

SESSION в PHP. Основы

- Введение
- Сессии в PHP
- Запуск сессии
- Получение информации о сессии
- Сохранение данных сессии
- Получение данных сессии
- Эксперимент с session_id
- Удаление данных сессии
- JSON в сессиях

Введение

Обработка сессии — это ключевой приём в PHP, который позволяет получить доступ к данным пользователя на любой странице веб-сайта или приложения. В этом разделе мы разберем основы обработки сессий в PHP.

Начну с разбора того, как работают сессии и как они связаны с cookie-файлами. Затем рассмотрим несколько примеров демонстрационного кода работы с сессиями. Разберем способы сохранения, изменения и удаления значений переменных сессий.

Кратко о самом важном

Иногда важно сразу заявить о принципе рассматриваемой технологии. Так вот:

- **Cookie** (куки) – это текстовые данные, хранящиеся на **компьютере пользователя** (клиенте). Обмен информацией между клиентом и сервером происходит путем передачи **данных куки** в HTTP-заголовках.
- **Session** (сессии) – это текстовые данные, хранящиеся на **сервере**. Обмен информацией между клиентом и сервером происходит путем

передачи в HTTP-заголовках **имени серверного файла** (идентификатора), в котором хранятся данные.

Кратко о существующем нюансе

Обмен информацией выполняется путем передачи **идентификатора** серверного файла. В свою очередь, идентификатор передается с помощью **cookie** (куки). Но, как известно, поддержка куки может быть отключена пользователем. В таком случае **идентификатор** может передаваться через URL или форму, например, подставляясь в строку запроса в качестве **GET-параметра**.

А теперь подробности и примеры.

Сессии в PHP

Сессия — это механизм сохранения информации на разных веб-страницах для идентификации пользователя пока он сёрфит по сайту или приложению.

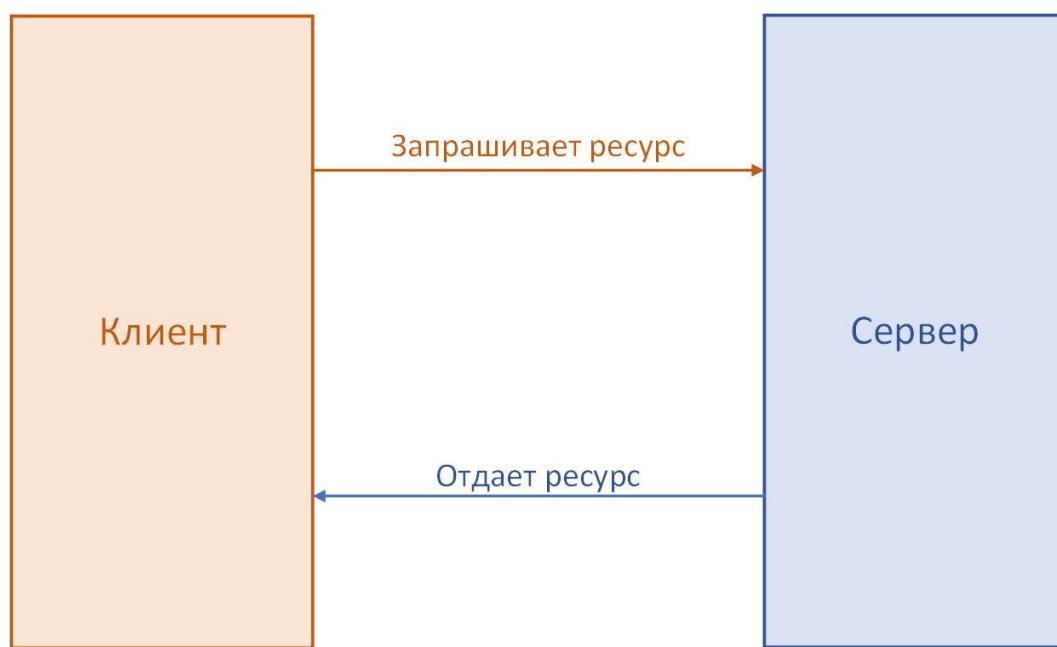
Информация сессии — это набор переменных, которые хранятся на сервере (либо часть на сервере, а часть - в cookie браузера) и которые относятся только к **текущему пользователю**.

Чтобы лучше разобраться в работе сессий необходимо вспомнить, как работает **HTTP-протокол**.

Протокол HTTP — протокол без сохранения состояния. Это означает, что сервер не может сопоставить конкретного пользователя по выполненным запросам. Каждый запрос можно рассматривать как переход между страницами сайта. Запрос может быть и без выполнения перехода, но сейчас это не принципиально.

Например, при обращении к Интернет-ресурсу, веб-сервер отвечает за предоставление контента запрашиваемого ресурса. Поэтому, когда пользователь обращается к разным страницам одного и того же веб-сайта, сервер интерпретирует каждый запрос отдельно, как если бы они не были связаны друг с другом и выполнялись разными пользователями.

На следующей диаграмме изображен общий принцип клиент-серверного взаимодействия по протоколу HTTP.



Важно. Серверу неизвестно, что каждый запрос исходит от одного и того же пользователя.

И в этом нет никакой проблемы, до тех пор, пока вы не захотите отобразить конфиденциальные персональные данные, когда придется **аутентифицировать пользователя в каждом запросе** (при каждом переходе между страницами).

Примечание. Представьте работу с сайтом (приложением), в котором вам нужно вводить **логин** и **пароль** на каждой странице с информацией о ваших персональных данных.

Это было бы громоздко и не практично, и именно здесь на помощь приходят сессии.

Сессия позволяет обмениваться информацией между разными страницами одного сайта или приложения, и помогает поддерживать состояние. Это позволяет серверу знать, что все запросы исходят от одного и того же пользователя, что позволяет сайту отображать конфиденциальную информацию и настройки пользователя.

Примечание. А еще это позволяет сохранять информацию о ваших предпочтениях и показывать ее в виде контекстной рекламы

Часто можно услышать вместо **сессии** слово – **сеанс**. Четкой границы между этими понятиями нет, предлагаю считать **сеанс** и **сессию** словами синонимами. Впрочем Интернет, может предложить и такую версию:

- сессия может длиться очень долго. Пока на сервере и на клиенте хранится ее идентификатор и пока не исчерпан лимит времени его жизни.
- термин сеанс больше относится к работе пользователя. В пределах одной сессии пользователь может выполнить несколько сеансов.

Клиент-сервер с сессиями и cookie

Рассмотрим общий пример входа на веб-сайт более подробно.

1. Пользователь открывает страницу с **формой входа** на веб-сайт.
2. После отправки формы входа клиентом, сервер пытается **аутентифицировать пользователя**, проверяя введённые им учётные данные.

3. Если учётные данные, введённые пользователем, верны, сервер создаёт новую сессию, при этом генерируется уникальное случайное число, которое называется **идентификатором сессии**. Идентификатор является частью имени для файла, создаваемого на сервере. Файл используется для хранения информации, относящейся к сеансу.
4. Идентификатор сеанса передаётся обратно пользователю, вместе с запрошенным им ресурсом. Отправка идентификатора происходит в заголовке ответа **cookie PHPSESSID** (имя cookie по умолчанию).
5. Когда браузер получает ответ от сервера, он получает заголовок cookie-файла PHPSESSID. Если в браузере разрешены cookie, то он сохранит этот PHPSESSID, в котором хранится идентификатор сеанса, переданный сервером.
6. Для последующих запросов, cookie **PHPSESSID** передаётся обратно на сервер. Когда сервер получает **cookie**, он пытается **инициализировать сеанс с этим идентификатором**. Инициализация означает загрузку файла, который был создан ранее во время создания сеанса.
7. После этого сервер инициализирует суперглобальную переменную массива **\$_SESSION** с данными, хранящимися в файле сеанса.

На диаграмме показано, как HTTP протокол работает с сессиями.



Таким образом, пользовательские данные сохраняются даже в нескольких запросах, и пользователь не теряется на протяжении всего **сеанса**.

Запуск сессии

Всякий раз, когда вы хотите поработать с переменными сессий, необходимо убедиться, что **сессия запущена**. Даже если вы собираетесь удалить сессию, сначала ее надо запустить. Сессии в PHP можно запустить двумя способами.

Запуск сессии функцией `session_start()`

Один из способов запуска сессий – использование функции `session_start()`.

Важно, чтобы функция `session_start()` вызывалась в начале скрипта, перед отправкой чего-либо браузеру. В противном случае, вы рискуете столкнуться с уже известной ошибкой "Headers are already sent".

Важно. Функция **session_start()** должна вызываться до отправки любого контента пользователю (пустой текстовый узел #text – тоже контент).

Пример запуска сессии:

```
<?php  
    session_start();
```

При запуске сессии с помощью функции **session_start()** возможны следующие варианты:

- **Пользователь первый раз заходит на сайт.** Функция назначает ему **уникальный идентификатор сессии**. Этот идентификатор с помощью **cookie**, который по умолчанию называется **PHPSESSID**, сохраняется в браузере пользователя. В дальнейшем, с помощью этого идентификатора пользователь ассоциируется с данными сессии.
- **Для пользователя уже установлена сессия.** Функция продлевает текущую сессию вместо установки новой.

Итак. Функция **session_start()**:

1. Создает **уникальный идентификатор сессии**.
2. Сохраняет созданный идентификатор в **cookie** с именем **PHPSESSID**.
3. Отправляет созданный **cookie** пользователю.

Все сказанное верно, если пользователь не отключил поддержку **cookie**. Алгоритм работы функции при выключенном **cookie** рассмотрим в следующе .

Автоматический запуск сессии

При необходимости применения сессий в приложении, есть возможность запускать сеанс **автоматически** без использования функции **session_start()**.

Конфигурационный PHP-файл PHP.INI.

В конфигурационном PHP-файле **php.ini** есть параметр **session.auto_start**, который позволяет запускать сеанс автоматически для каждого запроса. По умолчанию установлено значение 0 (выключено), вы можете установить его на 1 (включено), чтобы включить функцию **автоматического запуска**.

Конфигурационный PHP-файл .HTACCESS.

С другой стороны, если у вас нет доступа к файлу **php.ini**, и вы используете веб-сервер Apache, эту переменную можно задать с помощью файла **.htaccess**. Если вы добавите параметр **session.auto_start** в ваш **.htaccess** файл, то это будет автоматически запускать сессии в вашем PHP-приложении.

Важно. Идентификатор сессии передается через **cookie**, поэтому при **тестировании** демонстрационных примеров не забывайте **очищать браузер от данных cookie**. Контролируйте **cookie** с помощью режима разработчика.

Получение информации о сессии

С помощью специальных функций мы можем получить следующую информацию о созданной сессии:

- **имя** сессии
- значение **идентификатора** сессии
- **путь** к файлу сессии

<?php

```
// старт сессии

session_start();

// имя сессии

echo session_name(); // по умолчанию - PHPSESSID

// идентификатор сессии

echo session_id();
```

Немного информации об используемых далее функциях.

session_start

session_start() — стартует новую сессию, либо возобновляет существующую.

Описание

session_start(\$options = []):

Функция **session_start()** создаёт сессию, либо возобновляет существующую, основываясь на идентификаторе сессии, переданном через **GET**- или **POST**-запрос, либо переданный через **cookie**.

Когда вызвана функция **session_start()** или когда сессия создаётся автоматически, PHP извлечёт все существующие данные сессии (сохранённые в специальном сериализованном виде), **десериализует** их и занесёт в суперглобальный массив **\$_SESSION**.

Параметры

- **options** – если задано, то должно быть ассоциативным массивом, переопределяющим текущие директивы конфигурации сессий.

Возвращаемые значения

Функция возвращает **true**, если сессия успешно стартована, в противном случае **false**.

session_name

session_name() — получает или устанавливает имя текущей сессии.

Описание

```
session_name($name = null);
```

Функция **session_name()** возвращает имя текущей сессии. Если задан параметр **name**, **session_name()** обновит имя сессии и вернёт старое имя сессии.

Если новое имя сессии (**name**) предоставлено, **session_name()** изменяет сессионный cookie. **session_name()** необходимо вызывать до **session_start()** для правильной работы сессии.

Имя сессии сбрасываются на значение по умолчанию, хранящегося в **session.name()** во время запуска запроса. Таким образом, вызывать **session_name()** нужно для каждого запроса (и до **session_start()**).

Параметры

- **name** – имя сессии, ссылается на имя, которое используется в cookie и URL (например, PHPSESSID). Должно содержать только **буквенно-цифровые символы**, и должен быть коротким и понятными (например, для пользователей с включённым предупреждением cookie). Если задан параметр **name** и он не равен **null**, имя текущей сессии поменяется на него.

Важно. Имя сессии **не может состоять только из цифр**, по крайней мере, одна буква должна присутствовать. В противном случае каждый раз будет генерироваться новый идентификатор

Возвращаемые значения

Возвращает **имя текущей сессии**. Если задан параметр **name**, имя текущей сессии поменяется и будет возвращено старое или **false** в случае возникновения ошибки.

session_id

session_id() — получает и/или устанавливает идентификатор текущей сессии

Описание

```
session_id($id = null);
```

Функция **session_id()** используется для получения или установки **идентификатора текущей сессии (SID)**.

Константа **SID** также может быть использована для получения текущего имени и идентификатора сессии в виде строки, подходящей для добавления в URL-адреса.

Параметры

- **id** – если указан параметр **id** и он не равен **null**, то он заменит идентификатор текущей сессии. Для этого **session_id()** следует вызывать до **session_start()**. Идентификатор сессии содержит ограничения на используемые в нем символы.

Важно. При использовании сессионных **cookie**, указание **id** для **session_id()** приводит к тому, что при вызове **session_start()** **всегда будут отправлены новые cookie**, независимо от того, совпадает ли идентификатор текущей сессии с вновь установленным.

Возвращаемые значения

session_id() возвращает **идентификатор текущей сессии** или пустую строку (""), если нет текущей сессии (идентификатор текущей сессии не существует). В случае неудачи возвращает **false**.

Идентификатор сессии является **частью имени файла на сервере**, в котором будут храниться данные. Путь к файлу можно посмотреть используя функцию **session_save_path()**.

```
<?php  
    // путь к серверному файлу  
    echo session_save_path();
```

Кроме функции **session_id()** **идентификатор сессии** можно получить, обратившись к **cookie** напрямую:

```
<?php  
    // идентификатор сессии  
    echo $_COOKIE["PHPSESSID"];
```

Важно помнить, данные **cookie** мы получаем из HTTP-заголовков, а следовательно, идентификатор сессии из **cookie** будет доступен только при перезагрузке страницы.

В демонстрационном примере **example_1** приведен пример старта работы с сессиями.

example_1. index.php

```
<?php  
    // стартируем сессию  
    session_start();  
    // получим идентификатор сессии  
    echo 'Идентификатор сессии: <b>' . session_id() . '</b><p>';  
    // получим имя сессии  
    echo 'Имя сессии по умолчанию: <b>' . session_name() . '</b><p>';
```

```

'</b><p>' ;

// получим путь хранения файла с переменными сессий

echo 'Хранение файла сессии: <b>' . session_save_path() . '</b><p>' ;

// значение идентификатора можно получить напрямую из cookie

if (isset($_COOKIE['PHPSESSID'])) {

    echo 'Идентификатор сессии (cookie): <b>' .
        $_COOKIE['PHPSESSID'] . '</b>' ;

    else

        echo '<b>Для получения данных cookie перезагрузите
страницу!</b>' ;

    // массив cookies будет доступен после перезагрузки страницы

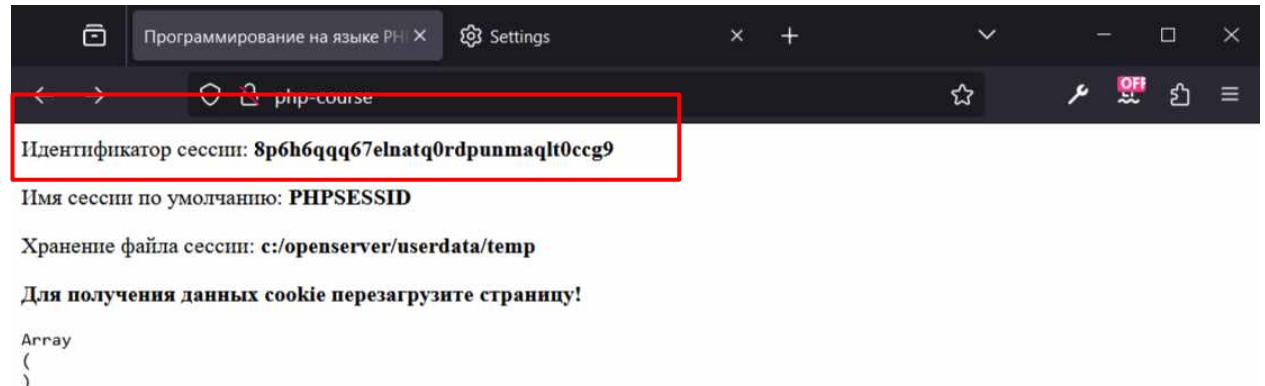
    echo '<pre>' ;

    print_r($_COOKIE);

    echo '</pre>' ;
}

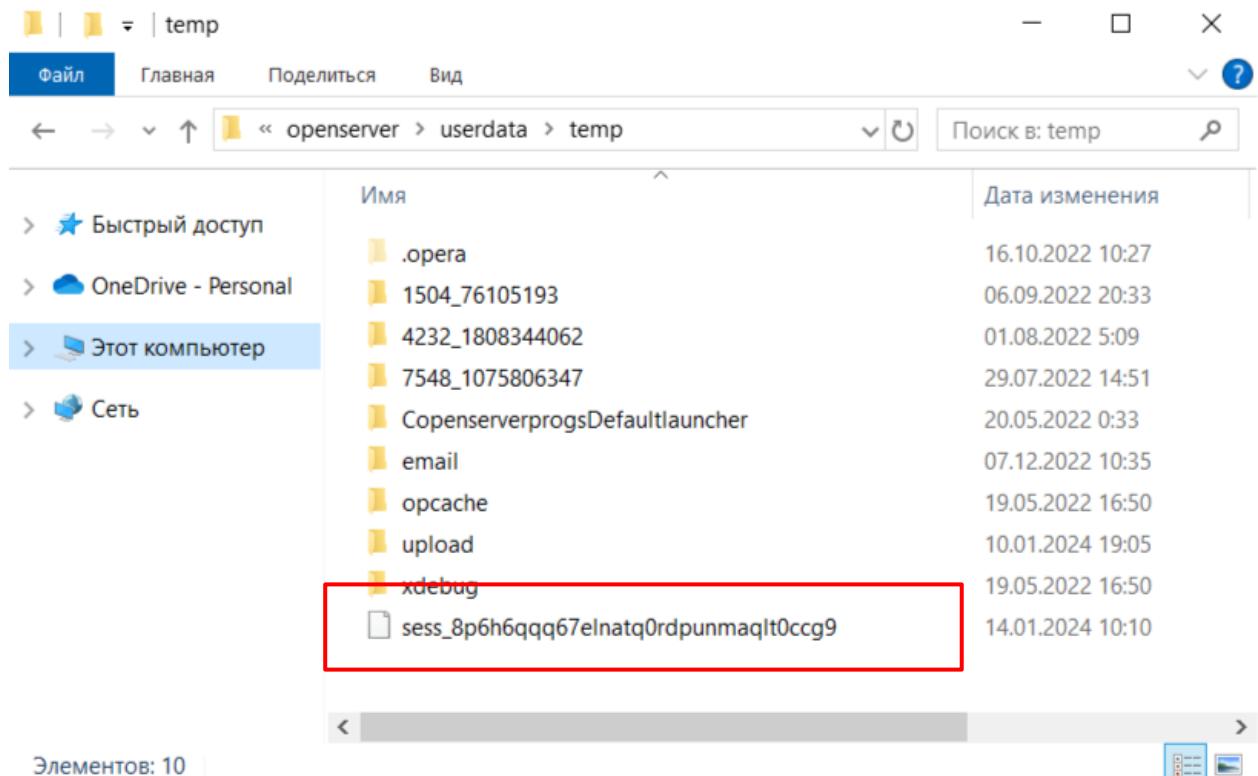
?>

```



Безопасность данных

На следующем скриншоте показана директория хранения сессионных файлов сервера.



Важно. Что бы вы не делали с сессиями, помните – **идентификатор сессии** (PHPSESSID) передается через **cookie**. Нет **cookie** – не будет и **сессий**.

В реальности дела обстоят несколько сложнее, но об этом в следующем разделе.

Сохранение данных сессии

Для сохранения или получения данных в сессии необходимо использовать **глобальный ассоциативный массив \$_SESSION**.

При сохранении или получении данных PHP **автоматически сериализует / десериализует** хранящиеся в файле данные сессии (что очень удобно).

Сохранение значения в сессии:

```
<?php  
    // в массиве с ключом role сохраняем строковое значение  
    $_SESSION["role"] = 'author';
```

Сохранение в массиве строковых значений

example_2. index.php

```
<?php  
  
    // стартируем сессию  
  
    session_start();  
  
    // получаем характеристики сессии  
  
    echo 'Идентификатор сессии: <b>' . session_id() . '</b>';  
  
    echo "<p>";  
  
    echo 'Имя сессии: <b>' . session_name() . '</b>';  
  
    // сохраняем в сессию значение логина  
  
    $_SESSION['login'] = 'DenisDream';  
  
    // сохраняем в сессию значение пароля  
  
    $_SESSION['pwd'] = '12345@%';  
  
    // данные сессии  
  
    echo '<h2>Данные сессии</h2>';  
  
    echo '<pre>';  
  
    print_r($_SESSION);  
  
    echo '</pre>';
```

```
?>
```

На рисунке представлен **вывод** сценария **example_2** и **сериализованные** данные файла сессии. Сериализованы два строковых значения для ключей **login** и **pwd**.

The screenshot shows a browser window titled "Программирование на языке PHP". The address bar says "php-course". The page content includes:

Идентификатор сессии: 8ehbbj9qbfvsem6dg5ll598ndkl1ojgm
Имя сессии: PHPSESSID

Данные сессии

```
Array
(
    [login] => DenisDream
    [pwd] => 12345@%
```

Безопасность

sess_8ehbbj9qbfvsem6dg5ll598ndkl1ojgm – Блокнот

Файл Правка Формат Вид Справка

```
Login|s:10:"DenisDream";pwd|s:7:"12345@%" ;
```

Сохранение в массиве структурированных данных

В сессии можно хранить структуры любой сложности, в зависимости от степени сложности выполняемой задачи.

example_3. index.php

```
<?php

    // стартируем сессию

    session_start();

    // получаем характеристики сессии

    echo 'Идентификатор сессии: <b>' . session_id() . '</b>';

    echo "<p>";

    echo 'Имя сессии: <b>' . session_name() . '</b>';
```

```

// сохраняем в ключ user массив со значениями логин / пароль

$_SESSION['user'] = [
    'login' => 'DenisDream',
    'pwd' => '12345@%'
];

// данные сессии

echo '<h2>Данные сессии</h2>';
echo '<pre>';
print_r($_SESSION);
echo '</pre>';

?>

```

На рисунке представлен **вывод** сценария **example_3** и **сериализованные** данные файла сессии. Обратите внимание, на этот раз в файле сессии хранится сериализованный массив.

```

Идентификатор сессии: b891c2satldtg5g2dqeut2geekhnaopg
Имя сессии: PHPSESSID

Данные сессии

Array
(
    [user] => Array
        (
            [login] => DenisDream
            [pwd] => 12345@%
        )
)

```

Безопасность данн

Важно. Сериализация / десериализация данных сессий происходит **автоматически**.

Для сохранения данных сессии можно создать **идентификатор** сессии **самостоятельно**.

example_4. index.php

```
<?php

    // набор символов для генерации идентификатора сессии

    $chars = '0123456789abcdefghijklmnopqrstuvwxyz';

    $session_id = substr(str_shuffle($chars), 0, 15);

    // устанавливаем пользовательский идентификатор

    session_id($session_id);

    // стартируем сессию

    session_start();

    // в сессиях можно сохранять что-нибудь умное ;)

    $_SESSION['author'] = 'Сократ';

    $_SESSION['citeat'] = 'Предпочитайте знание богатству, ибо
одно преходяще, а другоеечно';

    // получаем характеристики сессии

    echo 'Идентификатор сессии: <b>' . session_id() . '</b>';

    echo "<p>";

    echo 'Имя сессии: <b>' . session_name() . '</b>';

    // данные сессии

    echo '<h3>Данные сессии</h3>';

    echo '<pre>';

    print_r($_SESSION);

    echo '</pre>';
```

```
?>
```

Получение данных сессии

Получение сохраненного значения переменной сессии:

```
<?php  
    // сохраняем в переменную значение сессии role  
    $role = $_SESSION["role"];
```

Не важно сохраняем мы данные или получаем их, работа начинается со старта сессии.

example_5. index.php

```
<?php  
    // стартируем сессию  
    session_start();  
  
    // создаем сессионные данные  
    $_SESSION['login'] = 'superRoot';  
    $_SESSION['pwd'] = '@master@';  
  
    // данные сессии  
    echo '<h3>Данные сессии</h3>';  
    echo '<pre>';  
    print_r($_SESSION);  
    echo '</pre>';  
  
?>  
<!-- ... -->  
<a href='target.php'>Переход к файлу target.php</a>
```

example_5. target.php

```
<?php

    // стартируем сессию

    session_start();

    // характеристики сессии

    echo '<h3>Информация о сессии</h3>';

    echo 'Идентификатор сессии: ' . session_id();

    echo "<p>";

    echo 'Имя сессии: ' . session_name();

    // получаем данные сессии

    $login = $_SESSION['login'];

    $pwd = $_SESSION['pwd'];

    // выводим данные сессии

    echo '<h3>Данные сессии</h3>';

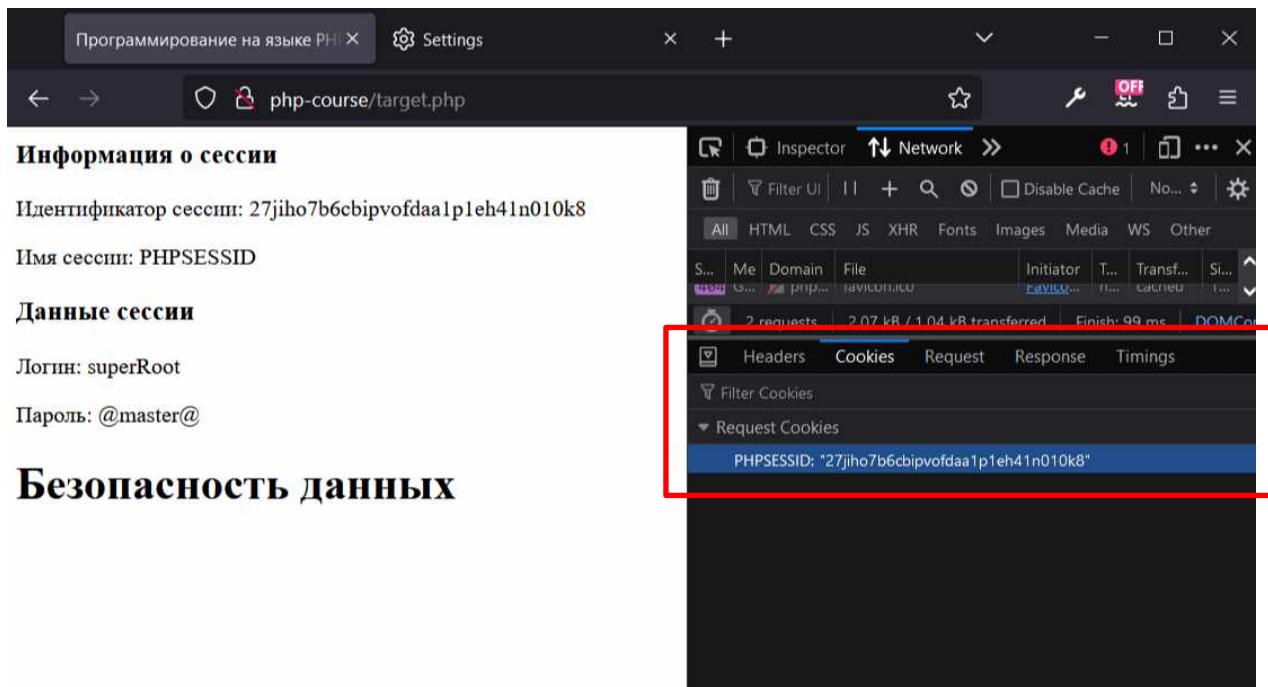
    echo 'Логин: ' . $login . '<p>';

    echo 'Пароль: ' . $pwd;

?

>
```

Результат вывода сценария **target.php** представлен на следующем скриншоте. Обратите внимание на **сессионную cookie** HTTP-заголовка.



Эксперимент с *session_id*

Теперь, когда мы умеем сохранять в сессию и получать из сессии данные, выполним один эксперимент. Весьма поучительный, кстати ;)

Смысл задания демонстрационного примера **example_6** смотрите в скриншотах ниже.

example_6. index.php

```
<?php

    // генерируем идентификатор сессии

    // в учебных целях сделаем его простым

    $id = rand(1, 10);

    session_id($id);

    // стартируем сессию с session_id = $id

    session_start();

    // проверяем сессию с текущим идентификатором

    echo "<h3>Стартовали сессию с id = " . session_id() . "
```

```

    "</h3>";

echo '$_SESSION["id"] = ' . $_SESSION["id"];

// сохраним идентификатор в сессию

$_SESSION['id'] = $id;

// вывод данных текущей сессии

echo "<h3>Данные сессии</h3>";

echo '$_SESSION["id"] = ' . $_SESSION["id"];

?>

```

Перед анализом кода откройте вместе с браузером директорию хранения файлов сессий.

Первая загрузка страницы. Сгенерирован **идентификатор сессии** (`id=7`). Стартуется сессия с текущим идентификатором, в текущей сессии нет ключа **id** для массива `$_SESSION`.

```

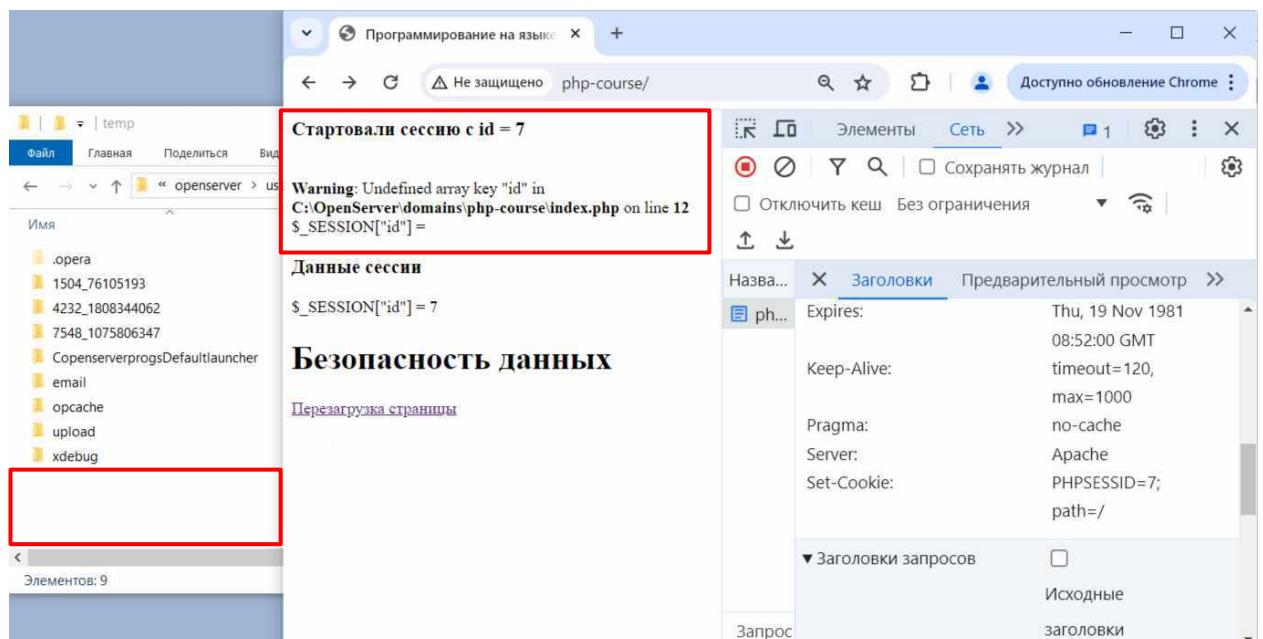
// проверяем сессию с текущим идентификатором

echo "<h3>Стартовали сессию с id = " . session_id() . "</h3>";

echo '$_SESSION["id"] = ' . $_SESSION["id"];

```

Получаем ошибку.



Обновим страницу. И вновь получаем новый идентификатор, а значит стартуем новую сессию. Каждый раз при обращении к `$_SESSION["id"]` получаем одну и ту же ошибку:

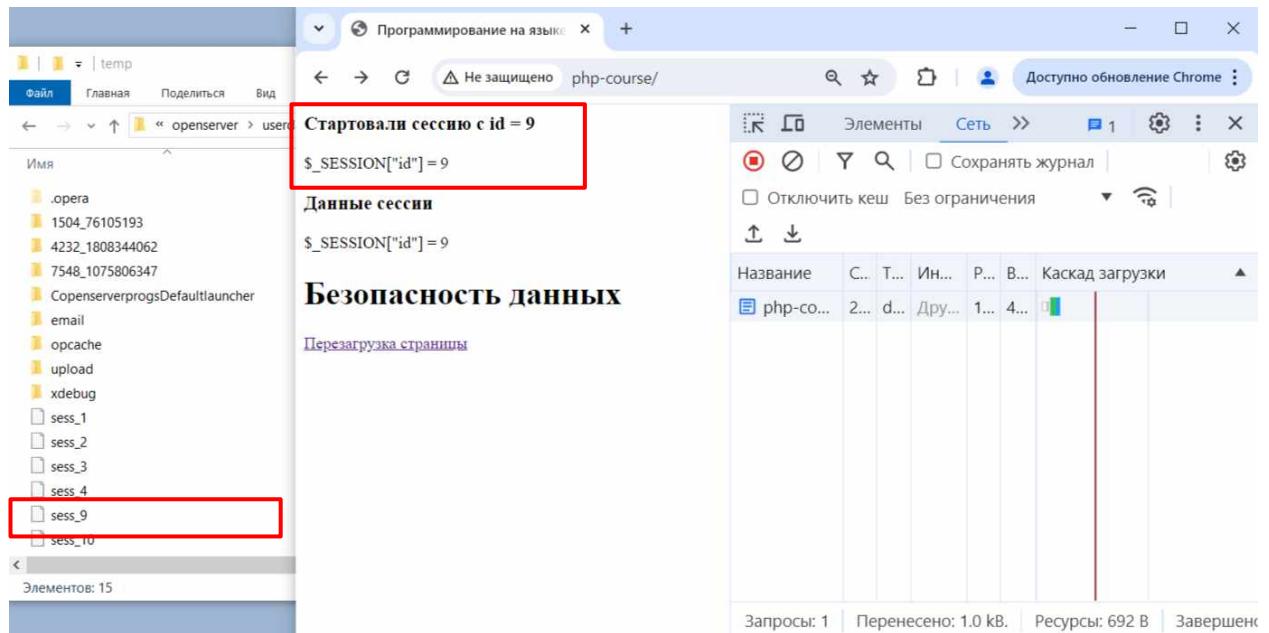
```
Warning. Undefined array key id in ...
```

Стоп. На одном из обновлений сгенерирован идентификатор с `id=9`. Такой идентификатор уже был, а значит файл сессии уже был создан и хранится на сервере! А следовательно строка кода:

```
echo '$_SESSION["id"] = ' . $_SESSION["id"];
```

теперь ошибку не вызовет.

Таким образом, в ход выполнения сценария вмешались данные, сохраненные предыдущим выполнением программного кода.



Данные сессий необходимо **своевременно удалять!**

Удаление данных сессии

В этом разделе мы увидим, как можно уничтожить сессию. Сессия уничтожается с закрытием браузера, однако мы также можем программно удалить либо какие-то отдельные, либо все данные сессии.

Если вы хотите удалить сразу все данные, связанные с сеансом, вы можете использовать функцию **session_destroy()**.

```
<?php  
    // старт сессии  
    session_start();  
  
    // удаление сессии  
    session_destroy();
```

Функция **session_destroy()** удаляет всё, что хранится в текущем сеансе. Таким образом, с последующими запросами вы увидите пустую переменную **\$_SESSION**, поскольку данные сеанса, хранящиеся на диске, были удалены.

Важно. Как правило, функция **session_destroy()** используется, когда пользователь выходит из системы.

session_destroy

session_destroy() — уничтожает все данные сессии.

Описание

```
session_destroy();
```

Функция **session_destroy()** уничтожает все данные, связанные с текущей сессией. Данная функция не удаляет какие-либо **глобальные переменные**, связанные с сессией, и **не удаляет сессионные cookie**. Чтобы вновь использовать переменные сессии, следует вызвать **session_start()**.

Важно. Функция **session_destroy()** не взаимодействует с заголовками, а поэтому, в отличии от **session_start()** вызывать

функцию (дестроить сессию) не обязательно в начале скрипта (но обязательно после **session_start()**).

Примечание. Нет необходимости вызывать **session_destroy()** в обычном коде. Вместо удаления данных достаточно очистить массив **\$_SESSION**.

Чтобы полностью удалить сессию, также необходимо удалить и её идентификатор. Если для передачи идентификатора сессии используются **cookie** (поведение по умолчанию), то сессионные **cookie** также должны быть удалены. Для этого можно использовать **setcookie()**.

Примечание. Немедленное удаление сессии может привести к нежелательным последствиям. При наличии конкурирующих запросов, другие соединения могут столкнуться с внезапной потерей данных сессии, например, это могут быть асинхронные запросы от JavaScript.

Чтобы избежать нежелательных последствий правильнее будет установить в **\$_SESSION** временную метку удаления и убрать доступ позже. Или удостовериться, что ваше приложение не имеет конкурирующих запросов.

Параметры

У функции нет параметров.

Возвращаемые значения

Возвращает **true** в случае успешного выполнения или **false** в случае возникновения ошибки.

Важно. Сессионный cookie **PHPSESSID** необходимо удалить

самостоятельно.

Для удаления **одной переменной** из сессии применяется функция **unset()**:

```
<?php  
    // старт сессии  
  
    session_start();  
  
    // удаляем из сессии переменную role  
  
    unset($_SESSION["role"]);
```

Кроме того, с помощью функции **session_unset()** можно удалить **все** зарегистрированные переменные текущей сессии.

```
<?php  
    // старт сессии  
  
    session_start();  
  
    // очищение сессии  
  
    session_unset();
```

session_unset

session_unset() — удалить все переменные сессии.

Описание

```
session_unset();
```

Функция **session_unset()** удаляет все зарегистрированные переменные текущей сессии.

Параметры

У функции нет параметров.

Возвращаемые значения

Возвращает **true** в случае успешного выполнения или **false** в случае возникновения ошибки.

Попробуем понять, как это работает в следующих демонстрационных примерах. В **example_6-example_7** напишем сценарий управления сессионными данными приложения.

example_7. index.php

```
<?php

    // стартируем сессию

    session_start();

    // создаем сессионный массив данных

    $_SESSION = array(
        'id' => 23,
        'name' => 'Тимофеев Илья',
        'course' => 'JavaScript разработчик с нуля',
        "time" => '7 месяцев',
        "shedule" => 'Будние дни 18:00-19:30',
        "level" => 'Junior',
        "teacher" => 'Данилов В.Е.',
        "price" => 40000
    );

?>

<!-- -->

<form action="manager.php" method="post">

    <?php
        // выведем сессионный массив в форму
    ?>

```

```

foreach ($_SESSION as $key => $val) {

    echo ""

    $key <input type='text' name='data[$key]'  

    value='$val'>

    <input type='checkbox' name='sel[$key]'><p>" ;

}

?>

<button name="del-sel">Удалить выбранное</button>

</form>

```

example_7. manager.php

```

<?php

// стартируем сессию

session_start() ;

// если есть данные для удаления

if (isset($_POST['sel'])) {

    foreach ($_POST['sel'] as $key => $val) {

        // удаляем сессионную переменную

        unset($_SESSION[$key]) ;

    }

}

// данные сессии

echo '<h3>Данные сессии</h3>' ;

echo '<pre>' ;

print_r($_SESSION) ;

echo '</pre>' ;

?>

```

Добавим функционал удаления сразу всех переменных сессии.

example_8. index.php

```
<?php

    // стартируем сессию

    session_start();

    // создаем сессионный массив данных

    $_SESSION = array(
        'id' => 23,
        'name' => 'Тимофеев Илья',
        'course' => 'JavaScript разработчик с нуля',
        "time" => '7 месяцев',
        "shedule" => 'Будние дни 18:00-19:30',
        "level" => 'Junior',
        "teacher" => 'Данилов В.Е.',
        "price" => 40000
    );

?>

<!-- ... -->

<form action="manager.php" method="post">

    <?php

        // выведем сессионный массив в форму

        foreach ($_SESSION as $key => $val) {
            echo "
                $key <input type='text' name='data[$key]' value='$val'>
                <input type='checkbox' name='sel[$key]'><p>";
        }
    
```

```

?>

<button name="del-sel">Удалить выбранное</button>
<button name="del-all">Удалить все</button>
</form>

```

example_8. manager.php

```

<?php

// стартируем сессию

session_start();

// если отправлена форма

// если нажата кнопка - Удалить все

if (isset($_POST['del-all'])) {

    // очищаем сессионные переменные

    session_unset();

    // удаляем сессию

    session_destroy();

    // удаляем сессионный cookie

    setcookie('PHPSESSID', '', time() -3600);

}

// если пытаемся удалить выбранное

elseif (isset($_POST['sel'])) {

    foreach ($_POST['sel'] as $key => $val) {

        // удаляем сессионную переменную

        unset($_SESSION[$key]);

    }

}

// данные сессии

```

```
echo '<h3>Данные сессии</h3>';

echo '<pre>';

print_r($_SESSION);

echo '</pre>';

?>
```

В демонстрационном примере **example_9** закольцуем страницы приложения ссылками. Отследите **передачу идентификатора сессии** при переходе между страницами.

example_9. index.php

```
<?php

// генерируем идентификатор сессии

// в учебных целях сделаем его простым

$id = rand(1, 10);

session_id($id);

// стартируем сессию

session_start();

echo 'Идентификатор сессии: ' . session_id() . "<br>";

echo 'Имя сессии: ' . session_name();

// начальные данные сессии

echo "<h3>Начальные данные сессии</h3>";

echo "<pre>";

print_r($_SESSION);

echo "</pre>";

// сохраняем массив в сессию

$_SESSION['team'] = array (
    'name' => "ZZ Top",
```

```

'alias' => "zz-top"
);

// данные сессии которые доступны на текущей странице

echo "<h3>Данные сессии</h3>";

echo "<pre>";

print_r($_SESSION);

echo "</pre>";

?>

<!-- ... -->

<a href='page-1.php'>Идем на первую страницу сайта</a>

```

example_9. page-1.php

```

<?php

// стартируем сессию

session_start();

echo 'Идентификатор сессии: ' . session_id() . "<br>";

echo 'Имя сессии: ' . session_name();

// сохраняем в массив сессии новое значение

$_SESSION['team']['country'] = "США (штат Техас)";

// данные сессии которые доступны на текущей странице

echo "<h3>Данные сессии</h3>";

echo "<pre>";

print_r($_SESSION);

echo "</pre>";

?>

<!-- ... -->

<a href='page-2.php'>Идем дальше</a>

```

На последней странице сохраним содержимое, а затем удалим все данные и саму сессию.

example_9. page-2.php

```
<?php

    // стартируем сессию

    session_start() ;

    // сохраняем в существующий массив сессии новое значение

    $_SESSION['team']['content'] = "Культовая американская блюз-
рок-группа, основанная в 1969 году в Хьюстоне, штат Техас.
Группа называет себя также «that li'l ol' band from Texas»";

    $team = $_SESSION['team'];

    $sessid = session_id();

    // уничтожаем сессию

    session_unset() ;

    session_destroy() ;

    // удаляем сессионный cookie

    setcookie('PHPSESSID', '', time() - 3600, '/');

    echo 'Идентификатор сессии: ' . $sessid . "<br>";

    echo 'Имя сессии: ' . session_name() ;

    // данные сессии которые доступны на текущей странице

    echo "<h3>Данные сессии</h3>";

    echo "<pre>";

    print_r($team) ;

    echo "</pre>";

?>

<!-- ... -->

<a href='/'>На стартовую</a>
```

В демонстрационном примере **example_10** выполним сбор данных в сессию и вывод данных сессии в форму.

example_10. index.php

```
<?php

    // стартируем сессию

    session_start();

    // создаем сессионный массив данных

    $_SESSION['team'] = array (
        'name' => "The Who",
        'alias' => "the-who"
    );

    // данные сессии

    echo "<h3>Данные сессии</h3>";
    echo "<pre>";
    print_r($_SESSION);
    echo "</pre>";

?>

<!-- ... -->

<a href='page.php'>Идем на следующую страницу сайта</a>
```

example_10. page.php

```
<?php

    // стартируем сессию

    session_start();

    // добавляем в сессию данные

    $_SESSION['team']['country'] = "Великобритания";
    $_SESSION['team']['content'] = "Британская рок-группа,"
```

сформированная в 1964 году. Первоначальный состав состоял из Пита Таунсенда, Роджера Долтри, Джона Энтистла и Кита Муна. Группа приобрела огромный успех за счёт неординарных концертных выступлений и считается одной из самых... ";

```
// данные сессии

echo "<h3>Данные сессии</h3>";

echo "<pre>";

print_r($_SESSION);

echo "</pre>";

?>

<!-- ... -->

<a href='/form.php'>Переход к форме</a>
```

example_10. form.php

```
<?php

// стартируем сессию

session_start();

?>

<!-- ... -->

<?php

// выводим данные сессии в форму

echo <<<HERE

<form action="server.php" method="post"
enctype="multipart/form-data">

Название: <input type="text" name="name"
value="{$SESSION['team']['name']} "><p>

Алиас: <input type="text" name="alias"
value="{$SESSION['team']['alias']} "><p>
```

```
Страна: <input type="text" name="country"  
value="{$SESSION['team']['country']}><p>  
<textarea  
name="content">{$SESSION['team']['content']}</p>  
<input type="submit">  
</form>  
  
HERE;  
?>
```

JSON в сессиях

PHP автоматически сериализует содержимое массива `$_SESSION`, но никто не запрещает хранить в массиве данные формата **JSON**. В следующем примере сохраним и передадим между страницами сессионные данные формата **JSON**.

example_11. index.php

```
<?php  
  
    // стартируем сессию  
  
    session_start();  
  
?>  
  
<!-- ... -->  
  
<?php  
  
    // создаем двумерный массив  
  
    $teams = array (   
  
        array (   
  
            'id' => null,  
  
            'name' => "Geordie",
```

```

    'alias' => "geordie",
    'country' => "Великобритания",
    'content' => "Британская рок-группа, образовавшаяся в
начале 1970-х годов в Ньюкасле, Англия, и исполнявшая
хард-рок с элементами глэма и блюз-рока."
),
array (
    'id' => null,
    'name' => "Queen",
    'alias' => "queen",
    'country' => "Великобритания",
    'content' => "Британская рок-группа, добившаяся
широкайшей известности в середине 1970-х годов, и одна из
наиболее успешных групп в истории рок-музыки."
)
);

// преобразуем массив в JSON-представление
// русские символы не кодируем
$teamsJSON = json_encode($teams, JSON_UNESCAPED_UNICODE);
$_SESSION['json'] = $teamsJSON;
?>

<p><a href='target.php'>Перейти</a>

```

example_11. target.php

```

<?php
    // стартируем сессию
    session_start();
?>

```

```
<!-- ... -->

<?php

    // декодируем данные массива

    $teams = json_decode($_SESSION['json'], 1);

    echo "<pre>";
    print_r($teams);
    echo "</pre>";

?>
```

Кроме стандартной возможности кодирования значений в формат JSON сессии имеют **специальные функции**:

- **session_encode()**
- **session_decode()**

session_encode

session_encode() — кодирует данные текущей сессии в формате строки сессии.

Описание

```
session_encode();
```

Функция **session_encode()** возвращает **сериализованную строку**, содержащую данные текущей сессии, хранящиеся в суперглобальном массиве **\$_SESSION**.

По умолчанию используется внутренний метод сериализации PHP и результат будет отличаться от формата, возвращаемого функцией **serialize()**.

Параметры

У функции нет параметров.

Возвращаемые значения

Возвращает **сериализованные данные** текущей сессии или **false** в случае возникновения ошибки.

session_decode

session_decode() — декодирует данные сессии из закодированной строки сессии.

Описание

```
session_decode(string $data);
```

Функция **session_decode()** декодирует сериализованные данные сессии, переданные в `$data` и заполняет суперглобальный массив `$_SESSION` полученным результатом.

По умолчанию используется внутренний метод десериализации PHP, и результат будет отличаться от формата, возвращаемого функцией **unserialize()**.

Параметры

- **data** – закодированные данные, которые необходимо сохранить.

Возвращаемые значения

Возвращает **true** в случае успешного выполнения или **false** в случае возникновения ошибки.

example_12. index.php

```
<?php  
    // стартируем сессию  
    session_start();  
?>
```

```

<!-- ... -->

<?php

    // сохраняем массив в сессию

    $_SESSION['teams'] = array (
        'id' => null,
        'name' => "Deep Purple",
        'alias' => "deep-purple",
        'country' => "Великобритания",
        'content' => "Британская рок-группа, образованная в
феврале 1968 года в Хартфорде, Англия, и считающаяся
одной из самых заметных и влиятельных в хард-роке 1970-х
годов."
    ) ;

    // сериализуем сессию

    $data = session_encode();

    echo '<h3>Данные сессии сериализованы:</h3>';
    echo $data;
?>

<p><a href='target.php?s=<?= $data ?>'>Перейти</a>

```

Обратите внимание, я обнулил сессию, но затем успешно восстановил ее из строки запроса.

example_12. target.php

```

<?php

    // стартируем сессию

    session_start();

    // обнулим данные сессии

    session_unset();

```

```
// сессия пуста

echo '<h3>Данные сессии обнулены:</h3>' ;

echo "<pre>";

print_r($_SESSION);

echo "</pre>";

// декодируем данные сессии из строки запроса

// восстановим сессию

session_decode($_GET['s']);

// сессия успешно восстановлена

echo '<h3>Данные сессии успешно восстановлены:</h3>' ;

echo "<pre>";

print_r($_SESSION);

echo "</pre>";

?>
```