

XAMPP Installation

Introducing XAMPP

- An *integration* package containing a number of useful packages that make it easy to host web sites on various platforms.
 - WAMP or LAMP
- Allow the ease of installation and set up
- Main Page:
<http://www.apachefriends.org/en/xampp.html>

Apache – MySQL - PHP



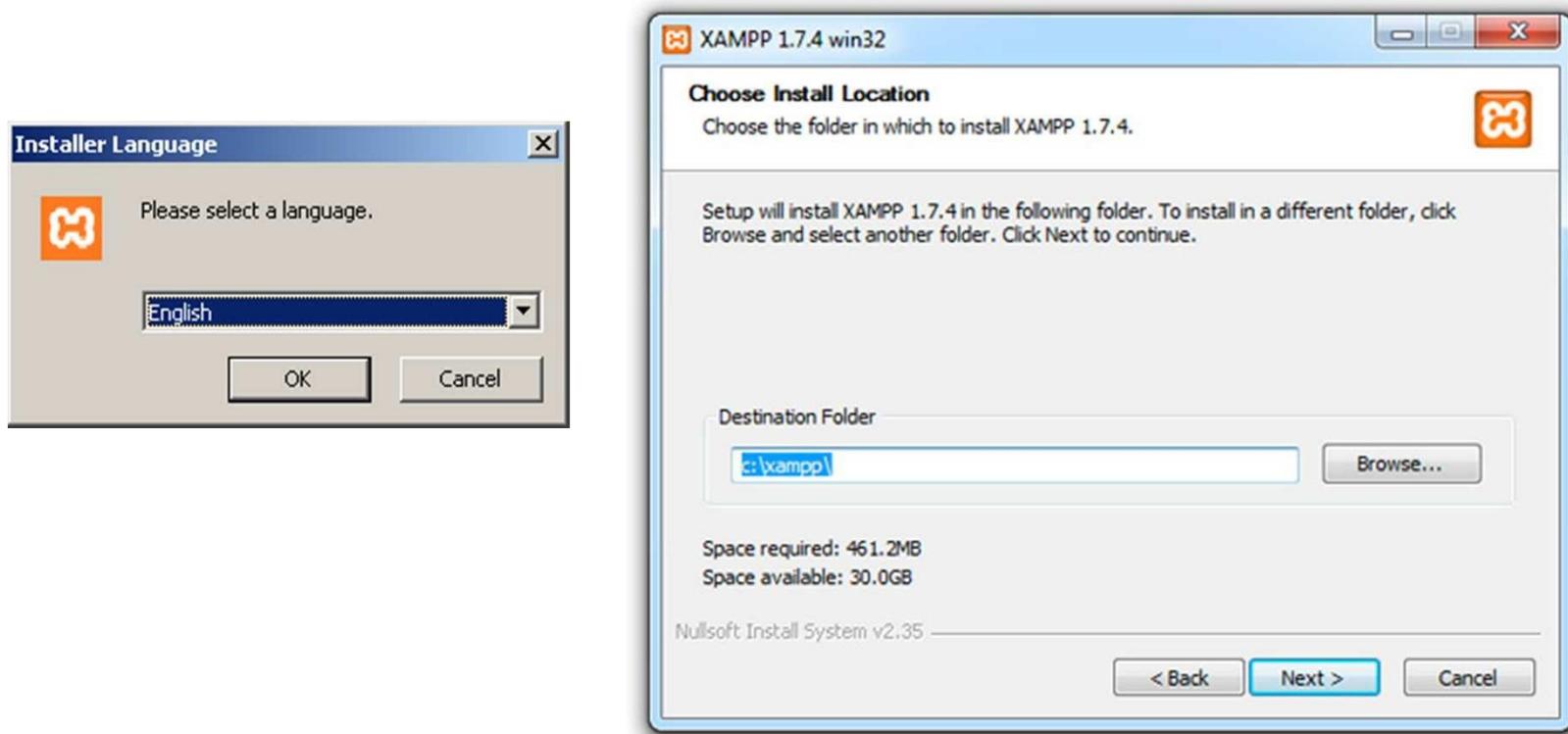
Introducing XAMPP (cont.)

Basic packages include system, programming & server software:

- **Apache**: the famous Web server
- **MySQL**: the widely-used, free, open source database
- **PHP**: the programming language
- **Perl**: the programming language
- **ProFTPD**: an FTP server
- **OpenSSL**: for secure sockets layer support
- **PhpMyAdmin**: for MySQLAdmin.

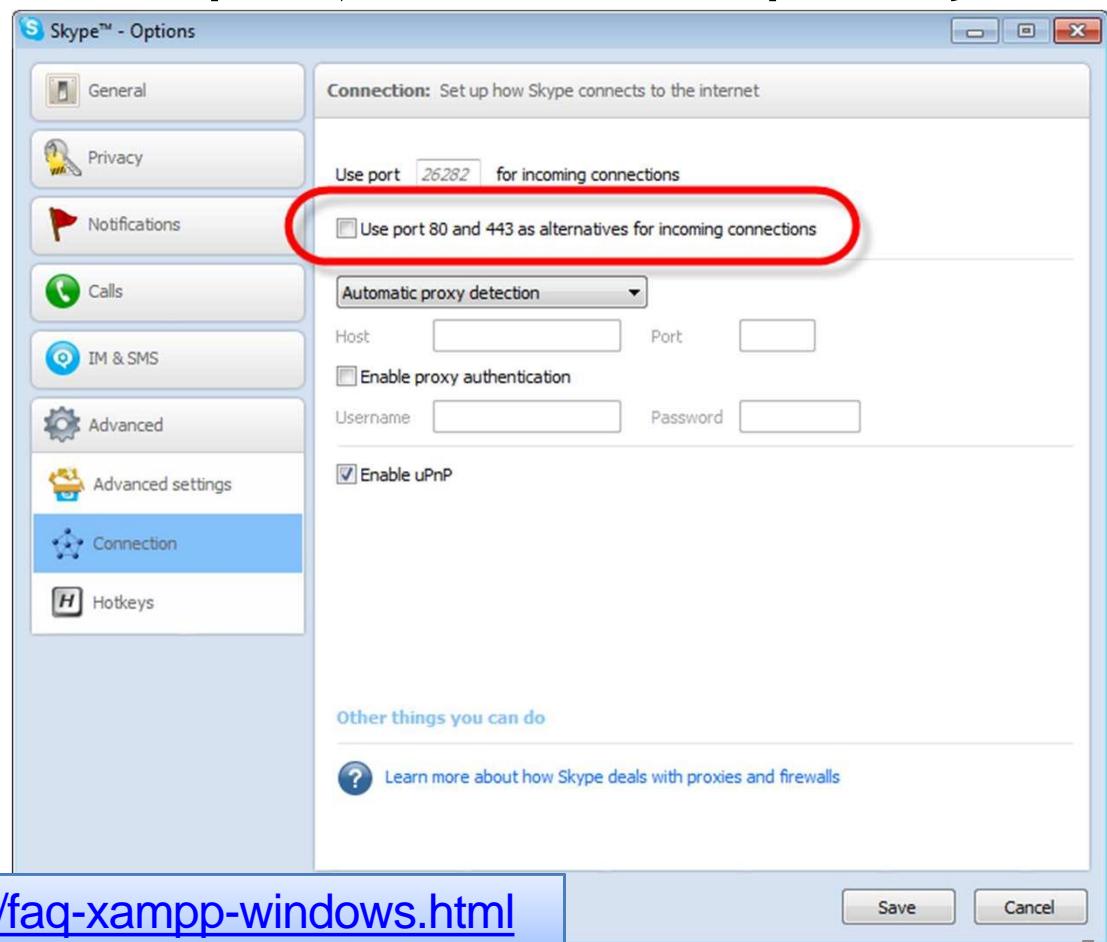
XAMPP Installation

- Download [XAMPP](#) installer and let the install begin:
 - Using the installer version is the easiest way to install XAMPP.
 - Use default directory for convenience



There can be some problems

Port 80 (Apache's default port) can be occupied by other programs €

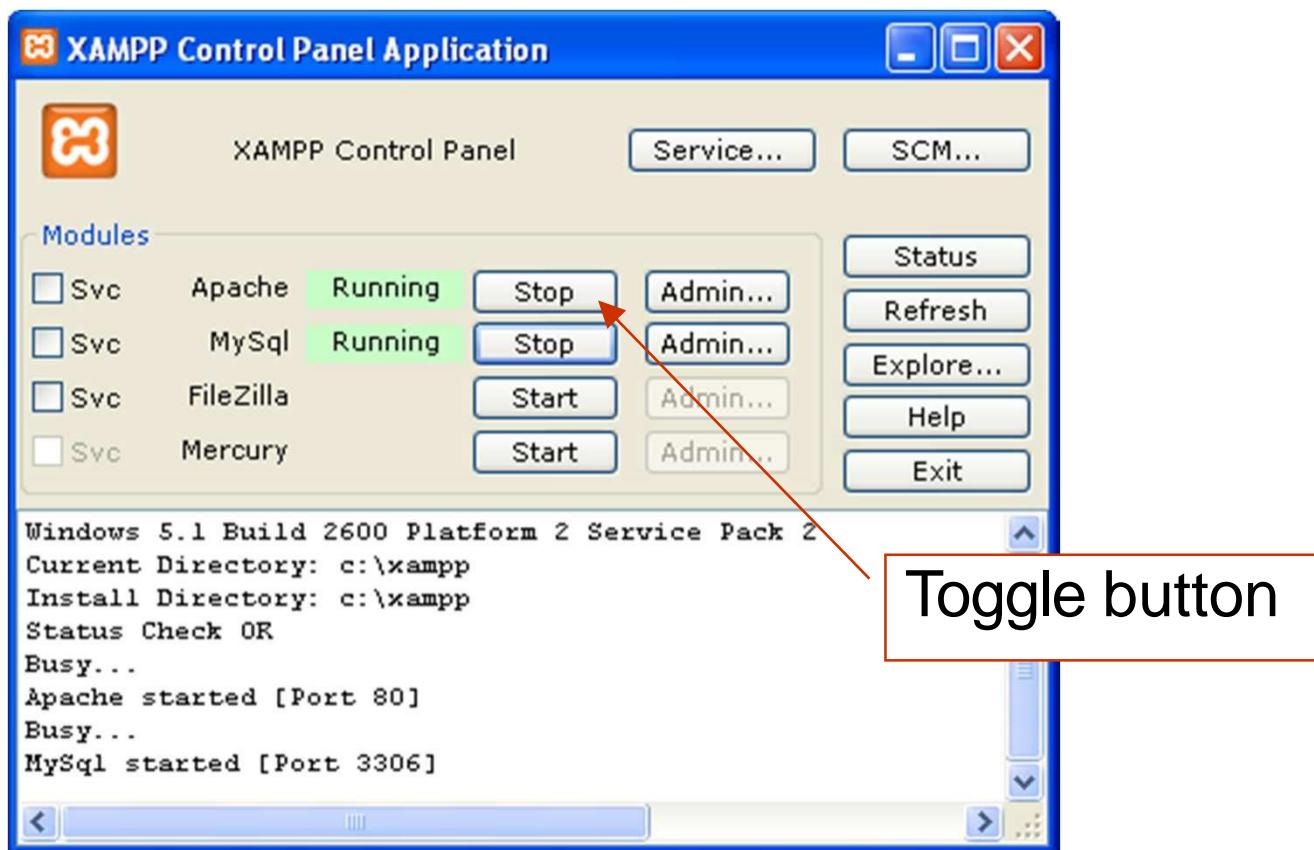


<http://www.apachefriends.org/en/faq-xampp-windows.html>

XAMPP Directories

- XAMPP default installation directory is c:/xampp/
- The directory of interest is “c:/xampp/htdocs/” and it’s called the **webroot** (or document root)
 - PHP files are put in the **webroot** (c:/xampp/htdocs/)
 - c:/xampp/htdocs/ maps to <http://localhost/>
 - For example, c:/xampp/htdocs/project/script.php maps to € <http://localhost/project/script.php>
 - If no file is specified, Apache looks for **index.php**
 - For example, c:/xampp/htdocs/project/ maps to € <http://localhost/project/index.php>

Starting Apache & MySQL



Toggle button

Check the “working environment”

The screenshot shows the XAMPP for Windows control panel. On the left, there's a sidebar with various links like XAMPP [PHP: 5.2.3], Welcome, Status, Security, Documentation, Components, and Demos. The 'Components' section has a 'phpinfo()' link, which is highlighted with a red arrow. The main content area displays the 'PHP Version 5.2.3' page. At the top right of this page is the PHP logo. Below it is a table containing detailed information about the PHP environment, such as the system, build date, configuration command, server API, and various extensions and filters. A blue callout box at the bottom left of the main content area contains the text: "Click on phpinfo() to check the working environment." In the bottom right corner of the main content area, there's a "Powered By Zend Engine 2" logo.

System	Windows NT HOME-5540F4DEE8 5.1 build 2600
Build Date	May 31 2007 09:36:39
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS.0
Loaded Configuration File	C:\xampp\apache\bin\php.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	enabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, data, http, ftp, compress.zlib, zip
Registered Stream Socket Transports	tcp, udp
Registered Stream Filters	convert.iconv.* , string.rot13, string.toupper, string.tolower, string.strip_tags, convert.* , consumed, zlib.*

Click on `phpinfo()` to check the working environment.

PHP Fundamentals

- ➡ • PHP “Hello World” Example
- PHP Variables
- Variable Types
- Working with User Input
- Variable Operators
- Conditional Statements

What is PHP?

- PHP: Hypertext Preprocessor
 - Server-side scripting language
 - Creation of dynamic content
 - Interaction with databases
 - Can be embedded in HTML
 - Open source, written in C
 - First introduced in 1995; at that time PHP stood for Personal Home Page.
 - Similar to Perl and C

Exercise #1: Hello PHP

- The PHP code is usually in files with extension ".php"

<?php denotes start
of PHP code

?> denotes end of
PHP code

- The PHP code can be placed between HTML code.

```
<html>
<head><title>Hello world page</title>
<body>
    <?php echo "Hello PHP!"; ?>
</body>
</html>
```

PHP statements
must be ended with
a semicolon.

Client-side vs. Server-side Script

The diagram illustrates the execution flow of client-side and server-side scripts. On the left, a screenshot of a web browser shows the URL `http://localhost/hello.php`. A blue arrow points from the browser's View menu (which is open) to a code editor window on the right. The code editor displays the original PHP source code:

```
1 <html>
2 <head><title>Hello world page</title></head>
3 <body>
4 Hello HTML!<br>
5 Hello PHP!</body>
6 </html>
7
```

A blue callout box highlights the text: "PHP code is never sent to a client's Web browser; only the HTML output of the processing is sent to the browser." Below the code editor, the browser window shows the rendered HTML output:

```
<html>
<head><title>Hello world page</title></head>
<body>
Hello HTML!<br>
<?php echo "Hello PHP!"; ?>
</body>
</html>
```

Another blue arrow points from the rendered output back to the browser window, indicating the final state.

Client-side vs. Server-side Script

- HTML code is processed by browsers as web pages are loading.
(client-side)
- PHP code is preprocessed by PHP Web servers that parse requested web pages as they are being passed out to the browser.
(Server-side)
- You can embed sections of PHP inside HTML:

```
<BODY>
<p>
<?php $test = "Hello World!";
echo $test; ?>
</p>
</BODY>
```

- Or, you can call HTML from PHP :

```
<?php
echo "<html><head><title>Hello</title>
...
?>
```

PHP Comments

- You can add comments to the code
 - Starting with "//", "#" or block in /*" and "*/"
 - Only "/*" – "*/" can be used over several lines
 - Comments are NOT executed

PHP Fundamentals

- PHP “Hello World” Example
- ➡ • PHP Variables
- Variable Types
- Working with User Input
- Variable Operators
- Conditional Statements

PHP Variables

- What are variables?
 - The values stored in computer memory are called **variables**.
 - The values, or data, contained in variables are classified into categories known as **data types**.
 - The name you assign to a variable is called an **identifier**.

PHP Variables

- Prefixed with a \$ (Perl style)
- Assign values with = operator
- Example: \$name = “John Doe”;
- No need to define type
- Variable names are case sensitive
 - \$name and \$Name are different

PHP Variables

```
<?php // declare string variable $output  
$output = "<b>Hello PHP!</b>";  
Echo $output; € Hello PHP!  
?>
```

- Each variable is declared when it's first assigned value.
- The type of the value determines the type of the variable.

PHP Fundamentals

- PHP “Hello World” Example
- PHP Variables
- Variable Types
- Working with User Input
- Variable Operators
- Conditional Statements

Variable/Data Types

- A **data type** is the specific category of information that a variable contains
- Possible PHP Variable Types are:
 - Numeric (real or integer)
 - Boolean (TRUE or FALSE)
 - String (set of characters)

Variable/Data Types

- Unlike C, PHP is **not** strictly typed.
- PHP decides what type a variable is based on its value.
- PHP can use variables in an appropriate way automatically
- For Example:
 - `$HST = 0.12;` // HST Rate is numeric
 - `echo $HST * 100 . "%";` //outputs “12%”
 - `$HST` is automatically converted to a string for the purpose of echo statement

Displaying Variables

- To display a variable with the echo statement, pass the variable name to the echo statement without enclosing it in quotation marks :
 - \$VotingAge = 18;
 - echo \$VotingAge;
 - To display both text strings and variables, you may used concatenation operator “.” :
 - echo "<p>The legal voting age is ".
\$VotingAge. "</p>";
- 

Naming Variables

- The name you assign to a variable is called an **identifier**
- The following rules and conventions must be followed when naming a variable:
 - Identifiers must begin with a dollar sign (\$)
 - Identifiers may contain uppercase and lowercase letters, numbers, or underscores (_).
 - The first character after the dollar sign must be a letter.
 - Identifiers cannot contain spaces or special characters.
 - Identifiers are **case sensitive**

Declaring and Initializing Variables

- Specifying and creating a variable name is called **declaring the variable**
- Assigning a first value to a variable is called **initializing the variable**
- In PHP, you must declare and initialize a variable in the same statement:
 - `$variable_name = value;`

PHP Strings

- String values
 - Strings may be in single or double quotes

```
<?
$output1 = "Hello PHP!";
$output2 = 'Hello again!';
?>
```

- Start and end quote type should match
- Difference between two types of quotes is the escape sequences

Strings escaping

- Special chars in strings are escaped with backslashes (C style)
- Double-quoted string

```
$str1 = "this is \"PHP\\\""; echo $str1;  
€ outputs this is "PHP"
```

- Single-quoted string

```
$str2 = ' "I\\'ll be back ", he said. ' ; echo $str2;  
€ outputs "I'll be back", he said.
```

Variables in strings

- Double quoted strings offer something more:

```
$saying = "I'll be back!";
$str1 = "He told me: $saying";
outputs € He told me: I'll be back!
```

- Variables are evaluated in double-quoted strings, but not in single-quoted strings.

```
$saying = "I'll be back!";
$str2 = 'He told me: $saying';
outputs € He told me: $saying
```

- For single-quoted strings, use concatenation:

```
$saying = "I'll be back!";
$str3 = 'He told me: ' . $saying;
outputs € He told me: I'll be back!
```