Search

Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

g h
: Goto homepage

g s
: Goto search
(current page)

/
: Focus search box

Константы »
« Переменные переменных

- Руководство по PHP
- Справочник языка
- Переменные

Change language: Russian

Переменные извне PHP

HTML-формы (GET и POST)

Когда происходит отправка данных формы PHP-скрипту, информация из этой формы автоматически становится доступной ему. Существует несколько способов получения этой информации, например:

Пример #1 Простая HTML-форма

```
<form action="foo.php" method="post">
    Имя:  <input type="text" name="username" /><br />
    Email: <input type="text" name="email" /><br />
    <input type="submit" name="submit" value="Отправь меня!" />
</form>
```

Есть только два способа получить доступ к данным из форм HTML. Доступные сейчас способы приведены ниже:

Пример #2 Доступ к данным из простой HTML-формы, отправленной через POST

```
<?php
echo $_POST['username'];
echo $_REQUEST['username'];
?>
```

GET-форма используется аналогично, за исключением того, что вместо POST, вам нужно будет использовать соответствующую предопределённую переменную GET. GET относится также к `QUERY_STRING` (информация в URL после '?'). Так, например, `http://www.example.com/test.php?id=3` содержит GET-данные, доступные как *$_GET['id']*. Смотрите также *$_REQUEST*.

> **Замечание**:
>
> Точки и пробелы в именах переменных преобразовывается в знаки подчёркивания. Например, `<input name="a.b" />` станет `$_REQUEST["a_b"]`.

PHP также понимает массивы в контексте переменных формы (смотрите соответствующие ЧАВО). К примеру, вы можете сгруппировать связанные переменные вместе или использовать эту возможность для получения значений списка множественного выбора select. Например, давайте отправим форму самой себе, а после отправки отобразим данные:

Пример #3 Более сложные переменные формы

```
<?php
if ($_POST) {
echo '<pre>';
echo htmlspecialchars(print_r($_POST, true));
echo '</pre>';
}
?>
<form action="" method="post">
Имя: <input type="text" name="personal[name]" /><br />
Email: <input type="text" name="personal[email]" /><br />
Пиво: <br />
<select multiple name="beer[]">
<option value="warthog">Warthog</option>
<option value="guinness">Guinness</option>
<option value="stuttgarter">Stuttgarter Schwabenbräu</option>
</select><br />
<input type="submit" value="Отправь меня!" />
</form>
```

> **Замечание**: Если внешнее имя переменной начинается с корректного синтаксиса массива, завершающие символы молча игнорируются. Например, `<input name="foo[bar]baz">` станет `$_REQUEST['foo']['bar']`.

Имена переменных кнопки-изображения

При отправке формы вместо стандартной кнопки можно использовать изображение с помощью тега такого вида:

```
<input type="image" src="image.gif" name="sub" />
```

Когда пользователь щёлкнет где-нибудь на изображении, соответствующая форма будет передана на сервер с двумя дополнительными переменными — *sub_x* и *sub_y*. Они содержат координаты нажатия пользователя на изображение. Опытные программисты могут заметить, что на самом деле имена переменных, отправленных браузером, содержат точку, а не подчёркивание, но PHP автоматически преобразовывает точку в подчёркивание.

HTTP Cookies

PHP прозрачно поддерживает HTTP cookies как определено в » RFC 6265. Cookies — это механизм для хранения данных в удалённом браузере и, таким образом, отслеживания и идентификации вернувшихся пользователей. Вы можете установить cookies, используя функцию setcookie(). Cookies являются частью HTTP-заголовка, поэтому функция SetCookie должна вызываться до того, как браузеру будет отправлен какой бы то ни было вывод. Это то же ограничение, что и для функции header(). Данные, хранящиеся в cookie, доступны в соответствующих массивах данных cookie, таких как *$_COOKIE* и *$_REQUEST*. Подробности и примеры смотрите в справочной странице setcookie().

> **Замечание**: Начиная с PHP 7.2.34, 7.3.23 и 7.4.11, соответственно, *имена* входящих cookie больше не декодируются из URL-закодированной строки из соображений безопасности.

Если вы хотите присвоить множество значений одной переменной cookie, вы можете присвоить их как массив. Например:

```
<?php
setcookie("MyCookie[foo]", 'Testing 1', time()+3600);
setcookie("MyCookie[bar]", 'Testing 2', time()+3600);
?>
```

Это создаст две разные cookie, хотя в вашем скрипте *MyCookie* будет теперь одним массивом. Если вы хотите установить именно одну cookie со множеством значений, сначала рассмотрите возможность использования к значениям такие функции, как serialize() или explode().

Обратите внимание, что cookie заменит предыдущую cookie с тем же именем в вашем браузере, если только путь или домен не отличаются. Так, для приложения корзины покупок вы, возможно, захотите сохранить счётчик. То есть:

Пример #4 Пример использования setcookie()

```
<?php
if (isset($_COOKIE['count'])) {
$count = $_COOKIE['count'] + 1;
} else {
$count = 1;
}
setcookie('count', $count, time()+3600);
setcookie("Cart[$count]", $item, time()+3600);
?>
```

Точки в именах приходящих переменных

Как правило, PHP не меняет передаваемых скрипту имён переменных. Однако следует отметить, что точка не является корректным символом в имени переменной PHP. Поэтому рассмотрим такую запись:

```
<?php
$varname.ext; /* неверное имя переменной */
?>
```

В данном случае интерпретатор видит переменную *$varname*, после которой идёт оператор конкатенации, а затем голая строка (то есть, не заключённая в кавычки строка, не соответствующая ни одному из ключевых или зарезервированных слов) 'ext'. Очевидно, что это не даст ожидаемого результата.

По этой причине важно отметить, что PHP будет автоматически заменять любые точки в именах, приходящих переменных на символы подчёркивания.

Определение типов переменных

Поскольку PHP определяет типы переменных и преобразовывает их (как правило) по мере необходимости, не всегда

очевидно, какой тип имеет данная переменная в любой момент времени. PHP содержит несколько функций, позволяющих определить тип переменной, таких как: gettype(), is_array(), is_float(), is_int(), is_object() и is_string(). Смотрите также раздел Типы.

HTTP является текстовым протоколом, и большинство, если не всё, содержимое, которое приходит в суперглобальные массивы, например, *$_POST* и *$_GET*, останется в виде строк. PHP не будет преобразовывать значения в определённый тип. В приведённом ниже примере *$_GET["var1"]* будет содержать строку "null", а *$_GET["var2"]* — строку "123".

```
/index.php?var1=null&var2=123
```

Список изменений

Версия	Описание
7.2.34, 7.3.23, 7.4.11	*имена* входящих cookie больше не декодируются из URL-закодированной строки из соображений безопасности.

+ add a note

User Contributed Notes 2 notes

up
down
19
Anonymous ¶
16 years ago

```
The full list of field-name characters that PHP converts to _ (underscore) is the following (not just dot):
chr(32) ( ) (space)
chr(46) (.) (dot)
chr(91) ([) (open square bracket)
chr(128) - chr(159) (various)

PHP irreversibly modifies field names containing these characters in an attempt to maintain compatibility with the
deprecated register_globals feature.
```

up
down
6
krydprz at iit dot edu ¶
18 years ago

```
This post is with regards to handling forms that have more than one submit button.

Suppose we have an HTML form with a submit button specified like this:

<input type="submit" value="Delete" name="action_button">

Normally the 'value' attribute of the HTML 'input' tag (in this case "Delete") that creates the submit button can be
accessed in PHP after post like this:

<?php
$_POST['action_button'];
?>

We of course use the 'name' of the button as an index into the $_POST array.

This works fine, except when we want to pass more information with the click of this particular button.

Imagine a scenario where you're dealing with user management in some administrative interface. You are presented with a
list of user names queried from a database and wish to add a "Delete" and "Modify" button next to each of the names in the
list. Naturally the 'value' of our buttons in the HTML form that we want to display will be "Delete" and "Modify" since
that's what we want to appear on the buttons' faceplates.

Both buttons (Modify and Delete) will be named "action_button" since that's what we want to index the $_POST array with.
In other words, the 'name' of the buttons along cannot carry any uniquely identifying information if we want to process
them systematically after submit. Since these buttons will exist for every user in the list, we need some further way to
```

distinguish them, so that we know for which user one of the buttons has been pressed.

Using arrays is the way to go. Assuming that we know the unique numerical identifier of each user, such as their primary key from the database, and we DON'T wish to protect that number from the public, we can make the 'action_button' into an array and use the user's unique numerical identifier as a key in this array.

Our HTML code to display the buttons will become:

```
<input type="submit" value="Delete" name="action_button[0000000002]">
<input type="submit" value="Modify" name="action_button[0000000002]">
```

The 0000000002 is of course the unique numerical identifier for this particular user.

Then when we handle this form in PHP we need to do the following to extract both the 'value' of the button ("Delete" or "Modify") and the unique numerical identifier of the user we wish to affect (0000000002 in this case). The following will print either "Modify" or "Delete", as well as the unique number of the user:

```
<?php
$submitted_array = array_keys($_POST['action_button']);
echo ($_POST['action_button'][$submitted_array[0]] . " " . $submitted_array[0]);
?>
```

$submitted_array[0] carries the 0000000002.
When we index that into the $_POST['action_button'], like we did above, we will extract the string that was used as 'value' in the HTML code 'input' tag that created this button.

If we wish to protect the unique numerical identifier, we must use some other uniquely identifying attribute of each user. Possibly that attribute should be encrypted when output into the form for greater security.

Enjoy!