

РНР. Управление сеансами

- Управление сеансами
- Аутентификация средствами управления сеансами

Управление сеансами

- Идея управления сеансами заключается в обеспечении отслеживания пользователя в течение одного сеанса связи с Web-сайтом. Если это удастся, мы сможем легко поддерживать подключение пользователя и предоставление ему содержимого сайта в соответствии с его уровнем прав доступа или персональными настройками.
- В версиях PHP до 4 управление сеансами осуществлялось средствами PHPLib, базовой библиотеки PHP.

Управление сеансами

- Четвертая версия РНР включает собственные встроенные функции управления сеансом. Концептуально они подобны RHPLib, но RHPLib помимо этого обеспечивает ряд дополнительных функциональных возможностей. Так что если окажется, что эти собственные функции не вполне отвечают вашим требованиям, ничто не мешает рассмотреть возможность использования RHPLib.

Управление сеансами

- Для контроля сеанса в РНР используется уникальный идентификатор сеанса, представляющий собой 32хразрядное шестнадцатиричное случайное число. Идентификатор сеанса генерируется РНР и сохраняется на стороне клиента в течение всего времени жизни сеанса. Для хранения идентификатора сеанса используется либо cookie-набор на компьютере пользователя, либо URL.
- Идентификатор сеанса играет роль ключа, обеспечивающего возможность регистрации некоторых специфических переменных в качестве так называемых переменных сеанса.

Управление сеансами

- Содержимое этих переменных сохраняется на сервере. Единственной информацией, "видимой" на стороне клиента, является идентификатор сеанса. Если во время определенного подключения к вашему сайту идентификатор сеанса является "видимым" либо в cookie-наборе, либо в URL, имеется возможность получить доступ к переменным сеанса, которые сохранены на сервере для данного сеанса. По умолчанию переменные сеанса хранятся в двумерных файлах на сервере.

Управление сеансами

- Cookie-набор — это небольшой фрагмент информации, который сценарии сохраняют на клиентской машине. Чтобы установить cookie-набор на машине пользователя, необходимо отправить ему HTTP-заголовок, содержащий данные в следующем формате.
- `Set-Cookie:NAME=VALUE;`
`[expires=DATE;] [path=PATH;]`
- `[domain=DOMAIN_NAME;]`
- `[secure]`
- Это создаст cookie-набор с именем **NAME** и значением **VALUE**. Все остальные параметры необязательны. В **expires** задается дата истечения срока действия cookie.

Управление сеансами

- Два параметра **path** и **domain** применяются для определения одного или нескольких URL, к которым относится данный cookie-набор. Ключевое слово **secure** означает, что cookie-набор не может отправляться через простое HTTP-соединение.
- Когда браузер соединяется с URL, он сначала ищет cookie-наборы, хранящиеся локально. Если какие-либо из них относятся к URL, с которым установлено соединение, они передаются обратно на сервер.

Управление сеансами

- Cookie-наборы в PHP можно установить, используя функцию `setcookie()`:

```
int setcookie (string name [, string value [,  
int expire [, string path [, string domain [,  
int secure]]]])
```

- Если cookie-набор установлен как

```
setcookie ("mycookie", "value");
```
- то когда пользователь обращается к следующей странице на вашем сайте (или перезагружает текущую), вы получаете доступ к переменной с именем `$mycookie`, которая содержит значение "value". Доступ к этой переменной можно получить также через `$_HTTP_COOKIE_VARS["mycookie"]`.

Управление сеансами

- Для удаления cookie-набора необходимо вызвать **setcookie()** с тем же именем, но без указания значения. Если cookie-набор устанавливался с другими параметрами (такими как специфические URL или даты истечения), потребуется отправить те же параметры повторно, иначе cookie-набор удален не будет.
- Для установки cookie-набора вручную можно воспользоваться также функцией **Header()** и описанным выше синтаксисом. Однако заголовки cookie-наборов должны отправляться перед всеми другими заголовками, иначе они работать не будут).

Управление сеансами

- В сеансе PHP нет необходимости задавать cookie-наборы вручную. Это делают функции сеанса. Чтобы просмотреть содержимое cookie-набора, установленное при управлении сеансом, можно воспользоваться функцией `session_get_cookie_params()`.
- Она возвращает ассоциативный массив, содержащий элементы `lifetime`, `path` и `domain`.
- Можно использовать также:
- `session_set_cookie_params($lifetime, $path, $domain);`
- Этот оператор устанавливает параметры cookie-набора для сеанса.

Управление сеансами

- В PHP cookie-наборы в сеансах используются по умолчанию. Если есть возможность установить cookie-наборы, то для сохранения идентификатора сеанса будет использоваться именно этот способ. Другой метод заключается в добавлении идентификатора сеанса к адресу URL. Чтобы идентификатор сеанса добавлялся к URL автоматически, следует скомпилировать PHP с опцией **--enable-trans-sid**.
- Идентификатор сеанса запоминается в константе PHPSESSID. Можно встроить его в ссылку, добавив в конец ссылки, аналогично параметру GET:
- `<A HREF="link.php?<?=PHPSESSID?>">`

Управление сеансами

- Основными этапами использования сеанса являются следующие:
 - Запуск сеанса
 - Регистрация переменных сеанса
 - Использование переменных сеанса
 - Отмена регистрации переменных и закрытие сеанса
- Прежде чем можно будет воспользоваться функциональными возможностями сеанса, следует запустить сам сеанс. Существует три способа сделать это.

Управление сеансами

- Первый заключается в том, что сценарий начинается с вызова функции `session_start()` ;
- Она проверяет, существует ли идентификатор текущего сеанса. Если нет, она его создает. Если да, она загружает зарегистрированные переменные сеанса, чтобы они стали доступными для использования.
- Вторым способом - сеанс запускается при попытке зарегистрировать переменные сеанса.
- Третий способ — задать установки PHP, при которых сеанс будет запускаться автоматически. Для этого служит опция `session.auto_start` в файле `php.ini`.

Управление сеансами

- Регистрация сеансовых переменных делается вызовом функции `session_register()` :
- `$myvar=5; session_register("var");`
- Вы должны передать в функцию `session_register()` строку, содержащую имя переменной без символа `$`. Отслеживание переменной будет осуществляться, пока не завершится сеанс либо пока вручную не отменится ее регистрация.
- За один прием можно зарегистрировать более одной переменной, передав разделенный запятыми список имен переменных:
- `session_register("var1", "var2");`

Управление сеансами

- Если опция `register_globals` включена, то доступ к этой переменной можно получить через сокращенную форму ее имени, например, `$var`. Если нет, то через ассоциативный массив `$HTTP_SESSION_VARS` ["var"], сокращенно `$_SESSION` ["var"].
- Далее от вас потребуются проверить, установлены ли уже переменные сеанса (например, с использованием `isset()` либо `empty()`). Проверить, является ли переменная зарегистрированной переменной сеанса, можно функцией `session_is_registered()`:
- `$res=session_is_registered("var");`

Управление сеансами

- Эта функция проверит, является ли `$myvar` зарегистрированной переменной сеанса, и вернет `true` или `false`. Можно проверить массив `$_HTTP_SESSION_VARS` на предмет наличия в нем переменной.
- После окончания работы с переменной сеанса ее регистрацию можно отменить:
- `session_unregister("var");`
- Данная функция за один раз может отменить регистрацию только одной переменной сеанса (в противоположность `session_register`). Однако, для отмены регистрации всех переменных текущего сеанса можно обратиться к `session_unset()`.

Управление сеансами

- По завершении сеанса сначала потребуется отменить регистрацию всех переменных, а затем для обнуления идентификатора сеанса вызвать
- `session_destroy()` ;
- Приведенный в [примере код](#) обеспечивает обработку трех страниц. На первой странице мы запустим сеанс и зарегистрируем переменную `$sess_var`.
- В конце сценария переменная сеанса преобразуется в последовательную форму (сериализуется) до своей перезагрузки через вызов `session_start()`. Следующий [сценарий](#) начинается с `session_start()`.

Управление сеансами

- После вызова `session_start()` переменная `$sess_var` станет доступной, а ее значением будет то, которое сохранено ранее.
- Сделав с переменной необходимые действия, мы вызываем `session_unregister()` для отмены ее регистрации. Обратите внимание: сеанс еще существует, но переменная `$sess_var` уже больше не является зарегистрированной.
- Как можно видеть в последнем [сценарии](#), доступа к значению `$sess_var` больше нет. И в завершение — вызов `session_destroy()` для разрушения идентификатора сеанса.

Конфигурирование управления сеансом

Имя опции	default	Действие
session.auto_start	0	Автоматический запуск сеансов.
session.cache_expire	180	Установка времени жизни для кэшированных страниц сеанса (в минутах).
session.cookie_domain	none	Домен для установки в cookie-наборе сеанса.
session.cookie_lifetime	0	Время существования cookie-набора идентификатора сеанса на машине пользователя.

Конфигурирование управления сеансом

Имя опции	default	Действие
session.cookie_path	/	Путь для установки в cookie-наборе сеанса.
session.name	PHPSESSID	Имя сеанса, которое в системе пользователя используется как имя cookie-набора.
session.save_handler	файлы	Определяет место хранения данных сеанса.
session.save_path	/tmp	Путь к месту хранения данных сеанса.

Конфигурирование управления сеансом

Имя опции	default	Действие
session.use_cookies	1	Конфигурация сеанса с возможностью использования cookie- наборов на стороне клиента.

Аутентификация средствами управления сеансом

- Наиболее часто управление сеансом применяется в целях отслеживания пользователей после того, как они были аутентифицированы через механизм входной регистрации. В предлагаемом примере можно видеть, как эти функциональные возможности обеспечиваются за счет сочетания аутентификации при помощи базы данных MySQL и использования механизма управления сеансом.

Аутентификация средствами управления сеансом

- В примере мы воспользуемся базой данных аутентификации. Это требуется для работы с модулем `mod_auth_mysql`. Пример включает три простых сценария.
- Первый обеспечивает форму для входной регистрации и аутентификации пользователей Web-сайта.
- Второй представляет информацию только для тех пользователей, которые успешно прошли входную регистрацию.
- Третий реализует выход пользователей из системы.

Аутентификация средствами управления сеансом

- Работа первого сценария сосредоточена вокруг переменной сеанса `$valid_user`. Основная идея здесь заключается в следующем: если кто-либо успешно прошел процедуру входной регистрации, мы регистрируем переменную сеанса с именем `$valid_user`, которая содержит идентификатор пользователя.
- При первом проходе по сценарию ни один из условных операторов `if` не сработает. После этого мы предоставляем посетителю форму, при помощи которой он сможет зарегистрироваться.

Аутентификация средствами управления сеансом

- Когда пользователь нажмет кнопку отправки (Submit), сценарий вызывается заново. На этот раз в нашем распоряжении будут имя пользователя и пароль, позволяющие его аутентифицировать (они хранятся в `$userid` и `$password`). Если эти переменные установлены, переходим к блоку аутентификации.
- Далее мы подключаемся к базе данных MySQL и проверяем имя пользователя и пароль. Если в базе данных существует соответствие этой паре, мы регистрируем переменную `$valid_user`, которая содержит идентификатор для конкретного пользователя.

Аутентификация средствами управления сеансом

- Поскольку уже известно, кто сейчас посещает сайт, то повторно предоставлять ему форму входной регистрации нет необходимости. Вместо этого мы сообщаем пользователю, что мы знаем, кто он такой, и даем ему возможность выхода из системы
- Если же при попытке произвести входную регистрацию пользователя, мы по какой-то причине терпим неудачу, то у нас имеется идентификатор пользователя, но нет переменной `$valid_user`, и ничего не остается, кроме как выдать сообщение об ошибке.

Аутентификация средствами управления сеансом

- Поскольку `$valid_user` является зарегистрированной переменной сеанса, ее нельзя перезаписать путем передачи другого значения через URL, например так:
- `members.php?valid_user=testuser`
- Второй сценарий запускает сеанс и проверяет, содержит ли текущий сеанс зарегистрированного пользователя, с использованием функции `session_registered_user()`. Если пользователь прошел процедуру входной регистрации, мы отображаем содержимое сайта, в противном случае сообщаем ему, что у него нет полномочий.

Аутентификация средствами управления сеансом

- В последнем сценарии мы запускаем сеанс, запоминаем старое имя пользователя, отменяем регистрацию переменной `$valid_user` и завершаем сеанс. После этого мы выдаем пользователю одно из следующих сообщений: он вышел из системы, не может выйти из системы или не может выйти из системы, поскольку первоначально даже не регистрировался.