CS 547

Week 2 Day 2

PHP : Forms

MySQL

Agenda

- Web Forms
- MySQL

Announcements

Review

- From previous lecture
  - Arrays
    - Indexed
    - Associative
    - Multidimensional
  - Bootstrap

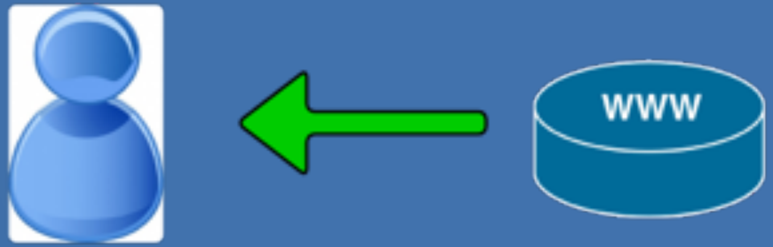
PHP Language

Today:

- Web Forms
- MySQL

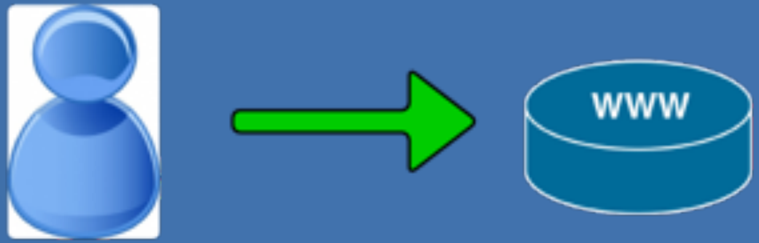
PHP Web Forms

- PHP can be used to display information from the server to the user.



PHP Web Forms

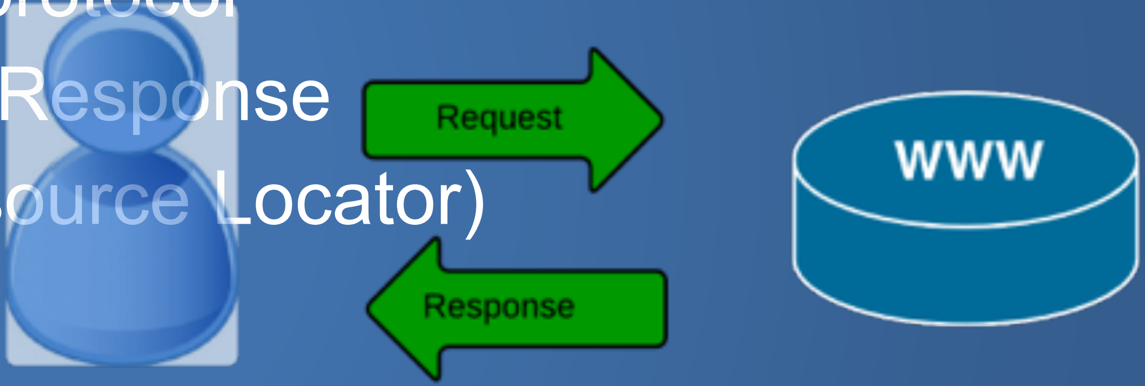
- But what if we wanted to get information *from the user* **back** *to the server*?



HTTP Request Review

The HTTP (Hyper Text Transfer Protocol) is used to exchange information. Recall that

- Stateless protocol
- Request and Response
- URL (Uniform Resource Locator)



HTTP Verbs

- **GET** *fetch* an existing resource. The URL contains all the necessary information the server needs to locate and return the resource.
- **POST** *create* a new resource. POST requests usually carry a payload that specifies the data for the new resource.
- **PUT** *update* an existing resource. The payload may contain the updated data for the resource
- **DELETE** *delete* an existing resource.

HTTP Verbs Less Popular

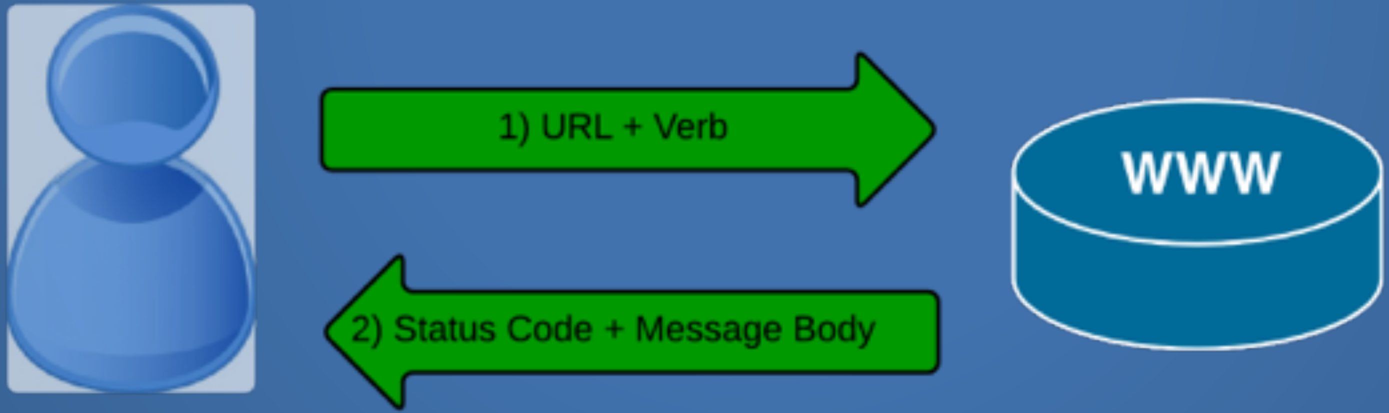
- **HEAD:** this is similar to GET, but without the message body. It's used to retrieve the server headers for a particular resource, generally to check if the resource has changed, via timestamps.

- **TRACE:** used to retrieve the hops that a request takes to round trip from the server. Each intermediate proxy or gateway would inject its IP or DNS name into the Via header field. This can be used for diagnostic purposes.

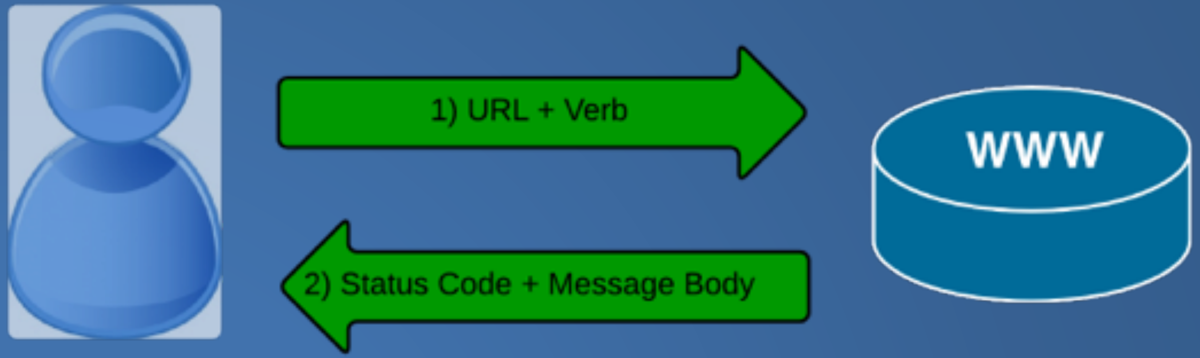
- **OPTIONS:** used to retrieve the server capabilities. On the client-side, it can be used to modify the request based on what the server can support.

Response Status Codes

With URLs and verbs, the client can initiate requests to the server. In return, the server responds with status codes and message payloads. The status code is important and tells the client how to interpret the server response.



HTTP Status Codes



There are lots of status codes. The codes are Numeric and defined in the

protocol definition.

HTTP Common Status Codes

2xx: Successful.

200 – OK

- **202** – Accepted: the request was accepted
- **204** – No Content: there is no message body

HTTP Common Status Codes

3xx: Redirection.

301 – Moved Permanently

303 – See other Response contains temp url

304 – Not modified - cache

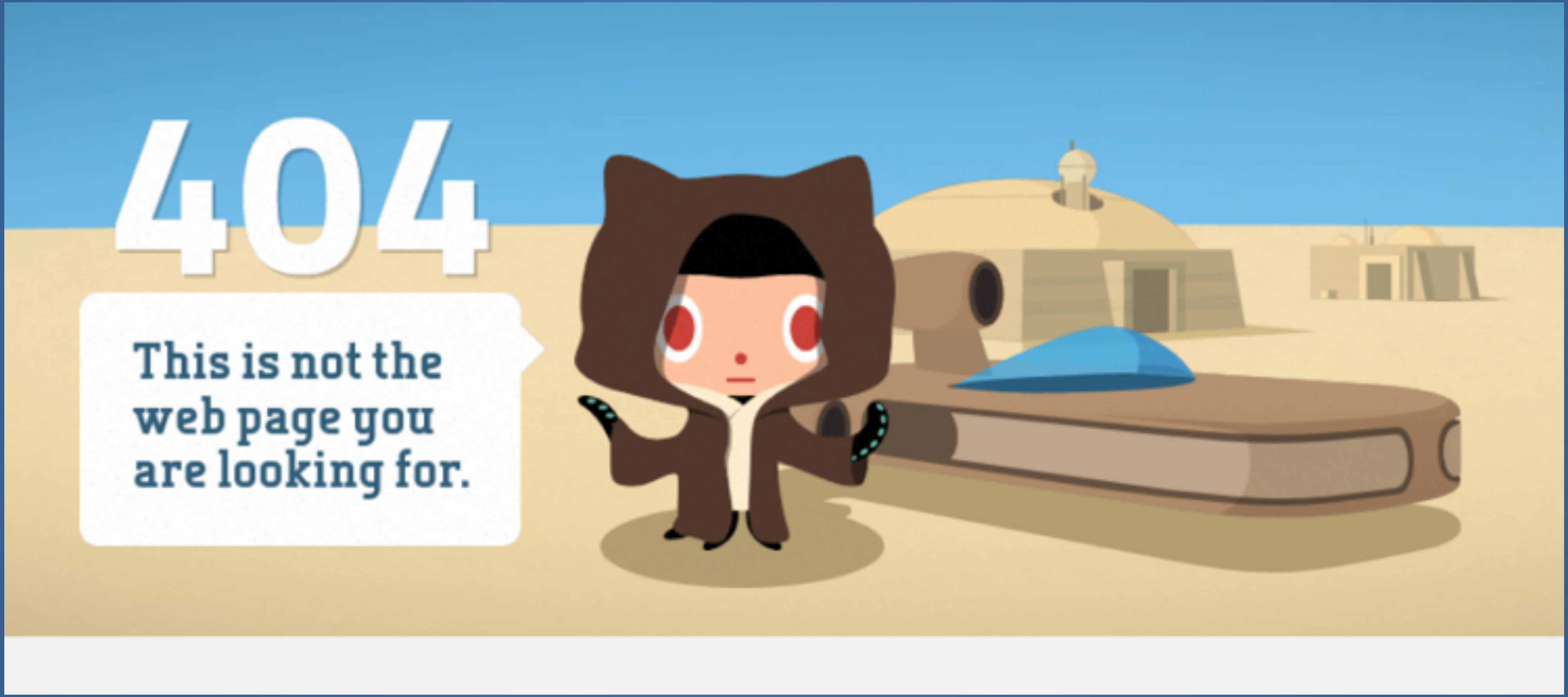
HTTP Common Status Codes

4xx: Client Error.

- 400 – Bad Request: the request was malformed
- 401 – Unauthorized: request requires authenticatior
- 403 – Forbidden
- 404 – Not found.

HTTP Common Status Codes

404: From GitHub.

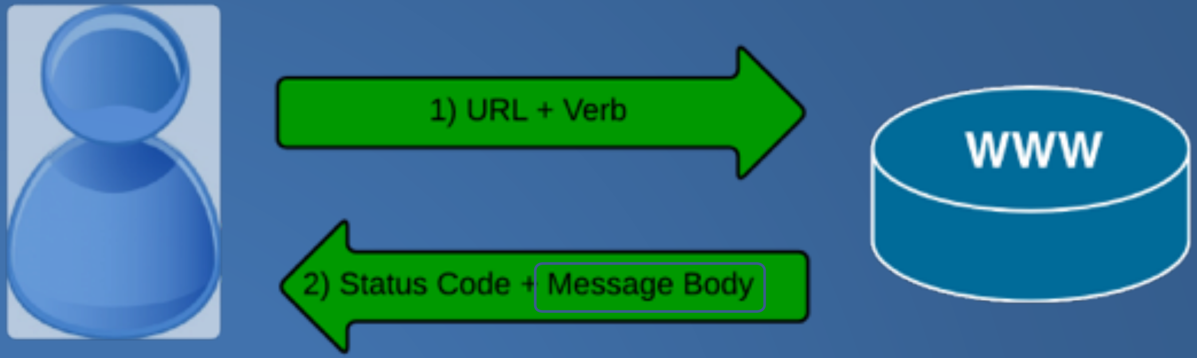
- 

HTTP Common Status Codes

5xx: Server Error.

- 500 – Internal Server Erros. Something is wrong
- 501 – Not Implemented
- 503 – Service Unavailable

HTTP Message Body



- The HTTP specification states that a request or response message has the following generic structure:

```
message = <start-line>
          *(<message-header>)
          CRLF
          [<message-body>]

<start-line> = Request-Line | Status-Line
<message-header> = Field-Name ':' Field-Value
```

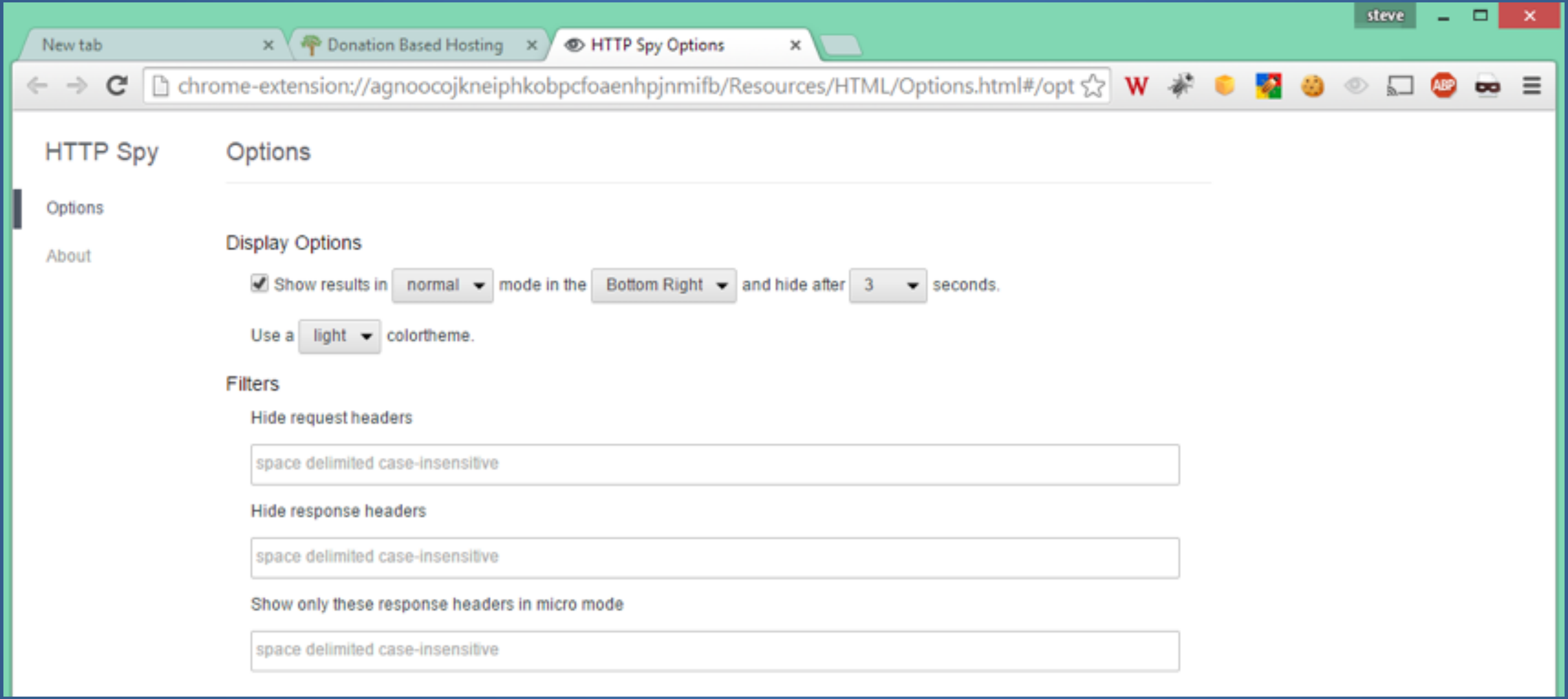
Tools for viewing HTTP Traffic

Popular tools

Chrome DevTools built into the browser

Firefox Developer Tools

Tools: Also usefull – Http Spy Addon


steve
New tab
Donation Based Hosting
HTTP Spy Options
chrome-extension://agnoocojkneiphkobpcfoaenhpjnmifb/Resources/HTML/Options.html#/opt

HTTP Spy

Options

About

Options

Display Options

[✓] Show results in normal mode in the Bottom Right and hide after 3 seconds.

Use a light colortheme.

Filters

Hide request headers

space delimited case-insensitive

Hide response headers

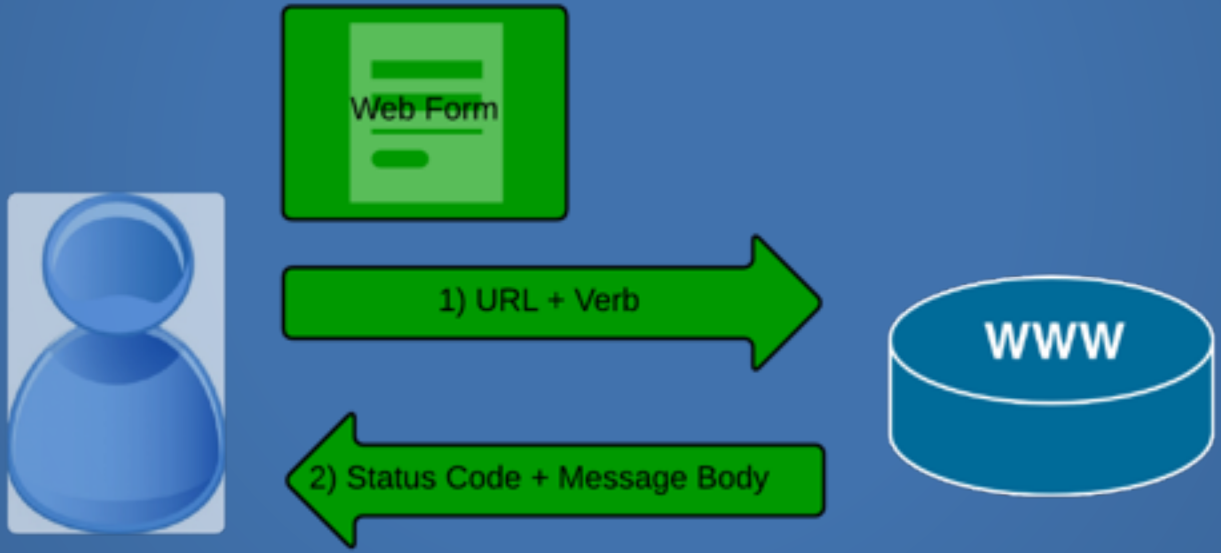
space delimited case-insensitive

Show only these response headers in micro mode

space delimited case-insensitive

Web Forms

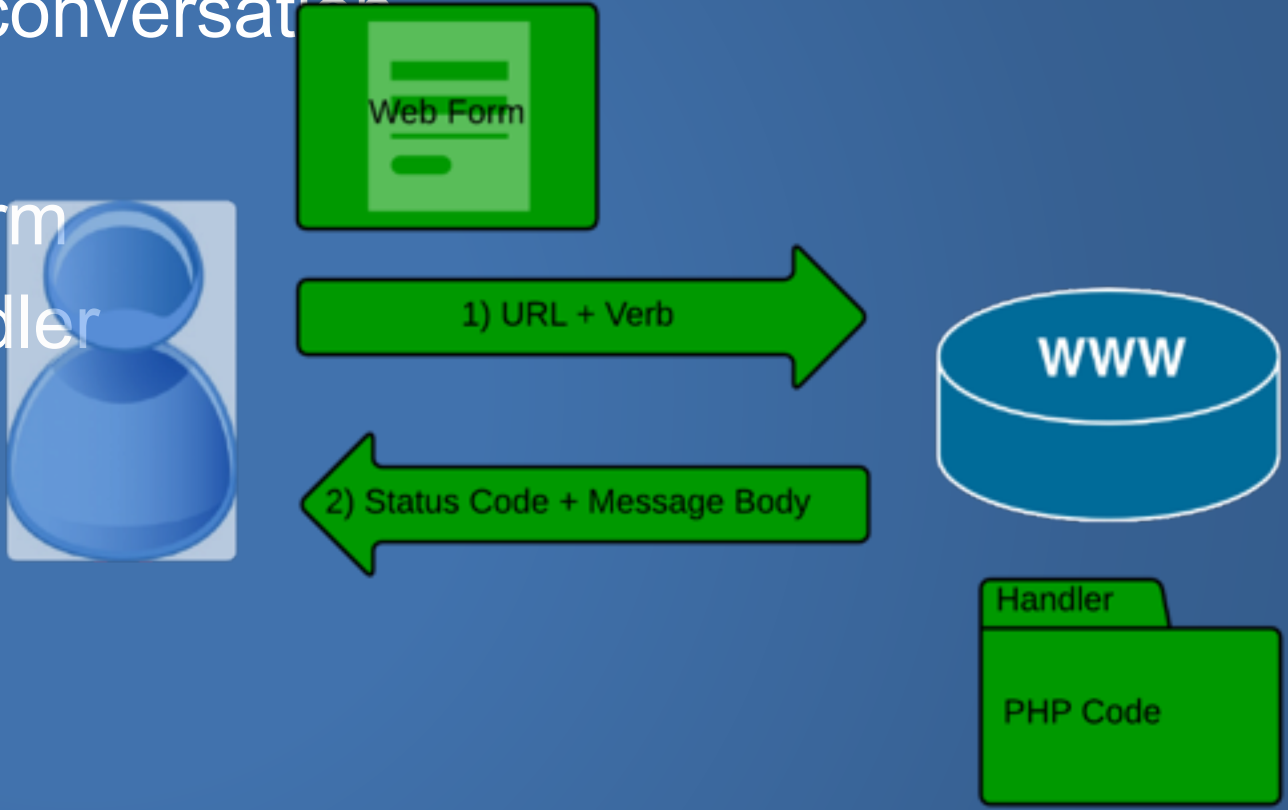
Now we can start talking about Web Forms.



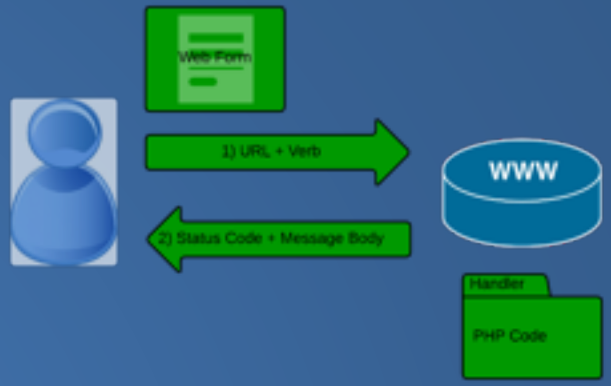
Web Forms

There are two parts to the conversati

1. HTML Form
2. Form Handler



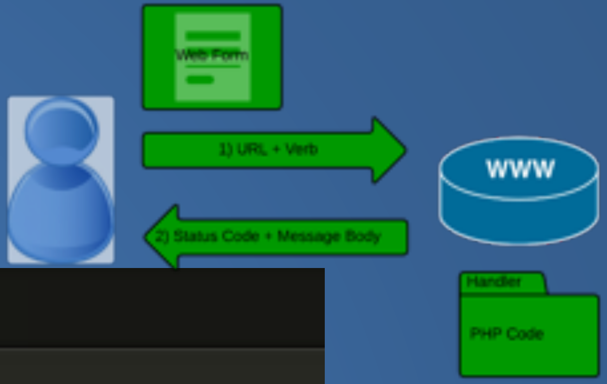
HTML Forms

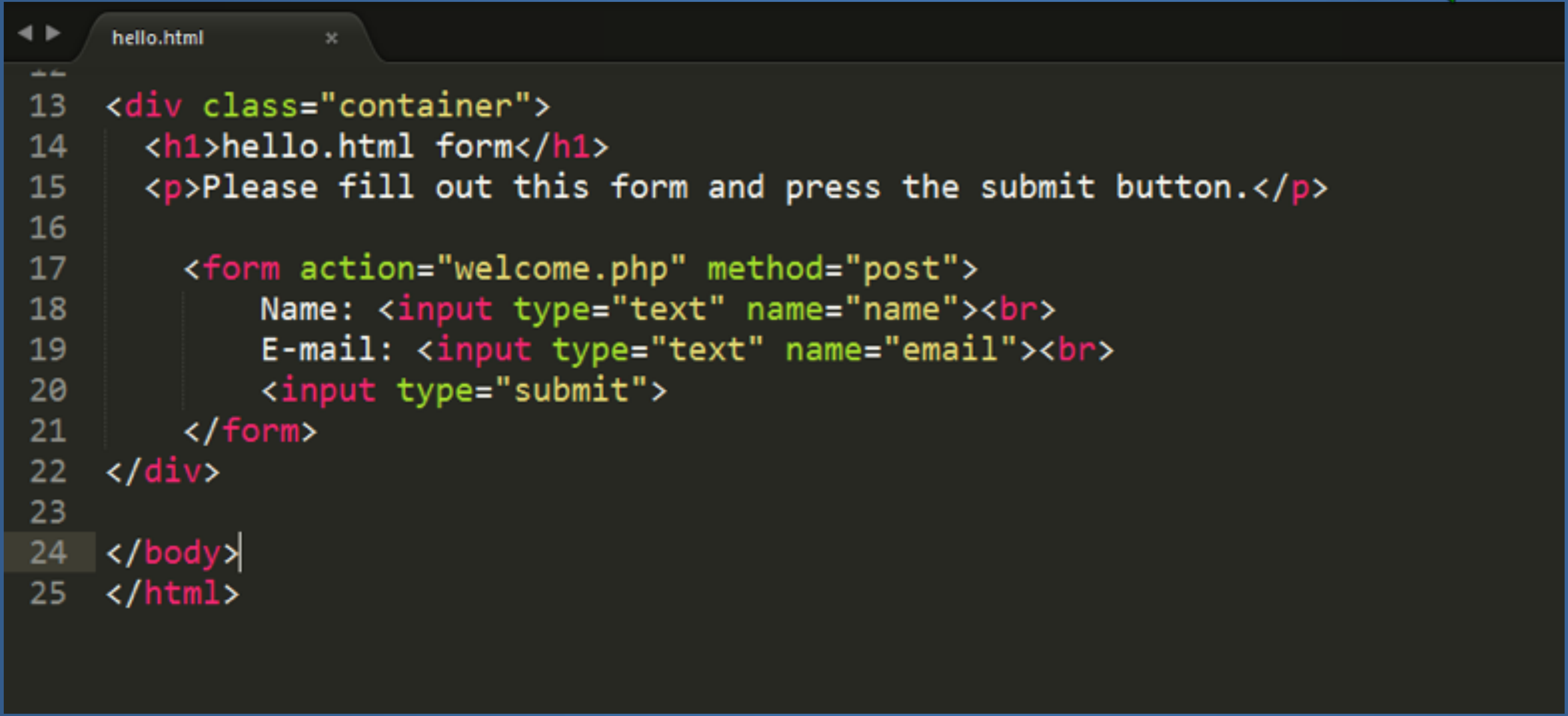


Coded in HTML as:

- <form **action**=“hello.php" method="**post**">
- <input type="submit">
  </form>

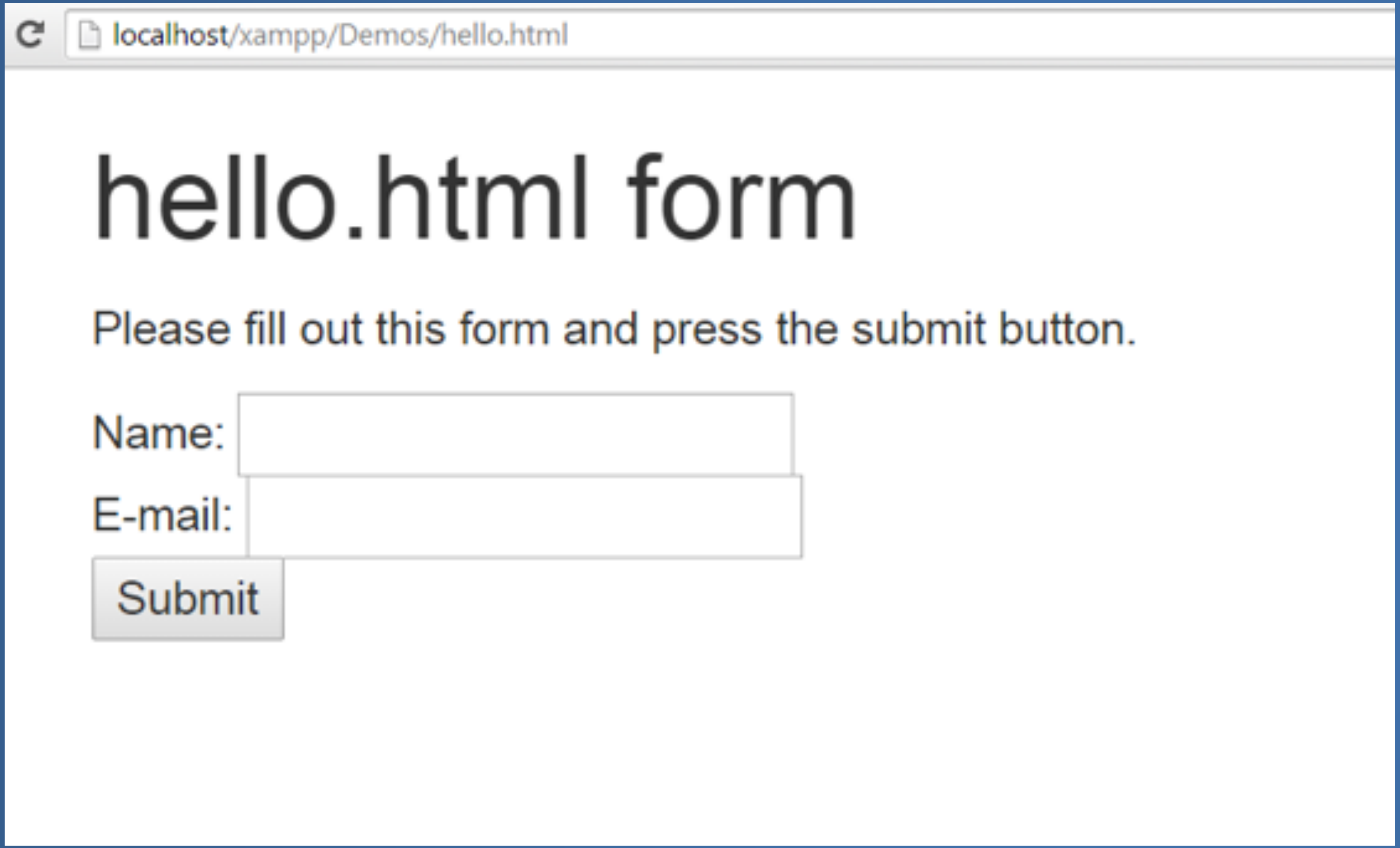
HTML Forms





```html
<div class="container">
  <h1>hello.html form</h1>
  <p>Please fill out this form and press the submit button.</p>

    <form action="welcome.php" method="post">
        Name: <input type="text" name="name"><br>
        E-mail: <input type="text" name="email"><br>
        <input type="submit">
    </form>
</div>

</body>
</html>
```

HTML Forms

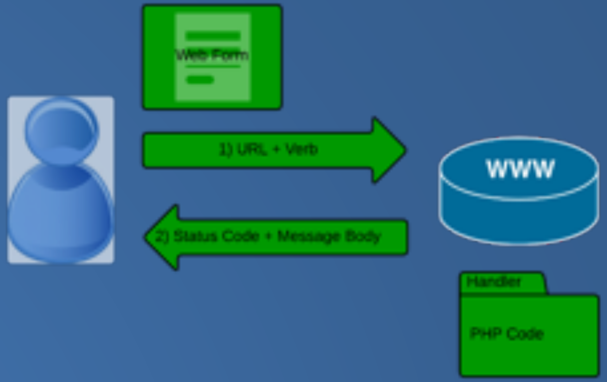

localhost/xampp/Demos/hello.html

hello.html form

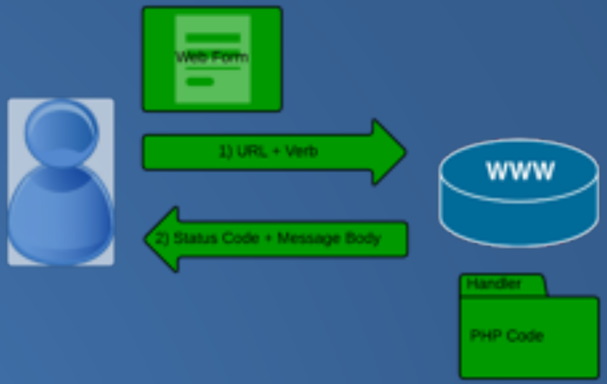
Please fill out this form and press the submit button.

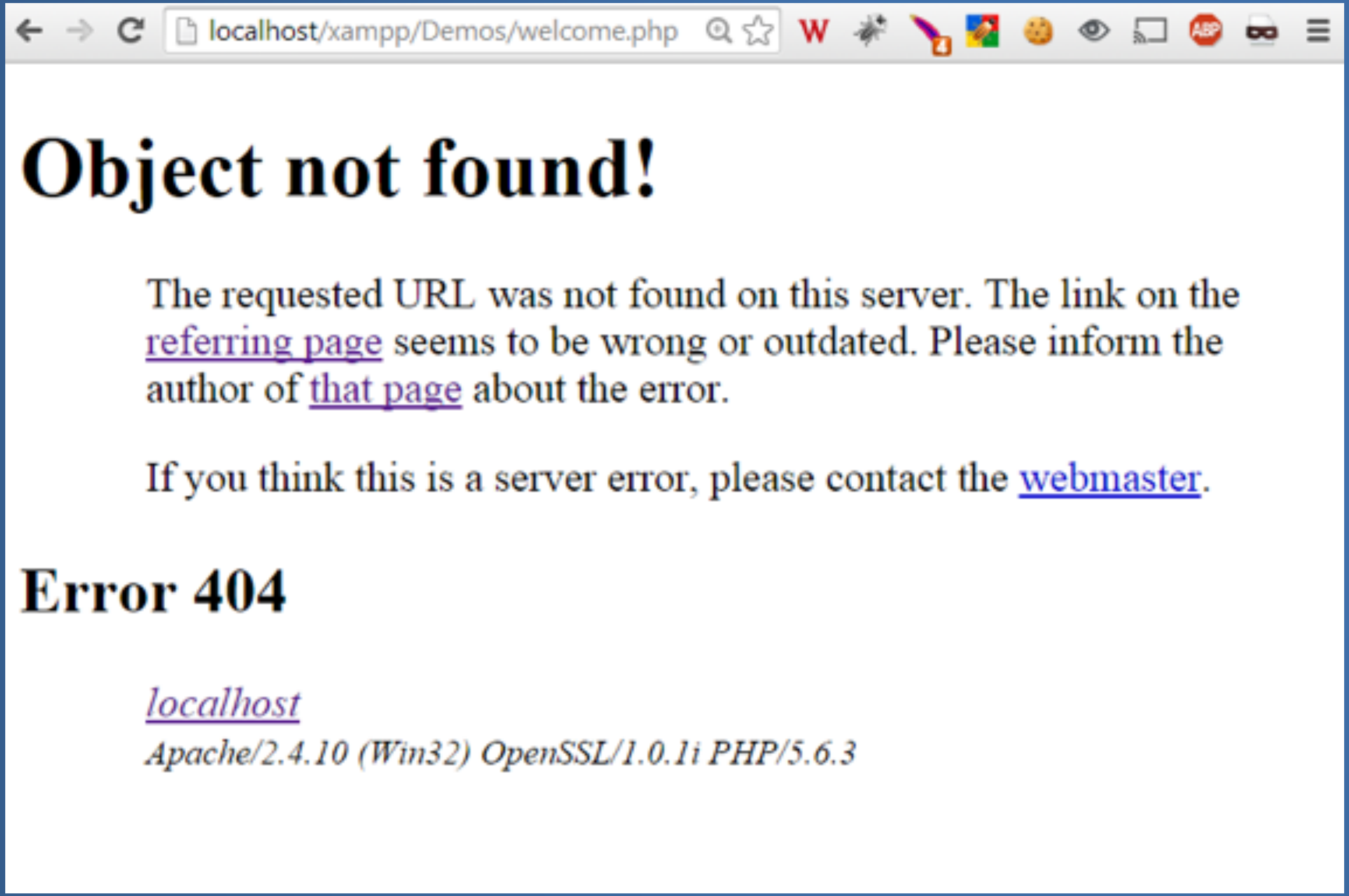
Name:

E-mail:



HTML Forms





Object not found!

The requested URL was not found on this server. The link on the referring page seems to be wrong or outdated. Please inform the author of that page about the error.

If you think this is a server error, please contact the webmaster.

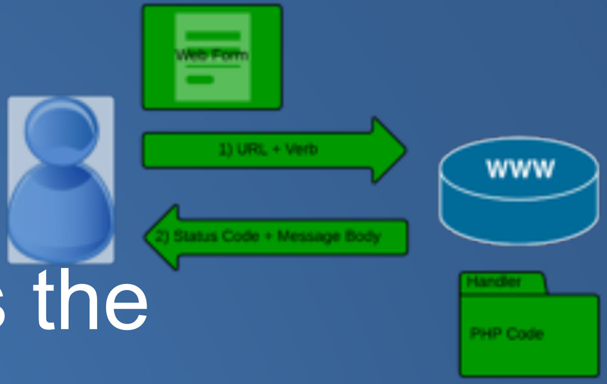
Error 404

localhost

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.6.3

This is ***not*** good.

How do we ***fix this***?



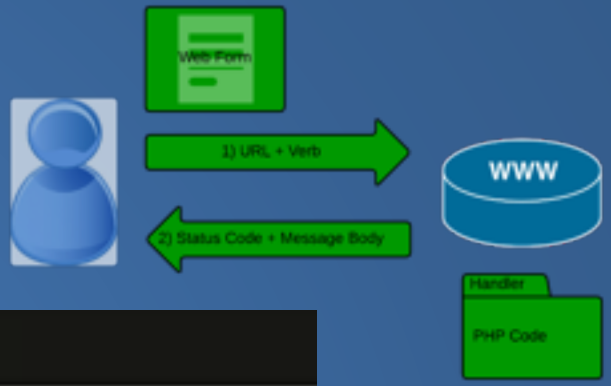
Form Handler

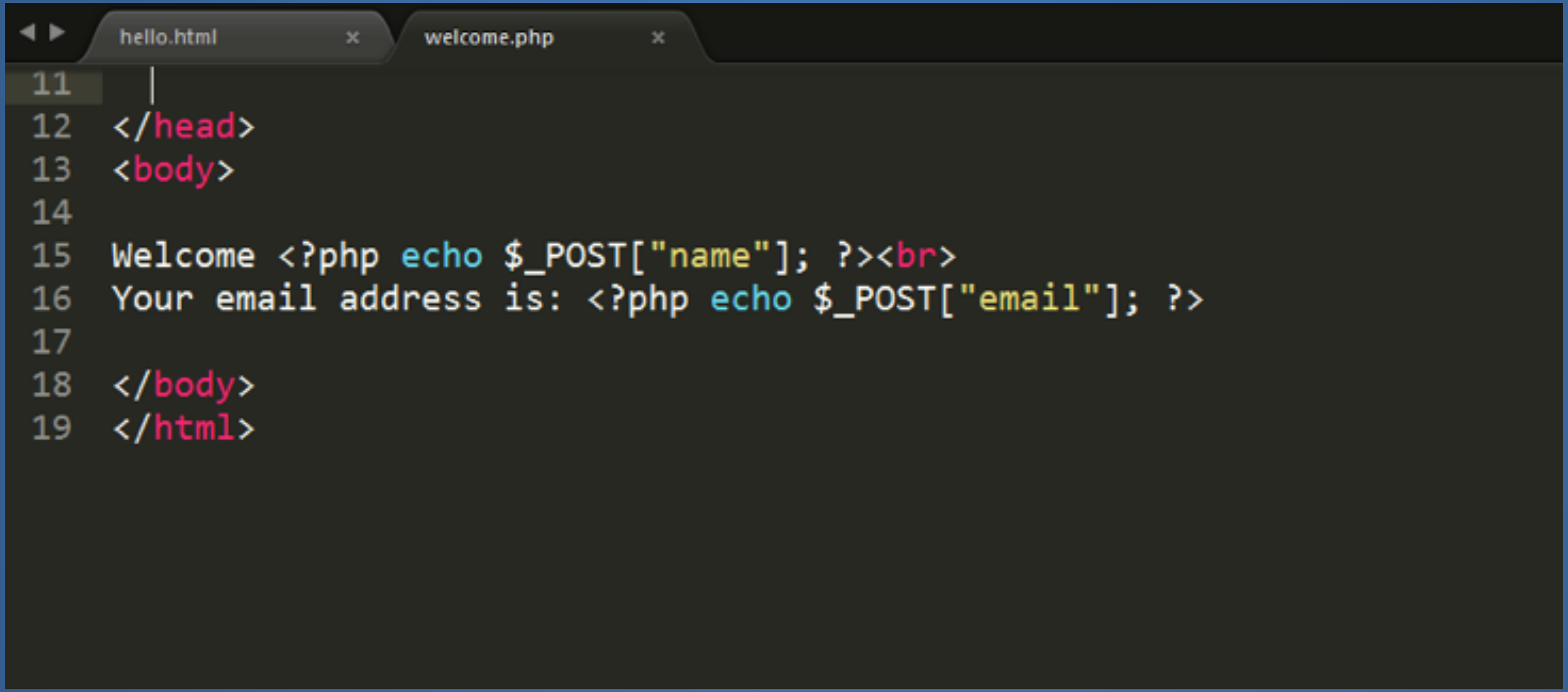
Need a form handler to process the form request no the server.

•

```
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
```

Form Handler

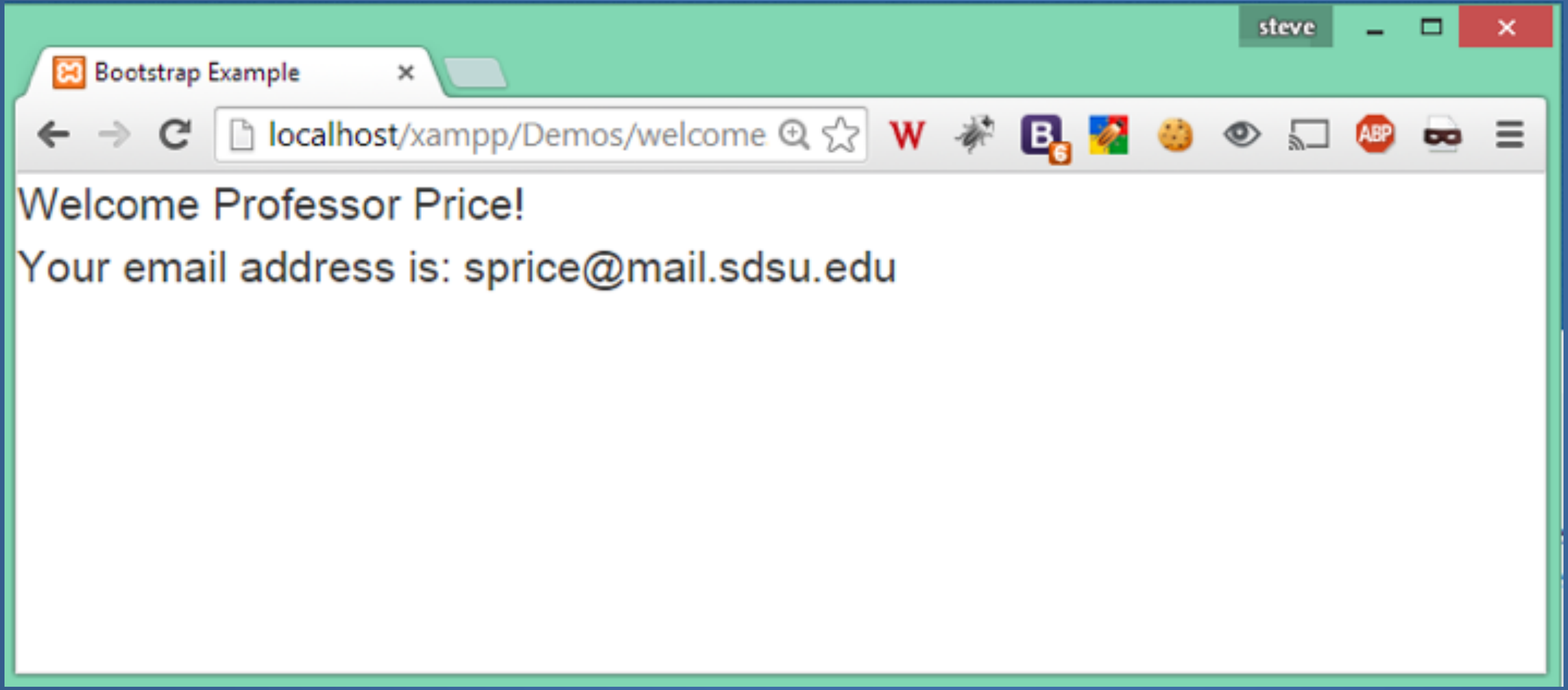




```

</head>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

Form Handler output

Web Form
1) URL + Verb
2) Status Code + Message Body
WWW
Handler
PHP Code

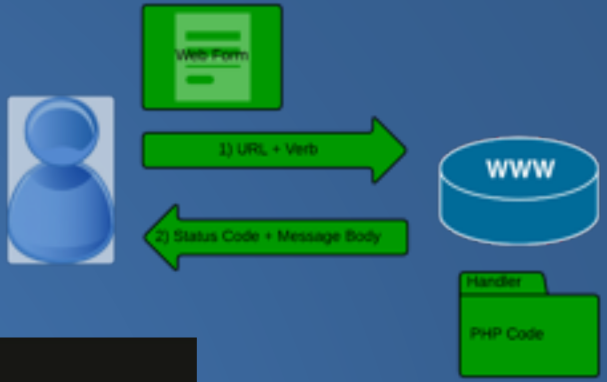


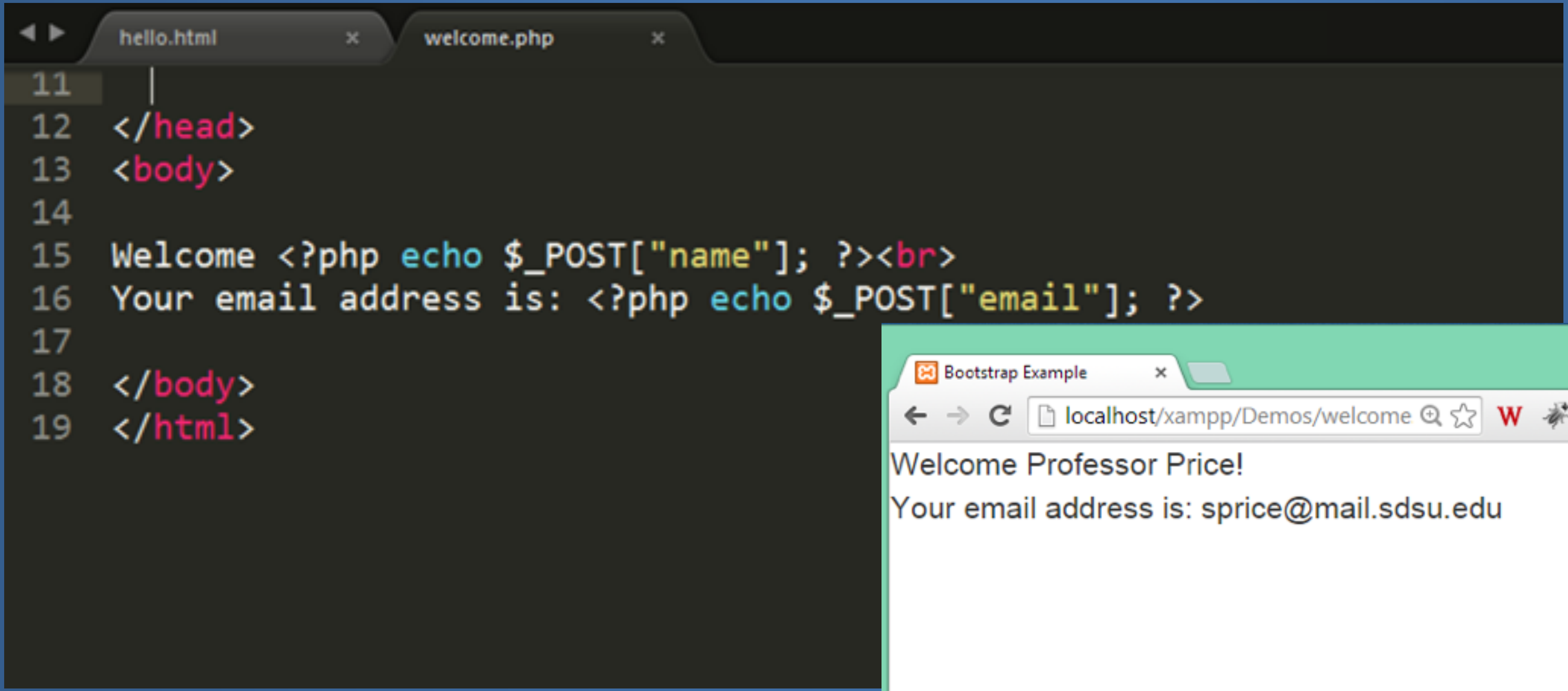
Welcome Professor Price!

Your email address is: sprice@mail.sdsu.edu

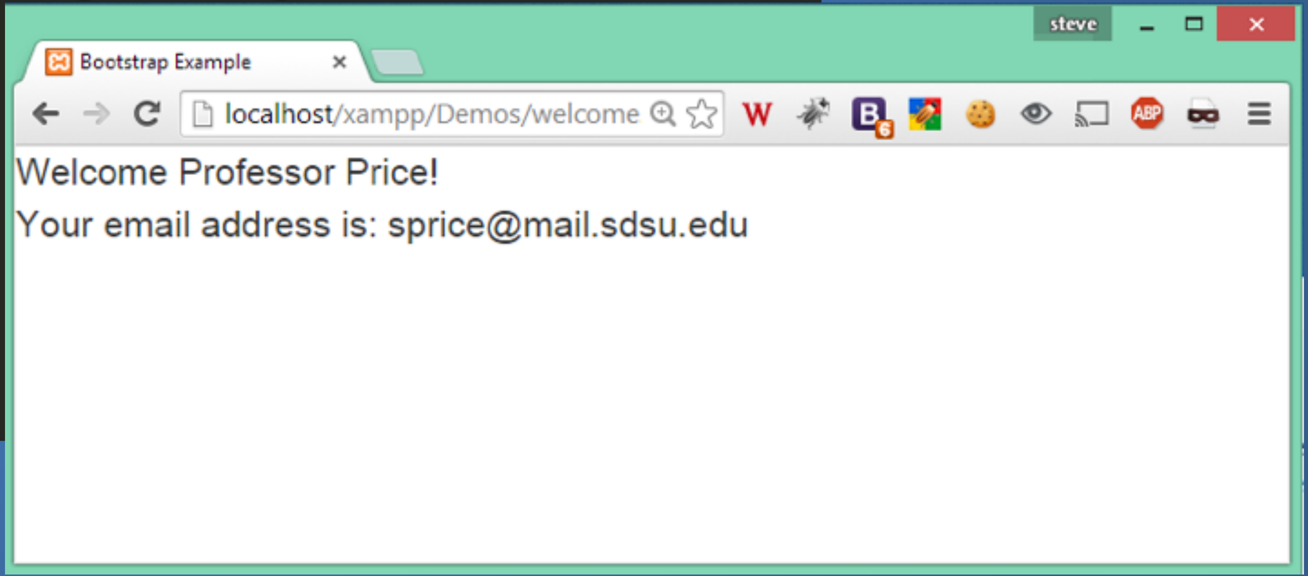
Form Handler output

Great! But how does it work?





```
</head>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```



```
Welcome Professor Price!
Your email address is: sprice@mail.sdsu.edu
```

Notice…

the

$_POST["name"];

And

$_POST["email"];

These are special ***Superglobal*** variables built in to PHP.

PHP SuperGlobal Variables

- The PHP superglobal variables are:
  - $GLOBALS
  - $_SERVER
  - $_REQUEST
  - $_POST
  - $_GET
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION

PHP $GLOBALS

- PHP stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.

  -
    ```
    <?php
    $x = 75;
    $y = 25;

    function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
    }

    addition();
    echo $z;
    ?>
    ```

PHP $_SERVER

- Holds information about headers, paths, and script locations.
- There are 24 elements defined.

PHP $_SERVER

Element/Code	Description
$_SERVER['PHP_SELF']	*Returns the filename of the currently executing script*
$_SERVER['GATEWAY_INTERFACE']	*Returns the version of the Common Gateway Interface (CGI) the server is using*
$_SERVER['SERVER_ADDR']	*Returns the IP address of the host server*
$_SERVER['SERVER_NAME']	*Returns the name of the host server (such as www.w3schools.com)*
$_SERVER['SERVER_SOFTWARE']	*Returns the server identification string (such as Apache/2.2.24)*
$_SERVER['SERVER_PROTOCOL']	*Returns the name and revision of the information protocol (such as HTTP/1.1)*
$_SERVER['REQUEST_METHOD']	*Returns the request method used to access the page (such as POST)*

PHP $_SERVER

Element/Code	Description
$_SERVER['REQUEST_TIME']	*Returns the timestamp of the start of the request (such as 1377687496)*
$_SERVER['QUERY_STRING']	*Returns the query string if the page is accessed via a query string*
$_SERVER['HTTP_ACCEPT']	*Returns the Accept header from the current request*
$_SERVER['HTTP_ACCEPT_CHARSET']	*Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)*
$_SERVER['HTTP_HOST']	*Returns the Host header from the current request*
$_SERVER['HTTP_REFERER']	*Returns the complete URL of the current page (not reliable because not all user-agents support it)*
$_SERVER['HTTPS']	*Is the script queried through a secure HTTP protocol*
$_SERVER['REMOTE_ADDR']	*Returns the IP address from where the user is viewing the current page*

PHP $_SERVER

Element/Code	Description
$_SERVER['REMOTE_HOST']	*Returns the Host name from where the user is viewing the current page*
$_SERVER['REMOTE_PORT']	*Returns the port being used on the user's machine to communicate with the web server*
$_SERVER['SCRIPT_FILENAME']	*Returns the absolute pathname of the currently executing script*
$_SERVER['SERVER_ADMIN']	*Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the*
$_SERVER['SERVER_PORT']	*Returns the port on the server machine being used by the web server for communication (such as 80)*

PHP $_SERVER

Element/Code	Description
$_SERVER['SERVER_SIGNATURE']	*Returns the server version and virtual host name which are added to server-generated pages*
$_SERVER['PATH_TRANSLATED']	*Returns the file system based path to the current script*
$_SERVER['SCRIPT_NAME']	*Returns the path of the current script*
$_SERVER['SCRIPT_URI']	*Returns the URI of the current page*

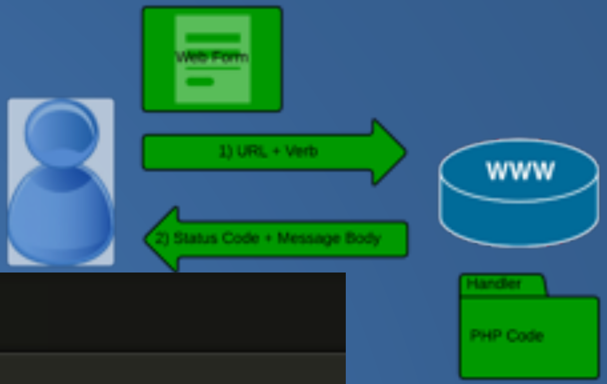
PHP $_REQUEST

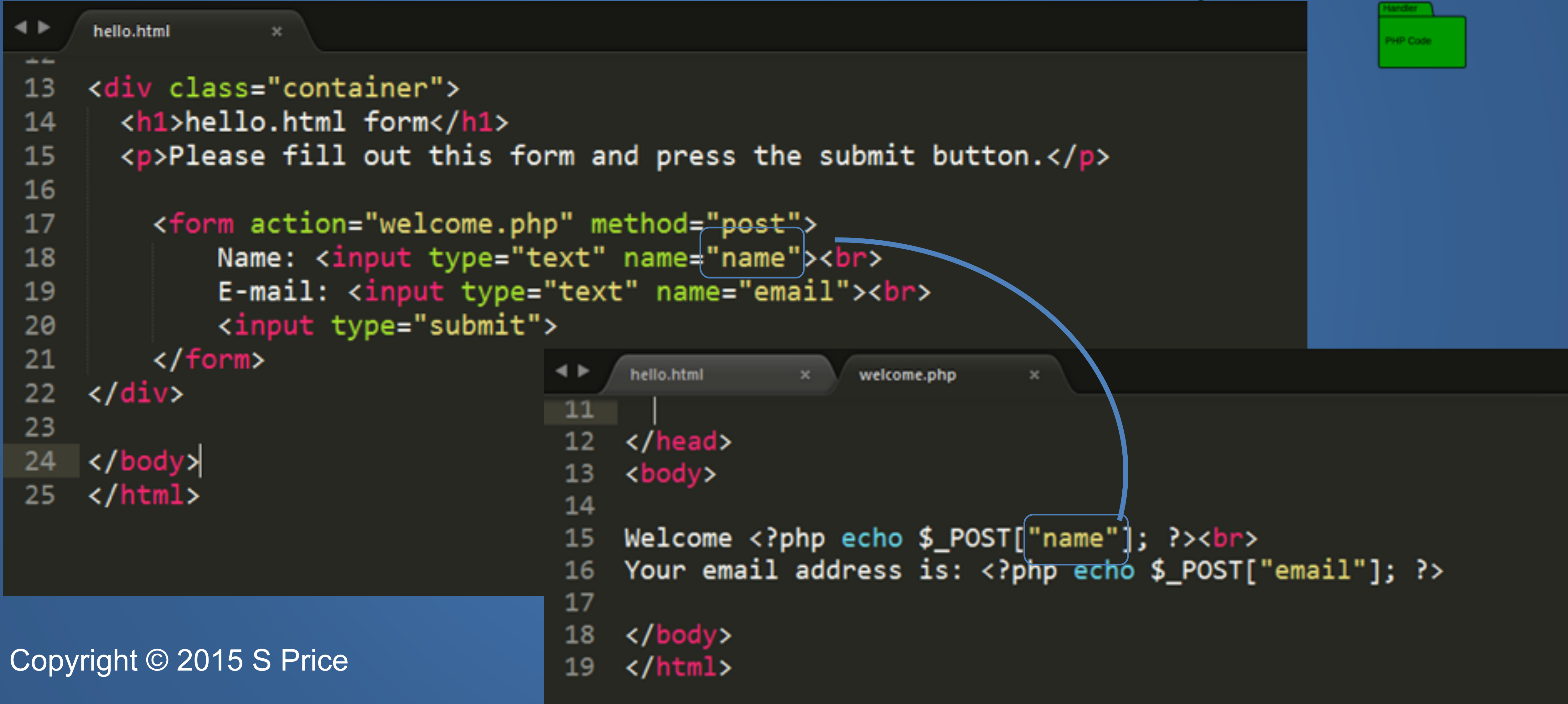
- $_REQUEST is used to collect data after submitting an HTML form.

PHP $_POST

- $_POST is widely used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.
- The values in your PHP Handler must match the values in your HTML Form.

PHP $_POST





```html
<div class="container">
  <h1>hello.html form</h1>
  <p>Please fill out this form and press the submit button.</p>

    <form action="welcome.php" method="post">
        Name: <input type="text" name="name"><br>
        E-mail: <input type="text" name="email"><br>
        <input type="submit">
    </form>
</div>

</body>
</html>
```

```php

</head>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

PHP $_GET

- $_GET can also be used to collect form data after submitting an HTML form with method="get".

Note about POST and GET

- The GET and POST verbs are often used in a similar fashion, but they have greatly different behaviours.

- GET – passes variables on the URL
- POST – passes variables in a new connection.

PHP $_FILES

Used to upload files.

PHP $_ENV

- Used to return the environment variables form the web server.

PHP $_COOKIE

- Cookies are small text files loaded from a server to a client computer storing some information regarding the client computer, so that when the same page from the server is visited by the user, necessary informations can be collected from the cookie itself, decreasing the latency to open the page.

PHP $_SESSION

Sessions are a way to store state information on the server.

To start a session call the function

```
session_start();
```

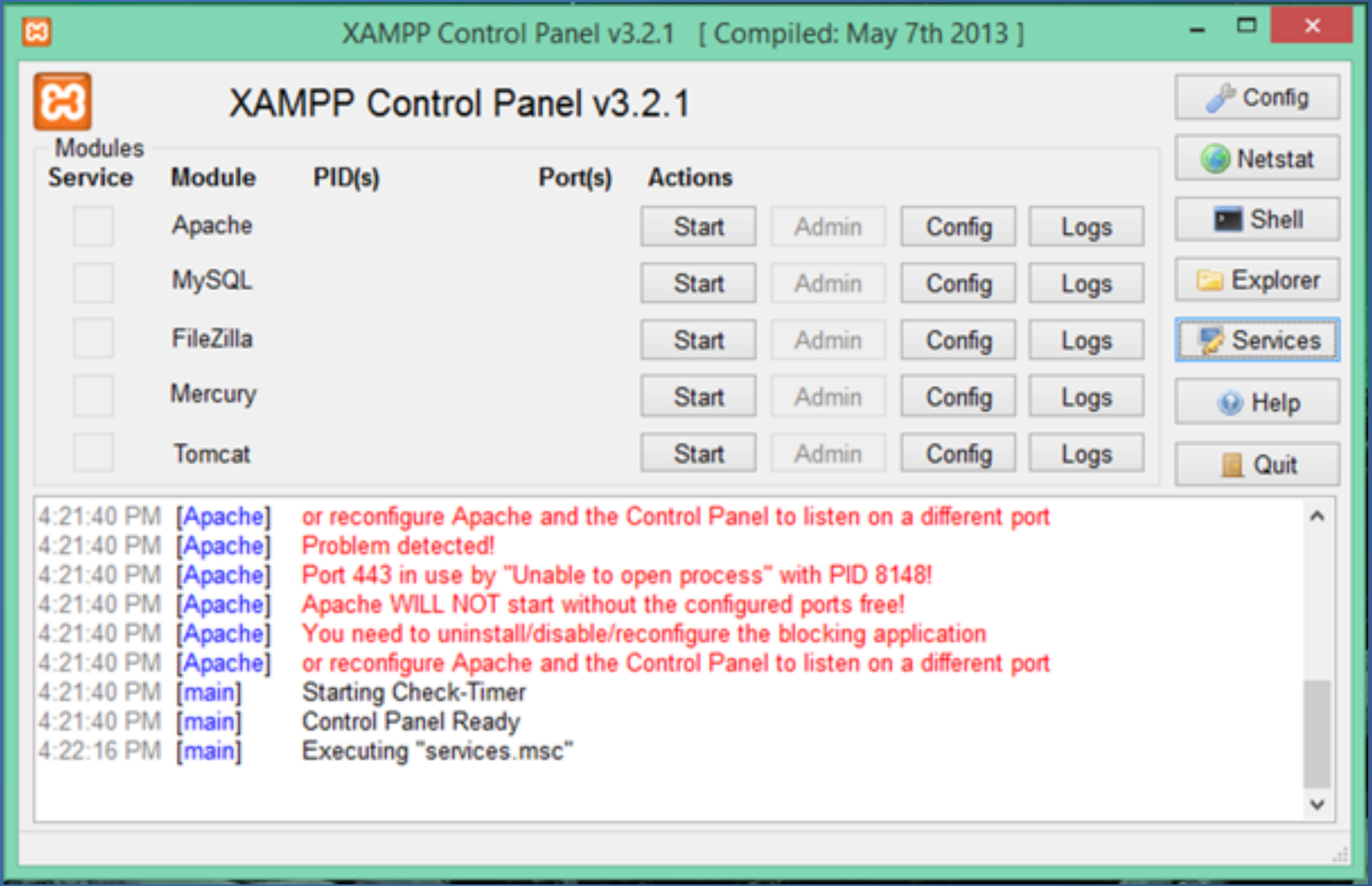
In your handler. Then you can add useful information to the $_SESSION array.

Next Topic

Setting up MySQL in your development environment.

XAMPP

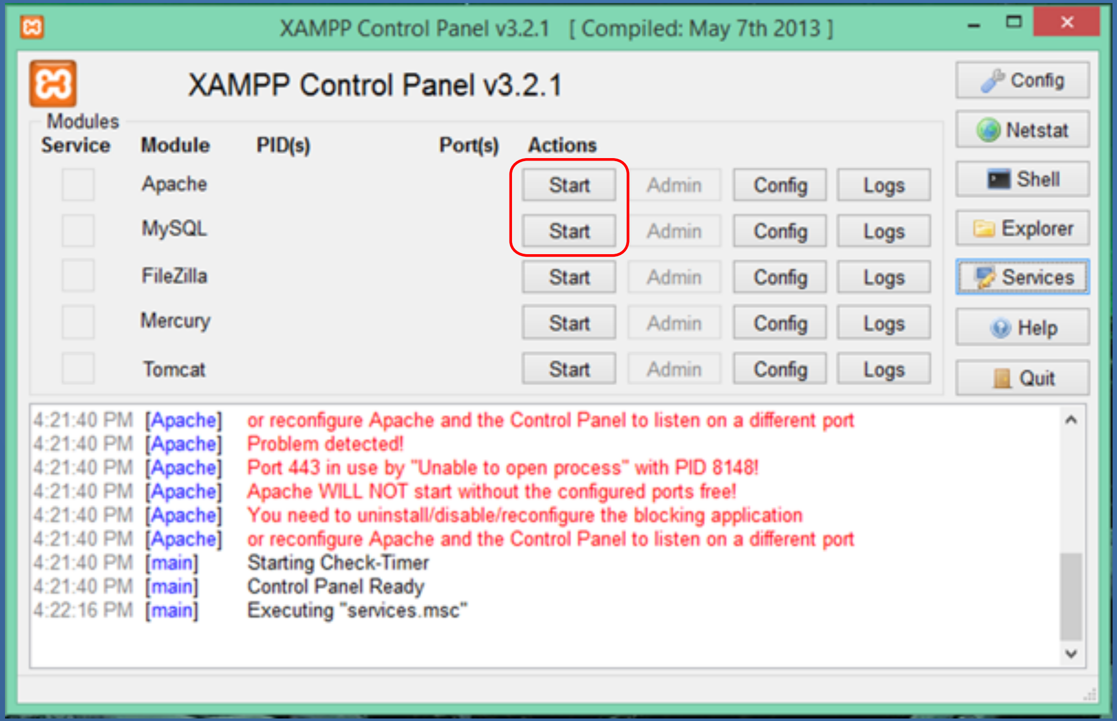
Open the XAMPP Control Panel



```
4:21:40 PM  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
4:21:40 PM  [Apache]   Problem detected!
4:21:40 PM  [Apache]   Port 443 in use by "Unable to open process" with PID 8148!
4:21:40 PM  [Apache]   Apache WILL NOT start without the configured ports free!
4:21:40 PM  [Apache]   You need to uninstall/disable/reconfigure the blocking application
4:21:40 PM  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
4:21:40 PM  [main]     Starting Check-Timer
4:21:40 PM  [main]     Control Panel Ready
4:22:16 PM  [main]     Executing "services.msc"
```

Start using the tools

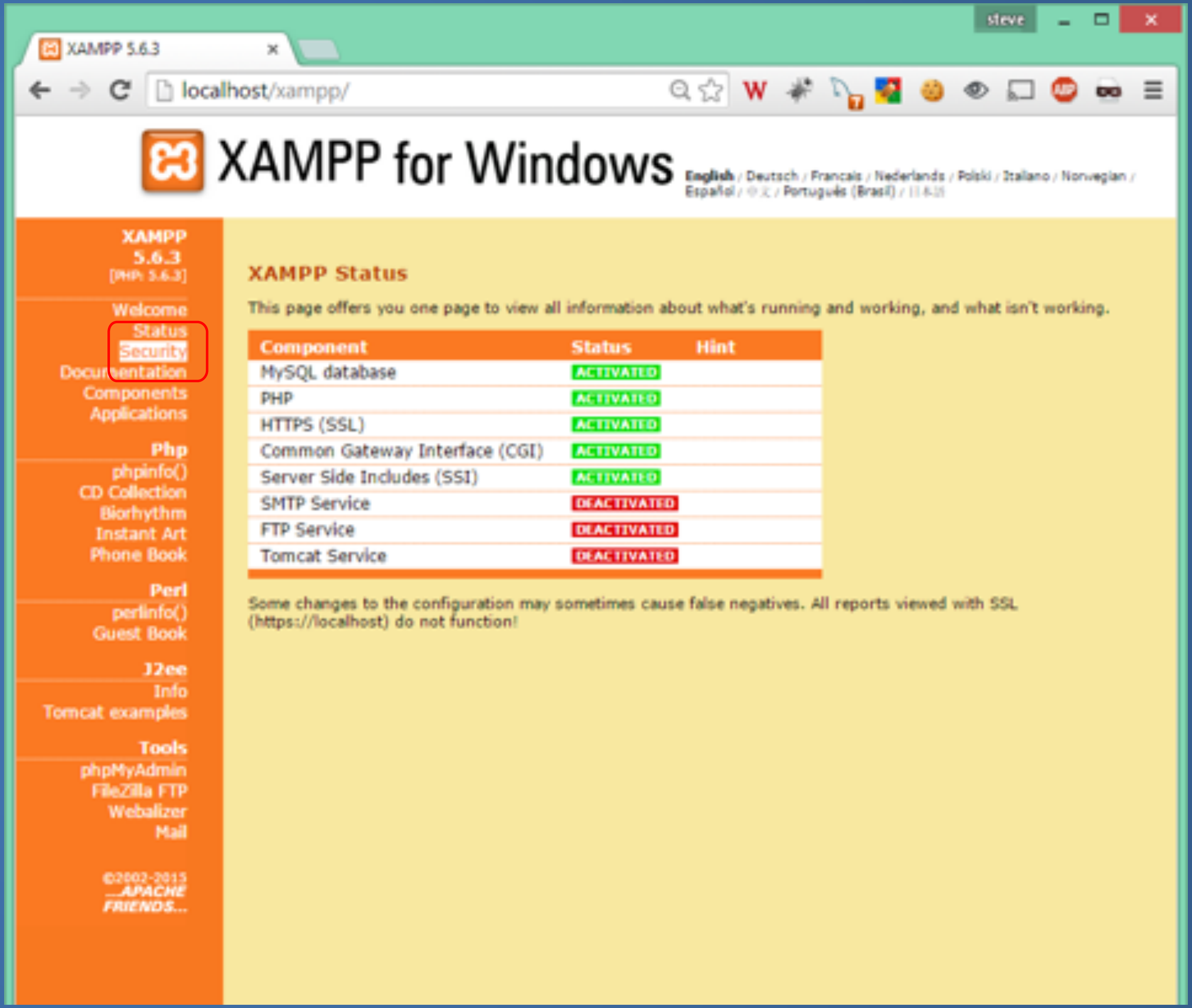
1. Start the xamp control panel
2. Start the apache web service



```
4:21:40 PM  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
4:21:40 PM  [Apache]   Problem detected!
4:21:40 PM  [Apache]   Port 443 in use by "Unable to open process" with PID 8148!
4:21:40 PM  [Apache]   Apache WILL NOT start without the configured ports free!
4:21:40 PM  [Apache]   You need to uninstall/disable/reconfigure the blocking application
4:21:40 PM  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
4:21:40 PM  [main]     Starting Check-Timer
4:21:40 PM  [main]     Control Panel Ready
4:22:16 PM  [main]     Executing "services.msc"
```

XAMPP

1. Navigate to the main page
2. Click the Security link



XAMPP 5.6.3

localhost/xampp/

XAMPP for Windows

English / Deutsch / Francais / Nederlands / Polski / Italiano / Norwegian / Español / [illegible] / Português (Brasil) / [illegible]

XAMPP
5.6.3
[PHP: 5.6.3]

Welcome
Status
Security
Documentation
Components
Applications

Php
phpinfo()
CD Collection
Biorhythm
Instant Art
Phone Book

Perl
perlinfo()
Guest Book

J2ee
Info
Tomcat examples

Tools
phpMyAdmin
FileZilla FTP
Webalizer
Mail

©2002-2015
...APACHE FRIENDS...

XAMPP Status

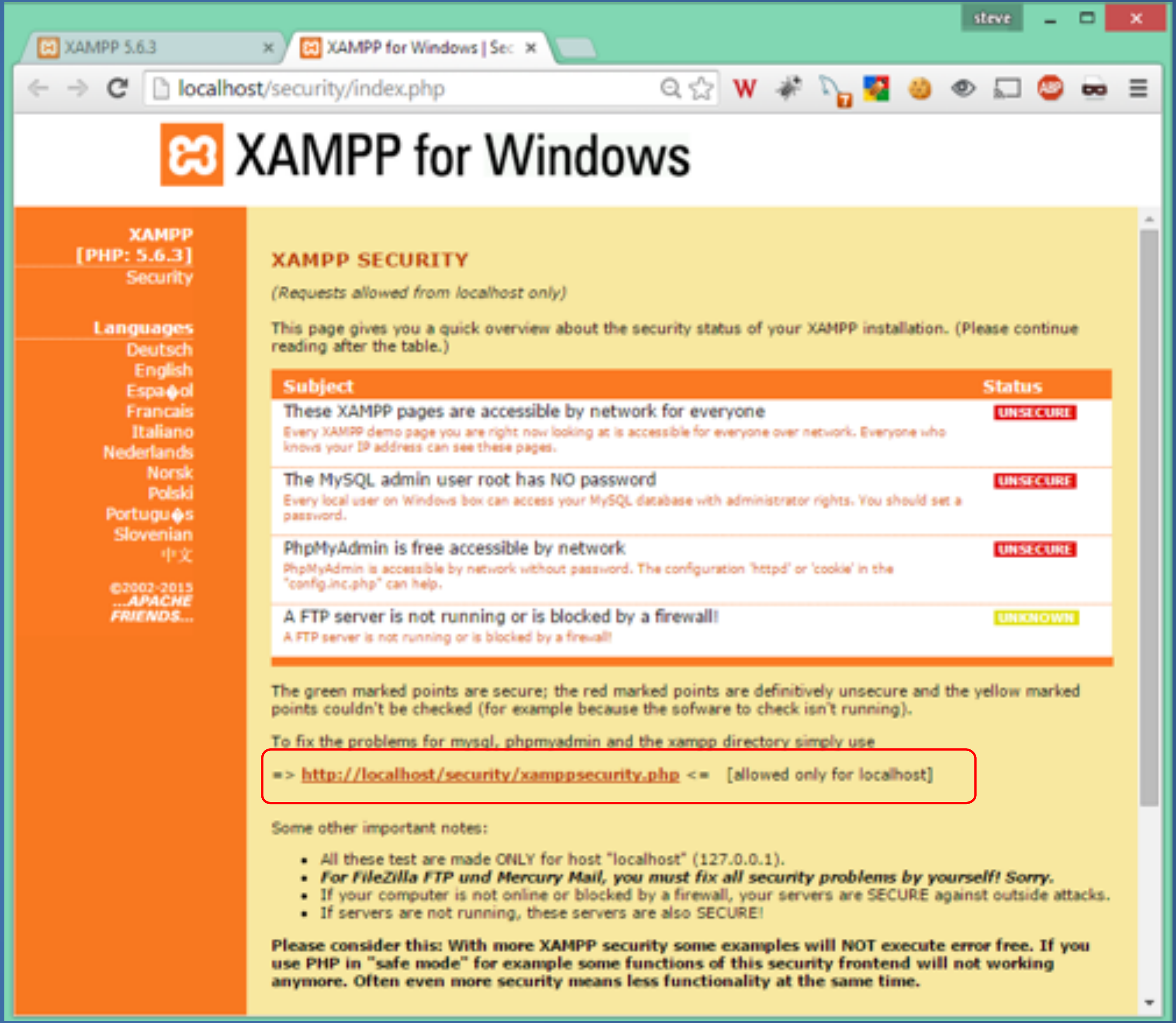
This page offers you one page to view all information about what's running and working, and what isn't working.

Component	Status	Hint
MySQL database	ACTIVATED	
PHP	ACTIVATED	
HTTPS (SSL)	ACTIVATED	
Common Gateway Interface (CGI)	ACTIVATED	
Server Side Includes (SSI)	ACTIVATED	
SMTP Service	DEACTIVATED	
FTP Service	DEACTIVATED	
Tomcat Service	DEACTIVATED	

Some changes to the configuration may sometimes cause false negatives. All reports viewed with SSL (https://localhost) do not function!

XAMPP

1. This will open a new tab
2. Click the xamppsecurity.php link to secure your system



XAMPP SECURITY

(Requests allowed from localhost only)

This page gives you a quick overview about the security status of your XAMPP installation. (Please continue reading after the table.)

Subject	Status
These XAMPP pages are accessible by network for everyone Every XAMPP demo page you are right now looking at is accessible for everyone over network. Everyone who knows your IP address can see these pages.	UNSECURE
The MySQL admin user root has NO password Every local user on Windows box can access your MySQL database with administrator rights. You should set a password.	UNSECURE
PhpMyAdmin is free accessible by network PhpMyAdmin is accessible by network without password. The configuration 'httpd' or 'cookie' in the "config.inc.php" can help.	UNSECURE
A FTP server is not running or is blocked by a firewall! A FTP server is not running or is blocked by a firewall!	UNKNOWN

The green marked points are secure; the red marked points are definitively unsecure and the yellow marked points couldn't be checked (for example because the sofware to check isn't running).

To fix the problems for mysql, phpmyadmin and the xampp directory simply use

=> http://localhost/security/xamppsecurity.php <= [allowed only for localhost]

Some other important notes:

- All these test are made ONLY for host "localhost" (127.0.0.1).
- ***For FileZilla FTP und Mercury Mail, you must fix all security problems by yourself! Sorry.***
- If your computer is not online or blocked by a firewall, your servers are SECURE against outside attacks.
- If servers are not running, these servers are also SECURE!

Please consider this: With more XAMPP security some examples will NOT execute error free. If you use PHP in "safe mode" for example some functions of this security frontend will not working anymore. Often even more security means less functionality at the same time.