The last good thing written in C was Franz Schubert's Symphony Number 9

Unknown

6 GET and POST

6.1 Introduction

How can you pass data between PHP pages? The two most used ways are to pass this data within GET or POST request to the server. By doing this you're not only sending a request for a specific PHP file to be compiled, but passing parameters that can be used in this PHP script.

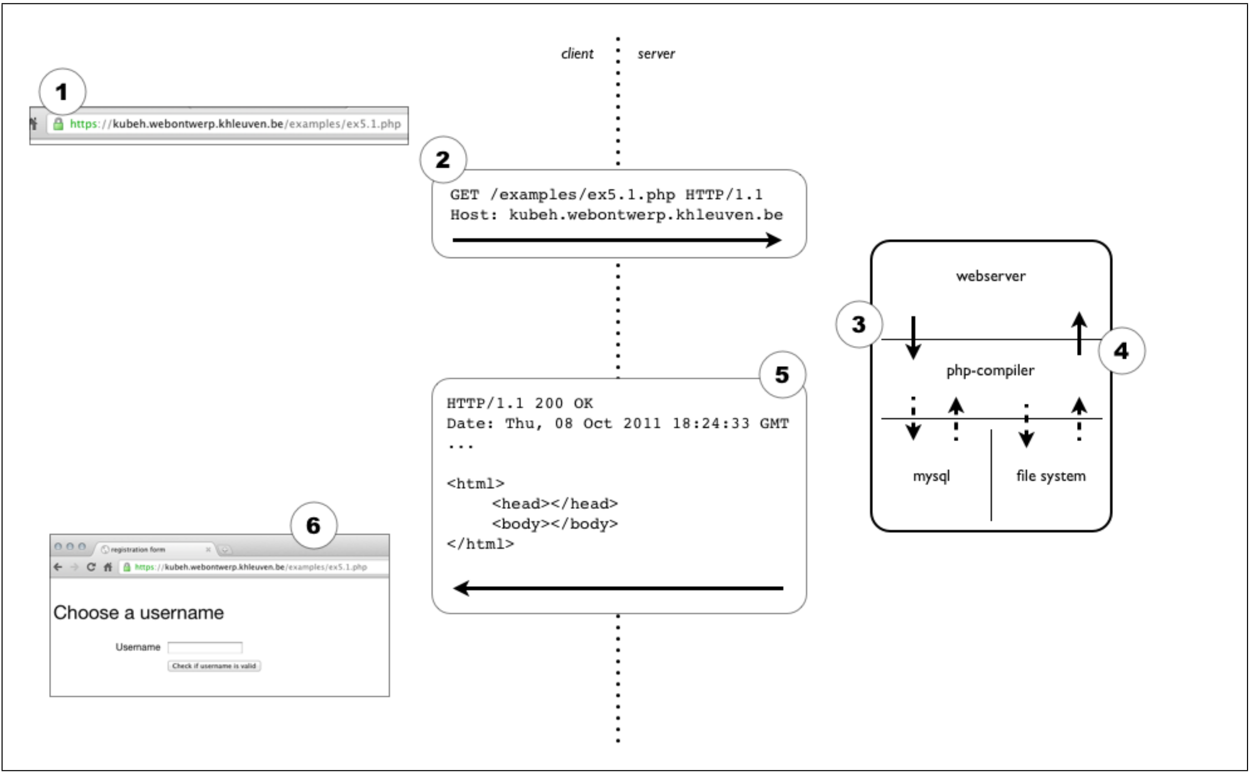
In most cases, a PHP application is run on webserver receiving HTTP requests, processing them, compiling the PHP files and returning the output back to the client.

Although most programmers aren't interested in the network layer and it's protocols, basic knowledge is needed to understand what's happening and to understand e.g. the difference between GET and POST requests.

Within these request, it is possible to add data (e.g. a filled out form or the id of the page that you want to see) so that it can be used by the PHP script.

6.2 Client-Server model

Have a look at this drawing, it explains the steps taken when a php-page is requested from a browser (client):


client
server
1
https://kubeh.webontwerp.khleuven.be/examples/ex5.1.php
2
3
4
5
6
webserver
php-compiler
mysql
file system
registration form
https://kubeh.webontwerp.khleuven.be/examples/ex5.1.php
Choose a username
Username
Check if username is valid

```
GET /examples/ex5.1.php HTTP/1.1
Host: kubeh.webontwerp.khleuven.be
```

```
HTTP/1.1 200 OK
Date: Thu, 08 Oct 2011 18:24:33 GMT
...

<html>
      <head></head>
      <body></body>
</html>
```

Fig. 6.1 Client-Server model simple HTTP request

1. An url is entered in a browser.
2. The browser translates this url in a GET request with at least two parts:
   - a host to contact *dynweb.webontwerp.khleuven.be*
   - a file (or other URI) to make the request for */examples/ex6.1.php*

   This GET request is sent to the server.
3. The server receives the request and finds out it is a PHP file that is requested, so it calls in the PHP compiler to process the script. The compiler then accesses a database or the filesystem to request data and/or other files.
4. The return of this PHP script is sent back to the webserver. The webserver adds headers to the return (PHP can add them as well, but the webserver checks and completes them).
5. The webserver returns the message also two parts:
   - HTTP headers
   - The generated (HTML)-code

   This message is sent to the client that made the HTTP request.
6. The browser reads and checks the headers, reads the code and interpretes the HTML inside to display the markup.

If you want to see the headers passing between client and server, install this plugin in firefox: `https://addons.mozilla.org/en-US/firefox/addon/live-http-headers/`

6.3 Adding GET data to the request

6.3.1 Using a form

When you create an HTML form, you have to define two things:

- a method: GET or POST
- an action: the uri to make the request to. (If you leave this empty, the browser will automatically make the reques against the current URI)

Listing 6.1 Passing GET-variables with a form

```
<form method="get" action="">
<dl>
        <dt>Username</dt>
        <dd><input type="text" name="username" value="" /></dd>
</dl>
<dl>
        <dt></dt>
        <dd><input type="submit" value="Check if username is valid"/></dd>
</dl>
</form>
```

When you make an HTML form like the one above, a GET request will be performed when the user clicks the submit button.

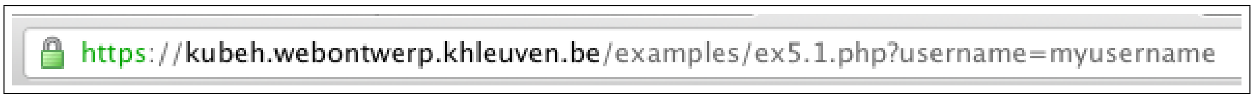
Within this GET-request, a GET-variable with the name `username` will be added. The value of this variable will be the value entered by the user.

Listing 6.2 HTTP headers after submitting the form with GET method

```
GET /examples/ex6.1.php?username=myusername HTTP/1.1
Host: dynweb.webontwerp.khleuven.be
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
DNT: 1
Connection: keep-alive
```

The interesting part is the first line. You'll see the variable with name `username` and value `myusername` (filled out by the user).

If you now have a look at the url in your browser's address bar: you'll see the variables as well.



https://kubeh.webontwerp.khleuven.be/examples/ex5.1.php?username=myusername

Fig. 6.2 URL changes after GET method

6.3.2 Manipulating a url

You can change the value of the variable in this url to something else. By doing this, you are manipulating GET variables too.

GET variables are easily changed and are often used to generate links to pages within a website.

e.g.
`https://portaal.khleuven.be/index.php?page=147`
and
`https://portaal.khleuven.be/index.php?page=156`

Both request the same PHP script, but the variable page is different, thus resulting in a different page.

Listing 6.3 Using an array and a loop to generate different links

```php
<?php
$data = array(
        'cats',
        'dogs',
        'turtles',
        'chicken',
        'guinea pigs',
);

...

<ul>
<?php foreach($data as $item): ?>
        <li><a href="animal.php?animal=<?php echo $item; ?>"><?php echo $item; ?></a></li>
<?php endforeach; ?>
</ul>
?>
```

6.3.3 Using GET variables in your code.

Whenever a PHP script is executed, a lot happens (more later). But an interesting thing is the fact that PHP automatically creates a registervariable (this is a variable that is accessible from everywhere in your PHP code, more on this later, too): `$_GET[]`.

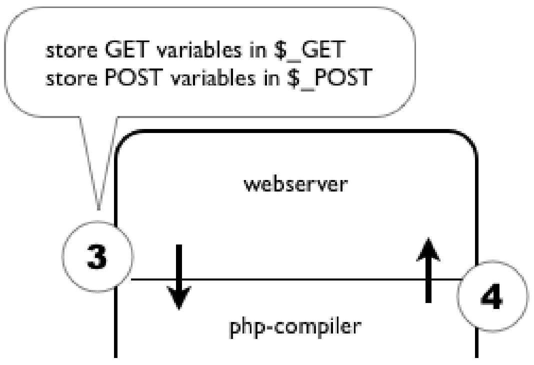


Fig. 6.3 Auto-initialisation of GET and POST variables

`$_GET[]` is an associative array with as key: the name of the variable and as value: the value of that corresponding variable. Like any other associative array, you can access a value or do a debug print of the complete array.

Listing 6.4 Printing the value of a GET variable, if set.

```php
<?php
$data = array(
        'cat',
        'dog',
        'turtle',
        'chicken',
        'guinea pig',
);
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>My animals</title>
    <link rel="stylesheet" media="all" href="stijl.css" />

</head>
<body>
<?php if (isset($_GET['animal'])): ?>
        <div id="flashmessage" class="correct">
        Jolly good, you choose: <?php echo $_GET['animal']; ?>.
        </div>
<?php endif; ?>

<h1>My Animals</h1>

<p>Pick one:</p>
<ul>
<?php foreach($data as $item): ?>
        <li><a href="?animal=<?php echo $item; ?>"><?php echo $item; ?></a></li>
<?php endforeach; ?>
</ul>

</body>
</html>
```

Listing 6.5 Debug print_r() f GET variable.

```php
...

<pre>
<?php print_r($_GET); ?>
</pre> <!-- if you don't know what the <pre> does, google it. -->

...
```

6.4 Adding POST data to the request

6.4.1 Using a form

The other possible method you can choose when sending a form, is POST. The main difference is the method used in the HTTP request.

Listing 6.6 Passing POST-variables with a form

```
<form method="post" action="">
<dl>
        <dt>Username</dt>
        <dd><input type="text" name="username" value="" /></dd>
</dl>
<dl>
        <dt></dt>
        <dd><input type="submit" value="Check if username is valid"/></dd>
</dl>
<div class="clearfix"></div>
</form>
```

Have a look at the header information for the same form, but now with a post method. As you can see, the variables are added at the bottom and some content information (content-type, content-length) is added as well.

Listing 6.7 HTTP headers after submitting the form with POST method

```
POST /examples/ex6.1.php HTTP/1.1
Host: dynweb.webontwerp.khleuven.be
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
DNT: 1
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 19

username=myusername
```

The url of the browser remains the same.

6.4.2 Using POST variables in your code.

In PHP there is no difference in the way to treat `$_GET[]` or `$_POST[]`, as long as you use the correct registervariable-name.

The big difference between POST and GET lies in the browser. A browser can set GET variables via the url or via an HTML form. POST variables can only be set by using an HTML form (or by manipulating your headers).

The best practise is to use GET variables for url modification and POST variables for form handling.

Remember the chapters about functions and the memory scope. `$_GET[]` or `$_POST[]` variables are called *superglobals*. This means that they are automatically available in every function's memory scope. Constants are also superglobal.

6.5 More examples and exercises

Have a look at the example at `https://dynweb.webontwerp.khleuven.be/dynweb/examples/ex6.1.php`. You can see the PHP source for these files by adding an 's' at the end, e.g. `https://dynweb.webontwerp.khleuven.be/dynweb/examples/ex6.1.phps`.

Exercise 6.1 - generate a listing of newsitems and detail newspage

- Download the source file from
  `http://dynweb.webontwerp.khleuven.be/exercises/6.1-source.txt`
- Create a PHP script that achieves the result you can see at
  `http://dynweb.webontwerp.khleuven.be/exercises/6.1.php`.
  Click on a link and see the result.
  Don't forget to also have a look at the html source-code.
- Upload your file so that it is accessible at
  `http://<studentnr>.webontwerp.khleuven.be/exercises/6.1.php`