الفصل الثاني عشر
التاريخ في PHP

دالة التاريخ في الـ PHP هي Date، ولها معاملين (أي قيمتين لإعداد مخرجات الدالة)، أحد المعاملين إجباري والثاني اختياري، أما الأول وهو الأهم تعتمد عليه مخرجات التاريخ بشكل أساسي مثل ضبط السنة بخانتين أو ضبط الشهر باسم الشهر.. وغيرها، أما 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 ٢٠٠٧-٠٣-١٣، ولأهمية الرموز التي يمكن استخدامها مع الـ Date سأذكر أهمها:

d رقم اليوم في الشهر على شكل خانتين من ٠١ إلى ٣١.

D اسم اليوم في الأسبوع على شكل ٣ خانات مثل Mon أي الاثنين.

g رقم الساعة في اليوم من ١ إلى ١٢.

j رقم اليوم في الشهر من ١ إلى ٣١ بدون وضع الصفر.

m رقم الشهر في السنة على شكل خانتين من ٠١ إلى ١٢.

y رقم السنة على شكل خانتين، مثلاً ٠٢.

Y رقم السنة على شكل أربع خانات، ومثالها ٢٠٠٧.

هذا من أهم الرموز لكي تتضح الصورة فقط، ولعلنا نتطرق لها بشكل أوسع قريباً.

لتحويل التاريخ إلى 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 أخرى تراها، الاهم انشاء الجدول.

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

'); كانون الثاني ،' INSERT INTO month_name VALUES (1
'); شباط ،' INSERT INTO month_name VALUES (2
'); أذار ،' INSERT INTO month_name VALUES (3
'); نيسان ،' INSERT INTO month_name VALUES (4
'); أيار ،' INSERT INTO month_name VALUES (5
'); حزيران ،' INSERT INTO month_name VALUES (6
'); تموز ،' INSERT INTO month_name VALUES (7
'); أب ،' INSERT INTO month_name VALUES (8
'); أيلول ،' INSERT INTO month_name VALUES (9
'); تشرين الأول ،' INSERT INTO month_name VALUES (10
'); تشرين الثاني ،' INSERT INTO month_name VALUES (11
');كانون الأول ،' INSERT INTO month_name VALUES (12
```

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

date	title
٢٠٠٧-٠٤-٢٠	الخبر الأول
٢٠٠٧-٠٤-٢٥	الخبر الثاني
٢٠٠٧-٠٥-٠١	الخبر الثالث

قم بإنشاء الجدول:

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

```
');INSERT INTO news VALUES ('الخبر الأول','٢٠٠٧-٠٤-٢٠
');INSERT INTO news VALUES ('الخبر الثاني','٢٠٠٧-٠٤-٢٥
');INSERT INTO news VALUES ('الخبر الثالث','٢٠٠٧-٠٥-٠١
```

بقي أن نقوم بتحويل التاريخ إلى 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 إلى كتابة طلبين لقاعدة البيانات أحدهما لجلب حقول العنوان (title) والآخر لجلب حقول التاريخ (date) كما يلي:

```
<?
$result = mysql_query("select * from news");
\" ،'%d')،$sql = "SELECT CONCAT(DATE_FORMAT(date
'%Y')) ،DATE_FORMAT(date،\" \"،month_name.month،\"
month_name ،AS date FROM news
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"];
 $date<br>"; ،echo "$title
}
?>
```

عند تنفيذ السكريبت، سترى ما يلي:

الخبر الأول، ٢٠ نيسان ٢٠٠٧ .

الخبر الثاني، ٢٥ نيسان ٢٠٠٧ .

الخبر الثالث، ٠١ أيار ٢٠٠٧.

في حالات كثيرة تكون كتابة السكريبت السابق بهذا الشكل مسببة 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 ٢٠٠٧-٠٤-٢٠ كان باسم آخر، مثلاً (العنوان الأول)، وبعد إضافة حقول ترتيب لجلب البيانات كالتالي:

```
<?
$result = mysql_query("select * from news
order by title");
\" ،'%d')،$sql = "SELECT CONCAT(DATE_FORMAT(date
'%Y')) ،DATE_FORMAT(date،\" \"،month_name.month،\"
month_name ،AS date FROM news
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"];
 $date<br>"; ،echo "$title
}
?>
```

ستجد أن النتائج هي:

الخبر الثالث، ٢٠ نيسان ٢٠٠٧

الخبر الثاني، ٢٥ نيسان ٢٠٠٧

العنوان الأول، ٠١ أيار ٢٠٠٧ وهذه بطبيعة الحال مشكلة في توافق البيانات.

ولحلها يجب أن نوافق بين الطلبين لقاعدة البيانات، بمعنى أنه إذا رتبنا الطلب الأول 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، ٠١ أيار ٢٠٠٧

الخبر الثاني، ٢٥ نيسان ٢٠٠٧

العنوان الأول، ٢٠ نيسان ٢٠٠٧

وهي بالتأكيد صحيحة.

بعد المقدمات السابقة والهامة في معرفة أساسيات اللغة يمكننا بداية كتابة البرامج بلغة الـ PHP، وبطبيعة الحال سنبدأ من أصغر الأساسيات وأهمها في كتابة البرامج عموماً وهي المتغيرات.

المتغيرات في لغة الـ PHP تبدأ بعلامة الدولار ($)، ولاسناد 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 (١٧) 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:

- اسماء المتغيرات في كثير من لغات البرمجة لا تتعدى ٢٥٥ 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 إلى أكثر من ١٥ خانة لتسمية أي متغير، لأن المبالغة في تسمية المتغيرات تسبب مشاكل في تذكر المتغيرات وما تحتوية من قيم.

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

- يمكن أن يحتوي اسم المتغير على الحروف أو الأرقام أو علامة (_) فقط، أما 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 إلى متغير معين، وبالتأكيد لو كان لديك أسلوب خاص في تسمية المتغيرات لسهولة الوصول إليها وتذكرها ستكون كتابة السكريبتات أسهل 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 إنه لطباعة متغير معين نستخدم أمر الطباعة (echo) أو (print) كما يلي:

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

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

أنواع البيانات [Data Types]:

في الأمثلة السابقة قمنا بإسناد قيمتين عددية ونصية إلى متغيرين، وبينا الفرق بينهما، وفي لغة الـ PHP بشكل عام يوجد أكثر من هذين النوعين من البيانات، سأشرح بعضاً منها الآن، والبقية في الفصول 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 أعداد أو رموز أو غيرها، ومثال ذلك كما ذكرنا سابقاً:

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

التعامل مع البيانات النصية [String]:

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

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

في هذه الحالة سيتم إسناد القيمة Khaled13.5 إلى 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 استخدم المتغيرات النصية داخل متغيرات نصية أخرى، حيث سيتم تعويض المتغير بقيمته الأصلية.

البيانات العددية [Numeric]:

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

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

التعامل مع البيانات العددية [Numeric]:

العمليات الحسابية المشهورة (+، -، ❖، /) بالاضافة إلى باقي القسمة (٪) عمليات شائعة جداً في التعامل مع المتغيرات العددية، وبطبيعة الحال لن نحتاج إلى ذكر أي مثال عن هذه العمليات، وسنكتفي بذكر بعض النقاط الأساسية التي قل ما يخلو سكريبت منها.

أول النقاط هي إضافة المتغير إلى 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 يزيد عدد الزوار بـ ١، يمكنك كتابة ما يلي:

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

بالتالي ستتم زيادة المتغير ($counter) بـ ١ في كل مرة يتم فيها تنفيذ السكريبت، وبطريقة أخرى يمكن كتابة السطر السابق كالتالي:

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

والـ ++ تعني زيادة قدررها (١) على قيمة المتغير الأصلية، وكذلك الـ -- تعني طرح ١ من القيمة الأصلية.

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

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

وهـذا يعـني زيـادة مقـدارها ٤ علـى قيمـة المـتغير الأصـلية، وبالـسالب كـذلك بـنفس الأسلوب.

ترتيب إنجاز العمليات الحسابية:

يوجد بعض الرموز والعمليات التي تسبق غيرها عند البدء في انجاز عملية حسابية معينة، والترتيب المستخدم في الـ PHP كالتالي:

] -

(int) (double) (string) (array) (object) -- ++ ~ ! -

* / % -

. - + -

<< >> -

> => < =< -

== =! === ==! -

& -

| -

&& -

|| -

? :-

= =+ =* =/ =. =% =& =^ =~ =>> =<< -

print -

AND -

XOR -

OR -

, -

بالتأكيد القائمة طويلة وفيها تفاصيل كثيرة، ولكن من المهم معرفة طريقة إنجاز 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، فمثلاً:

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

ستتم طباعة الرقم ١ إذا كان المتغير ($age) موجوداً (تم إنشائه مسباقاً)، والعكس إذا كان غير موجود ستتم طباعة الرقم ٠، وهذه الدالة يتم استخدامها كثيراً في الشروط وهذا ما سنتطرق إليه 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);
?>
```

ستتم طباعة الرقم (١) في حالة عدم وجود المتغير ($age) أو أن قيمة المتغير تساوي (٠) و(فراغ)، وفي حالة وجود المتغير ($age) لن تتم طباعة أي شيء.

للتحكم في المواقع عن طريق الـ Session أو الجلسات كما اصطلح على تسميتها، ففي البداية سنتعرف على الـ Session وعن التحكم فيها، ومن ثم استخداماتها بالإضافة إلى بعض الأمثلة، وفي النهاية سنتطرق إلى بعض الأخطاء في كتابة الـ Session وحلول تلك الأخطاء، وفي الفصل القادم بإذن الله تعالى سنتطرق إلى مثال كامل للوحة تحكم مبسطة تتعامل بالـ Session، والأمل أن يكون في هذا الشرح المبسط فائدة للجميع..

- مقدمة عن الـ Session:

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

إذا التعرف على الشخص الذي يقوم بتصفح الموقع هو الهدف الرئيسي للـ Session أو الجلسات، ولكن كيف يتم ذلك، وما هي النقاط الرئيسية التي يجب معرفتها لفهم طريقة التعامل مع الـ 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 إغلاق الموقع، ببساطة كل مرة تقوم بزيارة الموقع تبدأ جلسة أو Session جديدة، مع ملاحظة أن هناك طرق للتحكم بوقت الانتهاء كما في الـ cookies، بالإضافة إلى طرق أخرى عن طريق قواعد البيانات وهو حديث سابق لأوانه.

بالنسبة للنقطة الأخرى التي يجب وضعها في الحسبان هي ما يسمى بالـ Session ID أو اختصاراً SID ويعني ذلك (رقم الجلسة)، وهو رقم عشوائي فريد يصعب تكراره أو فلنقل إنه مستحيل لاحتوائه على أرقام وأحرف كبيرة وصغيرة في متغير طويل نسبياً، وهذه القيمة هي الأهم في ما ذكرت، لأنها القيمة الوحيدة التي تربط ما يسمى بالـ Session Variables أو (متغيرات الجلسة) مع جهاز المستخدم، فالـ SID هي القيمة الوحيدة التي يتم تخزينها في جهاز المستخدم (Client)، أما الـ

متغيرات الجلسة Session Variables يتم تخزينها في السيرفر (Server)، فعند التحقق منه وجود هذه القيمة على جهاز المستخدم يمكن الدخول إلى المتغير الآخر المتربط به والمسمى بالـ Session Variable.

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

- إعدادت الـ Session:

عن طريق ملف الـ php.ini والذي يحتوي على إعدادت الـ PHP يمكن التحكم بإعدادات الـ Session، وكاستعراض لتلك النقاط المتحكمة بالـ Session سنتطرق أهم النقاط ومعانيها، وللوصول إلى ما نحن بصدده تذكر أن ملف الـ php.ini يوجد في دليل الـ Windows، وللوصول إلى خصائص الـ Session ابحث عنه كلمة (Session) وستجد السطر التالي:

[Session]

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

١- الخيار **Session.auto_start**:

بداية تلقائية للـ Session (دون الحاجة لعمل ذلك يدوياً عن طريق Session_start).

٢- الخيار **Session.cache_expire**:

وقت انتهاء الجلسة بالدقائق.

٣- الخيار **Session.cookie_lifetime**:

وقت انتهاء الـ cookie المرتبطة بالجلسة، وهي افتراضياً ستكون ٠ أي أن الـ cookie ستنتهي فترتها مع إقفال الشخص المتصفح للموقع.

٤- الخيار **Session_name**:

اسم الـ Session التي ستستخدم كـ cookie وافتراضياً ستكون PHPSESSID.

٥- الخيار **session.save_path**:

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

هذه في نظري أهم الخيارات التي يجب فهمها.

- بداية الـ Session:

قبل أن تستخدم أياً من دوال الـ Session يجب إخبار 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 الـ Session الأخرى، أما إذا لم تتم كتابة هذا السطر فلن يتم ذلك.

ملاحظة مهمة حول عملية بداية الـ Session وهي أن تتأكد من أن هذا السطر لا يسبقه عملية إخراج مخرجات، بمعنى آخر 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 الـ header لأنك سنقوم بإدراج هذه الصفحة في كل الصفحات الأخرى وبالتالي يكون السطر هو الأول في كل الحالات..

- تخزين متغيرات الجلسات:

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

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

ما قمنا بعمله هو التالي:

١- عرفنا متغيراً هو **user** يحتوي على قيمة حرفية.

٢- قمنا بتسجيل هذا المتغير في متغير جلسة (Session Variable) وبنفس الاسم **user** ولكن بدون علامة **$**.

- التعامل مع متغيرات الجلسة:

بعد تسجيل المتغير، يمكن الرجوع إليه بعدة طرق تعتمد على الخيار **register_globals** في ملف الـ **php.ini**، أما إذا كان **on** وهذا هو الاختيار الافتراضي فإن المتغير الذي تم تسجيله في الـ Session يمكن الرجوع إليه كأي متغير آخر، عن طريق اسم المتغير فقط، وفي مثالنا الحالي سيكون **$user**، أما إذا كان الخيار غير مفعل وليس بالصورة التي ذكرتها فيمكن

الرجـــــــوع إلى المـــــــــتغير عـــــــــن طريـــــــــق الأمـــــــــر **$HTTP_SESSION_VARs["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 الـ Session له، في المثال التالي تلك الطريقة:

```
<?
if (session_is_registered("user")) {
echo "أهلاً وسهلاً بكم في السلسلة المعلوماتية الميسرة ";
}
else {
echo "لا يسمح لك بالدخول. ";
}
?>
```

في هذا المثال سيتم عرض الجملة (أهلاً وسهلاً بكم في السلسلة المعلوماتية الميسرة) إذا كانت عملية تسجيل الـ Session تمت للمتغير **user**، وسيتم عرض الجملة (لا يسمح لك بالدخول..) في حالة عدم تسجيل الـ Session.

نقطة أخيرة في التعامل مع متغيرات الجلسة، وهي عملية الغاء تسجيل الـ 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 إلغاء التسجيل لـ Session معينة، أي بتمرير اسم المتغير لها كما في المثال التالي:

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

إذاً سيتم إلغاء تسجيل الـ Session المتعلقة بالمتغير **user** فقط، أما الدالة الثانية فستقوم بإلغاء تسجيل جميع الـ Session التي تم تسجيلها من قبل، وفي النهاية

يجب استخدام الدالة الثالثة **session_destroy** لإلغاء الـ SID والانتهاء من التعامل مع الـ Session.

- مثال بسيط عن الـ Session:

سأتطرق إلى مثال بسيط جداً لتوضيح كيفية عمل الـ Session، في البداية قم بوضع الكود التالي في ملف وقم بتسميته phpex1.php:

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

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

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

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

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

ابدأ من الصفحة الأولى ومن ثم انتقل من صفحة إلى أخرى، حتى تصل إلى الثالثة، بافتراض أنك قمت بتجربة المثال، ستلاحظ أن الصفحة الأولى ستتم طباعة الـ Session التي تم تسجيلها وهي **age** وستظهر القيمة ١٢ في الجملة الطويلة التي تبين أن المتغير **age** يحتوي على قيمة معينة، وفي الصفحة الثانية ستلاحظ نفس الجملة ونفس القيمة تمت طباعتهما، أما في الصفحة الثالثة والأخيرة فتمت طباعة الجملة، لكن الاختلاف أن القيمة ١٢ في متغير الـ Session **age** لم تتم طباعتها، لماذا ؟

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