Lebanese University
Faculty of Science
Computer Science BS Degree

Server-side Web Development (I3302)

Server-side Web Development (I3302)
Grading

	Session 1	Session 2	Session 2
Mid-term exam	22 pts		X
Project	25 pts		
Final Exam	53 pts	53 pts	75 pts
Total	**100 pts**		

About

- Prerequisite
  - Already followed the course on introduction to web development [HTML, CSS, JavaScript]
  - Already followed one or more programming courses
  - Already followed a course on databases

Syllabus

1. Web architecture
2. PHP language
3. PHP forms
4. Sessions
5. Connection with a DBMS (MySQL)
6. Ajax and jQuery
7. Security(injections)
8. Introduction to XML and XSLT

Web architecture

Aim

- Understand
  - Web architecture
  - What is PHP
  - How a PHP script works with a Web browser and a Web server
- Discover the software used to begin with PHP

Web elements

- some define the Web as
  - a network application based on the use of TCP / IP
- Clients → software: browsers
  - Chrome, Firefox , Microsoft Edge, …
- a web server usually listens on TCP port number 80
- the dialogue between a client and a server is defined by
  - the hypertext transfer protocol (HTTP)
- ➔ thus
  - HTTP is a protocol of the network application based on the use of TCP / IP

Uniform Resource Locators (URL)

- URLs locate documents (texts, images, sounds, files …) on the Internet by:
  - the used protocol : http, ftp, file, …
  - the address of the machine (server) publishing the document
  - the path to access this document on the server
  - the additional information (parameters …)
- Relative URL
  - specify the path to a document published by the same server
  - relative to the position of the document that includes the link
- format of an absolute URL:

protocol://server_name[:port][/path][?list_of_parameters]

Uniform Resource Locators (URL)

protocol://server_name[:port][/path][?list_of_parameters]

Examples:

http://www.fsciences.ul.edu.lb/site/profile.php?id=294

The main protocols

- http
  - hyper text transfert protocol
  - web page transfer
  - uses TCP port 80 by default
- ftp
  - file transfert protocol
  - transfer files
  - uses TCP ports 20 and 21
- file
  - indicates a locally accessible file
  - on the client machine
  - useful for example to read HTML documentation

HTTP protocol

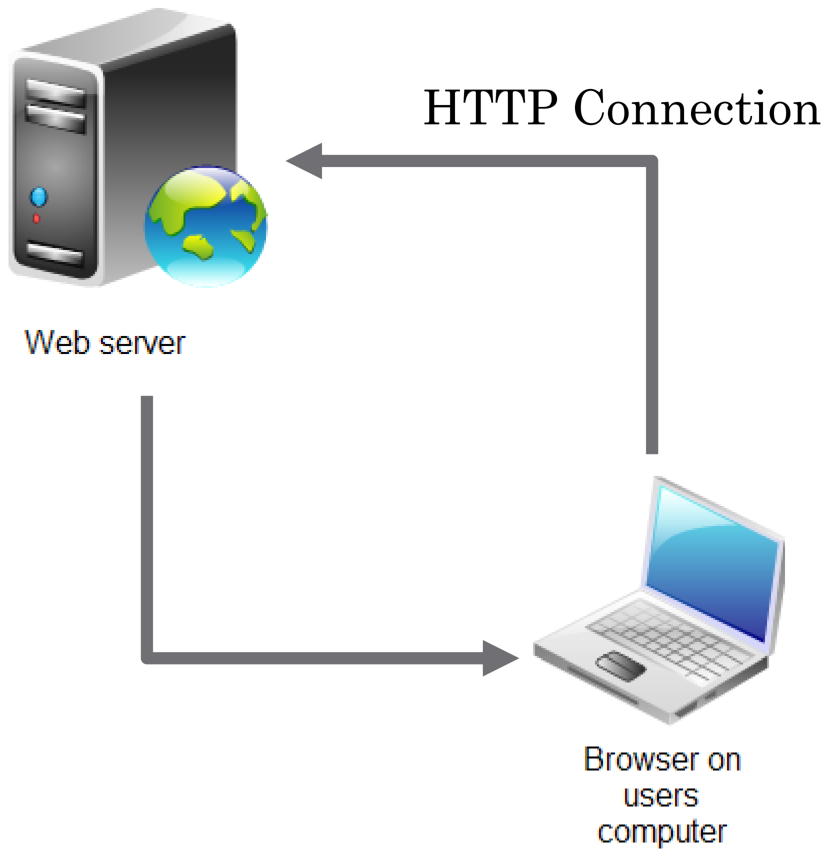
- the protocol used by the web
- version 1.0 → transfer messages
  - headers describing the contents of the message
  - using a MIME type encoding (as for emails)
- Goal
  - allow file transfer
  - identified by their URL
  - between a client (browser) and a web server

Standard MIME

- *Multipurpose Internet Mail Extensions*
  - standard proposed by the *Bell Communications Laboratories* in 1991
- For
  - extend the limited e-mail capabilities
  - to allow to insert documents (images, sounds, text, ...) in a mail
  - describe, through headers, the type of message content and the encoding used
- features:
  - have multiple attachments
  - unlimited message length
  - use of character sets other than ASCII
  - use rich text
  - binary attachments: executables, images, audio or video files, etc.

Classic Scenario

1. the client uses the browser to communicate with the server
2. the client uses the HTTP protocol to request an HTML document
3. the server sends the document to the browser
4. the browser displays the document as defined by HTML



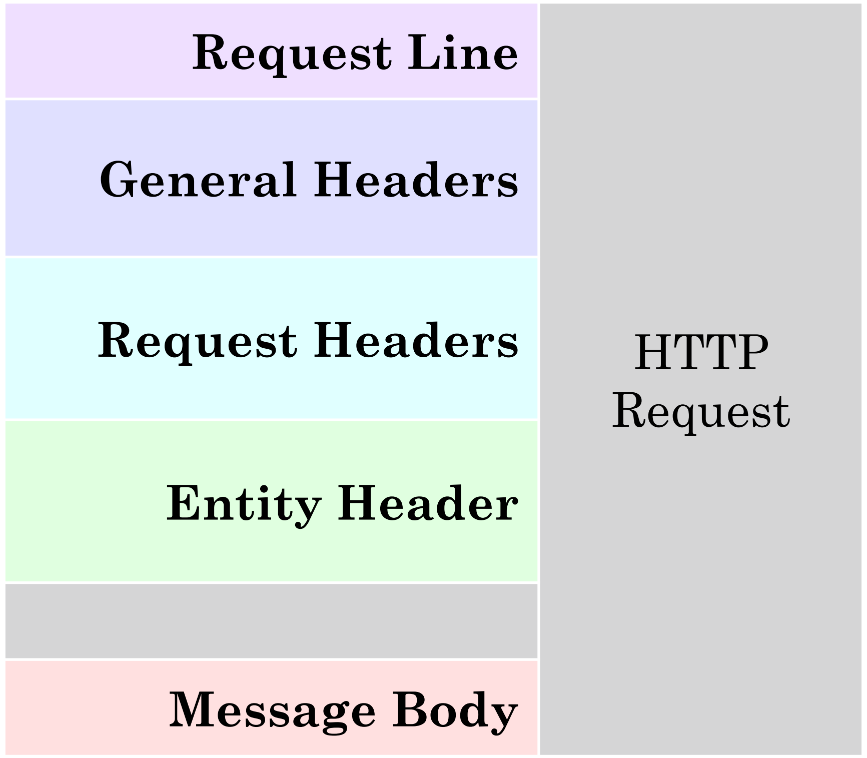
HTTP requests

- sent by the browser → server
- set of lines comprising :

1. a query line specifying:
   - the method
   - URL
   - the protocol version used by the client - example HTTP / 1.0

2. the header fields of the query
   - set of optional lines
   - additional information about the request and / or the client
   - name: value of the header

3. the body of the request
   - set of optional lines separated from previous lines by an empty line
   - allows for example a sending by a POST command



Example of HTTP **GET** requests

1. Just one request line
2. the header fields of the query
3. ~~the body of the query~~

- *Example*

GET /index.html HTTP/1.1	**Request Line**	HTTP Request
Date: Thu, 20 May 2017 21:12:55 GMT Connection: close	**General Headers**	
Host: www.bbbbb.me From: blabla@somewebsite.com Accept: text/html, text/plain User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	**Request Headers**	
	Entity Header	
	Message Body	

Example of HTTP **POST** requests

1. Just one request line
2. the header fields of the query
3. the body of the query

- *Example*

POST http://www.bbbbbbbbb.me/contacts.php HTTP/1.0	**Request Line**	HTTP Request
Date: Thu, 20 May 2017 21:12:55 GMT Connection: Keep-Alive	**General Headers**	
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	**Request Headers**	
Content-Type: application/x-www-form-urlencoded Content-Length: 48	**Entity Header**	
name=foo&age=23&country=virgin+islands	**Message Body**	

The most useful commands

Command	Description
`GET`	Query of the resource located at the specified URL.
`HEAD`	Query of the resource located at the specified URL, the response will not contain a body. Only the response headers will be sent to the client without the body.
`POST`	Sending data to the program at the specified URL
`PUT`	Sending data to the specified URL
`DELETE`	Delete the resource located at the specified URL, a DELETE query does not contain a body.

The **GET** method

- The first method that has emerged
- asks the server for the resource → to read it
- method used by web browsers
  - URL in address bar
  - click on an HTML link
- large percentage of total web requests
- possible to pass parameters
  - "query string" → variables after "**?**", separated by "**&**"
- ***GET***
  - should never involve a change to the server-side resource
  - is used to read data, not to send it, so no body
  - should not contain any information which may vary over time
    - can be cached
    - a web link can be put in favorites or shared among several people

POST

- provided by HTTP / 1.0
- POST can send an unlimited number of data to a Web server
- put the data in the body of the query after the header

POST /contacts.php HTTP/1.0	**Request Line**	HTTP Request
Referer: http://www.somedomain.com/directory/file.html Date: Thu, 20 May 2017 21:12:55 GMT Connection: Keep-Alive	**General Headers**	
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	**Request Headers**	
Content-Type: application/x-www-form-urlencoded Content-Length: 48	**Entity Header**	
name=nnnn+kkkk&Phone=01255881	**Message Body**	

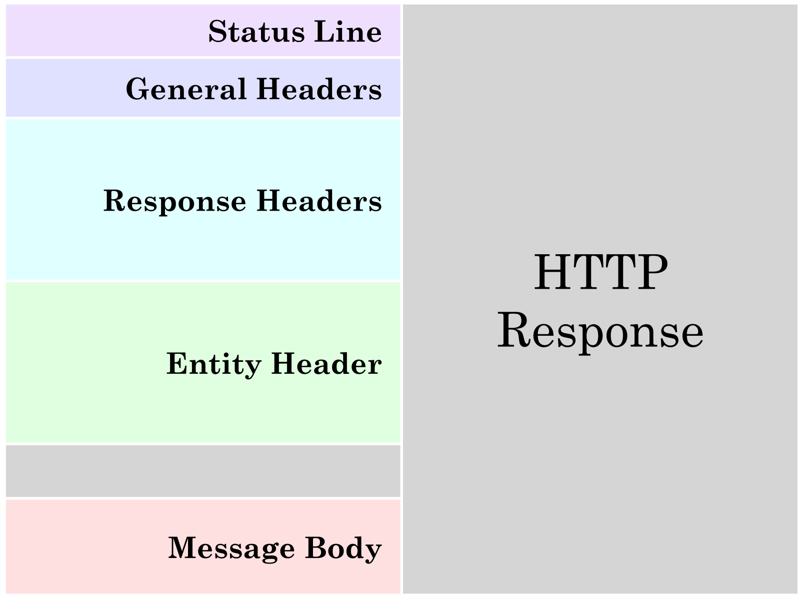
The most useful requests headers

Header Name	Description
Accept	Type of content accepted by the browser (eg text / html). MIME type.
Accept-Encoding	Data encoding accepted by the browser.
Accept-Language	Language expected by the client (default is English).
Authorization	Identification of the browser with the server.
Content-Encoding	The type of the body of the request.
Content-Language	The language of the request body.
Content-Length	Length of the request body.
Content-Type	Type of body content of the request (eg text / html). MIME type.
Date	Date of start of data transfer.
From	Specifies the email address of the client.
If-Modified-Since	Should only be sent if it has been changed since a certain date.
Orig-URL	The original URL of the request.
Referer	URL of the link from which the request was made.
User-Agent	Client information, name and version of browser, OS, etc.

HTTP response

- sent by the server → browser
- set of lines comprising:

1. a status line :
   - the protocol version
   - the status code
   - the meaning of the code
2. the header fields of the query (optional)
   - additional information about the response and/or server
   - name of header type: header value
3. the body of the response
   - contains the requested document



Example of an HTTP response

1. a status line
2. header fields
3. the body of the respons

HTTP/1.1 200 OK	**Status Line**	HTTP Response
Date: Thu, 20 May 2017 21:12:55 GMT Connection: close	**General Headers**	
Server: Apache/1.3.27 Accept-Ranges: bytes	**Response Headers**	
Content-Type: text/html Content-Length: 170 Last-Modified: Tue, 18 May 2017 10:14:49 GMT	**Entity Header**	
<html> <head><title>Welcome to the amazing site!</title></head> <body> <p>This site is under construction. Please come back later. Sorry!</p> </body> </html>	**Message Body**	

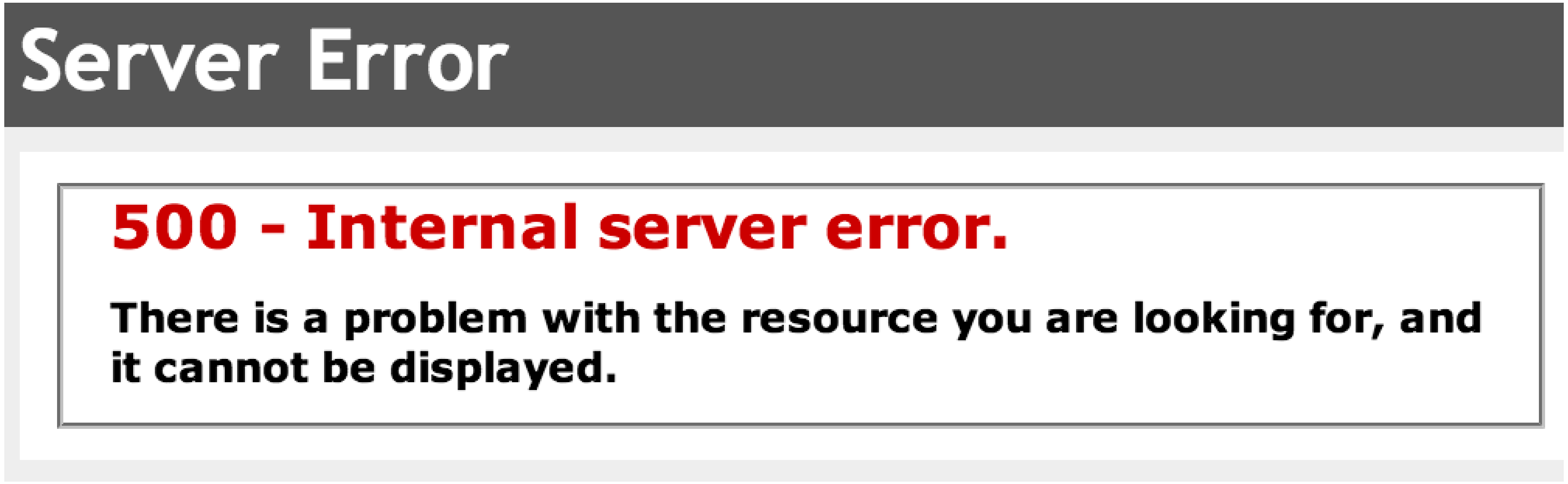
The most useful response headers

Name of the header	Description
Content-Encoding	Type of body encoding of the response.
Content-Language	Type of body language of the response.
Content-Length	Length of the response body .
Content-Type	Type of body content of the response. MIME type.
Date	Date of start of data transfer.
Expires	Deadline for data consumption.
Location	Redirect to a new URL.
Server	Characteristics of the server.

HTTP Error Codes

Code	Class	Usage
1xx	Information	not used, for future use
2xx	Success	the action was correctly received, interpreted, and executed
3xx	Redirection	an additional decision must be taken to complete the request
4xx	Client error	the request has an error of form and can not be satisfied
5xx	Server error	the request is valid, but the server can not satisfy it

Examples of HTTP Error Codes

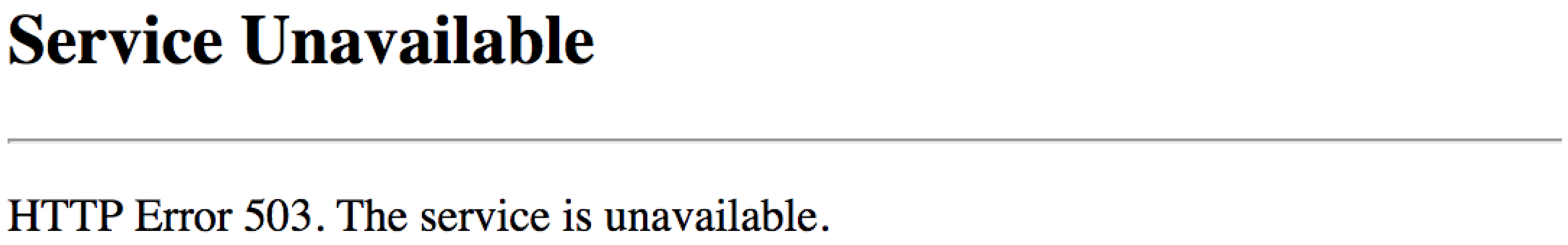


Server Error

500 - Internal server error.

There is a problem with the resource you are looking for, and it cannot be displayed.

Examples of HTTP Error Codes



Service Unavailable

HTTP Error 503. The service is unavailable.

Dynamic web applications

Dynamic web applications

- HTML-coded documents
  - simple formatting (text, images, forms)
- in some cases→ we need more functionality
  - More sophisticated visualizations: rich graphical interfaces, interactive 3D, ...
  - Exchange of persistent information in both directions between the client and the server: e-commerce
- 2 techniques
  - code execution by the client
    - in the web browser
  - code execution by the server
    - within the web server or by establishing a communication with another server

Client-side execution

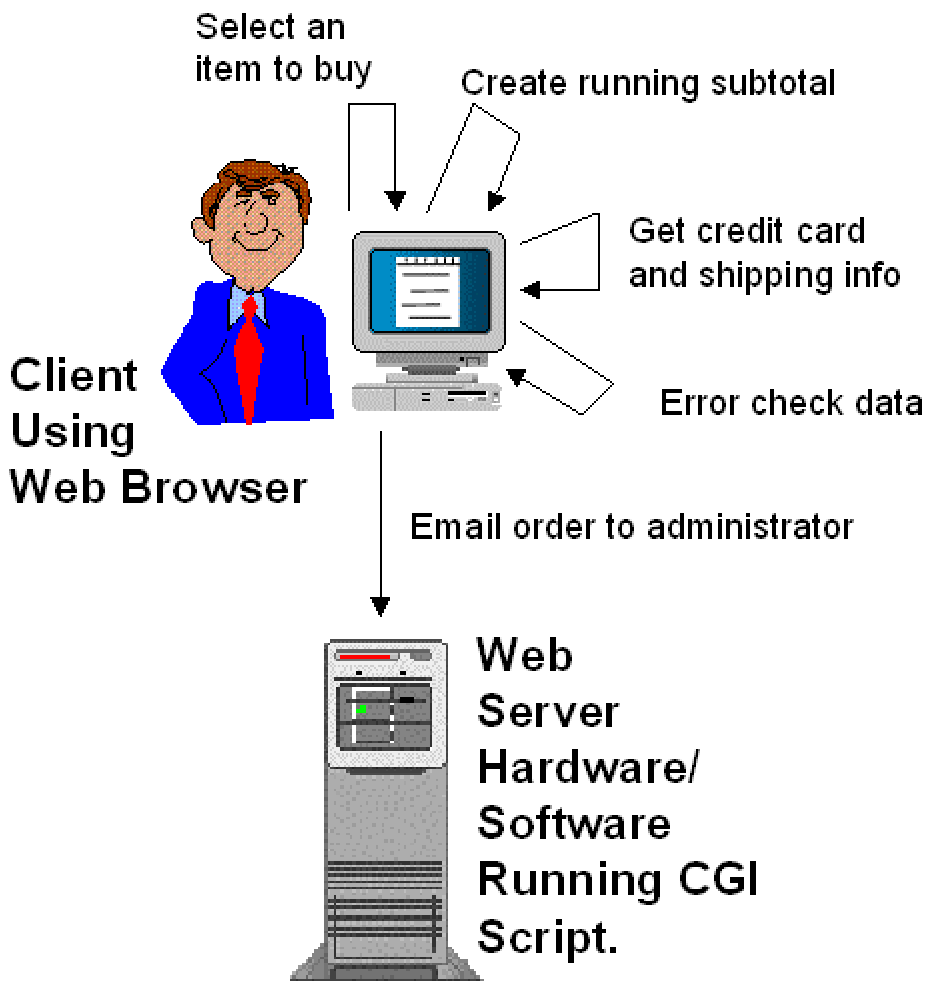
- 2 advantages:
  - the server has less work
  - network traffic is reduced
- 2 disadvantages:
  - clients are heterogeneous
    - hardware architecture, operating system, browser type
    - requires the use of a standardized programming language
  - the arbitrary execution of code by the client poses significant security problems

Client-side execution

- 2 types of solutions :
  - allow code execution from trusted servers
  - strict control of the environment in which the code will run on the client
- execution of client-side code triggered by an HTML document using
  - `<SCRIPT>` or `<APPLET>` tags

➔it is not simple to build a serious web application with code running on clients

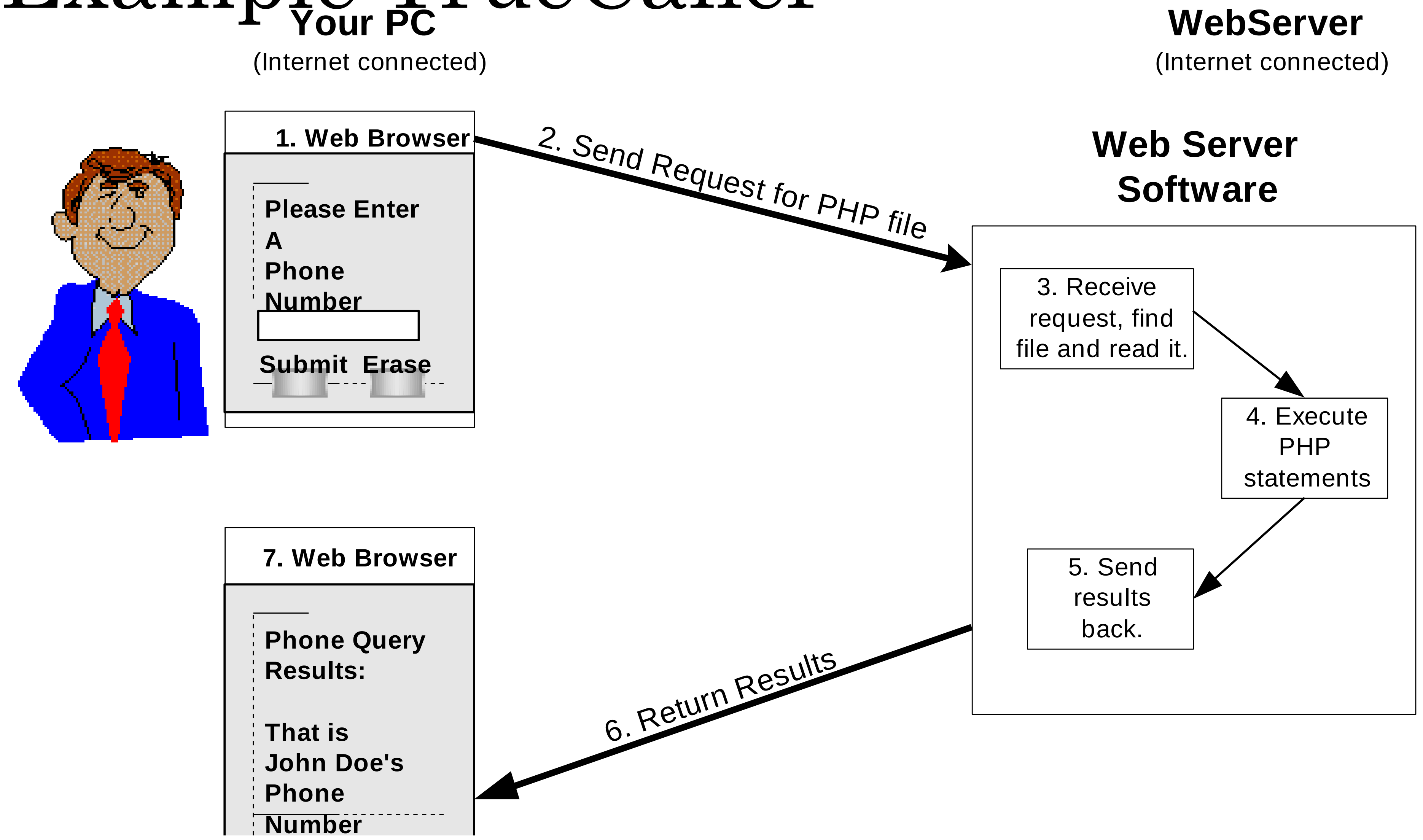
Client-side script


Select an item to buy
Create running subtotal
Get credit card and shipping info
Error check data
Client Using Web Browser
Email order to administrator
Web Server Hardware/ Software Running CGI Script.

Server-side execution

- increase server functionality
  - static pages →
    - serve stored documents only
  - dynamic pages →
    - execute instructions to respond to each request
    - and generate the appropriate response (HTML)

Server-side execution Example TrueCaller



ASP, PHP, Servlets, ...

- Active Server Pages (ASP)
  - solution proposed by Microsoft (scripts in Visual Basic or C# embedded in HTML pages)
- PHP : a scripting language developed to be integrated into HTML pages
  - the PHP syntax vaguely resembles that of the C language
  - instructions located in the `<?php .... ?>`
  - executed
  - replaced before sending to the customer
- Servlets : or Java Server Pages
  - solution developed by Sun MicroSystems
  - uses JAVA code run by the server and can be integrated into HTML pages

Which popular sites run on which platforms?

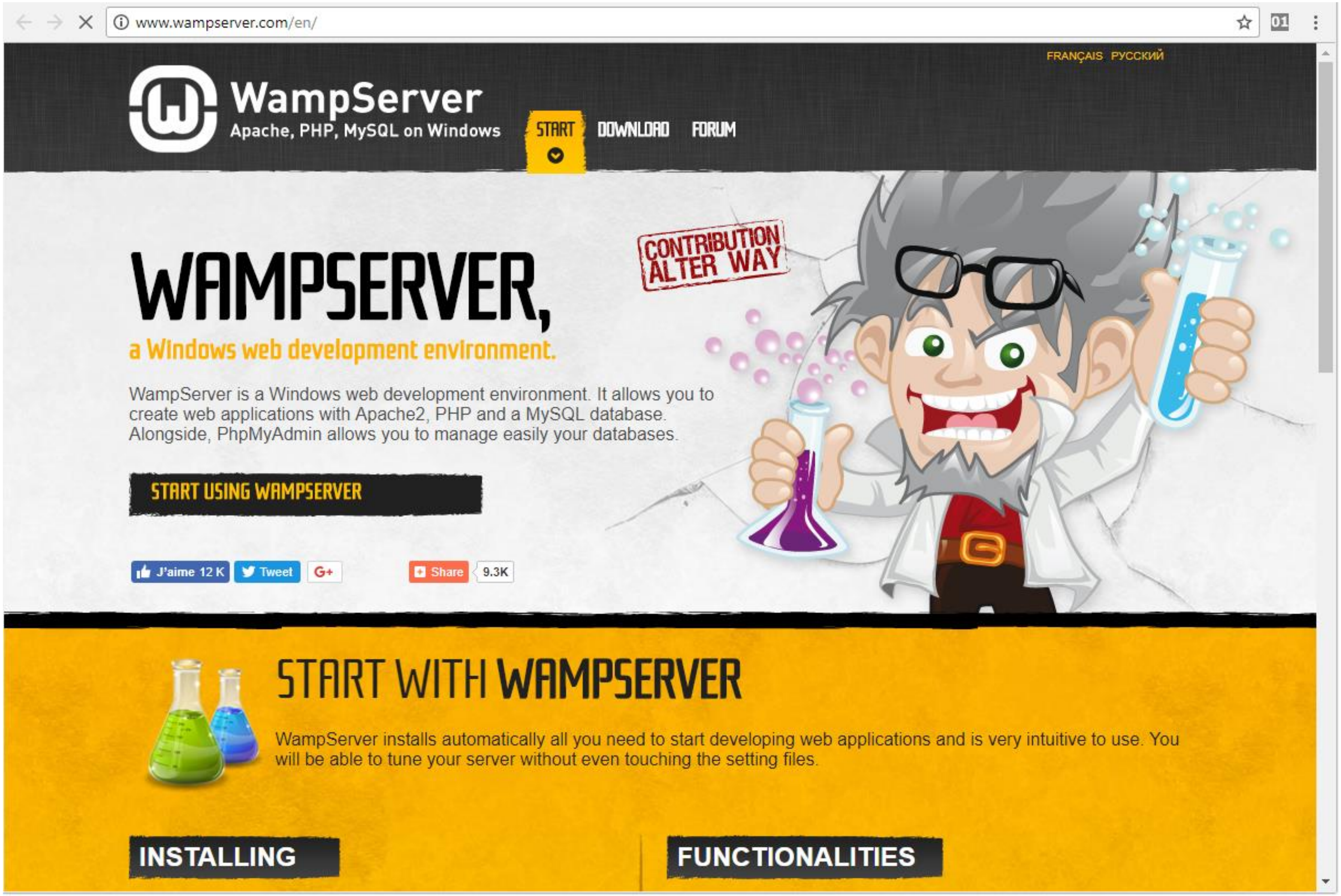
Site	Up Since	Server Platform	Programming Language
Google.com	November 1998	Linux	C, Java, C++, PHP & MySQL
Facebook.com	February 2004	Linux	PHP, MySQL and C++
YouTube.com	February 2005	Linux	C, Java and MySQL
Yahoo.com	August 1995	Linux	C++, C, Java, PHP & MySQL
MSN.com (owned by Microsoft)	August 1995	Windows	ASP.net
Live.com (owned by Microsoft)	August 2008	Windows	ASP.net
Wikipedia	January 2001	Linux	PHP & MySQL
Amazon.com	October 1995	Linux & Solaris	C++, Java, J2EE
WordPress.com	November 2005	Linux	PHP & MySQL

http://www.comentum.com/php-vs-asp.net-comparison.html

Start with PHP

Start with PHP

- To build and publish PHP scripts, you need:
  - A Web server including PHP
  - A client machine with a basic text editor and an Internet connection
  - FTP or Telnet Software



WAMPSERVER,

a Windows web development environment.

WampServer is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows you to manage easily your databases.

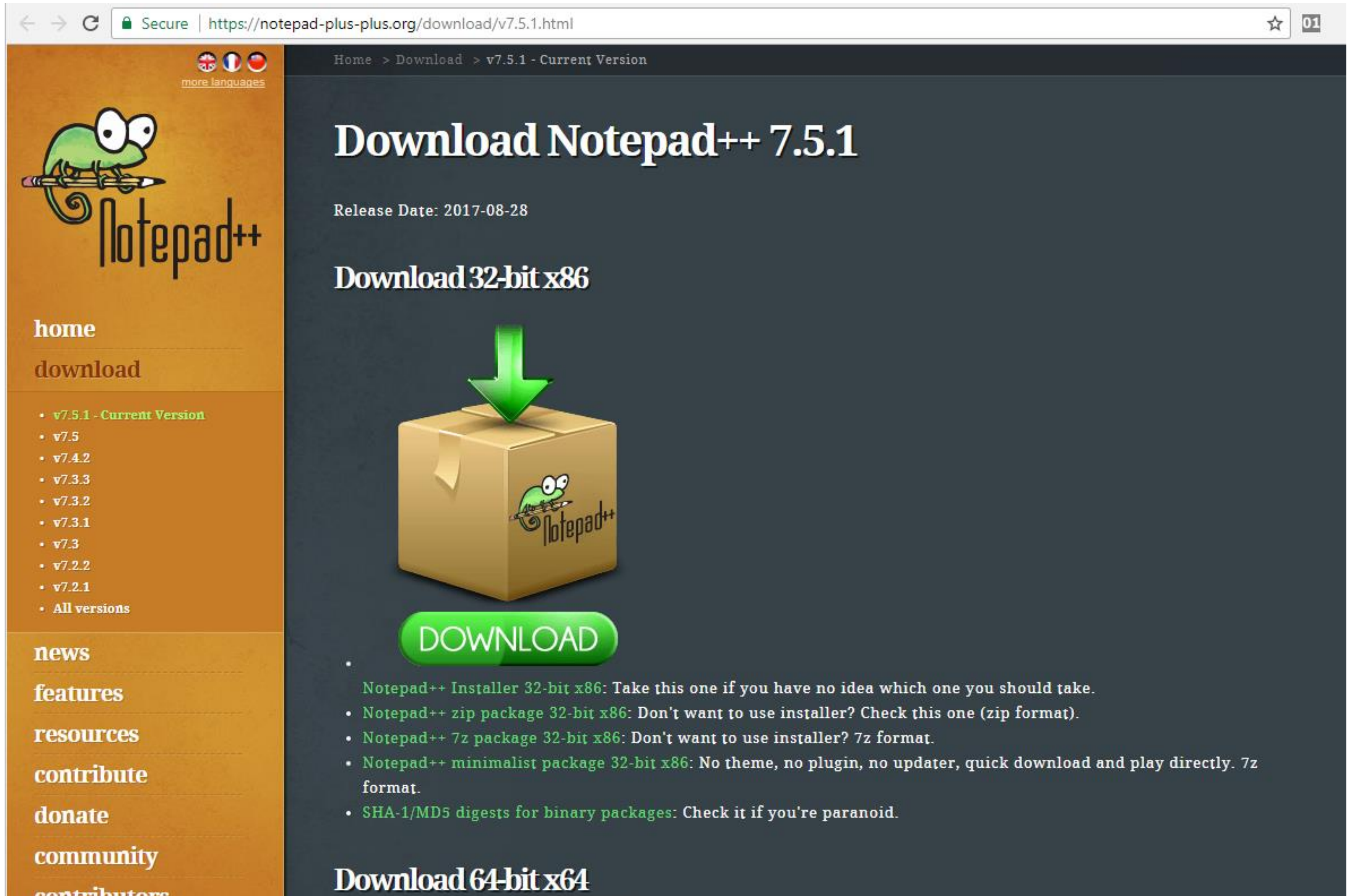
START WITH WAMPSERVER

WampServer installs automatically all you need to start developing web applications and is very intuitive to use. You will be able to tune your server without even touching the setting files.

PHP development process

1. Create a PHP script in a file and save it to a local disk.
2. Use FTP to copy the file to the server.
3. Access the file using a browser.

1. Create a PHP script in a file and save it to a local disk.


Secure | https://notepad-plus-plus.org/download/v7.5.1.html

Home > Download > v7.5.1 - Current Version

Download Notepad++ 7.5.1

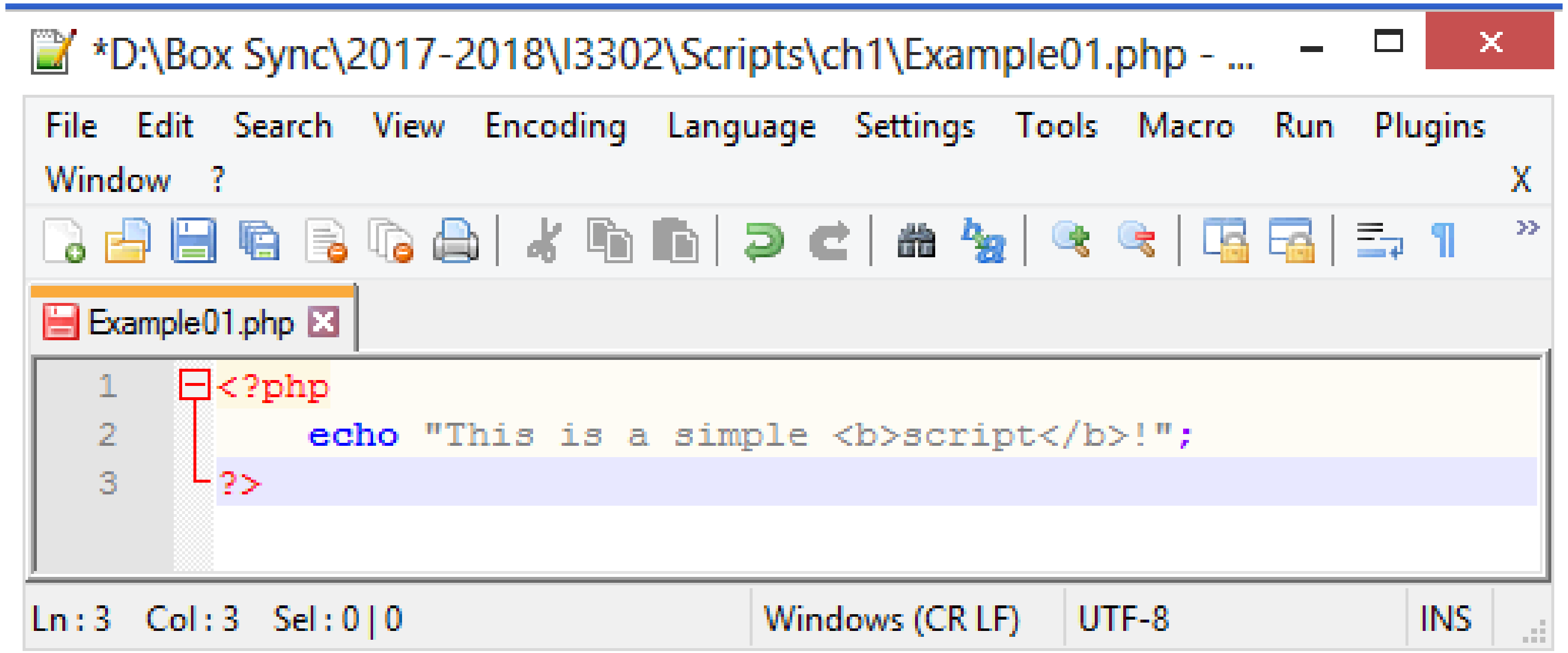
Release Date: 2017-08-28

Download 32-bit x86

- Notepad++ Installer 32-bit x86: Take this one if you have no idea which one you should take.
- Notepad++ zip package 32-bit x86: Don't want to use installer? Check this one (zip format).
- Notepad++ 7z package 32-bit x86: Don't want to use installer? 7z format.
- Notepad++ minimalist package 32-bit x86: No theme, no plugin, no updater, quick download and play directly. 7z format.
- SHA-1/MD5 digests for binary packages: Check it if you're paranoid.

Download 64-bit x64

1. Create a PHP script in a file and save it to a local disk.

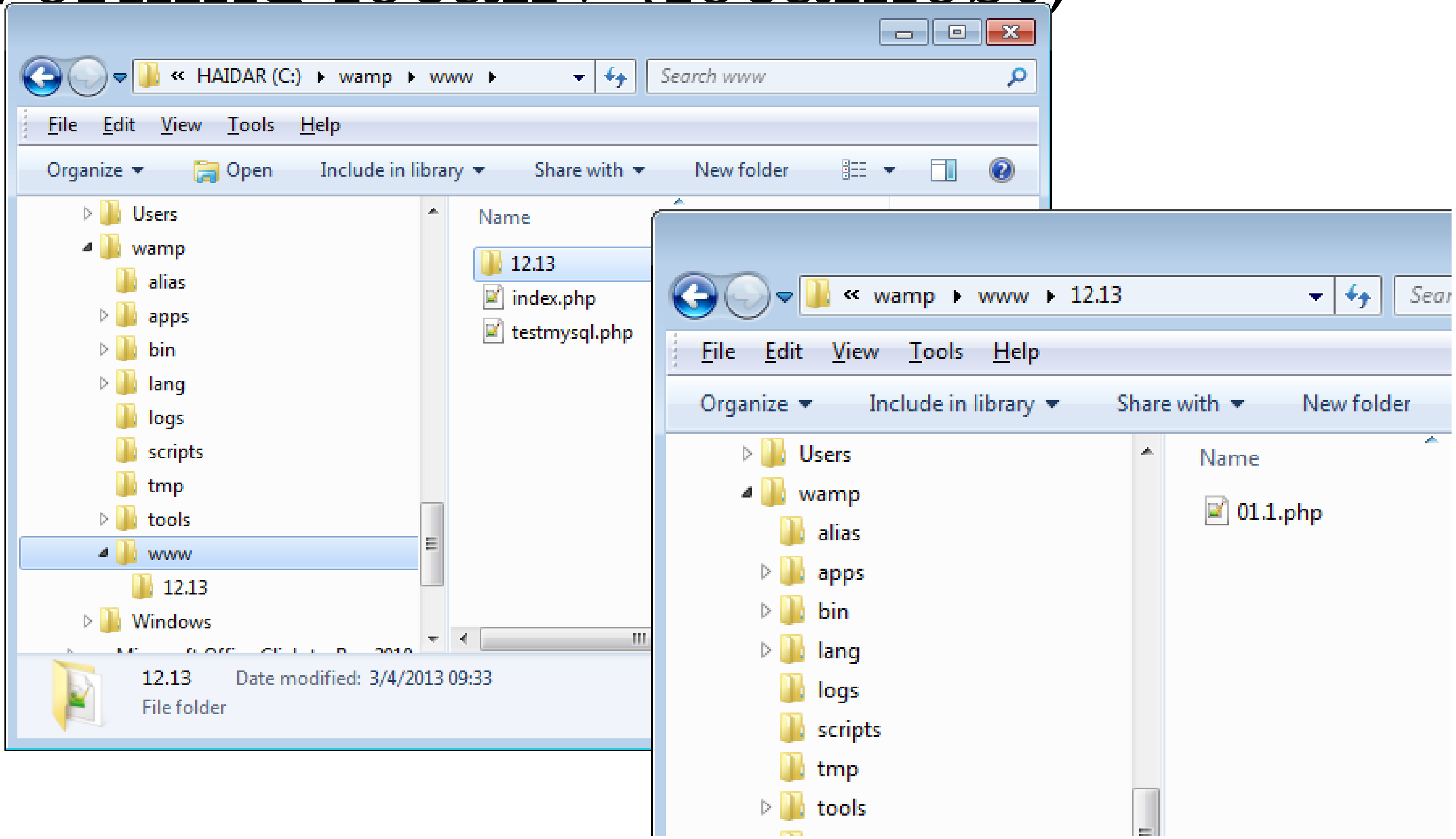


*D:\Box Sync\2017-2018\I3302\Scripts\ch1\Example01.php - ...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins
Window ?
Example01.php

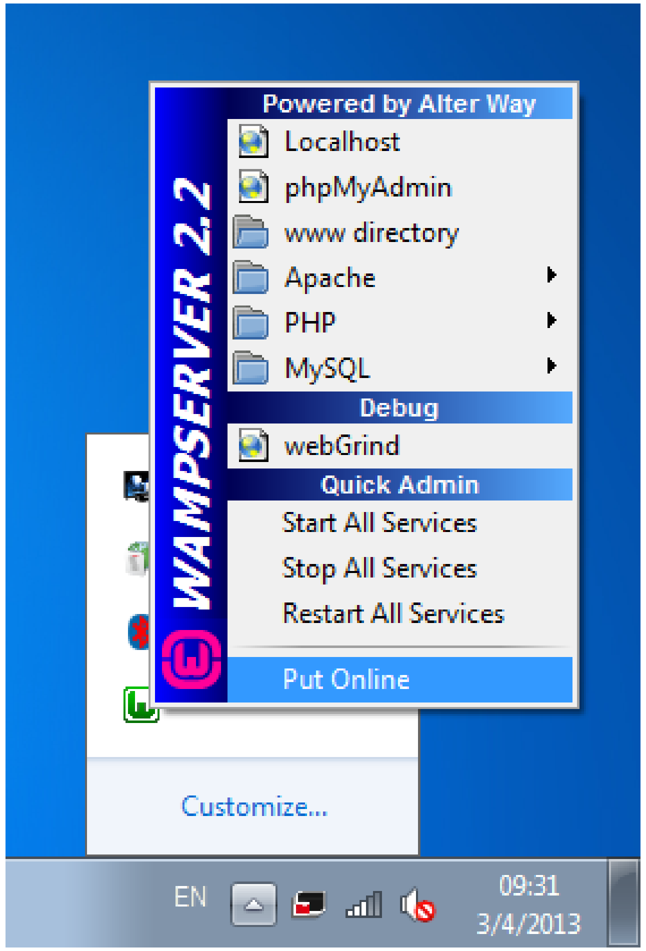
```
<?php
    echo "This is a simple <b>script</b>!";
?>
```

Ln : 3 Col : 3 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS

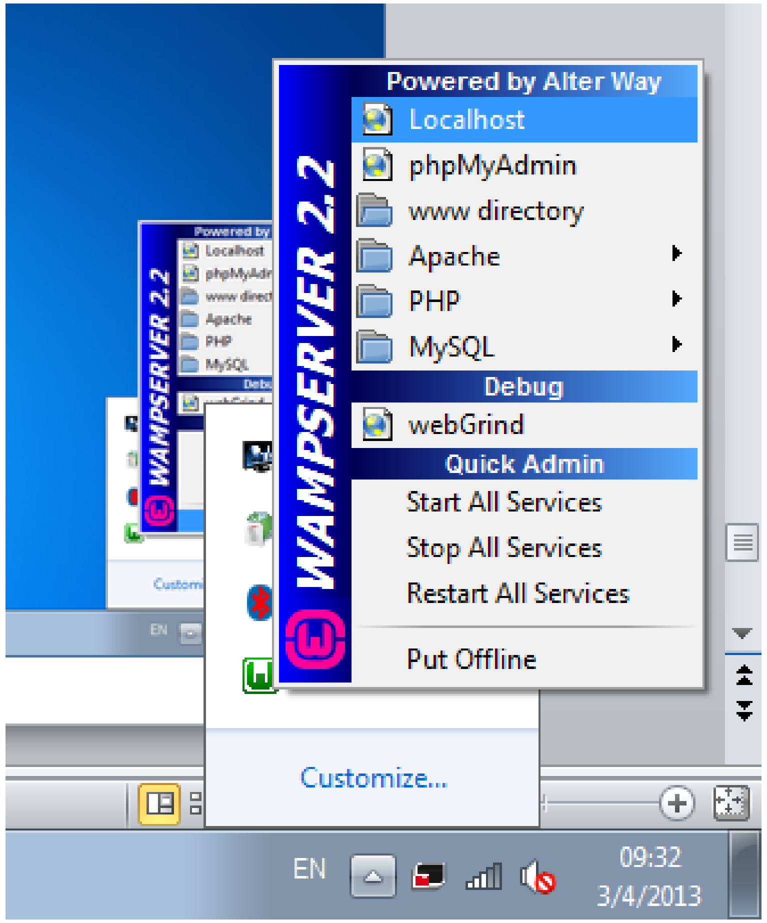
Working locally (localhost)


« HAIDAR (C:) ▸ wamp ▸ www ▸
Search www
File Edit View Tools Help
Organize ▾ Open Include in library ▾ Share with ▾ New folder
Users
wamp
alias
apps
bin
lang
logs
scripts
tmp
tools
www
12.13
Windows
Name
12.13
index.php
testmysql.php
12.13 Date modified: 3/4/2013 09:33
File folder
« wamp ▸ www ▸ 12.13
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ New folder
Users
wamp
alias
apps
bin
lang
logs
scripts
tmp
tools
Name
01.1.php

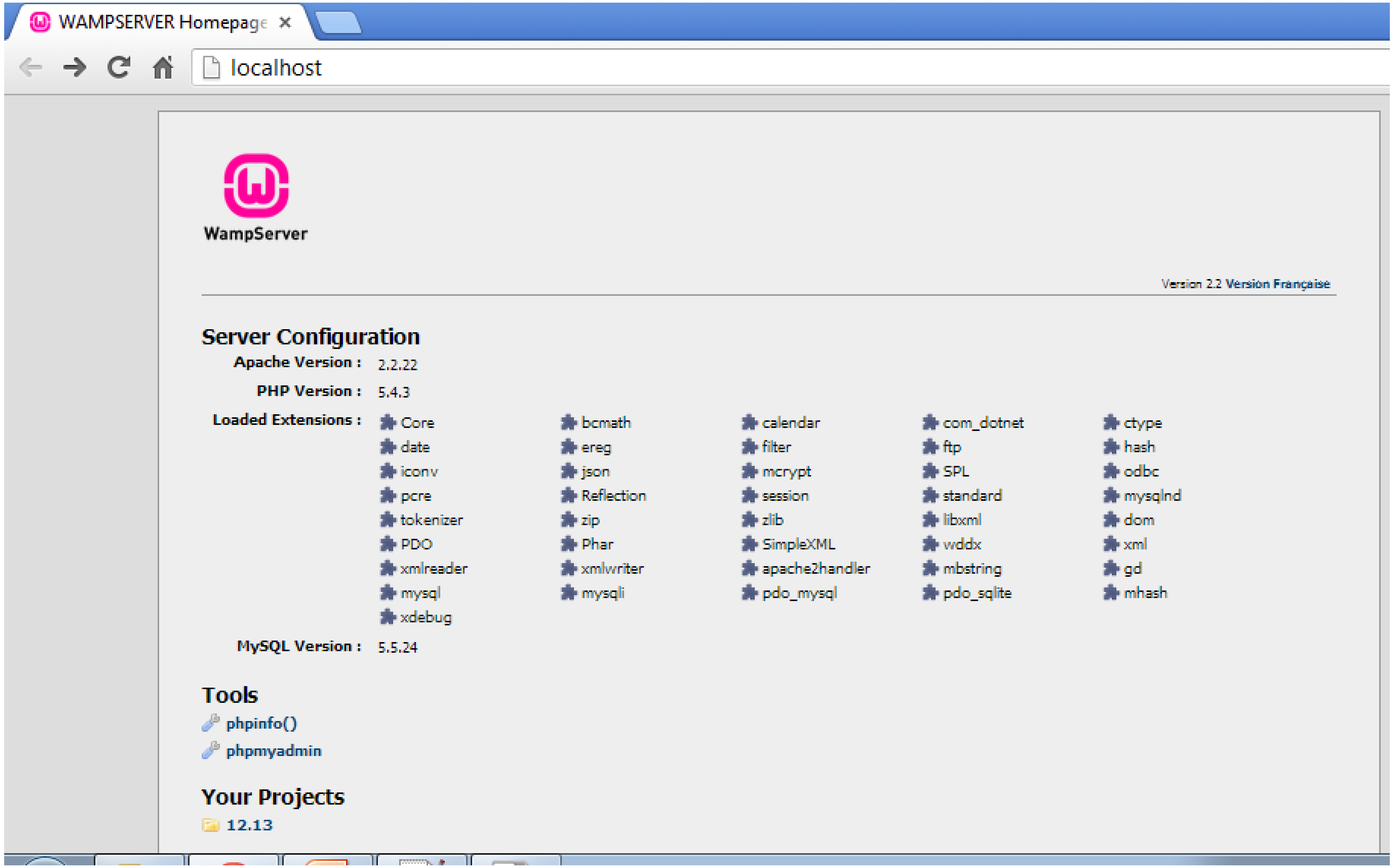
Start WAMP



Powered by Alter Way
WAMPSERVER 2.2
Localhost
phpMyAdmin
www directory
Apache
PHP
MySQL
Debug
webGrind
Quick Admin
Start All Services
Stop All Services
Restart All Services
Put Online
Customize...
EN
09:31
3/4/2013



Powered by Alter Way
WAMPSERVER 2.2
Localhost
phpMyAdmin
www directory
Apache
PHP
MySQL
Debug
webGrind
Quick Admin
Start All Services
Stop All Services
Restart All Services
Put Offline
Customize...
EN
09:32
3/4/2013



Server Configuration

Apache Version :	2.2.22
PHP Version :	5.4.3
Loaded Extensions :	Core, bcmath, calendar, com_dotnet, ctype, date, ereg, filter, ftp, hash, iconv, json, mcrypt, SPL, odbc, pcre, Reflection, session, standard, mysqlnd, tokenizer, zip, zlib, libxml, dom, PDO, Phar, SimpleXML, wddx, xml, xmlreader, xmlwriter, apache2handler, mbstring, gd, mysql, mysqli, pdo_mysql, pdo_sqlite, mhash, xdebug
MySQL Version :	5.5.24

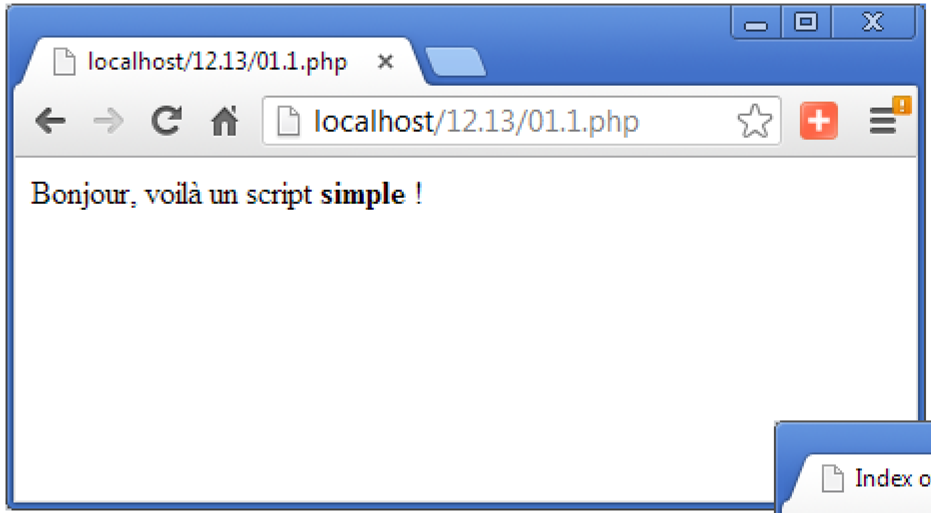
Tools

- phpinfo()
- phpmyadmin

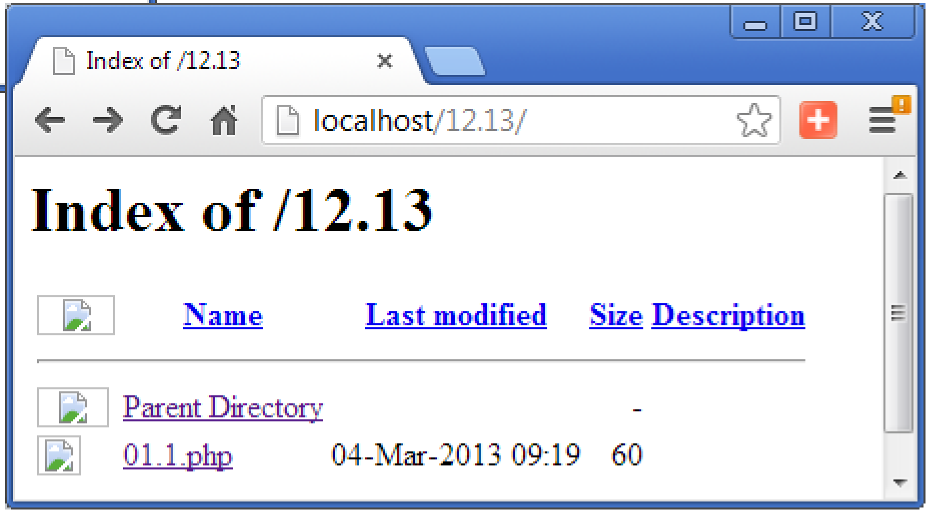
Your Projects

- 12.13

Open the page


localhost/12.13/01.1.php

Bonjour, voilà un script **simple** !


Index of /12.13

localhost/12.13/

Index of /12.13

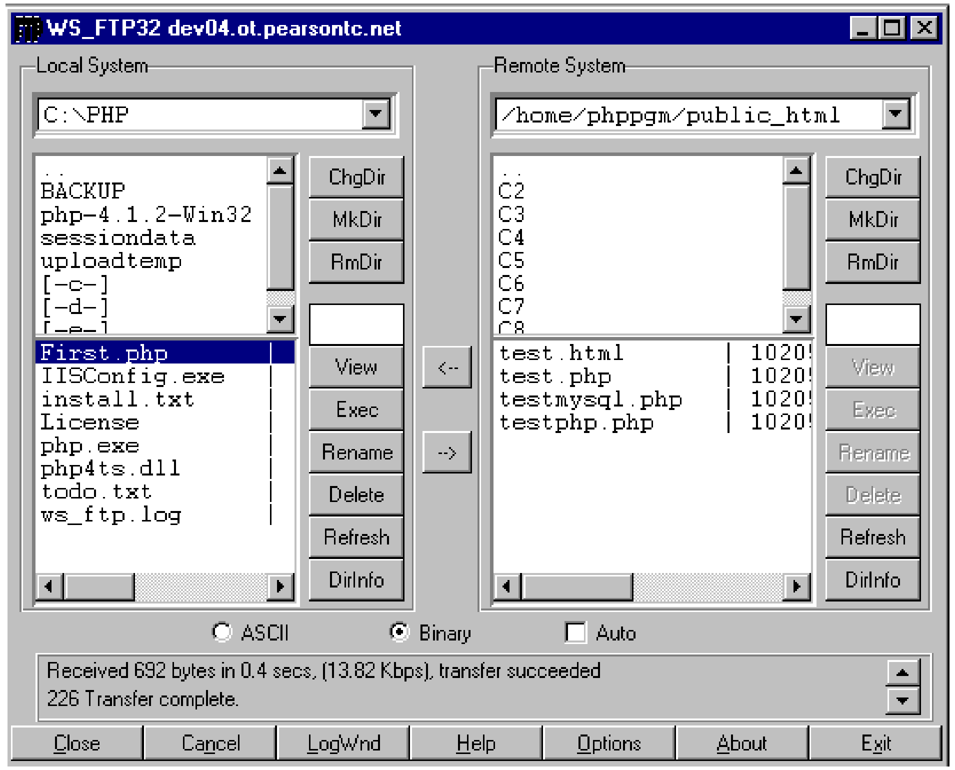
Name	Last modified	Size	Description
Parent Directory		-	
01.1.php	04-Mar-2013 09:19	60	

Copy files to the Web server using FTP

1. Connect to the Internet and start FTP

2. Connect to your server using FTP

3. Copy files to the Web server



WS_FTP32 dev04.ot.pearsontc.net

Local System

C:\PHP

```
..
BACKUP
php-4.1.2-Win32
sessiondata
uploadtemp
[-c-]
[-d-]
[-e-]
```

```
First.php
IISConfig.exe
install.txt
License
php.exe
php4ts.dll
todo.txt
ws_ftp.log
```

Remote System

/home/phppgm/public_html

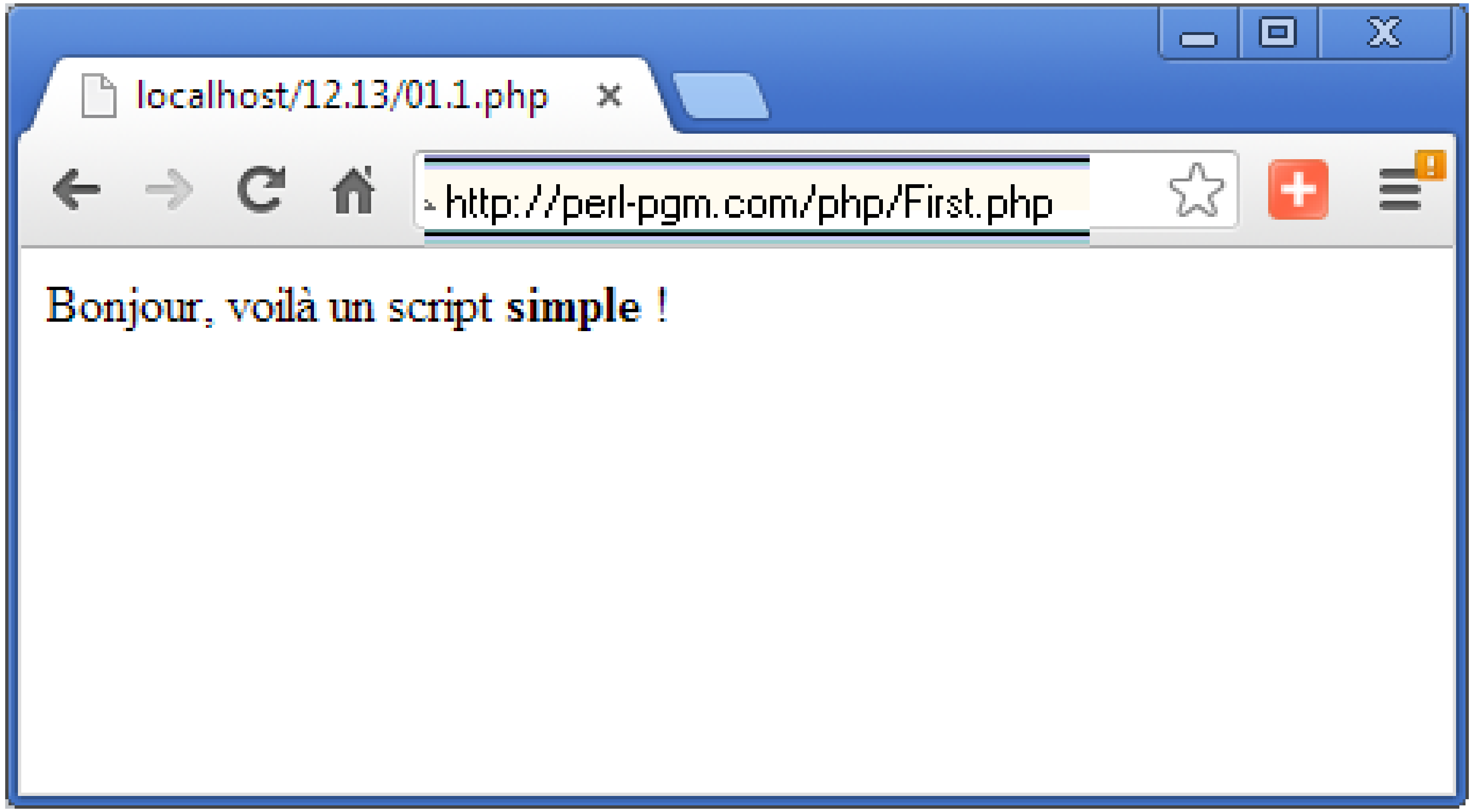
```
..
C2
C3
C4
C5
C6
C7
C8
```

```
test.html       | 1020
test.php        | 1020
testmysql.php   | 1020
testphp.php     | 1020
```

( ) ASCII (●) Binary [ ] Auto

```
Received 692 bytes in 0.4 secs, (13.82 Kbps), transfer succeeded
226 Transfer complete.
```

Access to your files using a web browser



http://perl-pgm.com/php/First.php

Bonjour, voilà un script **simple** !