**Сессии в PHP**

Сессия - это время, в течении которого посетитель находится на сайте. Часто возникает необходимость в том, что бы в течении сеанса пользователя сопровождали некоторые переменные. Можно записать эти переменные в cookie, но если переменных много, то этот способ не очень удобен. Гораздо эффективнее в такой ситуации является использование сеансовых переменных. Об этом и пойдет речь в этой статье.

**Механизм работы сессий**

Наверняка вам интересен механизм работы сессий. Ну что ж, я расскажу. К примеру при входе на сайт пользователю присваивается случайный идентификатор - SID. Он сохраняется на компьютере пользователя в cookie c именем PHPSESSID. Вообще это имя выставляется в файле php.ini, но речь сейчас не об этом. Если же в браузере посетителя отключен прием cookie, то этот идентификатор со значением цепляется ко всем url на протяжении сессии. Во время создания сессии на сервере создается файл, имя которого совпадает с PHPSESSID. В этом файле содержатся все сеансовые переменные. Однако сеансовые переменные так же могут быть сохранены и в базе данных - все зависит от конфигурации PHP. По умолчанию эти переменные сохраняются в файле в той дирректории, в которой лежит скрипт, создавший сеанс. При обращении к сеансовой переменной сервер открывает (или делает запрос к базе данных) сеансовый файл и ищет там значение нужной переменной.

**Создание сессии**

Сессия создается с помощью функции session\_start(). При вызове она проверяет, существует ли уже созданная сессия. Если нет, то она создает сессию. Синтаксис функции:

boolean session\_start();

При создании сеанса функция сначала присваивает пользователю SID, отправляет cookie и создает файл с сеансовыми переменными. Очень легко и удобно.

**Уничтожение сессии**

Если вам по каким то причинам приспичило уничтожить сессию, то воспользуйтесь функцией session\_destroy(). Синтаксис функции:

boolean session\_destroy();

Эта функция уничтожает сеансовые переменные и сеансовый файл (если эти переменные были в нем). Обратите внимание, что эта функция не уничтожает cookie с SID на комьютере посетителя.

**Создание сеансовых переменных**

Ну а теперь о том, ради чего вообще задумывалась технология сессий - о создании сеансовых переменных. Этим занимается функция session\_register(). Синтаксис функции:

boolean session\_register(mixed name [, mixed ...]);

Следует заметить, что эта функция лишь создает сеансовые переменные, и ничего более. Присвайвать значения им придется самостоятельно. Делается это как в случае с обычными переменными. К тому же эта функция неявно вызывает session\_start().Впрочем из следующего примера будет все ясно.

**Как узнать, была ли зарегистрирована переменная**

Это можно узнать с помощью функции session\_is\_registered(). Синтаксис функции:

booleansession\_is\_registered(string name);

Первый пример

Теперь вы знаете достаточно, что бы пользоваться сеансовыми переменными. На основе полученных знаний можно реализовать, к примеру, простейший счетчик посещений. Он будет считать сколько страниц просмотрел пользователь за один сеанс.

<?php

session\_start(); //создаем сессию

//если переменная page\_hits не была зарегистрирована

//то регистрируем её и присваиваем начальное значение

if (!session\_is\_registered('page\_hits')) {

session\_register('page\_hits');

$page\_hits = 0;

}

$page\_hits++;

echo "Вы просмотрели уже $page\_hits страниц";

?>

Такой код можно разместить на любой странице вашего сайта.

Уничтожение сеансовых переменных

Сеансовые переменные уничтожаются функцией session\_unregister() с таким же успехом, как и создаются. Синтаксис функции:

boolean session\_unregister(string name);

Это может быть полезным в том случае, если вам не нужна больше сеансовая переменная и вы хотите использовать её как обычную переменную.

Изменение/Получение SID

Получить или изменить текущий сеансовый идентификатор легко. Достаточно воспользоваться функцией session\_id(). Синтаксис функции:

string session\_id([string sid]);

Если был передан параметр sid, то сеансовый идентификатор изменяется. В противном случае функция просто возвращает SID.

Сохранение и восстановление сеансовых переменных

В PHP сохранение и восстановление сеансовых переменных сведено к использованию функций session\_encode() и session\_decode(). Это очень удобно, поскольку функция session\_encode() формирует строку которую можно где ни будь сохранить и в дальнейшем расшифровать функцией session\_decode(). Эту строку очень удобно хранить в базе данных, поскольку не нужно создавать отдельные поля да и обьем скрипта уменьшается. Итак, синтаксис этих фукнций:

boolean session\_encode();

string session\_decode(string data);

С первой функцией более менее понятно. А второй функции необходимо передать зашифрованную функцией session\_encode() строку. Без примера воспринимается не очень легко. Давайте рассмотрим простенький пример.

Предположим, что сессия уже начата и в переменной $u\_id хранится идентификатор пользователя.

<php

...

//где то в скрипте

$id = session\_id($u\_id);

//подключаемся к базе

mysql\_pconnect("localhost", "root", "") or die("Не могу подключиться к базе");

mysql\_select\_db("mysite"); //выбираем базу

//здесь выбирается поле с сохраненной строкой сеансовых переменных

$query = "SELECT s\_data FROM saved\_users\_data WHERE uid = '$id'";

$result = mysql\_query($query) or die("Query '$query' failed");

$user\_s\_data = mysql\_result($result, 0, 's\_data');

//теперь наша стркоа хранится в переменной $user\_s\_data

session\_decode($user\_s\_data);

//допустим, что среди сохраненных сеансовых переменных

//была переменная $my\_str

echo $my\_str;

//будет выведена строка, которая была сохранена в предыдущем сеансе

...

?>