Лабораторная работа № 5 – Отслеживание сеанса (session)

Цель работы:

Изучение технологии и получение практических навыков отслеживания сеанса пользователя.

**Теоретические сведения**

Сеансом (session) называется период времени, который начинается с момента прихода пользователя на сайт и завершается, когда пользователь покидает сайт. В течение сеанса часто возникает необходимость в сохранении различных переменных, которые бы «сопровождали» пользователя при перемещениях на сайте, чтобы вам не приходилось вручную кодировать многочисленные скрытые поля или переменные, присоединяемые к URL.

При входе на сайт пользователю присваивается уникальный идентификатор сеанса (SID), который сохраняется на компьютере пользователя в cookie с именем PHPSESSJD. Если использование cookie запрещено или cookie вообще не поддерживаются, SID автоматически присоединяется ко всем локальным URL на протяжении сеанса. В то же время на сервере сохраняется файл, имя которого совпадает с SID. По мере того как пользователь перемещается по сайту, значения некоторых параметров должны сохраняться в виде сеансовых переменных. Эти переменные сохраняются в файле пользователя. При последующем обращении к сеансовой переменной сервер открывает сеансовый файл пользователя и ищет в нем нужную переменную. В сущности, в этом и заключается суть отслеживания сеанса. Конечно, информация с таким же успехом может храниться в базе данных или в другом файле.

Рассмотрим процесс отслеживания сеанса более подробно. Первое - сеанс инициируется функцией session\_start( ).

Функция session\_start( ) имеет двойное назначение. Сначала она проверяет, начал ли пользователь новый сеанс, и если нет — начинает его. Синтаксис функции session\_start( ):

**boolean session\_start()**

Если функция начинает новый сеанс, она выполняет три операции: назначение пользователю SID, отправку cookie и создание файла сеанса на сервере. Второе назначение функции заключается в том, что она информирует ядро РНР о возможности использования в сценарии, в котором она была вызвана, сеансовых переменных.

Если сеанс можно создать, значит, его можно и уничтожить. Это делается функцией session\_destroy().

Функция session\_start( ) возвращает TRUE независимо от результата. Следовательно, проверять ее в условиях if или в команде die( ) бессмысленно.

**boolean session\_destroy( )**

Пример использования функции:

<?

session\_start( );

// Выполнить некоторые действия для текущего сеанса

session\_destroy( ):

?>

Теперь вы умеете уничтожать сеансы, и мы можем перейти к работе с сеансовыми переменными. Возможно, самой важной сеансовой переменной является SID (идентификатор сеанса). Его легко можно получить при помощи функции session\_id( ).

session\_id( )

Функция session\_id( ) возвращает SID для сеанса, созданного функцией session\_start( ). Синтаксис функции session\_id( ):

**string session\_id ([string sfd])**

Если в необязательном параметре передается идентификатор, то значение SID текущего сеанса изменяется. Однако следует учитывать, что cookie при этом заново не пересылаются. Пример:

<?

session\_start()

print "Идентификатор сессии = " . sessionjd( ):

session\_destroy( ):

?>

Результат, выводимый в браузере, выглядит примерно так:

Идентификатор сессии = 067d992a949114ee9832flcllcafc640

Сеансовая переменная создается с помощью функции session\_register( ).

session\_register( )

Функция session\_register( ) регистрирует имена одной или нескольких переменных для текущего сеанса. Синтаксис функции session\_register( ):

**boolean session\_register (mixed имя\_переменной1 [, mixed имя\_переменной2... ])**

Следует помнить, что регистрируются не переменные, а их имена. Если сеанс не существует, функция session\_register( ) также неявно вызывает session\_start( ) для создания нового сеанса.

Проверить зарегистрирована ли переменная в сессии можно функцией session\_is\_registered(). session\_is\_registered( )

Часто требуется определить, была ли ранее зарегистрирована переменная с заданным именем. Задача решается при помощи функции session\_is\_registered( ), имеющей следующий синтаксис:

**boolean session\_is\_registered (string имя\_переменной)**

Применение функций session\_register( ) и session\_is\_registered( ) будет продемонстрировано на классическом примере использования сеансовых переменных — счетчике посещений (листинг 1).

Листинг 1. Счетчик посещений сайта пользователем

<?

session\_start( ):

if (! sessionjs\_registered('hits')) :

session\_register( 'hits' ) ;

endif ;

$hits++:

print "You've seen this page $hits times.

?>

Сеансовые переменные удаляются применением функции session\_unregister( ).

session\_unregister( )

Ниже приведен ее синтаксис:

**boolean session\_unregister (string имя\_переменной')**

При вызове функции передается имя сеансовой переменной, которую вы хотите уничтожить.

<?

session\_start()

session\_register('username');

// Использовать переменную $username.

// Когда переменная становится ненужной - уничтожить ее.

session\_unregister('username');

session\_destroy();

?>

Как и в случае с функцией session\_register указывается не сама, а имя переменной.

session\_encode( )

Функция session\_encode( ) обеспечивает чрезвычайно удобную возможность форматирования сеансовых переменных для хранения (например, в базе данных). Синтаксис функции session\_encode( ):

**boolean session\_encode( )**

В результате выполнения этой функции все сеансовые данные форматируются в одну длинную строку, которую можно сохранить в базе данных.

Пример использования session\_encode( ) приведен в листинге 2. Предположим, что на компьютере «зарегистрированного» пользователя имеется cookie, в котором хранится уникальный идентификатор этого пользователя. Когда пользователь запрашивает страницу, содержащую листинг 2, UID читается из cookie и присваивается идентификатору сеанса. Мы создаем несколько сеансовых переменных и присваиваем им значения, после чего форматируем всю информацию функцией session\_encode( ) и заносим в базу данных MySQL.

Листинг 2. Использование функции session\_encode( ) для сохранения данных в базе данных MySQL

<?

// Инициировать сеанс и создать сеансовые переменные

session\_register('bgcolor');

session\_register('fontcolor');

// Предполагается, что переменная $usr\_id (с уникальным идентификатором пользователя) хранится в cookie

// на компьютере пользователя.

// При помощи функции session\_id( ) присвоить идентификатору

//сеанса уникальный идентификатор пользователя (UID), хранящийся в cookie.

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

// Значения следующих переменных могут задаваться пользователем на форме HTML

$bgcolor = "white"; $fontcolor = "blue";

// Преобразовать все сеансовые данные в одну строку

$usr\_data = session\_encode( );

// Подключиться к серверу MySQL и выбрать базу данных users

@mysql\_pconnect("localhost", "root", "") or die("Could not connect to MySQL server!");

@mysql\_select\_db("users") or die("Could not select user database!");

// Обновить пользовательские параметры страницы

$query = "UPDATE user\_info set page\_data='$usr\_data' WHERE user\_id= '$id'";

$result - mysql\_query($query) or die("Could not update user information!");

?>

Как видно, быстрое преобразование всех сеансовых переменных в одну строку избавляет от необходимости создавать несколько полей для хранения/загрузки данных, а также несколько уменьшает объем программы.

session\_decode( )

Все сеансовые данные, ранее преобразованные в строку функцией session\_encode( ), восстанавливаются функцией session\_decode( ). Синтаксис:

**string session\_decode (string сеансовые\_данные)**

В параметре сеансовые\_данные передается преобразованная строка сеансовых переменных, возможно — прочитанная из файла или загруженная из базы данных. Строка восстанавливается, и все сеансовые переменные в строке преобразуются к исходному формату.

В листинге 3 продемонстрировано восстановление закодированных сеансовых переменных функцией session\_decode( ).

Предположим, таблица MySQL с именем **user\_info** состоит из двух полей: **user\_id** и **page\_data**. Пользовательский UID, хранящийся в cookie на компьютере пользователя, применяется для загрузки сеансовых данных, хранящихся в поле **page\_data**. В этом поле хранится закодированная строка переменных, одна из которых ($bgcolor) содержит цвет фона, выбранный пользователем.

Листинг 4. Восстановление сеансовых данных, хранящихся в базе данных MySQL

<?

// Предполагается, что переменная $usr\_id (с уникальным идентификатором пользователя) хранится в cookie

// на компьютере пользователя.

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

// Подключиться к серверу MySQL и выбрать базу данных users

@mysq]\_pconnect("localhost", "web", "4tf9zzzf") or die("Could not connect to MySQL server!");

@mysql\_select\_db("users") or die("Could not select company database!");

// Выбрать данные из таблицы MySQL

$query = "SELECT page\_data FROM user\_info WHERE user\_id= '$id'",

$result = mysql\_query($query);

$user\_data = mysql\_result($result, 0, "page\_data");

// Восстановить данные

session\_decode($user\_data):

// Вывести одну из восстановленных сеансовых переменных

print "BGCOLOR: $bgcolor";

?>

Как видно из двух приведенных листингов, функции session\_encode() и session\_decode() обеспечивают очень удобные и эффективные сохранение и загрузку сеансовых данных.

Задание:

Модифицировать результаты лабораторной работы №5 путем добавления возможности формирования корзины товаров. Для этого необходимо напротив каждого товара установить кнопку (ссылку) «Добавить в корзину». Предусмотреть возможность просмотра корзины и ее модификации (добавление, удаление товаров). Осуществлять контроль над количеством каждого товара в корзине необязательно.

Контрольные вопросы

1. Чем отличаются cookie от session?

2. Основные ограничения, накладываемые на cookie?

3. Способ создания уникальных идентификаторов в PHP?

3. Назначение пользовательских функций для хранения сеансовых данных?

Содержание отчёта

Отчёт по практическому заданию должен содержать следующие сведения:·

1. ответ на контрольные вопросы;
2. название и цель работы;·
3. исходные тексты разработанной программы с комментариями;·
4. результат работы программы.