لا نستخدم علامات التنصيص وذلك ليعرف الـPHP أنها بيانات رقمية قد نستخدمها في عمليات حسابية معقدة ويمكننا تطبيق عمليات حسابية بسيطة عليها إذا كانت حرفية .

```
//هذا عدد فردي
$j=2
//هذا عدد مزدوج
$h=4.5
```

العمليات الحسابيه

هي مثل الجمع والطرح والضرب والقسمة وهي مرتبة كالتالي :
أولاً /الأقواس
ثانياً /الضرب ثم القسمه .
ثالثاً / الطرح ثم الجمع

```
<?
Echo 5*2/5;
Echo 5*(2/5) ;
?>
```

مثال آخر :

```
<?
Echo 5-6+9 ;
?>
```

مثال لعملية حسابية نستخدم فيها متغير حرفي

```
<?
$W="2L";
$E= 2;
$F = $W * $E;
echo $W .' ' .$E .' ' .$F;
?>
```

مثال لعملية أخري لكنها لم تعمل وعليك استنباط السبب بنفسك (هاه طل زين) :

```
<?
$W=”L10”;
$E= 2;
$F = $W * $E;
echo $W .’` .$E .’` .$F;
?>
```

يمكننا إضافة رقم واحد الى متغير بثلاث طرق متنوعة :
مثال

```
$j++
```

أو

```
$j = $j+1
```

أو

```
$j += 1
```

ويمكننا على ذلك إضافه المتغير إلى نفسه كالتالي :

```
$j += $j
```

أو كالتالي :

```
$j = $j + $j
```

متغيرات النظام

هناك متغيرات يستخدمها النظام يمكنك أن تستعملها ومنها

$HTTP_USER_AGENT

التي تظهر لديك نوع المستعرض الذي يستخدمه العميل

مثال :

```
<?
Echo $HTTP_USER_AGENT ;
?>
```

الثوابت

يمكننا تعريف الثوابت بقول أنها قيم ثابتة لاتتغير ونعرفها عن طريق الدالة define
الثوابت حساسة أيضا لحالة الآحرف

```
<?
Define (“author”, “alfarees”);
Echo “author is “ . author ;
?>
```

هناك ثوابت يستخدمها النظام مثل

PHP_OS

التي تقوم بعرض نظام التشغيل الذي يستخدمه السيرفر

مثال :

```
<?
Echo PHP_OS;
?>
```

معرفه وتحويل انواع البيانات

إذا أردت أن تعرف نوع متغير ما يمكنك استخدام الدالة gettype

مثال :

```
<?
$n=5;
$l ="hi";
echo "The n Is " . gettype ($n) . "<br>";
echo "The l is " . gettype ($l);
?>
```

إذا أردت تحويل نوع متغير ما يمكنك ذلك باستخدام الدالة settype :

مثال :

```
<?
$n = 10 ;
echo "Before is " . gettype ($n) ."<br>";
settype ($n,"string");
echo "After That is go " . gettype ($n);
?>
```

الدالة isset

لمعرفة إذا كان المتغير منشأ مسبقاً أم لم يتم انشاؤه وهي لا تتطلب غير اسم المتغير الذي تريد فحص وجوده وتقوم بارجاع القيمة (1) إذا كان المتغير تم انشاؤه ولاترجع أي قيمة إذا كان المتغير غير منشأ أو موجود .

مثال :

```
<?
$n = "n";
Echo isset ($n);
?>
```

الدالة unset

تقوم بحذف المتغير إذا كان موجوداً وتحرير الذاكرة منه (لذلك تأكد جيداً قبل استخدام هذه الدالة من اعطاء دمعة الوداع للمتغير المسكين)

```
<?
$n = "n";
unset ($n);
Echo isset ($n);
?>
```

الدالة empty

تقوم بإرجاع القيمة (1) إذا كان المتغير غير منشأ أو أن القيمة التي فيه صفر (0) أو نص فارغ ("") ولاتقوم بإرجاع أي شي اذا كان المتغير منشأ وفيه قيم غير المذكورة .

داوال الوقت التاريخ

نستطيع إيجاد الوقت و التاريخ عن طريق دوال في الـPHP من تلك الدوال الدالة gmdate ()

مثال :

```
<?
Echo gmdate (m);
Echo gmdate (M);
?>
```

لاحظ أن هناك فرق في النتائج مع أننا نستخدم نفس الحرف لكن طريقه العرض تختلف عندما يكون الحرف كبيراً أو صغيراً .

تحتجز الـphp بكثير من الدوال والكلمات المحجوزة التي تقوم بعمليات مختلفة مثل العمليات الحسابية المعقدة والقيام بإيجاد الوقت والتاريخ وإرسال الرسائل البريدية وإيقاف السكربتات لعدة ثواني هذه الدوال ليس مطلوب منك أن تحفظها كما تحفظ اسمك إنما المطلوب منك أن تفهم ماهية عملها واستخدامها في الوقت الذي تراه مناسباً .

يمكنك أيضا عرض اليوم والشهر

مثال

```
<?
Echo gmdate ("M D");
?>
```

لاحظ أننا استخدمنا علامات التنصيص لكي تنجح العملية عندما قمنا باستخدام أكثر من عامل في الدالة

جرب استخدام الكود التالي :

```
<?
Echo gmdate ("D, d M Y H:i:s")
?>
```

النماذج

النماذج في الويب أو صفحات الانترنت عبارة عن استمارات تقوم بتعبئتها ثم عند إرسالها لخادم الويب (السيرفر) يتلقاها برنامج يقوم بإجراء العمليات عليها مثل JavaScript أو ASP أو php (في حالتنا) .

فائدة النماذج

لنقل أنك مثلاً أردت شراء كتاب من الانترنت فإنك في الواقع تحتاج إلى تعبئة استمارة ببياناتك ورقم بطاقة الائتمان وغير ذلك من المعلومات ويتم ذلك عن طريق نموذج (فورم) .

في الواقع أنت تقوم بإختيار الكتاب الذي تريد وتكتب اسمك ورقم هاتفك وصندوق بريدك (ربما) في فراغات أو عن طريق الإشارة الى مجموعة من الخيارات .
يتم تخزين هذه القيم في المتغيرات التي يتم كتابتها في الخاصيه name (نتكلم عنها في هذا الدرس) ويتم ارسالها عند ضغط زر - ارسال البيانات - (submit) الى (البرنامج) الصفحة التي سوف تقوم بمعالجة هذه البيانات (والتي يتم تحديدها في الخاصية ACTION) وإجراء العمليات عليها مثل تخزينها مثلاً في قاعدة البيانات أو ارسالها إلى البريد الالكتروني وذلك عن طريق الphp .

ماذا يعمل العميل في النماذج ؟

إنه باختصار يقوم بتعبئة مربعات نصوص (textBox) ويقوم بوضع علامة صح في مربعات الاختيار (check boxes) أو يقوم بالتصويت أحيانا لشي معين فيختار زر اختيار (ازرار الراديو) .
هذه الأشياء كلها يتم انشاءها بواسطة الhtml ودرسنا لهذا اليوم يناقش كيفيه انشاءها وكيفيه التعامل والحصول على البيانات منها ، بقي علينا كبداية أن نعرف أن هذه الأدوات تنشأ في الواقع بين وسمين من وسوم لغه الhtml وهي الوسمين

```
<form>
</form>
```

خصائص النماذج

يجمع النموذج جميع خصائص المضيف لكننا هنا سنتطرق الى اثنين منهما وهما ACTION و METHOD التي تستخدم بكثرة و مهمة لنا في دروسنا القادمة
اما (ID;CIASS;NAME) فيلزمها تعمق في HTML خاصة عندما ندخل في ACCEPT-CHAR و ENCTYPE وستكون خارج نطاق موضوعنا حالياً وقد نفصلها في دروس قادمة إن شاء الله .

ACTION

وظيفة هذه الخاصية أن تخبر السيرفر مكان الصفحة التي يقوم بارسال معلومات النموذج إليها أوعنوانها أيا كان نوعها ، وطبعاً في حالتنا ستكون الصفحة الثانية هي الصفحة التي تحتوي على سكربت الphp .
ليس مهما أن تكون الصفحة php فقد تكون html ولكنها تحتوي على كود يختص بالتعامل مع برنامج تفاعلى لصفحات الويب مثل الجافا .
ولانريد أن نخرج عن نطاق الموضوع فدعنا نعطي مثالاً على هذه الخاصية :

```
<FORM ACTION =”TEST.PHP”>
.....
</FORM>
```

METHOD

هذه الخاصية تقوم بإخبار النموذج طريقة ارسال المعلومات الى الصفحة الهدف وفي الواقع هناك طريقيتين مشهورتين ومعروفتين لارسال المعلومات هما GET وPOST .

```
<FORM ACTION =”test.php” METHOD = “GET”>
```

أو

```
<FORM ACTIN = “test.php” METHOD =”POST”>
```

ملاحظه /في الواقع يوجد اكثر من هذه الطريقتن لارسال المعلومات وهي
(CONNECT;HEAD;OPTIONS:DELETE:TRACE) وغيرها ولكن لاتستخدم الا بشكل نادر .

دعنا الآن نفصل هاتين الطريقتين بشكل أوسع :

GET

تقوم هذه الخاصية بإخبار مستعرض الانترنت لديك بأن يقوم بإضافة المعلومات التي تمت كتابتها في النموذج إلى متصفح الانترنت لديك وتكون طريقة كتابته كاالتالي :

1- كتابه عنوان الصفحه المصدر .
2- اتباعها بعلامة استفهام .
3- كتابة العناوين والقيم .

http://localhost/test.html?name=value

قد تكون النقطتين الأخيرتين غير مفهومتين بشكل جيد بسبب أنك لم تتعامل مع النماذج من قبل .
لكن الحقيقة أن النموذج يتكون من عناصر (مربع علامة ، مربع نص ، زر اختيار) ولكل من هذه العناصر عنوان خاص بها (name) ولكل منها قيمه خاصه بها (value) .
وهي مشابهة للمتغيرات ويمكن أن يحتوي عنوان الصفحة على أكثر من عنوان (name)واكثر من قيمة (value) ويقوم بالتعريف عنهما باستخدام المعامل (&) .

مثال :

http://localhost/test.html?animal=cat&age=30

تسمى الإضافة التي تظهر بعد علامة الاستفهام (query String) نتيجه الاستعلام الحرفية.
العنوان دائما يكون باللغه الانجليزيه (name) ونعامله كانه اسم متغير من المفترض تعريفه في الصفحه الهدف (التي سنكتبها بالـPHP).

قد تحتوي القيم على فراغات او معاملات مثل (+،-،\،#،%)
يقوم المتصفح باستخدام لغه تشفير الصفحات URL ENCODING .
أيضا يستخدم الـURL ENCODING مع الأحرف العربية أو اللغات الأخري غير الإنجليزية في كتابة الحرف .

URL Encoding

هناك بعض الأحرف لايستطيع المتصفح إضافتها لعنوان الصفحة بصيغتها الحقيقية بل يستخدم لغة التشفير في التعريف عنها وهذه جداول بالرموز الذي يستخدم المتصفح كود بدلا من عرضها بصيغتها الحقيقية

الحرف	شفرته	الحرف	شفرته	الحرف	شفرته
Tab	%09	(	%28	;	%3B
Space	%20	)	%29	<	%3C
!	%21	+	%2B	>	%3E
"	%22	،	%2C	=	%3D
#	%23	.	%2E	?	%3F
@	%40	/	%2F	%	%25
\	%5C	:	%3A	&	%26

لاتقلق فليس عليك أن تحفظ كل هذه العلامات وتشفيراتها بل سيقوم المتصفح بالعمليه كلها بدلا عنك .

POST

في الواقع وظيفتها هي نفس وظيفة الـget ولكنها لاترسل المعلومات في عنوان صفحة الانترنت بل تقوم وضعها في الـbody التابع لـhttp response .
بالإضافه إلى أنه يستطيع ارسال البيانات بكمية أكبر من الـGET .

أيهما تستخدم GET أم POST ؟

قد يكون العيب في الخاصية GET عدم سرية المعلومات التي تقوم بكاتبتها ومن الممكن أن تظهر للشخص الذي يجلس الى جوارك ... خاصة عندما تريد الحفاظ على سرية معلوماتك .
أضف إلى ذلك أنها غير مفيدة في النصوص الكبيرة الحجم .
ولكنها مفيدة في أشياء كثيرة فمثلاً محركات البحث يجب أن تستخدم هذه الخاصية لكي يستطيع المستخدم أن يستخدم عنوان البحث ويحتفظ به لوقت آخر ولا يقوم من جديد بكتابة الكلمة التي يبحث عنها .

أيضا الـPOST مفيدة في إخفاء المعلومات وإحتواء كميات كبيرة من البيانات ولكن لايمكن الاحتفاظ بعنوان الصفحة .... مع ذلك فإنها أيضا ليست جيدة في الحماية بحيث أن أي هاكر خبير يمكنه الحصول على المعلومات إذا لم يكن لها تشفير معين في نقلها ..لكن اذا اردت فعلاً ان تجعلها محمية فيجب عليك استخدام اتصال محمي الى سيرفر محمي .... او مايسمونه (SCURE CONNECTION TO SCURE SERVER) .

أدوات التحكم في النماذج :

في الواقع أن أدوات التحكم عبارة عن مربعات النصوص العاديه (التي يدخل فيها المستخدم اسمه وعنوانه) وازرار الراديو (والتي يقوم المستخدم فيها باختيا شي معين (مثل الوجبه المفضلة لديه او المشروب المفضل اليه) ومربعات الاختيار (التي تتيح للمستخدم أن يختار مايشتهي ويحب من الخيارات المعروضة)
وأيضا القوائم التي تساعدك على اختيار أكثر من شي أو شي واحد .

في أغلب هذه الاشياء يتم استعمال الوسم
<INPUT>

وتلخيص تفصيله كالتالي :

```
<INPUT TYPE= type NAME= name VALUE= value other attribute>
```

الشرح :

1 - TYPE= type
نحدد نوع الكائن إذا كان زر راديو أو مربع نص عادي أو مربعات الاختيار .

2 - NAME= name
تقوم فيها بإعطاء اسم لمتغير يتم حفظ القيمة فيه .

3 - VALUE= value
سيتضح وظيفته أكثر عندما ندرج عليه أمثله إذ أن عمله يختلف من أداة إلى أخرى .

تطبيقات عمليه

سنقوم في هذه التطبيقات بصنع برامج بسيطة تتكون من ملفين ، الملف الاول يحتوي على كود HTML يقوم بتكوين النموذج والملف الثاني يقوم بإستقبال النتائج وطباعتها .

مربعات النصوص (TEXT Box) :
نقوم بعمل ذلك كالتالى :
1 - قم بتشغيل محرر النصوص لديك .
2 - اكتب الكود التالي :

```
<html dir =”rtl”>
<FORM METHOD = “GET” ACTION = “textbox.php”>
ماهي وجبتك المفضلة في الصباح ؟

<br>
<INPUT TYPE =”text” NAME = “food” value=”جبنة ومربي”>
<br>
<INPUT TYPE= submit VALUE=”إرسال”>
<INPUT TYPE= reset VALUE=”مسح”>
</form>
</html>
```

3 - قم بحفظ الملف كصفحة HTML . وقم بتسميته (textbox.html) .
4 - افتح محرر النصوص إذا كنت أغلقته .
5 - اكتب الكود التالي :

```
<?
Echo “وجبتك المفضله والى تموت في حبها هي” . “ “ . $food ;
?>
```

6 - قم بحفظ الملف كـ php . وقم بتسميته textbox.php .

7 - الآن قم بأخذ الملفين وضعهما في مجلد السيرفر لديك .
8-قم بتشغيل السيرفر واكتب في مستعرض الانترنت لديك
http://localhost/textbox.html
9 - قم بكتابة وجبتك المفضلة واضغط زر إرسال .
10- ستظهر النتيجة .

لاحظ كيف ظهر العنوان :
http://localhost/textbox.php?food=%CC%C8%E4%C9+%E6%E3%D1%C8%ED

الشرح

لقد قمنا في البداية بعمل صفحة تتكون من نص و مربع نص وزر يقوم بعملية إرسال البيانات
قمنا بصناعة بداية النموذج بواسطة الوسم <FORM> وقمنا بتحديد المكان الذي سيتم ارسال البيانات إليه بواسطة
ACTION="textbox.php"

وقمنا بصنع مربع النص بواسطه الوسم INPUT واخترنا الـ

TYPE="text"

كما قمنا بوضع القيمة الإفتراضية فيه بواسطة القيمة

Value= "جبنه ومربي"

وقمنا بوضع الناتج الذي يضعه المستخدم في مربع النص في المتغير food .

(لاحظ ان تسمية المتغيرات حساسه لحاله الاحرف في PHP واننا لم نقم بوضع $ في صفحه المتغير في كود الـhtml).

وأيضا لقد قمنا بإضافه زر بواسطة

TYPE=SUBMIT

وقمنا بوضع كلمة على الزر وهي كلمة (إرسال)

VALUE = "إرسال"

أيضا قمنا بصنع زر آخر

Type =reset

وقمنا بجعل العبارة التي عليه (مسح)

Value="مسح"

هناك نوعين من الازرار هي SUBMIT وRESET
1- الـsubmit يقوم بإرسال المعلومات .
2- الـreset يقوم بمسح البيانات في جميع الأدوات في النموذج لإعاده إدخالها من جديد .

بعد ماقمنا بادخال البيانات وضغط زر الارسال قام النموذج بارسال البيانات إلى الصفحة المحددة في الخاصية ACTION وقامت الصفحة المحددة بإستقبال النتائج الموجودة في النموذج وهي نتيجة واحدة في مربع نصوص تم حفظ قيمته في المتغير food .
وقامت بطباعتها بواسطة الدالة echo .

نظرا لاننا استخدمنا الاسلوب GET فقد تم اعطاءنا عنوان الصفحه بالاضافه الى (?) وايضا المعلومات المسجله في المتغيرات والتي تم استخدام الـURL ENCODING فيها لانها تستخدم حروف عربيه .

مربعات النصوص الكبيره (text area) طلبات اكبر للطعام الشهي !

إذا كنت تريد أن تكتب رسالة متعددة الأسطر فإنك تحتاج إلى أداة تحكم تختلف تماماً عن مربع النص العادي وهي مربعات النصوص الكبيرة التي يمكنك فيها من إدخال نصوص كبيرة الحجم ومتعددة الاسطر .

تستخدم هذه الأداة وسم فتح ووسم إغلاق

<TEXTAREA>
</TEXTAREA>

ويمكنك تحديد حجمها بواسطه تحديد الصفوف بالخاصية rows والأعمدة بالخاصية cols .

تمرين عملي

1- قم بفتح محرر النصوص لديك
2- قم بكتابة الكود التالي :

```
<html dir="rtl" >
<FORM ACTION = "TAREA.PHP"  METHOD="POST">
ما هي وجبتك المفضلة ؟
<br>
<TEXTAREA NAME = "food" ROWS="10" COLS ="50" >
جبنة
مربي
مكرونة
بيف برغر
سمبوسة
معصوب
مطبق
ماشادونا
ماخلص لو قعدت اكتب هاها
</TEXTAREA>
<br>
<INPUT TYPE = SUBMIT VALUE ="قم بإرسال الطلبات إلى الجرسون">
</FORM>
</html>
```

3- قم بحفظ الملف باسم TAREA.html .
4- الآن قم بفتح ملف جديد في محرر النصوص .
5- قم بكتابة الكود التالي :

```
<html dir="rtl">
وجبتك المفضلة هي :
<br>
<?
Echo $food;
?>
</html>
```

6- قم بحفظ الملف باسم tarea.php
7- قم بوضعهما في مجلد السيرفر لديك .
8- قم بتشغيل البرنامج .

http://localhost/tarea.html

9- قم بضغط الزر لارسال البيانات .
10- شاهد النتيجه.

= الشرح

لانضيف شيئاً علي قولنا هنا سوى أننا نريدك أن تلاحظ كيف جهزنا القيمة الإفتراضية بكتابة نصوص بين وسومات الـtextarea وأيضا أننا استخدمنا الأسلوب POST في ارسال البيانات مما جعلها لاتظهر في شريط العنوان .
وأن الـNAME تحدد اسم المتغير التي ستذهب إليه القيمة واسم المتغير في الكود لايحتوى على $ لأنه كود HTML وليس PHP .

مربعات الاختيار (Check Box) اكثر من خيار في وقت واحد !

في الواقع قد نرى مربعات الاختيار في صفحات الويب عندما نريد الاشتراك في موقع معين لرؤيه محتوياته أو عندما نريد تسجيل بريد إلكتروني أو حجز مساحة عند موقع .
وفائدتها هي إتاحه فرصة للمستخدم لتحديد أنواع الأشياء التي يريد أن يشترك فيها مثلاً أو إتاحه فرصة له لقبول إتفاقية أو غير ذلك أو رفض الجميع أو قبول الجميع .

يمكننا صنع مربع العلامة بواسطه الوسم INPUT

```
<INPUT TYPE = "CHECKBOX" NAME = "swalif" value= "سوالف" checked>
```

نقوم بتحديد نوع الأداة بأنها مربع علامة في هذا الجزء

TYPE = "CHECKBOX"

نقوم بتحديد اسم المتغير في هذا الجزء

NAME = "swalif"

ونقوم بتحديد القيمه التي يتم وضعها في المتغير اذا قام المستخدم باختيار مربع العلامه في هذا الجزء :

value= "سوالف"

اذا لم تقم بوضع الخيار value فستكون القيمه الافتراضيه هي on عند اختيار المستخدم مربع العلامه وستكون فراغ اذا لم يقم المستخدم باختيار المربع.

ونقوم بوضع القيمه الإفتراضية بإضافه الكلمة checked فإذا تم وضع هذه الكلمة يكون مربع العلامة مختار تلقائياً أما إذا لم نتكبها فسيكون بدون علامة الاختيار .

Checked

تطبيق عملي (1) :

1- قم بفتح المفكرة وقم بكتابة الكود التالي :

```
<html dir="rtl">
<FORM ACTION="CHECK.PHP" METHOD = "POST">
مالذي تريد أن تفعله في الحياة ؟ (يمكنك اختيار أكثر من إجابه )
<br>
<INPUT TYPE="CHECKBOX" NAME = "WIFE" CHECKED>
الذي أريد أن أفعله في الحياة هو أني أتزوج وأخلص وافتك من الزهق.
<br>
<input type= submit value = "إرسال">
</FORM>
</html>
```

2- قم بحفظ الملف باسم check.html .

3- قم بفتح ملف جديد في المفكرة وقم بكتابة التالى :

```
<?
Echo $WIFE ;
?>
```

4- قم بحفظ الملف باسم check.php .

5- قم بنقل الملفين الى مجلد السيرفر .

6- اكتب في المتصفح

http://localhost/check.html

7- النتيجه

تطبيق عملي (2) :

1- افتح المفكرة واكتب الكود التالي وقم بحفظه في ملف جديد باسم check2.html

```
<html dir="rtl">
<FORM ACTION="CHECK2.PHP" METHOD = "POST">
مالذي تريد أن تفعله في الحياة ؟ (يمكنك إختيار أكثر من إجابة )
<br>
<INPUT TYPE="CHECKBOX" NAME = "WIFE" value= "زوجة" CHECKED>
الذي أريد أن أفعله في الحياة هو أني أتزوج وأخلص وافتك من الزهق.
<br>
<INPUT TYPE="CHECKBOX" NAME = "jihad" value= "جهاد" >
أبغى أروح الجهاد واخمع رؤوس الكفرة والمشركين
<br>
<INPUT TYPE="CHECKBOX" NAME = "qran" value= "قران" CHECKED>
والله لو ألتحق بتحفيظ قرآن واحفظ القران كامل وأطبقه في عملي وحياتي حرتاح في حياتي كثير
<br>
<input type= submit value = "إرسال">
```

```
</FORM>
</html>
```

2- قم بفتح ملف جديد وقم بوضع الكود التالى فيه :

```
<html dir = "rtl">
<?
Echo $WIFE . " " . $jihad . " " . $qran ;
?>
</html>
```

3- قم بحفظه باسم check2.php
4- قم بتشغيل الملف .
5- النتيجه

تطبيق عملي (3)

1- افتح محرر النصوص واكتب الكود التالي :

```
<html dir="rtl">
<FORM ACTION="CHECK3.PHP" METHOD = "POST">
مالذي تريد أن تفعله في الحياة ؟ (يمكنك إختيار أكثر من إجابة )
<br>
<INPUT TYPE="CHECKBOX" NAME = "alswalif[]" value= "زوجة" CHECKED>
الذي أريد أن أفعله في الحياة هو أني أتزوج وأخلص وافتك من الزهق.
<br>
<INPUT TYPE="CHECKBOX" NAME = "alswalif[]" value= "جهاد" >
أبغى أروح الجهاد واخمع رؤوس الكفرة والمشركين
<br>
<INPUT TYPE="CHECKBOX" NAME = "alswalif[]" value= "قران" CHECKED>
والله لو ألتحق بتحفيظ قرآن واحفظ القران كامل وأطبقه في عملي وحياتي حرتاح في حياتي كثير
<br>
<input type= submit value = "إرسال">
</FORM>
</html>
```

2- قم بحفظه باسم check3.html وافتح محرر النصوص من جديد واكتب الكود التالي :

```
<html dir="rtl">
<?
Echo "$alswalif[0] <br>" ;
Echo "$alswalif[1] <br>" ;
Echo "$alswalif[2] <br>" ;
?>
</html>
```

3- قم بحفظه باسم check3.php وقم بنقلهما الى ملف السيرفر .
4- قم بتشغيل البرنامج

http://localhost/check.html

5- قم بضغط زر ارسال وانظر للنتيجه

الشرح

في الواقع لقد قمنا بتطبيق ثلاث تمارين التمرين الاول أردنا لفت النظر إلى أننا قمنا بعدم استخدام value للمتغير وتم إعطاء القيمة on عند اختيار المستخدم مربع العلامة بالإضافة أن مربع العلامة كان مختاراً بسبب وضعنا الخاصية CHECKED ولكن التمرين غير عملي وغير جيد بدون وضع قيم VALUE عند وضعنا لأكثر من مربع اختيار لذلك فقد قمنا باضافه قيم يتم وضعها في المتغيرات عند اختيار المستخدم لها كما في التمرين الثاني وأردنا لفت النظر في التمرين الى شي يسمي بالمصفوفات فإذا أردنا مثلا أن نجعل اسم المتغير متشابها واجراء عمليات تكون أسرع عليه نستخدم المصفوفات ولن نتطرق إلى المصفوفات حالياً ولكن أردنا لفت نظرك فقط وسنقوم بالتكلم عن المصفوفات بالتفصيل في الدروس القادمه باذن الله هي والتكرارات بعد التكلم عن العبارات الشرطية في الـPHP .

ازرار الراديو(RADIO BUTTONS) (اختر المشروب المفضل !)

ماهو اختيارك المفضل ؟ علما بانه لايمكنك اختيار اكثر من خيار واحد !!

في الواقع إن زر الراديو يتيح لك أن تختار شي واحد من بين عدة اختيارات ونراه كثيراً عند اتفاقيات البرامج حيث يعطيك فرصه إما بقبول الإتفاقية أو رفضها ويكون واحد من الاختيارين محددا (وهو خيار الرفض!) .

يتم استخدام ازرار الراديو باستخدام العبارة <INPUT> كالتالى :

```
<INPUT TYPE = "radio" NAME = "name" value= "value" checked>
```

نقوم بتحديد نوع الكائن بانه زر راديو في هذا الجزء :

```
TYPE = "radio"
```

نقوم بتحديد اسم المتغير في هذا الجزء :

```
NAME = "name"
```

نقوم بتحديد القيمة التي ستكون في المتغير هنا :

```
value= "value"
```

في الواقع مع ازار الراديو نقوم بجعل اسم المتغير name هو نفسه والقيم مختلفة value لكل سؤال . وإذا لم نقم بوضع قيمة فسيقوم PHP بوضع القيمة on للمتغير .

تطبيق عملي :

1- قم بتشغيل محرر النصوص لديك واكتب الكود التالي وقم بحفظه في ملف اسمه radio.html .

```
<html dir="rtl">
<form action = radio.php method = "post">
ماهو مشروبك المفضل ؟
<br>
<br>
<INPUT TYPE = "radio" NAME = "mshroob" value= "شاي" checked>
شاي
<br>
<INPUT TYPE = "radio" NAME = "mshroob" value= "قهوة" >
قهوة
<br>
<INPUT TYPE = submit  value= "إرسال" >

</form>
</html>
```

2- قم بفتح محرر النصوص واكتب الكود التالي وقم بخفظه باسم radio.php

```
<html dir = "rtl">
<?
echo "مشروبك المفضل هو :" . " " . $mshroob;
?>
</html>
```

3 - قم باختيار المشروب المفضل واختر إرسال .

الشرح :

في الواقع لقد قمنا بصنع أزرار راديو ولقد قمنا بوضع قيمة لكل زر تكون تابعة للعباره التي بجوار الزر . ولقد قمنا بوضع عبارة checked لكي ترى كيف أن الأداة التي تحتوي على العبارة تكون محددة تلقائيا ولاحظ أن العبارة التي تكون بجانب الزر تكون موجودة أسفل كود الزر مثل :

```
<INPUT TYPE = "radio" NAME = "mshroob" value= "شاي" checked>
شاي
```

العبارة هي الملونة باللون الأحمر .

وأيضا لاحظ أننا استخدمنا متغيراً واحداً فقط لجميع الإختيارات بحيث أن جميع الأزرار قيمتها تعود إلى هذا المتغير .

القوائم (Lists Or drop down menus) اختر مواصفات زوجتك للمستقبل واسمها :

تستخدم القوائم في الـ html بشكل مختلف قليلاً عن الأداوت السابقة إذ أننا نستخدم وسمين من وسوم لغة html وهما :

<select> لنقوم بإنشاء القائمة و <OPTION> ونستخدم الخاصية MULTIPLE إذا كنا نريد إتاحه الفرصه للمستخدم أن يختار أكثر من قيمة ونقوم بوضع القيمة التي يختارها المستخدم في متغير بواسطه الخاصية NAME أو في مصفوفة متغيرات (وسيتضح مفهوم المصفوفات لديك جيدا في درس المصفوفات باذن الله .

تطبيق عملي :

1- قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه في ملف باسم lists.html :

```
<html dir="rtl">
<form action = "lists.php" method = "post">
ماذا تريد ان يكون اسم زوجة المستقبل(لغير المتزوجين ) ؟
<br>
<select name = "wife" >
<option> هناء </option>
<option>جمانة </option>
<option>رزان </option>
<option>سحـــر </option>
<option>سارة </option>
<option>سمية </option>
<option>روان </option>
<option>دلال </option>
<option>اسم اخر </option>
</select>
<BR>
ماذا تريد أن تكون مواصفاتها ؟
<Br>
<select name="dis[]" multiple>
<option>جميلة</option>
<option>متدينة</option>
<option>شقراء </option>
<option>جعداء الشعر</option>
<option>سوداء </option>
<option>سمراء </option>
<option>بيضاء </option>
</select>
<br>
<INPUT TYPE=SUBMIT VALUE="إرســال">
</html>
```

2- قم بفتح ملف جديد واكتب فيه الكود التالي وقم بحفظه باسم lists.php :

```
<html dir="rtl">
<?
Echo "لقد أردت أن يكون اسم زوجتك " . " " . $wife ;
Echo "<br><br>";
Echo "ولقد أردت أن تكون مواصفاتها ";

Echo "<br><br>";
Echo "$dis[0] <br>";
Echo "$dis[1] <br>";
Echo "$dis[2] <br>";
Echo "$dis[3] <br>";
Echo "$dis[4] <br>" ;
Echo "$dis[5] <br>";
Echo "$dis[6] <br>";
?>
</html>
```

قم بتشغيل البرنامج

http://localhost/lists.html

واختر ماتريد ثم اضغط زر ارسال

الشرح :
لقد قمنا بصناعة قائمة تسمح بإختيار قيمة واحدة منها ثم تذهب هذه القيمة الى المتغير wife وصنعنا قائمة ثانيه تسمح بإختيار أكثر من عنصر واحد وقمنا بوضع هذه القيم في مصفوفه متغيرات (سيتضح معني المصفوفات في دروس قادمه ان شاء الله) .

الاداة الخفيه (والمعلومات السريه!) (hidden control)

هناك بعض الأوقات تحتاج فيها إلى إرسال بعض المعلومات من صفحة ويب الى صفحة ويب اخرى عن طريق النماذج وفي نفس الوقت أنت لاتريد المستخدم أن يقوم برؤية هذه المعلومات .

في الواقع هناك أداة تساعدك على اخفاء هذه المعلومات على المستخدم يسمونها بحقل النموذج المخفي أو الأداة الخفية (hidden form field or hidden control) .

هذه الأداة تلعب دوراً مختلفاً ومتميزاً عن بقية الأدوات وهي اخفاء المعلومات التي تم ادخالها كما شرحنا في السابق وهي مفيدة جدا مع النماذج المصنوع بواسطة الـPHP إذ أنها تسمح لنا أيضا بان تكون المعلومات المخفيه هي متغيرات PHP .

يتم صنع هذه الحقول المخفية كالتالى :

```
<INPUT TYPE=HIDDEN NAME =hidden1 VALUE="الرسالة السرية">
```

نقوم بوضع HIDDEN لكي يعرف المتصفح أن هذه المعلومات خفية (لاتظهر للمستخدم) ونضع اسما للمتغير الذي يقوم بالاحتفاظ بالمعلومات والذي يتخزن اسمه في الـNAME ونقوم بوضع المعلومات التي نريد اخفاءها في الـVALUE .

نستطيع الاستفاده أيضا منها عن طريق الـphp وذلك عن طريق كتابة كود الـHTML بواسطة الأمر echo() في الـPHP كما في المثال التالى :

```
<?
$msg1= " هذه العبارة لن تظهر ";
Echo "<form>" ;
Echo "<input type=hidden name =secret value= '$msg1'>";
Echo "<input type=submit>";
Echo "</form>";
?>
```

هذا الكود الذي تراه عبارة عن كود HTML تم كتابته بالـPHP عن طريق الامر echo() ولقد استطعنا تخزين قيمة متغير php ($msg) في متغير html (secret) .

تطبيق عملي :

1 - افتح محرر النصوص واكتب الكود التالي واحفظه باسم hid.php :

```
<html dir="rtl">
<head></head>
<body>
```

```
<?
$car1= "لكزس";
$car2= "ماكسيما";
$car3="لاندكرزور";
Echo "<form method =get action='hid2.php'>";
Echo "ماهي السيارة التي تتمنى أن تشتريها أو تحظي بها ؟";
Echo "
<select name= 'favcar'>
<option>$car1</option>
<option>$car2</option>
<option>$car3</option>
</select><br><br>
<input type =hidden name = hid1 value='$car1'>
<input type =hidden name = hid2 value='$car2'>
<input type =hidden name = hid3 value='$car3'>
<input type = submit  value='إرسال'>
</form>";
?>
</body>
</html>
```

3- افتح محرر النصوص واكتب الكود التالي واحفظه باسم hid2.php

```
<html dir="rtl">
<head></head>
<body>
<?
Echo ": لقد قمنا بعرض السيارات التالية عليك<br>";
Echo "$hid1<br>";
Echo "$hid2<br>";
Echo "$hid3<br>";
Echo "<br>:ولقد قمت باختيار<br>";
Echo $favcar;
?>
</body>
</html>
```

3- قمت بنقل الملفين الى مجلد السيرفر ثم قم بتشغيل السكربت :
http://localhost/hid.php

الشرح :

لقد قمنا بعمل نموذج بسكربت الـphp لاحظ أننا استخدمنا الـ(') بـدلاً مـن (") كما كنا نعمل في الـ html وذلك لاننا قلنا سابق أن القيم الحرفية ..... (راجع درس المتغيرات) ولقد قمنا بإدراج قيم متغيرات الـphp في كود الـhtml مما يوفر علينا الكثير من إعاده الكتابة (في حال كان النص المستخدم طويلاً) .
اقرا المثال أكثر من مر وسيتضح لك المقال أكثر باذن الله .

استخدام حقل كلمات السر (Password fields)

لكي تجعل المعلومات أكثر حماية من التعرض إلى السرقة أو غير ذلك يمكنك استخدام حقول كلمات السر الذي هو عبارة عن مربع نص بسيط يقوم بإظهار النص على شكل نجوم **** في حال كان الجهاز يستخدم على يد أكثر من شخص فان هذه الطريقة جيدة قليلاً في أن لايري شخص معلومات الآخر السرية .
في الواقع مع ذلك فإنك لاتكون قد اديت حماية إذا كان الاسلوب المستخدم في ارسال بيانات المستخدم هو الاسلوب get إلا إذا كنت تستخدم تشفير البيانات ويكون أكثر جودة اذا استخدمت الاسلوب post وايضا لن يكون محمياً من الهاكر إذا لم تكن تستخدم SSL (Secure Socket Layer) لكي تقوم بتنشيط تشفير البيانات .

تطبيق عملي

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم pass.php

```
<html dir=”rtl”>
<body>
<form method=post action=”pass1.php”>
اسم المستخدم
<br>
<input type="text" name ="user">
<br>
كلمة المرور
<input type="password" name ="pass">
<br>
<input type = submit  value=’إرسال’>
</form>
<body>
</html>
```

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم pass1.php

```
<?
Echo “ : اسم المستخدم هو”;
Echo “<br>$user<br>”;
Echo “: وكلمه المرور هي”;
Echo “<br><br>$pass”
<?
```

قم بنقل الملفين الى مجلد السيرفر لديك
قم بتشغيل البرنامج ولاحظ النتيجه .

ارسال البريد الالكتروني بواسطه الـphp :

البريد الإلكتروني هو الحياة التي تنبض بها السكربتات فمثلاً هناك سكربتات ارسال بريد الى صاحب الموقع تخبره بشي معين أو ملحوظة أو غير ذلك ويمكن استخدامها في أكثر من مجال .
والدالة التي تستخدم في ذلك هي الدالة mail()

```
mail("$to", "$sub", "$msg”,”From:$you”);
```

وتقوم بوضع بريد الذي ستصله الرسالة في الخانة $to وموضوع الرسالة في الخانة $sub والرسالة في الخانة $msg وبريدك أنت أو بريد المرسل في الخانة $you .

تطبيق عملي

قم بكتابة الكود التالي واحفظه في ملف باسم mail.html

```
<html dir=rtl>
<head>
  <title>برنامج إرسال بريد</title>
</head>
<body>
<form action="mail.php" method="post">
عنوان المرسل
<br>
<input type="text" name = "you">
<br>
عنوان المستقبل
<br>
<input type="text" name = "to">
<br>
موضوع الرسالة
<input type="text" name = "sub">
<br>
الرساله
<textarea rows=10 cols=20 name = "msg" >
</textarea>
<input type="submit" value = "إرسال البريد الالكتروني">
</form>
</body>
</html>
```

قم بانشاء ملف اخر وقم بكتابة الكود التالي وقم بإعطاءه الاسم mail.php .

```
<?
mail("$to", "$sub", "$msg","From:$you");
?>
```

قم بوضع الملفين في مجلد السيرفر وقم بتشغيل البرنامج واملا البيانات واضغط زر الارسال وستري ان الرساله تم ارسالها بنجاح .

برامج عمليه

برنامج او سكربت ارسال بطاقات بسيط
يحتوي من ملفين الملف الأول به البطاقات وعنوان البريد الإلكتروني والملف الثاني هو الذي يقوم بعملية الإرسال

الملف الأول هو chcard.php وكوده كالتالى :

```
<html dir=”rtl”>
<form action =card.php method = “post”>
اختر البطاقة التي تريد ارسالها
<br>
<br>
<INPUT TYPE = ”radio” NAME = “card” value= ” http://www.khalaad.f2s.com/MADINA9_small.JPG”
checked>
البطاقه الاولى
<br>
<br>
<img src="http://www.khalaad.f2s.com/MADINA9_small.JPG" width="100" height="100" alt="" border=0>
<br>
<br>
<INPUT TYPE = ”radio” NAME = “card” value= ” http://www.khalaad.f2s.com/Haram3.jpg” >
البطاقه الثانيه
<br>
<img src="http://www.khalaad.f2s.com/Haram3.jpg" width="100" height="100" alt="" border=0>
<br>
اسمك
<br>
<input type="text" name = "myname">
<br>
بريدك الالكتروني
<br>
<input type="text" name = "you">
<br>
بريد صديقك
<br>
<input type="text" name = "to">
<br>
موضوع التهنئه
<input type="text" name = "sub">
<br>
الرساله
<br><br>
<textarea rows=10 cols=20 name = "msg" >
</textarea>
<br>
<INPUT TYPE = submit  value= ”إرسال البطاقة” >
</form>
</html>
```

الملف الثاني يقوم بعملية ارسال البطاقة وتقوم بكتابة الكود التالي واحفظه في ملف باسم card.php

```
<?
$message = " لقد قام$myname بارسال بطاقه اليك " . "\n". " وهو يقول في نص رسالته لك:\n$msg"." وتجدها على الرابط التالى" ."\n". "$card" ."\n";
mail("$to", "$sub", "$message","From:$you");
echo "<center> لقد تم ارسال الرساله بنجاح،،، مبارك</center>";
?>
```

ملاحظة :

الدالة \n تقوم فقط ببدا سطر جديد لاننا لانستطيع استخدام
 في نص الرسالة

الأوامر الشرطية

لقد أخذنا في الدروس السابقة فكرة عن المتغيرات وكيفية تعامل البيانات مع النماذج ...في هذا الدرس سنتعلم كيفية التحكم بالكود بمعني تنفيذ سطر معين من الكود عند حصول شرط معين وعند عدم حصوله نتجاهل السطر ونتجه الى السطر الذي يليه ..هذا يمنحنا تحكماً أكبر بالكود ويجعلنا نستخدم قرارات وتنفيذ أشياء ممتازة وبرامج رائعة بالـPHP .

دعنا نعطيك فكرة من حياتنا اليومية ....
تقوم في الصباح وتريد أن تحضر فطورك الذي يتكون من التالي :

عسل
جبنة
خبز
شاي

ستقوم بالذهاب إلى الثلاجة ثم تقوم بالبحث عن الأشياء التي يتكون منها فطورك ، فإذا لم تجد ما تريد تستعد للذهاب إلى المركز التجاري لشرائه حاجتك ، تذهب إلى المطبخ وتتأكد مره أخرى وتبحث عن المؤونة التي يحتاجها البيت بشكل عام .

1- تبحث عن جبنة وإذا لم تجدها تنتقل إلى الخطوة 3 .
2- إذا وجدت جبنة فإنك تبحث عن العسل فإذا وجدته تنتقل الى الخطوة 4 ، واذا لم تجده تنتقل الى الخطوة 5 .
3- تقوم بكتابتها في ورقة جانبية وتقوم بالبحث عن العسل .
4- تتجهز للذهاب إلى المركز التجاري .
5- تكتبه في ورقه جانبيه ثم تتجهز للذهاب إلى المركز التجاري .

هل لاحظت انك كنت تقوم بالبحث عن أشياء معينة فاذا وجدتها (true) قمت بالبحث عن التي تليها وإذا لم تجدها (false) تقوم بتسجيلها في قائمة المشتريات لديك .

القيم المنطقية والدوال الشرطية

في الواقع لقد تكلمنا عن المتغيرات سابقاً وذكرنا بأن هناك متغيرات منطقية (قيمتها إما صحيح إم خطأ) ولم نقم بشرحها ، وهذا الدرس سيتولي شرحها وإعطاء أمثلة على كيفية التعامل معها .

العبارة IF

```
IF condition is true ( إذا كان الشرط صحيحاً )
{
excute this code (قم بتنفيذ هذا الكود)
}
```

إن الدالة IF معروفة تقريباً في جميع لغات البرمجة ...حيث أنها تقوم بعملية التحقق من شي معين وتنفيذ بعض الأشياء إذا كان الشرط صحيحا (true) والقيام بتنفيذ أشياء أخرى أذا لم يكن صحيحا ....
سيقوم الـPHP بتنفيذ الكود التي بين { و } فقط اذا كان الشرط صحيحاً .
أما إذا لم يكن صحيحاً فسيقوم بتجاوزه وتنفيذ الكود الذي يليه .

ويمكنك أيضا أن تقوم بجعلها بسطر واحد ولا تستخدم الأقواس بل تكتب الأمر مباشرة :

```
IF condition is true excute function;
```

لاحظ أنه لابد من استخدام { و } إذا كان الكود يتكون من عدة أسطر أما إذا كان يتكون من سطر واحد فلا داعي لاستخدامها .

فالمثالين التاليين كلهما صحيحين
مثال(1)

```
<?
$S=10
IF ($S=10) echo 11;
?>
```

مثال (2)

```
<?
$S=10
IF ($S=10){
 echo 11;
}
?>
```

لنتخيل مثلاً أن الجو ممطر وسنقوم بإعطاء المطر متغيراً ونسميه rain ونقوم بإعطاء المظلة اسم متغير آخر ونسميه umbrella وسنقوم بإفتراض أن هناك أمر في الـphp يسمي go out حسناً الآن الكود الذي نريد أن نقوم بكتابته هو :

```
If $rain = true
{
$umberrlla = true
}
go out();
```

فائدة هذا الكود هو أن تأمر الـPHP بحمل المظلة ($umberrlla=true) معه إذا كان الجو ممطراً ($rain=true)وإذا لم يكن ممطراً ولم يتحقق الشرط فإنه سيخرج إلى النزهه بدون أي مظلة .

طبعا ليس هناك دالة تقوم بذلك إنما قمنا بذلك من أجل التوضيح للمستخدم هيكلية عمل الدالة بشكل عام .

مقدمه الى القيم المنطقيه (Boolean Values)

القيم المنطقية ترمز إلى الأشياء التي لاتحتمل أكثر من احتمالين وهما إما صح وإما خطأ ، وهي نوع جديد من القيم غير التي كنت نعرفها سابقا (مثل الرقميه والنصيه) .

مثال

```
<?
$variable=true;
echo "$variable";
?>
```

لو قمت برؤيه النتيجة ستجد أنه يطبع الرقم واحد وهو قيمة المتغير إذا كان صحيحاً ، أما أذا كان خطأ أو غير صحيح فقيمته ستكون(0) .

المعاملات المنطقية

لقد أخذنا المعاملات الرياضية فيما سبق بشيء من التفصيل (+،-،/،*) والان سنأخذ شيئاً جديداً من المعاملات وهي المعاملات المنطقية التي تساعدنا في صناعة الشروط والتقييدات على شي معين وتعطينا تحكما أكبر في الكود .

المعاملات : < و >

من المفترض أن تكون متآلفاً مع علامتي الأكبر من والأصغر من في الرياضيات التي تتعلمها في

المدرسة مما يجعل فهم هذا الأمثله بسيطاً .

```
<?
If (6>5)
{
echo " الرقم ستة أكبر من الرقم خمسة";
}
Echo "end";
?>
```

سيقوم الـPHP في مثالنا هذا بفحص الشرط (6>5) فإذا كان صحيحاً (true) سيقوم بطباعة السطر (الرقم سته اكبر من الرقم خمسه) ثم يقوم بطباعة end ، وإذا لم يكن صحيحاً فسيقوم بتجاهل الكود وطباعة (end) فقط .

يمكننا أيضا استعمالها في المقارنة بين متغير ورقم أو بين متغير وثابت (constant) أو العكس أو المقارنة بين متغيرين .

مثال (1)

```
<html dir ="rtl">
<?
$LuckeyNumber = 5;
If ($LuckeyNumber<6)
{
echo (" رقم الحظ أصغر من الرقم ستة");
}
?>
```

مثال (2)

```
<html dir="rtl">
<?
```

```
$f=5;
$r=10;
If ($f >$r)
{
echo " أكبر من المتغير $r$f المتغير";
}
?>
```

تطبيق عملي :

قم بتشغيل محرر النصوص واكتب الكود التالي واحفظه باسم thegame.php

```
<html dir = "rtl">
<body>
<form method =get action="game.php">
ماهو الرقم الذي أفكر به الآن والذي هو بين 1 و10 ؟
<input type="text" name="number">
<br>
<br>
<input type =submit>
</form>
</body>
</html>
```

قم بفتح محرر النصوص لديك من جديد واكتب الكود التالي واحفظه باسم game.php

```
<html dir="rtl">
<body>
<?
$num = rand (1,10);
if ($number>$num)
{
echo "لقد اخترت رقم أكبر من الذي أفكر فيه" ;
Echo " الرقم الذي أفكر فيه هو ";
Echo $num;
Echo "<br>"." يؤسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة" ;
}
if ($number<$num)
{
echo "لقد اخترت رقم أصغر من الذي أفكر فيه" ;
Echo " الرقم الذي كان في مخيلتي هو$num";
Echo "<br>". " يؤسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة";
}
?>
 لقد نجحت
</body>
</html>
```

شرح التطبيق :

الداله rand

تقوم هذه الدالة باختيار رقم عشوائي من بين رقمين يتم اعطاءها إياها الرقم الاول(x) هو الأصغر والرقم الثاني هو الأكبر(y)

```
Rand (x,y);
```

يمكنك حفظ القيمة التي تقوم بإخراجها هذه الدالة في متغير مباشرة

مثال

```
$Num = rand (5.57);
```

وهذا يوضح ماقمنا به في الكود

```
$num=rand(1,10);
```

لقد قمنا باختيار قيمة عشوائية ثم قمنا بمقارنتها مع القيمة التي تم إدخالها من قبل المستخدم فإذا كانت القيمة التي أدخلها المستخدم أكبر من قيمة العدد العشوائي أخبرناه بأن الرقم الذي أدخله أكبر من الرقم الصحيح ... وهذا ماتجده جليا في الأسطر التالية :

```
if ($number>$num)
{
echo "لقد اخترت رقم أكبر من الذي أفكر فيه" ;
Echo " الرقم الذي أفكر فيه هو ";
Echo $num;
Echo "<br>"." يؤسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة" ;
}
```

فإذا لم ينطبق الشرط وكان الرقم الذي اختاره المستخدم أصغر من الرقم العشوائي فإنه يترك الشرط الأول ويتجه الى الشرط الثاني ويطبق الأوامر التي فيه والتي تقوم باخباره بأن الرقم الذي قام باختياره أصغر من الرقم المطلوب ، وهذا ماتجده جليا في الأسطر التالية :

```
if ($number<$num)
{
echo "لقد اخترت رقم أصغر من الذي أفكر فيه" ;
Echo " الرقم الذي كان في مخيلتي هو$num";
Echo "<br>". " يؤسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة";
}
```

فإذا لم يتطبق الشرطين فإنه يتركهما ويكتب الكلمة (لقد نجحت) بدون أي كلمات آخرى مثلما كنا نكتب الكلمه (يؤسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة) قبل كلمة (لقد نجحت) ، أتمني أنك قد فهمت جيداً ما أقول .... وتظهر هذه العبارة جلية في الأسطر التالية :

```
?>
 لقد نجحت
</body>
</html>
```

على هذا نكون قد صنعنا لعبة كاملة تقوم بإخبار المستخدم عند نجاحه او خسارته .

معاملات المساواة : == و ===

لقد قمنا باستخدام علامة المساواة الفردية سابقاً في تخزين قيمة في متغير وهانحن نأخذ نوعاً من علامات المساواة وهو علامة الساواة المزدوجة (==) وعلامة المساواة المضاعفة (===) .

لقد كنا نستخدم علامة المساواة الفردية او العادية في تخزين القيم في المتغيرات .

مثال :

```
<?
$m=12;
?>
```

ولكن العلامات التي نتكلم عنها الآن تستخدم في تحديد إذا ماكانت قيمة معينة تساوي قيمة اخري .

مثال :

```
<?
$m=”11”;
$u=11;
If ($m==$u)
{
Echo “ القيم متساوية”;
}
?>
```

لاحظ أن $m متغير حرفي وان $u متغير رقمي .
إذا كنا نريد ارجاع قيمة إلى متغير نستخدم علامة المساواة العادية (=) وإذا أردنا اختبار متغيرين أو قيمة معينة من أنها متساوية نقوم باختبار القيم بواسطة علامة المساواة المزدوجه (==) .
في الـphp4.01 تم إصدار علامة مساواة جديدة تقوم باختبار القيم ولاتعطي القيمه (true) إلا إذا كانت أنواع القيم متساوية وأنواع البيانات في المتغيرات ايضا متساوية .

مثال (1) :

```
<?
$m=”11”;
$u=11;
If ($m==$u)
{
Echo “ القيم  متساوية”;
}
?>
```

مثال (2) :

```
<?
$m=”11”;
$u=11;
If ($m===$u)
{
Echo “ القيم  متساوية”;
}
?>
```

التوضيح

لاحظ أننا في المثال الأول استخدمنا علامة المساواة المزدوجة لاختبار القيم وكانت القيم متساوية في المتغيرين فتم طباعة أن القيم متساوية (مع أن نوع البيانات مختلف) ولكن في المثال الثاني عندما استخدمنا علامة المساواة المضاعفة لم يتم طباعة أي شي وذلك لان القيم متساوية ولكن نوع البيانات مختلف فالمتغير $m حرفي بينما المتغير $u رقمي .

المعاملات : =! و <>

إن عكس علامة المساواة هي علامة عدم المساواة (=!)

مثال :

```
<?
If (5!=99) echo "القيم غير متساوية";
?>
```

لاحظ أن 5 لاتساوي 99 لذلك فإن الشرط صحيح (true) لذلك قام بطباعة أن القيم غير متساوية

إن الضد من علامة أكبر من وأصغر من هو علامة الـ(<>) وهو يقوم بارجاع قيمة (true) إذا كانت القيمتين مختلفتين عن بعضهما أي أنه مثل علامة =! تقريباً .

مثال :

```
<?
If (5<>99) echo "القيم غير متساوية";
?>
```

تطبيق عملي على علامات المساواة وعدم المساواة

قم بفتح محرر النصوص لديك واكتب الكود التالي :

```
<html>
<head></head>
<body>
<Form method =get ACTION= "quiz.php">
ماهو اسم الرجل الذي يسمي بالفاروق ؟
<br><br>
<input type ="radio" name = "man" value="عمر">
عمر بن الخطاب رضي الله عنه
<br>
<input type ="radio" name = "man" value="أبوبكر">
أبو بكر الصديق رضي الله عنه
<br>
<input type ="radio" name = "man" value="عثمان">
عثمان بن عفان رضي الله عنه
<br>
<input type = submit>
</form>
</body>
</html>
```

احفظها باسم quiz.html ...

قم بفتح محرر النصوص لديك واكتب الكود التالي :

```
<html dir="rtl">
<head></head>
<body>
<?
If ($q==" عمر") echo "الإجابة صحيحة";
If ($q!="عمر") echo "الإجابة خاطئة";
?>
```

قم بحفظه باسم quiz.php وضعهما في مجلد السيرفر

قم بتشغيل الملف quiz.html

المعاملات المنطقيه (AND,OR,NOT)

إن هذه المعاملات المنطقية تتيح لك بتنفيذ الكود بعد التحقق من مجموعة شروط وأيضا تنفيذ الكود إذا تحقق أكثر من شرط : (AND)
أو تحقق شي معين من بين عده أشياء : (OR)
ويمكنك مثلاً التحقق من عدم صحة شي لكي تقوم بتنفيذ شي آخر : (NOT)

فيمكنك مثلاً أن تقول : إذا كان الجو ممطراً والعاصفة شديده فلن أخرج من البيت .
ويمكنك أن تقول : إذا كان الجو هادئاً أو لايوجد أمطار فسأقوم بالخروج الى المنتزه .
ويمكنك أيضا أن تقول : إذا لم يكن الجو ممطراً سأقوم بالخروج إلى نزهة .

ولكن عند استخدامك لهذه الدوال عليك مراعاة أن تقوم بجعل هذه الشروط بين قوسين .

المعامل (AND) ونظيره(&&)

يمكننا استعمال المعامل (AND) والمعامل (&&) للتحقق من صحة عدة شروط لتنفيذ شي معين

مثال(1)

```
<?
$w=10;
$g=12;
IF ($w=10 and $g=12) echo ("لقد تحققت جميع الشروط");
?>
```

مثال (2)

```
<?
$w=10;
$g=12;
IF ($w=10 && $g=15) echo ("لقد تحققت جميع الشروط");
?>
```

في المثالين السابقين قمنا بعملية التحقق من أكثر من شرط باستخدام المعاملين (&& و and) فعندما تحققت جميع الشروط تم تنفيذ الأمر .... وعندما لم تكن جميع الشروط صحيحة تم تجاهل الأمر .

لاحظ أننا قمنا بجعل الشروط بين قوسين () لكي يعمل الكود بشكل صحيح :

```
($w=10 && $g=15)
($w=10 and $g=12)
```

المعامل (OR) ونظيره (||)

المعامل OR يقوم بالتحقق من عدة شروط وإذا تحقق أي واحد منها فإنه يقوم بتنفيذ الكود ونظيره (||) الذي يقوم بنفس العمليه .

مثال (1)

```
<?
$E=100;
$T=8;
IF ($E=14 OR $E=55 OR $E = 10 OR $T=8 ) echo ("لقد تحقق أحد هذه الشروط");
?>
```

مثال (2)

```
<?
$E=100;
$T=458;
IF ($E=14 || $E=55 || $E = 10 || $T=8 ) echo ("لقد تحقق أحد هذه الشروط");
?>
```

إذن عندما تحقق واحد من هذه الشروط تم طباعة السطر (لقد تحقق أحد هذه الشروط) .

ملحوظة قد لاتكون بتلك الأهمية لكن يجب أن تعرف أن الرموز && و || لها الأسبقية والأفضلية على استخدام AND و OR .

المعامل NOT ونظيره (!)

في الواقع لايمكنك استخدام NOT أبدا لأنها ليست أصلاً موجودة في لغة PHP لكن يمكنك استخدام المعامل (!) كبديل لها فهو يؤدي نفس وظيفتها وهي القيام بالتأكد من أن هناك قيمة غير صحيحة (FALSE) لكي يتم تنفيذ شي معين .

```
<?
$F="الفارس";
IF !($F=="نعمان") echo ("أهلاً بك");
?>
```

في المثال السابق يقوم الـPHP بالتأكد من أن المتغير $F لايحتوي على القيمة الحرفية (نعمان) ويتم ذلك باستخدام المعامل (!) وعندما يتم التأكد من ذلك يقوم بطباعة السطر (أهلاً بك)

ونشير إلى أننا عندما نقوم باختبار متغير بواسطة المعامل (!) فإن الـPHP إذا وجد المتغير فارغاً أو لم يتم انشاؤه يعطيه القيمة صفر وهي FALSE .

مثال

```
IF (!($R)) echo (10);
```

استخدام المعاملات <= و >=

من المعاملات المعروفة والمشهورة في الرياضيات هي علامتي أصغر من أو يساوي <= أو أكبر من أو يساوي >= وهي تستخدم بنفس وظيفتها بالـphp وهي معرفة إذا ماكانت قيمة أصغر أو أكبر من أو تساوي قيمة آخري ، وهذه الأمثلة تعطيك مدخلاً أشمل لفهم هذه الدوال :

```
<?
$t = 15;
If ($t >= 10 ) echo (“ممتاز” . “<br>”);
$t = 5;
If ($t <= 9 ) echo (“جيد جدا”);
?>
```

تجميع المعاملات

يمكننا في الشرط أن نتحقق من مجموعة من القيم باستخدام مجموعة من المعاملات ، ونقوم بتجميع هذه المجموعات داخل أقواس () مثلما كنا نستخدم سابقاً أكثر من معامل (+ ، - ، / ، *) باستخدام الأقواس .

وسيبدو ذلك واضحاً وجلياً في مثالنا التالى :

```
<?
$a=10;
$y=5;
$t =29;
If (($a == 10) or ($a==54) and ($y !=25) and ($t >= 11)) echo "تحققت جميع الشروط";
?>
```

سيتم طباعة 18 لأنه قيمة تجميع التعبير السابق تكون صحيحة ولو قمنا بشرح المثال فسنقوم برؤية القسم الأول وهو :

($a == 10) or ($a==54)

وطبعاً المتغير يحمل القيمة 10 فسيكون هذا الجزء صحيحاً .

ثم نقوم برؤية الجزء :

($y !=25) and ($t >= 11)

وطبعاً تم التحقق من جميع الشروط وتم طباعة الكلمة (تحققت جميع الشروط) .

تعدد الشروط (else if و else)

يمكننا استخدام أكثر من هيكلية للعباره if فهناك مثلاً الهيكلية التالية :

```
If condtion is true
{
Excute code
}
Else
{
Excute other code
 }
```

وهي تقوم بالتحقق من الشرط فإذا وجدته صحيحاً قامت بتنفيذ الكود الأول وإذا لم تجده صحيحاً ستقوم بتنفيذ الكود الآخر .
مثال

```
<?
$age=10;
If ($age>18)
 {
echo "مرحبا بك في أكبر موقع تجاري إلكتروني";
}
else
{
echo "ممنوع دخول الأطفال الموقع لأنهم لايملكون المال";
}
?>
```

ويمكننا أيضا استخدام الهيكلية التالية :

```
If condtion is true
{
Excute code
}
Elseif
{
Excute other code
 }
Else
{
Excute other code
 }
```

وهي تقوم بتطبيق أكثر من شرط فإذا لم يكن أي شرط من الشروط صحيحاً سيتم تنفيذ الكود الذي يقع بعد كلمه else . مثال :

```
<?
$age=10;
 If ($age<=18)
 {
echo "مرحبا بك في أكبر موقع تجاري إلكتروني";
}
elseif ($y >= 44);
{
echo " مافي مشكلة برضه إذا كنت كبير";
}
else
{
echo "ممنوع البقية";
}
?>
```

تعشيش العبارات الشرطية

يمكنك تعشيش العبارات الشرطية ، ونعني بتعشيش العبارات الشرطية هي أن تقوم بعملية تعشيش الشروط فمثلاً إذا كان شرط ما صحيحاً فإنه يجب أن يكون شرط آخر صحيحاً لكي يتم حصول شي معين وغير ذلك .
مثال :

```
<?
$h=”ahmed”;
$f=45;
If  ($h = = “ahmed” )
{               If ( $f= = 45)
        {
        echo “الاسم والرقم صحيحان”;
                }
                 else
                {
                echo (“ الرقم غير صحيح “ );
        }       }
else    {
echo “ اسم تسجيل الدخول غير صحيح “ ;
}
      ?>
```

هذا مجرد مثال بسيط جداً لتعشيش الدوال الشرطية حيث يقوم بإجراء اختبار على قيمة معينة ثم يقوم عند تجاوزه ذلك الاختبار بنجاح بإجراء اختبار ثاني فإذا تم تجاوز الاختبار الثاني يتم طباعة الاسم والرقم صحيحان وإذا لم يتم الاجتياز يتم طباعة عبارة الفشل في الاجتياز .

تطبيق عملي

سنقوم في هذا التطبيق بصناعة مسابقة بسيطة نستخدم فيها ماتكلمنا عنه سابقاً

1- قم بانشاء ملف Msabqa.html .
2- قم بكتابه الكود التالي فيه :

```
<html>
<body>

<form method="POST" action="msabqa.php" dir="rtl">
<br>من هو أول الخلفاء الراشدين
<p><br><input type="radio" value="abubaker" name="s">أبوبكر الصديق<br><input
type="radio" value="3mar" name="s">عمر
<br><input type="radio" value="3thman" checked name="s">عثمان<br><br><br>

</p>

  <p><input type="submit" value="ارسال">    <input type="reset" value="حـــــــذف"></p>
</form>

</body><html>
```

قم بفتح ملف وقم بتسميته msabqa.php

```
<?
<html dir = “rtl”>
If $s == “3mar”{
```

```
 الإجابه صحيحة
}
else
{
echo "الإجابه خاطئة";
}
?>
```

العبارة Switch

```
Switch (VARIABLE) {
  CASE THING1 :
                Excute code ;
            break;
        CASE THING2 :
                Excute code ;
                break;
 Default;
                Excute code ;

  }
```

تقوم العبارة بنفس عملية العبارة if ولكن بهيكلية أسهل ومحببة أكثر وتتيح لك اختبار قيمة متغير وإجراء أكثر من اختبار عليه .

```
break;
```

تقوم بالخروج من عباره معينه مثل switch و if والذهاب الى الأوامر والعبارات التي بعدها .

```
EXIT;
```

تقوم بعملية الخروج من الكود نهائياً ولاتطبق أي أوامر بعدها ، وفي الأمثله التوضيحية التالية ستجد أن break; تخرج من العبارة فقط (Statement) بينما الـexit; تقوم بالخروج من كامل الكود (code).

مثال :

```
<?
$s=10;
if ($s=10) {
echo "number=10";
exit;
}
elseif ($s<11) {
  echo "number is less than 11"
{
echo "hello";
?>
```

مثال :

```
<?
$s=10;
if ($s=10) {
echo "number=10";
break;
}
elseif ($s<11) {
  echo "number is less than 11"
{
echo "Hello";
?>
```

Defualt;

إذا لم تصلح جميع الحالات (Cases) في العبارة (Switch) فسيتم تنفيذ الأوامر التي تقع بعد هذه الكلمة وهي تؤدي نفس عمل else تقريباً في العباره if .

مثال (1)

```
<?
$g= "ahmed";
Switch ($g) {
        Case "ahmed":
                Echo "مسموح ";
                Break ;
  Case "khaled " :
                Echo "ممنوع " ;
                Break ;
  Case "salem" :
                Echo "ممنوع " ;
                Break ;
   Case "Mohmed " :
                Echo "مسموح " ;
                Break ;
Default ;
        Echo "لقد ادخلت اسم غير صالح";
}
?>
```

مثال (2)

```
Switch ($g) {
        Case $g>50:
                Echo "كبير ";
                Break ;
  Case  40 :
                Echo "لاباس " ;
                Break ;
```

```
 Case ($g<15) :
                Echo "أطفال ممنوع " ;
                Break ;
  Case 30 :
                Echo "مسموح " ;
                Break ;
}
```

لاحظ أننا عند اختبارنا لنصوص نحتاج الى علامتي تنصيص مزدوجة وعند الارقام فاننا لانحتاج الي ذلك

تطبيق عملي

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم age.html

```
<html>
<form method=post action="age.php">
كم عمرك ؟
<br>
<input type="text" name = "g">
<input type=submit value="ارسال">
</form>
</html>
```

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم age.php

```
<?
 Switch ($g) {
        Case $g>50:
                Echo "كبير ";
                Break ;
  Case  40 :
                Echo "لاباس " ;
                Break ;
  Case ($g<15) :
                Echo "أطفال ممنوع " ;
                Break ;
   Case 30 :
                Echo "مسموح " ;
                Break ;
}
?>
```

الشرح

تقوم العبارة Switch باختبار قيمة متغير ما ويمكنك إجراء أكثر من افتراض عليه ويجب عليك كتابة الكلمة break; لكي تقوم بإيقاف تنفيذ العبارة switch فمثلاً لو قمت بكتابة الكود التالي :

```
<?
$g=40
Switch ($g) {
        Case $g<50:
```

```
                Echo "1 ";

  Case  40 :
                Echo "2" ;
        }
?>
```

فاذا ادخل المستخدم الرقم 40 فسيتم طباعة الرقمين واحد واثنين كلاهما وذلك لأنك لم تقم بإيقاف العبارة فاكملت التحقق وطبقت جميع العمليات المطلوبة .

التخلص من وسوم الـ html

إذا قمت بوضع مربع نص وأردت من المستخدم كتابة شي فيه فإنه يستطيع ادخال أي شي ولنفترض أنه كتب في مربع النص كالتالى :

```
I am <b>ahmed</b> ...
```

فسيقوم المتصفح بعرضها بعد معالجتها كالتالى :

```
I am ahmed ....
```

ولنقم بتطبيق عملي على ذلك

قم بفتح محرر النصوص واكتب الكود التالي واحفظه باسم htmlch.html

```
<html dir="rtl">
<form method=post action="html.php">
أدخل اسمك الكريم
<br>
<input type="text" name = "fname">
<input type=submit value="ارسال">
</form>
</html>
```

قم بفتح محرر النصوص واكتب الكود التالي واحفظه باسم html.php

```
<?
Echo " هذا هو الشكل الطبيعي للعبارة عند طباعتها";
Echo "<br>" . $fname;
 ?>
```

قم بوضع الملفات في مجلد السيرفر ثم قم بتشغيل الملف htmlch.html واكتب في مربع النص أي شي وضعه بين وسوم html

مثال :

```
I am <b><i>alfareees</b></i>
```

ستجد أنه قدتم التعامل مع الوسوم كـ html وليس كنص عادي ولكي تعرضها كنص عادي فإنك تقوم باستخدام الدالة

```
HtmlSpecialChars();
```

حيث أنها ستقوم بمعاملة كود الـhtml كنص عادي وطبيعي تماماً .

إذاً نقوم بتعديل ملف الـhtml.php ليصبح كالتالى :

```
<?
$fname = HtmlSpecialChars($fname);
Echo " هذا هو الشكل بعد استخدام الدالة";
Echo "<br>" . $fname;
 ?>
```

التكرارت والمصفوفات

لقد اخذنا في الدرس السابق شيئاً من أساسيات البرمجة وهو الدوال الشرطية وصناعة القرارات والآن نحن نتجه إلى شي يحب جهاز الكمبيوتر عمله وهو التكرارت والمصفوفات .

في الواقع قد يكون لديك يومياً شي تفعله بشكل مستمر مثل الإفطار في الصباح الباكر والنوم مساء ، انك تستمر على هذا الروتين دائماً .... نحن نسمي هذا الشي في لغة البرمجه التكرار .

هناك شي آخر يسمي المصفوفات ... في الواقع قد يحتوي درج مكتبك الخاصة بالكتب على عدة أدراج الدرج الاول منها يحتوي على الكتب الإسلامية والدرج الثاني منها يحتوي على الكتب الرياضية والدرج الثالث يحتوي على كتب الرياضيات ... أو لنفرض أنك مدرس في إحدي المدارس ولديك جدول للحصص ففي الحصة الأولى لديك مثلاً تدريس مادة الرياضيات .... والحصة الثانية لديك تدريس مادة العلوم والثالثة لديك تدريس مادة الكيمياء .... إن حصصك مرتبة بشكل معين مع أنها كلها تسمي حصص إلا أن كل حصة تختلف عن الأخري في المادة ! وهي مرتبه بشكل تصاعدي (الحصة الاولى ، الثانية ،الثالثة .....) .

نسمي هذه التقنية بالمصفوفات .... المصفوفات عبارة عن متغير اسمه ثابت ولها اكثر من قيمة وكل قيمة لها رقم معين ولكي تحصل على القيمة فانك تكتب المتغير ثم رقم القيمة التي فيه، لايشترط أن تكون هذه القيم متسلسلة فقد يكون هناك قيمتين ولكل قيمة رقم يختلف تماماً ويبعد كل البعد عن القيمه الثانية مثال رقم 1 و 258 كلاهما مختلف تماماً ويبتعد كل البعد عن الاخر .
إن دمج ميزة التكرارات مع المصفوفات يساعدك على توفير عدد الأسطر للكود ويساعدك على صنع أشياء عجيبة في أقل عدد ممكن من الأسطر .

التكرارات

التكرارات عبارة عن تكرار أمر معين بعدد معين من المرات ولقد اخذنا سابقاً الدوال الشرطية أو العبارات الشرطية بالأصح فوجدنا أن الكود الذي نكتبه في العبارات الشرطية لا تتنفذ إلا عندما يكون الشرط صحيحاً
أيضا التكرارت فهي تختبر الشرط فإذا كانت قيمته صحيحة فإنها تقوم بعمل الكود المطلوب ثم تقوم بإعادة اختبار القيمة فإذا كان صحيحاً فإنها تقوم بإعادة تنفيذ الكود وهكذا ، أما عندما لا يكون الشرط صحيحاً فإنها تتوقف عن تنفيذ الكود ويتم اكمال البرنامج بشكل عادي ... هناك ثلاثة أنواع من التكرارات .
إن أول دالة نقوم بأخذها في البداية هي الدالة while

التكرار while

لقد قمنا بأخذ التكرار while لأنه بسيط جداً وصيغه هذا التكرار هي :

```
While (condition  شرط  )
{
code
}
```

مثال :

```
<?
$d =10 ;
while ($d<15)
{
echo "$d <br>";
$d++;
}
?>
```

سيقوم الـPHP أولا بإعطاء المتغير $d القيمة 10 ثم يقوم بعد ببدء التكرار while فإذا كان الشرط صحيحاً (وهو أن المتغير أصغر من الرقم 15) فإنه يقوم بتنفيذ الكود الذي بين الأقواس وعمل هذا الكود أن يقوم بطباعة المتغير ثم يقوم بإضافة واحد على القيمة الموجودة في المتغير $d ثم بعد ذلك سيتم اختبار الشرط مرة ثانية فإذا كان صحيحاً فسيتم نفس العملية حتي يكون الشرط غير صحيح فيتوقف عندها التكرار ويتم إكمال الكود التي تقع بعد الاقواس .

إذا لم تقم بوضع حد للتكرار فلن يتوقف التكرار وقد يكون لانهائي ....
مثال :

```
<?
$d =10 ;
while ($d<15)
{
echo "$d <br>";
}
?>
```

سيتم طباعة الرقم 10 ولن يتوقف التكرار لأن الشرط صحيح دائماً وليس هناك مايوقفه بينما في الكود السابق استطعنا إيقاف الكود بسبب أننا كنا نضيف واحد على القيمة الموجودة في المتغير وكلما يتم اعادة اختبار الكود كل ما تتغير القيمة حتي يصبح الشرط غير صحيح بسبب أن $d أكبر من 15 .

التكرار do - while

هذا التكرار يعمل بنفس طريقه التكرار الأول إلا أنه يوجد بعض الاختلافات البسيطة وصيغته كالتالى :

```
do
code
while (condition شرط);
```

مثال :

```
<?
$f=15 ;
do
{
echo "$f";
$f ++
}
while () ;
```

سيقوم التكرار بتنفيذ السطر الموجود بين القوسين أولا ثم يقوم بتنفيذ بإختبار الشرط فإذا كان الشرط صحيحاً قام بإعادة العملية الموجودة بين القوسين وهي إضافة واحد على المتغير $f وهكذا

حتي يكون الشرط غير صحيح فيتم التوقف .. لاحظ أننا في التكرار الأول قمنا باختبار الشرط قبل صناعة أي عمل بينما في التكرار الثاني قمنا بتنفيذ الكود أولا ثم قمنا بإجراء الاختبار .

التكرار FOR

يختلف هذا التكرار عن سابقيه لكن وظيفته هي نفس وظيفتهما وهي تكرار الأوامر عند حصول شي معين

الصيغة :

```
For (counter عداد ; test value  اختبار القيمة ; set counter اداء عمليه على العداد )
{
code  شفرة
}
```

مثال :

```
<?
For ($u = 18 ; $u>10 ; $u--)
{
echo $u;
}
 ?>
```

يتكون هذا التكرار من ثلاثة أقسام .... القسم الأول نضع فيه متغير يحتوي على قيمة حيث سيبدأ التكرار العمل من عند هذه القيمة والقسم الثاني نكتب فيه الشرط الذي سيقوم التكرار بفحصه (والذي هو كالمعتاد اختبار لقيمة المتغير في القسم الاول) والقسم الثالث نضع فيه العمل الذي سيجري على المتغير عند كل تكرار ثم نقوم بكتابة كود التي سيقوم بتنفيذها التكرار بين القوسين .

كأننا نقول للـphp بشكل عامي أن يقوم في البداية بإعطاء المتغير $u القيمة 18 وقبل ان يقوم بتنفيذ الكود عليه أن يقوم بتحليل الشرط فإذا كان الشرط صحيحاً فإنه يقوم بإنقاص واحد من المتغير $u ويتم تنفيذ الكود حتي يصبح المتغير$u قيمته 9 فيقوم الـPHP آنذاك بالخروج من التكرار والذهاب الي الكود الذي يلى القوسين .

المصفوفات

لقد قمنا بتعريف المصفوفات سابقاً بشكل بسيط وحان الوقت الآن لنعرفها ونعرف كيفية عملها . المصفوفات عبارة عن متغير وهذا المتغير يحتوي على أكثر من قيمة أو عنصر (element) وكل عنصر له فهرسة (Index) تبدأ هذه الفهرسة من الصفر إذا لم تقم بتحديدها

مثال :

```
<?
$A[ ] = “alfareees”;
$A[ ] = 13;
?>
```

في هذا المثال سيقوم الـPHP بإعطاء الفهرسة تلقائياً فسيقوم بوضع الرقم فتصبح المتغير فهرسته كالتالى :

```
$A[0] = “alfareees”;
$A[1] = 13;
```

إننا لم نقم بإدخال هذه الأرقام من تلقاء أنفسنا ولكن الـPHP قام بوضعها مع أنه يمكننا أن ندخلها بشكل عادي فمثلاً لو كتبنا :

```
<?
$A[0]= “alfareees”;
$A[1] = 13;
?>
```

سيقوم الـPHP بأخذ الفهرسة المعتمدة ولن يضع أي فهرسة أخري يمكننا أيضا أن نكتب أي فهرسة ولانعتمد على الترتيب في الارقام .

مثال :

```
<?
$A[10 ] = “alfareees”;
$A[ 25] = 13;
?>
```

هل لاحظت أيضا أننا لم نقم بتعريف نوع متغيرات المصفوفة وقام الـPHP بتعريفها تلقائياً بدلاً منا فمرة استخدمنا قيمة حرفية ومرة استخدمنا رقماً ورغم ذلك فلم يقم الـPHP بعمل أي اعتراض إضافة إلى ذلك فإن الـPHP يقوم بتحديد عدد عناصر المصفوفة تلقائياً فهو يعرف مثلاً من المثال السابق أن عدد عناصر المصفوفة الكلي هو عنصرين .
يمنحنا الـPHP ميزة أخري وهي عدم التقيد بالأرقام في الفهرسة فمثلاً يمكننا استخدام حروف عادية .

مثال :

```
<?
$A[“a” ] = “alfareees”;
$A[“b” ] = 13;
?>
```

لاحظ أننا استخدمنا القيم الحرفية ولم يعترض الـPHP بتاتاً ويمكننا طباعة أي عنصر من عناصر المصفوفة بكل بساطة .

مثال :

```
<?
$r [“aa”] = “ahmed ali”;
$r [1] = 13273;
$r [20] = 13273;
echo $r[aa];
echo $r[20];
echo $r["aa"];
?>
```

لا فرق بين أن نكتب النص الحرفي (aa) بين علامتي تنصيص عند الطباعه وعند كتابته بدون علامات تنصيص ... سيقوم الـPHP بمعرفة ذلك تلقائياً .

يمكننا تعريف المصفوفات أيضا بطريقة أخري

```
$variable = array (elements) ;
```

مثال :

```
<?
$t =array (“ahmed”, “ali”, “salem”, “alfarsi”);
echo $t [0];
?>
```

يقوم الـPHP بإعطاء كل عنصر من عناصر المصفوفة رقم فهرسة فتصبح كالتالى :

الفهرسه Index	العنصر Element
0	Ahmed
1	Ali
2	Salem
3	alfarsi

إذن القيمة التي سيطبعها الـPHP في النهايه هي ahmed ، لاحظ أن الـPHP قام بإعطاء رقم الفهرسة وقام بالبدء من الصفر ولكن يمكننا جعل الـPHP يبدأ الفهرسة من الرقم واحد كالتالي :

```
<?
$r = array (1=>”ahmed”, “ali”,”salem”, “alfarsi”);
?>
```

عند تعريفك لرقم الفهرسة للقيمة الأولى سيقوم الـPHP بإعطاء أرقام فهرسة بشكل تسلسلي ، عندئذ ستصبح الفهرسة كالتالى :

الفهرسه Index	العنصر Element
1	ahmed
2	Ali
3	salem
4	alfarsi

هناك طريقة لتكون أيضا الفهرسة هي عبارة عن حروف :

```
<?
$r = array (“ss”=>”ahmed”, “sf”=> “ali”, “da”=>”salem”, “bv”=> “alfarsi”);
?>
```

عندئذ ستصبح الفهرسة كالتالى :

الفهرسه Index	العنصر Element
Ss	Ahmed
Sf	Ali
Da	Salem
Bv	Alfarsi

عندما نريد تغيير أي عنصر في المصفوفة فيمكننا عمل ذلك ببساطه .

مثال :

```
$r [ss]= "لمياء";
```

لاحظ أننا قمنا بتغيير القيمة من (ahmed) الى (لمياء) ....طريقة بسيطة أليس كذلك :)

قراءه المصفوفات واستخراج القيم

تكلمنا سابقا عن التكرار For
يمكننا استخراج عناصر مصفوفة وطباعتها في بساطة وتوفير وقت عن طريق التكرارات

لنفرض أن لديك هذه المصفوفة :

```
<?
$people =array ("ahmed", "ali", "salem", "alfarsi");
?>
```

واردت أن تطبع أسماء جميع الاشخاص المتواجدين فيها
أولاً نحن نعرف أن المصفوفة إذا لم نقم بتعريف رقم فهرسة لها فإن الـPHP يقوم ببداية فهرستها من الصفر وعلى ذلك فان رقم العنصر الأول 0 ورقم العنصر الرابع 3 ... على ذلك يمكننا بكل بساطه كتابة الكود التالي الذي يقوم بطباعة المصفوفة كالتالى :

```
<?
$people =array ("ahmed", "ali", "salem", "alfarsi");
echo "$people[0]. <br>";
echo "$people[1]. <br>";
echo "$people[2]. <br>";
echo "$people[3]. <br>";
?>
```

لنفرض أن لديك ثلاثين أو ثلاثة آلاف اسم في مصفوفة ألن تبدو هذه الطريقة متعبة قليلا !!!
هناك طريقة أخري وهي عن طريق التكرارات .
لنفرض أننا أردنا كتابة تكرار يقوم بطباعة الارقام من واحد الى عشرة فإننا نستطيع كتابة التكرار بالشكل التالى :

```
<?
For ($I=1;$I<11;$I++)
{
Echo "$I <br>";
}
?>
```

والآن لنقل أننا نريد طباعة الأربعة عناصر في المصفوفة كل ماعلينا هو إجراء عملية بسيطة على الكود لكي يتم ذلك :

```
<?
$people =array ("ahmed", "ali", "salem", "alfarsi");

For ($I=0;$I<4;$I++)
{
Echo "$people[$I] <br>";
}
?>
```

لاحظ أننا بدأنا العداد بالقيمة صفر ثم اشترطنا أن يكون أقل من 4 لأن آخر عنصر في المصفوفة رقم فهرسته 3 ثم قمنا بجعله يزداد بقيمة 1 لأننا نريد طباعة جميع عناصر المصفوفة وقمنا بوضع رقم العداد في خانة الفهرسة وعلى ذلك سيتم في كل تكرار طباع عنصر المصفوفة الذي فهرسته تساوي رقم العداد .

لقد تكلمنا سابقاً في درس النماذج عن إخراج القيم من قائمة على شكل مصفوفة .

مثال :

```
<form action = "array.php" method = post>
ما هو مشروبك المفضل ؟
<br>
<select name = "a[]" multiple>
<option>شاي</option>
<option>قهوة</option>
<option>كابتشينو</option>
<option>توت</option>
<option>برتقال</option>
</select>
<br>
<input type=submit value = "لذيذ" >
</form>
```

في ملف الـarray.php اكتب :

```
<html>
 لقد قمت باختيار التالى :
<?
For ($I=0;$I<4;$I++)
{
Echo "$a[$I] <br>";
}
?>
</html>
```

لقد عرضنا في القائمة خمسة عناصر ... لاحظ أننا وضعنا في اسم المتغير للقائمة قوسين [ ] لكي يتعرف الـhtml على أنه سيتم تخزين البيانات تلقائياً بعد ذلك قام الـPHP بفهرسة العناصر التي تم إرسالها من قبل العميل سواء كانت ثلاثة أو أربعة ولكنها بالطبع لن تزيد على خمسة .... على ذلك سيكون آخر رقم تنتهي به المصفوفة هو 4 .

أتوقع أنك الآن بدأت تحب المصفوفات .... يمكننا صناعة القائمة عن طريق المصفوفة أيضاً ....

مثال :

```
<form action = "list.php" method = post>
ما هو مشروبك المفضل ؟
<br>
<select name = "s" >
<?
$shrab =array("شاي","قهوة","كابتشينو","توت","برتقال");
For ($k=0;$k<4;$k++)
```

```
{
echo "<option>".$shrab[$k]."</option>";
}
?>
</select>
</form>
```

عند اختيار المستخدم للقيمة سيتم وضعها في المتغير $s يمكنك مراجعة درس النماذج لكي تفعل ذلك ، هذا المثال يقوم بصناعة مصفوفة للمشروبات ثم يقوم بإخراجها في قائمة مما يوفر علينا الوقت في كتابة الكود فلو كان لديك مثلا حوالى مئة دولة فيمكنك مثلا وضعها في مصفوفة وبعد ذلك بناء القائمة التي سوف تقوم ببناء القائمة التي ستحتوي على هذه الدول عن طريق المصفوفات والتكرارات .

قم بحفظ التغييرات في ملف إمتداده php وقم بكتابة الملف list.php اعتمادا على معلوماتك السابقة في درس النماذج .

دوال المصفوفات

الدالة key

لنفرض أن لدينا مصفوفة مكونة من عنصرين :
مثال :

```
$s= array ("على","محمد");
```

الآن لنضف إليها هذه السطور

```
<?
$s= array ("على","محمد");
$t=key ($s);
echo $t;
?>
```

يقوم الأمر key بإيجاد رقم الفهرسه (index) العنصر النشط حالياً .... وهو الرقم صفر حيث أننا لم نضع فهرسة وهذه هي الفهرسة التي وضعها الـPHP تلقائياً عندما لم نضع فهرسة ... قد تحيرك كلمة النشط لكن ستعرف أننا نستطيع التجول بين عناصر المصفوفة لاحقاً .

قد يكون رقم الفهرسة حروف أو كلمات

مثال :

```
<?
$s= array ("ع"=>"على","م"=>"محمد");
$t=key ($s);
echo $t;
?>
```

الدالة current()

تقوم الدالة current بإيجاد القيمة لعنصر المصفوفة الحالى (index value) .

مثال :

```
<?
$s= array ("ع"=>"على","م"=>"محمد");
$p=current ($s);
echo $p;
?>
```

في المثال السابق قمنا بإيجاد القيمة الحالية للعنصر النشط .... لاحظ أننا أوجدنا بالأمر key رقم الفهرسة بينما أوجدنا بالأمر current القيمة للعنصر المفهرس .

كيف يمكننا تنشيط العناصر الاخري للمصفوفه ؟!

يمكننا ذلك عن طريق الدالتين next() و prev اللتان تقومان بالتجول بين عناصر المصفوفة .....

لنفرض أن لدينا مصفوفة تتكون من ثلاثة عناصر

مثال :

```
<?
$s= array ("ع"=>"على","م"=>"محمد","ا"=>"احمد");
echo key($s)."<br>";
echo current($s) ."<br>";
?>
```

لقد قمنا في هذا المثال بطباعة قيمة رقم الفهرسة للعنصر الحالى وقيمته (اقصد برقم الفهرسة الحرف(ع) واقصد بالقيمة (على) .... لنقم الآن بالتجول بين عناصر المصفوفة ولنر نتيجة الطباعة .

مثال :

```
<?
$s= array ("ع"=>"على","م"=>"محمد","ا"=>"احمد");
next($s);
echo key($s)."<br>";
echo current($s) ."<br>";
?>

<?
$s= array ("ع"=>"على","م"=>"محمد","ا"=>"احمد");
next($s);
next($s);
echo key($s)."<br>";
echo current($s) ."<br>";
?>
```

لاحظ أننا كتبنا الدالة next() قبل أن نقوم بالإنتقال لكي يتم تنشيط العنصر الثاني في أول مثال ولتنشيط العنصر الثالث في ثالث مثال (ولاحظ أننا كتبنا next() مرتين) .

يمكننا الرجوع لتنشيط العنصر السابق بوضع الدالة prev() فمثلاً يمكننا تعديل المثال التالى :

```
<?
$s= array ("ع"=>"على","م"=>"محمد","ا"=>"احمد");
next($s);
next($s);
prev($s);
echo key($s)."<br>";
```

```
echo current($s) .”<br>”;
?>
```

فسيقوم الـPHP في هذه الحاله طباعة العنصر الثاني وليس الثالث لأنه تم التراجع خطوه عن طريق prev()

ماذا سيحصل إذا قمنا بإضافة عنصر على مصفوفة غير محدودة الفهرسة ؟!
لنفرض أن لدينا مصفوفة وأضفنا إليها عنصر غير محدد الفهرسة . مثل :

```
<?
$s= array (12=>“على”,5=>”محمد”,44=>”احمد”);
$s[ ]= “هشام”;
Next($s);
Next($s);
Next($s);
Echo key ($s).”<br>”;
Echo current( $s ).”<br>”;
?>
```

سيقوم الـPHP ببساطة بالبحث عن أكبر رقم فهرسة وبعد ذلك يبدأ بإعطاء الفهرسة تسلسلاً بعده فإذا كانت أرقام الفهرسة حروفاً بدأ من الصفر في اعطاء الرقم .. ولاحظ في هذا المثال بأنه قام بإعطاء العنصر الرقم 45 لأن أكبر عنصر في المصفوفة هو 44 وعلى ذلك قام بإعطاء الأرقام تسلسلاً بعد هذا الرقم .

الدالة List و Each

لنفرض أنك قد قمت بصنع مصفوفة غير مفهرسة بالترتيب
مثال :

```
<?
$s= array (12=>“على”,5=>”محمد”,44=>”احمد”);
?>
```

على ذلك دعنا نخبرك بخبر سار وهو أنك تستطيع أن تجعل حياتك مع PHP أسهل مع حياتك مع نفسك !

```
While (list( ارقام الفهرسة Index,Element value قيمة العنصر )=each (array)
```

تستطيع بواسطة هذه الدالتين وعن طريق التكرار while استخراج جميع العناصر الموجودة في المصفوفة

```
While (list($e,$r) = each ($s))
{
echo “<br> $e<br> $r”;
}
```

أولاً أنت تقوم بتسمية متغيرين واحد منهما لرقم الفهرسة ($e) والثاني للعنصر ($r) ويمكننا تسميتهما بأي اسم وفي حالة ما إذا أردنا عرض العنصر فقط أو معرفة العنصر فقط فيمكننا حذف ($e) ولكننا لانحذف الفاصلة

```
While (list(,$r) = each ($s))
{
echo “<br> $e<br> $r”;
}
```

لنعد الى المثال الذي فيه رقم الفهرسة والعنصر ... سيقوم التكرار بوضع رقم الفهرسة (الذي قد يكون نصياً) في المتغير $e وسيضع قيمة العنصر الذي رقم الفهرسة له هو $e في المتغير $r ثم سيقوم بطباعة العناصر حتي ينتهي منها جميعها ...

ملاحظة مهمة : إذا لم تقم بتعريف فهرسة للمصفوفة (حروف أو أرقام أيا كان) فسيتم استخدام العناصر عندما يطلب التكرار الفهارس .
مثال :

```
<?
$e=array("fsda","terhfgfd","tewr");
While (list ($I,$V)=each($e))
{
echo "<br>$e[$I]";
}
?>
```

لاحظ معH ننا طلبنا طباعة الفهرسة (index) إلا أنه تم أخذ العناصر(elements) بدلاً من الفهرسة

يمكننا بواسطة هذه الدالة صناعة أشياء مفيده وكمثال لذلك لنفرض أن لدينا مصفوفة أرقام هواتف ونريد أن نخرج هذه المصفوفة على جدول html فسنستطيع صناعة هذا الجدول عن طريق التكرار السابق بكل سهولة .
مثال :

```
<table align='center' dir = "rtl" border="1" width="100%" cellspacing="0"
bordercolorlight="#000000" bordercolordark="#000000" bordercolor="#000000">
<tr>
   <td align='center'>الاسم</td>
   <td align='center'>رقم التلفون</td>
  </tr>
<?
$s = array (658=>"465873 ، "سالم"<=456546، "عادل);
While (list($e,$r) = each ($s))
{
echo "<tr><td align='center'>". $r . "</td><td align='center'>" .$e . "</td></tr>";
}
?>
</table>
```

أرايت كيف استخرجنا جميع أرقام التلفونات في جدول بواسطه تكرار بسيط ، يمكنك صناعة الأكثر واختصار الكثير من الوقت على ذلك إذا كانت المصفوفة تحتوي على المئات من الأرقام بواسطة هذا الكود بدلاً من أن تكتب الكود على شكل html وتكتب البيانات وتتعب نفسك .

يمكنك أيضا معرفة عدد العناصر في مصفوفة معينة إذا كنت تريد معرفة عددها وذلك بالطريقة التالية :

```
<?
$s= array (12=>"على",5=>"محمد",44=>"احمد");
$S=0;
While (list($E,$r) = each ($s))
{
$S++;
}
ECHO " عدد عناصر المصفوفه" . $S++;
?>
```

فرز المصفوفات

هناك العديد من الدوال التي يوفرها لنا الـPHP لفرز المصفوفات . نحن سنأخذ نظرة عن الخمسة دوال الأكثر استخداماً :

الدالة Sort()

هذه الدالة من أساسيات فرز المصفوفات وهي جداً أساسية وهي تقوم بأخذ محتويات المصفوفة ومن ثم تقوم بفرزها هجائياً اعتماداً على الأحرف الكبيرة أولا ثم الصغيرة .. تتطلب هذه الدالة اسم المصفوفة التي سيتم عليها الفرز

```
Sort (ArrayName);
```

إذا قمنا بإنشاء مصفوفة بالشكل التالى :

```
$NaNo=array (“ali”,“salem”,“hythem”,“Khaled”,“Ammar”,“Hesham”);
```

فإذا أردنا فرزها عن طريق الدالة sort() فإننا نقوم باستخدامها كالتالى :

```
<?
$NaNo=array (“ali”,“salem”,“hythem”,“Khaled”,“Ammar”,“Hesham”);
sort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo “<br> $e<br> $r”;
}
?>
```

لاحظ أنه عند تنفيذك للمثال ستجد أن الـPHP قام بالفرز اعتماداً على الأحرف الكبيرة أولاً ثم قام بالفرز بعدها اعتماداً على الأحرف الصغيرة .

الدالة Arsort()

هذه الدالة تعمل نفس عملية الدالة sort() ولكن هناك اختلاف بسيط فمثلاً لو كتبنا المصفوفه كالتالى :

```
$NaNo=array ( “ad”=>”ahmed”, “kh”=> “khaled”);
```

وأردنا فرزها وطباعة الفهارس والقيم كما في المثال التالى :

```
<?
$NaNo=array ( “ad”=>”ahmed”, “kh”=> “khaled”);
sort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo “<br> $e<br> $r”;
}
?>
```

قارن ناتج المثال السابق مع هذا المثال :

```
<?
$NaNo=array ( “ad”=>”ahmed”, “kh”=> “khaled”);
asort($NaNo);
While (list($e,$r) = each ($NaNo))
{
```

```
echo "<br> $e<br> $r";
}
?>
```

اعتقد انك قد عرفت الفرق ففي المثال الاول قامت الدالة sort باستبدال الحروف بأرقام في الفهرسة أما في المثال الثاني فقد تم وضع الحروف كما هي وتم فرزها كما تفعل الدالة sort في الفرز .
باختصار لايوجد فرق بين sort و asort إلا في أن الدالة sort تستبدل فهرسة الحروف بأرقام .

الدالة ()Rsort و arsort

تقوم بنفس عمل sort و asort ولكن بشكل عكسي جرب الأمثلة التالية :
مثال :

```
<?
$NaNo=array ( "ad"=>"ahmed", "kh"=> "khaled");
rsort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
```

مثال :

```
<?
$NaNo=array ( "ad"=>"ahmed", "kh"=> "khaled");
arsort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
```

ستجد أن الدالة rsort تقوم بنفس عملية الدالة sort ولكن بشكل عكسي
أيضاً الدالة arsort تقوم بنفس عملية asort ولكن بشكل عكسي .
يمكنك استعمال كل هذه الدوال في الفرز مع الحروف العربية (إذا كان السيرفر يدعم اللغة العربية)
قم بتطبيق المثال التالى :

```
RSORT()
<?
$NaNo=array ( "ad"=>"سالم", "kh"=> "احمد");
rsort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ARSORT()
<?
$NaNo=array ( "ad"=>"احمد", "kh"=> "أحمد");
arsort($NaNo);
```

```
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ASORT()
<?
$NaNo=array ( "ad"=>"هاشم", "kh"=> "جمال");
asort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
SORT()
<?
$NaNo=array ( "ad"=>"هاشم", "kh"=> "جمال");
sort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
```

الدالة ksort

تكلمنا سابقاً عن طريقة فرز المصفوفات ولكن نريد أن نلفت نظرك أننا كنا نعتمد على العنصر في الفرز (element) ولكن هذه الدالة تقوم بالاعتماد على رقم الفهرسه في الفرز (index)

sort

مثال :

```
<br>--------<br>
asort()
<?
$NaNo=array ( "ad"=>"هاشم", "kh"=> "جمال");
asort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ksort()
<?
$NaNo=array ( "ad"=>"هاشم", "kh"=> "جمال");
ksort($NaNo);
While (list($e,$r) = each ($NaNo))
{
```

```
echo "<br> $e<br> $r";
}
?>
```

لقد اعتمد الـphp على index ولم يعتمد على الـelement في الفرز .

دوال المصفوفات الإضافيه

هناك الكثير من الدوال التي يمنحنا إياها الـPHP للتعامل مع المصفوفات والتي لايكفي الوقت لذكرها الآن سنقوم بشرح أهم دالتين والمستخدمة بكثره وهي **array_push()** و **array_pop()**

لنفرض أننا قمنا بإنشاء مصفوفة بالشكل التالى :

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
?>
```

وأردنا أن نضيف عنصر جديد لها فقمنا بالتالى :

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
$saher[ ]=”Alfarees”;
?>
```

انظر إلى العنصر الأخير الذي سيعطيه الـPHP رقم الفهرسة (index) وسيكون رقم فهرسته هو 86 .
نريد أن نلفت نظرك بأننا نستطيع عمل إضافة لعنصر على المصفوفة بطريقة أخري وهي عن طريق الدالة array_push() كالتالى :

```
array_push (ArrayNameاسم المصفوفه,Elemnt1, Elemnt2, Elemnt3,.....)
```

نضع في القسم الأول من الدالة اسم المصفوفة التي نريد إضافة العنصر لها ونضع في القسم الثاني عنصر واحد أو أكثر وهي التي سيتم إضافتها للمصفوفة .

مثال :

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
array_push ($saher,Alfarees)
?>
```

مثال :

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
array_push ($saher,Alfarees,salem,sameer,thamer)
?>
```

ولو أردنا حذف مثلاً عنصر من المصفوفة فإننا نقوم بتعريف المصفوفة من جديد أو يمكننا استخدام الدالة array_pop التي تقوم بحذف آخر عنصر من المصفوفة والتي تتطلب فقط اسم المصفوفة

```
Array_pop(ArrayName اسم المصفوفه)
```

مثال :

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
array_pop($saher)
?>
```

سيتم حذف العنصر hajeer من المصفوفة ولن يكون في المصفوفة غير ثلاث عناصر .

Implode و Explode

تقوم هذه الدالتين باقتصاص قيمة معينة من مصفوفة أو نصوص وتقوم بإضافة قيمة معينة على مصفوفة أو نصوص .

الدالة Implode

تقوم بإضافة قيمة على بين عناصر المصفوفة .

مثال :

```
<?
$stng =array ("ahmed", "salem", "ali", "alfarsi");
$r =implode ("H",$stng);
echo $r;
?>
```

الدالة explode

تقوم بحذف قيمة من مصفوفة وذلك لايعني حذف عناصر من المصفوفة .

مثال :

```
<?
$stng =array ("ahmed", "salem", "ali", "alfarsi");
$r =implode ("-",$stng);
echo $r;
$r= explode ("-",$stng);
echo $r;
?>
```

HTTP_GET_VARS و HTTP_POST_VARS

هذه ليست متغيرات بل مصفوفات ، نعم هذه مصفوفات ولكن في ماذا نستخدمها ولماذا ؟
في الواقع تحدثنا في الدرس السابق عن طريقة التعامل مع النماذج والحصول على البيانات من المستخدم وتكلمنا عن أسلوبين لنقل البيانات وهما GET و POST

عندما تصل البيانات محفوظة في متغيرات إلى صفحة الـPHP فإنه يقوم بتعريفها تلقائياً ويمكنك طباعة المتغيرات وقيمها مباشرة من غير تعريف .... ولكن هذه الميزة في الـPHP يمكن إلغاؤها عن طريق الملف PHP.INI وذلك بإغلاق ميزة register_globals
وذلك بوضع off بدلا من on

الوضع الافتراضي لها هو on ولكن تستطيع إغلاقها وقد تكون مستاجراً عند مزود خدمة ويب وسيط فيقوم بإغلاق هذه الميزة من باب الحماية ليس إلا .... لاتقلق يمكنك الحصول على البيانات فهي ما زالت موجودة ولكن يجب عليك أن تقوم بإستخدام هذه المصفوفتين لكي تستخرج البيانات .

لنفرض أنك اشتركت عند مزود ويب وكان قد أغلق ميزة (register_globals) حسناً لنفرض أنك قد صنعت نموذجاً يستخدم مربع نص ويحفظ قيمته في متغير اسمه Dorrah ثم بعد ذلك يقوم بإرسال هذه القيمة باستخدام الأسلوب GET إذاً سيكون جزء من الكود في الصفحة الأولى والتي تحتوي على النموذج كالتالى

```
<form method =get action = "try.php">
!! ماهو اسم الطفل الذي استيقظ به العالم الاسلامي من غفلته قبل عده شهور
<br>
<input type=text name = "Dorrah">
<br>
```

في الملف الثاني(try.php) سنقوم بكتابة الجزء الذي سيقوم بطباعة القيمة كالتالى

```
<?
Echo HTTP_GET_VARS["Dorrah"];
?>
```

لاحظ أننا لم نستخدم $ ولكن إذا أردنا الإحتفاظ بقيمة المتغير في متغير آخر فيمكننا ذلك بشكل عادي كالتالى :

```
<?
$Dorrah= HTTP_GET_VARS["Dorrah"];
?>
```

طريقه بسيطة .... أليس كذلك ولكن .... لنفترض أن مزود خدمة الويب لديك حريص جداً ولذلك فقد ألغي أيضا ميزة استقبال هذه القيم في المصفوفات .... يمكنه ذلك في ملف الـphp.ini في اعدادات الـtrack_vars الذي يقوم بمنع السيرفر من استخدام هذه المصفوفات (هذه الميزة يمكن إلغاؤها في php4) .... على ذلك انصحك بإرسال رسال تذمر وشكوي إلى مزود الخدمة لديك .. تعلن فيها أن الأمر اصبح لايحتمل .

مصفوفه متعدده الابعاد

يمكنك صناعة مصفوفات بداخل مصفوفات على حسب ماتحتاجه في معلوماتك الرياضية فقد تحتاج مثلاً إلى إنشاء أشياء معقدة (ومقلقة نفسياً) نريد أن نخبرك على أية حال أنه يمكنك صناعة المصفوفات المتعددة الأبعاد ويمكنك استخدام حتي مائة مصفوفة متداخلة ولكن يجب أن تراعي حجم الذاكرة المستخدمة في السيرفر لديك (وعلى كل حال إن استطعت أن تقوم بالتركيز في صناعة عشر مصفوفات متداخلة بدون أي مشاكل أو مرض نفسي أو .... فأنت تستحق جائزة) .

يمكننا كتابة مصفوفة متداخلة كالتالى :

```
<?
$mon= array (1=>array ("sharkeh al-jafali",154786) ,2 => array ("salem almazen",1257)
);
while (list($personnum) =each ($mon))
```

```
{
echo ("<br>$personnum<br>");

while (list(,$phone)=each ($mon[$personnum]))
    {
     echo ("$phone");
     }
}
?>
```

الشرح

هذا المثال قد يكون غامضاً جداً لكن فكرته بسيطة أولاً افترض أنك تعلم عن list..each جيداً وتعرف صيغة التكرار الذي يستخدمهما .
الآن لدينا مصفوفة تتكون من رقمين للفهرسة هذين الرقمين كل واحد منهما عنصره عبارة عن مصفوفة هذه المصفوفة تحتوي على عنصرين (ولنتناسي أنهما يحتويان على أرقام فهرسة) وهما اسم شخص ورقم هاتفه .

في أول خطوة :

```
while (list($personnum) =each ($mon))
{
echo (“<br>$personnum”);
```

قمنا بإخراج رقم الفهرسة الأساسي للمصفوفة والذي يعتبر هو الرقم التسلسلي للاشخاص أصحاب الهواتف ومن بعد ذلك يقوم بطباعة هذا الرقم التسلسلي ويبدأ من سطر جديد .

في الخطوة الثانية :

```
while (list(,$phone)=each ($mon[$personnum]))
    {
     echo ("$phone");
     }
```

نقوم بإخبار الـPHP بطباعة العناصر الذي تحتويها المصفوفة التى تم طباعة رقم فهرستها ، ولاحظ (,$phone) أنها تشير إلى عناصر مصفوفة وليس فهارسها لأننا تجاهلنا فهارس المصفوفة الداخلية .
لاتقلق الأمر سهل ولكنه يحتاج الى تدرب فقط ، وعليك أن تتدرب وصدقني أنني حاولت ان أبسط المثال من أجلك ... أتمني أن تكون قد فهمت .

تطبيق عملي

افتح محرر النصوص لديك واكتب الكود التالي :

```
<?
Echo "<form method =post action = 'exam2.php' " ;
$boy=array ("أحمد" , "خالد", "سعد","حسن");
while (list(,$Name) = each ($boy))
{
echo " ماهي السنة الدراسية لـ $Name ؟";
Echo "<select name = 'school[]'>
<option>اول ثانوي</option>
```

```
<option>ثاني ثانوي</option>
<option>ثالث ثانوي</option>
</select>";
echo "<br><br>";
echo "<input type =hidden name =boy[] value ='$Name'>";
}
echo "<input type =submit ></form>";
?>
```

احفظ الكود باسم exam.php

افتح محرر النصوص واكتب الكود التالي واحفظه في ملف باسم exam2.php

```
<html dir = "rtl">
<?
While (list($I,$V)=each($school))
{
    $friendschool[] = $school[$I].$boy[$I];
}
asort ($friendschool);
While (list ($I,$V)=each($friendschool))
{
echo "<br>$boy[$I]"."  ".$school[$I];
}
?>
```

قم بتشغيله بعد نقله لمجلد السيرفر

الشرح

الذي قمنا به في المثال السابق هو أننا قمنا بإنشاء مصفوفة لعده أشخاص ($boy) ونريد أن نعرف مرحلهم الدراسية في الثانوية فأنشأنا لكل طالب قائمة منسدلة بواسطة التكرار (list-each) بصناعة قوائم منسدلة وحقول مخفية يتم تخزين قيم الحقول (التي تحتوي على أسماء الأشخاص) في المصفوفة ($boy) وسيتم تخزين نتائج كل القوائم في مصفوفة ($school) وبعد أن يختار المستخدم الإجابات التي تناسبه وارسال البيانات سيتم استقبال المصفوفة التي فيها نتائج القوائم المنسدلة ($school) واستقبال المصفوفة التي فيها أسماء الأشخاص ($boy) ومن ثم يتم إنشاء مصفوفة جديدة باسم $friendschool[] ويؤخذ منها معلومات المصفوفتين ويتم دمجها فيها ومن ثم يتم بتكرار آخر طباعة عناصر المصفوفتين $boy و$school .

تكرار foreach

هذا التكرار هو من الأشياء الجديدة في الـphp4 وهو يساعدك على معرفة عناصر مصفوفة معينة أو طباعة محتوياتها .

```
Foreach ($ArrayName As $ArrayItem)
{
        code شفره
}
```

مثال :

```
<?
 $T= array (a=>"ahmed " , b => "basem", c=>"car")
```

```
Foreach ($T As $A => $r)
{
     echo $A .”-----”. $r;
}
?>
```

الدالة count

تقوم بحساب عدد العناصر الموجودة في المصفوفة

مثال :

```
<?
$c=array("a","b","c");
$v=count($c);
echo $v;
?>
```

ترتيب الكود البرمجي

تعلمنا في الدروس السابقة أساسيات من أساسيات البرمجة واعطينا مثال عن الروتين في الحياة اليومية وهو أن تقوم بعمل شي أكثر من مرة في الحياة اليومية مثل شرب الشاي أو شرب القهوة وغير ذلك ، درسنا اليوم يتكلم عن ترتيب الكود ويتكلم تقريباً عن نفس فكره الروتين اليومي فأنت في حياتك تكرر بعض الأعمال بشكل روتيني
وقد تكون مللت الروتين فأحضرت شي 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 الأرضيه وترتيب المكتبة وترتيب غرفة النوم و ... و ... الخ) عند ذلك فإنك تبحث عن طريقة عملية لكي يتم انجاز هذه المهمة في أسرع وقت فتقوم بتقسيم هذه المهمة الكبيرة على عده أقسام (التنظيف ،الترتيب ، الغسيل ،.....) ثم تقوم باستدعاء أطفالك وفلذات اكبادك وتقسم على كل واخد 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 شي معين بواسطتها ، تقوم الدالة بأخذ قيم وتسمي (arguments معطيات) كمدخلات ،، ثم تقوم بعمل بعض التعديلات على هذه المدخلات وتقوم بإخراج قيمة أخري في أكثر الأحيان تقوم الدالة بأخذ القيم ووضعها في متغيرات أخري تسمي بالـ(parameters) لكي يتم اجراء العمليات عليها داخل الدالة وهذه المتغيرات لاتعمل خارج الدالة أي أنها متغيرات خاصه بالدالة فقط ! ...في دروسنا السابقه قمنا باستخدام دوال عديده مثل دوال فرز المصفوفات ودوال ايجاد نوع البيانات ،،، هذه المرة سنقوم ببناء دوالنا الخاصة بنا ،، ومن صنعنا نقوم باعطاءها المعلومات والبيانات وهي تقوم باجراء العمليات عليها ومن ثم اخراج الحلول ...

تعريف واستدعاء الدوال

لكي تقوم بتعريف دالة فإنك تقوم بكتابة الكلمة function متبوعة باسم الدالة والبارمترات الازمة والتي سيتم اجراء العمليات عليها بين قوسين ومن ثم تقوم بكتابة الكود الازم 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 بكتابه الكود الذي سوف يقوم بالمطلوب بين القوسين بدلاً من function code

دعنا الآن نقوم بكتابة دالة من إنشاءنا والتي تقوم باجراء عملية الجمع على متغيرين وسنقوم بتسمية الدالة باسم sumnoraml وهو اسم من تأليفنا ويدل على وظيفة وهدف الدالة ويمكن أن تقوم بتمسية الدالة باي اسم تريده ولست مجيراً بكتابه اسم معين

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

نقوم في هذه الدالة بإجراء عملية إضافة 100 على المتغير أو القيمة التي يتم تمريرها .

Return

يجب أن نضعها في نهاية كل دالة ، نستخدم هذه الكلمة لكي نقوم بإعلام الداله ان وظيفتها انتهت وايضا نستخدمها إذا كان لدينا أكثر من قيمة ونريد أن نقوم بإخبار الـPHP ماهي القيمة التي سيتم اعتمادها ففي مثالنا هذا أردنا إخبار الـPHP بأن يقوم بأخذ المتغير $a بانه هو القيمة النهائية مع أنه لو لم نضع المتغير فسيتم اعتباره هو الناتج النهائي لانه لايوجد متغير اخر تم عليه أي عمليات

الذي اقصده أننا لو كتبنا الكود بالشكل التالى :

```
<?
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 التالى :

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

هنا يجب تحديد أي المتغيرين سيكون هو القيمة النهائيه للدالة .

شرح الداله (sumnormal)

تقوم الدالة التي صنعناها بأخذ قيمتين ومن ثم فإنها تقوم بزياده العدد الذي يتم تمريره 100
ولكي نقوم بإخراج نتيجة الدالة فإننا ببساطة نسطيع ذلك باجراء أحد الأمرين echo أو print .
مثال :

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

لقد قمنا بتمرير رقم بدلاً من المتغير ويمكننا أيضا 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 معني ذلك أننا لانستطيع استخدام متغيرات بنفس الاسم المذكور في الدالة فيمكننا مثلاً كتابة نفس اسم المتغير بدون حصول أي مشاكل كالتالى :

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

يمكننا أيضا 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 ولايوجد بينهما اختلاف سوى أن الدالة echo قديمة وهي الأصل أما الدالة print فقد تم إنشاؤها في php4 ولايوجد أي فرق بينهما اطلاقا .
مثال :

```
<?
Print "احمد";
?>
```

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 آخرها أي أنه لافرق بين :

```
<?
//لاحظ اننا قمنا بتعريف الداله اولا ثم استدعاءها
Function fares($d)
{
print "alfareees@hotmail.com";
}

fares($d) ;
?>
```

وبين :

```
<?
//  لاحظ اننا قمنا باستدعاء الداله اولا ثم تعريفها
fares($d) ;

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

يمكنك أيضا عدم وضع متغيرات في الدالة كالتالى :

```
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 او عوامل او متحولات (سمها كما شئت) .

تمرير القيم الى الدالة

هناك نوعين من تمرير القيم

1 - تمرير القيمه مباشره الى الداله (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 .

2 - تمرير القيمه عن طريق المرجع (passing by reference)

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

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

اعداد قيمة افتراضيه للدالة

تستطيع أن تجعل الـPHP4 يقوم بإدراج قيمة إفتراضية 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 40 مباشرة .
أما إذا تم تمرير قيمة أو متغير فإنه سيتم العمل بالقيمة التي تم تمريرها بدلاً من القيمة الإفتراضية
مثال :

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

مدي 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 محلى
$s = "progrramer";
}
echo $r ;
ala($s);
echo $s;
?>
```

مثال :

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

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

مثال :

```
<?
// هذا متغير عام
$r = "salem";
function ala($s)
{
//هذا متغير محلى
$s = "programmer";
}
//استطعنا طباعته بشكل مباشر
echo $r ;
ala($s);
// يجب استخدام الداله لكي يتم طباعته
echo ala($s);
?>
```

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

لكي نقوم بجعل المتغير الذي بداخل الدالة متغيراً عاماً فيمكننا ذلك بإحدي الطريقتن 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 سهوله .

الطريقة الثانيه : هي أن نستخدم المصفوفة $GLOBALS التي تستخدم في PHP لتعريف المتغيرات العامة أيضا

مثال :

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

```
echo $s;
?>
```

المتغيرات المستقره (static variable)

اقصد 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 فانه يحتفظ بقيمته حتي 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 الاساس التي 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 ماتشاء ثم احفظه باسم a.txt
قم بانشاء ملف php واكتب فيه ومن ثم احفظه باسم b.php

```
<?
Include (“a.txt”);
?>
```

انقلهما الى 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 استدعاءها .

داله تلوين الكود

هل رايت مواقع تقوم بتلوين الكود بشكل مذهل مثل موقع zend ؟.... الأمر بسيط كل ماعليك أولا قم بوضع الكود في ملف نصي وسمه باي اسم (مثلا file.txt) وبعد ذلك قم باستخدام الدالة Show_source
مثال :

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

تتبع وتصيد ومنع الاخطاء
(avoiding and handling errors)

إن مصطلح الـdebbug هـو مـن المـصطلحات الـشائعة والـشيقة فـي عـالم البرمجـة ، هـذا المـصطلح يشير إلى كيفية إصلاح أخطاء البرنامج وتوقعها قبل حدوثها ، هناك أنـواع مـن الأخطـاء تحـدث بـسبب المبرمج وهناك أنواع من الأخطاء تحصل بسبب المستخدم ، في العادة يجب أن يكون المبرمج متآلفـاً مع مصطلح تتبع الأخطاء وإصلاحها .

قد يكون من أهداف تتبع الأخطاء الحماية بقدر أهميه البرنامج الجـاري العمـل عليـه أو الموقـع فكلمـا كان الموقع مهماً كان وجوب حمايته أكبر .

قد يكون من الأسباب التي تسبب تدميراً للمواقع هو أن صاحب الموقع يغطي كل صـغيرة وكبيـرة عـن برنامجه الذي يركبه في موقعه وقد يكون برنامجه هذا غير محمي بـسبب كـاف أو يكـون مـسير بعـدة ملفات فيقوم شخص بحذف ملف من الملفات الأساسية بسبب عدم دقة في التراخيص المعطاة ممـا يؤدي إلى دمار الموقع نهائياً .

وقد يكون صاحب الموقع مهملاً في الحـد ذاتـه فـلا يحـتفظ بالمعلومـات الـسرية لموقعـه ممـا يـسبب مشاكل أكبر من التدمير مثل احتلال الموقع بشكل كامل .

رسائل الخطا في الـPHP لها طريقتها وتقنيتهـا الخاصـة التـي تـسير عليهـا فهـي ليـست مثـل الجافـا وليست مثل cgi
فالـ PHP لا تقوم بإرسال الخطأ إلى السيرفر بل تقوم بكتابة رسالة خطأ في مكان الخطأ .

قد يكون هناك أخطاء يصعب تتبعها أو معرفة مكانها في الأصل ، وقد يكون هـذا بـسبب أنـك تـستخدم الـPHP في صناعة موقع ديناميكي وتشرك معها الجافا سكربت وتضع علامـات التعليـق الخاصـة التـي تقوم بإخفاء الأخطاء في الجافا مما قد يجعلك تشعر بالحيره وتجن أين مكان الخطأ

```
<!- -
 رساله الخطا
- - >
```

أنواع الأخطاء

هناك أنواع من الأخطاء منها الإملائية (Syntax Error) ومنها المنطقية ومنها أخطاء تحدث في وقت التنفيذ

ومثال الأخطاء الإملائية :

```
<?
Eco "1";
// : من المفترض أن تكت التالي
Echo "1";
?>
```

هذا سيعطيك رسالة خطأ Parse error

ومن الأخطاء الإملائية نسيان الفاصلة المنقوطة (semi-colon) في نهاية الدالة :

```
<?
Echo “hello”
// : من المفترض أن تكت التالي
Echo “hello”;
?>
```

هنا سوف يعطيك الـPHP رسالة خطأ لكن العجيب أنه لن يعطيك إياها بـشكل صـحيح فرسـالة الخطـأ تشير إلى أن السطر الرابع يحتوي على الخطأ بينما الخطأ هو في السطر الثاني .

وهناك خطأ آخر يحصل بسبب نسيان الـbrace (وهي الاقواس) :

```
<? Php
for ($loop = 0 ; $loop < 5 ; $loop ++ )
{
Echo “”;
?>
```

اذا كنت قد نسيت إغلاق القوس فهذا من الأخطاء الشائعة ، والأخطاء الإملائية لايمكن حصرها ، إنهـا أشبه بقواعد اللغة ، لكن أكثر الأخطاء الإملائية الشائعة في برامج الـPHP

1 – نسيان الأقواس . مثال :

```
<?
for ($loop = 0 ; $loop < 5 ; $loop ++ )
{
for ($loop1 = 0 ; $loop1 < 10 ; $loop1 ++ )
{
for ($loop = 0 ; $loop < 5 ; $loop ++ )
{
code ….
}
}
```

في المثال السابق ينقصنا قوس إغلاق التكرار الأخير (})

2 - نسيان الفاصلة المنقوطة . مثال :

```
<?
Echo 10
<?
```

3 - خطأ إملائي في اسم function . مثال :

```
<?
Htmlspecialchar($I);
?>
```

سيعطيك رسالة خطأ :

```
Fatal error : call to Undefined function : htmlspecialchar().
```

وتصحيحها أن تكون :

```
<?
Htmlspecialchars($I);
?>
```

4 - نسيان إغلاق النص . مثال :

```
<?
Echo “arabbuilder;
?>
```

نسي الـ(")في نهاية الكلمة . وسيعطيك Parse error

الأخطاء المنطقية (Logical Errors)

إن الأخطاء المنطقية هي الأكثر صعوبة في التتبع فقد تجد برنامجك يعمل بشكل صحيح وبكل سلامة ولكنه عند نقطة ما لايتم تنفيذها كما تريد أنت ، لنضرب مثلاً على خطأ منطقي بسيط جداً ، لنفرض أنك قمت بعمل نموذج مكون من مربع نص وزر ، عند ضغطك لهذا الزر فأنت تريد أن يتم كتابة كلمة كبير إذا كان الرقم أكبر من 30 وكلمة صغير إذا كان الرقم أصغر من 30 لنقم بكتابة الكود للمثال الأول :

```
<?
echo "ادخل عمرك :" ;
echo '<br>
<form method = "post" action = "age.php">
<input type= "text" name = "age">
<br>
<input type= submit value = "هل أنا كبير أم صغير ؟" >
</form>' ;
?>
```

في ملف age.php اكتب الكود التالي :

```
<?
If ($age<30) echo “انت صغير”;
If ($age>30) echo “انت كبير”;
?>
```

سيعمل السيكريبت بشكل صحيح .. ولكن ربما تخطأ أنت في كتابة العلامات المنطقية (التي باللون الأحمر) فتأتي النتائج بشكا خاطئ .

ومن الأخطاء المطنقية الأخطاء التي تقع في وقت التشغيل (Run times error) والتي تكون قد تقوم بإيقاف برنامجك بشكل كامل

مثال :

```
<?
$t=0;
$r=1;
$f=$r/$t;
?>
```

وعندها سينتج لك الرسالة التالية
Warning : Division by zero in (path) on line (line number)

هناك نوع آخر من الأخطاء المنطقية (unexpected) وهو لا يقوم بإيقاف البرنامج نهائياً بل يقوم بإخراج رسالة الخطأ في مكان الخطأ أو قد يقوم بتنفيذ البرنامج وإخراج البيانات بشكل غير صحيح أو قد لا يقوم بإخراج بيانات
وهو المثال الاول الذي ذكرناه سابقا (تقييم العمر) .

اخطاء التكرارات
قد يكون لديك أيضا تكرار فيه خطأ ولا يقوم بالتوقف نهائياً مثل هذا التكرار :

```
$c=1;
$t=true;
while ($t=true)
{
$c++;
}
```

لم نقم بعمل شي يوقف التكرار مثل أن تضع شرط يختبر قيمة المتغير ($c) ثم يقوم بإيقافه عند تعديه رقم معين وعلى ذلك فإن التكرار سيستمر بشكل غير متوقف ولن يعمل البرنامج .

عدم ارجاع قيمه من function
مثال :

```
<?
Function ($d)
{
$d =$d+$d;
}
```

الخطا هنا اننا لم نستخدم الـreturn لكي ننهي الدالة أو قد تكون الدالة تحتوي على أكثر من قيمة وننسي أن نقوم بتحديد القيمة النهائية للدالة

الخلط في المعاملات الحسابية والمنطقية
مثال :

```
If ($y=10) echo 12 ;
```

والمفترض أن تكون :

```
If ($y= =10) echo 12 ;
```

أفكار جيدة لتفادي الأخطاء
التعليقات
إن من الأفكار الجيدة للتقليل من الأماكن التي تبحث فيها عن الخطأ هو وضع تعليقات لوصف وظيفة دالة معينة . مثال :

```
<?
// هذه الكود يقوم بطباعة كلمة أحمد
Echo "أحمد" ;
?>
```

الدوال

وأيضا من الأفكار الجيدة أن تقوم بتقسيم وظائف البرنامج على دوال بحيث أن لكل دالة وظيفتها المعينة :

```
<?
/*
        +- ----------------------------------------------------------+
        |        هذه الداله تقوم بقسمه العدد علي 2                  |
        +- ----------------------------------------------------------+
* /
function ($U)
{
$U=$U/2;
return $U ;
}
?>
```

Regular Expressions

هذه التقنية تساعدك على تفادي الأخطاء في صفحتك عند حدوثه مثل أن يقوم مستخدم ما بكتابة بريد الكتروني غير صحيح (مثال : a@y@.k.d) هذا البريد غير صحيح ولأجل أن تقوم بمنع حصول أي خطأ مثل ذلك وتقييد العبارات التي يدخلها المستخدم فإنك تقوم بإستخدام ال RE/ (Regular Expressions) إنك بالأصح تجعل قواعد للكلمات التي يدخلها المستخدم فمثلاً تجعل المستخدم لايدخل سوي أرقام أو حروف فقط أو شكل معين من الكلمات ، تقوم أولاً بإنشاء نمط للكلمه التي تريد المستخدم أن يقوم بادخالها .

النمط (pattren)

ماهو النمط ؟ مارأيك إذا كتب المستخدم جملة في مربع نص تحتوي على عدة كلمات وتريد أن تتأكد من وجود كلمة معينة وسط هذه الجملة ، على حسب ما اخذناه من معلومات على المصفوفات سابقاً نستطيع فعل ذلك كالتالى :

```
<?
$words=”one, two, three, four, five,”;
$ty =explode (“”,$ty);
foreach ($ty as $w) {
        if ($w = =  “six”) echo “found string ‘two’”;
}
?>
```

لقد كان المتغير $words يحتوي على جملة تتكون من عده كلمات وعندما أردنا فحصه قمنا باستخلاصه في مصفوفة ثم بعد ذلك قمنا بفحص المصفوفة باستخدام التكرار foreach ، ومع ذلك الذي فعلناه فإن هذا الاستخدام غير عملي بتاتاً وهنا تبرز قوه Regular Expressions لاحظ الآن كيف نستخرجه بواسطة الـRegular Expressions :

```
<?
$words=”one, two, three, four, five,”;
if (ereg(“one”,$words)) echo “ لقد وجدت العدد ‘ one’ “ ;
?>
```

في هذا المثال قمنا باستخدام الدالة (ereg) ووضعنا في خانتها الأولى النمط (pattern) الذي نريد أن نتأكد من وجوده (أوالكلمة المراد البحث عنها) ووضعنا في الخانة الثانية المتغير الذي سيتم البحث فيه عن الكلمة أو النمط .
تقوم الدالة ereg بإعطاء القيمة true إذا تم العثور على الكلمة .
في الواقع هناك استخدامات أكثر فعالية للأنماط .
يمكننا مثلاً تخزين الكلمة إذا تم وجودها في مصفوفة خاصة كالتالى :

```
<?
$words=”one, two, one, four, five,”;
if (ereg(“one”,$words,$rok))  ;
echo $rok[0];
echo $rok[1];
?>
```

نقوم بوضع اسم المصفوفة التي نريد تخزين البيانات في الخانة الثالثة .. لاحظ مع أنه يوجد كلمتين في الجملة توافق النمط إلا انه أعطانا كلمة واحدة فقط إذ أن وظيفته أن يتأكد من وجود النمط في الجملة فقط فإذا تأكد من وجودها مرة واحدة استكفى واعتبر الموضوع قد انتهي .

ماذا لو أردنا من التأكد من عدة كلمات ، عند ذلك فإننا نفعل التالى :

```
<?
$words=”one, two, one, four, five,”;
if (ereg(“one”,$words,$rok))  echo $rok[0];
if (ereg(“two”,$words,$rok))  echo $rok[0];
?>
```

واريد أن أنبهك أن الـereg يقوم بإنشاء المصفوفة من جديد عند كل استعمال له فخذ حذرك من هذه النقط
أيضا فإن الـereg حساس لحالة الأحرف لاحظ هذا المثال :

```
<?
$words="one, two, vcx, four, five,";
if (ereg("One",$words,$rok))  echo $rok[0];
?>
```

لن يقوم بإخراج أي شي فقط لأن حرف الـO مختلف .

أيضا يمكنك البحث عن كلمة يسبقها فراغ مثلاً كالتالى :

```
<?
$words="one, two, vcxone, four, five,";
if (ereg("one",$words,$rok))  echo $rok[0];
?>
```

مثال آخر :

```
<?
$words="oned, two, vcxone, four, five,";
if (ereg("one",$words,$rok))  echo $rok[0];
?>
```

لاحظ في هذين المثالين أنه مع أن كلمة one غير موجودة بمفردها إنما موجودة كجزء من vcxone و oned ورغم ذلك فإن الدالة لم تأخذ اعتبارا لذلك بينما لو كتبنا كالتالى :

```
<?
$words="oned, two, vcxone, four, five,";
if (ereg(" one",$words,$rok))  echo $rok[0];
?>
```

فإنه سيبحث عن الكلمة مفصولة عن أي حرف ولن يجد كلمة كذلك فلن يقوم بكتابة أي شي .

يمكننا أن نفحص قيمة موجودة في متغير كالتالى :

```
<?
$reu = "one";
$words="one, two, vcxone, four, five,";
if (ereg($reu,$words,$rok))  echo $rok[0];
?>
```

هل لاحظت أننا فحصنا قيمة المتغير $rue بواسطة ereg مع $word ولم يتطلب منا ذلك أي شي إضافي غير اسم المتغير المراد البحث عن قيمته في الجمله .

يمكننا بالـ Regular Expression استعمال بعض الأحرف بشكل خاص التي لها استعمالها الخاص بواسطة الـ Regular Expressions

الأحرف الخاصة في الـ Regular Expression هي كالتالى :

**. * ? + [ ] ( ) { } ^ $ | **

هذه الأحرف لها معناها الخاص في الـ Regular Expression

فقديماً مثلاً كنا نقول أنه لا يمكننا أن نستخدم علامتي تنصيص متداخلة من نفس النوع كالتالى :

```
<?
$r=”u\””;
?>
```

ولكي يتجاهل الـPHP هذا المعني فإننا نقوم بوضع (\) قبل علامة التنصيص .
ايضا مع الـ ereg فإن للـ(.) قداستها ولكي يتم تجاهلها فإننا نستخدم الـ(\)

تقوم الـ(.) بأخذ مكان حرف أو فراغ فمثلاً لاحظ المثال التالى :

```
<?
$P="I love yamen";
if (ereg ("love....",$P,$R)) echo $R[0];
?>
```

هل لاحظت الناتج ؟؟

ولكي يتم تجاهل قداسة الـ(.) في الـRegular Expressions نقوم بوضع (\) قبلها . مثال :

```
<?
$P="I love yamen";
if (ereg ("love\.\.\.\.",$P,$R)) echo $R[0];
?>
```

في هذا المثال لن يتم طباعة أي شي لأنه لايوجد أي كلمة تطابق (love....) لأن الـ(.) فقدت قداستها وبدأ التدقيق في الكلمة حرفاً حرفاً .

صناعة فئة حروف [xyz]
أقصد بذلك أنني احدد نطاق معين من الكلمة من الممكن أن يكون في هذا النطاق أي حروف من الفئة التي أقوم بتحديدها أو الحروف التي أقوم بتحديدها .
مثال :

```
<?
$y=”how are you ? “ ;
if (ereg(“h[oe]” , $y)) echo “true”;
?>
```

هنا قام الـregular expression بالبحث عن أي كلمة تبدأ بالحرف h ومن ثم يتبعها أحد الحرفين o أو e مثال هذه الكلمات :

Hey – He – Hew - Homer

ولكنها لاتطابق :

Hty – Hnt - Hlay

أتمني أن تكون فهمت ما أرمي إليه

يمكننا أيضا أن نقوم بإخبار الـregular expression بأن لا يقوم باختيار كلمات تحتوي على حروف معينة وذلك فقط بإضافة ^

```
<?
$y=”how are you ? “ ;
if (ereg(“h[^oe]” , $y)) echo “true”;
?>
```

نقوم هنا بإخبار الـre بأن يقوم بفحص الجملة فإذا وجد أي كلمة تبدأ بـh ولاتحتوي على o أو e فإنه يقوم بإعطاء true وإذا لم يجد يقوم باعطاء false

وهذا الكلام يطابق الكلمات التالية :

Hay - Hana - Hkg

ولايوافق هذه الكلمات :

Home – Hore - Here

يمكننا استعمال اختصارات لبعض الأمور فمثلاً إذا كنا نريد كلمة لاتحتوي على أي رقم كنا سنكتب كالتالى

[^123456789]

يمكننا أن نستعمل اختصار لهذا الموضوع كالتالى :

[^0-9]

وحتي إذا أردنا أن يتأكد من وجود رقم من واحد الى تسعه فقط علينا مسح الـ^

[0-9]

وأيضا الحروف الصغيرة من a الى z

[a-z]

وإذا نريد التأكد من عدم وجودها

[^a-z]

نفس القصة مع الحروف الكبيرة .

هناك اختصارات اخري لهذا الموضوع كالتالى :

الاختصار	المطابق له	معناه ووظيفته
\d	[0-9]	أي رقم من 0 الى 9
\D	[^0-9]	ممنوع الأرقام من 0 الى 9
\w	[0-9A-Za-z_]	أي رقم من 1-9 أو حروف A-Z او احرف صغيره او_
\W	[^0-9A-Za-z_]	عكس السابق
\s	[\t\n\r]	يقبل مسافة أو سطر جديد أو علامة جدولة (tab)
\S	[^\t\n\r]	عكس السابق

تحديد مكان الكلمة

يمكننا أن نقوم بتحديد مكان الكلمة ، اقصد بذلك أنه يمكنك تحديد مكان الكلمة إذا كانت في بداية أو نهاية النص ونستخدم لهذا الأمر العلامتين (^)لتحديد المكان لبداية الجملة و ($) لنهاية الجمل .
مثال :

```
<?
$y="how are you ? " ;
if (ereg("^h",$y)) echo "true";
?>
```

هنا سيقوم الـphp بالبحث عن في الحملة فإذا وجد الجملة تبدا بحرف h كانت قيمة الـereg تساوي true وإذا لم يجد كانت قيمة الـereg تساوي false

```
<?
$y="how gone?" ;
if (ereg("^g",$y)) echo "true";
?>
```

في هذا المثال ستكون قيمة الـereg خطأ لأن العبارة لاتبدأ بحرف g
يمكننا فعل العكس بواسطة العلامه ($) التي عملها عكس (^) فهي تفحص إذا كان الحرف المراد فحصه موجود في نهاية الجملة
مثال :

```
<?
$y="how g" ;
if (ereg("g$",$y)) echo "true";
?>
```

يمكننا أيضا اختيار إذا ما كان واحد من نمطين صحيحاً بواسط العلامة (|)

```
<?
$y="how g" ;
if (ereg("^y | g$",$y)) echo "true";
?>
```

في هذا المثال سيقوم الـPHP بفحص الجملة فإذا وافقت أحد النمطين كانت قيمة الـereg عند ذلك true .

يمكننا أيضا تحديد إذا ما كان حرف أو جملة متكررة بعدد من المرات أو مره واحدة باستخدام أحد هذه الثلاث رموز (* ، + ، ?)
تقوم علامه الضرب بالتحقق من أن الحرف الذي يسبقها مكرر مرة أو أكثر أو غير موجود بتاتاً
مثال :

```
Bea*t
```

وتوافق :

Bet
Beat
Beaat

تقوم علامة الجمع (+) بالتأكد من وجود عنصر مرة أو أكثر :

```
Bea+t
```

وتوافق :

Beat
Beaat
Beaaaaat

أما علامة الاستفهام فتقوم بالتأكد من وجود عنصر مرة واحده أو عدم وجوده بتاتاً :

```
Bea?t
```

وتوافق :

Bet
Beat

وتأكد دائماً أن هذه الثلاث علامات مسبوقه بحرف .

وعند إرادتك مثلاً التأكد من سبق حرفين أو ثلاث بشكل تحديدي يمكنك استخدام القوسين
مثال :

```
(wo)?man
```

ويوافق :

man
woman

يمكننا التأكد من تكرر حرف بشكل معين من المرات أو أكبر من عدد معين من المرات أو أصغر من عدد معين من المرات باستخدام القوسين {x,y}
فمثلاً لو أردنا أن نتأكد من أن حرف (d) مكرر مرتين إلى أربع مرات :

```
d{2,4}
```

أما إذا أردنا أن نتأكد من أنه مكرر أكثر من مرتين إلى عدد غير محدود من المرات :

```
d{2,}
```

أما إذا أردناه أن يتكرر 4 مرات على الأكثر :

```
d{,4}
```

أو إذا اردناه أن يتكرر بعدد محدود من المرات :

```
d{8}
```

أخيراً نريد أن نلفت النظر إلى الاختصار (\b) الذي معناه أي شي ولكن ليس حرفاً (الحروف التي بين \w وبين \W تقريباً)

ملخص ما أخذناه من القواعد تجدونه في الجدول التالى :

القاعده	المعني
[abc]	أي حرف كان a او b او c
[^abc]	أي حرف غير a و b و c
[a-z]	كل الحروف من a الى z
\d\D	\d للارقام و \D لغير الارقام
\w\W	\w للحروف جميعها و\W لغير الحروف
\s\S	\s للفراغ (space) و\S لغير الفراغ (no space)
\b	الحروف التي بين \w و\W
.	أي حرف
(abc)	تقوم باعتبار abc كمجموعه ..
؟	حرف او مجموعة حروف مكرره مره اوغير مكرره نهائيا
+	حرف او مجموعة حروف تتكرر مره او اكثر
*	حرف او مجموعة حروف تتكرر مره او اكثر او قد لاتتكرر نهائيا
{x,y}	تكرير بعدد معين من المرات ..
{,y}	تكرير بحد اقصي من المرات ..
{x,}	تكرير بحد ادني من المرات ...
{x}	تكرير بعدد معين من المرات
^	في بدايه النص
$	في نهايه النص

تعبير للتاكد من ايميل

```
^[_a-zA-Z0-9-]+(\.[_A-Za-z0-9-]+)*@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$
```

شرح التعبير

الرمز	الشرح
^	يجب ان يبدا النص
[_A-Za-z0-9-]	أي حرف من a-z كبيرا كان او صغيرا او _او ارقام
+	وقد يكون هذا الحرف متكررا اكثر من مره
(\.[_A-Za-z0-9-]+)	بالاضافه الى انه قد يتبع النقطه وحروف وارقام
*	وقد لايتبعه اوقد يتبعه ويتكرر اكثر من مره
@	وبعد ذلك يكون لديه حرف ال@
[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$	وايضا نفس القواعد في النهايه

مثال :

```
<?
Function mailcheck($mail,$t)
{
$T="^[_a-zA-Z0-9-]+(\.[_A-Za-z0-9-]+)*@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$";
If (EREG($T,$mail))
{
$r="the mail is true";
echo $r;
}
else
{
$r="the mail is not true";
echo $r;
}
return ;
}
mailcheck("alfareees@hotmail.com",$t);
?>
```

eregi()

الفرق بين هذه الدالة والدالة ereg أنه غير حساسة لحالة الأحرف كبيرة أو صغيرة أي أنه يمكننا كتابة المثال السابق كالتالى :

```
<?
Function mailcheck($mail,$t)
{
$T="^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*$";
If (EREG($T,$mail))
{
$r="the mail is true";
echo $r;
}
else
{
$r="the mail is not true";
echo $r;
}
return ;
}
mailcheck("alfareees@hotmail.com",$t);
?>
```

ereg_replace()

ماذا لو أردت تحرير عبارة ما من أحرف معينة وقد تكون متكررة في جملة أوغير ذلك لنفرض أن لدينا العبارة التالية :

```
Mohmed love his game .....
```

ونريد أن نتخلص من النقاط التي في نهاية العبارة
أو لدينا مثلاً هذا المسار :

```
C:\windows\desktop
```

ونريد أن نستبدل العلامة (\) بـ (/)

كل ذلك ممكن بواسطة الدالة ereg_replace وقواعد الـ rgular expression التي أخذناها سابقاً
البنية التي نستخدمها للدالة كالتالى :

```
Ereg_replace(reg,string,var);
```

نضع في مكان reg القاعدة للـ regular expression ونضع مكان الـstring الحرف الجديد ونضع بدلاً من الـvar المتغير الذي نريد استخلاص الحروف منه .
مثال :

```
<?
$path =" C:\windows\desktop";
$tell= "Mohmed love his game .....";
$newpath= Ereg_replace("[\.]","/",$path);
$newtell= Ereg_replace("\.","",$tell);
echo $newpath;
echo "<br><br>";
echo $newtell;
?>
```

أساليب أخرى لتتبع الأخطاء

استخدام عباره echo

هو من أقدم الاساليب وكان يستخدم مثلاً في فحص بعض متغيرات نموذج فمثلاً أنت لديك نموذج يقوم بإرسال معلومات إلى النموذج وقد تستخدم في اختبار الأخطاء المنطقية التي يستصعب متابعتها في الكود
مثال :

```
<?
Echo “this is : $name”;
Echo “<br>”;
Echo “this is : $Email”;
//كود يقوم بمعالجة معلومات المتغيرين
//طباعة المتغيرين بعد اداء عملية المعالجة ورؤية النتائج
Echo “this is after : $name”;
Echo “<br>”;
Echo “this is after: $Email”;
?>
```

فحص كود الـhtml

قد تستخدم كود جافا سكربت ويتم إخفاء الأخطاء وسط علامات التعليقات فعليك حينئذ فحص كود الـhtml لرؤية إن كان هناك بعض الأخطاء المخفية أم لا .

تجاهل الأخطاء

لنفترض أنك تعلم أن الدالة التي صنعتها بها أخطاء ولكنك تريد تجاهل هذه الأخطاء فكل ماعليك أن تقوم بوضع @ أمام الدالة لكي يتم تجاهل الخطأ عند حدوثه .
مثلاً نحن نعلم أن القسمة على الصفر من الأشياء الغير مقبولة في الـPHP وأنت صنعت دالة تقوم بالقسمة على صفر ولن يتم تنفيذها لأنها بالأصل خطأ ولكنك تريد أن يقوم PHP بتجاهلها فكل ما عليك أن تفعله هو وضع @ أمام الدالة .
مثال :

```
<?
function amail ($y)
{
$y=$y/0;
return $y;
}
$s= @amail(44);
echo $s;
?>
```

التعامل مع العميل

كما رأينا في الدروس السابقة ، فإن الـPHP يوفر رقم عظيم من المميزات عن الـhtml لبناء مواقع الويب ، من الأشياء الأساسيه التي لم نتكلم عنها حتي الآن هي الموثوقية (أو الاستقرار) وهو بالمعني الصحيح والصريح :

القابيلة على الاحتفاظ بالمعلومات بين صفحتين منفردتين أو مختلفيتن في المستعرض ...

بدون أي إضافات ، HTTP لايوفر أي ميكانيكيه للحفاظ على البيانات وجعلها مستقرة لمعالجة تتم بين صفحتين ، كل طلب لصفحة في الانترنت (request) ليس له أي علاقة بأي طلب آخر ... مثلاً عندما تتطلب موقع المطور العربي ومن ثم منتدي المطور العربي فان كل الطلبين ليس لهما علاقه ببعضهما ...

بمصطلح آخر يمكننا أن نقول أن الـHTTP فاقدة لحالتها (stateless) أي أنها لاتعرف أي أن أمر طلب الصفحة ينتهي عند انتهاء الطلب ، فهي عندما تقوم بنقل بيانات صفحة من السيرفر الى المستخدم فهي تعرف من هو المستخدم الذي يطلب البيانات وعلى أي نافذه سيتم نقل البيانات وعند انتهاء ذلك فان كل هذا الموضوع ينتهي وإذا عاد المستخدم فطلب صفحة أخري فإنها لاتعرف إن كان هو نفس المستخدم أو لا !

إن القدرة على الحفاظ على وجود البيانات ليست وسيلة أو ميزة او قوة مقتصرة على الـPHP فقط .

فلقد رأيت كيف استطعنا ارسال معلومات من صفحة إلى صفحة بدون خسران أي معلومات وذلك عن طريق الـhtml وبالرغم من ذلك فإن المستخدم عندما يقوم بإغلاق الصفحة عند استقبالها للبيانات فان ذلك يعني فقدانها للأبد ، عن طريق استخدام الـPHP يمكننا اخبار السيرفر بأن يقوم بارجاع البيانات بطريقة تمكننا من الحفاظ عليها ، مثلما سنري في هذا الدرس ، هناك ثلاث طرق لعمل ذلك ....

التميز الحقيقي في قوة الفهم للـPHP ، يتطلب منا مفهومية جيدة في كيفيه استعمال الـPHP في التفاعل مع المستخدم والمتصفح الذي يستخدمه لكي نتغلب على نقاط الضعف التي في الـhttp .

هذا هو موضوعنا لهذا اليوم والذي سنتكلم فيه عن :

1 - الـHTTP والـ html ومحدودية قدراتهم ، وكيف يستطيع الـPHP التغلب علي القصور فيهم .
2 - الاحتفاظ بالمعلومات التي نريد أن نستخدمها بين طلب لصفحتين مختلفيتن .
3 - مكنكة الحفاظ على البيانات .
4 - الكعكات (cookies) وكيفية استخدامها .
5 - PHP4 والـ native session – المكنكة الداخلية للحفاظ على وجودية البيانات .

هذا الدرس مفيد بشكل ظاهري لمن هو جديد على انشاء مواقع متفاعلة متوسطة – كبيرة الحجم بواسطة الـPHP ..
إنه يحتوي على الكثير من بعض الأمثله التي تفيدك .

الهدف من هذا الدرس هو أن تتعرف على كيفية الحفاظ على معلومات المستخدم عبر متغير أو أكثر بين أكثر من صفحة ، مثل أن تجعل اسم المستخدم ظاهر في كل صفحة يقوم بالولوج إليها ... مما يؤكد استمرارية وجود البيانات .

لنفرض أن لدينا موقعاً على الانترنت هذا الموقع يهتم ببيع وتسويق مواد غذائية أو أن هذا الموقع يقدم مسابقات ثقافية ، في العادة عندما يقوم المستخدم بطلب شراء سلعة معينة أو عندما يختار الدخول في مسابقة من المسابقات الثقافية فإنه يقوم بدخول أكثر من صفحة بالتتابع

يختار السلعة في الصفحة الأولى وبعد ذلك يقوم برؤية معلومات السلعة في الصفحة الثانية والصفحة الثالثة يقوم فيها بتعبئة معلوماته للشراء أو غير ذلك إلى أن ينتهي من كافة المعلومات وبعد ذلك تنتج له في النهاية صفحة فيها معلوماته والسلعة التي قام باختيارها وفاتورة شراء !!

أو يقوم باختيار نوع المسابقة الثقافية في الصفحة الأولى وبعد ذلك يقوم بالحصول على عدة أسئله مقسمة على عدة صفحات إلي أن ينتهي من المسابقة فتخرج له في النهاية مجموع الدرجات للأسئلة ومعلوماته وهل هو فائز أم خاسر!!

في الواقع هذا مايسمونه بالمحافظة على الجلسة (maintain session) وأقصد بذلك دخول المستخدم إلى صفحة وانتقاله من صفحة إلى صفحة مع المحافظة على معلوماته وغير ذلك من البيانات ، لكي نستطيع متابعته أولاً بأول .

في بروتوكول الـhtml والـhttp لانستطيع معرفة إذا ما كان الشخص عندما يطلب صفحة ما هو نفسه عندما يذهب إلى الصفحة الثانية إذ أن المستخدم عندما يطلب صفحة ما (request) من السيرفر فإن السيرفر يقوم بمعرفة من أي مكان بالعالم يتكلم هذا الشخص ويقوم بارسال استجابه إليه باعطاءه الصفحة التي كان يطلبها (response) ولكن بعد ذلك فإن السيرفر لا يعرف إذا كان هذا الشخص هو نفسه الذي يقوم بطلب الصفحه الثانية أو الثالثة في السيرفر .

هنا تأتي ميزة الـPHP وغيره من لغات برمجة الانترنت لصناعة ميكانيكية إبقاء تفاعل مستمر بين المستخدم والسيرفر عن طريق الـsession و الـcookie ، ولكي لا نعقد الموضوع دعونا نتكلم عن ذلك عملياً فذلك أفضل لفهم الموضوع من الثرثرة التي لا فائدة منها .

استخدام الحقول المخفيه

سنقوم الآن بإنشاء ثلاث صفحات ، الصفحة الأولى تطلب من المستخدم ادخال اسمه ، والصفحة الثانية تقوم بالترحيب به واعطاءه ثلاثة أسئلة ، والصفحة الثالثة تقوم باعطاءه النتيجة .

افتح محرر نصوص لديك واكتب الكود التالي :

```
<p dir="rtl" align="center">ادخل اسمك الكريم </p>
<form method="POST" action="quiz2.php">
<hr>
<input type="text" name="name" size="20"><br>
<input type="submit" value="إرسال" ></p>
</form>
```

احفظها باسم quiz.php

قم بفتح محرر النصوص واكتب الكود التالي :

```
<html dir ="rtl">
<?
If  (isset($name)) {
Echo "مرحبا بك يا" . $name ;
 Echo '
<br>
<form method="POST" action="quiz3.php" dir="rtl">
<input type="hidden" name = "thename" value = "'.$name.'">"
  من هو أول الخلفاء الراشدين ؟ </p>
  <p dir="rtl"><input type="radio" value="أبوبكر الصديق"
name="khlifa">أبوبكر
  الصديق .</p>
  <p dir="rtl"><input type="radio" value="عمر بن الخطاب" checked
name="khlifa">عمر
  بن الخطاب</p>
  <p dir="rtl">من هو الفاروق ؟</p>
  <p dir="rtl"><input type="radio" name="faroq" value=" عمر بن
الخطاب"><عمر بن
  الخطاب</p>
  <p dir="rtl"><input type="radio" name="faroq" value="سالم"
checked>سالم بن
  عامر</p>
<input type ="submit" value = "إرسال" dir="rtl">
</form>' ;
}
else
{
echo "غير مصرح لك بدخول هذه الصفحة" ;
}
?>
```

احفظها باسم quiz2.php

قم بفتح المفكرة واكتب الكود التالي :

```
<?
If  ((isset($thename)) && (isset($khlifa)) && (isset($faroq)))
{
echo 'لقد انتهت المسابقه يا '. $thename ;
$range=0;
$co = 0;
        if ($khlifa =="أبوبكر الصديق") {
        $range=$range+10;
        $co = $co +1;
        }
        if ($faroq =="عمر بن الخطاب")
             {
             $range=$range+10;
             $co=$co+1;
             }
          if ( $range < 10)
```

```
            {
            echo "ليس هناك أي إجابة صحيحة";
            }
            else
            {
            echo "<br>"."عدد الاسئلة التي أجبت عليها =" . $co ;
            echo "<br>" . "الدرجة التي حصلت عليها " . $range ;
            }
}
?>
```

قم بوضع الملفات الثلاثه السابقه في مجلد السيرفر ثم قم بتشغيلها

الشرح

قمت في هذا المثال بمحاولة صنع مكنكة تواصل للبيانات ، بمعني أنني أحاول أن أقوم بالاختفاظ بالبيانات عبر الثلاث صفحات بشكل متواصل ، لاحظ أنني كنت اختبر في quiz2 و quiz3 باختبار المتغيرات قبل طباعة أي شي فقد يقوم المستخدم مثلاً بالاحتفاظ بالصفحة التي وصل إليها في المفضلة ثم يقوم باكمال المسابقة في وقت آخر ولكني لا اريد ذلك بل أريد ان أجعل وقتها محدوداً (طبعاً هذا الكلام سيحصل إذا كانت المسابقة طويلة) لذلك فإنني في كل عند الانتقال من صفحة إلى صفحة اقوم باختبار إن كانت جميع هذه القيم موجودة ولاحظ أنني كنت اجتفظ دوماً بقيم المتغيرات في متغيرات جديدة في حقول مخفية وكلما كان عدد المعلومات أكبر في كل مره كان عدد الحقول المخفية أكثر ، إن لهذه الطريقة أيضا مشاكلها فقد يفتح المستخدم كود الـhtml ويقوم بتفحص كيفية ملاحقته عبر المسابقة وقد يصنع هو الكود في وقت لاحق لكي يستطيع اكمال المسابقة بهذه الخدعة الماكرة ... لذلك يفضل أن لا تقوم بذلك وتقوم بجعل المسألة السابقة اكثر تعقيداً باستخدام الـregular expression بمحاولة تلغيم البيانات بواسطته ومن ثم فك هذا التلغيم في الصفحات التي تصل اليها البيانات .

ارسال بيانات بواسطة query strings

نستطيع ارسال بيانات بسيطة بواسطة الاستعلامات التي نقوم بإضافتها الى اسم الصفحة في الأعلى متبوعة بـ(؟) علامة استفهام ثم اسم متغير وقيمته وإذا كان هناك أكثر من متغير يتم الربط بينهم بعلامة & وراجع درس النماذج لمزيد من المعلومات .

قم بعمل صفحة وسمها ask.php وقم بكتابة الكود التالى فيها :

```
<?
If (isset($ask)) {
 If ($ask == login) {
  Echo "تم تسجيل الدخول إلى الصفحة";
 }
}
if (!isset($ask)) {
echo "لم يتم تسجيل الدخول إلى الصفحة" . "<br>";
Echo "<A HREF=$PHP_SELF?ask=login>اضغط هنا ليتم تسجيل دخولك
</a><br>";
}
?>
```

قم بتجربة هذا المثال على موقع يدعم PHP على نظام تشغيل لينوكس إذا لم يعمل بشكل جيد على الوندوز

لاحظ أننا في أول الولوج الى الصفحة لم نستخدم أي استعلامات وعند الضغط على الرابط قام الرابط بارسال قيمه المتغير الذي يقوم الPHP باختبارها فاذا وجد انه قد تم ارسالها (بواسطه الرابط الذي تم الضغط عليه) قام بطباعة (تم تسجيل الدخول) واذا لم يجدها قام بطباعة (لم يتم تسجيل الدخول) بالإضافة إلى طباعة الرابط الذي يحتوي على المتغير في طياته

الكوكيز أو الكعكعات (cookies)

إذاً ماهي الكوكيز ، الكوكيز هي عبارة عن بعض المعلومات أو القطع الصغيرة من البيانات يتم الاحتفاظ بها في جهاز العميل لكي يتم الاحتفاظ بها عند الزيارات المختلفة للمستخدم (العميل) ، أنت لا تقوم بالاحتفاظ فيها بقيم ضخمة لكنك تستفيد منها في أشياء أخري مثل :

1 - جعل لكل مستخدم الألوان الخاصة التي يري فيها صفحتك (أي أن تجعل للمستخدم مثلاً إعدادات الألوان الخاصة لرؤية موقعك) .
2 - جعل مفتاح للمستخدم لكي يستطيع به التحكم في بياناته الخاصة عند زياراته لموقعك في مرات اخري.

الكوكيز مفيد للاستخدام في الأشياء البسيطة والغير خطيرة ، لكنه الآن يستخدم بشكل سئ ، مثل استخدامه مثلاً في معرفة معلومات عن المستخدم بدون علم منه ، أو تخزين كميات كبيرة من البيانات فيه والتي من الأجدر أن يتم حفظها في ملف على السيرفر .
ويكون استخدامه مفيدا عندما تضمن أن جميع زوار موقعك تسمح متصفحاتهم بالكوكيز (مثل طلبة المدارس أو شبكات انترانت) .
عندما يكون فقط لأشياء بسيطة لاضرر منها عند عدم السماح بالكوكيز بجهاز العميل .

بدايتك مع الكوكيز

قبل أن نبدأ علينا معرفة بعض الأساسيات عن الكيوكيز
الكوكيز عبارة عن قطعة صغيرة من البيانات التي تستخدم لتخزين اسم متغير وقيمته مع معلومات حول الموقع التي أتت منه وتاريخ انتهاءها .

الكوكيز عباره عن تقنية للتخزين من جهة العميل (client-side storage) تتخزن في ملفات في جهاز العميل

يتم العبور إلى هذه الكوكيز ومسحها من المكان التي ارسلت منه .

عندما يطلب المستعرض صفحة من السيرفر وهذه الصفحة تقوم بتتخزين كوكيز فإن السيرفر يقوم باخبار المستعرض بأنه سيقوم بوضع كوكيز للاستعمال لاحقا .

عندما يتم طلب الصفحة في مرة آخري يقوم المستعرض بارسال البيانات التي تم إنشاؤها سابقاً عند طلب الصفحه .
يتم انتهاء مده الكوكيز بإنتهاء وقت صلاحيتها االمحدد من قبل السيرفر ويتم مسحها فورياً عند اغلاق الصفحة إذا كان وقت صلاحيتها صفراً من الثواني .

بإختصار عندما يعطي السيرفر الكوكيز للمستعرض فإنه يقول لك هذا شي اتذكرك به في وقت لاحق (قد يكون هذا الوقت من ضغط رابط آخر في الصفحة التى زرتها حتي بعد أسبوع أو أكثر) .

يقوم السيرفر بإرسال الكوكيز عبر الـHTTP Headers الذي يتم إرساله قبل أي مخرج من مخرجات الـhtml
والمستعرض أيضا يقوم بإرسال الكوكيز عبر الـHTTP Header بالإضافه إلى أن المستعرض يتعرف على من سيقوم بإرسال الكوكيز فلو كانت الكوكيز مثلاً مرسلة من قبل الموقع www.php.net فإنه لن يقوم بإرسالها إلى موقع www.phpbuilder.com .

باستطاعتك عند إنشاء الكوكيز تحديد مسار يتم ارسال الكوكيز لكي يتم اقتصار عملية العبور إلى الكوكيز إلى أماكن معينة .
قبل أن نقوم بوضع كود بسيط سنقوم الآن بتعريف كيفية تخزين الكوكيز وكيفية قراءتها :
كون الـPHP لغة حديثه لعمل سكربتات ويب فإنها تأتي بدعم كامل للكوكيز بواسطة الدالة setcookie() باستثناء أنك عند استعمالها يجب استعمالها قبل طباعة أي مخرجات html .

تاخذ الدالة setcookie() ثلاث معاملات ، الثلاثة الأولى هي الأهم والأمثل استخداماً وهي بالترتيب :

- ❖ قيمة حرفية يتم تخزينها كاسم للمتغير
- ❖ قيمة حرفية يتم تخزينها كقيمة لذلك المتغير
- ❖ Unix timestamp الذي يقوم بالإشارة إلى تاريخ إنتهاء الكوكيز

Unix timestamp عبارة عن رقم صحيح لا يحتوي على فواصل عشرية يقوم بحساب الثواني من منتصف ليلة 01/01/1970 . وإذا كنا نريد مثلاً أن نقوم بمسح الكوكيز بعد ساعة من تخزينه فإننا نقوم باستعمال الدالة time() التي تقوم بحساب الـtimestamp ثم نضيف عليه الوقت الذي نريده وفي حالتنا الساعه تساوي 3600 ثانية وعلى ذلك سنقوم بإضافة ناتج الدالة time على 3600 لكي يتم مسح الكوكيز بعد ساعة واحدة !

الثلاث العوامل الأخري التي يتم ستخدامها أيضا في الكوكيز ولكنها نادرة الاستخدام ولن نناقشها في موضوعنا هذا هي :

- ✓ المسار الذي يتم إرسال الكوكيز إليه فلو تم فتح نفس الصفحة من نفس الموقع ولكن من مسار اخر (مثلاً المسار كان pag\url\one وتم تغييره إلى page\url\two فان المستعرض لن يقوم بارسال البيانات إلي الصفحة لأنه تم تحديد المسار الذي سيتم ارسال الكوكيز اليه)
- ✓ الدومين الذي سيتم إرسال البيانات إليه وهو مفيد في حالة ما إذا كان هناك أكثر من دومين تريد ارسال الكوكيز اليه
- ✓ متغير من نوع integer يتم الإشارة إليه بـ secure يتم في حالة استخدام عمليات تشفير بالـ SSL

العبور الى الكوكيز بسيط جداً فالمتغير الذي يتم ارساله يتم تخزينه ضمن المتغيرات العامة (global) وعندئذ فإنه لو كان لدينا كوكيز اسمه ahmed فإن قيمته توضع مباشرة في متغير اسمه $ahmed !!

يمكننا مسح الكوكيز بأكثر من طريقة ، بالطبع فإن المستخدم يستطيع مسح الكوكيز وتغيير محتوياتها بنفسه ولكن في حالة ما إذا أردنا أن نجعل السيرفر يقوم بمسحها فإننا نستخدم إحدي هاتين الطريقتين

إما أن نقوم بإخبار السيرفر بوقت قديم :

```
<?
Set cookie (“ahmed” , “0”, time()-999);
?>
```

وإما القيام بمسح الكوكيز بكتابة اسمه فقط :

```
<?
Setcookie (“ahmed”);
?>
```

مثال لتخزين وقراءة كوكيز

قم بفتح المفكرة واكتب الكود التالي :

```
<?
If  ($thename)  setcookie ("rname", $thename, time()+3600);
Echo '<form method="post">
<input type ="text" name="thename">
<input type="submit" value="تسجيل">
</form>';
echo "قيمه المتغير الذي لديك". " ". $thename ."<br><br>";
echo "قيمة الكوكيز =" . $rname ;
?>
```

الشرح

عند تشغيل الصفحه لاول مره

عند تشغيلك للصفحة سيتم اختبار ما إذا كان هناك متغير بالاسم $thename فإذا تم الحصول عليه فسيتم وضع قيمته في كوكيز باسم (rname) (وطبعا لن يتم الحصول عليه في أول مرة لأننا لم نقم بارسال أي بينات بعد) وبعد ذلك طباعة نموذج من مربع نص واحد وزر لإرسال المعلومات .
ويتم طباعة قيمة المتغير إذا كان هناك أي متغير تم ارساله باسم $thename ويتم فحص قيمة الكوكيز $rname وطباعتها وبالطبع لا يوجد حتي الآن أي كوكيز .

المرحلة الثانية

الآن قم بكتابة أي شي في مربع النص (اكتب اسمك مثلاً) ثم قم بضغط زر الارسال سيتم ارسال البيانات الى نفس الصفحة ولكن هذه المرة سيتم تسجيل قيمة المتغير الذي يحمل البيانات في الكوكيز (rname) وبعد ذلك سيتم طباعة النموذج بشكل عادي وسيتم طباعة قيمة المتغير $thename ولكن لن يتم طباعة قيمة المتغير $rname لأننا فقط قمنا بتسجيله ولم يتم ارساله عند طلب الصفحة (لأننا نعرف أنه يتم ارسال الكوكيز عند طلب الصفحة وهذه المرة عندما طلبنا الصفحة لم يكن الكوكيز موجوداً بالأصل فلم يرسله السيرفر وقمنا نحن بتسجيله استعداداً للمرحلة القادمة) .

المرحلة الثالثة

في هذه المرة سيكون الكوكيز موجوداً فسيتم ارساله على هيئة متغير ويتم ارساله ومن ثم طباعة النموذج وقيمه المتغير $thename وقيمة الكوكيز الذي يوجد بجهازك !

بدايتك الى الـsession

الـ session هي عبارة عن تقنية للترابط مع المستخدم وهي موجودة ضمن الـPHP4 ولم تكن موجودة ضمن الإصدارات التي قبله بل كان يجب أن تقوم بتركيب مكتبة لكي تستطيع استخدام هذه التقنية ، يعتمد فهمنا للـsession على فهمنا للكوكيز وكيفية استعمالها ولقد تكلمنا عن الكوكيز بشكل جيد في الدرس السابق ، يستخدم الـsession لعمل ميكانيكية تواصل بين المستخدم والسيرفر ، فلقد قلنا أن الـhttp لا يوفر لنا ميكانيكية لعمل تواصل ، فإذا طلب المستخدم صفحة من السيرفر فإن السيرفر يقوم بإعطائه ما أراد وينتهي عند ذلك فلا يعرف إن كان هو نفس المستخدم أو ليس هو ... لأجل ذلك تم انشاء تقنية الـsession لأجل عمل تقنية تواصل بين المستخدم والموقع ، فبإستطاعتك مثلاً أن تقوم بتحديد عدد زيارات مستخدم معين لصفحتك ليوم واحد أو لأسبوع أو لمدة معينة من الوقت .... أو يمكنك عمل متجر إلكتروني بسيط يستطيع المستخدم شراء عدة أشياء دفعة واحدة من الموقع ويكون على تواصل بينه وبين الموقع عندما يقوم بإضافة مشترى إلي سلة التسوق أو حذف مشتريات.

قبل أن أتكلم عن كيفية استخدام الـSession وإعطاء بعض الأمثلة البسيطة ، سأقوم بالتكلم عن كيفية إعداد الـsession مع الـPHP .

اعدادات الـsession في الـPHP

لكي تستطيع التعامل مع الـsession بشكل جيد يجب عليك أن تتعرف على بعض الإعدادات التي في ملف الـphp.ini
عندما تفتح الملف ستجد قسماً خاصاً فيه بالـsession هناك حوالى 19 إعداد ولكن لن نتطرق إليها كلها بل سنتكلم عن الأساسية والمهمة منها فقط كبداية لنا للتعرف على الـsession وكيفية عمله .

اعداد طريقة التخزين

```
session.save_handler (files | mm | user)
```

ستجد هذه العبارة مكتوبة في الملف كالتالى بشكل افتراضي :

```
session.save_handler = files
```

وهذا الإعداد يقوم بتحديد طريقة التخزين للـsession وهناك ثلاث حالات للتخزين :

1 - التخزين في ملفات عادية على السيرفر :

```
session.save_handler = files
```

2 - التخزين على ذاكرة السيرفر :

```
session.save_handler = mm
```

3 - التخزين بطريقة أخري معرفة ومعينة من قبل المستخدم مثل التخزين في قواعد البيانات وهذا ما سوف نقوم بالتفصيل عنه بعد الكلام عن قواعد البيانات :

```
session.save_handler = user
```

يجب أن تأخذ في اعتبارك عدد الملفات التي سيقوم الـsession بتخزينها عند استخدامك للأعداد الأول والإفتراضي خاصة عندما يكون عدد الزوار بالمئات أو الآلاف .

قد يكون استعمال الذاكرة أسرع ولكن المشكلة أنه من السهل مسح البيانات منها ببساطة .

الطريقة الثالثة قد تكون أكثر الطرق مرونة ، ولكنها معقدة وصعبة جداً ، وهي تعطيك مرونة لتخزين البيانات في أي وسائط مدعومة من قبل الـPHP مثل قواعد بيانات mysql و oracle .

الذي افترضه الآن أنك قمت بوضع قيمة هذه الخاصية إلى files

إعداد مكان التخزين

```
session.save_path (path/to/directory)
```

هذه الخاصية مفيدة إذا كنت قد ضبط الإعداد السابق إلى files
تقوم هذه الخاصية بتحديد مكان التخزين على السيرفر ومن الأفضل أن تقوم بتحديد مكان التخزين بعيداً عن مجلد السيرفر لكي تمنع تصفح هذه الملفات .

الإنشاء التلقائي للـsession

```
session.auto_start (0 | 1)
```

هذا الإعداد يقوم بتحديد إذا ما كان الـsession سيتم إنشاؤه تلقائياً عند كل زيارة للموقع أو لأي صفحة من صفحاته بدون إدراج كود الـsession في كل صفحة ... وعلى ذلك فإنك تقوم بوضع القيمة إلى (1) إذا أردت ذلك .
وعلى إفتراض أنك لا تحتاج إلى أن تجعل الـPHP يقوم بعمل session لكل صفحة تلقائياً ومن غير طلب فستقوم بوضع قيمة هذا الإعداد الى (0)

الـSID

عندما يقوم الزائر بزيارة صفحتك فإن الـsession يستطيع تتبع هذا الزائر وعدد المرات التي قام فيه الزائر بالدخول لليوم الواحد ، يقوم الـPHP بعمل SID (session identefier) أو رقم معرف تلقائي بشكل افتراضي عندما تقوم بطلب إنشاء session بالزائر ، وكل رقم معرف يختلف عن الآخر تماماً ، إن رقم المعرف الذي ينشئه الـPHP شبيه للشكل التالى :

```
fc94ad8b1ee49ef79c713ee98ac1fcc4
```

هناك طريقتين يستطيع بها الـPHP متابعة الـSID للمستخدم :
1 - عن طريق المتابعة والتخزين بتسلسل في الكوكيز .
2 - عن طريق اتباع رقم المعرف بعنوان الصفحة في الانترنت .

سنأخذ أمثلة عن كلا الطريقتين :

1 - استخدام الكوكيز
بالطبع هذه هي أكثر الطرق شيوعاً للحصول على ترابط بين المستخدم والموقع وهي الأسهل ، ولكن يجب أن تضع في اعتبارك أن المستخدم قد يكون قد ألغي أو منع ميزة الكوكيز في المتصفح أو قد يكون متصفحه لا يدعم الكوكيز .
خذ في اعتبارك أن بعض المتصفحات لا تسمح بأن يزيد حجم الكوكيز عن 5 كيلوبايت .

هناك بعض الإعدادات البسيطة في ملف php.ini التي يجب معرفة معلومات عنها قبل البدء باستخدام الـsession مع الكوكيز :

```
session.use_cookies (0 | 1)
```

هذه الخاصية تحدد ماذا كان يمكنك استخدام الكوكيز مع الـsession أو لا وعند وضع القيمة (0) فهذا يمنعك من استخدام الكوكيز مع الـ(session) وأما اذا كانت قيمته (1) فهذا يسمح باستخدام الكوكيز مع الـsession

```
session.name (Default: PHPSESSID)
```

هذا الإعداد يقوم بتحديد اسم الكوكيز الذي سيحتفظ برقم المعرف (SID) والإعداد الافتراضي هو PHPSESSID ولن اقوم بتغيير هذا الإعداد لكي تستطيع فهم المثال الذي سأطرحه بعد قليل

```
session.cookie_lifetime (Default: 0)
```

يقوم هذا الإعداد بتحديد المدة التي سيبقي فيها الكوكيز الذي يحتفظ بقيمة الـ(SID) والإعداد الافتراضي هو صفر ، أي أنه سيتم مسح الكوكيز تلقائياً بعد اغلاق المستخدم لنافذة المتصفح مباشرة

```
session.cookie_path (Default: /)
```
يقوم هذا الإعداد بتحديد مسار دومين يتم تخزين الكوكيز له .. لا تقم بتغيير قيمته ودعه كما هو

```
session.cookie_domain (Default: null)
```
يقوم هذا الإعداد بتعريف اسم دومين يتم تخزين الكوكيز لصالحه .. والقيمة الافتراضية هي null ، لا تقم بتغييرها

ضع في اعتبارك انه اذا كانت قيمه الاعداد (session.use_cookies) تساوي واحد فان لا داعي لاستدعاء الدالة set_cookie() لإعداد الكوكيز بل سيتم اعدادها تلقائياً بواسطة الـPHP

2 - الإضافة أوالكتابة إلى عنوان الصفحة
إن إضافة عنوان الـSID إلى عنوان الصفحة يعتبر من الأشياء البشعة جداً رغم أن طريقته سهلة ومفيدة في حالة ما إذا كان الكوكيز غير مدعوم في المتصفح بشكل جيد
مثال :

```
<a href="configure.php?<?=SID?>">Go to the configuration page</a>
```
بهذه الطريقة نقوم بإضافة المتغير المرجعي SID الذي سيقوم بإعطاء رقم معرف للمستخدم .

متابعة الـsession

لقد أخذنا حتي الآن معلومات تجعلنا ندخل عالم البرامج المسيره بالـsession بدون خوف ، سأبدأ الآن في طرح بعض الأمثلة البسيطة التي تثبت لديك بعض المفاهيم الأساسيه في الـsession ... سأشرح في هذا المثال كيفية إنشاء الـSID وتخزينه لاستعماله لاحقاً ، وخلاصة السيناريو للصفحة أننا نريد من المستخدم أن يفهم أنه يستطيع تخصيص لون الخلفية الذي يريد أن يشاهد به صفحات موقعنا ... سأقوم بتخزين قيمة مبدئية في المتغير الذي يقوم بتحديد لون الصفحة ، أنا افترض طبعاً أن المتصفح يدعم الكوكيز :
سكربت يقوم بانشاء وتسجيل متغير session

```
<?
session_start();
session_register("zx");
session_register("co");
$zx=10;
$co++;
echo 'مرحبا بك في موقعنا أيها الزائر الكريم;"<br>".'

echo "عدد زياراتك لهذه الصفحة; $co ."=
echo "<br>";
echo '<a href="php2.php">الصفحة الثانية;</a>'
?>
```
اقصد بالجلسة هي الـ(session) وإن كانت الترجمة غير صحيحة ولكن فقط نأخذه كمصطلح .
متغير الجلسة هو الـ(session-variable) أو متغير الـsession أو سمه ما شئت .

الشرح
يقوم هذا السكربت في البداية بإنشاء متغير اسمه (zx) ومتغير اسمه (co) وقمنا بإعطاء القيمة (10) للمتغير (zx) وقمنا بزيادة القيمة الموجودة (وهي الصفر) في (co) بواحد وكتبنا مرحبا بك ايها

الزائر الكريم في موقعنا ، ثم قلنا له إن عدد زياراتك لهذه الصفحة هي قيمة المتغير(co) ثم اعطيناه رابط للصفحة الثانية .
في الواقع إن هذه المتغيرات وقيمها يتم الاحتفاظ بها في كوكيز له اسم خاص قمنا بتحديده سابقاً من ملف PHP.ini ، وهذا الكوكيز يحتفظ بقيمة الـSID للـsession .

نحن لا نقوم بإخبار الـPHP أين سيحتفظ بقيمة المتغيرات لأننا بدأنا بكلمة الـ:

```
session_start();
```

وعلى هذا فإن الـPHP سيفهم أنه سيقوم بتخزين القيمة في الكوكيز الخاص بالـsession .
قمنا بجعل المتغير co كعداد بسيط لعدد المرات التي سوف نقوم بها بزيارة الصفحة فعند عمل تحديث للصفحة سيتم زيادة العداد بمقدار واحد

```
$c++;
```

وطبعاً قبل زيادة العداد بقيمة واحد فإنه يتم حساب القيمة السابقة للمتغير عند إنشاءه تلقائياً ...
ومن ثم يتم الزيادة وبعد ذلك طباعة القيمة .
كتابة رقم الـSID

اكتب الآن الكود التالي واحفظه باسم php2.php

```
<?
session_start();
echo $PHPSESSID ."<br>";
echo $zx;
?>
```

في هذه الصفحة نقوم بطباعة قيمة الـSID وذلك بطباعة قيمه المتغير $PHPSESSID (الذي هو اسم الكوكيز الخاصة بالـsession) .
بعد ذلك قمنا في النهاية بطباعة قيمة المتغير $zx لكي ألفت نظرك بأن الكوكيز ما زال يحتفظ بها ولم يفقدها لأننا قد حددنا الإعداد في ملف php.ini الخاص بوقت الكوكيز الـ 3600 أي لمده ساعة ثم بعد تلك الساعة سيتم مسح الكوكيز ولن يمكنك استرجاع قيمة أي متغير :

```
session.cookie_lifetime = 3600
```

واضف إلى معلوماتك أنه لا يمكنك قراءة القيم للكوكيز الخاصة بالـsession إلا عن طريق إضافه الأمر

```
session_start();
```

يجب أن تبدأ بهذا الأمر دائماً إذا أردت قراءة قيم المتغيرات التي يحتفظ بها الكوكيز الخاص بالـ session .

مسح متغير من الـsession

كل ما عليك فعله هو استخدام هذه الدالة :

```
session_unregister(variable name);
```

تقوم بوضع اسم المتغير في مكان الـ(variable name)
مثال :

```
session_unregister(“brn”);
```

سيقوم هذا الأمر بمسح المتغير (brn) من الكوكيز الخاصة بالـ(session)

قراءة قيم المتغيرات في الـكوكيز الخاصة بالـsession

كل ما عليك فعله هو استخدام الدالة :

```
session_encode();
```

مثال :

```
<?
session_start();
session_register("bgcolor");
session_register("name");
session_register("email");
$bgcolor = "#8080ff";
$name = "alfareees almolthem";
$email = "php@php.com";
$e = session_encode();
print "The encoded string is: $e";
?>
```

بهذا السكربت نكون قد أنهينا درسنا عن مقدمة بسيطة للـsession . هذه مجرد مقدمة ولكي نستطيع أن نتعمق بالـsession فيجب علينا أن نتعلم شيئاً عن قواعد البيانات .

سيتم التطرق في هذا الدرس بمشيئة الله الى ما يلي :

- عبـاراة الـ if .
- عبـاراة الـ else .
- عبـاراة الـ elseif .
- عبـاراة الـ switch .
- حلقة التكرار while .
- حلقة التكرار for .
- حلقة التكرار do while .

- عبـاراة الـ if :

استخدام الـ if في كتابة السكربات شي أساسي ، وكما في لغات البرمجة الأخرى فإن الـ PHP تتبع نفس الاسلوب في كتابة الـ if ، فيمكن تحديد شرط معين مقترن بالـ if وبالتالي اذا كان الشرط صحيحاً (true) فسيتم تنفيذ الاسطر المحددة ، وبتفصيل أكثر يجب وضع الشرط بين قوسين ( ) ، ووضع الاسطر المطلوب تنفيذها بين العلامات { } ، مع ملاحظة أنه يمكن التخلي عن العلامات { } في حال وجود سطر واحد فقط .

فلنفترض وجود نموذج بريدي (Mail Fourm) ، يحتوي على الإسم والبريد والرسالة ، ونرغب في معرفة ما اذا كان المرسل قد ملأ جميع الحقول وبالتالي إرسـال الرسـالة ، او انه لم يفعل ذلك وبالتالي عرض رسـالة (فضلاً قم بتعبئة البيـانات كاملة) ، لعمل ذلك نحتاج معرفة أسمـاء المتغيرات في النموذج ، ولذلك فلنفترض أن المتغيرات كالتالي :
(الإسم $name) ، (البريد $email) ، (الرسـالة $later) ، ولعمل الشرط الأول (اذا كان الإسم لم يُدخل فلن يتم ارسـال الرسـالة) :

```
<?
 if ( $name == "" )
"; فضلاً قم بتعبئة البيـانات كاملة echo "
 ?>
```

والمعنى أنه إذا كان المتغير $name لا يحتوي على أي قيمة (أي فراغ) فسيتم تنفيذ السطر التالي وطبـاعة الجملة ، مع ملاحظة أن المطلوب تنفيذه هم سطر واحد فقط ولذلك لم نستخدم { } ، بل في حالة وجود أكثر من سطر يجب استخدامها كالتالي :

```
<?
if ( $name == "" )  {
<br> "; فضلاً قم بتعبئة البيـانات كاملةecho "
 "; لم تقم بإدخـال الإسمecho "
}
?>
```

- عبـاراة الـ else :

هذه العبـارة تتيح امكانية وجود اجراء ثاني لعدم تحقق الشـرط ، ففي مثالنا السـابق كان الاجراء طباعة الجملة اذا تحقق الشـرط ، ولكن في حالة عدم تحقق الشـرط فلن يكون هناك اجراء لتنفيذه ، بل ان الاجراء سـيتم تنفيذه اذا تحقق الشـرط ومن ثم سـيتم اكمال بقية الاسـطر ، وفي حالة مثل هذه الحالة يتم اسـتخدام الـ else لوضع اجراء آخر في حالة عدم تحقق الشـرط ، وبالمثال يتضح المقال :

```
<?
if ( $name == "" ) {
"; فضلاً قم بتعبئة البيـانات كاملةecho "
}
else
{
"; تم ارسـال الرسـالة ، شـكرا لكecho "
}
?>
```

في هذا المثال سـيتم طباعة الجملة (فضلاً قم بتعبئة البيـانات كاملة) اذا تحقق الشـرط أن المتغير $name لا يحتوي على أي قيمة ، وسـيتم طباعة الجملة (تم ارسـال الرسـالة ، شـكرا لك) في حالة عدم تحقق الشـرط ، أي في حالة وجود قيمة في المتغير $name ، مع ملاحظة أن هذا المثال يحتوي على شـرطين وليس شـرط واحد ، فالظاهر هو شـرط واحد ($name == "") ولكن العبـارة else تعتبر شـرطاً بحد ذاتها ولو لم يكن هذا الشـرط مكتوباً ، وكما هو واضح فمعنى هذا الشـرط هو (إذا كان غير ذلك) فقم بطباعة الجملة .

يمكن أن يكون الشـرح غير واضح تماماً ، ولكن أهمية فهم الطريقة سـتتضح في الأسـطر القليلة القادمة .

- عبـاراة الـ elseif :

في العبارة السـابقة ذكرنا أنه يوجد شـرطين واجرائين ، أحد هذين الشـرطين غير مكتوب بل هو مفهوم من ادراج العبـارة else ، وفي حالات كثيرة لا يكفينا مجرد شـرطين واجرائين لاتمام بعض السـكربات المعقدة ، فلذلك يمكن نسـتخدم العبـارة elseif مع الـ if لعمل مثل هذه السـكربات ، فلو افترضنا أن لدينا عداد لزوار الموقع ونريد اظاهر العداد بحيث يتم قراءته بشـكل جيد ، اي بمعني اخر اذا كان عدد الزوار (1) فسـيتم طباعة الجملة (عدد الزوار : زائر واحد فقط) واذا كان (2) فسـيتم طباعة الجملة (عدد الزوار : زائرين) ... وقس على ذلك ، فعندما يكون عدد الزوار (1) فسـيتم عرض الجملة الأولى فقط وعندما يكون عدد الزوار (2) فسـيتم عرض الجملة الثانية فقط ، وهكذا لبقية الشـروط .

بافتراض أن المتغير ($counter) هو عداد الزوار ، فالمثال التالي يبين ما تم شرحه سـابقاً :

```
<?
if ( $counter == 1 ) {
 "; عدد الزوار : زائر واحد فقطecho "
}
elseif ( $counter == 2 ) {
 "; عدد الزوار : زائرينecho "
}
elseif ( $counter >= 3 && $counter <= 10 ) {
"; عدد الزوار $counter : زوارecho "
}
else {
"; عدد الزوار $counter : زائرecho "
}
?>
```

كما هو واضح في المثـال السـابق سـيتم ما يلي :

الشرط : العداد يساوي 1
الإجراء : طباعة (عدد الزوار : زائر واحد فقط)
الشرط : العداد يساوي 2
الإجراء : طباعة (عدد الزوار : زائرين)
الشرط : العداد أكبر أو يساوي 3 و اصغر أو يساوي 10
الإجراء : طباعة (عدد الزوار : (العداد) زوار)
الشرط : العداد لا يحقق أي من الشروط
الإجراء : طباعة (عدد الزوار : (العداد) زائر)

ملاحظة بسيطة فقط ، وهي على العلامة && التي تعني (و) ، وهي من علامات الجمع بين جملتين ، فيجب أن تكون الجملتين صحيحتين لتحقق الشرط .

- عباراة الـ switch :

هذه العبارة قريبة جداً من العبارة if ، ولكن يمكن استخدام اكثر من شرطرين باسلوب أخر ، غير انه يجب اسناد قيمة معينة للـ case وهي هنا بمثابة الشرط ، لكي يتم تنفيذ الاجراء المحدد لذلك الشرط أو الـ case ، وفي النهاية الأمر يعود الى المصمم وايهما يفضل ، وكما في المثال السابق يمكن كتابة مثال بالـ switch بنفس الطريقة ، والمشكلة الوحيدة هي كما قلنا أنه يجب اسناد قيمة معينة لكل case وبالتالي فإن الشرط الثالث من المثال السابق يجب تفريقه لكل قيمة من (3 الى 10) ، وهذه العميلة مجهدة لانه يجب كتابة سطر لكل قيمة كما يلي :

القيمة : 3
الـ case : 3
الاجراء : طباعة (عدد الزوار : (العداد) زوار)
القيمة : 4
الـ case : 4
الاجراء : طباعة (عدد الزوار : (العداد) زوار)
القيمة : 5
الـ case : 5
الاجراء : طباعة (عدد الزوار : (العداد) زوار)
..... الخ ...

وفي المثال التالي ساتغاضى عن الشرط الثالث بكامله ، واذكر بقية الشروط والحالات لمجرد فهم طريقة عمل هذه العبارة :

```
<?
switch ($counter)
{
case "1";
 "; زائر واحد فقط : عدد الزوارecho "
break;
case "2";
 "; زائرين : عدد الزوارecho "
break;
default;
"; زائر : $counter عدد الزوارecho "
break;
}
?>
```

استخدمنا في هذه المثال بعض الجمل وتعني ما يلي :

Switch وتكتب في البداية مع ادراج اسم المتغير الذي سيتم عمل الشروط عليه .
Case أي في حالة (....) ويكتب بجانبها الشرط .
Break وتعني ايقاف العملية والخروج من الشرط بعد تنفيذ أحد الإجراءات .
Defaukt وهي تقابل العبارة else أي بمعنى أنها لأي حالة لم يتم ذكرها في الشروط .

- حلقة التكرار while :

وهي ابسط حلقات التكرار على الأطلاق ، بحيث تاخذ شرط واحد فقط وتبني على تنفيذ ما بين علامات الشروط { } ، والفرق الوحيد بينها وبين الـ if هو انها ستقوم بتنفيذ الاجراءات طالما كان الشرط صحيحاً ، وهذا يعني احتمال تنفيذ الإجراء أكثر من مرة ، وهذ الدالة مفيدة في ادراج الحقول من الجدوال وغيرها من الاستخدامات ، بحيث لو افترضنا وجود جدول معين في قاعد بيانات ونريد ادراجه في صفحة PHP ، فسيكون من اهم خيارات الاستخدام هذه الدالة ، وباذن الله سيتم التطرق لقواعد البيانات في الدروس القادمة ، وفي الوقت الحالي ساذكر مثال بسيط على هذه الدالة لفهم طريقة استخدامها :

```
<?
$total = 10;
while ( $total <= 50 )
{
<br>"; 50 العدد أقل من echo "
$total +=10;
}
?>
```

كير بسيط يمكن معرفة أن الجملة (العدد أقل من 50) سيتم طباعتها 5 مرات ، لان حلقة التكرار while قامت بتنفيذ الاجراء طالما أن الشرط صحيح ، وفي المرة الأولى كان المتغير ($total) يساوي (10) والشرط صحيح لان الـ ($total) فعلاً اصغر أو يساوي الـ (50) ، فتم تنفيذ ما بين علامات الشرط ، ومن ذلك زيادة متغير المجموع ($total) بقيمة (10) ومن ثم الرجوع والمقارنة من جديد ، وفي هذه الحالة صار المتغير ($total) يساوي (20) وأيضاً الشرط صحيح وبالتالي الدخول مرة أخرى وتنفيذ الأجراء .... وهكذا حتى يتم الوصول الى أن قيمة الـ ($total) يساوي (50) وبالتالي الشرط صحيح ، ومن ثم تصبح قيمة الـ ($total) تساوي (60) وفي هذه الحالة يتم ايقاف تنفيذ الاجراءات لأن الشرط غير صحيح .

- حلقة التكرار for :

يوجد طريقة أسهل للتعامل مع المثال السابق ، فاستخدام حلقة التكرار while كانت القيمة الابتدائية للمتغير ($total) في سطر ، والشرط في سطر والزيادة على المتغير في سطر أخر ، وبالتالي زيارة في عدد الأسطر عن ما يمكن استخدامه مع حلقة التكرار for ، فالمثال التالي يبين طريقة أخرى لاستخدام مثال الـ while بطريقة أسهل :

```
<?
for ( $total = 10;  $total <=50; $total +=10 )
{
<br>"; 50 العدد أقل من echo "
}
?>
```

وللتوضيح فان تركيب الـ for هو على الشكل التالي :

```
for ( القيمة الأفتراضية؛ الشرط؛ مقدار الزيادة ) ;
{
الإجراء المطلوب تنفيذه
}
```

- حلقة التكرار do while :

وهي نسخة أخرى من الـ while والفرق الوحيد بينهما أن التأكد من الشرط وصحته من عدمها يتم بعد تنفيذ الاجراء وليس قبله كما في الـ while وكمثال عليها :

```
<?
$total = 10;
do
{
echo "العدد أقل من 50 <br>";
$total +=10;
}
while ( $total <= 50 );
?>
```

قواعد البيانات الـ Mysql

سنتطرق الى مفاهيم عامة عن قواعد البيانات عموما وعن الـ Mysql خصوصاً ، لتكون بداية فهم لقواعد البيانات الهامة لأي لغة برمجة .

في البداية سنتعرف على مصطلح الـ RDBMS ، ونعني بذلك قواعد البيانات العلائقية ، والتي من خصائصها سهولة الوصول الى البيانات المخزنة فيها ، وسرعة اتمام عمليات الاستعلام المختلفة ، وبالاضافة الى المميزات الأخرى فان هذه النوع يعتبر الأكثر استخداماً في جميع التطبيقات سواء المستخدمة فيي الانترنت أو ذات الطابع البرمجي الخاص ، وبطبيعة الحال فإن الـ Mysql من هذا النوع .

ومن المهم معرفة بعض الاساسيات في الـ RDBMS ، والتي من شأنها تسهيل عملية فهمك التام لطريقة عملها والتعامل معها ..

1- الجداول Tables :
تعتبر أكبر جزء في قاعد البيانات ، وهي عبارة عن أعمدة وصفوف تحتوي على قيم معينة .

2- الأعمدة Columns :
لكل عمود في الجدول أسم خاص يختلف عن أسماء الأعمدة الأخرى في نفس الجدول ، ويجب ان يكون لكل عمود نوع خاص به يصف نوع البيانات التي ستخزن فيه ، وكم يظهر في الصورة ، فان عمود الرقم من النوع الرقمي Integer ، اما الحقلين الآخرين فهي نصوص Text .

3- الصفوف Rows :
كل صف من صفوف الجدول يحتوي على قيم مختلفة ويمثل معلومات متكاملة عن قطاع معين ، وفي مثالنا يمثل معلومات متكاملة عن سخص معين .

4- القيم Values :
وهي ما تحتوي عليه تقاطعات الصفوف بالاعمدة .

5- المفاتيح Keys :
وتعتبر من اساليب تسهيل الوصول الى المعومات في قواعد البيانات ، وفي مثالنا السابق نرى أن العمود Id يحتوي على ارقام متسلسلة لا تتكرر نهائياً بل أنها تتكون بشكل تلقائي عند ادراج أي صف جديد للجدول ، وبالتالي فإنها تعتبر المفتاح المناسب لكل صف من صفوف الجدول لضمان عدم الالتباس في اختيار الصفوف .

فلو افترضنا أن لدينا جدولين في قاعدة بيانات ، يحتوي الجدول الأول على معلومات عن الدروس مفصلة على عدة حقول لتلك الدروس ، على سبيل المثال :
الرقم (id) ، الدرس (lesson) ، رقم الكاتب (Key_author) ..
ويحتوي الجدول الثاني على بيانات الأعضاء كما يلي :
الرقم (Key_author) ، الاسم (name) ..

والمطلوب هو طريقة لربط الجدولين ، بحيث أن رقم الكاتب في جدول الدروس (Key_author) يدل على اسم الكاتب في جدول الاعضاء (name) .

بالتدقيق في المثال يتضح أن الحقلين (أو العمودين) Key_author في كلا الجدولين هو مفتاح الربط بينهما ، ولذلك يمكن الوصول الى اسم الكاتب اعتماداً على رقمه من جدول الدروس ، وبالتالي الربط بين الجدولين .

لن اتحدث طويلاً عن مقدمات قواعد البيانات Mysql ، ولكن بهذه المقدمة البسيطة يمكن على الاقل تصور بعض الاساسيات حول قواعد البيانات عموماً والـ Mysql خصوصاً ، ومن وجهة نظري فالاهم هو كيفية التعامل مع قواعد البيانات بما يخدم احتاجاتنا مع الـ PHP ، ولذلك ساتطرق في هذ الدرس الى نقطة هامة جداً وهي ادارة قواعد البيانات ، وأعني بذلك عملية انشاء قواعد البيانات والجداول والتحكم في الحقول والبيانات وغيرها ، لتكون الاساس للتعامل مع قواعد البيانات لاحقاً عن طريق الـ PHP ، ولعمل ذلك يوجد عدة طرق من اهمها الطريقة التقليدية المباشرة بالاعتماد على نظام الدوس في ذلك وبدون استخدام أي برامج أخرى للادارة .

الاتصال بالـ Mysql ، والتعامل معها :

كما قلنا أن الطريقة التقليدية هي الاتصال بقواعد البيانات عن طريق سيرفر الـ Mysql وبدون استخدام أي مكونات أخرى ، ولعمل ذلك نحتاج أن نعرف مسار سيرفر الـ Mysql على الجهاز المستخدم بعد عملية التثبيت ، كما قمنا بذلك في درس المقدمة ، وعادة يكون المسار كالتالي (C:\mysql\bin) ، وبذلك يمكن تشغيل البرنامج mysql.exe من داخل الـ Dos .

عموماً طريقة الاتصال بقاعدة البيانات هي كالتالي :

```
mysql –h HostName –u UserName –p
```

مع استبدال الـ HostName باسم السيرفر لديك ، سواء كان السيرفر على نفس الجهاز وفي هذه الحالية تكتب localhost ، أو أن السيرفر الذي تود الاتصال به ليس على نفس الجهاز وبذلك تكتب المسار الكامل لاسم السيرفر (HostName) ، ومع استبدال الـ UserName باسم المستخدم الخاص بالـ Mysql لديك ، بعد ذلك سيتم طلب كلمة المرور الخاصة بقاعدة البيانات بعد الضغط على Enter ، قم بادخالها وسيتم فتح الاتصال بالـ Mysql ، كما يمكن كتابة mysql فقط ليتم فتح الاتصال بقاعدة البيانات فقط اذا كنت تعمل على نفس الجهاز وليس جهاز أخر .

سيظهر المؤشر الخاص باوامر الـ Mysql كالتالي :

```
mysql>
```

وبهذا نكون وصلنا الى المكان المطلوب لكتابة اوامر الـ Mysql والتحكم بها .

الأمر الأول الذي سنقوم بكتابته يقوم باستعراض قواعد البيانات الموجودة على السيرفر والامر هو :

```
show databases;
```

بعد كتابة هذا الأمر (بعد مؤشر الـ mysql>) ، سيتم استعراض قواعد البيانات في السيرفر الذي قمنا بالاتصال به ، وفي حالة عدم وجود أي قاعدة بيانات قمت باعدادها من قبل ، فان من الطبيعي أن تجد قاعدتي بيانات موجودة بشكل تلقائي عند تثبيت السيرفر Mysql ، وتلك القاعدتان هي test – mysql .

ولمحاولة فهم الموضوع بشكل أكبر ، سنقوم بالتطرق الى مثال يبين كيفية انشاء قاعدة بيانات ، وكيفية الدخول لها والتعامل معها وانشاء الجداول ، ومن ثم حذفها ..

بعد استعراض قواعد البيانات بالأمر السابق ، سنقوم بانشاء قاعدة بيانات باسم PHP ، ولعمل ذلك قم بكتابة الأمر التالي :

```
create database PHP;
```

لو قمنا بكتابة الأمر السابق (show database) سنرى أن قواعد البيانات أصبحت 3 باضافة القاعدة PHP الى القاعدتين test – mysql ، ولاستخدام اي منها نقوم بكتابة الأمر التالي في مثالنا مع القاعدة PHP :

```
use PHP;
```

وهذه يعني الدخول في قاعدة البيانات PHP واستخدام المؤشر (mysql>) لكتابة الأوامر المتعلقة بالتعامل مع قاعدة بيانات بعينها .

أول هذه الاوامر هو أمر انشاء جدول في قاعدة البيانات ، وهذه الأمر يحتاج الى تفصيل دقيق لبعض الخصائص مثل اسماء الحقول وانواع البيانات فيها ، وبعض الاشياء الأخرى ، عموماً قم بكتابة الأمر التالي وساقوم بشرح كافة التفاصيل بعد المثال :

```
create table users (
id Int not null auto_increment Primary Key,
name text not null,
counter int
);
```

شرح المثال :

- قمنا بكتابة (create table users) وهذا يعني انشاء جدول باسم users .
- القوس) يعني بداية تسمية حقول الجدول وخصائص تلك الحقول .
- السطر الأول من اسماء الحقول هو (id) والرمز (int) يعني وصف نوع البيانات التي ستخزن في الحقل (id) ، وهي في هذه الحالة تعني نوع البيانات الرقمية ، اما الرمز (not null) فيعني عدم امكانية أن يكون هذا الحقل فارغاً ، بل يجب أن يحتوي على قيمة ، والـ (auto_increment) يجعل الحقل يحتوي على قيم متسلسلة يستحيل تكرارها ، وسيبدأ من الرقم 1 ويبدأ بالزيادة بمقدار واحد في كل مرة يتم ادخال صف جديد الى هذا الجدول ، وفي النهاية الرمز (Primary Key) يعني أن الحقل هو المفتاح الرئيسي لهذا الجدول أو بمعنى أنه سيتم التفريق بين صفوف الجدول اعتماداً على هذا الحقل ولهذا وضعنا (auto_increment) لضمان عدم اختلاط البيانات .
- السطر الثاني يحتوي على اسم الحقل (name) ونوع البيانات (text) أي نصي ، ونفس الرمز السابق الذي ذكرناه وهو (not null) .
- السطر الثالث يحتوي على اسم الحقل (counter) ونوع البيانات (int) ، ولاحظ أننا لم نذكر (not null) وبالتالي يمكن أن يكون هذا الحقل فارغاً لا يحتوي على أي قيمة ، ولن يكون هناك أي تعارض أو مشكلة بعكس الحقلين السابقتين .
- في السطر قبل الأخير ، أي قبل علامة الاغلاق (، سيكون بدون فاصلة .
- السطر الأخير يحتوي على اقفال عملية انشاء الجدول بالعلامة)؛ .

عموماً هذا المثال يعطي نبذة بسيطة عن كيفية اجراء مثل هذه الاوامر ، وسنتطرق الى بقية الأوامر في الأسطر القليلة القادمة .

يمكنك استعراض الجداول الموجودة في قاعدة بيانات عن طريق الأمر :

```
show tables;
```

ولو قمت بتطبيق ذلك على المثال السابق فسترى أن الجدول users موجود في قاعدة البيانات PHP التي قمنا بانشاءها .

يمكن كذلك استعراض خصائص الجدول السابق users الذي قمنا بانشاءه في المثال السابق ، عن طريق الأمر التالي :

```
describe users;
```

سترى أن حقول الجدول وخصائص كل جدول ظهرت لك بشكل واضح .

- التعامل مع بيانات الجداول :

بقي أن نذكر الطرق التي يمكن من خلالها ادخال البيانات الى الجدول users ، بل وكيفية التعامل مع تلك البيانات بالتعديل والحذف وغير ذلك ، وكما قلنا سابقاً أن هذه الاساسيات مفيدة جداً في البرمجة بلغة الـ PHP ، بل إن فهم هذه الطرق هو المفتاح الاساسي للتعامل مع قواعد البيانات عن طريق البي اتش بي ،

عموماً أول تلك الأوامر هو اضافة صف جديد الى الجدول ، وهذا ما يبينه المثال التالي :

```
insert into users set
name = "Ahmad";
counter = 3
;
```

مع ملاحظة أن users هو اسم الجدول ، name اسم الحقل (العمود) الأول ، counter اسم الحقل (العمود) الثاني ، كما تلاحظ أن الحقل id لم نتطرق له ، لاننا في اعدادنا للجدول ذكرنا أن الحقل id (auto_increment) أي ستضاف اليه القيم بشكل تلقائي وبشكل منظم ، كما قلنا في كل مرة يزيد العداد بقيمة 1 ، و بطبعية الحال يمكنك القياس على هذا المثال باستبدال ما يجب استبداله من اسم الجدول (users) واسماء الحقول (name – counter) وكذلك البيانات بما يناسب الذي تريد القيام به .

هذا بالنسبة لاضافة بيانات جديدة الى جدول معين ، اما بالنسبة لاستعراض البيانات في الجدول فكما يلي :

```
select * from users;
```

ومعني select (اختر) ، ولذلك ستجد أن جميع البيانات التي في الجدول users قد تم سردها ، واذا كنت ملتزماً بالمثال السابق حرفياً فستجد أن البيانات التي اضفناها في المثال السابق ظهرت على شكل صف من صفوف الجدول ، وبالتالي كلما اضفت صفاً جديداً الى الجدول وقمت باستعراض البيانات تجد أن بياناتك قد تم تخزينها ، وينطبق الكلام السابق حول الاستبدال هنا ايضا ، فيمكن استبدال اسم الجدول users باي اسم لجدول في قاعدة البيانات المستخدمة ، وللتاكد من اسماء الجداول قم باستخدام الطريقة السابق ذكرها وهي (show tables) .

النقطة الأخيرة التي ساتطرق لها هي ما يجب معرفته حول الأمر select وهو كثرت استخدامه في التعامل عن طريق البي اتش بي ، وبالتالي يجب عليك فهم طريقة كتابته بشكل كامل ، بالاضافة الى خيارات الاختيار إن صح التعبير ، وهي ما يتم كتابته بعد الجملة السابقة من خيارات تحدد طريقة اختيار البيانات من شروط وترتيب وحدود وهذا ما ساذكره في الاسطر القليلة القادمة .

فلفترض أن الجدول السابق يحتوي على أكثر من صف من البيانات بالشكل التالي :

اما البيانات التي نود جلبها فهي كما يلي لكل نقطة على حدة :

1- بيانات الاعضاء الذين ليس لهم أي موضوع .
2- بيانات الاعضاء الذين لهم مواضيع أكثر من 5 مرتبين من الاكثر الى الاقل .
3- بيانات العضو Ahmed .
4- بيانات جميع الاعضاء مرتبين حسب الاسم .
5- بيانات العضو الأكثر مواضيعاً .

سنأخذ كل حالة على حدة ، أما **الحالة الأولى** فيمكن التعامل معها كما يلي :

```
select * from users where counter=0;
```

الزيادة التي قمنا بوضعها هي (where counter=0) أي بحيث أن الحقل (counter) يساوي صفر ، وبالتالي سيتم اهمال أي صف من البيانات التي لا يحتوي الحقل (counter) فيها على القيمة صفر ، وسيتم جلب البيانات التي يحتوي هذا الحقل فيها على صفر .

الحالة الثانية :

```
select * from users where counter >= 5 order by counter;
```

في هذا المثال اضفنا الشرط (where counter =< 5) وهو واضح كما في المثال السابق ولكن تم تغيير الشرط لا اقل ولا اكثر ، اما الاضافة الأخرى فهي طريقة الترتيب وهي (order by counter) وتعني (قم بترتيب البيانات المختارة بحسب الحقل counter) ، وهناك طريقة أخرى للتحكم في الترتيب اما تصاعدي أو تنازلي وذلك باضافة كلمة asc ليكون الترتيب تنازلياً كما هو الحال في المثال السابق ، فسواء ذكرت ذلك أو سيتم اعتبارها تنازلياً بشكل تلقائي ، اما الاهم فهو طريقة الترتيب التصاعدي من الاقل الى الاكبر ويتم ذلك عن طريق كتابة الكلمة desc بعد الترتيب مباشرة لتصبح كما يلي :

```
select * from users where counter >= 5 order by counter desc;
```

الحالة الثالثة :

```
select * from users where name = "Ahmed";
```

لاحظ أن الفرق الوحيد هنا هو استخدام علامات التنصيص ، لان نوع البيانات نصية .

الحالة الرابعة :

```
select * from users order by name;
```

وقد أوردت هذا المثال لبيان أنه يمكن استخدام أحد الخيارات لجلب البيانات وترك باقي الخيارات ، فيمكن كما في المثال استخدام خيار الترتيب (order) وعدم استخدام الخيارات الباقية (where – limit) ، اما الخيار where فقد تطرقنا لنا سابقاً وتعرفنا على فائدته ، والخيار الأخر limit هي ما سيتم التطرق اليه في المثال التالي الخاص بالحالة الخامسة :

الحالة الخامسة :

```
select * from users order by counter limit 1;
```

والـ limit تعني عدد الصفوف المختارة ، أي لو قمنا بكتابة المثال السابق بدون الـ limit ستجد أن جميع البيانات سيتم اختيارها ، ولكن باستخدام الـ limit نقوم بتحديد عدد الصفوف التي سيتم اختيارها استناداً على طريقة ترتيبنا للبيانات ، فكما تلاحظ قمنا بترتيب البيانات بحسب الحقل counter ولم نذكر (desc) ولذلك فالبيانات يتم ترتيبها من الاكبر الى الاصغر ، وبالتالي فاختيارنا للحقل الأول يقضي باختيار بيانات الشخص الأكثر كتابة للمواضيع .

بقي أن نذكر طريقي التعديل والحذف ليكتمل الدرس ، وسنبدأ بطريقة التعديل على البيانات الموجودة في الجدول users من قاعدة البيانات PHP ، والمثال التالي يوضح الطريقة التي سيتم شرحها بعد المثال :

```
update users set
name = "Naser",
counter = 30
where name="Ahmad";
```

الجملة update تعني تحديث أو (قم بتحديث) ، والـ users هو اسم الجدول الذي نعمل عليه ، وفي السطر الثاني قمنا باسناد القيمة Naser الى الحقل name ، والسطر الذي يليه قمنا باسناد القيمة 30 الى الحقل counter ، ولكن لو توقفنا هنا بدون ذكر الصف الذي سيتم التعديل عليه ، سيتم

تعديل كافة الصفوف في الجدول مهما كان عددها ، ولذلك كتبنا في النهاية "where name="Ahmad ، بمعنى أن التغيرات السابقة ستحدث فقط على الصف من البيانات التي يحتوي فيها الحقل name على القيمة Ahmad .

ربما يكون المثال غير واضح بشكل كافي ، ولكن مع التمرس والمحاولة ستجد أن المسألة منطقية وواضحة بشكل كبير ، عموماً لم يبقى لدينا الا طريقة الحذف ، سواء كان لكل البيانات في الجدول ، أو لصف معين من البيانات وسنرى ذلك في المثالين التاليين ، وهما ما سنختم به هذا الدرس :

```
delete from users;
```

الأمر السابق كفيل بالغاء جميع الصفوف في الجدول users كما هو واضح ، ولذلك كن متأكداً من أن التجارب التي تقوم بها هي على بيانات غير هامة .

```
delete from users
where id = 1 ;
```

وهذا الحذف سيتم على الصف الذي يتحقق عليه الشرط ، وفي هذه الحالة على الصف من البيانات التي يحتوي فيها الحقل id على القيمة 1 .

الدوال (Function)

يوجد في PHP العديد من الدوال التي تقوم بوظيفة معينة (محددة) كذلك توجد إمكانية إنشاء دوال تؤدي وظيفة خاصة وحديثنا هنا عن هذا النوع من الدوال (كيفية إنشاء دوال)
الدالة تقوم بتنفيذ شئ معين حيث تأخذ (متغيرات – معطيات) ثم تقوم بمعالجة هذه المتغيرات وتخرج قيمة أخرى .

- الشكل العام – التركيب :

```
Function اسم الدالة ( المعطيات – المتغيرات – البارامتر )
{
 هنا يتم كتابة الكود
Return ( المعطيات – المتغيرات – البارامتر )  ;
}
```

- تعريف الدالة :

لكي نقوم بتعريف دالة نكتب كلمة function بعدها اسم الدالة وبعد الاسم نكتب المعطيات – المتغيرات بين قوسين .

مثال :

```
<?
Function aa($s)
?>
```

حيث aa هو اسم الدالة ، وبالتأكيد يمكن أن يكون أي اسم .
($s) هو (المتغر – المعطى – البارامتر) ، أي اسم من هذه كما تحب أن تسميه .
مع ملاحظة عدم وضع فاصلة منقوطة بعد هذا السطر .

بعد ذلك نقوم بكتابة كود الدالة (عمل الدالة) بين العلامتين { } ، كما يجب أن ننهي الدالة بكلمة return لإعلام الدالة بأن وظيفتها قد انتهت بالاضافة الى ذكر اسم المتغير المذكور في تعريف الدالة سابقا ..

مثال :

```
<?
Return($s) ;
?>
```

- استخدامات الدالة :

يمكن وضع الدالة في أي مكان في شفرة php في أولها أو آخرها بمعنى انه يمكن استدعاء دالة تم تعريفها في أخر الشفرة أو العكس .

- إظهار نتيجة الدالة (طباعة الدالة) :

نستخدم الأمر الخاصة بالطباعة echo أو print وبعده طبعا اسم الدالة ..

مثال :

```
<?
echo aa(5);
print aa(5);
?>
```

مثال كامل :

```
<?
 تعريف الدالة//
function aa($a)
{
$a=$a*$a*$a*$a;
return($a);
}
 طباعة ناتج الدالة عند ادخال الرقم 5 فيها//
echo aa(5);
?>
```

هذه الدالة تقوم بحساب عدد مرفوع لأس أربعة بمعنى أن العدد مضروب في نفسه أربع مرات اسم الدالة aa وعند طباعة مخرجات الدالة لرقم ، كتبنا أمر الطباعة قبل اسم الدالة والرقم المراد حساب الأس الرابع له بين قوسين (5) وهكذا إذا وضعنا أي رقم آخر سوف تقوم الدالة بحساب الأس الرابع للرقم مباشر وفي مثالنا هذا يتم طبع الرقم 625 .

نقطة أخرى هي أننا قمنا بتمرير قيمة ثابتة الى الدالة ، ولذلك يمكننا أن نمرر للدالة متغير كما في المثال التالي :

```
<?
function as($a)
{
$a=$a*$a*$a*3  ;
return($a) ;
}
$z=10 ;
echo as ($z) ;
?>
```

في هذا المثال تقوم الدالة بضرب العدد في نفسه ثلاث مرات ثم في الرقم 3 ، ونلاحظ أننا مررنا المتغير $z الى الدالة as وكتبناها جميعها في سطر طباعة نتيجة الدالة بالأمر echo . ولذلك تقوم الدالة في هذا المثال بضرب الرقم 10 في نفسه ثلاث مرات ثم في 3 يكون الناتج 3000 ومن ثم يتم طباعة الناتج ، وبطبيعة الحال كلما غيرنا قيمة المتغير أختلفت نتيجة الدالة .

العمليات الرياضية

هي نفسها العمليات التي درستها في المرحلة الابتدائية من (جمع + ، طرح - ، ضرب * ، قسمة /) والزائد عليهم التي لم تدرسه تقريبا هو باقي القسمة (%) ..

مثال شامل على كل العمليات في الـ PHP :

```
<?
$a = 6;
$b=2;
$c= $a +$b;
 8 سوف نحصل على ناتج الجمع//

$c= $a -$b;
 4 سوف نحصل على ناتج الطرح//

$c= $a * $b;
 12 سوف نحصل على ناتج الضرب//

$c= $a /$b;
 3 سوف نحصل على ناتج القسمة//

$a = 7;
$b=2;
$c= $a % $b;
 1 سوف نحصل على باقي القسمة//
?>
```

- عمليات Assignment :

احفظ القيمة في المتغير ، بمعنى خزن القيمة 3 في المتغير $a :

=

```
<?
$a = 3;
print $a;
 3 يطبع//
?>
```

+=

إضافة قيمة إلى قيمة في نفس المتغير :

```
<?
$a = 3;
$a += 3;
print $a;
 6 يطبع//
?>
```

-=

اطرح المقدار واحد من المقدار ثلاثة في المتغير a$

```
<?
$a = 3;
$a -= 1;
print $a;
```

```
// 2 يطبع
?>
```

=*

يضرب القيمة 3 بالقيمة 2 ويكون الناتج مخزن في نفس المتغير :

```
<?
$a = 3;
$a *= 2;
print $a;
 6 يطبع الناتج//
?>
```

=/

يقسم قيمة على قيمه أخرى :

```
<?
$a = 6;
$a /= 2;
print $a;
    3 يطبع ناتج القسمة//
?>
```

=.

دمج سلسلة حرفية :

```
<?
$a = "This is ";
$a .= "a test.";
print $a;
 يطبع الجملة التالية// :
// This is a test.
?>
```

- عوامل الإضافة و الطرح :

لو افترضنا أننا لدينا المتغير $a=3 و أردنا إضافة واحد إليه بحيث يصبح 4 أو طرح واحد منه بحيث يصبح 2 ، لدينا العوامل التالية :

$a++ ارجع قيمة a ثم اضف واحد إليها
a$++ اضف واحد إليها ثم ارجع القيمة
$a-- ارجع القيمة ثم اطرح واحد منها
a$-- اطرح واحد ثم ارجع القيمة

value++

يتم إضافة واحد إلى الرقم خمسة :

```
<?
$a = 5;
print ++$a;
 6 يطبع القيمة//
?>
```

++value

يرجع القيمة نفسها وفي استخدام ثاني تزيد القيمة واحد :

```
<?
$a = 5;
print $a++;
 6 طباعة الرقم//
print "<br>";
```

```
print $a;
 5 طباعة الرقم//
?>
```

value--

يطرح من القيمة واحد :

```
<?
$a = 5;
print --$a;
 4 يطبع الرقم//
?>
```

--value

يرجع القيمة نفسها وفي استخدام ثاني يطرح منها واحد :

```
<?
$a = 5;
print $a--;
 4 يطبع الرقم//
print "<br>";
print $a;
 5 يطبع الرقم//
?>
```

- عمليات المقارنة Comparasion Operators :

b == a المتغيران متساويان ..
b===a المتغيران متساويان و من نفس النوع ..
a$!=b$ المتغير الاول لا يساوي الثاني ..
b===! a المتغير الاول لا يساوي الثاني وليس من نفس النوع ..
a$ < b$ أكبر من ..
a$ > b$ أصغر من ..
a$ =<b$ أكبر من او يساوي ..
a$ =>b$ أصغر من او يساوي ..

== (تساوي)

تساوي القيمة المخزنة في المتغير الأول بالقيمة المخزنة في المتغير الثاني :

```
<?
$x = 7;
$y = "7";
 " . $y; تساويif ($x == $y) print $x . "
 7 تساوي 7 يطبع//
?>
```

=== (تساوي ومن نفس النوع)

تساوي القيمة المخزنة في المتغير الأول بالقيمة المخزنة في المتغير الثاني وتكون القيم من نفس النوع (حرفية – عددية) :

```
<?
$x = 7;
$y = 7;
if ($x === $y) print $x . " is identical to " . $y;
 is identical to 7 7 يطبع//
?>
```

!= (لا تساوي)

إذا كانت القيم المخزنة في المتغيرين غير متساويين :

```
<?
$x = 8;
$y = 4;
 " . $y; لا تساويif ($x != $y) print $x . "
 4 لا تساوي 8 //يطبع
?>
```

!== (لا تساوي ولا من نفس النوع)

إذا كانت القيم المخزنة في المتغيرين غير متساويين وليست من نفس النوع :

```
<?
$x = 8;
$y = 9;
 " . $y; ليست من نفس نوعif ($x !== $y) print $x . " i
 9 ليست من نفس نوع 8 //يطبع
?>
```

< (أقل من)

مقارنة بين قيمتين واحدة أقل من الاخرى :

```
<?
$x = 5;
$y = 9;
  " . $y; أقل منif ($x < $y) print $x . "
 9 أقل من 5 //يطبع
?>
```

> (أكبر من)

مقارنة بين قيمتين واحدة أكبر من الاخرى :

```
<?
$x = 9 ;
$y = 5;
  " . $y; اكبر منif ($x > $y) print $x . "
 5 أكبر من 9  //يطبع
?>
```

<= (أقل من ويساوي)

مقارنة بين قيمتين واحدة أقل من الاخرى أو مساوية لها :

```
<?
$x = 5;
$y = 5;
if ($x <= $y) print $x;
 5 القيمة //يطبع
?>
```

>= (أكبر من ويساوي)

مقارنة بين قيمتين واحدة أكبر من الاخرى و مساوية لها :

```
<?
$x = 7;
$y = 5;
if ($x >= $y) print $x;
 7 القيمة //يطبع
?>
```

العمليات المنطقية Logical Operations :

لكي تكون قيمة الشرط صحيحة فيجب أن تنطبق القواعد التالية الخاصة بكل عامل منطقي على حدة ، والعوامل هي :
(and) يجب تحقق الاثنين $a and $b
(or) يجب تحقق كلاهما او احدهما $a or $b
(Xor) يجب تحقق احدهما و ليس كلاهما $a xor $b
(!) نفي تحقق الشرط نفي لقيمة $a !a

ملاحظة : يمكن كتابة الـ (and) بالشكل التالي (&) والـ (or) بالشكل التالي (|) والـ (Xor) بالشكل التالي (^) ..

And (و)

إذا تحقق الشرطان ، بمعنى المتغير الأول يساوي 7 والمتغير الثاني يساوي 5 نفذ أمر الطباعة واطبع صحيح :

```
<?
$x = 7;
$y = 5;
"; صحيحif (($x == 7) and ($y == 5)) print "
 //يتم طباعة صحيح
?>
```

Or (أو)

إذا كان أحد الشرطين صحيح أو الاثنين صحيحين نفذ أمر الطباعة :

```
<?
$x = 7;
$y = 5;
if (($x == 7) or ($y == 8)) print "True";
 //يطبع True
?>
```

Xor

إذا تحقق أحد الشرطين وليس الاثنين معا ينفذ أمر الطباعة :

```
<?
$x = 7;
$y = 5;
if (($x == 7) xor ($y == 8)) print "True";
 //تحقق شرط واحد فقط فيتم طباعة كلمة True
?>
```

! (النفي)

إذا كانت جملة الشرط غير صحيحة نفذ أمر الطباعة :

```
<?
$y = 5;
if (! ($y == 10)) print "True";
    يطبع True لأن المتغير القيمة المخزنة فيه غير صحيحة
?>
```

&&

المعامل && له نفس وظيفة (and) لكن الاختلاف في ترتيب تنفيذ أولويات العمليات :

```
<?
$x = 7;
$y = 5;
if (($x == 7) && ($y == 5)) print "True";
 يطبع// True
?>
```

| |

المعامل || له نفس وظيفة (or) لكن الاختلاف في ترتيب تنفيذ أولويات العمليات :

```
<?
$x = 7;
$y = 5;
if (($x == 7) || ($y == 5)) print "True";
 يطبع// True
?>
```

دوال قواعد البيانات

لاهمية موضوع قواعد البيانات ، سوف نقوم في هذه الدورة بتغطية دوال قواعد البيانات وهي اثنتان وثلاثون دالة فإلى الدرس الأول :

1- الدالة mysql_connect :

```
integer mysql_connect(string host, string username,
string password);
```

تقوم هذه الدالة بالاتصال مع قاعدة البيانات وتعيد لك رقم يفيدك اذا كان لديك أكثر من اتصال بقواعد البيانات ، احتفظ به لاستخدامه في دوال أخرى تالية اذا كان هناك حاجة لذلك كما قلنا ، اما الوضع الطبيعي فلا يحتاج الا الى الاتصال بالطريقة السابقة فقط وبدون الاحتفاظ بأي رقم ، فقط مرر للدالة اسم الخادم واسم المستخدم وكلمة المرور ، ولكن يتوجب عليك بعد الانتهاء أن تغلق الاتصال باستخدام الدالة mysql_close

مثال :

```
<?
$link = mysql_connect("db.azzozhsn.f2s.com","mag","Pass");
?>
```

2- الدالة mysql_pconnect :

```
integer mysql_pconnect(string host, string username,
strin password);
```

هذه الدالة تقوم بما تقوم به الدلة السابقة إلا أنه لا يتوجب عليك إغلاق الاتصال ، مثال:

```
<?
$link = mysql_pconnect("db.azzozhsn.f2s.com","mag","Pass");
?>
```

3- الدالة mysql_select_db :

```
boolean mysql_select_db(string database, integer link);
```

تقوم هذه الدالة باختيار قاعد البيانات المحدد لها. مثال:

```
<?
mysql_select_db(string database, integer link);
?>
```

4- الدالة mysql_db_query :

```
boolean mysql_db_query(string database, string query,
integer link);
```

تقوم هذه الدالة بتنفيذ سطر SQL على قاعدة البيانات المفتوحة بالمعطى database مثال:

```
<?
$link = mysql_connect("db.azzozhsn.f2s.com","mag","Pass");
$Query = "DELETE FROM magazine";
$result = mysql_db_query("mag", $Query, $link);
?>
```

5- الدالة mysql_close :

```
boolean mysql_close(integer link);
```

تقوم هذه الدالة بقطع (إغلاق) قاعدة البيانات ، مرر لها رقم الاتصال المعاد من الدالة mysql_connect

مثال:

```
<?
 .. الاتصال بقاعدة البيانات//
$link = mysql_connect("localhost","mag","Pass");
 .. اغلاق الاتصال بقاعدة البيانات//
mysql_close($link);
?>
```

6- الدالة mysql_query:

```
integer = mysql_query(string query, integer link);
```

تقوم هذه الدالة بما تقوم به الدالة mysql_db_query تقريباً إلا أن الدالة mysql_query يقتصر عملها على قاعدة البيانات المحددة بالدالة mysql_select_db .
في حالة عدم تمرير رقم الاتصال فستعمل الدالة على الاتصال الأخير.

مثال:

```
<?
$link = mysql_connect("localhost","mag","Pass");
$query = "DELETE FROM magazine";
$result = mysql_query($query, $link);
?>
```

7- الدالة mysql_errno :

```
integer mysql_errno(integer link);
```

تقوم هذه الدالة بإعادة رقم آخر خطأ حدث في التعامل مع قاعدة البيانات.

8- الدالة mysql_error :

```
string mysql_error(integer link);
```

تعيد هذه الدالة رسالة الخطأ الحاصل في قاعدة البيانات .

9 - الدالة mysql_create_db :

```
boolean mysql_create_db(string databasename, integer link);
```

تقوم هذه الدالة بإنشاء قاعدة بيانات جديدة مرر لها اسم قاعدة البيانات ورقم الاتصال العائد من الدالة **mysql_connect** أومن الدالة **mysql_pconnect** ..
مثال:

```
<?
 الإتصال بقاعدة بيانات اسمها az حيث أن الفراغ هو الباسورد//
$link = mysql_pconnect("localhost", "az", "");
 انشاء قاعدة بيانات جديدة//
if (! mysql_create_db($link, "mag"))
{
") فشل إنشاء قاعدة البيانات الجديدة    print("
   exit();
}
?>
```

10- الدالة mysql_drop_db :

```
boolean mysql_drop_db(string databasename, integer link);
```

تقوم هذه الدالة بحذف قاعدة البيانات المحددة بالمعطى **databasename** ..

11- الدالة mysql_list_dbs :

```
integer mysql_list_dbs(integer link);
```

تقوم هذه الدالة بإعادة موشر لكل قواعد البيانات الموجودة في الخادم لغرض استعمالها مع الدالة **mysql_fetch_row** وأمثالها .

12- الدالة mysql_field_seek :

boolean mysql_field_seek(integer result, integer field);

تقوم هذه الدالة بتحديد الحقل المرر إليها رقمه . مثال :

```
<?
//حيث أن الفراغ هو الباسورد az الإتصال بقاعدة بيانات اسمها
$dbLink = mysql_pconnect("localhost","az","");
//أختيار قاعدة البيانات  Authers
mysql_select_db("Authers", $dbLink);
// اختيار جميع الحقول من الجدول Adress
$Query = "SELECT * FROM adress";
$result = mysql_query($Query, $dbLink);
//الانتقال الى الحقل الثاني اعتماداً على عملية الاختيار
mysql_field_seek($reslut, 1);
?>
```

13- الدالة mysql_field_name :

string mysql_field_name(integer result, integer feild);

تعيد هذه الدالة اسم الحقل المحدد بالرقم المرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول مثالها سيأتي بعد قليل .

14- الدالة mysql_field_type :

string mysql_field_type(integer result, integer feild);

تعيد هذه الدالة نوع الحقل المحدد بالرقم المرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول . المثال سيأتي بعد قليل أيضاً ..

15- الدالة mysql_field_len :

string mysql_field_len(integer result, integer feild);

تعيد هذه الدالة طول الحقل بالبايت المحدد بالرقم المرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول . المثال بعد قليل ..

16- الدالة mysql_field_flags :

string mysql_field_flags(integer result, integer feild);

تعيد هذه الدالة وصف الحقل المحدد بالرقم المرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول

17- الدالة mysql_list:

```
mysql_list(string database, string table, integer link);
```

المثال الشامل :

```
<?
//حيث أن الفراغ هو الباسورد az الإتصال بقاعدة بيانات اسمها
$link = mysql_pconnect("localhost", "az", "");
//ترتيب الحقول وجلبها
$result = mysql_list_field("mag", "table", integer link);
//حلقة تكرار للمرور على كل حقل
for ($a = 0; $a < mysql_field_num($result); $a++)
{
   print(mysql_field_name($result, $i);
   print(mysql_field_type($result, $i));
   print(mysql_field_len($result, $i));
   print(mysql_field_flags($result, i));
}
?>
```

18- الدالة mysql_fetch_field :

```
<?
object mysql_fetch_field(integer result, integer field);
?>
```

استخدم هذه الدالة لتحصل على معلومات حول حقول الجدول المراد، الحقول ترقم بدايةً من صفر
وصف الحقل مشروح في الجدول التالي:

الخاصة	الوصف
blob	إذا كانت TRUE فالحقل عبارة عن عن حقل بيانات كبير
maxlength	الطول الأقصى للحقل
multiple_key	تكون TRUE إذا كان الحقل مفتاحاً
name	أسم الحقل
not_null	تكون TRUE إذا كان الحقل لا يمكن أن يكون فارغاً
numric	تكون TRUE إذا كان الحقل يرقم تلقائياً
primary_key	تكون TRUE إذا كان الحقل يمثل مفتاحاً رئيساً
unque_key	تكون TRUE إذا كان الحقل يمثل مفتاحاً ثانوياً
zerofill	تكون TRUE إذاكان الحقل يملأ بالقيمة 0

19 - الدالة mysql_fetch_lengths :

```
<?
array mysql_fetch_lengths(integer result);
?>
```

استخدم هذه الدالة لتعيد مصفوفة تحتوي على الطول الأقصى لكل حقل محدد في المعطيresult.

```
<?
//Connect to server as azzozhsn no password
$link = mysql_pconnect("localhost","azzozhsn","");
//Select th magazine database
mysql_select_db("magazine",$link);
//Get name and id from magazine
$Query = 'SELECT name, id FROM magazine';
$result = mysql_query($Query, $link);
$length = mysql_fetch($result);
//Print length of the third column
print($lengths[2]);
?>
```

20 - الدالة mysql_fetch_array :

```
<?
array mysql_fetch_array(integer result);
?>
```

هذه الدالة تعيد مصفوفة تحتوي على قيم سجل وتنقل المؤشر إلى السجل التالي.
مثال:

```
<?
//Connect to server as azzozhsn no password
$link = mysql_pconnect("localhost","azzozhsn","");
//Select th magazine database
mysql_select_db("magazine",$link);
//Get name and id from magazine
$Query = 'SELECT name, id FROM magazine';
$result = mysql_query($Query, $link);
//Get every row
while($row=mysql_fetch_array($result, MYSQL_ASSOC)){
 //Print mane and id
    print({$row["id"]}={$row["name"]});
}
?>
```

21- الدالة mysql_fetch_object :

```
<?
object mysql_fetch_object(integer result)
?>
```

هذه الدالة تشبه الدالة mysql_fetch_array إلا أنها تعيد كائن. عند استدعاء الدالة ينتقل المؤشر إلى السجل التالي في الجدول، وإذا ةصل إلى نهاية الجدول ثم استدعيت الدالة مرة أخرى فإنها تعيد القيمة FALSE مثال:

```
<?
while($row=mysql_fetch_object(result)){
 //print id and name
    print ("$row->id, $row->name")
}
?>
```

22- الدالة mysql_fetch_row :

هذه الدالة تعيد مصفوفة تحتوي على قيم حقول سجل من الجدول وكل استدعاء يعيد قيمة الحقول في السجل التالي في الواقع هذه الدالة تشبه الدالتين السابقتين. مثال:

```
<?
while($row=mysql_fetch_row(result)){
 //print id and name
    print ("$row[0], $row[1]")
}
?>
```

23- الدالة mysql_change_user :

```
<?
musql_change_user(string user, string password, string db, integer link);
?>
```

استخدم هذه الدالة لتغيير مستخدم قاعدة بيانات المتصل بها.
المعطيان db، link اختيارية وفي حالة فقدهما يستعاض عنهما بالاتصال الحالي.
هذه الدالة تتطلب إصدار MySQL 3.23.3 أو ما بعدها.

التعامل مع الملفات والمجلدات

كل مبرمج يجب أن يتعامل مع الملفات والمجلدات في بعض النقاط ، برنامجك سوف يستخدم الملفات لكي يقوم بتخزين معلومات الإعداد للسكربت ، أو يقوم بتخزين البيانات لقراءتها وكتابتها ، أو لكي يقوم بحفظ البيانات المؤقتة ، وكمثال فإن أتفه برنامج عداد يحتاج إلي ملف يقوم بتخزين آخر رقم تم الوصول إليه ..

الملف : ليس عبارة عن أكثر من بايتات متسلسلة يتم تخزينها على القرص الصلب أو أي ماده تخزينية أخرى .
والمجلد : هو عبارة عن نوع محدد من الملفات يحتفظ بأسماء ملفات أخري ومجلدات أخري (تسمي بالمجلدات الفرعية) ، كل ما تحتاجه لتتعامل مع الملفات والمجلدات هو كيف يمكنك ربط سكربتك بهم ..

هذا الدرس سيأخذك إلي جولة لتعلم التعامل مع الملفات والمجلدات وفي نفس الوقت يوفر لك مرجعية لبعض الدوال التي تساعدك في ذلك مما يجعل مهمتك أسهل ...

سيقوم هذا الدرس بتغطيه المواضيع التالية :

1 فتح وإغلاق الملف .
2 القراءة من الملف والكتابة إليه .
3 مسح وإعادة تسمية الملفات
4 استعراض وتجول في الملف .
5 فتح وإغلاق المجلدات .
6 نسخ وإعادة تسميه المجلدات .

ملاحظة :
قبل أن نبدأ دعنا ننبهك أن التعامل مع الملفات يختلف من نظام تشغيل إلي آخر ففي أنظمة اليونكس تستخدم المسارات العلامة المائلة للأمام
مثال
/home/usr/bin/data.txt
بينما في الويندوز فإن المسار يكون كالتالى
C:\usr\bin\perl
وإذا استخدمنا العلامة الأمامية في PHP للويندوز فإنه يقوم بتحويلها بشكل تلقائي إلي علامة خلفية
بينما إذا أردنا استخدام العلامة الأمامية فإننا يجب أن نقوم بتكرار العلامة لكي يتم التعرف عليها
PHP\C:\\windows\

التعامل مع الملفات

يوفر الـPHP نوعين من الدوال المتعلقة بالملفات فهناك نوع من الدوال يستخدم مقبض للملف (file handle) أو ما يسمونه بالمؤشر (pointer) في العادة ، بينما بعض الدوال يستخدم قيمه حرفيه تشير إلي موضع الملف مباشره ...

مقبض الملف ليس أكثر من عدد صحيح (integer) يقوم بتعريف الملف المراد فتحه حتي يتم إغلاقه ، إذا كان هناك أكثر من ملف مفتوح فان لكل ملف مقبضه التعريفي الخاص به ، وبالطبع فإنه لا يتوجب عليك معرفه هذا الرقم ....

على سبيل المثال فإن الدالة fwrite() تقوم بفتح الملف لكتابة بيانات إليه وهي تستخدم مقبض لكي تقوم بالتعرف إلي الملف وفتحه ..

```
Fwrite ($fp,'Hello World');
```

بينما الدالة file() التي تستخدم للقراءة من الملف تقوم باستخدام قيمة نصية تقوم بالإشارة إلى مكان الملف بشكل مباشر لكي يتم التعامل معه ..

لا تصب بالرعب والخوف من هذا الكلام فأنا أعلم أنه قد يكون غامضاً عليك .. تنفس الصعداء وجهز لنفسك كأساً من الشاي لأننا سنبدأ في الجد الآن ....

ملاحظة : ستجد أن اغلب الدوال أو معظمها أو كلها تقريبا تقوم بإرجاع القيمة True إذا تمت بنجاح والقيمة False إذا فشلت في الحصول على هدفها ..
لنبدأ الآن مع سكربتات مبسطة للعمل مع الملفات ..

فتح واغلاق الملفات

Fopen

تستخدم هذه الدالة ثلاث عوامل هي مسار الملف (path) والوضع له (للقراءة ، للكتابة .....) بالاضافة إلى مسار الـInclude فيه وتقوم هذه الدالة بإرجاع مقبض للملف ...

قد تواجهنا مشاكل أحيانا فقد يكون الملف غير منشأ أو أننا لا نملك صلاحيات عليه ولذلك فإنه يمكننا اختبار القيمة التي ترجعها هذه الدالة فإذا كانت القيمة صفر فهذا معناه أن الدالة فشلت في إرجاع مقبض الملف أو نوعه ، أما إذا كانت القيمة هي واحد فهذا معناه أن الدالة قد نجحت في فتح الملف

مثال

```
$fp=fopen ("./data.txt", "r");
if (!$fp) die ("فشل في قراءه الملف تأكد من التراخيص ومن مسار الملف");
```

يمكننا كتابة المثال أيضا بالشكل التالي :

```
If  (!($fp=fopen ("./data.txt", "r"))) die ("لا يمكن القراءة من الملف");
```

لاحظ أننا قلنا سابقاً أن هناك دوال تستخدم للتعامل مع الملفات تستخدم مقبض وهذا المقبض هو عبارة عن رقم ، في مثالنا هذا يتحدد رقم المقبض الذي هو المتغير $fp الذي يخزن فيه مكان الملف وما إذا كان قابلاً للفتح أو لا أو يعمل أو لا يعمل ، والنتيجة التي تتخزن في المتغير $fp هي رقم مثلما قلنا سابقاً وهو صفر إذا كان الملف لا يعمل أو واحد إذا تم فتح الملف بنجاح ..

الآن دعنا نناقش معاملات الدالة fopen الذي تقوم بإعطائنا رقم المقبض ..

أول معامل هو مسار الملف على القرص الصلب

لنفرض أن لديك مجلدا قمت بإنشائه في مجلد السكربتات الرئيسي لديك الذي يسمى htdocs وأسميته data

ولنفرض أن سكربتك يستخدم ملفين :
1- ملف للقراءة والكتابة يسمي data.txt .
2- وملف يقوم بعرض المدخلات والإضافة إليها اسمه script.txt .

حسنا لدينا الآن ثلاث حالات للسكربت

الحالة الأولى :
أن يكون الملفين في نفس المجلد (data) وعند ذلك يمكنك فتح الملف الذي تريد فتحه بذكر اسمه فقط من غير إضافات

```
$fp=fopen ("data.txt", "r");
```

الحالة الثانية:
أن يكون هناك مجلد في نفس مجلد الـdata باسم آخر ولنقل أن هذا الاسم هو gb وفيه ملف data.txt على ذلك فإننا نكتب المسار المطلق لهذا المجلد كالتالي :

```
$fp=fopen ("./gb/data.txt", "r");
```

الحالة الثالثة :
أن يكون الملف الذي تريد قراءته موجود في المجلد htdocs بينما السكربت موجود في المجلد data الموجود داخل htdocs على ذلك نكتب المسار النسبي كالتالي

```
$fp=fopen ("../data.txt", "r");
```

لاحظ النقطه التي تسبق العلامة الأمامية جيدا..
أتمني أن تكون فهمت من هذا الكلام ما هو المقصود بالمسار المطلق والمسار النسبي ..
يمكننا أيضا وضع رابط صفحة في موقع آخر ولكننا لن نستطيع الكتابة عليه بل قراءته فقط

مثال :

```
If  (!($fp=fopen ("http://www.swalif.net/softs/index.php", "r"))) die (" لا يمكن القراءة من الملف");
```

ينقصنا نقطه يجب أن نتكلم عنها وهي عند تحديد العامل use_include_path

العامل الثاني الذي نستخدمه للملفات هو حاله الملف

(للقراءة ، للكتابة ، للإضافة إليه) يحدد وضعية الملف حال فتحه إذا كان للقراءة فقط أو للكتابة فقط أو للاثنين معاً أو للإضافة ، وأرتبها هنا في جدول بسيط ..

القيمة	الوصف
r	تفتح الملف للقراءة فقط ويكون المؤشر في بداية الملف
r+	يفتح الملف للقراءة والكتابة ويضع المؤشر في بداية الملف
w	يفتح الملف للقراءة فقط ، أي بيانات موجودة سيتم مسحها ، إذا لم يكن الملف موجودا سيحاول PHP إنشاؤه
w+	يفتح الملف للقراءة والكتابة ، أي بيانات موجودة سيتم مسحها ، إذا لم يكن الملف موجودا سيحاول PHP إنشاؤه
a	يفتح الملف للإضافة فقط ، إذا لم يكن الملف موجودا سيحاول PHP إنشاؤه ، سيكون المؤشر في نهاية الملف
a+	يفتح الملف للقراءة و للإضافة ، إذا لم يكن الملف موجودا سيحاول PHP إنشاؤه ، سيكون المؤشر في نهاية الملف
b	يستخدم لفتح وقراءه ملفات الصور على نظام أو سيرفرات الويندوز فقط .. أما الينوكس فالعوامل السابقه تتعامل مع ملفات الصور بشكل عادي ..

هناك مؤشر للملفات يحدد إذا ما كنت ستكتب من نهاية أو بداية الملف أو حتى من وسطه أو من أي مكان بالملف ، ستعرف كيفية التحكم بهذا المؤشر بعد قليل .

العامل الثالث هو تحديد use_include_path

فإذا قمت بتحديد قيمته إلي (1) وقمت بكتابة اسم الملف مباشرة فسيبحث الـPHP عن الملف في نفس المجلد الموجود به السكربت ثم سيقوم بالبحث عن الملف في المجلدات التي تم تحديدها في المتغير use_include_path في ملف php.ini

```
;;;;;;;;;;;;;;;;;;;;;;;
; Paths and Directories ;
;;;;;;;;;;;;;;;;;;;;;;;

; UNIX: "/path1:/path2"
;include_path=".:/php/includes"
;
; Windows: "\path1;\path2"
;include_path=".;c:\php\includes"
```

مثال :

```
$fp=fopen (“./data.txt”, “r”,1);
```

fclose

عندما تنتهي من التعامل مع الملف ، تحتاج إلى إغلاقه لكي يتم حفظ التعديلات عليه ، إذا تم إحباط سكربتك لأي سبب أو أن السكربت انتهي عمله فإن الـPHP يقوم بإغلاق جميع الملفات تلقائيا
تقوم الدالة ()fclose بإغلاق الملف عندما تريد إغلاقه وهي تحتاج إلي معامل واحد فقط وهو مقبض الملف الذي تريد إغلاقه
مثال :

```
Fclose ($fp) ;
```

قراءه وكتابه الملفات

لقد تعرفنا الآن كيفية فتح وإغلاق الملف ، لنقم الآن بالتعرف علي كيفية قراءة و كتابة البيانات من الملف ،

Fread

تقوم هذه الدالة بقراءة واستخراج البيانات الموجودة في الملفات ووضعها بمتغير وهي تأخذ معاملين المعامل الأول هو مقبض الملف والعدد الثاني هو عدد الحروف المراد قراءتها ..
مثال :

```
$fp=fopen("data.txt","r");
$data=fread($fp,10);
```

وخذ باعتبارك نقطتين وهما :

1- إذا مثلاً قرأت عشر حروف من الملف وكان في الملف عشرين حرف وقمت بطلب الدالة fread مره أخري فسيتم قراءة العشر أحرف الثانية ..
2- إذا كان في الملف أقل من عشر أحرف فسيتم قراءة الموجود .

Fwrite

تقوم هذه الدالة بالكتابة إلي الملف وتحتاج إلى عاملين وهي مقبض الملف والقيمة المراد كتابتها إلى الملف ، فعلى افتراض أنك قد فتحت الملف والمقبض هو $fp فإننا نكتب الكلمة PHP إلى الملف بالطريقة التالية :

```
Fwrite ($fp, "PHP");
```

وهناك معامل ثالث لهذه الدالة يحدد كم حرفا سنقوم بكتابته من القيمة الحرفية الموجودة في المعامل الثاني فلو مثلا كتبنا

```
Fwrite ($fp, "PHP",1);
```

فسوف يتم كتابه أول حرف فقط ...

قراءة وكتابة الحروف في الملفات

Fgetc

تستخدم هذه الدالة لقراءة حرف واحد من الملف في كل مرة ، وهي تستخدم معاملاً واحداً وهو مقبض الملف
وتقوم بإرجاع حرف واحد من الملف أو (False) عند الوصول إلى نهاية الملف ..

Feof

تقوم هذه الدالة بخدمتنا في هدف بسيط وشي ممتاز وهي معرفة إذا ما كنا قد وصلنا إلى نهاية الملف عند قراءته وتقوم بإرجاع (true) عند الوصول إلى نهاية الملف أو حصول خطأ ما ، وهي تأخذ معاملاً واحد وهو مقبض الملف .
فقد تكون مثلاً تريد أن تتأكد أن المؤشر لم يصل إلي نهاية الملف بعد استخدامك لأحد الدوال التي تقوم بنقل المؤشر من مكان إلي آخر ، عند ذلك ستكون هذه الدالة مفيدة لتخبرك إذا ما وصلت إلى نهاية الملف أو لا ...

تطبيق عملي :

قم بإنشاء ملف سمه file.txt واكتب فيه أكثر من سطر ثم قم بإنشاء ملف PHP وسمه بأي اسم وضع فيه الشفرة التالية ثم اختبره ، لكي ترى عمل الدالتين

```
 <?
$fp= fopen("file.txt","r");
While (!feof($fp))
{
 $char=fgetc($fp);
    echo $char;
} ?>
```

Fgets

إذا استخدمنا الدالة fgetc لقراءة الملفات الطويلة فإنها ستأخذ وقتاً وعمراً حتى يتم قراءتها ، يقوم الـPHP بتوفير دالة fgets لتساعدنا في قراءة عدد محدد من البايتات وهي تأخذ معاملين ، المعامل الأول هو مقبض الملف والمعامل الثاني هو عدد الحروف المراد قراءتها +1 ، فإذا مثلا أردت قراءة ملف يتكون من خمس حروف فسيكون المعامل الثاني للدالة هو الرقم 6 وتتوقف الدالة عند حدوث أحد من ثلاث حالات
الأول : هو إذا تم قراءة عدد البايتات المحدد ة
الثاني : إذا تم الوصول إلى نهاية سطر في الملف .

الثالث : إذا وصلت إلى نهاية الملف .
مثال :

```
$fd = fopen ("/tmp/inputfile.txt", "r");
while (!feof ($fd)) {
   $buffer = fgets($fd, 4096);
   echo $buffer;
}
fclose ($fd);
```

Fputs

تقوم بنفس وظيفة الداله fwrite وتأخذ نفس معاملاتها ونفس طريقتها ..

القراءة داخل الملفات

File

تحتاج هذه الدالة إلى معامل واحد هو مسار الملف ولا تحتاج إلي مقبض ، وعملها هو قراءة ما بداخل الملف وتخزينه سطراً سطراً في مصفوفة حيث أن هذه المصفوفة تقوم بأخذ كل سطر في الملف كأنه عنصر لوحده وتظل السطور سطورا (أي أن المصفوفة تحتفظ بالمعامل للسطر الجديد (\n) بداخلها) ، هذه الدالة لا تحتاج إلي مقبض للملف بل تحتاج إلى مسار الملف فقط ، وهي تقوم بفتح وقراءة وإغلاق الملف تلقائيا ...
وكغيرها من الدوال فإنها تستطيع قراءة صفحات الإنترنت الخارجية ..
مع ذلك يستحسن أن لا تقوم باستعمال هذه الدالة لقراءة الملفات الطويلة لأنها تقوم باستخدام قدر كبير من الذاكرة المحجوزة للـ PHP وقد تستخدمها كلها ...
مثال :

```
<?
$fcontents = file ('file.txt');
while (list ($line_num, $line) = each ($fcontents)) {
   echo "<b>Line $line_num:</b>  $line  <br>\n";
}
?>
```

Fpassthru

تقوم هذه الدالة بقراءة محتويات الملف بداية من النقطة التي توقف منها المؤشر الوهمي عند أي عملية قراءه أخرى ، وتقوم بالتوقف عند نهاية الملف وتقوم بإغلاق الملف من تلقاء نفسها لذلك لا داعي لإغلاق الملف بواسطة الدالة fclose بعد استخدامك لهذه الدالة ، وتقوم الدالة بقراءة المحتويات وطباعتها بشكل قياسي ، وهي تحتاج إلى معامل واحد فقط وهو مقبض الملف ...
مثال :

```
<?
$fp=fopen("file.txt","r");
fpassthru($fp)
?>
```

Readfile

تقوم هذه الدالة بقراءة جميع محتويات الملف ولا تحتاج إلي مقبض بل إلى مسار الملف فقط وتقوم بقراءة كامل محتويات الملف ثم طباعتها بشكل قياسي وتقوم بإرجاع عدد البايتات التي تم قراءتها أو (false) عند حدوث خطأ ما

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

الوصول العشوائي إلى الملفات

أخبرناكم سابقا بأن هناك طريقة تجعلك تتحكم في التحكم بالمؤشر الوهمي للملف والوصول إلي أي مكان في الملف أو عند أي حرف تريده ، بالدوال السابقة كنا عندما نصل إلي حرف معين مثلاً بدالة من الدوال فإننا نقوم بإغلاق الملف ثم نعاود فتحه كي نكمل القراءة من عند الحرف الذي تم الوصول إليه ولكن هذه الطريقة غير عمليه نهائيا .....
يوصل لنا الـPHP بعض الدوال التي تمكننا من الوصول إلي الملف بالمكان الذي نريده ومن هذه الدوال :

Fseek

تحتاج هذه الدالة إلى عاملين ، العامل الأول هو مقبض الملف $fp والعامل الثاني هو عبارة عن رقم صحيح يسمونه كمصطلح بالـ(offset) أي المكان الذي سيتوقف فيه المؤشر ، سيقوم الـPHP بالتحرك في الملف إلي أن يصل إلى المكان الذي تم تحديده .. أي أنه إذا كان في الملف سطر واحد مكون من عشره حروف وقمنا بجعل الـoffset خمسة ، سيقوم الـPHP بالتحرك حتى يصل إلى نهاية الحرف الخامس ...
وهناك معامل ثالث اختياري لهذه الدالة ويسمونه كمصطلح بالـ (whence) وله إحدى ثلاث خيارات :

Seek_set ويقوم بقراءة الملف من بدايته حتى يصل إلى المكان المطلوب بالـoffset
Seek_cur يقوم بالقراءة من المكان الحالي حتى يصل إلي المكان المطلوب بالـoffset
Seek_End يقوم بالقراءة من نهاية الملف حتى يصل إلي المكان المحدد بالـoffset

تعتبر هذه الدالة نادرة في عملها (أو كما يسميها المبرمجون شاذة) بسبب أنها تقوم بإرجاع القيمة (0) عند نجاحها والقيمة (-1) عند حصول خطأ ما ..

مثال :
قم بفتح ملف واكتب فيه ثمان حروف متتالية ثم قم بحفظه باسم file.txt ثم قم بوضعه مع ملف PHP فيه الشفرة التالية ، ثم بعد ذلك شغل ملف الـPHP وانتظر النتيجة :

```
 <?
$fp = fopen("file.txt");
fseek($fp,4,SEEK_SET);
fpassthru($fp);
?>
```

Ftell

هذه الدالة من الدوال المفيدة فهي تقوم بإرجاع مكان الـoffset (أو المؤشر الوهمي) في الملف وتحتاج إلي معامل واحد وهو مقبض الملف ...

```
<?
$fp = fopen ("file.txt");
$p = ftell($fp);
echo $p;
?>
```

Rewind

تقوم بإرجاع المؤشر إلي بداية الملف ...

```
<?
$fp = fopen ("file.txt");
rewind($fp)
?>
```

جلب معلومات الملف

يوفر الـPHP دوال تساعدنا في معرفه حجم الملف وما إذا كان الملف موجوداً أم لا من هذه الدوال :

File_exists
تقوم هذه الدالة بالقيام بالتأكد ما إذا كان الملف موجوداً أم لا وهي تحتاج على معامل واحد وهو مسار الملف ، وتقوم بإرجاع true (1) إذا كان الملف موجوداً و false إذا كان الملف غير موجود

```
<?
$Th=File_exists("file.txt");
echo $Th ;
?>
```

Filesize
تقوم هذه الدالة بإرجاع حجم الملف بالبايتات أو false عند حصول خطأ ...

دوال الملفات المتعلقة بالوقت :

هذه الدوال تقوم بإرجاع معلومات مفيدة عن وقت التغيير الذي طرا على الملف أو آخر مره تم قراءته وهي على حسب نظام التشغيل فإذا كان نظام السيرفير هو يونكس أو لينوكس ستقوم الدوال بإرجاع الوقت بنظام (timestamp) وهو الوقت مترجم إلي عدد الثواني منذ صدور يونكس ومولده على العالم ، بينما تقوم بإرجاع وقت التعديل على نظام الويندوز مباشره ...

يقوم الـPHP بتزويدنا بدالتين لمعرفه الوقت :
Filectime وتقوم بإرجاع آخر وقت تم فيه التغيير على الملف على شكل timestamp ويشمل هذا آخر وقت تم فيه إنشاء الملف أو الكتابة إليه أو تغيير تراخيصه ...
Filemtime تقوم بإرجاع آخر وقت تم فيه التعديل على الملف على شكل timestamp ويشمل هذا إنشاء الملف أو تغيير محتوياته

تقوم الدالة getdate بعمل مفيد وهو تحويل الوقت من timestamp إلي الوقت العادي

الملكية والتراخيص

على أنظمة تشغيل اليونكس مثل يونكس ترتبط الملفات مع مستخدم خاص أو مجموعة من المستخدمين (group) وتحتوي على علامات وتراخيص تقوم بتوضيح من له صلاحية على استخدامها ..
يمكننا أن نخلص التراخيص كالتالي :
1 / ممتلك الملف (owner) ، بشكل افتراضي ، وهو المستخدم الذي تم استخدام حسابه في استخدام الملف .

2 / مجموعه من المستخدمين (group) ، بشكل افتراضي ، المجموعة التي يكون ضمنها مالك الملف

3 / جميع المستخدمين (all) كل شخص له حساب على النظام .

المستخدمين والمجموعات في أنظمة اليونكس يتم تعرفهم عن طريق رقم (ID) مثلما يتم تعريفهم عبر أسمائهم ، إذا كنت تريد معرفه معلومات شخص عن طريق رقمه ، يمكنك استخدام هذه الدالة :

Posix_getpwind

التي ستقوم بإعطائنا مصفوفة تحتوي على المعلومات التالية

Name	اسم المستخدم الذي يدخل به في حسابه
passwd	كلمة السر المشفرة للمستخدم
uid	رقم الحساب للمستخدم
gid	رقم حساب المجموعة التي فيها المستخدم
gecos	اسم المستخدم الكامل ، رقم تلفونه ومعلومات إضافية
dir	المجلد الرئيسي للمستخدم
shell	المسار الرئيسي لحساب المستخدم

Posix_getgrgid

تقوم هذه الدالة بإرجاع مصفوفة عن معلومات المجموعة ، وهي تحتاج إلى معامل واحد فقط وهو رقم الـID للمجموعة ...

وسوف تحتوي على العناصر التالية :

Name	اسم المجموعة
Gid	رقم المجموعة
members	عدد أعضاء المجموعة

وهناك ايضا خمس دوال تساعدنا في معرفة معلومات أكثر عن الملفات وتحتاج فقط إلي مسار الملفات

Fileowner

تقوم بإرجاع رقم المعرف (ID) لمالك الملف ...

Filegroup

تقوم بإرجاع رقم المعرف (ID) لرقم المجموعة التي يعتبر مالك الملف ضمنهم ..

Filetype

تقوم بإرجاع رقم نوع الملف وقد تعود بإحدى هذه القيم (file ، dir ، char ، fifo، link ، block) والذي يهمنا منهم هو file وdir ...

Is_dir

وتقوم بإرجاع True إذا كانت قيمه المسار هو مجلد ..

Is_file

وتقوم بإرجاع True إذا كانت قيمه المسار هو ملف ..

الحصول على اسم الملف من وسط مسار الملف ..

Basename()

مثال :

```
<?
$path = "/home/httpd/html/index.php3";
$file = basename ($path);
echo '$file <br>';
$file = basename ($path,".php3");
echo '$file <br>';
?>
```

هذه الدالة مفيدة جداً للحصول على الملف من وسط مسار مجلد ..

نسخ ، اعادة تسمية وحذف الملفات

تسمح لك الـPHP بنسخ ، وإعادة تسمية ، وحذف والدوال التي تستخدم لتنفيذ هذه العمليات هي

Copy ()

تقوم بأخذ قيمتين حرفتين وتشير إلي مصدر الملف الرئيسي الذي يوجد فيه الملف والمصدر الهدف الذي سيتم نسخ الـPHP إليه ...

```
<?
if (!copy($file, $file.'.bak')) {
    print ("failed to copy $file...<br>\n");
}
?>
```

Rename

نستطيع الآن استخدام هذه الدالة لإعادة تسمية الملف وهي تحتاج إلي قيمتين حرفتين وهي المصدر الملف أو مكانه واسمه الرئيسي ثم الاسم الجديد الذي تريد إعادة التسمية به ..

مثال :

```
<?
Rename ('file.txt','newfile.txt');
?>
```

Unlink()

تحتاج إلي قيمه حرفيه واحده وهي مسار الملف الذي تريد حذفه

```
<?
unlink ('file.txt');
?>
```

العمل مع المجلدات

مثلما تعاملنا مع الملفات في الـPHP فإننا نتعامل مع المجلدات ، فهناك دوال للمجلدات تتطلب مقبض المجلد ، وهناك دوال تحتاج فقط إلي القيمة الحرفية فقط وبدلاً من الإطالة دعنا نقوم بالدخول في الموضوع مباشرة

Opendir
تقوم بفتح المجلد وإعطائنا مقبض المجلد

Closedir()
تقوم بإغلاق المجلد المفتوح وتحتاج فقط إلى مقبض المجلد ...

Readdir
تقوم بقراءة المدخل الحالي للمجلد ...

Rewindir
تقوم بإرجاع المدخل من الصفر ..

Chdir
للانتقال إلي مجلد آخر ، وتتطلب المسار للمجلد الذي تريد الانتقال إليه ..

Rmdir
تقوم بمسح مجلد ، ولكن يجب أن يكون المجلد خاليا من أي ملفات أو مجلدات ، وتتطلب مسار المجلد الذي تريد مسحه ..

Mkdir
تقوم بإنشاء مجلد جديد وتتطلب أن يكون هذا المجلد غير موجود مسبقاً وتحتاج إلي قيمتين وهما اسم المجلد الجديد مع مساره ، والترخيص المطلوب له ..

Dirname
تقوم بإعطائنا اسم المجلد الحالي الذي فيه الملف ، وتحتاج إلي مسار الملف ..

تطبيق عملي :
أنشئ مجلد اسمه tmp في مجلد الـ htdocs وضع فيه ملفات ، ثم أنشئ ملف اسمه test.php في مجلد الـhtdocs واكتب الشفرة التالية ثم شغله :

```
<?php
if ($dir = @opendir("/tmp")) {
  while($file = readdir($dir)) {
    echo "$file\n";
  }
  closedir($dir);
}

?>
```

Dir()
عبارة عن كائن يحتوي على ثلاث وظائف .. ونقوم بإعطائه مسار المجلد الذي نريـده أن يتعامـل معـه ثم بعد ذلك نقوم بوضع قيمته في متغير يقوم بوراثة صفاته

خصائص الكائن :

handle
تقوم بإعطائك مقبض المجلد ..

Path
تقوم بإعطائك المسار للمجلد ..

Read
تقوم بإعطائنا المجلدات اعتماداً على المؤشر الحالي للمجلد ..

Rewind
تقوم بإرجاع مؤشر المجلد من الصفر ..تقريباً نفس عمليه rewinddir ..

Closedir
تقوم بإغلاق المجلد ..

بهذا يكون انتهي الدرس
قد تكون بعض النقاط غير واضحة ، في الواقع لن تحتاج إلي كل هذه الأمور في تعاملك مع الملفات

::

تم الكتاب بحمد الله

::

22 ذو القعدة / 1428 هـ

تصميم

---

aLTar3q
