

تم تحميل الملف من موقع  
البوصلة التقنية  
[www.boosla.com](http://www.boosla.com)

# لغة ( PHP )

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

## السهولة

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

تمتلك لغة PHP بنية وقواعد ثابتة وواضحة جدًا، 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 الإصدار 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 تتركب 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 استدعاه 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 للاستخدام 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، وهناك أيضا مجموعة من الدوال لمعالجة ملفات XML، ودوال أخرى لإرسال واستقبال الملفات عن بعد باستخدام بروتوكول FTP، وهناك 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 الحقيقية 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 لأيّة تغييرات عند نقله إلى مزود Apache، بالطبع تظل بعض الأمور البسيطة جدا التي يوفرها بعض المزودات دون غيرها، ولكن جميع البرامج التي كتبها منذ أن بدأت تعلمي للغة إلى الآن تعمل على جميع المزودات دون الحاجة لأيّ تغييرات، إضافة إلى ذلك فإن التغييرات التي حدثت باللغة الأساسية من الإصدار 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 تتركب على المترجم لزيادة ميزاته والوظائف المبيتة 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 Lerdorf 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 قسمين: 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.

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 إغلاق 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 تلويها 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 استخداما، 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، وتسهيل 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 يتيح 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 البدء 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 سكريبتاً 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 عبارته مكتوب فيها

التحية لدي أهل الإسلام هي السلام عليكم ورحمة الله وبركاته

شي بسيط أليس كذلك ؟

يتكون كود الـ php من نصوص و كود و علامات ولغة html وقد لا تحتوي على نصوص html .  
لكي يعمل الكود يجب أن يكون إمتداد الملف php أو بأي إمتداد من إمتدادات الـ php  
مثلاً php3 و phtml

--  
عندما تطلب صفحة في الإنترنت فإنك تجري اتصالاً مباشراً مع السيرفر هذه العملية تدعى request  
للسيرفر (يعني طلبية للسيرفر) يقوم السيرفر بتفسير طلبك والبحث عن الصفحة المطلوبة ويرسل اليك  
الصفحة المطلوبة كجزء مما يسمى response (استجابة) لمستعرض الانترنت لديك يقوم بعدها المتصفح  
لديك بأخذ الكود الذي 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 :

لأنريد 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 إعادته 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 يدعى 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 عنوانتها والبروتوكول الذي يقوم بعنوانة البيانات يدعى HTTP  
يقوم هذا البروتوكول بوضع عنوان للبيانات لكي يعرف البروتوكول TCP أين سينقل البيانات (فهو لا يستطيع  
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 بـ http:// فإنك تعلم مباشرة أن الانترنت يستخدم  
البروتوكول HTTP لإحضار هذه الصفحة يمكنك أن تأخذ صورة بأن الـ TCP عبارة عن ساعي بريد الذي يقوم  
بإيصال رسالة ، هذه الرسالة فيها طابع بريد وعنوان وهو مانسميه بالـ HTTP .

يتم تمرير الطلب من المستعرض إلى ملقم أو سيرفر الويب وهو ما يعرف بـ HTTP request ويقوم السيرفر برؤية مستودع البيانات لديه لكي يحصل على البيانات المطلوبة فإذا وجد الصفحة في المستودع قام بإرسالها على شكل حزم إلى الجهة التي قامت بالطلب باستخدام بروتوكول TCP ويعنون هذه الحزم لمستعرض الانترنت لديك باستخدام بروتوكول http (ننبه دائماً إلى 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 فإنه يقوم بإرسال صفحة تحتوي على رسالة خطأ 404 وهذه الصفحة التي أرسلت من ملقم الويب إلى المستعرض لديك تسمى HTTP response .

## بروتوكول ال 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 ال URL وهي ترسل كجزء من http request . نفس الموضوع مع ال http response هناك أمور أخرى تصل معه كجزء منه .

الكثير من هذه المعلومات تولد تلقائياً في رسالة ال HTTP ولا يقوم المستخدم بالتعامل معها مباشرة , إذن لا يحتاج أن تقلق نفسك بشأن هذه المعلومات إذا أنت لم تنشأها في الأصل ويجب أن تأخذ أيضاً في معلوماتك أن هذه المعلومات ترسل كجزء من ال HTTP request وال HTTP response لأن سكربت ال PHP الذي نصنعه يمنحنا تحكماً إضافياً بهذه المعلومات .

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

Request/response line - 1

Http header - 2

Http body - 3

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 ال request على الأقل ال request line (سطر الطلب) وال HOST . يرسل مستعرض الانترنت طلبية (HTTP request) إلى ملقم الويب تحتوي على التالي :

### The Request Line - 1

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

أ - أمر HTTP وهو ما يعني بـ method .

ب - المسار من السيرفر إلى المصادر المطلوبة (صفحات الانترنت) المطلوبة من قبل العميل (المستعرض)

ج - إصدار ال HTTP .

إذن كمثال على ال Request Line أنظر إلى السطر التالي :

GET /testpage.htm HTTP/1.1

الـ method يخبر السيرفر كيف يتعامل مع الطلب هناك ثلاثة أنواع شائعة من ال method

## HTTP Header -2

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 الـ HTTP Header يحتوي على معلومات نستطيع تقسيمها الى ثلاث فئات وهي :

- أ - عامة GENERAL : تحتوي معلومات إما عن العميل أو السيرفر ولا تخص إلى فرد أو مجموعة .
- ب - شخصية Entity : تحتوي على معلومات عن البيانات التي أرسلت بين المتصفح والسيرفر .
- ج - 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 :Iexplore (win98

مثلا 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 اختياري , يقوم الـ HTTP بالإخبار عن إنتهاء معلومات الـ header بترك سطر فارغ (وهذا يكون في الـ HTTP1.1) .

## 3 - The HTTP Body :

إذا تم استخدام الأمر POST في الـ HTTP Request Line عندها يقوم الـ HTTP بطلب المعلومات التي أرسلت في الـ body 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 إصدار الـ HTTP .

2 - شفره أو كود الـ http request التي تقوم بتحديد إذا كان الـ request ناجحاً أم فاشل .

مثال :

HTTP/1.1 200 OK

في هذا المثال يقوم الـ response line بإرجاع القيمة 200 متبوعة بالكلمة OK هذه تشكل وتشير إلى نجاح الـ request ويكون الـ response 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 الـ request hader الذي ناقشناه في الأعلى . وتنقسم المعلومات التي فيه أيضا إلى ثلاثة أنواع :

أ - عامة GENERAL : معلومات عن الـ client أو السيرفر ولا تخص إلى واحد منهما .

ب - شخصية Entity : يحتوي على معلومات عن البيانات التي يتم إرسالها بين السيرفر والعميل .

ج - الإجابة Response : يحتوي معلومات عن السيرفر الذي قام بإرسال الرد وكيفية تعامله ومعالجته للرد ( Response) .



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

مثال :

HTTP/1.1 200 OK -the status 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 الشيء المطلوب من السيرفر .

### 3 - Http Body

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

### أين سكربت الـ PHP من ذلك كله ؟

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

تكلّمنا عن أن سكربت الـ php يتكون من ثلاثة أشياء : نص وكود الـ php وكود الـ html ، لانستطيع وصف الـ html بأنها لغة برمجة بشكل جيد ونستطيع أن نقول أن الـ php لغة سكربتات Scripting Language لأنها تضيف قدرات الـ html عليها مثل الجداول والفريمات بكود الـ html بداخل كود الـ php هناك لغات تسمى لغات سكربتات قد تكون متآلفاً معها مثل الجافا سكربت والفجول بيسك سكربت بإستثناء أن الفرق بينها وبين الـ php هو أن الـ php لغة تعتمد على جهة المزود أي السيرفر ويمكنك تخصيص المتصفح الذي يستعرضها .  
تجعلنا الـ html نضمن سكربتات الـ php فيها ضمن قواعد لذلك لكي نستطيع تشغيلها ولكننا لانسي أن إمتداد الملفات يظل كما هو الـ php أو الـ php3 بدون تغير فيه لكي يتم إرسال السكربت الى مكتبة الترجمة (scripting engine) التي تقوم بترجمة السكربت إلى الـ html (كانك تترجم من عربي لإنجليزي أو العكس )

### مفهوم الـ parsing و الـ Execution :

ممكّن أن نقسم عملية الترجمة الذي يقوم بها سيرفر الـ php إلى قسمين أو عمليتين :  
**العملية الأولى :** هي أن السيرفر يقوم أولاً بفحص قواعد اللغة وهذا لا يضمن أن السكربت صحيح مائة بالمائة ولكنه تدقيق في الأوامر وقواعد اللغة وهذا مايسمونه بالـ Parsing  
**العملية الثانية :** هي تنفيذ السكربت بعدها وإخراجه على شكل كود الـ html وهذا مايسمى 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 ، إذن 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 كالتالي :

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

مثال آخر :

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

وأیضا 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 الـ PHP ، المتغيرات تبدأ بعلامة الـ \$ ولكي تقوم بإدخال قيمة في المتغير فإنك تستخدم المعامل (=) إذن لكي تقوم بإنشاء متغير يحتوي على قيمة يمكنك القيام بذلك كالتالي :

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

لاحظ أن السطر السابق يتكون من خمسة أشياء :

- 1 / المتغير وهو alfares
- 2 / وقبله علامة الـ \$ لكي يعرف مترجم الـ PHP 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 القيمة هو أنت (مبرمج الـ php) )

## ملاحظات :

1- اسماء المتغيرات حساسة لحالة الأحرف إذا كانت كبيرة وصغيرة

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

المتغيرين الذين بالأعلى مختلفين بسبب حالة الأحرف.

2 - يمكنك استخدام المعامل ( \_ )

\$First\_name

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

## علامات التنصيب

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

## هناك أنواع 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 لا يتم تعريف نوعها من قبل المبرمج إنما مترجم الـ PHP يقوم بالتعرف عليها لكي يتم إتمام العمليات المختلفه عليها .

## البيانات الحرفية /

في الـ PHP أي قيمة تكون بين علامتي تنصيب عادية أو علامة تنصيب مفردة يعتبرها الـ PHP قيمة حرفية  
أمثلة :

"هذا النص بين علامتي تنصيب عادية او مزدوجة"  
'هذا النص بين علامتي تنصيب مفردة او وحيدة'

يجب أن يبدأ النص وينتهي بنفس علامة التنصيب ، وإلا فلن تتعرف الـ PHP على القيمة الحرفية أو على النص .

```
<?
$d="غلط `
echo `خطا`
?>
```

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

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

وتصحيحه

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

وأيضاً مثال آخر

```
<?
$r = "This is"BAD"; // خطأ
$t = "This is `good"; // صحيح
?>
```

أما إذا كنت مصرّاً على ذلك أو تحتاج إليها في عمليات ضرورية (كما سوف نرى 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; // النتيجة c:windowssystem.ini

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

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

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

وأيضاً 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 حتى اعطاءها هذا الاسم يجعل الواحد يشعر بالاحباط والخوف )  
لاحظ أننا لا نستخدم علامات التنصيص وذلك ليعرف الـ PHP 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 الحسابية

هي مثل الجمع والطرح والضرب والقسمة وهي مرتبة كالتالي :  
أولاً /الأقواس  
ثانياً /الضرب ثم القسمة .  
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 أخرى لكنها لم تعمل عليك 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 لا تتغير ونعرفها عن طريق الدالة define  
الثوابت حساسة أيضا لحالة الأحرف

```
<?
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;
?>
```

## معرفة 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 بإرجاع القيمة (1) إذا كان المتغير تم انشاؤه ولا ترجع 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 دمة 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 ("" ) ولا تقوم بإرجاع أي شيء إذا كان المتغير منشأ وفيه قيم غير المذكورة .

## داوال الوقت التاريخ

نستطيع إيجاد الوقت و التاريخ عن طريق دوال في الـ PHP 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 الـ php بكثير من الدوال والكلمات المحجوزة التي تقوم بعمليات مختلفة مثل العمليات الحسابية المعقدة والقيام بإيجاد الوقت والتاريخ وإرسال الرسائل البريدية وإيقاف السكريبتات 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 أيضاً 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 لخدم 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 بياناتك ورقم بطاقة الائتمان وغير ذلك من المعلومات ويتم ذلك عن طريق نموذج (فورم) .

في الواقع أنت تقوم باختيار الكتاب الذي تريد وتكتب اسمك ورقم هاتفك وصندوق بريدك (ربما) في فراغات أو عن طريق الإشارة إلى مجموعة من الخيارات .

يتم تخزين هذه القيم في المتغيرات التي يتم كتابتها في الخاصية name (نتكلم عنها في هذا الدرس) ويتم إرسالها عند ضغط زر - إرسال البيانات - (submit) إلى (البرنامج) الصفحة التي سوف تقوم بمعالجة هذه البيانات (والتي يتم تحديدها في الخاصية ACTION) وإجراء العمليات عليها مثل تخزينها مثلاً في قاعدة البيانات أو إرسالها إلى البريد الإلكتروني وذلك عن طريق الـ php .

## ماذا يعمل العميل في النماذج ؟

إنه باختصار يقوم بتعبئة مربعات نصوص (textBox) ويقوم بوضع علامة صح في مربعات الاختيار (check boxes) أو يقوم بالتصويت أحياناً لشيء معين فيختر زر اختيار (ازرار الراديو) . هذه الأشياء كلها يتم انشاءها بواسطة الـ HTML ودرسنا لهذا اليوم يناقش كيفية انشاءها وكيفية التعامل والحصول على البيانات منها ، بقي علينا كبدية أن نعرف أن هذه الأدوات تنشأ في الواقع بين وسمين من وسوم لغة الـ HTML وهي الوسمين

<form>

</form>

## خصائص النماذج

يجمع النموذج جميع خصائص المضيف لكننا هنا سنتطرق إلى 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 نفضلها في دروس قادمة إن شاء الله .

## ACTION

وظيفة هذه الخاصية أن تخبر السيرفر مكان الصفحة التي يقوم بإرسال معلومات النموذج إليها أو عنوانها أي كان نوعها , وطبعاً في حالتنا ستكون الصفحة الثانية هي الصفحة التي تحتوي على سكربت الـ php . ليس مهماً أن تكون الصفحة php فقد تكون html ولكنها تحتوي على كود يختص بالتعامل مع برنامج تفاعلي لصفحات الويب مثل الجافا . ولانريد أن نخرج عن نطاق الموضوع فدعنا نعطي مثلاً على هذه الخاصية :

<FORM ACTION ="TEST.PHP">

.....

</FORM>

## METHOD

هذه الخاصية تقوم بإخبار النموذج طريقة إرسال المعلومات إلى الصفحة الهدف وفي الواقع هناك طريقتين مشهورتين ومعروفتين لإرسال المعلومات هما GET و POST .

<FORM ACTION ="test.php" METHOD = "GET">

أو

<FORM ACTIN = "test.php" METHOD ="POST">

ملاحظه /في الواقع يوجد اكثر من هذه الطريقتين لإرسال 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 كالتالي :

- 1- كتابته عنوان الصفحة المصدر .
- 2- اتباعها بعلامة استفهام .
- 3- كتابة العناوين والقيم .

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

قد تكون النقطتين الأخيرتين غير مفهومين 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 قيمة خاصة بها (value) . وهي مشابهة للمتغيرات ويمكن أن يحتوي عنوان الصفحة على أكثر من عنوان (name) وأكثر 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) نتيجة الاستعلام الحرفية. العنوان دائما يكون باللغة الانجليزية (name) ونعامله كانه اسم متغير من المفترض تعريفه في الصفحة الهدف (التي سنكتبها بال-PHP).

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

## URL Encoding

هناك بعض الأحرف لا يستطيع 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	شفرتة	الحرف	شفرتة	الحرف	شفرتة
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 بالعملية كلها بدلا عنك .

## POST

في الواقع وظيفتها هي نفس وظيفة الـ get ولكنها لاترسل المعلومات في عنوان صفحة الانترنت بل تقوم وضعها في الـ body التابع لـ http response . بالإضافة إلى أنه يستطيع ارسال البيانات بكمية أكبر من الـ GET .

## أيهما تستخدم GET أم POST ؟

قد يكون العيب في الخاصية [GET](#) عدم سرية المعلومات التي تقوم بكاتبها 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 .

أيضاً [POST](#) مفيدة في إخفاء المعلومات وإحتواء كميات كبيرة من البيانات ولكن لا يمكن الاحتفاظ بعنوان الصفحة .... مع ذلك فإنها أيضاً ليست جيدة في الحماية بحيث أن أي هacker خبير يمكنه الحصول على المعلومات إذا لم يكن لها تشفير معين في نقلها .. لكن إذا اردت فعلاً ان تجعلها محمية فيجب عليك استخدام اتصال محمي الى سيرفر محمي .... [او ماسمونه \( SCURE CONNECTION TO SCURE SERVER \)](#).

## أدوات التحكم في النماذج :

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

في أغلب هذه الأشياء 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 عملية

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

### مربعات النصوص (TEXT Box) :

نقوم بعمل ذلك كالتالي :

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="جبنه ومربي">
<br>
<INPUT TYPE= submit VALUE="إرسال">
<INPUT TYPE= reset VALUE="مسح">
</form>
</html>
```

3 - قم بحفظ الملف كصفحة HTML . وسميته (textbox.html) .

4 - افتح محرر النصوص إذا كنت أغلقته .

5 - اكتب الكود التالي :

```
<?
Echo $food ;
"وجبتك المفضله والى تموت في حبها هي"
?>
```

6 - قم بحفظ الملف كـ php . وسميته 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 و زر 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 بواسطة الوسم INPUT واخترنا الـ

TYPE="text"

كما قمنا بوضع القيمة الافتراضية فيه بواسطة القيمة

Value= "جبنه ومربي"

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

(لاحظ ان تسميه المتغيرات حساسه لحاله الاحرف في PHP واننا لم نقم بوضع \$ في صفحه المتغير في كود الhtml.)

وأيضاً لقد قمنا بإضافه زر بواسطة

TYPE=SUBMIT

وقمنا بوضع كلمة على الزر وهي كلمة (إرسال)

VALUE = "إرسال"

أيضاً قمنا بصنع زر آخر

Type =reset

وقمنا بجعل العبارة التي عليه (مسح)

Value="مسح"

هناك نوعين من الأزرار هي SUBMIT و RESET

1- submit يقوم بإرسال المعلومات .

2- reset يقوم بمسح البيانات في جميع الأدوات في النموذج لإعادته إدخالها من جديد .

بعد ما قمنا بإدخال البيانات وضغط زر الإرسال قام النموذج بإرسال البيانات إلى الصفحة المحددة في الخاصية ACTION وقامت الصفحة المحددة باستقبال النتائج الموجودة في النموذج وهي نتيجة واحدة في مربع نصوص تم حفظ قيمته في المتغير food . وقامت بطباعتها بواسطة الدالة echo .

نظراً لأننا استخدمنا الأسلوب GET فقد تم إعطاءنا عنوان الصفحة بالاضافه الى (?) وايضاً المعلومات المسجلة في المتغيرات والتي تم استخدام ال URL ENCODING فيها لأنها تستخدم حروف عربه .

### مربعات النصوص الكبيره (text area) طلبات اكبر للطعام الشهوي !

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 الأسطر .

تستخدم هذه الأداة وسم فتح ووسم إغلاق

<TEXTAREA>

</TEXTAREA>

ويمكنك تحديد حجمها بواسطة 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" >
جينة
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 النتيجة.

### الشرح

لأنضيف شيئاً على قولنا هنا سوى أننا نريدك أن تلاحظ كيف جهزنا القيمة الافتراضية بكتابة نصوص بين وسومات textarea وأيضاً أننا استخدمنا الأسلوب POST في ارسال البيانات مما جعلها لا تظهر في شريط العنوان .  
وأن الـ NAME تحدد اسم المتغير التي ستذهب إليه القيمة واسم المتغير في الكود لا يحتوي 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 إتاحة فرصة للمستخدم لتحديد أنواع الأشياء التي يريد أن يشترك فيها مثلاً أو إتاحة فرصة له لقبول إتفاقية أو غير ذلك أو رفض الجميع أو قبول الجميع .

يمكننا صنع مربع العلامة بواسطة 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 القيمة التي يتم وضعها في المتغير اذا قام المستخدم باختيار مربع العلامة في هذا الجزء :  
value= "سوالف"

إذا لم تقم بوضع الخيار value فستكون القيمة الافتراضية هي on عند اختيار المستخدم مربع العلامة وستكون فراغ إذا لم يقم المستخدم باختيار المربع.

ونقوم بوضع القيمة الافتراضية بإضافة الكلمة checked فإذا تم وضع هذه الكلمة يكون مربع العلامة مختار تلقائياً أما إذا لم نكتبها 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 إجابة )
<br>
<INPUT TYPE="CHECKBOX" NAME = "WIFE" CHECKED>
الذي أريد أن أفعله في الحياة هو أنني 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 التالي :

```
<?
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- النتيجة

## 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 اختيار أكثر من إجابة )
<br>
<INPUT TYPE="CHECKBOX" NAME = "WIFE" value= "زوجة" CHECKED>
الذي أريد أن أفعله في الحياة هو أنني أتزوج وأخلص وافتك من الزهق.
<br>
<INPUT TYPE="CHECKBOX" NAME = "jihad" value= "جهاد" >
أبغى أروح الجهاد واخلع رؤوس الكفرة والمشركين
<br>
<INPUT TYPE="CHECKBOX" NAME = "qran" value= "قرآن" CHECKED>
والله لو ألتحق بتحفيظ قرآن واحفظ القرآن 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 التالي فيه :

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

3- قم بحفظه باسم check2.php

4- قم بتشغيل الملف .

5- النتيجة

### تطبيق عملي (3)

1- افتح محرر النصوص واكتب الكود التالي :

```
<html dir="rtl">
<FORM ACTION="CHECK3.PHP" METHOD = "POST">
    مألذي 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 واخلع رؤوس الكفرة والمشركين
<br>
<INPUT TYPE="CHECKBOX" NAME = "alswalif[]" value= "قرآن" CHECKED>
    والله لو ألتحق بتحفيظ قرآن واحفظ القرآن 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 للنتيجة

### 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 بإضافه قيم يتم وضعها في المتغيرات عند اختيار المستخدم لها كما في [التمرين الثاني](#) وأردنا لفت النظر في التمرين الى شي يسمى بالمصفوفات فإذا أردنا مثلاً أن نجعل اسم المتغير متشابهاً وأجراء عمليات تكون أسرع عليه نستخدم المصفوفات ولن نتطرق إلى المصفوفات حالياً ولكن أردنا لفت نظرك فقط وسنقوم بالتكلم عن المصفوفات بالتفصيل في الدروس القادمة باذن الله هي والتكرارات بعد التكلم عن العبارات الشرطية في PHP .



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

ماهو اختيارك المفضل ؟ علما بأنه لايمكنك اختيار أكثر 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 بأنه 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 بحفظه 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 للعبارة التي بجوار الزر .  
ولقد قمنا بوضع عبارة checked لكي ترى كيف أن الأداة التي تحتوي على العبارة تكون محددة تلقائياً  
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 .  
وأيضاً لاحظ أننا استخدمنا متغيراً واحداً فقط لجميع الاختيارات بحيث أن جميع الأزرار قيمتها تعود إلى  
هذا المتغير .

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

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

**<select>** لنقوم بإنشاء القائمة و **<OPTION>** ونستخدم الخاصية **MULTIPLE** إذا كنا نريد إتاحة  
الفرصة للمستخدم أن يختار أكثر من قيمة ونقوم بوضع القيمة التي يختارها المستخدم في متغير  
بواسطة الخاصية **NAME** أو في مصفوفة متغيرات (وسيتضح مفهوم المصفوفات لديك جيداً 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> سحر </option>
<option> سارة </option>
<option> سميرة </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="إرسال">
</html>
```

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

```
<html dir="rtl">
<?
Echo " $wife . " " لقد أردت أن يكون اسم زوجتك ";
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 باختيار قيمة واحدة منها ثم تذهب هذه القيمة الى المتغير wife وصنعنا قائمة ثانية تسمح باختيار أكثر من عنصر واحد وقمنا بوضع هذه القيم في مصفوفة متغيرات (سيتضح معني المصفوفات في دروس قادمة ان شاء الله ) .

### الاداة الخفية (والمعلومات السرية!) (hidden control)

هناك بعض الأوقات تحتاج فيها إلى إرسال بعض المعلومات من صفحة ويب الى صفحة ويب أخرى 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 جداً مع النماذج المصنوع بواسطة الـ PHP إذ أنها تسمح لنا أيضاً 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 الـ NAME ونقوم بوضع المعلومات التي نريد اخفاءها في الـ VALUE .

نستطيع الاستفادة أيضا منها عن طريق الـ php وذلك عن طريق كتابة كود الـ HTML بواسطة الأمر (echo) في الـ PHP كما في المثال التالي :

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

هذا الكود الذي تراه عبارة عن كود HTML تم كتابته بالـ PHP عن طريق الأمر (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";
$car3="لاندكروزور";
Echo "<form method =get action='hid2.php'>";
Echo "ماهي السيارة التي تتمنى أن تشتريها أو تحظى بها ؟";
Echo "
<select name= 'favcar'>
<option>$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 "<br>لقد قمنا بعرض السيارات التالية عليك :";
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 السكريبت :

<http://localhost/hid.php>

### الشرح :

لقد قمنا بعمل نموذج بسكريبت الـ php لاحظ أننا استخدمنا الـ (") بدلاً من (') كما كنا نعمل في الـ html وذلك لأننا قلنا سابقاً أن القيم الحرفية ..... (راجع درس المتغيرات) ولقد قمنا بإدراج قيم متغيرات الـ php في كود الـ html مما يوفر علينا الكثير من إعادة الكتابة (في حال كان النص المستخدم طويلاً) .  
اقرأ المثال أكثر من مر وسيتضح لك المقال أكثر بأذن الله .

### استخدام حقول 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 فإن هذه الطريقة جيدة قليلاً في أن لا يري شخص معلومات الآخر السرية .

في الواقع مع ذلك فإنك لا تكون قد أدت حماية إذا كان الأسلوب المستخدم في إرسال بيانات المستخدم هو الأسلوب get إلا إذا كنت تستخدم تشفير البيانات ويكون أكثر جودة إذا استخدمت الأسلوب post وايضاً 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>$user<br>";
Echo " : وكلمة المرور هي ";
Echo "<br><br>$pass"
<?
```

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

## ارسال البريد الالكتروني بواسطة php :

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 سيتصله 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 و قم بتشغيل البرنامج واملا البيانات واضغط زر الارسال وستري ان الرساله تم ارسالها بنجاح .

# برامج عمليه

برنامج او سكربت ارسال بطاقات بسيط  
يحتوي من ملفين 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>

<br>
<br>
<INPUT TYPE = "radio" NAME = "card" value= " http://www.khalaad.f2s.com/Haram3.jpg" >
البطاقه الثانيه
<br>

<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 البطاقه" >
</form>
</html>
```





# الأوامر الشرطية

لقد أخذنا في الدروس السابقة فكرة عن المتغيرات وكيفية تعامل البيانات مع النماذج... في هذا الدرس سنتعلم كيفية التحكم بالكود بمعنى 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 بالـ PHP .

## 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 وتؤكد 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 ، وإذا 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 جانبية ثم تتجهز للذهاب إلى المركز التجاري .

هل لاحظت انك كنت تقوم بالبحث عن أشياء معينة فاذا وجدتتها ( 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

( إذا كان الشرط صحيحاً ) `IF condition is true`

{

`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 صحيحاً (true) والقيام بتنفيذ أشياء أخرى إذا لم يكن صحيحاً .... سيقوم الـ PHP بتنفيذ الكود التي بين { و } فقط اذا كان الشرط صحيحاً . أما إذا لم يكن صحيحاً فسيقوم بتجاوزه وتنفيذ الكود الذي يليه . ويمكنك أيضاً أن تقوم بجعلها بسطر واحد ولا تستخدم الأقواس بل تكتب الأمر مباشرة :

`IF condition is true excute function;`

لاحظ أنه لا بد 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 بإفترض أن هناك أمر في الـ php يسمى go out حسناً الآن الكود الذي نريد أن نقوم بكتابته هو :

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

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

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

## مقدمه الى القيم المنطقية (Boolean Values)

القيم المنطقية ترمز إلى الأشياء التي لا تحدث أكثر من احتمالين وهما إما صح وإما خطأ ، وهي نوع جديد من القيم غير التي كنت تعرفها سابقاً (مثل الرقميه والنصيه) .

مثال

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

لو قمت برؤيه النتيجة ستجد أنه يطبع الرقم واحد وهو قيمة المتغير إذا كان صحيحاً ، أما إذا 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 شيء 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 الأمثلة بسيطاً .

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

سيقوم الـ PHP في مثالنا هذا بفحص الشرط (6>5) فإذا كان صحيحاً (true) سيقوم بطباعة السطر (الرقم ستة أكبر من الرقم خمسة) ثم يقوم بطباعة end ، وإذا لم يكن صحيحاً فسيقوم بتجاهل الكود وطباعة (end) فقط .

يمكننا أيضاً 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 "r أكبر من المتغير $f$ 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 و 10 ؟
<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<$num)
{
echo "لقد اخترت رقم أصغر من الذي أفكر فيه" ;
Echo "$numالرقم الذي كان في مخيلتي هو " ;
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 اعطاؤها 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);
```

```
$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>" ;
}
```

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 بإخباره 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 "$numالرقم الذي كان في مخيلتي هو " ;
Echo "<br>" ;
}
```

فإذا لم يتطبق الشرطين فإنه يتركهما ويكتب الكلمة (لقد نجحت ) بدون أي كلمات أخرى مثلما كنا نكتب الكلمة (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 المساواة 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 المزدوجة (==) .  
في php4.01 تم إصدار علامة مساواة جديدة تقوم باختبار القيم ولا تعطي القيمة (true) إلا إذا كانت **أنواع القيم متساوية وأنواع البيانات** في المتغيرات ايضاً 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 شيء وذلك لأن 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 لا تساوي 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 يسمى 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 المنطقية (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 عدة 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 لا يوجد أمطار فسأقوم بالخروج الى المنتزه .  
ويمكنك أيضاً 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 العملية .

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 لا تكون بتلك الأهمية لكن يجب أن تعرف أن الرموز && و || لها الأسبقية والأفضلية على استخدام AND و OR .

## المعامل NOT ونظيره ( ! )

في الواقع لا يمكنك 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 شيء معين .

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

في المثال السابق يقوم الـ PHP بالتأكد من أن المتغير \$F لا يحتوي 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 الـ PHP إذا وجد المتغير فارغاً أو لم يتم إنشاؤه 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 بالـ php وهي معرفة إذا ما كانت قيمة أصغر أو أكبر من أو تساوي قيمة أخرى ، وهذه الأمثلة تعطيك مدخلاً أشمل لفهم هذه الدوال :

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

## 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 معامِل ( + ، - ، / ، \* ) باستخدام الأقواس .

وسيبدو ذلك واضحاً وجلياً في مثالنا التالي :

```
<?
$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 و else if )

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

```
If condition 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 لا يملكون المال";
}
?>
```

ويمكننا أيضا استخدام الهيكلية التالية :

```
If condition 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 ماتكلما عنه سابقاً

1- قم بإنشاء 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">أبو بكر 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="نصف"></p>
</form>

</body><html>
```

قم بفتح ملف وقم بتسميته msabqa.php

```
<?
<html dir = "rtl">
If $s == "3mar"{
الإجابة صحيحة
}
else
{
echo "الإجابة 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 ومحبية أكثر وتتيح لك اختبار قيمة متغير وإجراء أكثر من اختبار عليه .

**break;**

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

**EXIT;**

تقوم بعملية الخروج من الكود نهائياً ولاتطبق أي أوامر بعدها ، وفي الأمثلة التوضيحية التالية ستجد أن break تخرج من العبارة فقط (Statement) بينما exit; 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";  
?>
```

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

مثال ( 1 )

```
<?
$g= "ahmed";
Switch ($g) {
    Case "ahmed":
        Echo "مسموح";
        Break ;
    Case "khaled " :
        Echo "ممنوع";
        Break ;
    Case "saalem":
        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 ال html

إذا قمت بوضع مربع نص وأردت من المستخدم كتابة شيء فيه فإنه يستطيع ادخال أي شيء ولنفتراض أنه كتب في مربع النص كالتالي :

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

فسيقوم المتصفح بعرضها بعد معالجتها كالتالي :

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 شيء وضعه بين وسوم html

مثال :

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

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

htmlspecialchars();

حيث أنها ستقوم بمعاملة كود الـ html كنص عادي وطبيعي تماماً .  
إذاً نقوم بتعديل ملف الـ html.php ليصبح كالتالي :

```
<?
$name = htmlspecialchars($fname);
Echo “هذا هو الشكل بعد استخدام الدالة“;
Echo “<br>” . $name;
?>
```



# التكرارات والمصفوفات

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

في الواقع قد يكون لديك يومياً شيء تفعله بشكل مستمر مثل الإفطار في الصباح الباكر والنوم مساءً ، انك تستمر على هذا الروتين دائماً .... نحن نسمي هذا الشيء في لغة البرمجة التكرار .

هناك شيء آخر يسمى 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 إحدى 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 تسمى حصص إلا أن كل حصة تختلف عن الأخرى في المادة ! وهي مرتبة بشكل تصاعدي (الحصة الاولى ، الثانية ، الثالثة .....).

نسمي هذه التقنية بالمصفوفات .... المصفوفات عبارة عن متغير اسمه ثابت ولها أكثر من قيمة وكل قيمة لها رقم معين ولكي تحصل على القيمة فانك تكتب المتغير ثم رقم القيمة التي فيه، لا يشترط أن تكون هذه القيم متسلسلة فقد يكون هناك قيمتين ولكل قيمة رقم يختلف تماماً ويبعد كل البعد عن القيمة الثانية مثال رقم 1 و 258 كلاهما مختلف تماماً ويبعد كل البعد عن الآخر .  
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 تنفذ إلا عندما يكون الشرط صحيحاً أيضاً التكرارات 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 إكمال 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 وصيغته هذا التكرار هي :

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

مثال :

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

سيقوم الـ PHP أولاً بإعطاء المتغير \$d القيمة 10 ثم يقوم بعد بدء 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 الأقواس .

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 ما يوقفه بينما في الكود السابق استطعنا إيقاف الكود بسبب أننا كنا نضيف واحد على القيمة الموجودة في المتغير وكلما يتم إعادة 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 كالتالي :

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

مثال :

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

سيقوم التكرار بتنفيذ السطر الموجود بين القوسين أولاً ثم يقوم بتنفيذ باختبار 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 أولاً ثم قمنا بإجراء الاختبار .

### التكرار FOR

يختلف هذا التكرار عن سابقه لكن وظيفته هي نفس وظيفتهما وهي تكرار الأوامر عند حصول شيء معين

الصيغة :

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

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 للphp 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 الPHP آنذاك بالخروج من التكرار والذهاب الي الكود الذي يلي القوسين .

## المصفوفات

لقد قمنا بتعريف المصفوفات سابقاً بشكل بسيط وحاد 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 بتحديددها

مثال :

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

في هذا المثال سيقوم الPHP بإعطاء الفهرسة تلقائياً فسيقوم بوضع الرقم فتصبح المتغير فهرسته كالتالي :

```
$A[0] = "alfareees";
$A[1] = 13;
```

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

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

سيقوم الPHP بأخذ الفهرسة المعتمدة ولن يضع أي فهرسة أخرى يمكننا أيضاً أن نكتب أي فهرسة ولانعتمد على الترتيب في الارقام .

مثال :

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

هل لاحظت أيضاً أننا لم نقم بتعريف نوع متغيرات المصفوفة وقام الPHP بتعريفها تلقائياً بدلاً منا فمرة استخدمنا قيمة حرفية ومرة استخدمنا رقماً ورغم ذلك فلم يقم الPHP بعمل أي اعتراض إضافة إلى ذلك فإن الPHP يقوم بتحديد عدد عناصر المصفوفة تلقائياً فهو يعرف مثلاً من المثال السابق أن عدد عناصر المصفوفة الكلي هو عنصرين .  
يمنحنا الPHP ميزة أخرى 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 الـ PHP 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 الطباعة وعند كتابته بدون علامات تنصيص ... سيقوم الـ PHP بمعرفة ذلك تلقائياً .

يمكننا تعريف المصفوفات أيضاً بطريقة أخرى

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

مثال :

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

يقوم الـ PHP بإعطاء كل عنصر من عناصر المصفوفة رقم فهرسة فتصبح كالتالي :

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

إذن القيمة التي سيطيعها الـ PHP في النهاية هي ahmed ، لاحظ أن الـ PHP قام بإعطاء رقم الفهرسة وقام بالبدء من الصفر ولكن يمكننا جعل الـ PHP يبدأ الفهرسة من الرقم واحد كالتالي :

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

عند تعريفك لرقم الفهرسة للقيمة الأولى سيقوم الـ PHP بإعطاء أرقام فهرسة بشكل تسلسلي ، عندئذ ستصبح الفهرسة كالتالي :

الفهرسة 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 كالتالي :

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

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

مثال :

```
$r [ss]= "لمياء";
```

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

### قراءة المصفوفات واستخراج القيم

تكلّمنا 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");
?>
```

وإردت أن تطبع أسماء جميع الأشخاص المتواجدين فيها  
أولاً نحن نعرف أن المصفوفة إذا لم نقم بتعريف رقم فهرسة لها فإن PHP يقوم ببداية فهرستها من الصفر  
وعلى ذلك فإن رقم العنصر الأول 0 ورقم العنصر الرابع 3 ... على ذلك يمكننا بكل بساطة كتابة الكود التالي  
الذي يقوم بطباعة المصفوفة كالتالي :

```
<?
$people =array ("ahmed", "ali", "salem", "alfarsi");
echo "$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 أخرى وهي عن طريق التكرارات .

لنفرض أننا أردنا كتابة تكرار يقوم بطباعة الأرقام من واحد الى عشرة فإننا نستطيع كتابة التكرار بالشكل التالي :

```
<?
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 و قمنا 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 array.php اكتب :

```
<html>
لقد قمت باختيار التالي :
<?
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 الhtml على أنه سيتم تخزين البيانات تلقائياً بعد ذلك قام الPHP 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 مثلاً حوالي مئة دولة فيمكنك مثلاً 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 و قم بكتابة الملف list.php اعتماداً على معلوماتك السابقة في درس النماذج .

## دوال المصفوفات

### الدالة key

لنفرض أن لدينا مصفوفة مكونة من عنصرين :

مثال :

```
$s= array ("محمد", "علي");
```

الآن لنصف إليها هذه السطور

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

## الدالة current()

تقوم الدالة current بإيجاد القيمة لعنصر المصفوفة الحالي (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 الاخرى للمصفوفة ؟!

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 الحالي وقيمتة ( 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 بالانتقال 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 التالي :

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

فسيقوم الـ PHP في هذه الحالة بطباعة 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 PHP ببساطة بالبحث عن أكبر رقم فهرسة وبعد ذلك يبدأ بإعطاء الفهرسة تسلسلاً بعده فإذا كانت أرقام الفهرسة حروفاً بدأ من الصفر في إعطاء 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( Index,Element value العنصر =each (array) أرقام الفهرسة )

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 مع H أننا طلبنا طباعة الفهرسة (index) إلا أنه تم أخذ العناصر (elements) بدلاً من الفهرسة

يمكننا بواسطة هذه الدالة صناعة أشياء مفيدة 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>
<?
$s = array (658=>"465873", "سالم", "عادل", "عادل");
While (list($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 بواسطة 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 أيضاً 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 الـ PHP 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 أولاً ثم الصغيرة .. تتطلب هذه الدالة اسم المصفوفة التي سيتم عليها الفرز

```
Sort (ArrayName);
```

إذا قمنا بإنشاء مصفوفة بالشكل التالي :

```
$NaNo=array ("ali","saalem","hythem","Khaled","Ammar","Hesham");
```

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

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

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

### الدالة Arsort()

هذه الدالة تعمل بنفس عملية الدالة sort() ولكن هناك اختلاف بسيط فمثلاً لو كتبنا المصفوفة كالتالي :

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

وأردنا فرزها وطباعة الفهارس والقيم كما في المثال التالي :

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

قارن ناتج المثال السابق مع هذا المثال :

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

اعتقد أنك قد عرفت الفرق ففي المثال الأول 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 لا يوجد فرق بين sort و arsort إلا في أن الدالة sort تستبدل فهرسة الحروف بأرقام .

### الدالة arsort و Rsort()

تقوم بنفس عمل sort و arsort 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";
}
?>
```

```
<?
$NaNo=array ( "ad"=>"ahmed", "kh"=> "khaled");
arsort($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 التالي :

```
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

تكلّمنا 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) مثال :

sort

```
<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 الـ php على index ولم يعتمد على الـ element في الفرز .

## دوال المصفوفات الإضافية

هناك الكثير من الدوال التي يمنحنا إياها الـ PHP للتعامل مع المصفوفات والتي لا يكفي الوقت لذكرها الآن سنقوم بشرح أهم دالتين والمستخدم بكثرة وهي **array\_push()** و **array\_pop()**

لنفرض أننا قمنا بإنشاء مصفوفة بالشكل التالي :

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

وأردنا أن نضيف عنصر جديد لها فقمنا بالتالي :

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

انظر إلى العنصر الأخير الذي سيعطيه الـ PHP رقم الفهرسة (index) وسيكون رقم فهرسته هو 86 . نريد أن نلفت نظرك بأننا نستطيع عمل إضافة لعنصر على المصفوفة بطريقة أخرى وهي عن طريق الدالة **array\_push()** كالتالي :

**array\_push (ArrayName,اسم 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(اسم المصفوفة ArrayName)

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 .

## Explode و Implode

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;
?>
```

تقوم بحذف قيمة من مصفوفة وذلك لايغني 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 الـ PHP فإنه يقوم بتعريفها تلقائياً ويمكنك طباعة المتغيرات وقيمها مباشرة من غير تعريف .... ولكن هذه الميزة في الـ PHP 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 بسيط فيقوم بإغلاق هذه الميزة من باب الحماية ليس إلا .... لاتقلق يمكنك الحصول على البيانات فهي ما زالت موجودة ولكن يجب عليك أن تقوم باستخدام 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 كالتالي

```
<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 كالتالي

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

لاحظ أننا لم نستخدم \$ ولكن إذا أردنا الاحتفاظ بقيمة المتغير في متغير آخر فيمكننا ذلك بشكل عادي كالتالي :

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

طريقه بسيطة .... أليس كذلك ؟ ولكن .... لنفترض أن مزود خدمة الويب لديك حريص جداً ولذلك فقد ألغى أيضاً ميزة استقبال هذه القيم في المصفوفات .... يمكنه ذلك في ملف الـ php.ini في اعدادات الـ track\_vars الذي يقوم بمنع السيرفر من استخدام هذه المصفوفات (هذه الميزة يمكن إلغاؤها في php4 ) .... على ذلك انصحك بإرسال رسال تضرر وشكوي إلى مزود الخدمة لديك .. تعلن فيها أن الأمر أصبح لايحتمل .

## مصفوفه متعددة 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 :

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

echo

في أول خطوة :

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

قمنا بإخراج رقم الفهرسة الأساسي للمصفوفة والذي يعتبر هو الرقم التسلسلي للأشخاص 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 الـ PHP بطباعة العناصر الذي تحتويها المصفوفة التي تم طباعة رقم فهرستها ، ولاحظ (\$phone) أنها تشير إلى عناصر مصفوفة وليس فهرستها لأننا تجاهلنا فهرس 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 لعدة أشخاص (\$boy) ونريد أن نعرف مرحلهم الدراسية في الثانوية فأنشأنا لكل طالب قائمة منسدة بواسطة التكرار (list-each) بصناعة قوائم منسدة 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 وإرسال البيانات سيتم استقبال المصفوفة التي فيها نتائج القوائم المنسدة (\$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 php4 وهو يساعدك على معرفة عناصر مصفوفة معينة أو طباعة محتوياتها .

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

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

تعلّمنا في الدروس السابقة أساسيات من أساسيات البرمجة و أعطينا مثال عن الروتين في الحياة اليومية وهو أن تقوم بعمل شيء أكثر من مرة في الحياة اليومية مثل شرب الشاي أو شرب القهوة وغير ذلك ، 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 شيء يساعدك على التخفيف من هذا الروتين ... فمثلاً عند استخدامك لبرنامج MS Word قد تكون مللت من تنسيق عدة نصوص بطريقة معينة فأنت عند ذلك تقوم بصناعة ماكرو يقوم بفعل العمل الذي كنت تفعله في عدة خطوات بخطوة واحد فقط !!

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

## Function

الدالة هي جزء من كود البرنامج يتم تعريفه عن طريق المبرمج ليتم تنفيذ شيء معين بواسطتها ، تقوم الدالة بأخذ قيم وتسمى (arguments معطيات ) كمدخلات ، ثم تقوم بعمل بعض التعديلات على هذه المدخلات وتقوم بإخراج قيمة أخرى في أكثر الأحيان تقوم الدالة بأخذ القيم ووضعها في متغيرات أخرى تسمى بال(parameters) لكي يتم إجراء العمليات عليها داخل الدالة وهذه المتغيرات لا تعمل خارج الدالة أي أنها متغيرات خاصة بالدالة فقط ! ... في دروسنا السابقة قمنا باستخدام دوال عديدة مثل دوال فرز المصفوفات ودوال إيجاد نوع البيانات ،،،، هذه المرة سنقوم ببناء دوالنا الخاصة بنا ،، ومن صنعنا نقوم بإعطائها المعلومات والبيانات وهي تقوم بإجراء العمليات عليها ومن ثم إخراج الحلول ...

## تعريف واستدعاء الدوال

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

الصيغة :

```
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 بإجراء عملية الجمع على متغيرين وسنقوم بتسمية الدالة باسم sumnormal وهو اسم من تأليفنا وبدل على وظيفة وهدف الدالة ويمكن أن تقوم بتسمية الدالة بأي اسم تريده ولست مجبراً 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 الدالة ان وظيفتها انتهت وايضا نستخدمها إذا كان لدينا أكثر من قيمة ونريد أن نقوم بإخبار الـ PHP ماهي القيمة التي سيتم اعتمادها ففي مثالنا هذا أردنا إخبار الـ PHP بأن يقوم بأخذ المتغير \$a بأنه هو القيمة النهائية مع أنه لو لم نضع المتغير فسيتم اعتباره هو الناتج النهائي لأنه لا يوجد متغير آخر تم عليه أي عمليات

الذي اقصد أنه لو كتبنا الكود بالشكل التالي :

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

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

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

هنا يجب تحديد أي المتغيرين سيكون هو القيمة النهائية للدالة .

## شرح الدالة (sumnormal)

تقوم الدالة التي صنعناها بأخذ قيمتين ومن ثم فإنها تقوم بزيادته العدد الذي يتم تمريره 100

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

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

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

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

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

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

يمكننا أيضاً استدعاء دالة بشكل عادي إذا كانت هي تقوم بالطباعة  
مثال :

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

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

## print

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

```
<?
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 لا فرق 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 كالتالي :

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

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

## تمرير القيم الى الدالة

هناك نوعين من تمرير القيم

### 1 - تمرير القيمة مباشرة 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 القيمة عن طريق المرجع (passing by reference)

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

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

### اعداد قيمة افتراضية للدالة

تستطيع أن تجعل الـ PHP4 يقوم بإدراج قيمة افتراضية عند عدم تمرير متغيرات إليه  
مثال :

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

إذا لم يتم إعطاء قيمة للدالة فإنها ستفترض أن القيمة هي 40 مباشرة .  
أما إذا تم تمرير قيمة أو متغير فإنه سيتم العمل بالقيمة التي تم تمريرها بدلاً من القيمة الافتراضية  
مثال :

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

### مدى المتغيرات (variable scope)

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

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

مثال :

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

في المثال الأول استطعنا طباعة المتغير \$r ولم نسطع طباعة المتغير \$s لأنه محلي (لا يتم 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 محلي
$s = "programmer";
}
// استطعنا طباعته بشكل مباشر
echo $r ;
ala($s);
// يجب استخدام الدالة 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 بإحدى الطريقتين التاليتين :  
**الطريقة الأولى :**

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

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



**الطريقة الثانية :** هي أن نستخدم المصفوفة \$GLOBALS التي نستخدم 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 المستقرة (static variable)**  
أقصد بالمتغيرات المستقرة هي التي تكون قيمتها ثابتة  
مثال :

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

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

لاحظ عندما عرفنا المتغير بأنه static فإنه يحتفظ بقيمته 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 الأساس 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 إدراج نص كبير الحجم في صفحات متعددة  
هنا يمكنك اشتمال ملفات في داخل ملفات ال-PHP . 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 بإنشاء ملف php واكتب فيه ومن ثم احفظه باسم b.php

```
<?
Include ("a.txt");
?>
```

انقلهما الى مجلد السيرفر .. شغل ملف ال-b.php وانظر النتيجة .  
يمكنك أن تقوم بإنشاء ملف PHP وتحتفظ فيه بجميع الfunction 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 ما عليك أولاً  
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 الأخطاء (avoiding and handling errors)

إن مصطلح debug هو من المصطلحات الشائعة والشيقة في عالم البرمجة ، هذا المصطلح يشير إلى كيفية إصلاح أخطاء البرنامج وتوقعها قبل حدوثها ، هناك أنواع من الأخطاء تحدث بسبب المبرمج وهناك أنواع من الأخطاء تحصل بسبب المستخدم ، في العادة يجب أن يكون المبرمج متآلفاً مع مصطلح تتبع الأخطاء وإصلاحها .

قد يكون من أهداف تتبع الأخطاء الحماية بقدر أهميه البرنامج الجاري العمل عليه أو الموقع فكلما كان الموقع مهماً كان وجوب حمايته أكبر .

قد يكون من الأسباب التي تسبب تدميراً للمواقع هو أن صاحب الموقع يغطي كل صغيرة وكبيرة عن برنامجه الذي يركبه في موقعه وقد يكون برنامجه هذا غير محمي بسبب كاف أو يكون مسير بعدة ملفات فيقوم شخص بحذف ملف من الملفات الأساسية بسبب عدم دقة في التراخيص المعطاة مما يؤدي إلى دمار الموقع نهائياً .

وقد يكون صاحب الموقع مهملًا في الحد ذاته فلا يحتفظ بالمعلومات السرية لموقعه مما يسبب مشاكل أكبر من التدمير مثل اختلال الموقع بشكل كامل .

رسائل الخطأ في PHP لها طريقتها وتقنياتها الخاصة التي تسير عليها فهي ليست مثل الجافا وليست مثل cgi  
فال 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 هذا بسبب أنك تستخدم ال-PHP في صناعة موقع ديناميكي وتشترك معها الجافا سكريبت وتضع علامات التعليق الخاصة التي تقوم بإخفاء الأخطاء في الجافا مما قد يجعلك تشعر بالحيرة وتجن أين مكان الخطأ

```
<!--  
رساله الخطأ  
-->
```

## 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 ال-PHP رسالة خطأ لكن العجيب أنه لن يعطيك إياها بشكل صحيح فرسالة الخطأ تشير إلى أن السطر الرابع يحتوي على الخطأ بينما الخطأ هو في السطر الثاني .

وهناك خطأ آخر يحصل بسبب نسيان brace ( وهي الأقواس ) :

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

إذا كنت قد نسيت إغلاق القوس فهذا من الأخطاء الشائعة ، والأخطاء الإملائية لا يمكن حصرها ، إنها أشبه بقواعد اللغة ، لكن أكثر الأخطاء الإملائية الشائعة في برامج الـ PHP

### 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 :

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

سيعطيك رسالة خطأ :

Fatal error : call to Undefined function : htmlspecialchars().

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 لا يتم 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 المنطقية 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 الأول الذي ذكرناه سابقاً (تقييم العمر) .

## اخطاء التكرارات

قد يكون لديك أيضاً تكرار فيه خطأ ولا يقوم بالتوقف نهائياً مثل هذا التكرار :

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

لم نقم بعمل شيء 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 return 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 الدالة 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 باستخدام الـ 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 للكلمة التي تريد المستخدم أن يقوم بادخالها .

## النمط (pattern)

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 عدة 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 الـ Regular Expressions :

```
<?
$words="one, two, three, four, five,";
if (ereg("one",$words)) echo " one ' لقد وجدت العدد " ;
?>
```

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 انتهى .

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 الـ ereg يقوم بإنشاء المصفوفة من جديد عند كل استعمال له فخذ حذرك من هذه النقط أيضا فإن الـ ereg 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 الـ O 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 وoned ورغم ذلك فإن الدالة لم تأخذ اعتباراً 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 شيء .

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 \$reu بواسطة ereg مع \$word ولم يتطلب منا ذلك أي شيء إضافي غير اسم المتغير المراد البحث عن قيمته في الجملة .

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 الـ PHP هذا المعنى فإننا نقوم بوضع (\) قبل علامة التنصيص .  
أيضاً 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 ال(.) في الRegular Expressions نقوم بوضع (\) قبلها . مثال :

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

في هذا المثال لن يتم طباعة أي شيء لأنه لا يوجد أي كلمة تطابق (love....) لأن ال(.) 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 بتحديددها أو الحروف التي أقوم بتحديددها .  
مثال :

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

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

Hey – He – Hew - Homer

ولكنها لا تطابق :

Hty – Hnt - Hlay

أتمني أن تكون فهمت ما أرمي إليه

يمكننا أيضاً أن نقوم بإخبار الregular 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 الre بأن يقوم بفحص الجملة فإذا وجد أي كلمة تبدأ بـ h ولا تحتوي على o أو e فإنه يقوم بإعطاء true وإذا لم يجد يقوم بإعطاء false وهذا الكلام يطابق الكلمات التالية :

Hay - Hana - Hkg

ولا يوافق هذه الكلمات :

Home – Hore - Here

يمكننا استعمال اختصارات لبعض الأمور فمثلاً إذا كنا نريد كلمة لا تحتوي على أي رقم كنا سنكتب كالتالي  
[^123456789]

يمكننا أن نستعمل اختصار لهذا الموضوع كالتالي :

[^0-9]

وحتي إذا أردنا أن يتأكد من وجود رقم من واحد الى تسعة فقط علينا مسح ال^

[0-9]

وأيضاً 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 0-9 أو حروف A-Z أو 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 الـ php بالبحث عن في الجملة فإذا وجد الجملة تبدأ بحرف h كانت قيمة الـ ereg تساوي true وإذا لم يجد كانت قيمة الـ ereg تساوي false

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

في هذا المثال ستكون قيمة الـ ereg خطأ لأن العبارة لا تبدأ بحرف g يمكننا فعل العكس بواسطة العلامة (\$) 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 بواسطة العلامة ( | )

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

في هذا المثال سيقوم الـ PHP بفحص الجملة فإذا وافقت أحد النمطين كانت قيمة الـ ereg 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 بتاتا  
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 واحدة 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 أردناه 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 التالي :

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\B]	الحروف التي بين \w و \W
.	أي حرف
(abc)	تقوم باعتبار abc كمجموعه ..
?	حرف او مجموعه حروف مكرره مره اوغير مكرره نهائيا
+	حرف او مجموعه حروف تتكرر مره او اكثر
*	حرف او مجموعه حروف تتكرر مره او اكثر او قد لاتتكرر نهائيا
{x,y}	تكرير بعدد معين من المرات ..
{,y}	تكرير بحد اقصى من المرات ..
{x,}	تكرير بحد ادني من المرات ...
{x}	تكرير بعدد معين من المرات
^	في بدايه النص
\$	في نهايه النص

### تعبير للتأكد من ايميل

^[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)\*@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)\*\$

شرح التعبير

الرمز	الشرح
^	يجب ان يبدأ النص
[a-zA-Z0-9-]	أي حرف من a-z كبيراً كان او صغيراً او _ او ارقام
+	وقد يكون هذا الحرف متكرراً اكثر من مره
(\.[a-zA-Z0-9-]+)	بالاضافه الى انه قد يتبع النقطه وحروف وارقام
*	وقد لايتبعه او قد 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 كالتالي :

```
<?
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 أو غير 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 (\) ب (/)

كل ذلك ممكن بواسطة الدالة `ereg_replace` وقواعد الـ `regular expression` التي أخذناها سابقاً  
البنية التي نستخدمها للدالة كالتالي :

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

نضع في مكان `reg` القاعدة للـ `regular expression` ونضع مكان `string` الحرف الجديد ونضع بدلاً من `var` 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 عبارة `echo`

هو من أقدم الأساليب 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 الـ `html`

قد تستخدم كود جافا سكريبت ويتم إخفاء الأخطاء وسط علامات التعليقات فعليك حينئذ فحص كود الـ `html` 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 ما عليك أن تقوم بوضع `@` أمام الدالة لكي يتم تجاهل الخطأ عند حدوثه .  
مثلاً نحن نعلم أن القسم على الصفر من الأشياء الغير مقبولة في الـ `PHP` 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 الـ PHP يوفر رقم عظيم من المميزات عن الـ html لبناء مواقع الويب ، من الأشياء الأساسية التي لم نتكلم عنها حتي الآن هي الموثوقية (أو الاستقرار ) وهو بالمعني الصحيح والصريح :

القابلية على الاحتفاظ بالمعلومات بين صفحتين منفردتين أو مختلفتين في المستعرض ...

بدون أي إضافات ، HTTP لا يوفر 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 منتدى المطور العربي فان كل الطلبين ليس لهما علاقة ببعضهما ...

بمصطلح آخر يمكننا أن نقول أن الـ HTTP 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 أخرى فإنها لاتعرف إن كان هو نفس المستخدم أو لا !

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

فلقد رأيت كيف استطعنا ارسال معلومات من صفحة إلى صفحة بدون خسران أي معلومات وذلك عن طريق الـ html وبالرغم من ذلك فإن المستخدم عندما يقوم بإغلاق الصفحة عند استقبالها للبيانات فان ذلك يعني فقدانها للأبد ، عن طريق استخدام الـ PHP 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 للـ PHP ، يتطلب منا مفهومية جيدة في كيفية استعمال الـ PHP في التفاعل مع المستخدم والمتصفح الذي يستخدمه لكي نتغلب على نقاط الضعف التي في الـ http .

هذا هو موضوعنا لهذا اليوم والذي سنتكلم فيه عن :

- 1 - الـ HTTP والـ html ومحدودية قدراتهم ، وكيف يستطيع الـ PHP التغلب علي القصور فيهم .
- 2 - الاحتفاظ بالمعلومات التي نريد أن نستخدمها بين طلب لصفحتين مختلفتين .
- 3 - مكنكة الحفاظ على البيانات .
- 4 - الكعكات (cookies) وكيفية استخدامها .
- 5 - الـ 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 الـ PHP .. إنه يحتوي على الكثير من بعض الأمثلة 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 أسئلة 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 الhtml والhttp 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 الصفحة الثانية أو الثالثة في السيرفر .

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

## استخدام الحقول المخفية

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 "مرحباً بك يا " . $name ;
Echo '
<br>
<form method="POST" action="quiz3.php" dir="rtl">
<input type="hidden" name = "thename" value = "'.$name.'">
</p>
<p dir="rtl"><input type="radio" value="أبوبكر الصديق"
name="khlifa">أبوبكر
</p>
</p>
```



```

<p dir="rtl"><input type="radio" value="عمر بن الخطاب" checked
name="khlifa">عمر
    </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 المسابقة يا $thename ;
$range=0;
$co = 0;
    if ($khlifa=="أبو بكر الصديق") {
        $range=$range+10;
        $co = $co +1;
    }
    if ($faroq=="عمر بن الخطاب")
    {
        $range=$range+10;
        $co=$co+1;
    }
    if ( $range < 10)
    {
echo "ليس هناك أي إجابة صحيحة";
    }
    else
    {
echo "<br>". "$co = عدد الاسئلة التي أجبت عليها";
echo "<br>". "$range = الدرجة التي حصلت عليها";
    }
}
?>

```

قم بوضع الملفات الثلاثة السابقة في مجلد السيرفر ثم قم بتشغيلها

## الشرح

قمت في هذا المثال بمحاولة صنع مكنكة تواصل للبيانات ، بمعنى أنني أحاول أن أقوم بالاحتفاظ بالبيانات عبر الثلاث صفحات بشكل متواصل ، لاحظ أنني كنت اختبر في quiz2 و quiz3 باختبار المتغيرات قبل طباعة أي شيء فقد يقوم المستخدم مثلاً بالاحتفاظ بالصفحة التي وصل إليها في المفضلة ثم يقوم باكمال المسابقة في وقت آخر ولكني لا أريد ذلك بل أريد أن 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 أحتفظ دوماً بقيم المتغيرات في متغيرات جديدة في حقول مخفية وكلما كان عدد المعلومات أكبر في كل مرة كان عدد الحقول المخفية أكثر ، إن لهذه الطريقة أيضا مشاكلها فقد يفتح المستخدم كود الhtml 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 أكثر تعقيداً باستخدام الregular expression بمحاولة تلغيم البيانات بواسطة 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 وقيمه 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 التالي فيها :

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

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 فإذا وجد انه قد تم ارسالها (بواسطة الرابط الذي تم الضغط عليه) قام بطباعة (تم تسجيل الدخول) وإذا لم يجدها قام بطباعة (لم يتم تسجيل الدخول) بالإضافة إلى طباعة الرابط الذي يحتوي على المتغير في طياته

## الكوكيز أو الكعكات (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 أخرى 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 زيارته لموقعك في مرات أخرى.

الكوكيز مفيد للاستخدام في الأشياء البسيطة والغير خطيرة ، لكنه الآن يستخدم بشكل سيئ ، 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 مفيداً عندما تضمن أن جميع زوار موقعك تسمح متصفحاتهم بالكوكيز (مثل طلبية 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 الكوكيز  
الكوكيز عبارة عن قطعة صغيرة من البيانات التي تستخدم لتخزين اسم متغير وقيمتها مع معلومات حول الموقع التي أتت منه وتاريخ انتهاءها .

الكوكيز عبارة عن تقنية للتخزين من جهة العميل (client-side storage) تتخزن في ملفات في جهاز العميل  
يتم العبور إلى هذه الكوكيز ومسحها من المكان التي أرسلت منه .

عندما يطلب المستعرض صفحة من السيرفر وهذه الصفحة تقوم بتخزين كوكيز فإن السيرفر يقوم بإخبار المستعرض بأنه سيقوم بوضع كوكيز للاستعمال لاحقاً .

عندما يتم طلب الصفحة في مرة أخرى يقوم المستعرض بإرسال البيانات التي تم إنشاؤها سابقاً عند طلب الصفحة .  
يتم انتهاء مدة الكوكيز بانتهاء وقت صلاحيتها المحدد من قبل السيرفر ويتم مسحها فوراً عند إغلاق الصفحة إذا كان وقت صلاحيتها صفراً من الثواني .

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

يقوم السيرفر بإرسال الكوكيز عبر الـ HTTP Headers الذي يتم إرساله قبل أي مخرج من مخرجات الـ html والمستعرض أيضاً يقوم بإرسال الكوكيز عبر الـ HTTP Header بالإضافة إلى أن المستعرض يتعرف على من سيقوم بإرسال الكوكيز فلو كانت الكوكيز مثلاً مرسلة من قبل الموقع [www.php.net](http://www.php.net) فإنه لن يقوم بإرسالها إلى موقع [www.phpbuilder.com](http://www.phpbuilder.com) .

بإستطاعتك عند إنشاء الكوكيز تحديد مسار يتم إرسال الكوكيز لكي يتم إقتصار عملية العبور إلى الكوكيز إلى أماكن معينة .

قبل أن نقوم بوضع كود بسيط سنقوم الآن بتعريف كيفية تخزين الكوكيز وكيفية قراءتها :  
كون الـ PHP لغة حديثة لعمل سكريبتات ويب فإنها تأتي بدعم كامل للكوكيز بواسطة الدالة `setcookie()` باستثناء أنك عند استعمالها يجب استعمالها قبل طباعة أي مخرجات `html` .

تأخذ 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 الـ timestamp 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 الأخرى التي يتم استخدامها أيضاً 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 آخر ( مثلاً المسار كان `pag\url\one` وتم تغييره إلى `page\url\two` فإن المستعرض لن يقوم بإرسال البيانات إلي الصفحة لأنه تم تحديد المسار الذي سيتم إرسال الكوكيز إليه )
- ✓ الدومين الذي سيتم إرسال البيانات إليه وهو مفيد في حالة ما إذا كان هناك أكثر من دومين تريد إرسال الكوكيز إليه
- ✓ 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 إحدى 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 "<br><br>". $thename . "قيمة المتغير الذي لديك";
echo " = قيمة الكوكيز" . $rname ;
?>
```

## الشرح

### عند تشغيل الصفحة لأول مرة

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

### المرحلة الثانية

الآن قم بكتابة أي شيء في مربع النص (اكتب اسمك مثلاً) ثم قم بضغط زر الإرسال سيتم إرسال 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 إرساله عند طلب الصفحة (لأننا نعرف أنه يتم إرسال 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 إرساله على هيئة متغير ويتم إرساله ومن ثم طباعة النموذج وقيمة المتغير \$thename وقيمة الكوكيز الذي يوجد بجهازك !

# بدايتك الى session

ال session هي عبارة عن تقنية للترابط مع المستخدم وهي موجودة ضمن ال PHP4 ولم تكن موجودة ضمن الإصدارات التي قبله بل كان يجب أن تقوم بتركيب مكتبة لكي تستطيع استخدام هذه التقنية ، يعتمد فهمنا لل session على فهمنا للكوكيز وكيفية استعمالها ولقد تكلمنا عن الكوكيز بشكل جيد في الدرس السابق ، يستخدم ال session لعمل ميكانيكية تواصل بين المستخدم والسيرفر ، فلقد قلنا أن ال http 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 ال session لأجل عمل تقنية تواصل بين المستخدم والموقع ، فإستطاعتك 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 مشتري إلي سلة التسوق أو حذف مشتريات.

قبل أن أتكلّم عن كيفية استخدام ال Session وإعطاء بعض الأمثلة البسيطة ، سأقوم بالتكلم عن كيفية إعداد ال session مع ال PHP .

## اعدادات ال session في ال PHP

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

### اعداد طريقة التخزين

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

ستجد هذه العبارة مكتوبة في الملف كالتالي بشكل افتراضي :

```
session.save_handler = files
```

وهذا الإعداد يقوم بتحديد طريقة التخزين لل session 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 أخرى 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 ال session 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 ال PHP مثل قواعد بيانات ال 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 للـ session

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

هذا الإعداد يقوم بتحديد إذا ما كان الـ session سيتم إنشاؤه تلقائياً عند كل زيارة للموقع أو لأي صفحة من صفحاته بدون إدراج كود الـ session في كل صفحة ... وعلى ذلك فإنك تقوم بوضع القيمة إلى (1) إذا أردت ذلك .

وعلى إفتراض أنك لا تحتاج إلى أن تجعل الـ PHP يقوم بعمل session لكل صفحة تلقائياً ومن غير طلب فستقوم بوضع قيمة هذا الإعداد الى (0)

## SID

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

```
fc94ad8b1ee49ef79c713ee98ac1fcc4
```

هناك طريقتين يستطيع بها الـ PHP متابعة الـ SID 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 الـ session مع الكوكيز :

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

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

```
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 قيمة الاعداد (session.use\_cookies) تساوي واحد فان لا داعي لاستدعاء الدالة set\_cookie() لإعداد الكوكيز بل سيتم اعدادها تلقائياً بواسطة الـ PHP

## 2 - الإضافة أو الكتابة إلى عنوان الصفحة

إن إضافة عنوان الـ SID 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 الـ session

لقد أخذنا حتي الآن معلومات تجعلنا ندخل عالم البرامج المسيره بالـ session بدون خوف ، سأبدأ الآن في طرح بعض الأمثلة البسيطة التي تثبت لديك بعض المفاهيم الأساسية في الـ session ... سأشرح في هذا المثال كيفية إنشاء الـ SID 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 :  
سكرت يقوم بإنشاء وتسجيل متغير session

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

echo "عدد زيارتك 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 نأخذ كمصطلح .  
متغير الجلسة هو الـ (session-variable) أو متغير الـ session أو سمه ما شئت .

## الشرح

يقوم هذا السكرت 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 مرحباً بك أيها الزائر الكريم في موقعنا ، ثم قلنا له إن عدد زيارتك 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 بتحديدده سابقاً من ملف PHP.ini ، وهذا الكوكيز يحتفظ بقيمة الـ SID للـ session .

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

```
session_start();
```

وعلى هذا فإن الـ PHP سيفهم أنه سيقوم بتخزين القيمة في الكوكيز الخاص بالـ session .  
قمنا بجعل المتغير co كعداد بسيط لعدد المرات التي سوف نقوم بها زيارة الصفحة فعند عمل تحديث للصفحة سيتم زيادة العدد بمقدار واحد

```
$c++;
```

وطبعاً قبل زيادة العدد بقيمة واحد فإنه يتم حساب القيمة السابقة للمتغير عند إنشائه تلقائياً ... ومن ثم يتم الزيادة وبعد ذلك طباعة القيمة .

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

في هذه الصفحة نقوم بطباعة قيمة الـ SID وذلك بطباعة قيمه المتغير \$PHPSESSID (الذي هو اسم الكوكيز الخاصة بالـ session) .

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 لمدة 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 بالـ session 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 الـ session

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 الكوكيز الخاصة بالـ session

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 السكريبت نكون قد أنهينا درسنا عن مقدمة بسيطة للـ session . هذه مجرد مقدمة ولكي نستطيع أن نتعمق بالـ session فيجب علينا أن نتعلم شيئاً عن قواعد البيانات .



بسم الله الرحمن الرحيم والصلاة والسلام على اشرف الانبياء والمرسلين ، سيتم التطرق في هذا الدرس بمشيئة الله الى ما يلي :

- عبارة ال if .
- عبارة ال else .
- عبارة ال elseif .
- عبارة ال switch .
- حلقة التكرار while .
- حلقة التكرار for .
- حلقة التكرار do while .

### - عبارة ال if :

استخدام ال if في كتابة السكريبت شي أساسي ، وكما في لغات البرمجة الأخرى فإن ال PHP تتبع نفس الاسلوب في كتابة ال if ، فيمكن تحديد شرط معين مقترن بال 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 إرسال الرسالة ، او انه لم يفعل ذلك وبالتالي عرض رسالة ( فضلاً قم بتعبئة البيانات كاملة ) ، لعمل ذلك نحتاج معرفة أسماء المتغيرات في النموذج ، ولذلك فلنفترض أن المتغيرات كالتالي :  
( الإسم \$name ) ، ( البريد \$email ) ، ( الرسالة \$later ) ، ولعمل الشرط الأول ( اذا كان الإسم لم يُدخل فلن يتم ارسال الرسالة ) :

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

والمعنى أنه إذا كان المتغير \$name لا يحتوي على أي قيمة ( أي فراغ ) فسيتم تنفيذ السطر التالي وطباعة 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 == "" ) {
    echo فضلاً قم بتعبئة البيانات كاملة ;
    <br>
    echo لم تقم بإدخال الإسم ;
}
?>
```

### - عبارة ال else :

هذه العبارة تتيح امكانية وجود اجراء ثاني لعدم تحقق الشرط ، ففي مثالنا السابق كان الاجراء طباعة الجملة اذا تحقق الشرط ، ولكن في حالة عدم تحقق الشرط فلن يكون هناك اجراء لتنفيذه ، بل ان الاجراء سيتم تنفيذه اذا تحقق الشرط ومن ثم سيتم اكمال بقية الاسطر ، وفي حالة مثل هذه الحالة يتم استخدام ال else لوضع اجراء آخر في حالة عدم تحقق الشرط ، وبالمثال يتضح المقال :

```
<?
if ( $name == "" ) {
    echo فضلاً قم بتعبئة البيانات كاملة ;
}
else
{
    echo تم ارسال الرسالة ، شكرا لك ;
}
?>
```

في هذا المثال سيتم طباعة الجملة (فضلاً قم بتعبئة البيانات كاملة ) اذا تحقق الشرط أن المتغير \$name لا يحتوي على أي قيمة ، وسيتم طباعة الجملة (تم ارسال الرسالة ، شكراً لك ) في حالة عدم تحقق الشرط ، أي في حالة وجود قيمة في المتغير \$name ، مع ملاحظة أن هذا المثال يحتوي على شرطين وليس شرط واحد ، فالظاهر هو شرط واحد ( \$name == "" ) ولكن العبارة else تعتبر شرطاً بحد ذاتها ولو لم يكن هذا الشرط مكتوباً ، وكما هو واضح فمعنى هذا الشرط هو ( إذا كان غير ذلك ) فقم بطباعة الجملة .

يمكن أن يكون الشرح غير واضح تماماً ، ولكن أهمية فهم الطريقة ستوضح في الأسطر القليلة القادمة .

### - عبارة ال elseif :

في العبارة السابقة ذكرنا أنه يوجد شرطين واجرائين ، أحد هذين الشرطين غير مكتوب بل هو مفهوم من ادراج العبارة else ، وفي حالات كثيرة لا يكفي مجرد شرطين واجرائين لاتمام بعض السكريبات المعقدة ، فلذلك يمكن نستخدم العبارة elseif مع ال if لعمل مثل هذه السكريبات ، فلو افترضنا أن لدينا عداد لزوار الموقع ونريد اظهار العداد بحيث يتم قراءته بشكل جيد ، اي بمعنى اخر اذا كان عدد الزوار ( 1 ) فسيتم طباعة الجملة ( عدد الزوار : زائر واحد فقط ) واذا كان ( 2 ) فسيتم طباعة الجملة ( عدد الزوار : زائرين ) ... وقس على ذلك ، فعندما يكون عدد الزوار ( 1 ) فسيتم عرض الجملة الأولى فقط وعندما يكون عدد الزوار ( 2 ) فسيتم عرض الجملة الثانية فقط ، وهكذا لبقية الشروط .

بافتراض أن المتغير ( \$counter ) هو عداد الزوار ، فالمثال التالي يبين ما تم شرحه سابقاً :

```
<?
if ( $counter == 1 ) {
    echo "عدد الزوار : زائر واحد فقط ";
}
elseif ( $counter == 2 ) {
    echo "عدد الزوار : زائرين ";
}
elseif ( $counter >= 3 && $counter <= 10 ) {
    echo "عدد الزوار : $counter زوار ";
}
else {
    echo "عدد الزوار : $counter زائر ";
}
?>
```

كما هو واضح في المثال السابق سيتم 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 لتحقيق الشرط .

### - عبارة ال switch :

هذه العبارة قريبة جداً من العبارة if ، ولكن يمكن استخدام أكثر من شرطين بأسلوب آخر ، غير انه يجب اسناد قيمة معينة لل case وهي هنا بمثابة الشرط ، لكي يتم تنفيذ الاجراء المحدد لذلك الشرط أو ال case ، وفي النهاية الأمر يعود الى المصمم وإيهما يفضل ، وكما في المثال السابق يمكن كتابة مثال بال 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 العملية مجهدة لانه يجب كتابة سطر لكل قيمة كما يلي :

القيمة : 3  
ال 3 : case  
الاجراء : طباعة ( عدد الزوار : ( العدد ) زوار )  
القيمة : 4  
ال 4 : case  
الاجراء : طباعة ( عدد الزوار : ( العدد ) زوار )  
القيمة : 5  
ال 5 : case  
الاجراء : طباعة ( عدد الزوار : ( العدد ) زوار )  
..... الخ ...

وفي المثال التالي ساتغاضى عن الشرط الثالث بكامله ، واذكر بقية الشروط والحالات لمجرد فهم طريقة عمل هذه العبارة :

```
<?
switch ($counter)
{
case "1";
    echo "عدد الزوار : زائر واحد فقط ";
break;
case "2";
    echo "عدد الزوار : زائرين ";
break;
default;
    echo "عدد الزوار $counter : زائر ";
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 .  
Default وهي تقابل العبارة else أي بمعنى أنها لأي حالة لم يتم ذكرها في الشروط .

### - حلقة التكرار while :

وهي أبسط حلقات التكرار على الإطلاق ، بحيث تأخذ شرط واحد فقط وتبني على تنفيذ ما بين علامات الشروط { } ، والفرق الوحيد بينها وبين ال if هو انها ستقوم بتنفيذ الاجراءات طالما كان الشرط صحيحاً ، وهذا يعني احتمال تنفيذ الإجراء أكثر من مرة ، وهذا الدالة مفيدة في ادراج الحقول من الجداول وغيرها من الاستخدامات ، بحيث لو افترضنا وجود جدول معين في قاعد بيانات ونريد ادراجه في صفحة PHP ، فسيكون من اهم خيارات الاستخدام هذه الدالة ، وبإذن 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 )
{
    echo "العدد أقل من 50 ";
    $total +=10;
}
?>
```

كبير بسيط يمكن معرفة أن الجملة ( العدد أقل من 50 ) سيتم طباعتها 5 مرات ، لأن حلقة التكرار while قامت بتنفيذ الاجراء طالما أن الشرط صحيح ، وفي المرة الأولى كان المتغير ( \$total ) يساوي ( 10 ) والشرط صحيح لأن 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 الاجراء .... 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 آخر ، 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 )
{
    echo "العدد أقل من 50 ";
}
?>
```

وللتوضيح فان تركيب الـ for هو على الشكل التالي :

```
for ( القيمة الافتراضية; الشرط; مقدار الزيادة )
{
    الاجراء 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 ";
    $total +=10;
}
while ( $total <= 50 );
?>
```

وفي نهاية الدرس اتمنى الفائدة للجميع

بسم الله الرحمن الرحيم والصلاة والسلام على اشرف الانبياء والمرسلين اما بعد ، ففي هذا الدرس بمشيئة الله تعالى سنتطرق الى مفاهيم عامة عن قواعد البيانات عموماً 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 ، وبالإضافة الى المميزات الأخرى فان هذه النوع يعتبر الأكثر استخداماً في جميع التطبيقات سواء المستخدمة في 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 شخص 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 ) .

بالتحقيق 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 احتياجاتنا مع الـ PHP ، ولذلك سأتطرق 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 أهمها الطريقة التقليدية المباشرة بالاعتماد على نظام الدوس في ذلك وبدون استخدام أي برامج أخرى للإدارة .

### الاتصال بال Mysql ، والتعامل معها :

كما قلنا أن الطريقة التقليدية هي الاتصال بقواعد البيانات عن طريق سيرفر ال Mysql وبدون استخدام أي مكونات أخرى ، ولعمل ذلك نحتاج أن نعرف مسار سيرفر ال 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 ال Dos .

عموماً طريقة الاتصال بقاعدة البيانات هي كالتالي :

```
mysql -h HostName -u UserName -p
```

مع استبدال ال HostName باسم السيرفر لديك ، سواء كان السيرفر على نفس الجهاز وفي هذه الحالة تكتب localhost ، أو أن السيرفر الذي تود الاتصال به ليس على نفس الجهاز وبذلك تكتب المسار الكامل لاسم السيرفر ( HostName ) ، ومع استبدال ال UserName باسم المستخدم الخاص بال Mysql لديك ، بعد ذلك سيتم طلب كلمة المرور الخاصة بقاعدة البيانات بعد الضغط على Enter ، قم بادخالها وسيتم فتح الاتصال بال Mysql ، كما يمكن كتابة mysql فقط ليتم فتح الاتصال بقاعدة البيانات فقط اذا كنت تعمل على نفس الجهاز وليس جهاز آخر .

سيظهر المؤشر الخاص باوامر ال Mysql كالتالي :

```
mysql>
```

وبهذا نكون وصلنا الى المكان المطلوب لكتابة اوامر ال 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 ال <mysql ) ، سيتم استعراض قواعد البيانات في السيرفر الذي قمنا بالاتصال به ، وفي حالة عدم وجود أي قاعدة بيانات قمت باعدادها من قبل ، فإن 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 السابقين .

- في السطر قبل الأخير ، أي قبل علامة الاغلاق ( ) ، سيكون بدون فاصلة .  
- السطر الأخير يحتوي على افعال عملية انشاء الجدول بالعلامة ( ; ) .

عموماً هذا المثال يعطي نبذة بسيطة عن كيفية اجراء مثل هذه الاوامر ، وسنتطرق الى بقية الاوامر 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 بانشائها .

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 الاوامر 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 بطبيعة 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 ، وإذا 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 ايضاً ، فيمكن استبدال اسم الجدول users بأي اسم لجدول في قاعدة البيانات المستخدمة ، وللتأكد من اسماء الجداول قم باستخدام الطريقة السابق ذكرها وهي ( show tables ) .



النقطة الأخيرة التي سأتطرق لها هي ما يجب معرفته حول الأمر select وهو كثر استخدامه في التعامل عن طريق البي اتش بي ، وبالتالي يجب عليك فهم طريقة كتابته بشكل كامل ، بالإضافة 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 .

فلنفترض 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 الآخر limit هي ما سيتم التطرق اليه في المثال التالي الخاص بالحالة الخامسة :

### الحالة الخامسة :

```
select * from users order by counter limit 1;
```

وال limit تعني عدد الصفوف المختارة ، أي لو قمنا بكتابة المثال السابق بدون ال limit ستجد أن جميع البيانات سيتم اختيارها ، ولكن باستخدام ال 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 طريقتي التعديل والحذف ليكتمل الدرس ، وسنبداً 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 ) ، وال users هو اسم الجدول الذي نعمل عليه ، وفي السطر الثاني قمنا بإسناد القيمة Naser الى الحقل name ، والسطر الذي يليه قمنا بإسناد القيمة 30 الى الحقل counter ، ولكن لو توقفتنا 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 كافٍ ، ولكن مع التمرس والمحاولة ستجد أن المسألة منطقية وواضحة بشكل كبير ، عموماً لم يبق لدينا إلا 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 ( البارامتر - المتغيرات - المعطيات )
{
    هنا يتم كتابة الكود
Return ( المتغيرات - 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 ( المتغير - 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 علامتين { } ، كما يجب أن ننهي الدالة بكلمة return لإعلام الدالة بأن وظيفتها قد انتهت بالإضافة إلى 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 آخر الشفرة أو العكس .

### - إظهار نتيجة الدالة ( طباعة الدالة ) :

نستخدم الأمر الخاص 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 اختلفت 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 ال PHP :

```
<?
$a = 6;
$b=2;
$c= $a +$b;
// سوف نحصل على ناتج الجمع 8

$c= $a -$b;
// سوف نحصل على ناتج الطرح 4

$c= $a * $b;
// سوف نحصل على ناتج الضرب 12

$c= $a /$b;
// سوف نحصل على ناتج القسمة 3

$a = 7;
$b=2;
$c= $a % $b;
// سوف نحصل على باقي القسمة 1
?>
```

## - عمليات Assignment :

=

احفظ القيمة في المتغير ، بمعنى خزن القيمة 3 في المتغير \$a :

```
<?
$a = 3;
print $a;
// يطبع 3
?>
```

+=

إضافة قيمة إلى قيمة في نفس المتغير :

```
<?
$a = 3;
$a += 3;
print $a;
// يطبع 6
?>
```

-=

اطرح المقدار واحد من المقدار ثلاثة في المتغير \$a

```
<?
$a = 3;
$a -= 1;
print $a;
// يطبع 2
?>
```

**\*=**

يضرب القيمة 3 بالقيمة 2 ويكون الناتج مخزن في نفس المتغير :

```
<?
$a = 3;
$a *= 2;
print $a;
// يطبع الناتج 6
?>
```

**/=**

يقسم قيمة على قيمه أخرى :

```
<?
$a = 6;
$a /= 2;
print $a;
// يطبع ناتج القسمة 3
?>
```

**.=**

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 اضع واحد إليها  
++\$a اضع 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;
// يطبع القيمة 6
?>
```

**++value**

يرجع القيمة نفسها وفي استخدام ثاني تزيد القيمة واحد :

```
<?
$a = 5;
print $a++;
// طباعة الرقم 6
print "<br>";
print $a;
// طباعة الرقم 5
?>
```

## value--

يطرح من القيمة واحد :

```
<?
$a = 5;
print --$a;
// يطبع الرقم 4
?>
```

## --value

يرجع القيمة نفسها وفي استخدام ثاني يطرح منها واحد :

```
<?
$a = 5;
print $a--;
// يطبع الرقم 4
print "<br>";
print $a;
// يطبع الرقم 5
?>
```

## - عمليات المقارنة Comparison Operators :

a == b\$ المتغيران متساويان ..  
 a === b\$ المتغيران متساويان و من نفس النوع ..  
 b\$ != a\$ المتغير الاول لا يساوي الثاني ..  
 a !== b\$ المتغير الاول لا يساوي الثاني وليس من نفس النوع ..  
 b\$ < a\$ أكبر من ..  
 b\$ > a\$ أصغر من ..  
 b\$ <= a\$ أكبر من او يساوي ..  
 b\$ >= a\$ أصغر من او يساوي ..

## == (تساوي)

تساوي القيمة المخزنة في المتغير الأول بالقيمة المخزنة في المتغير الثاني :

```
<?
$x = 7;
$y = "7";
" . $y; if ($x == $y) print $x . "
// يطبع 7 تساوي 7
?>
```

## === (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 يطبع 7
?>
```

## != (لا تساوي)

إذا كانت القيم المخزنة في المتغيرين غير متساويين :

```
<?
$x = 8;
$y = 4;
if ($x != $y) print $x . " تساوي لا " . $y;
```

```
// يطبع 8 لا تساوي 4  
>?
```

**!= ( لا تساوي ولا من نفس النوع )**

إذا كانت القيم المخزنة في المتغيرين غير متساويين وليست من نفس النوع :

```
<?  
$x = 8;  
$y = 9;  
if ($x != $y) print $x . " من نفس نوع ليست!  
// يطبع 8 ليست من نفس نوع 9  
>?
```

**> ( أقل من )**

مقارنة بين قيمتين واحدة أقل من الأخرى :

```
<?  
$x = 5;  
$y = 9;  
if ($x < $y) print $x . " من أقل  
// يطبع 5 أقل من 9  
>?
```

**< ( أكبر من )**

مقارنة بين قيمتين واحدة أكبر من الأخرى :

```
<?  
$x = 9 ;  
$y = 5;  
if ($x > $y) print $x . " من أكبر  
// يطبع 9 أكبر من 5  
>?
```

**=> ( أقل من ويساوي )**

مقارنة بين قيمتين واحدة أقل أو مساوية لها :

```
<?  
$x = 5;  
$y = 5;  
if ($x <= $y) print $x;  
// يطبع القيمة 5  
>?
```

**=< ( أكبر من ويساوي )**

مقارنة بين قيمتين واحدة أكبر أو مساوية لها :

```
<?  
$x = 7;  
$y = 5;  
if ($x >= $y) print $x;  
// يطبع القيمة 7  
>?
```

## 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 الاثنین \$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 ال ( and ) بالشكل التالي ( & ) وال ( or ) بالشكل التالي ( | ) وال ( Xor ) بالشكل

التالي ( ^ ) ..

### And ( و )

إذا تحقق الشرطان ، بمعنى المتغير الأول يساوي 7 والمتغير الثاني يساوي 5 نفذ أمر الطباعة واطبع صحيح :

```
<?
$x = 7;
$y = 5;
if (($x == 7) and ($y == 5)) print "صحيح";
// 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";
// True يطبع
?>
```

### 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";
لأن المتغير القيمة المخزنة فيه غير صحيحة True يطبع
?>
```

### &&

المعامل && له نفس وظيفة ( and ) لكن الاختلاف في ترتيب تنفيذ أولويات العمليات :

```
<?
$x = 7;
$y = 5;
if (($x == 7) && ($y == 5)) print "True";
// True يطبع
?>
```

### ||

المعامل || له نفس وظيفة ( or ) لكن الاختلاف في ترتيب تنفيذ أولويات العمليات :

```
<?
$x = 7;
$y = 5;
if (($x == 7) || ($y == 5)) print "True";
// True يطبع
?>
```



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,  
string password);
```

هذه الدالة تقوم بما تقوم به الدالة 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:

```
<?  
// البيانات الاتصال بقاعدة ..  
$link = mysql_connect("localhost","mag","Pass");  
// اغلاق الاتصال بقاعدة البيانات ..  
mysql_close($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 مرور لها اسم قاعدة البيانات ورقم الاتصال العائد من الدالة mysql\_connect أو من الدالة mysql\_pconnect ..  
مثال:

```
<?
// الإتصال بقاعدة بيانات اسمها az الباسورد حيث أن الفراغ هو
$link = mysql_pconnect("localhost", "az", "");
// انشاء قاعدة بيانات جديدة
if (!mysql_create_db($link, "mag"))
{
    print("الجديدة فشل إنشاء قاعدة البيانات");
    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 مؤشر 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 :

```
<?
// الإتصال بقاعدة بيانات اسمها az الباسورد حيث أن الفراغ هو
$dbLink = mysql_pconnect("localhost","az","");
// اختيار قاعدة البيانات Authors
```

```
mysql_select_db("Authers", $dbLink);
// اختيار جميع الحقول من الجدول
$query = "SELECT * FROM address";
$result = mysql_query($query, $dbLink);
// الانتقال الى الحقل الثاني اعتماداً على عملية الاختيار
mysql_field_seek($result, 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 :

```
<?
// الإتصال بقاعدة بيانات اسمها az الباسورد حيث أن الفراغ هو
$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 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
unique_key	تكون TRUE إذا كان الحقل يمثل مفتاحاً ثانوياً
zerofill	تكون TRUE إذا كان 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 المعطي result.

```
<?
//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 وصل 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 :

```
<?
mysql_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 فقدتهما يستعاض عنهما بالاتصال الحالي. هذه الدالة تتطلب إصدار MySQL 3.23.3 أو ما بعدها.

بسم الله الرحمن الرحيم والصلاة والسلام على اشرف الانبياء والمرسلين ، في هذا الدرس احببت أن اجيب على تساؤل كثيراً ما يطرح في المنتديات وهو عن التاريخ باللغة العربية ، واتمنى من الله التوفيق في طرح الموضوع بشكل مبسط وفي متناول الجميع ..

في البداية ومقدمة للموضوع أود الإشارة الا أن الطريقة المشروحة في هذا الدرس تعتمد على التاريخ الميلادي ، وبالتأكيد يمكن استخدامها مع التاريخ الهجري ولكن تحتاج الى تعديل وحسابات خاصة ، وبإذن الله سيتم شرحها في المستقبل .

دالة التاريخ في البي اتش بي هي Date ، ولها معاملين ( أي قيمتين لاعداد مخرجات الدالة ) ، احد المعاملين اجباري والثاني اختياري ، اما الاول وهو الاهم تعتمد عليه مخرجات التاريخ بشكل اساسي مثل ضبط السنة بخانتين أو ضبط الشهر باسم الشهر .. وغيرها ، اما المعامل الثاني فهو ما يسمى بـ ( UNIX time stamp ) وهو خاص بنظام اليونكس وكيفية تخزين التاريخ فيه ، عموماً ما يهمنا هنا هو المعامل الأول وهو ما يسمى بـ ( Format String ) ، وكمثال على ما ذكرنا :

```
<?
$today = date(Y-m-d);
echo $today;
?>
```

هذا المثال سيقوم بطباعة تاريخ اليوم على الشكل التالي 13-03-2002 ، ولاهمية الرموز التي يمكن استخدامها مع الـ Date سأذكر أهمها :

d رقم اليوم في الشهر على شكل خانتين من 01 الى 31 .  
D اسم اليوم في الاسبوع على شكل 3 خانات مثل Mon أي الاثنين .  
g رقم الساعة في اليوم من 1 الى 12 .  
j رقم اليوم في الشهر من 1 الى 31 بدون وضع الصفر .  
m رقم الشهر في السنة على شكل خانتين من 01 الى 12 .  
y رقم السنة على شكل خانتين ، مثلاً 02 .  
Y رقم السنة على شكل اربع خانات ، ومثالها 2002 .

هذه من اهم الرموز لكي تتضح الصورة فقط ، ولعلنا نتطرق لها بشكل اوسع قريباً .

لتحويل التاريخ الى اللغة العربية نحتاج أن ننشئ جدولاً في قاعدة البيانات ، فلذلك قم بنسخ الكود التالي والصقه في خانة Run SQL query في الـ PHPMyadmin أو بأي طريقة اخرى تراها ، الاهم انشاء الجدول .

```
CREATE TABLE month_name (
id tinyint(4) NOT NULL default '0',
month text NOT NULL
) TYPE=MyISAM;

INSERT INTO month_name VALUES (1, 'يناير');
INSERT INTO month_name VALUES (2, 'فبراير');
INSERT INTO month_name VALUES (3, 'مارس');
INSERT INTO month_name VALUES (4, 'ابريل');
INSERT INTO month_name VALUES (5, 'مايو');
INSERT INTO month_name VALUES (6, 'يونيو');
INSERT INTO month_name VALUES (7, 'يوليو');
INSERT INTO month_name VALUES (8, 'اغسطس');
INSERT INTO month_name VALUES (9, 'سبتمبر');
INSERT INTO month_name VALUES (10, 'اكتوبر');
INSERT INTO month_name VALUES (11, 'نوفمبر');
INSERT INTO month_name VALUES (12, 'ديسمبر');
```

بعد انشاء هذا الجدول يجب أن يكون لديك جدول اخر يحتوي على التاريخ المراد تحويله ، ولنفترض أن لديك الجدول ( news ) يحتوي على الحقول ( title , date ) ويحتوي على البيانات التالية :

date title

20-04-2002 الخبر الأول

25-04-2002 الخبر الثاني

01-05-2002 الخبر الثالث

قم بانشاء الجدول :

```
CREATE TABLE news (  
  title text NOT NULL,  
  date date NOT NULL default '0000-00-00'  
) TYPE=MyISAM;
```

```
INSERT INTO news VALUES ('20-04-2002','الخبر الأول');  
INSERT INTO news VALUES ('25-04-2002','الخبر الثاني');  
INSERT INTO news VALUES ('01-05-2002','الخبر الثالث');
```

بقي أن نقوم بتحويل التاريخ الى العربية ، وإدراجه في صفحة PHP ، ولعمل ذلك سنقوم باستخدام دالة تسمى Date\_Format من خلال طلب لقاعدة البيانات ، نحدد من خلاله طريقة جلب البيانات ووضعها بالصورة المطلوبة .

بقي أن نذكر أننا سوف نضطر الى كتابة طلبين لقاعدة البيانات احدهما لجلب حقول العنوان ( title ) والاخر لجلب حقول التاريخ ( date ) كما يلي :

```
<?  
$result = mysql_query("select * from news");  
$sql = "SELECT CONCAT(DATE_FORMAT(date,'%d'),\"  
\",month_name.month,\" \",DATE_FORMAT(date,'%Y'))  
AS date FROM news ,month_name  
WHERE month_name.id = month(date)";  
$result2 = mysql_query("$sql");  
while ($row=mysql_fetch_array($result)  
and $row2=mysql_fetch_array($result2))  
{  
  $title = $row["title"];  
  $date = $row2["date"];  
  echo "$title , $date<br>";  
}  
?>
```

عند تنفيذ السكريبت ، ستري ما يلي :

الخبر الأول , 20 ابريل 2002

الخبر الثاني , 25 ابريل 2002

الخبر الثالث , 01 مايو 2002

في حالات كثيرة تكون كتابة السكريبت السابق بهذا الشكل مسببه للكثير من المشاكل ، و خاصة عند طلب ترتيب للجدول على حسب حقل معين ، وهذه المشاكل هي في توافق البيانات مع بعضها البعض ، فلو افترضنا في مثالنا السابق أن الخبر الأول الذي يحمل التاريخ 20-04-2002 كان باسم آخر ، مثلا ( العنوان الأول ) ، وبعد اضافة حقول ترتيب لجلب البيانات كالتالي :

```
<?  
$result = mysql_query("select * from news  
order by title");  
$sql = "SELECT CONCAT(DATE_FORMAT(date,'%d'),\"  
\",month_name.month,\" \",DATE_FORMAT(date,'%Y'))  
AS date FROM news ,month_name  
WHERE month_name.id = month(date)";  
$result2 = mysql_query("$sql");  
while ($row=mysql_fetch_array($result)
```

```

and $row2=mysql_fetch_array($result2))
{
$title = $row["title"];
$date = $row2["date"];
echo "$title , $date<br>";
}
?>

```

ستجد أن النتائج هي :  
 الخبر الثالث , 20 ابريل 2002  
 الخبر الثاني , 25 ابريل 2002  
 العنوان الأول , 01 مايو 2002 وهذا بطبيعة الحال مشكلة في توافق البيانات .

ولحلها يجب أن نوافق بين الطلبين لقاعدة البيانات ، بمعنى أنه إذا رتبنا الطلب الاول حسب ( title ) يجب أن نفعل ذلك مع الطلب الثاني بتعديله ليصبح :

```

<?
$sql = "SELECT CONCAT(DATE_FORMAT(date,'%d'),\"
\",month_name.month,\" \",DATE_FORMAT(date,'%Y'))
AS date FROM news ,month_name
WHERE month_name.id = month(date)
order by title";
?>

```

وبالتالي تصبح البيانات المخرجه كالتالي :  
 الخبر الثالث , 01 مايو 2002  
 الخبر الثاني , 25 ابريل 2002  
 العنوان الأول , 20 ابريل 2002

وهي بالتأكيد صحيحة .



بسم الله الرحمن الرحيم والصلاة والسلام على اشرف الانبياء والمرسلين ، بعد المقدمات السابقة والهامة في معرفة اساسيات اللغة يمكننا بداية كتابة البرامج بلغة البي اتش بي ، وبطبيعة الحال سنبدأ من اصغر الاساسيات واهمها في كتابة البرامج عموماً وهي المتغيرات .

المتغيرات في لغة الـ PHP تبدأ بعلامة الدولار ( \$ ) ، ولإسناد قيمة لذلك المتغير نستخدم علامة المساواة ( = ) ، فرضاً لدينا المتغير ( Name ) والقيمة ( Khaled ) فنكتب ما يلي :

```
<?
$name = "Khaled";
?>
```

هذا في حالة المتغيرات النصية ( Text ) ، وفي حالة المتغيرات الرقمية ( Numbers ) يمكن تعريف متغير ( Counter ) الذي يحمل القيمة ( 17 ) كالتالي :

```
<?
$counter = 17;
?>
```

الفرق الواضح في طريقة تعريف المتغيرين النصي والرقمي هو عدم وجود علامات التنصيص في تعريف المتغيرات الرقمية بينما يجب وضع علامات التنصيص في تعريف المتغيرات النصية .

### نقاط هامة في تسمية المتغيرات :

- أسماء المتغيرات في كثير من لغات البرمجة لا تتعدى 255 حرف ( المقصود بها الخانات سواء كانت حروف أو أرقام أو علامات أخرى ) ، و في لغة الـ PHP لا يوجد حدود على عدد الخانات في تسمية المتغيرات ، ولكن في الغالب لن تحتاج الى أكثر من 15 خانة لتسمية أي متغير ، لان المبالغة في تسمية المتغيرات تسبب مشاكل في تذكر المتغيرات وما تحتوية من قيم .

- بداية كل متغير يجب أن تبدأ بحرف ( يعني حرف هجائي ) أو علامة ( \_ ) Underscore ، مع تجاهل علامة الـ \$ لانها لا تحسب من اسم المتغير .

- يمكن أن يحتوي اسم المتغير على الحروف أو الأرقام أو علامة ( \_ ) فقط ، اما العلامات الأخرى مثل ( + , - , \* , / ) أو الـ & لا يمكن كتابتها في اسم المتغير .

- المتغير ( \$Name ) يختلف عن المتغير ( \$name ) لاختلاف حالة حرف الـ N ، ولذلك يجب التأكد من اسم المتغيرات بدقة لتجنب حدوث مشاكل في الوصول الى متغير معين ، وبالتأكيد لو كان لديك اسلوب خاص في تسمية المتغيرات لسهولة الوصول اليها وتذكرها ستكون كتابة السكريبت اسهل بكثير .

- يستحسن أن تكون أسماء المتغيرات دالة على معانيها ، بمعنى أنه لمتغير مثل عداد الزوار يستحسن أن يكون ( \$counter ) ، ولمتغير مثل اسم المستخدم ( \$user ) .. الخ .

### التعامل مع المتغيرات :

فائدة المتغيرات تكمن في طريقة استخدامها في كتابة السكريبت ، وكما ذكرنا سابقاً أنه لطباعة متغير معين نستخدم امر الطباعة ( echo ) أو ( print ) كما يلي :

```
<?
$name = "Naser";
echo $name;
?>
```

في البداية سيتم إسناد القيمة ( Naser ) الى المتغير ( \$name ) ، وفي السطر الثاني يتم طباعة المتغير ، أو بالأحرى القيمة المسندة الى المتغير .

### انواع البيانات ( Data Types ) :

في الامثلة السابقة قمنا بإسناد قيمتين عددية ونصية الى متغيرين ، وبيناً الفرق بينهما ، وفي لغة الـ PHP بشكل عام يوجد أكثر من هذين النوعين من البيانات ، سأشرح بعضاً منها الآن ، والبقية في الدروس القادمة :

- البيانات النصية ( String ) .

- البيانات العددية الصحيحة ( Integer ) .

- البيانات العددية الكسرية ( Double ) .
- المصفوفات ( Array ) .
- الكائنات ( Object ) .
- البيانات الغير معروفة ! .

### البيانات النصية ( String ) :

هي البيانات التي تكون بين علامات التنصيص " " بغض النظر عن محتواها ، فيمكن أن تكون حروف أو اعداد أو رموز أو غيرها ، ومثال ذلك كما ذكرنا سابقاً :

```
<?
$user = "Khaled";
$age = "13.5";
?>
```

### التعامل مع البيانات النصية ( String ) :

لاضافة المتغيرات التي تحتوي على بيانات نصية مع متغيرات من نفس النوع نحتاج الى عملية دمج بين المتغيرات ، ولعمل ذلك نكتب :

```
<?
$total = $user . $age;
?>
```

في هذه الحالة سيتم اسناد القيمة Khaled13.5 الى المتغير ( \$total ) .  
إذا أردنا وضع مسافة بين المتغيرين نضيف متغير جديد يحتوي على المسافة وهو ( \$space ) ثم نقوم بعملية الدمج كالتالي :

```
<?
$space = " ";
$total = $user . $space . $age;
?>
```

وفي هذه الحالة سيتم وضع القيمة Khaled 13.5 في المتغير ( \$total ) ، وبطبيعة الحال يمكن استخدام المتغيرات النصية داخل متغيرات نصية أخرى ، حيث سيتم تعويض المتغير بقيمته الأصلية .

### البيانات العددية ( Numeric ) :

وكما ذكرنا في التقسيم السابق أنها نوعين ( الاعداد الصحيحة Integer ) و ( الاعداد الكسرية Double ) ، وكمثال على النوعين :

```
<?
$integer1 = 233;
$integer2 = -29
$double1 = 5.27
$double2 = -4.6
?>
```

### التعامل مع البيانات العددية ( Numeric ) :

العمليات الحسابية المشهورة ( + , - , \* , / ) بالاضافة الى باقي القسمة ( % ) عمليات شائعة جداً في التعامل مع المتغيرات العددية ، وبطبيعة الحال لن نحتاج الى ذكر أي مثال عن هذه العمليات ، وسنكتفي بذكر بعض النقاط الاساسية التي قل ما يخلو سكرت منها .

اول النقاط هي اضافة المتغير الى نفسه ، بمعنى تعريف عملية حسابية على متغير معين بحيث تخزن القيمة في نفس المتغير ، مثلاً لو كان لديك عدد الزوار وتريد في كل مرة أن يزود عدد الزوار بـ 1 ، يمكنك كتابة ما يلي :

```
<?
$counter = $counter + 1;
?>
```

بالتالي سيتم زيادة المتغير ( \$counter ) بـ 1 في كل مرة يتم فيها تنفيذ السكرت ، وبطريقة أخرى يمكن كتابة السطر السابق كالتالي :

```
<?
$ccounter = $counter++;
?>
```

وال ++ تعني زيادة قدرها ( 1 ) على قيمة المتغير الأصلية ، وكذلك ال -- تعني طرح 1 من القيمة الأصلية .

وفي حالة الرغبة بزيادة أي عدد آخر ( غير الواحد ) على أي متغير بأسلوب الطريقة الثانية يمكن كتابة ما يلي :

```
<?
$counter +=4;
?>
```

وهذا يعني زيادة مقدارها 4 على قيمة المتغير الأصلية ، وبالسالب كذلك بنفس الأسلوب .

### ترتيب انجاز العمليات الحسابية :

يوجد بعض الرموز والعمليات التي تسبق غيرها عند البدء في انجاز عملية حسابية معينة ، والترتيب المستخدم في البي اتش بي كالتالي :

```
] -
- ! ~ ++ -- ( object ) ( array ) ( string ) ( double ) ( int )
% / * -
. - + -
< < > > -
> => < =< -
== != === !== -
& -
| -
&& -
|| -
?: -
= += *= /= =. =% =& ^= ~ == >> =<< -
print -
AND -
XOR -
OR -
,-
```

بالتأكيد القائمة طويلة وفيها تفاصيل كثيرة ، ولكن من المهم معرفة طريقة انجاز العمليات الحسابية المختلفة لسهولة اكتشاف الأخطاء ومعرفة الطريقة الصحيحة لكتابة بعض العمليات المعقدة للحصول على ناتج صحيح .

### بعض الدوال الهامة في التعامل مع المتغيرات :

- isset : وهي دالة للتأكد من وجود متغير معين ، فمثلا :

```
<?
echo isset($age);
?>
```

سيتم طباعة الرقم 1 اذا كان المتغير ( \$age ) موجوداً ( تم انشاءه مسبقاً ) ، والعكس اذا كان غير موجود سيتم طباعة الرقم 0 ، وهذه الدالة يتم استخدامها كثيراً في الشروط وهذا ما سنتطرق اليه لاحقاً .

- unset : هذه الدالة تعمل على مسح المتغير من الذاكرة كلياً ، فقط قم بعمل التالي :

```
<?
unset($age);
?>
```

وفي هذه الحالة سيتم مسح المتغير ( \$age ) بشكل كامل .

- empty : وهذه الدالة معاكسة للدالة isset بحيث لو كتبنا ما يلي :

```
<?
echo empty($age);
?>
```

سيتم طباعة الرقم 1 في حالة عدم وجود المتغير ( \$age ) أو أن قيمة المتغير تساوي 0 أو ( فراغ ) ، وفي حالة وجود المتغير ( \$age ) لن يتم طباعة أي شيء .

بسم الله الرحمن الرحيم والصلاة والسلام على اشرف الانبياء والمرسلين اما بعد ، في هذا الدرس بمشيئة الله سنتحدث عن مقدمة للتحكم في المواقع عن طريق ال Session أو الجلسات كما اصطلح على تسميتها ، ففي البداية سنتعرف على ال Session وعن التحكم فيها ، ومن ثم استخداماتها بالإضافة الى بعض الامثلة ، وفي النهاية سنتطرق الى بعض الأخطاء في كتابة ال Session وحلول تلك الأخطاء ، وفي الدرس القادم بإذن الله تعالى سنتطرق الى مثال كامل للوحة تحكم مبسطة تتعامل بال Session ، والأمل أن يكون في هذا الشرح المبسط فائدة للجميع ..

### - مقدمة عن ال Session :

عند الانتقال من صفحة الى أخرى في موقع معين فإن بروتوكول ال HTTP لا يمكنه معرفة أن تلك الصفحات قد تم تصفحها من قبل نفس الشخص ، ولكن مع ال cookies وما نحن بصدده هنا ال Session تقدم تلك الطريقة ، ولذلك وببساطة فإن ال Session هي مكان على جهاز المتصفح يمكن من خلاله تخزين قيمة معينة للرجوع اليها في حال قام نفس الشخص بالانتقال من صفحة الى أخرى ، ولعل هذا التعريف يصف ببساطة معناها العام ولا يعني ذلك أنه تعريف شامل لكل المعاني ..

إذاً التعرف على الشخص الذي يقوم بتصفح الموقع هو الهدف الرئيسي لل Session أو الجلسات ، ولكن كيف يتم ذلك ، وما هي النقاط الرئيسية التي يجب معرفتها لفهم طريقة التعامل مع ال Session ؟

أول تلك النقاط أن عملية تسجيل المتغير على جهاز المستخدم له مدة معينة تنتهي بانتهاء الجلسة ، ومن هنا جاءت التسمية ، أما ما تعنيه الجلسة فهي مصطلح لقيامك بالتصفح من الموقع ومن ثم اغلاق الموقع ، ببساطة كل مرة تقوم بزيارة الموقع تبدأ جلسة أو Session جديدة ، مع ملاحظة أن هناك طرق للتحكم بوقت الانتهاء كما في ال cookies ، بالإضافة الى طرق أخرى عن طريق قواعد البيانات وهو حديث سابق لاوانه .

بالنسبة للنقطة الأخرى التي يجب وضعها في الحسبان هي ما يسمى بال Session ID أو اختصاراً SID ويعني ذلك ( رقم الجلسة ) ، وهو رقم عشوائي فريد يصعب تكراره أو فلنقل أنه مستحيل لاحتوائه على ارقام واحرف كبيرة وصغيرة في متغير طويل نسبياً ، وهذه القيمة هي الأهم في ما ذكرت ، لإنها القيمة الوحيدة التي تربط ما يسمى بال Session Variables أو ( متغيرات الجلسة ) مع جهاز المستخدم ، فال SID هي القيمة الوحيدة التي يتم تخزينها في جهاز المستخدم ( Client ) ، أما ال متغيرات الجلسة Session Variables يتم تخزينها في السيرفر ( Server ) ، فعند التحقق منه وجود هذه القيمة على جهاز المستخدم يمكن الدخول الى المتغير الآخر المترابط به والمسمى بال Session Variable .

النقطة الثالثة هي طريقة التخزين لل SID و ال Session Variables ، أما ال SID وكما قلنا أنها تخزن على جهاز العميل ( Client ) إما عن طريق ال cookies والتي لها سلبياتها المتعددة أو عن طريق تمريرها عبر ال HTTP ، أما بالنسبة لل Session Variables فيتم تخزينها في ملفات فارغة على جهاز ال Server وكذلك في مستويات متقدمة يمكن التحكم بها وتخزينها في قواعد بيانات .

### - إعدادات ال Session :

عن طريق ملف ال php.ini والذي يحتوي على إعدادات ال PHP يمكن التحكم باعدادات ال Session ، وكاستعراض لتلك النقاط المتحكم بها ال Session سنتطرق أهم النقاط ومعانيها ، وللوصول الى ما نحن بصدده نذكر أن ملف ال php.ini يوجد في دليل ال Windows ، وللوصول الى خصائص ال Session إبحث عنه كلمة ( Session ) وستجد السطر التالي :

#### [Session]

من هنا تستطيع التحكم بخيارات ال Sessions ، وكما يظهر في الجدول التالي وصف لأهم الخيارات ..

#### -1 الخيار Session.auto\_start :

بداية تلقائية لل Session ( دون الحاجة لعمل ذلك يدوياً عن طريق Session\_start ) .

#### -2 الخيار Session.cache\_expire :

وقت انتهاء الجلسة بالدقائق .

#### -3 الخيار Session.cookie\_lifetime :

وقت انتهاء ال cookie المرتبطة بالجلسة ، وهي افتراضياً ستكون 0 أي أن ال cookie ستنتهي فترتها مع

اقفال الشخص المتصفح للموقع .

#### -4 الخيار **Session\_name** :

إسم ال Session التي ستستخدم ك cookie وافترضياً ستكون PHPSESSID .

#### -5 الخيار **session.save\_path** :

هذا السطر يعني مكان تخزين ملفات ال Session في جهازك باعتباره سيرفر ، وهنا تستطيع أن تضع أي عنوان في جهازك ، أما تركه فارغاً فيعني عدم تفعيل ال Session لديك ، بالنسبة لي أقترح أن يكون المجلد Temp داخل ال Windows هو الدليل الأمثل لاحتوائه على ملفات مؤقتة يمكن حذفها ، اذا العنوان سيكون c:\windows\Temp

هذه في نظري أهم الخيارات التي يجب فهمها ،

#### - بداية ال Session :

قبل أن تستخدم أيّاً من دوال ال Session يجب اخبار السكريبت أن يبدأ جلسة Session ، والطريقة هي أن تضع في بداية السكريبت وفي أول سطر فيه بعد علامة الفتح ما يلي :

```
<?
session_start();
?>
```

في هذه الحالة فقط يمكن أن تقوم باستخدام دوال ال Session الأخرى ، أما اذا لم يتم كتابة هذا السطر فلن يتم ذلك .

ملاحظة مهمة حول عملية بداية ال Session وهي أن تتأكد من أن هذا السطر لا يسبقه عملية اخراج مخرجات ، بمعنى أخرى أي استخدام لدوال مثل echo أو print ، وكذلك لا يسبق هذا السطر أي فراغ وتأكد من هذه النقطة جيداً لأنها كثيرة الحدوث وتعطى الخطأ التالي :

وأسلم طريقة من وجهة نظري أن تضع هذا السطر في بداية ملف ال header لانك سنقوم بادراج هذه الصفحة في كل الصفحات الأخرى وبالتالي يكون السطر هو الأول في كل الحالات ..

#### - تخزين متغيرات الجلسات :

وهي ما نسميها بال Session Variables ، ولعمل ذلك يوجد لدينا الدالة الواردة في المثال التالي :

```
<?
$user = "AbdulAziz";
session_register("user");
?>
```

ما قمنا بعمله هو التالي :

- 1- عرفنا متغيراً هو **user** يحتوي على قيمة حرفية .
- 2- قمنا بتسجيل هذا المتغير في متغير جلسة ( Session Variable ) وبنفس الاسم **user** ولكن بدون علامة \$ .

#### - التعامل مع متغيرات الجلسة :

بعد تسجيل المتغير ، يمكن الرجوع اليه بعدة طرق تعتمد على الخيار **register\_globals** في ملف ال **php.ini** ، أما اذا كان **on** وهذا هو الاختيار الافتراضي فإن المتغير الذي تم تسجيله في ال Session يمكن الرجوع اليه كأي متغير آخر ، عن طريق اسم المتغير فقط ، وفي مثالنا الحالي سيكون **\$user** ، أما اذا كان الخيار غير مفعل وليس بالصورة التي ذكرتها فيمكن الرجوع الى المتغير عن طريق **\$\_SESSION["user"]** .

أيضاً كنقطة مهمة يجب معرفتها وهي طريقة التحقق من أن متغيراً معيناً قد تم تسجيله أم لا ، وهذه الطريقة مفيدة في الصفحات التي يجب أن يكون فيها المستخدم قد سجل الدخول وبالفعل تمت عملية تسجيل ال Session له ، في المثال التالي تلك الطريقة :

```
<?
if (session_is_registered("user")) {
echo "أهلاً وسهلاً بكم في قرية بي اتش بي";
}
else {
echo "لا يسمح لك بالدخول";
}
?>
```

في هذا المثال سيتم عرض الجملة (أهلاً وسهلاً بكم في قرية بي اتش بي) إذا كان عملية تسجيل ال Session تمت للمتغير **user** ، وسيتم عرض الجملة (لا يسمح لك بالدخول ..) في حالة عدم تسجيل ال Session .

نقطة أخيرة في التعامل مع متغيرات الجلسة ، وهي عملية الغاء تسجيل ال Session لمتغير معين ، وهذه الطريقة تتم عن طريق الدوال **session\_unregister** و **session\_unset** و **session\_destroy** ، أما الفرق بينهم فهو أن الدالة الأولى تقوم بعملية الغاء التسجيل ل Session معينة ، أي بتمرير إسم المتغير لها كما في المثال التالي :

```
<?
session_unregister("user");
?>
```

إذا سيتم الغاء تسجيل ال Session المتعلقة بالمتغير **user** فقط ، أما الدالة الثانية فستقوم بالغاء تسجيل جميع ال Session التي تم تسجيلها من قبل ، وفي النهاية يجب استخدام الدالة الثالثة **session\_destroy** لالغاء ال SID والانهاء من التعامل مع ال Session .

- مثال بسيط عن ال Session :  
سأنتقل الى مثال بسيط جداً لتوضيح كيفية عمل ال Session ، في البداية قم بوضع الكود التالي في ملف وسم بتسميته `phpex1.php` :

```
<?
$age = 12;
session_register("age");
echo "<br>الجلسة age تحتوي على القيمة $age";
echo "<a href=phpex2.php>التالي ..</a>";
?>
```

الصفحة الثانية احفظها بإسم `phpex2.php` ، وضع الكود التالي فيها :

```
<?
echo "<br>أنت في الصفحة الثانية";
echo "<br>الجلسة age تحتوي على القيمة $age";
session_unregister("age");
echo "<a href=phpex3.php>التالي ..</a>";
?>
```

الصفحة الثالثة تحتوي على الكود التالي ، واسمها `phpex3.php` :

```
<?
echo "<br>أنت في الصفحة الثالثة";
echo "<br>الجلسة age تحتوي على القيمة $age";
?>
```

ابداً من الصفحة الأولى ومن ثم انتقل من صفحة الى أخرى ، حتى تصل الى الثالثة ، بافتراض أنك قمت بتجربة المثال ، ستلاحظ أن الصفحة الأولى سيتم طباعة ال Session التي تم تسجيلها وهي **age** وستظهر القيمة **12** في الجملة الطويلة التي تبين أن المتغير **age** يحتوي على قيمة معينة ، وفي الصفحة الثانية ستلاحظ نفس الجملة ونفس القيمة تمت طباعتها ، أما في الصفحة الثالثة والأخيرة فتمت طباعة الجملة ، لكن الاختلاف أن القيمة **12** في متغير ال Session **age** لم تتم طباعتها ، لماذا ؟

لسبب بسيط وهو أننا في الصفحة السابقة قمنا بالغاء تسجيل ال Session للمتغير **age** وبالتالي فإن الصفحة الثالثة لم تتعرف على متغير مباشر له الاسم **age** ولا على متغير ال Session **age** ، وبالتالي تم طباعة الجملة بدون القيمة .

## التعامل مع الملفات والمجلدات

كل مبرمج يجب أن يتعامل مع الملفات والمجلدات في بعض النقاط ، برنامجك سوف يستخدم الملفات لكي يقوم بتخزين معلومات الإعدادات للسكربت ، أو يقوم بتخزين البيانات لقراءتها وكتابتها ، أو لكي يقوم بحفظ البيانات المؤقتة ، وكمثال فإن أطفه برنامج عدد 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 مادة تخزينية أخرى .

والمجلد : هو عبارة عن نوع محدد من الملفات يحتفظ بأسماء ملفات أخرى ومجلدات أخرى ( تسمى بالمجلدات الفرعية ) ، كل ما تحتاجه للتعامل 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 وتحويل في الملف .
- 5 فتح وإغلاق المجلدات .
- 6 نسخ وإعادة تسمية 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 كالتالي

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

C:\\windows\\PHP

## التعامل مع الملفات

يوفر الـ PHP 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 مباشرة ...

مقبض الملف ليس أكثر من عدد صحيح (integer) يقوم بتعريف الملف المراد فتحه حتي يتم إغلاقه ، إذا كان هناك أكثر من ملف مفتوح فإن لكل ملف مقبضه التعريف 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 أغلب الدوال أو معظمها أو كلها تقريبا تقوم بإرجاع القيمة True إذا تمت بنجاح والقيمة False إذا فشلت في الحصول على هدفها ..  
لنبدأ الآن مع سكريبتات مبسطة للعمل مع الملفات ..

## فتح واغلاق الملفات

### Fopen

تستخدم هذه الدالة ثلاث عوامل هي مسار الملف (path) والوضع له ( للقراءة ، للكتابة ..... ) بالإضافة إلى مسار Include 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 قراءة 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 مجلدًا قمت بإنشائه في مجلد السكريبتات الرئيسي لديك الذي يسمىhtdocs وأسميته data

ولنفرض أن سكريبتك يستخدم ملفين :

- 1- ملف للقراءة والكتابة يسمى data.txt .
- 2- وملف يقوم بعرض المدخلات والإضافة إليها اسمه script.txt .

حسنًا 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 data 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 المجلدhtdocs بينما السكريبت موجود في المجلد data الموجود داخلhtdocs على ذلك نكتب المسار النسبي كالتالي

```
$fp=fopen("../data.txt", "r");
```

لاحظ النقطة 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 نقطة 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 للثنتين معاً أو للإضافة ، وأرتبها هنا في جدول بسيط ..

الوصف	القيمة
تفتح الملف للقراءة فقط ويكون المؤشر في بداية الملف	r
يفتح الملف للقراءة والكتابة ويضع المؤشر في بداية الملف	r+
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	w+
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	a+
يستخدم لفتح وقراءة ملفات الصور على نظام أو سيرفرات الويندوز فقط .. أما النيوكس فالعوامل السابقة تتعامل مع ملفات الصور بشكل عادي ..	b

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 الـ PHP عن الملف في نفس المجلد الموجود به السكريبت 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 انتهى عمله فإن الـ PHP يقوم بإغلاق جميع الملفات تلقائياً تقوم الدالة fclose() بإغلاق الملف عندما تريد إغلاقه وهي تحتاج إلى معامل واحد فقط وهو مقبض الملف الذي تريد إغلاقه  
مثال :

```
Fclose ($fp) ;
```

## قراءة وكتابه الملفات

لقد تعرفنا الآن كيفية فتح وإغلاق الملف ، لنقم الآن بالتعرف على كيفية قراءة و كتابة البيانات من الملف ،

**Fread**

تقوم هذه الدالة بقراءة واستخراج البيانات الموجودة في الملفات ووضعها بمتغير وهي تأخذ عاملين 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 أخرى فسيتم قراءة العشر أحرف الثانية ..
- 2- إذا كان في الملف أقل من عشر أحرف فسيتم قراءة الموجود .

**Fwrite**

تقوم هذه الدالة بالكتابة إلى 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 حرفاً سنقوم بكتابته من القيمة الحرفية الموجودة في المعامل الثاني فلو مثلاً كتبنا

```
Fwrite ($fp, "PHP",1);
```

فسوف يتم كتابته 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 إلى نهاية الملف بعد استخدامك لأحد الدوال التي تقوم بنقل المؤشر من مكان إلى آخر ، عند ذلك ستكون هذه الدالة مفيدة لتخبرك إذا ما وصلت إلى نهاية الملف أو لا ...

## تطبيق عملي :

قم بإنشاء ملف اسمه 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 سنأخذ وقتاً وعمراً حتى يتم قراءتها ، يقوم الـ PHP 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 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 المحددة  
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 الدالة fwrite وتأخذ نفس معاملاتها ونفس طريقته ..

## القراءة داخل الملفات

## File

تحتاج هذه الدالة إلى معامل واحد هو مسار الملف ولا تحتاج إلى مقبض ، وعملها هو قراءة ما بداخل الملف وتخزينه سطرًا سطرًا 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 إلى مقبض للملف بل تحتاج إلى مسار الملف فقط ، وهي تقوم بفتح وقراءة وإغلاق الملف تلقائياً ...  
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 إلى 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 سابقاً بأن هناك طريقة تجعلك تتحكم في التحكم بالمؤشر الوهمي للملف والوصول إلى أي مكان في الملف أو عند أي حرف تريده ، بالدوال السابقة كنا عندما نصل إلى 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 عملية نهائياً .....  
يوصل لنا PHP بعض الدوال التي تمكننا من الوصول إلى 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 الـ PHP بالتحرك في الملف إلى أن يصل إلى المكان الذي تم تحديده .. أي أنه إذا كان في الملف سطر واحد مكون من عشرة حروف وقمنا بجعل الـ offset خمسة ، سيقوم الـ PHP 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 بالـ offset  
Seek\_cur يقوم بالقراءة من المكان الحالي حتى يصل إلى المكان المطلوب بالـ offset  
Seek\_End يقوم بالقراءة من نهاية الملف حتى يصل إلى المكان المحدد بالـ offset

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 الـ PHP وانتظر النتيجة :

```
<?
$fp = fopen("file.txt");
fseek($fp,4,SEEK_SET);
fpassthru($fp);
?>
```

### Ftell

هذه الدالة من الدوال المفيدة فهي تقوم بإرجاع مكان الـ offset ( أو المؤشر الوهمي ) في الملف وتحتاج إلى معامل واحد وهو مقبض الملف ...

```
<?
$fp = fopen ("file.txt");
$fp = ftell($fp);
```

```
echo $p;  
?>
```

## Rewind

تقوم بإرجاع المؤشر إلي بداية الملف ...

```
<?  
$fp = fopen ("file.txt");  
rewind($fp)  
?>
```

## جلب معلومات الملف

يوفر الـ PHP 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 طرأ 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 مباشرة ...

يقوم الـ PHP بتزويدنا بدالتين لمعرفة 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 تعريفهم عن طريق رقم (ID) مثلما يتم تعريفهم عبر أسمائهم ، إذا كنت تريد معرفة 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 الـ ID للمجموعة ... وسوف تحتوي على العناصر التالية :

Name	اسم المجموعة
Gid	رقم المجموعة
members	عدد أعضاء المجموعة

وهناك أيضا خمس دوال تساعدنا في معرفة معلومات أكثر عن الملفات وتحتاج فقط إلى 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 و dir ...

### 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 ، إعادة تسمية وحذف الملفات

تسمح لك الـ PHP بنسخ ، وإعادة تسمية ، وحذف والدوال التي تستخدم لتنفيذ هذه العمليات هي

### Copy ()

تقوم بأخذ قيمتين حرفيتين وتشير إلي مصدر الملف الرئيسي الذي يوجد فيه الملف والمصدر الهدف الذي سيتم نسخ الـ PHP 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 حرفيتين 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 الـ PHP 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 خالياً 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 الـ htdocs 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 نريده أن يتعامل معه 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 عملية rewinddir ..

### Closedir

تقوم بإغلاق المجلد ..

بهذا يكون انتهى الدرس  
قد تكون بعض النقاط غير واضحة ، في الواقع لن تحتاج إلي كل هذه الأمور في تعاملك مع الملفات

دعنا نأخذ فكرة عن طرف التخزين في البداية وكيف كانت على الإنترنت في السابق ....  
كان من أكثر طرق التخزين انتشاراً في السابق على الإنترنت وربما هو لا يزال يستخدم في بعض المواقع  
والمنتديات يعتمد على الملفات .... فكان صاحب الموقع الذي لديه هذه الطريقة في تخزين البيانات خوف  
وتعب من فقدانها مثلاً وكان الشبح الذي يكدر عليه صفوة نجاح موقعه هو عمل نسخ احتياطية لهذه  
الملفات لكي يتمكن من استرجعها في حال فقدانها فكانت هذه العملية تأخذ وقت وجهد ومال .... كما كان  
من عيوب تخزين البيانات في الجداول ضغط الخادم أو (server) في حال الاستعلام عن معلومة معينة  
والبحث عنها كما أنه يستهلك الكثير من ذاكرة هذا الخادم في عملية بحث معينة فهو يحجز مساحة  
ليست بالهينة في عملية بحث عن أسم مثلاً أو ما شابهها .

ربما يتردد عند البعض ذلك السؤال وهو .... ما هي قواعد البيانات بالضبط ؟

قواعد البيانات ببساطة جمع المعطيات أو المدخلات .  
كل قاعدة بيانات ربما تتكون من جدول (Table) واحد أو عدة جداول هذه الجداول تحتوي علي أعمدة  
وصفوف تهيكّل البيانات وترتيبها ,,,,,, وسوف أجعل لك مهمة اكتشاف فوائد قواعد البيانات في آخر الدرس .  
لنرى الجدول الذي بالأسفل كمثال :

### #Table "Coustomers"

Lname	Fname	Id
صالح	عبدالواهب	025
خالد	محمد	044

كما تلاحظ , الجدول قسم البيانات إلى صفوف ... مع كل إضافة عميل جديد سوف يكون هناك صف (سجل)  
جديد ... ربما لو تطلق لخيالك العنان سوف تلاحظ أن هذا الجدول مشابه للدولاب والصفوف رفوف فإذا أردت  
أن تضيف كتب أو ملابس أو أي كان سوف تضيفها في رف جديد .. كما يحصل في إضافة عميل جديد سوف  
تضيفه في صف (سجل) جديد .

البيانات في كل صف قسمت إلى مدى أبعد في الخلايا ( أو الحقول ) , كل من هذه البيانات تحتوي على  
قيمة محددة وصفة محددة , على سبيل المثال محمد خالد سوف ترى أن هذا العميل انقسمت بياناته في  
الحقل إلى id والاسم الأول والاسم الأخير .

الصفوف في الجدول ليس لها ترتيب معين .. يمكن أن يكون الترتيب أبجدياً ويمكن أن يكون باسم العضو أو  
باسمه الأخير أو بأي معيار آخر يمكن أن تحدده مسبقاً لترتيب الصفوف ولهذا يكون من الضروري تحديد  
طريقة ليسهل عليك تحديد صف (سجل) معين .... في المثال السابق نستطيع إخراج السجل من بين باقي  
السجلات بـ id وهو رقم العميل الذي هو عدد فريد لا يتكرر في أي صف (سجل) آخر وسبب استنادي في  
استخراج السجل علي id لأنه ربما يكون هناك عميلان لها نفس الاسم .... وهذا ليس شرط أن يكون  
للجدول مفتاح فريد لكن هنا حددته لكي يتم استخراج السجلات المطلوبة بسهولة وبسرعة اكبر .

## العلاقات

الكثير من قواعد البيانات اليوم هي نظم إدارة قواعد بيانات علائقية ( relational database management systems )  
تختصر في RDBMS , قواعد البيانات العلائقية هذه عبارة عن مجموعة من الجداول أو نموذج من  
الجداول النموذجية المتعددة التي تحتوي على معلومات مترابطة .  
ربما تسمع أيضاً الكثير عن SQL وهي اختصار لـ (Structured Query Language) وهي تسمح لك أن توحد  
هذه المعلومات من الجداول المترابطة وبذلك تسمح لك بإنشاء وتحليل العلاقات الجديدة .

المثال السابق للعملاء كان عبارة عن جدول واحد فقط , ولذلك لن تحتاج إلى ربط بينه وبين جدول آخر لأنه  
لا يجد .

لكن إذا كان هناك أكثر من جدول وكانت هذه الجداول مترابطة مع بعضها البعض في البيانات سوف تلاحظ أنك بحاجة إلى نظم إدارة البيانات العلائقية (RDBMS).... فلنرى هذه المثال لكي تتضح الصورة أكثر :

#Table "Coustomers "

Id	Fname	Lname
025	عبدالواهب	صالح
044	محمد	خالد
022	حمد	طارق

#Table "Address"

Id	Tel	Street	City	Country
044	018522	شارع الاهرام	القاهرة	مصر
022	01225505	طريق الملك فهد	الرياض	السعودية
025	0122505	طريق الاربعين	الكويت	الكويت

#Table "Account"

Id	accountb
044	10.0000
025	20.0000
022	20.000

كل من هذه الجداول الثلاثة كيان مستقل .... لكن تلاحظ أنهم مرتبطين مع بعضهم البعض ب (id) , على سبيل المثال بأمأكننا أن نعرف رصيد العميل عبد الواهب صالح من id , كما يمكننا معرفة اين يسكن حمد طارق وكم رقم التلفون وايضاً يمكننا أن نعرف من هو صاحب الرصيد 20.000 أيضاً كم واحد من مدينة القاهرة والكثير الكثير ..... ربما اتضح لك اهمية العلاقات .

إذاً عرفنا أن العلاقات هي الأساس الجوهرى لنظم قاعدة البيانات العلائقية , يجعلها مرنة وسهلة بحث تتمكن من ربط السجلات المختلفة مع بعضها البعض في الجداول .

## المفتاح الأجنبي

سوف تلاحظ أن حقل ( id ) الذي يظهر في الجداول في الثلاثة والذي جعل من الممكن ربط الجداول المختلفة معاً أنه مفتاح أجنبي لأنه بالأصل مفتاح فريد ( primary key ) في جدول ( COUSTOMERS ) ... ليس ضرورياً أن يكون هناك مفتاح أجنبي في كل جدول ولكن يتم إضافة على حسب حاجتك فإذا كنت تريد ربط بيانات الجداول مع بعضها فسوف تحتاج إليها.

في كل جدول يوجد به المفتاح الأجنبي سوف يكون له مرجعية للجدول الأصل فمثلاً هنا المرجعية ستكون جدول ( customers )....بمعنى أن المفتاح الأجنبي سوف يقوم بربط البيانات ما بين الجدول الأصل وبين الجدول الذي يتواجد به كمفتاح أجنبي ....من هنا يتضح لنا مفهوم الاستقامة المرجعية وهذا مفهوم أساسي ومهم عندما تصمم قاعدة بيانات بأكثر من جدول . سوف يكون للمفتاح الأجنبي قيمة ثابتة في جميع الجداول بمعنى لو كان قيمة المفتاح الأجنبي في جدول الأصل عدد صحيح فسوف يكون بنفس

القيمة في جميع الجداول,,,,, ونقطة أخرى إذا حدث تحديث أو تغيير أو حذف لا حد القيم في المفتاح الأجنبي فسوف تتم في جميع الجداول... هذا هو مفهوم الاستقامة المرجعية .

كثير من قواعد البيانات اليوم يتم تعديل القيم بها تلقائياً كمكر سوفت أكسس وبعض قواعد البيانات الأخرى , لكن هناك بعض قواعد البيانات التي تحتاج إلى تعديل يدوي على كل قيمة يتم التعديل عليه... وهذا لاشك انه متعب !!

## الفهرسة

لو كان لديك جدول به الكثير من السجلات , يمكنك أن تستعلم بسرعة كبيرة عن أيمن هذه السجلات بواسطة "فهرسة" كل السجلات . هذا المفهوم تقريباً شبيه جداً بالفهرس الذي يوجد نهاية كل كتاب... كما يسهل عليك هذا الفهرس الموجود في الكتاب في سرعة البحث عن المواضيع التي يتضمنها الكتاب , نفس الكلام ينطبق على فهرسة السجلات في الجدول .... دعنا نرى مثال لتتضح الصورة :

```
SELECT * FROM names WHERE ID = 220;
```

سوف يقوم هذا الاستعلام في البحث في جميع المعلومات وإرجاع قيمتها بشرط أن يكون رقم السجل (الفهرس) 022

هنا سهلت علينا المهمة كثيراً وذلك لان السجلات مرتبه بأرقام فلي كل سجل رقم فريد يميزه عن الآخر وبهذه الحالة سوف يقوم هذا الاستعلام السابق بجلب جميع بيانات العميل "حمد طارق " ....

سوف نبدأ بأخذ فكرة عن البرنامج الذي سوف أشرح عنه وهو عبارة عن دليل مواقع بسيط جداً جداً يعتمد علي قواعد البيانات.

بالطبع لا يوجد فيه تقسيمات والسبب لأننا سوف نضطر إلي وضع أو خلق أكثر من جدول (Table) في قواعد البيانات وسوف نضطر إلي وضع علاقات بينهم وهذا حالياً يعقد الموضوع فسأكتفي الآن بجدول واحد (one table) وسوف اشرح إذا اتسع لدي الوقت الكثير عن (sql) عامة وعن العلاقات (Relationstips) لأنها مثيرة أيضاً جداً

نبدأ في الخطوة الأولى وهي قاعدة البيانات الخاصة بدليل الموقع الذي لدينا , فالبيانات والمعلومات التي غالباً ما يهتم بها دليل الموقع هي عنوان الموقع واسم الموقع والبريد الخاص بالموقع

**هذا هو الكود الخاص بإنشاء قاعدة البيانات الخاصة بدليل الموقع**

```
CREATE TABLE dalal (  
  id int not null auto_increment,  
  sitename varchar(100),  
  add_date date,  
  email varchar(100),  
  site_url varchar(100),  
  description text,  
  primary key (id));
```

هنا سوف يخزن في قواعد البيانات المعلومات التالية :  
id وهي المرشح الأساسي والفريد الذي يتعرف علي الجدول بواسطته .  
sitename وهي اسم الموقع وهي من نوع char  
add\_date تاريخ إضافة الموقع وهو من نوع date  
email البريد الإلكتروني وهو من نوع char وطولها نفس طول اسم الموقع (100)  
site\_url وهو عنوان الموقع أيضاً من نوع char

سوف اكتفي بشرح id لأنه ربما يكون جديد علي البعض وسوف اشرح عمله وليس المهم أن تفهم علمه في هذه الخطوة إنما سوف تفهمها بالتفصيل في الخطوات القادمة بعد قليل إن شاء الله  
id هو عبارة عن المرشح الأساسي لهذا الجدول (primary key)

وتلخص في آخر سطر من الجدول وجود:

**primary key (id)**

ربما ترجمة المصطلحات العلمية وخاصة في الكمبيوتر تضر أكثر مما تنفع وهذا سبب وجيه لي في كتابتي السكريبت مهما كان بلغة إنجليزية ومن ثم تعريبها  
المهم نحن الآن نريد أن نعرف ماذا يعني id سوف أعطي مثال لكي يتضح فلو أردنا مرشح أساسي أو كود فريد نميز به هذا الجدول فلن نستطيع أن نضع هذا الكود الفريد اسم الموقع لأن ربما يكون لدي موقعاً بنفس الاسم والاختلاف بينهما في الوصلة وربما يكون هناك موقع بعنوانين مثل سالف وياهوو إذاً اسم الموقع لا يكمن أن نضعه كود أساسي وأيضاً نفس الكلام ينطبق علي عنوان الموقع وأيضاً علي التاريخ وحتى البريد لا نستطيع أن نضع البريد هو المرشح الوحيد والأساسي للجدول ربما يأتي هذا لكن منطقياً لا يجوز

ربما ترى id من نوع عدد صحيح (int) لكن ربما يصعب عليك الذي بجانبه وهو (not null) وهي تعني بعدم السماح لهذا الحقل أن يكون بدون قيمة  
not null تعني بكل وضوح ربما يسمح بقيمة فارغة ربما تفكر في الصفر فهذا غير صحيح .  
وترى أيضاً هذه العبارة بجانب not null وهي (auto\_increment) وهي تعني بكل وضوح الإضافة التلقائية أو الأتوماتيكية ودائماً ما يبدأ بالقيمة (1)  
وهذا يعني مع كل إضافة موقع سوف تزيد قيمة (id) وهي تبدأ من (1)

**config.php**

وهو عبارة عن ملف التوجيه الذي يحتوي معلومات الموقع وهي المعلومات التالية:

<?

//this moudel config

```
$dbname = "links";  
$dbuname = "root";  
$dbpws = "";  
$dbhost = "localhost";  
?>
```

لا يحتاج إلي شرح لأنه واضح فالذي جرب أن يركب أي برنامج php يعتمد علي قواعد البيانات فالاكيد راح يعدل في ملف مشابه له أو نفسه ومع ذلك سنشرحه :

**الآن نبدأ فى add.php وهذا هو :**

[illegible]

من رؤيتك له تلاحظ أنه عبارة عن html و php لكن كود php أكثر

بالنسبة html فهي بسيطة بإمكانك فهمها وهي عبارة عن حقول تطلب من المستخدم إدخال معلومات الموقع سوف أبدأ في شرح كود php فتلاحظ بدايته ونهايته بهذين الوسمين <? و >?

وبعدها يأتي شرط التحقق وهو

```
if ($action == 'addsite')
```

فهذا يعني التحقق هل من إضافة جديدة علي الحقول الخاصة بعمومات الموقع وتلاحظ أن التحقق من هذا الشرط بهذه القيمة ('addsite') بين وسمين اقتباس وحيدة ربما يكون هناك بعض التساؤل قليلاً عليه لكن لو تلاحظ في كود html في form وضعت الانتقال بعد إضافة المعلومات إلي (add.php?addsite) هنا

[illegible]

هذا هو كود الفورم الخاص بإدخال معلومات الموقع فإذا لاحظته في بدايته هنا

```
<form method="POST" action="add.php?action=addsite">
```

وضعت الإرسال بال ( post ) وقمت بوضع قيمة ( action ) الصفحة ( add.php?addsite )  
 هذا يعني أن قيمة (action) في حال ضغط submit ستكون ما هو مكتوب بعد علامة الاستفهام ( ؟ ) بعد  
 اسم الصفحة وهي (add.php?addsite)  
 أتمنى أنك فهمت المقصود

بعد ذلك إذا تحقق الشرط ووجد من أرسل سوف ينفذ الأوامر والأولى منها

```
include("config.php");
```

وأمر include يعني استجلاب هذا الملف وقيم المتغيرات الموجودة به بعد هذا الأمر طلبت الاتصال بقواعد البيانات لكي يتم تجهيز الجدول للإضافة وهو هنا

```
mysql_pconnect($dbhost, $dbuname, $dbpw);
```

```
@mysql_select_db("$dbname") or die ("Unable to select database");
```

الدالة `mysql_pconnect` تطلب الاتصال بخادم قاعدة البيانات علي أساس المعلومات الموجودة بها  
وهنا نحن استعنا بعد الله بالمتغيرات الموجودة في ملف `config.php`  
وربما هذا يوضح لنا أكثر فائدة الدالة `include`

ربما لاحظت في طلب الاتصال في الدالة `mysql_pconncet` في السطر الثاني بعدها هذا

```
@mysql_select_db("$dbname") or die ("Unable to select database");
```

هنا في الدالة `mysql_select_db` وضعنا قيمة المتغير `dbname` وهو اسم قاعدة البيانات  
فهذه الدالة تقوم بتحديد اسم قاعدة البيانات المراد الاتصال بها وهذا ربما يوضح استخدام علامة `@`  
لكي يعمل كل شيء بنفس الوقت  
بعد هذا نلاحظ السطر التالي :

```
mysql_query("insert into dalal values ('','$name','','$email','$siteurl','$des')");
```

قمنا في هذا السطر بتنفيذ الدالة mysql\_query وهذه الدالة تقوم باختصار بتنفيذ استعلامات (sql) وفي هذا السطر استخدمنا أمر الاستعلام المراد تنفيذه بواسطة الدالة mysql\_query هو (Insert) هذا الأمر أو الدالة في sql تقوم بإضافة البيانات في قاعدة البيانات والصيغة أو القاعدة الثابتة له هي

```
INSERT INTO tablename VALUES (var and values )
```

## INSERT INTO

هذا ثابت لدينا بعده يأتي اسم الجدول الذي نريد إضافة البيانات بداخله  
ويعد ذلك تأتي كلمة VALUES (var) وتفتح قوس وتيحت المتغيرات والقيم المراد إضافتها بأقواس اقتباس ("") وتنهيه بفاصلة منقوطة ;  
تلاحظ حين استخدمت mysql\_query و insert لم أضع القيم بل وضعت المتغيرات التي تحمل القيم المراد تخزينها في جدول dalal ربما هذا يسهل علينا الكثير  
بعد ذلك في تأتي الدالة mysql\_close وهي تعني طلب إغلاق الاتصال بخادم قاعدة البيانات  
بهذا انتهينا من ملف add.php و config.php

هنا سوف نعرف كيف نستخرج البيانات الموجودة داخل قاعدة البيانات  
وهي في غاية السهولة

### بداية سوف أضع كود show.php كاملاً وأشرحه مثل سابقه

```
<html>
<head>
<style type="text/css">
body {font-family:verdana,verdana; font-size:10pt}
TD {font-family:verdana,verdana; font-size:10pt}
#header {color:black; font-weight:bold; font-family:verdana,verdana}
A {color:navy; text-decoration:underline}
A:hover {color:red}
A:visit {color:navy}
</style>
</head><titel><center><b>view The sites</b></center><br><center><a href='add.php'>
Add Your Site</a></center> <hr color="#000080"></titel>

<body>
<?php
include("config.php");
mysql_pconnect($dbhost, $dbuname, $dbpw);
    @mysql_select_db("$dbname") or die ("Unable to select database");

$query="select * from dalal ";

$result=mysql_query($query);

mysql_close();

/*Display Results*/

$num=mysql_numrows($result);

$i=0;
while ($i < $num) {

    $sitename=mysql_result($result,$i,"sitename");
    $add_date=mysql_result($result,$i,"add_date");
    $email=mysql_result($result,$i,"email");
    $siteurl=mysql_result($result,$i,"site_url");
    $description=mysql_result($result,$i,"description");
    $id=mysql_result($result,$i,"id");
?>
```



```

<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="100%"><b>SiteName :</b><a href='<?echo $siteurl;?>'><i><? echo $sitena
me;?></i></a>
<p><b>Date Add :</b> <i> <?echo $add_date;?></i></p>
<p><b>description :</b><?echo $description;?></p>
<p><b>URL :</b><i><?echo $siteurl;?></i></p>
<p><b>Email :</b><i> <a href="mailto:<?echo $email;?>"><?echo $email;?></a></i>
</p>
</td>
</tr>
</table>
<a href='edit.php?id=<?echo $id;?>'><b><b>[Edit This Site]</b></b></a>
<hr color="#000080">
<?
++$i;
}

?>
</body>
</html>

```

كما تلاحظ تقريباً شبيهه بالملف add.php في عملية الاتصال لكن الجديد لدينا هنا السطر الذي يلي عملية الاتصال وهو

```
$query="select * from dalal ";
```

في هذا السطر قمنا بتعريف متغير يدعى query وقمنا بوضع قيمة هذا المتغير عملية الاستعلام عن المعلومات الموجودة في الجدول dalal  
أظن هذا واضح لكن دعني أشرح عمل select فهي تقوم بتحديد الحقول بداخل الجداول أو الجدول وهي دائماً تستخدم في إخراج البيانات من الجداول فتلاحظ وضعنا قيمة الحقل أو الحقول في استعلامنا هذا علامة الضرب ( \* ) وهي تعني تحديد جميع الحقول الموجودة وبعدها استعملنا from وهذا يعني من ودائماً يأتي بعده أسماء الجدول أو اسم الجدول الذي قمت بتحديد الحقول بداخله ببساطة واختصاراً لهذا السطر select \* from dalal حدد جميع الحقول من جدول الدليل

هنا سوف يحدد جميع الحقول الموجودة في جدول الدليل ويقوم بتخزينها في المتغير query بسيطة

لا تنس أن تضعها بين أقواس اقتباس مزدوجة ( " " )  
بعد ذلك يأتي هذا السطر

```
$result=mysql_query($query);
```

قمنا هنا أيضاً بتعريف المتغير result وقمنا بتخزين قيمة الدالة mysql\_query وقيمة المتغير الذي استعملت به الدالة mysql\_query في المتغير result  
بوضوح هنا قمنا بالاستعانة بالدالة mysql\_query وانسبنا إليها قيمة المتغير query الموجود به مسبقاً قيمة الاستعلام مسبقاً في السطر الذي يسبق هذا السطر وقمنا بهذا لكي يتم فرز جميع القيم الموجودة في الجدول dalal بواسطة الدالة mysql\_query ( هذا أبسط من اللي قبل )

بعد ذلك قمنا بإغلاق قاعدة البيانات بالدالة mysql\_close وبعد ذلك قمنا بتنويه بسيط وتعليق يوضح أننا سوف نعرض ونطبع القيم الموجودة في الجدول في هذا السطر

```
/*Display Results*/
```

بعد هذا السطر قمنا بتعريف متغير يدعى num وأضافنا القيمة إليه باستخدام الدالة mysql\_numrows مسندين إليه قيمة المتغير result

الدالة mysql\_numrows تقوم باسترجاع ومعالجة نتيجتنا المخزنة في المتغير result وتقوم بتخزينها في المتغير num علي شكل مصفوفة  
بعد ذلك قمنا بتعريف عداد يدعى i ووضعنا قيمته صفر لكي نستخدمه مع أمر التكرار while وقمنا بوضع الشرط في أمر التكرار while

```
i < $num$
```

هذا يعني إذا كان قيمة العداد i اصغر من قيمة المتغير num المتواجد بداخله جميع المعلومات فاستمر طباعة التالي وهي هذه السطور

```
$sitename=mysql_result($result,$i,"sitename");
$add_date=mysql_result($result,$i,"add_date");
$email=mysql_result($result,$i,"email");
$siteurl=mysql_result($result,$i,"site_url");
$description=mysql_result($result,$i,"description");
$id=mysql_result($result,$i,"id");
?>
<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="100%"><b>SiteName :</b><a href='<?echo $siteurl;?>'><i><? echo $sitename;?></i></a>
<p><b>Date Add :</b> <i> <?echo $add_date;?></i></p>
<p><b>description :</b><?echo$description;?></p>
<p><b>URL :</b><i><?echo$siteurl;?></i></p>
<p><b>Email :</b><i> <a href="mailto:<?echo$email;?>"><?echo $email;?></a></i>
</p>
</td>
</tr>
</table>
<a href='edit.php?id=<?echo $id;?>'><b><b>[Edit This Site]</b></b></a>
<hr color="#000080">
<?
++$i;
}
```

هنا قمنا بالكثير في أمر التكرار WHILE فسوف يطبع جميع القيم الموجودة في الجدول كما رمزت إليها في هذا السطر

```
$sitename=mysql_result($result,$i,"sitename");
```

قمنا في هذا السطر والسطور الخمسة التي تليه بتعريف العديد من المتغيرات وقمنا بتخزين قيم المعلومات بداخله باستخدام الدالة mysql\_result مسندين إليه المتغيرات result و i و الحقل الذي نريد إخراج القيمة منه  
واستخدمنا في هذا السطر الحقل sitename

كما استخدمنا كود html لكي ننسق ونرتب شكل ظهور المعلومات  
وفي آخر سطر قبل أن ننهي عملية التكرار while بالقوس } جعلنا قيمة العداد i تزيد بمقدار واحد مع كل لفة أو بمعنى طباعة قيم من قاعدة البيانات لكي يتم توقفها أرجو أن تنتبه لهذه الأمور الصغيرة لأنها من مشاكل البرمجة المشهورة

هنا انتهينا من ملف show.php يتبقى لنا ملف edit.php وهو الملف الخاص بتعديل الموقع فربما يخطئ أحدهم في إدخال اسم أو عنوان موقعه فهنا يمكنك تعديلها حتى بعد تخزينها في قاعدة البيانات  
هذا آخر ملف سوف اشرحه اليوم من ضمن مشروع دليل المواقع وربما اشرح عملية الحذف والبحث في وقت آخر

فقط ادعوا لي بالستر والهداية والوقت الكافي

**سوف أضع كود edit.php كاملاً كسابقيه**

```

<html>
<head>
<style type="text/css">
body {font-family:verdana,verdana; font-size:10pt}
TD {font-family:verdana,verdana; font-size:10pt}
TH {font-size:10pt; color:white; font-weight:bold; font-family:verdana,verdana}
A {color:navy; text-decoration:underline}
A:hover {color:red}
A:visit {color:navy}
#mySelect {font-family:verdana,arial; font-size:9pt}
</style>
</head>
<title><center><b>Edit Site</b></center><br><center><a href='add.php'><b>Add Your Si
te </b></a> |
<a href='show.php'><b>View Edit database</b></center></a> <hr color="#000080"></title
|>
<body>

<table border=0 cellpadding=4 cellspacing=0 width=600>
<? echo "<form method='post' action='$PHP_SELF'>"; ?>
<table border="1" width="100%" bordercolor="#000080" cellspacing="1" cellpadding>
<tr>
<td width="100%">
<p align="center">Edit The Info </td>
<?
include("config.php");
mysql_pconnect($dbhost, $dbuname, $dbpw);
    @mysql_select_db("$dbname") or die ("Unable to select database");

$query1 = "select * from dalal where id = '$id'";

$result1 = mysql_db_query('links',$query1);

while($row = mysql_fetch_object($result1)) {
    echo "<td align=middle>SiteNmae<br><input type=text name='name' value='$row-
>sitename'></td>";
    echo "<td align=middle>Add Date<br><input type=text name='add_date' value='$row-
>add_date'></td>";
    echo "<td align=middle>Email<br><input type=text name='email' value='$row-
>email'></td>";
    echo "<td align=middle>SiteUrl<br><input type=text name='siteurl' value='$row-
>site_url'></td></tr>";
    echo "<td align=middle>Description<br><input type=text name='desc' value='$row-
>description'></td></tr>";
$id = $row->id;
}
echo "</table><p>";
echo "<br><input type=hidden name='id' value='$id'>";
?>

<input type=submit name="update" value="Edit The Info">
</form>

```

```

</tr>
</table>
<?
if ($update) {
    if($name == "" || $add_date == "" || $email == "" || $siteurl == "" || $desc == "" ) {
        die("<b>You left one or more fields blank.</b>");
    }

    $query2 = "update dalal set sitename='$name', add_date='$add_date', email='$email', site_
url='$siteurl', description='$desc'
        where id='$id'";
    mysql_db_query('links',$query2);
    echo "<b>Your record has been updated</b></p>";
}
mysql_close();
?>
</body>
</html>

```

هنا سوف يسهل علي شرح هذا الملف أكثر من سابقه لأنه يكرر ما يفعله إخوانه لكن مع بعض التعديلات التي لا تذكر فالجديد هنا وفي هذا السطر

```
$query1 = "select * from dalal where id = '$id'";
```

هنا تلاحظ عرفنا متغير يدعى query1 وقمنا بتخزين ناتج الاستعلام بداخله فهذا ليس بجديد الجديد ما يوجد في الاستعلام فقد استخدمنا where وهي تعني بالترجمة إلي العربية "بحيث" وتعني في البرمجة وخاصةً مع أوامر sql "بشرط" أي هنا اشترطنا في استعلامنا أن تكون قيمة المتغير id مساوية لنفس القيمة الموجودة في الجدول وهي id وربما هنا يسعني أن أوضح أكثر ما هو id ولماذا عرفته وجعلته primary key

هنا إذا ضغط أحد الذين أضافوا مواقعهم بعد أن أخطأ في إضافة موقعه علي edit فسوف يعتمد علي الرقم المخزن في id وهو كما سبق وأن عرفتها في قاعدة البيانات عدد صحيح ويزداد تلقائياً فلو كان لدينا مثلاً في قاعدة البيانات أربعة مواقع فلن يجري التعرف علي الأربعة مواقع ومعلوماتهم إلا عن طريق id لأن كل موقع منهم رقم بمعنى الأول منهم 1 والآخر 4 أرجو أنك فهمته وأرجو أن تحاول تشغيل السكريبت لكي تفهمه أكثر وأكثر

بعد ذلك يأتي هذا السطر

```
$result1 = mysql_db_query('links',$query1);
```

هنا أيضاً عرفنا المتغير result1 وخرنا بداخله قيمة استعلام الدالة mysql\_db\_query وأسندنا إليها اسم قاعدة البيانات والمتغير query

الدالة mysql\_db\_query وهي مشابهة في عملها مع mysql\_query ما عدا الفرق بينهم أنك تستطيع تحديد اسم قاعدة البيانات الذي تريد الاستعلام عنه واستعملتها هنا لكي يتم إعطائنا النتائج بوضوح أكثر بعد هذا السطر يأتي أمر التكرار while مرة أخرى في هذا الملف

```
while($row = mysql_fetch_object($result1))
```

هنا قمنا بخلق المتغير row ولا تنسى في php ليس من الضروري تعريف المتغيرات قبل استخدامها وقمنا بمساواته ( وهو الشرط الخاص بأمر التكرار while ) بالدالة mysql\_fetch\_object مسندين إليه قيمة المتغير result1

الدالة mysql\_fetch\_object وظيفته أو من ضمن الوظائف التي تقوم به أنها تقوم بانتزاع وعرض نتائج الاستعلام

فإذا تحقق شرط أمر التكرار while فستنفذ هذا الكود الطويل قليلاً

```

while($row = mysql_fetch_object($result1)) {
    echo "<td align=middle>SiteNmae<br><input type=text name='name' value='$row-
>sitename'></td>";
    echo "<td align=middle>Add Date<br><input type=text name='add_date' value='$row-
>add_date'></td>";
}

```

```

    echo "<td align=middle>Email<br><input type=text name='email' value='$row-
>email'></td>";
    echo "<td align=middle>SiteUrl<br><input type=text name='siteurl' value='$row-
>site_url'></td></tr>";
    echo "<td align=middle>Description<br><input type=text name='desc' value='$row-
>description'></td></tr>";
    $id = $row->id;
}
echo "</table><p>";
echo "<br><input type=hidden name='id' value='$id'>";
?>

<input type=submit name="update" value="Edit The Info">
</form>
</tr>
</table>
<?
if ($update) {
    if($name == "" || $add_date == "" || $email == "" || $siteurl == "" || $desc == "" ) {
        die("<b>You left one or more fields blank.</b>");
    }

    $query2 = "update dalal set sitename='$name', add_date='$add_date', email='$email', site_
url='$siteurl', description='$desc'
        where id='$id'";
    mysql_db_query('links',$query2);
    echo "<b>Your record has been updated</b></p>";
}
mysql_close();
?>

```

هنا قمنا بطباعة المعلومات والتحقق بواسطة شرط IF دعنا نأخذ عملية طباعة المعلومات :

```

echo "<td align=middle>SiteNmae<br><input type=text name='name' value='$row-
>sitename'></td>";
    echo "<td align=middle>Add Date<br><input type=text name='add_date' value='$row-
>add_date'></td>";
    echo "<td align=middle>Email<br><input type=text name='email' value='$row-
>email'></td>";
    echo "<td align=middle>SiteUrl<br><input type=text name='siteurl' value='$row-
>site_url'></td></tr>";
    echo "<td align=middle>Description<br><input type=text name='desc' value='$row-
>description'></td></tr>";
    $id = $row->id;

```

هنا في أول سطر قام بطباعة كود HTML وبداخله أيضاً بعض الأشياء من PHP لا تخف فالمقصود هنا أنه قام بتخزين جميع ما يطبعه بداخل المتغير الخاص بلغة HTML فهنا مثلاً

```

echo "<td align=middle>SiteNmae<br><input type=text
name='name' value='$row->sitename'></td>";

```

قام بطباعة كود HTML وهو عبارة عن حقل الإدخال (INPUT) من نوع (TEXT) وقام بترمز إلي الحقل الموجود بقاعدة البيانات من المتغير row وتخزينه في قيمة المتغير في (html name)

والأكثر وضوحاً ما قمنا به في السطر السابق قمنا بوضع المتغير row ثم يليه اسم الحقل في الجدول وهنا اسم الحقل sitename

أظن مع تشغيلك للسكربت سوف تتضح أكثر وأكثر وهذا الذي استخدمته مع الجميع

بعد ذلك قمت بطباعة أكواد html وهي عبارة عن فورم للإدخال ويأتي بعده شرط if وهو هنا

```

<input type=submit name="update" value="Edit The Info">
</form>
</tr>
</table>
<?
if ($update) {
    if($name == "" || $add_date == "" || $email == "" || $siteurl == "" || $desc == "" ) {
        die("<b>You left one or more fields blank.</b>");
    }

    $query2 = "update dalal set sitename='$name', add_date='$add_date', email='$email', site_
url='$siteurl', description='$desc'
        where id='$id'";
    mysql_db_query('links',$query2);
    echo "<b>Your record has been updated</b></p>";
}
mysql_close();
?>

```

في الشرط نريد التأكد هل تم تعديل أحد البيانات أم لا وهو شبيه بالشرط الذي استخدمته مع ملف add.php بعده إذا كان هناك إدخال فسيقوم بعدة أوامر وأولها شرط آخر if يتأكد هل هناك من الحقول التي جعلت فارغة أم لا وهي هنا

```

if($name == "" || $add_date == "" || $email == "" || $siteurl == "" || $desc == "" ) {
    die("<b>You left one or more fields blank.</b>");
}

```

إذا كان هناك حقول فارغة سوف يطبع له  
 You left one or more fields blank  
 أي أنك تركت حقل أو أكثر فارغ  
 بعد ذلك يأتي هذا

```

$query2 = "update dalal set sitename='$name', add_date='$add_date', email='$email', site_url
='$siteurl', description='$desc'
        where id='$id'";
mysql_db_query('links',$query2);
echo "<b>Your record has been updated</b></p>";
}
mysql_close();

```

قمنا هنا في أول سطر بتعريف المتغير query2 بعد ذلك خزنا بداخله قيمة أمر SQL التحديث update أمر يقوم بتحديث المعلومات في قواعد البيانات من تجديد وحذف ويكتب هكذا

Update tablename Set (values

الكلمات التي تبدأ بحروف كبيتل هي الثوابت بمعنى لا يتم تغييرها والتي تبدأ بحروف سمول مثل sitename و values فهي التي يتم تغييرها  
 كما وضحت في الكود السابق فقد قمت بمساوات المتغيرات name,add\_date,email,site\_url,description  
 بما يقابلها من الحقول في الجدول dalal ووضعت شرط where  
 أن يكون المتغير

id=id\$

وفي السطر الذي يليه هذا قمت باستخدام الدالة mysql\_db\_query مسنداً إليه اسم قاعدة البيانات والمتغير الذي خزن قيمة التحديث وهو query2

لكي يتم تحديث قاعدة البيانات  
وفي السطر الذي يليهما استعملت الدالة mysql\_colse لكي يتم إغلاق الاتصال بقاعدة البيانات  
وهو آخر سطر اليوم

السكربت يتكون من ثلاثة ملفات، الأول view.php لعرض المدخلات في سجل الزوار، والثاني add.php لإضافة مدخل جديد إلى سجل الزوار، والثالث هو config.php ويحتوي على بيانات قاعدة البيانات. في الملف view.php، يتم جلب البيانات المدخلة من قاعدة البيانات وعرضها واحدة تلو الأخرى في صورة جدول HTML، وأسفل هذا الجدول يوجد نموذج لإضافة تعليق جديد في سجل الزوار. الملف add.php يقوم بأخذ البيانات المرسله من النموذج الموجود في الملف view.php، ثم يقوم بمراجعة هذه البيانات والتحقق منها، ثم إضافتها إلى قاعدة البيانات وإعادة المستخدم إلى الملف view.php. يجب أن نحدد الآن البيانات التي نريد تخزينها في قاعدة البيانات:

- الاسم.
- البريد الإلكتروني.
- الصفحة الشخصية.
- التعليق على الموقع.

### \* قاعدة البيانات

من خلال المعلومات السابقة نلاحظ أننا سنحتاج إلى جدول واحد فقط في قاعدة البيانات، وليكن اسمه guestbook، هذه هي الشفرة التي يجب تنفيذها للحصول على البنية الأساسية للجدول:

```
CREATE TABLE guestbook (  
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  name CHAR(100),  
  email CHAR(128),  
  homepage CHAR(200),  
  date DATETIME,  
  ip CHAR(15),  
  message TEXT  
);
```

شفرة SQL السابقة تعرف جدولاً اسمه guestbook يحتوي على الحقول التالية:

id - هذا الحقل يخزن رقماً تسلسلياً يزداد بمقدار واحد مع كل حقل جديد يضاف إلى الجدول، ومواصفاته:  
رقم INTEGER  
بلا إشارة (موجب دائماً) UNSIGNED  
إجباري (لا يمكن أن يترك خالياً) NOT NULL  
يتم تحديثه تلقائياً إلى رقم أكبر من السابق بواحد AUTO\_INCREMENT  
وهو المفتاح الأساسي للوصول إلى بيانات الجدول PRIMARY KEY

name - هذا الحقل يخزن اسم الشخص الموقع في سجل الزوار، ومواصفاته كالتالي:  
يخزن نصاً بطول 100 حرف كحد أقصى ( 100 ) CHAR

email - لتخزين البريد الإلكتروني، 128 حرفاً بحد أقصى (هذا الرقم قياسي تقريباً في أغلب البرامج القائمة على الويب لتخزين عناوين البريد الإلكتروني).

homepage - لتخزين عنوان الصفحة الشخصية للموقع، 200 حرفاً بحد أقصى -- قيمة معقولة.  
date - لتخزين تاريخ المدخل في سجل الزوار، وهو يخزن قيماً من النوع:  
تاريخ ووقت DATETIME

message - لتخزين نص الرسالة (تعليقك على الموقع)،  
وهو حقل يسع لتخزين عدد كبير جداً من الحروف TEXT

### \* الملف config.php

يحتوي على الملف على شفرة برمجية تقوم بإعداد المتغيرات اللازمة للوصول إلى قاعدة البيانات، وسنقوم بالمرّة بإضافة شفرة الاتصال بقاعدة البيانات إلى هذا الملف بدلاً من تكرارها في كل من الملفين view.php و add.php. أولاً سنقوم بإعداد مجموعة من المتغيرات:



```

/* store the hostname of the MySQL server */
$dbhost = "localhost";
/* store the username to login to MySQL */
$dbuser = "root";
/* store the password to login to MySQL */
$dbpass = "";
/* store the name of the MySQL database */
$dbname = "";

```

كما تلاحظ، قمنا بتعريف أربعة متغيرات مختلفة هي:  
 \$dbhost -- وتقوم بتخزين عنوان مزود قاعدة البيانات.  
 \$dbuser -- وتقوم بتخزين اسم المستخدم الذي سيستخدم للدخول إلى قاعدة البيانات.  
 \$dbpass -- وتقوم بتخزين كلمة المرور التي ستستخدم مع اسم المستخدم للدخول إلى قاعدة البيانات.  
 \$dbname -- تخزن اسم قاعدة البيانات التي ستحتوي الجدول guestbook.

يمكن للمستخدم الآن تعديل معلومات هذه المتغيرة عن طريق فتح الملف في المفكرة مثلاً وتعديل القيم الموجودة بين علامات الاقتباس مقابل أسماء المتغيرات.  
 بعد تعريف المتغيرات السابقة، سنقوم كما اتفقنا بوضع الشفرة اللازمة للاتصال بقاعدة البيانات، وهي عبارة عن استدعاء لدالتين:

```

/* connect to the MySQL server */
mysql_connect($dbhost,$dbuser,$dbpass);
/* set $dbname as the database to be used */
mysql_select_db($dbname);

```

## \* شرح الدوال

**mysql\_connect(hostname, username, password)**

تقوم الدالة السابقة بالاتصال بمزود قاعدة البيانات الموجود على العنوان hostname، وتحاول الدخول إلى المزود قاعدة البيانات عن طريق إرسال اسم المستخدم username وكلمة المرور password، في حال نجاح العملية تعيد مورداً ( resource ) نستطيع استخدامه مع دوال أخرى لتنفيذ أوامر قاعدة البيانات على هذا الإتصال المفتوح (يمكن فتح أكثر من اتصال بأكثر من قاعدة بيانات في نفس الوقت).

جميع المتغيرات الممررة إلى الدالة اختبارية، إذا لم تدخل password فإن الدالة ستحاول إجراء الاتصال دون تمرير كلمة مرور، وإذا لم تدخل username فستحاول الدالة إجراء الاتصال دون اسم مستخدم، وإذا لم تحدد hostname فإن الدالة ستحاول الإتصال بقاعدة البيانات على العنوان المحلي ( localhost ).  
 إذا كنت قد ركبت MySQL مباشرة دون تغيير الإعدادات فإن مزود قواعد البيانات سيسمح مباشرة بالاتصال من الجهاز المحلي باسم المستخدم root والذي لن تكون له كلمة مرور افتراضية، وبالتالي يمكنك الاتصال مباشرة إلى مزود MySQL المحلي عن طريق القيم الموضحة في المتغيرات التي في الأعلى.

**mysql\_select\_db(database\_name, link\_identifier)**

تقوم هذه الدالة بتحديد قاعدة البيانات التي سيتم العمل عليها، حيث أن مزود قاعدة البيانات يمكن أن يحتوي على أكثر من قاعدة بيانات واحدة.

تحدد اسم قاعدة البيانات عن طريق المتغيرة database\_name، أما المتغيرة الثانية فهو الإتصال الذي تريد أن تحدد قاعدة البيانات له، فإذا كنت قد خزنت القيمة التي أعادتها الدالة mysql\_connect في متغيرة، وأرت الآن تعيين قاعدة البيانات الذي يجب استخدامها في ذلك الإتصال، فيمكنك تمرير المتغيرة إلى الدالة mysql\_select\_db في الوسيطة الثانية، ولكن هذه الوسيطة الثانية اختيارية، فإذا لم تقم بتمريرها فإن الدالة ستعيد قاعدة البيانات التي يجب العمل عليها لآخر اتصال تم فتحه بقاعدة البيانات، وإذا لم يكن هنالك اتصال مفتوح من قبل ستحاول الدالة فتح إتصال بقاعدة البيانات كما لو كنت قد شغلت mysql\_connect() بدون تحديد الـ hostname و الـ username والـ password و ثم تعيين قاعدة البيانات التي يجب العمل عليها لهذا الإتصال الجديد.

في ملفنا المثال قمنا بتنفيذ هذه الدالة مباشرة بعد فتح اتصال بقاعدة البيانات باستخدام الدالة mysql\_connect، وبالتالي عينا قاعدة البيانات \$dbname كقاعدة البيانات التي يجب استخدامها مع آخر اتصال فتح بقاعدة البيانات.  
 هذه الدالة تغير true ( صح ) في حالة نجاحها و false ( خطأ ) في حالة فشلها.

يمكننا الآن النظر إلى نسخة نهائية كاملة من الملف config.php :

```
<?php
/* store the hostname of the MySQL server */
$dbhost = "localhost";
/* store the username to login to MySQL */
$dbuser = "root";
/* store the password to login to MySQL */
$dbpass = "";
/* store the name of the MySQL database */
$dbname = "";
/* connect to the MySQL server */
mysql_connect($dbhost,$dbuser,$dbpass);
/* set $dbname as the database to be used */
mysql_select_db($dbname);
?>
```

### \* الملف view.php :

في البداية المقدمات العادية ورأس المستند بلغة HTML:

```
<html dir="rtl">
<head>
</title>سجل الزوار<title>
<style>
body,td,input,select,textarea {
font-family: Tahoma;
font-size: x-small;
}
h1,h2,h3,h4,h5,h6 {
font-family: Arial;
}
</style>
</head>
<body>
</h1>سجل الزوار<h1>
```

بعد أن تتم طباعة رأس المستند بلغة HTML نأتي إلى الجزء التالي وهو جلب البيانات من قاعدة البيانات وعرضها للمستخدم.

في البداية نحتاج إلى اتصال بقاعدة البيانات، وكما قلنا من قبل، الملف config.php يحتوي على الشفرة الكاملة لتجهيز اتصال بقاعدة البيانات، ما علينا القيام به الآن هو إعطاء أمر بتنفيذ البيانات الموجودة في الملف config.php، وهذا الأمر هو include، كالتالي:

```
/* get a database connection */
include 'config.php';
```

النتيجة التي يعطيها الأمر include السابق هي نفسها النتيجة التي كنا سنحصل عليها لو كنا وضعنا الشفرة الموجودة في الملف view.php مباشرة، بمعنى أننا بدلا من أن ننسخ شفرة قاعدة البيانات الموجودة في الملف config.php يدويا ونضعها في أعلى الملف view.php للحصول على وصول إلى قواعد البيانات، فإننا نطلب من مترجم PHP أن يقوم بالعملية آليا، حيث نطلب منه تنفيذ الشفرة الموجودة في الملف config.php كما لو كانت جزءا من الملف الذي استدعى الأمر ( view.php ).

### \* الأمر include :

يستخدم الأمر include كالتالي :

```
include filename;
```

أي الكلمة include وبعدها اسم الملف الذي نريد إدراجه، وفي مثالنا كان الملف هو config.php.

\* طلب البيانات من قاعدة البيانات

المفترض الآن بأن جميع الإضافات التي تمت إلى سجل الزوار مخزنة في الجدول guestbook في قاعدة البيانات، وأن لدينا اتصالاً الآن بقاعدة البيانات من خلال الملف config.php، نستطيع البدء بعمل استعلام لقاعدة البيانات نطلب فيه البيانات التي نريدها، ويكون ذلك بالشفرة التالية:

```
$res = mysql_query("SELECT
name,email,homepage,
UNIX_TIMESTAMP(date) AS date,
message
FROM guestbook
ORDER BY date DESC" );
```

### الدالة mysql\_query:

```
mysql_query(query,link_identifier);
```

تقوم الدالة بإرسال الإستعلام (query) إلى قاعدة البيانات عبر الإتصال الذي يمرر إليها (link\_identifier)، وإذا لم يكن هنالك اتصال ممرر، فإن الدالة تستخدم آخر اتصال فتح بقاعدة البيانات، وإذا لم يكن هنالك اتصال قد فتح من قبل ستحاول الدالة إجراء اتصال بقاعدة البيانات عن طريق تنفيذ الدالة mysql\_connect وتمرر إليها قيم hostname و username و password فارغة.

في مثالنا السابق، قمنا بتمرير قيمة واحدة فقط إلى الدالة هي الاستعلام الطويل المحاط بعلامتي الاقتباس " و "، وستقوم الدالة باستخدام آخر اتصال فتح بقاعدة البيانات وهو الاتصال الذي فتح في الملف config.php الذي أدرجناه من قبل في ملفنا (view.php).

نأتي الآن إلى الإستعلام، وهو عبارة عن استعلام بسيط من النوع SELECT كتب بلغة SQL . وسأقدم شرحاً سريعاً هنا،  
:SELECT  
يعني اختيار أو جلب.

```
name,email,homepage,UNIX_TIMESTAMP(date) AS date,message
```

هذه هي أسماء الحقول التي طلبناها (جلبناها) من الجدول، وهي عبارة عن قائمة بسيطة من أسماء الحقول، ما عدى التالي:

```
UNIX_TIMESTAMP(date) AS date
```

تقوم الدالة UNIX\_TIMESTAMP بإعادة التاريخ المخزن في قاعدة البيانات في صورة TIMESTAMP الخاص بنظام unix، وهو عبارة عن نظام للتاريخ يحسب عدد الثواني منذ منتصف الليل في 1/1/1970، وبالتالي فإن هذه القيمة تتغير في كل ثانية، تعتبر ال Timestamp الوحدة القياسية لتخزين المعلومات عن التاريخ والوقت في PHP وأغلب لغات برمجة ونظم Unix، وبالتالي فإننا طلبنا من قاعدة البيانات أن تحول لنا قيمة التاريخ ( date ) إلى ال Timestamp المقابل حتى يسهل التعامل معها في PHP وعرضها بالطريقة التي نريدها ..

بعد ذلك أضفنا المقطع AS date، هذا المقطع يعني بأن الحقل هذا يجب أن يسمى بالاسم date، السبب في إضافتنا لهذا المقطع هي أن الحقول تسمى عادة بأسماء التعبيرات التي تتكون منها، فهذا الحقل مثلاً: UNIX\_TIMESTAMP(date)

سيكون اسمه هو :

```
UNIX_TIMESTAMP(date)
```

وهذا سيجعل التعامل معه في PHP صعباً نوعاً ما، لذا قلنا لقاعدة البيانات أن تسمى البيانات هذه بالاسم date بدلاً من الاسم السابق.

بعد الانتهاء من قائمة الحقول نجد المقطع التالي:

```
FROM guestbook
```

## FROM

الكلمة FROM تعني (من) ونحدد بعدها اسم الجدول الذي نريد طلب (جلب) البيانات منه، وهو guestbook في مثالنا السابق.

ORDER BY date DESC

المقطع السابق يحدد الطريقة التي ترتب بها البيانات ..

ORDER BY

تعني (ترتيب حسب) وهي تحدد الحقل الذي نريد ترتيب تبعاً له، وفي مثال الحقل date، أي أننا نريد ترتيب المدخلات حسب تاريخ إضافتها، ولكن استخدام ORDER BY وبعده اسم الحقل يعني فرز البيانات حسب ذلك الحقل تصاعدياً، أي من الأقل إلى الأكبر، ولكننا في الواقع نريد فرزها في برنامجنا تنازلياً (من الأحدث أو الأعلى تاريخاً إلى الأقدم) ولهذا السبب أضفنا المقطع DESC وهو اختصار لكلمة descending أي تنازلياً.