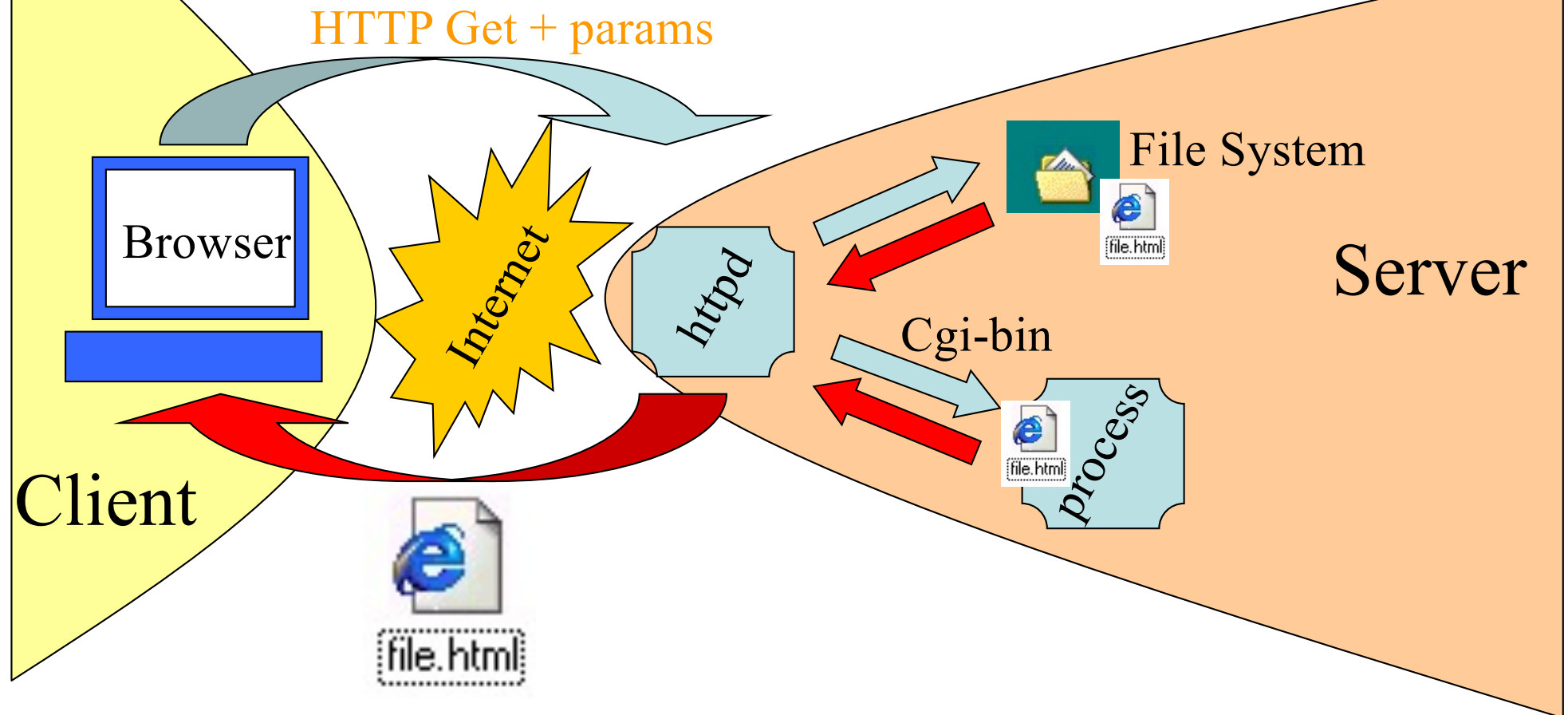


Q

**What are dynamic pages and
how do we manage them ?**

The original web architecture: dynamic pages

The web programmer also writes programs, using the programming language of her/his choice.



Evolution 1: dynamically create (interlinked) documents

STEP 1:
LET'S INSTALL A (APACHE)
WEB SERVER

Let's install a Web Server

We will, step by step:

1. install Apache (with a series of extra tools: DB and languages – Perl and PHP)
2. Customize it (e.g. by changing port)
3. Customize its response (static files)
4. Attack the dynamic content problem

XAMPP

<https://www.apachefriends.org/download.html>

What does XAMPP means? Generalization of WAMP, LAMP, MAMP

W = Windows, L=Linux, M=Mac, X=Anything

A = Apache Web Server

M = MySQL , MariaDB

P = PHP

P = Perl

XAMPP download

<https://www.apachefriends.org/download.html>

[Apache Friends](#)

[Download](#)


[Add-ons](#)

[Hosting](#)

[Community](#)

[About](#)

[Search](#)

 [EN](#)

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

 XAMPP for **Windows** 7.2.27, 7.3.14 & 7.4.2

Version		Checksum		Size
7.2.27 / PHP 7.2.27	What's Included?	md5	sha1	Download (64 bit) 147 Mb
7.3.14 / PHP 7.3.14	What's Included?	md5	sha1	Download (64 bit) 147 Mb
7.4.2 / PHP 7.4.2	What's Included?	md5	sha1	Download (64 bit) 148 Mb

[Requirements](#) [Add-ons](#) [More Downloads »](#)

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#).

Documentation/FAQs

There is no real manual or handbook for XAMPP. We wrote the documentation in the form of FAQs. Have a burning question that's not answered here? Try the [Forums](#) or [Stack Overflow](#).

- [Linux FAQs](#)
- [Windows FAQs](#)
- [OS X FAQs](#)
- [OS X XAMPP-VM FAQs](#)

Add-ons and Themes



Bitnami provides a free all-in-one tool to install Drupal, Joomla!, WordPress and many other popular open source

Follow the documentation



XAMPP for **Linux** 7.2.27, 7.3.14 & 7.4.2

Version		Checksum			Size
7.2.27 / PHP 7.2.27	What's Included?	md5	sha1	Download (64 bit)	148 Mb
7.3.14 / PHP 7.3.14	What's Included?	md5	sha1	Download (64 bit)	149 Mb
7.4.2 / PHP 7.4.2	What's Included?	md5	sha1	Download (64 bit)	151 Mb

[Requirements](#) [Add-ons](#) [More Downloads »](#)



XAMPP for **OS X** 7.2.27, 7.3.14, 7.4.2, 7.2.27, 7.3.14 & 7.4.2

Version		Checksum			Size
7.2.27 / PHP 7.2.27	What's Included?	md5	sha1	Download (64 bit)	159 Mb
7.3.14 / PHP 7.3.14	What's Included?	md5	sha1	Download (64 bit)	159 Mb
7.4.2 / PHP 7.4.2	What's Included?	md5	sha1	Download (64 bit)	160 Mb
7.2.27 / PHP 7.2.27	What's Included?	md5	sha1	Download (64 bit)	322 Mb

VM Installation (e.g. on Mac)...

LAMPP is installed on a virtual machine.

We need to:

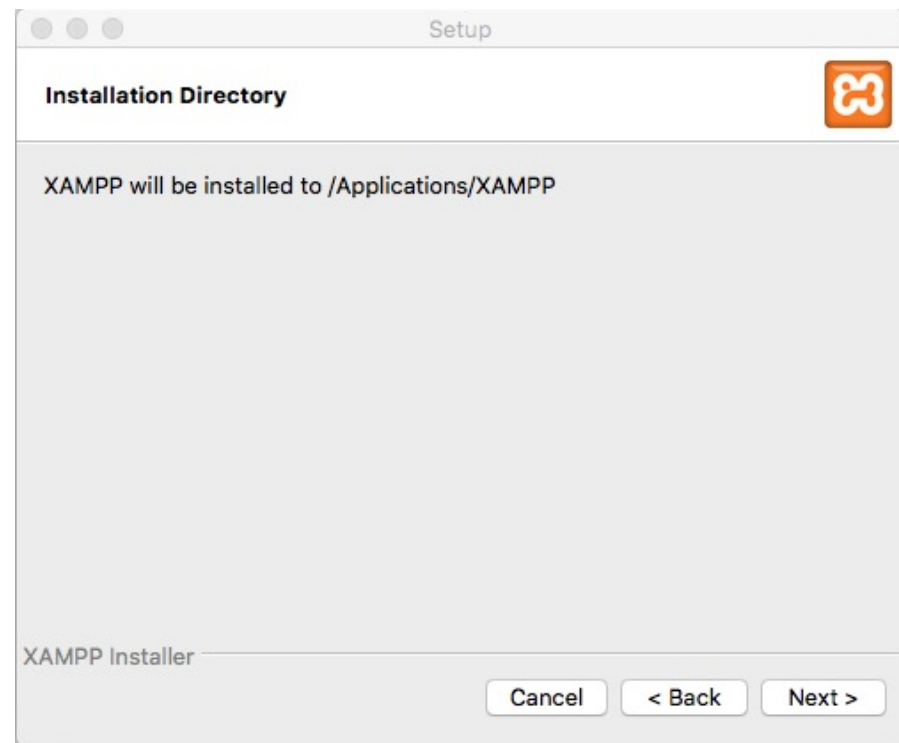
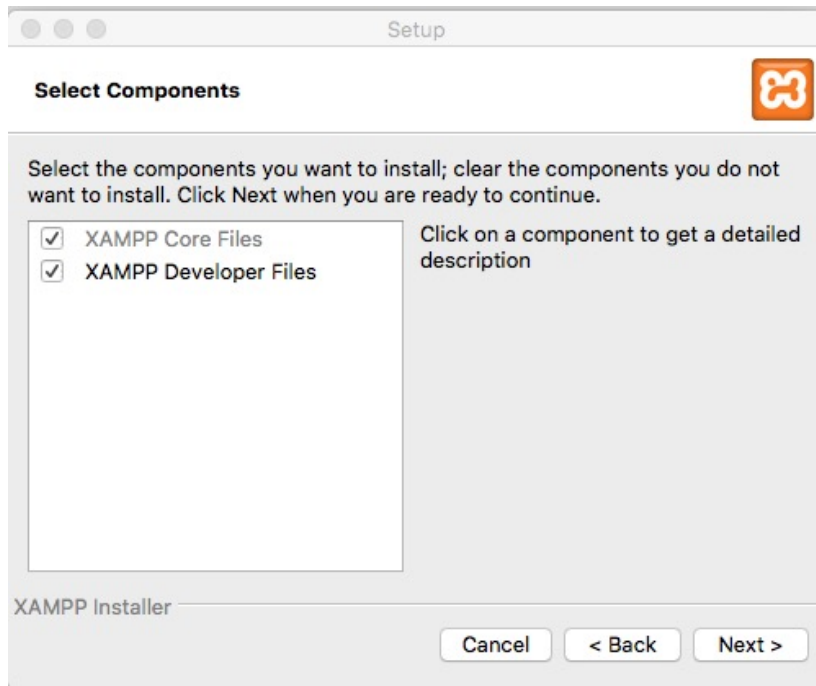
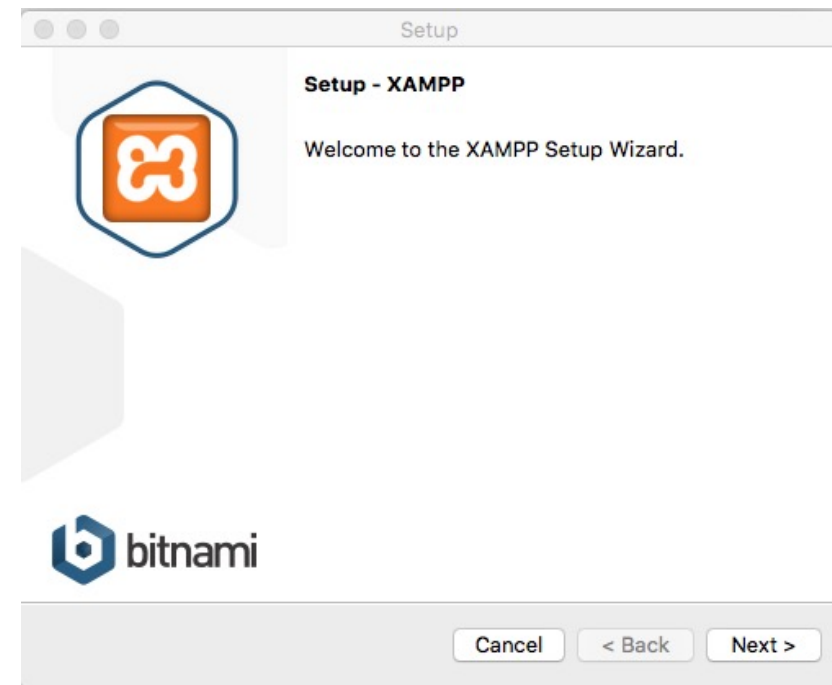
- Connect the port on our machine (e.g. 8080) to the 80 port on the VM
- Access the VM file system from our file system

Mac users: do not use the VM version, but rather use this:

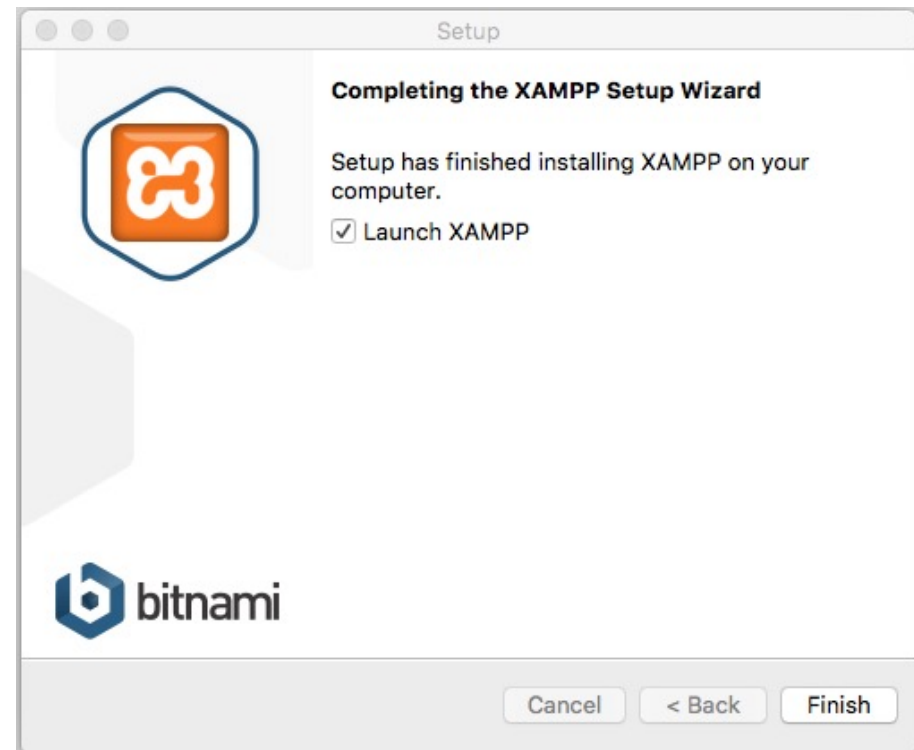
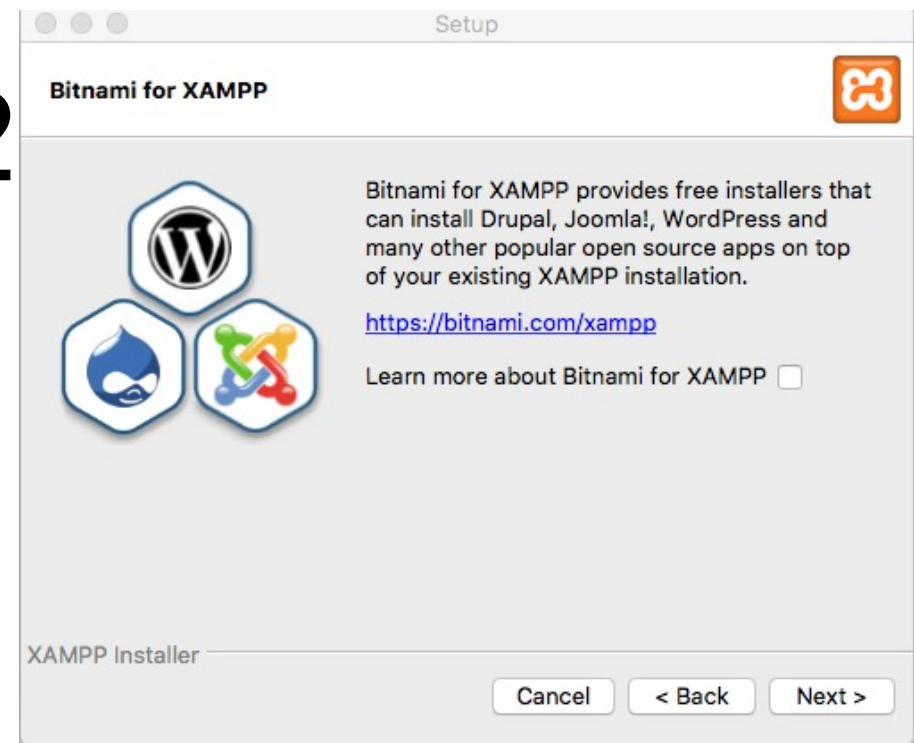
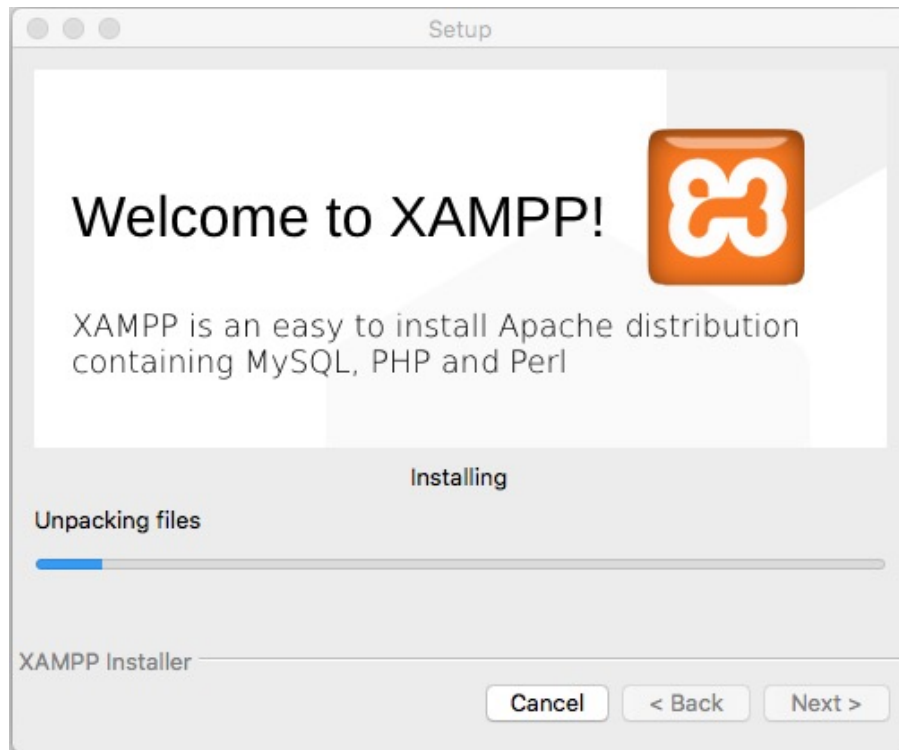
<https://sourceforge.net/projects/xampp/files/XAMPP%20Mac%20OS%20X/7.4.2/>

(Current version: 7.4.23)

First run: setup - 1



First run: setup - 2



First run:



Welcome to XAMPP for 7.4.2-0

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the [FAQs](#) section or check the [HOW-TO Guides](#) for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the [FAQs](#) to learn how to protect your site. Alternatively you can use [WAMP](#), [MAMP](#) or [LAMP](#) which are similar packages which are more suitable for production.

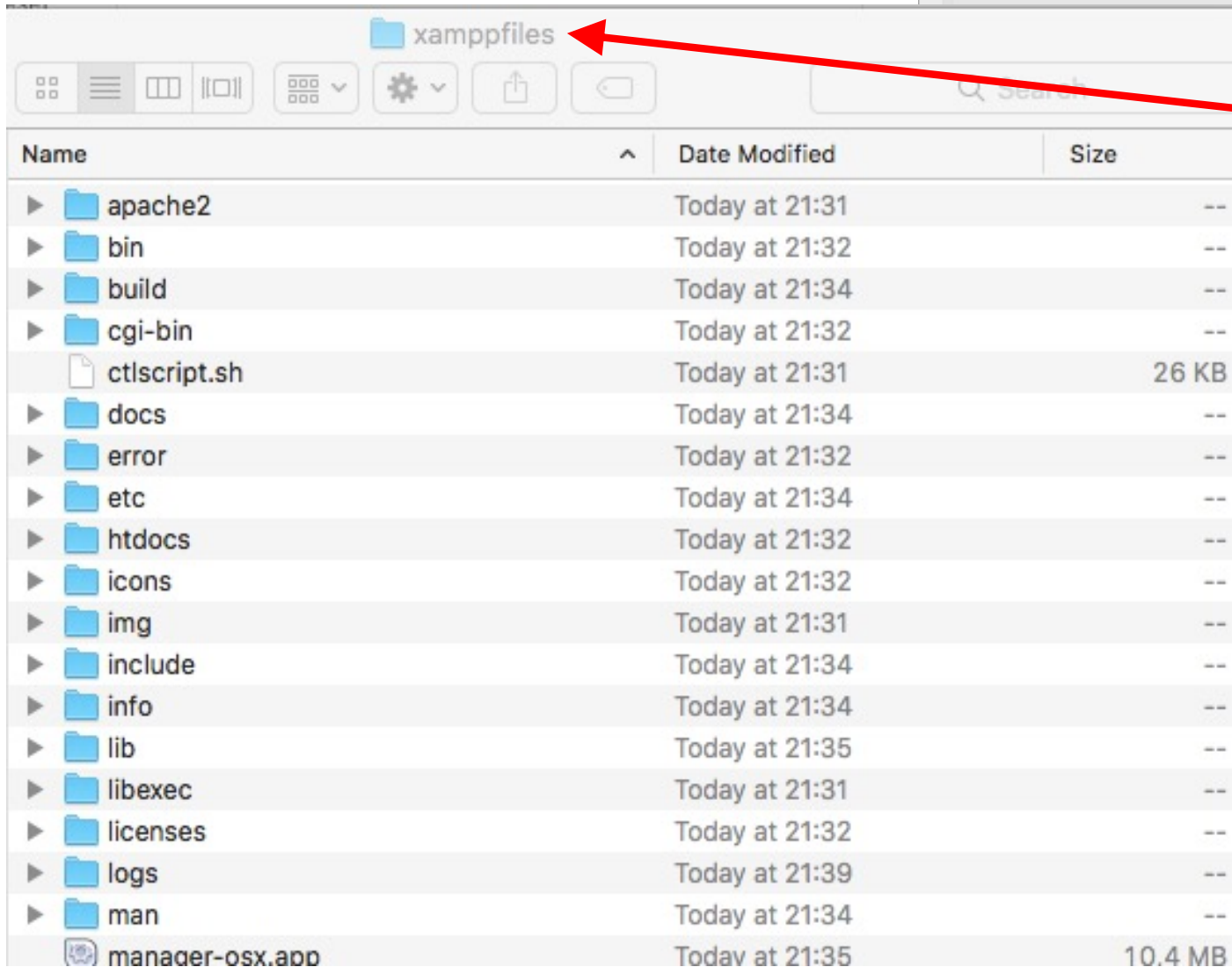
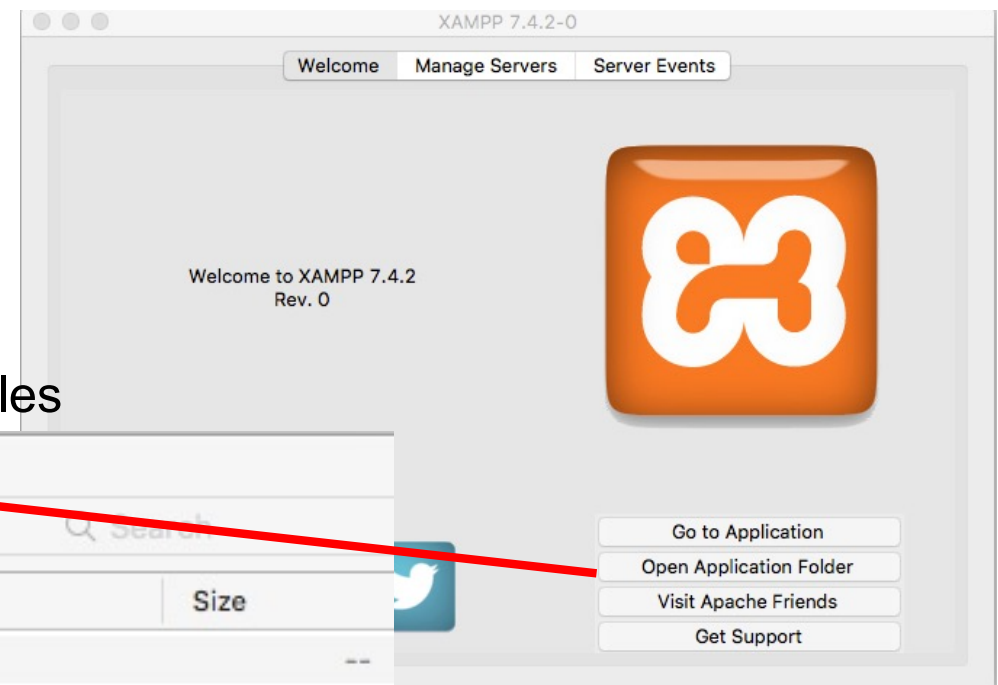
Start the XAMPP Control Panel to check the server status.

Community

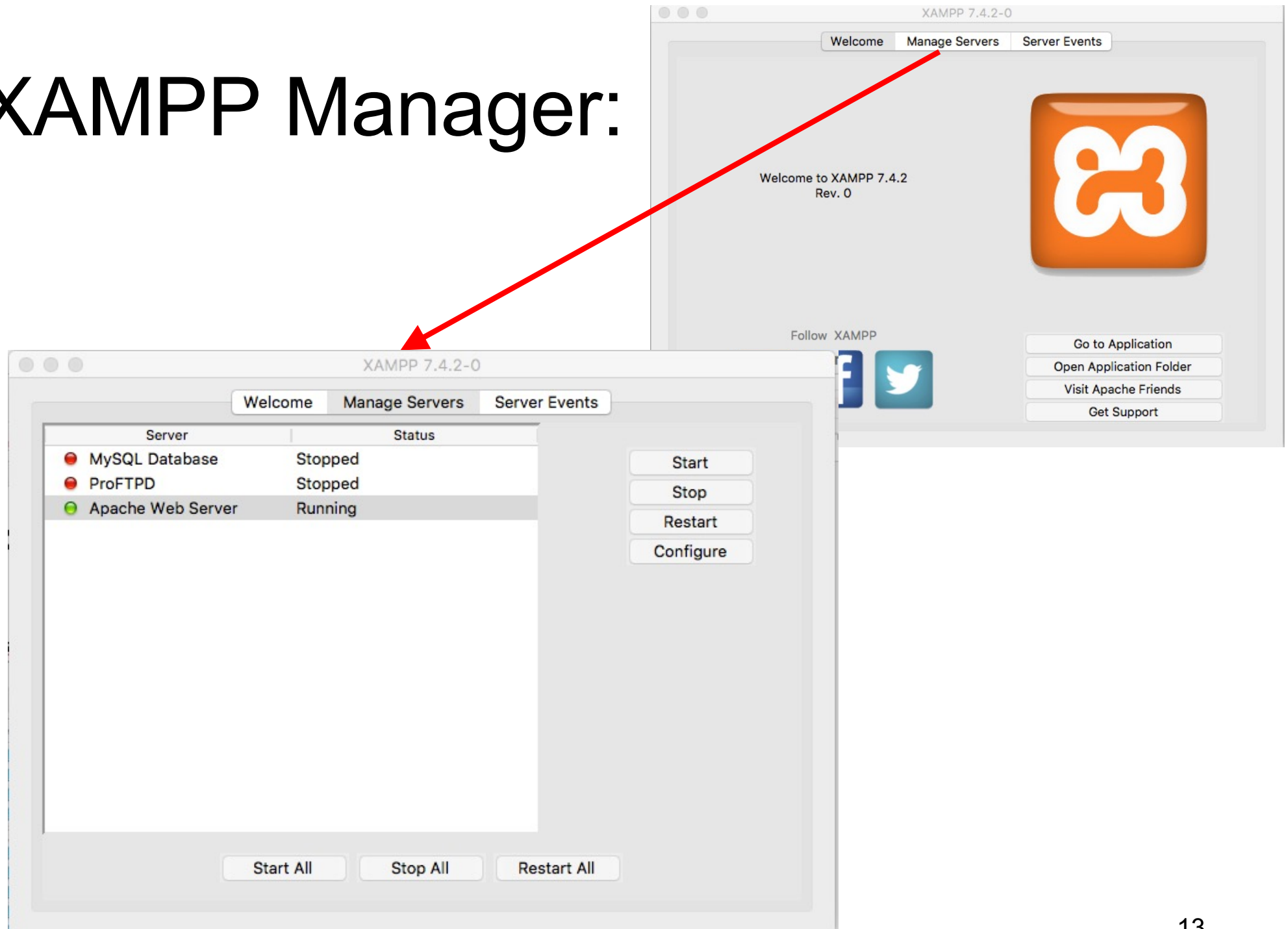
XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our [Forums](#).

XAMPP Manager:

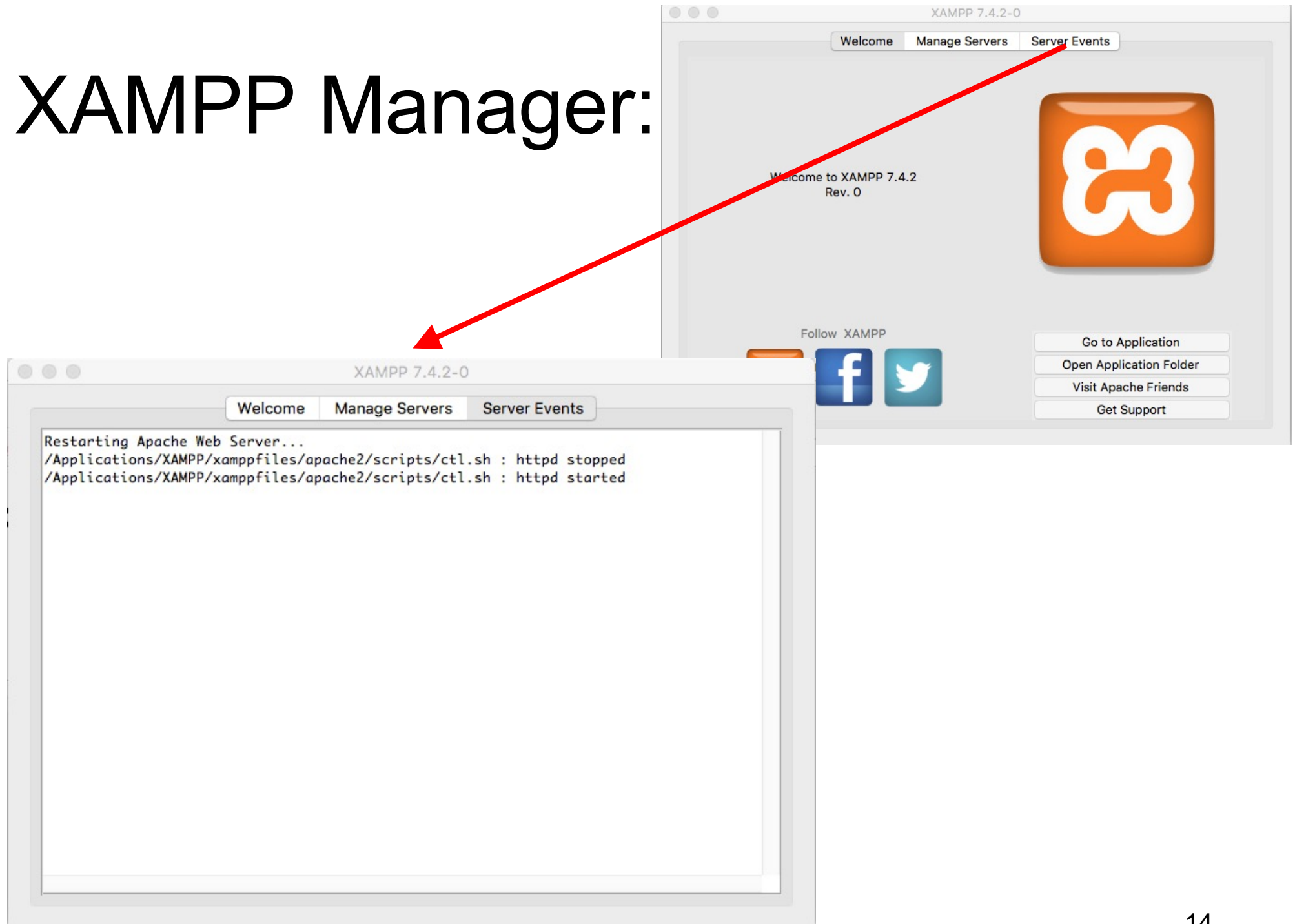
/MacintoshHD/Applications/XAMPP/xamppfiles



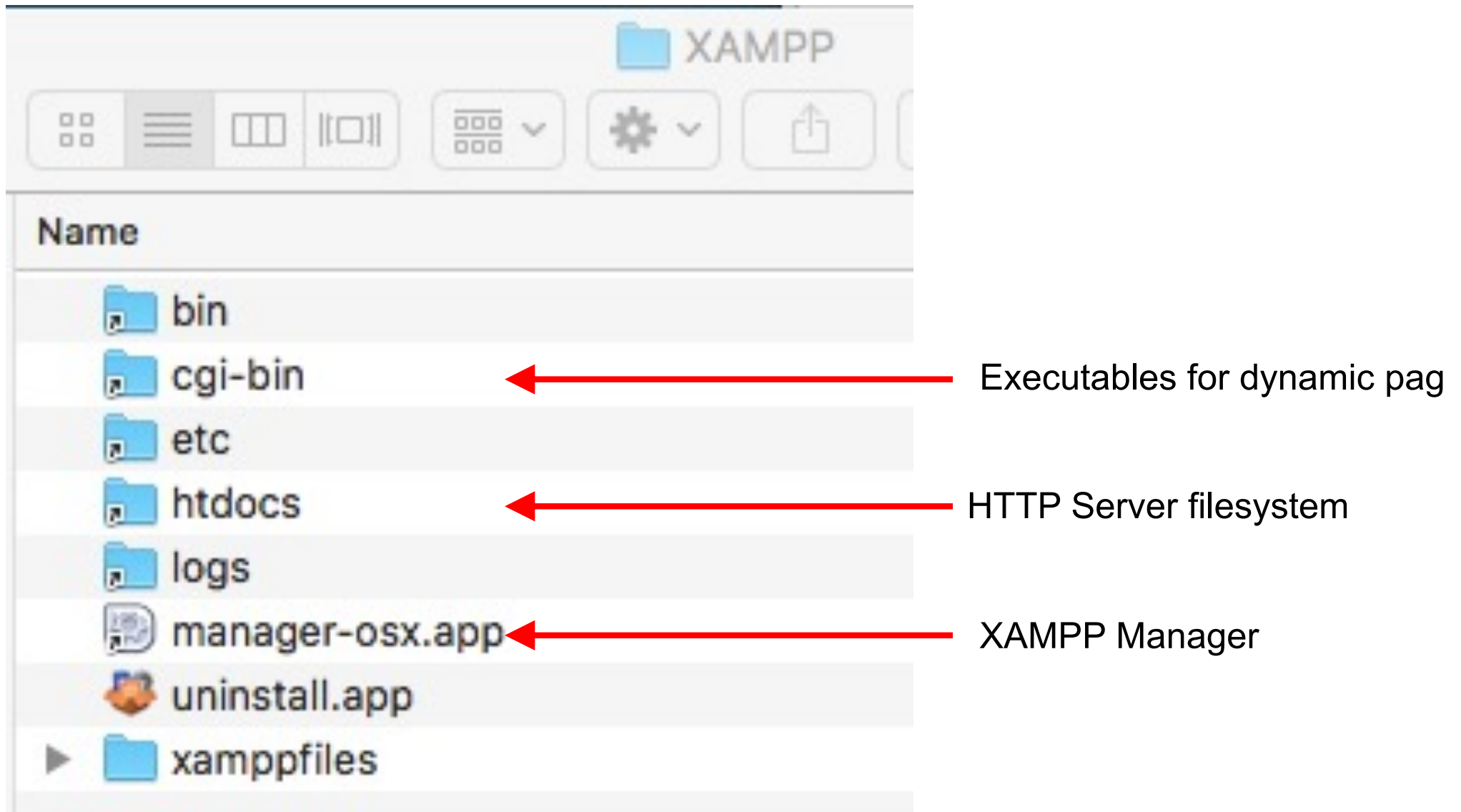
XAMPP Manager:



XAMPP Manager:



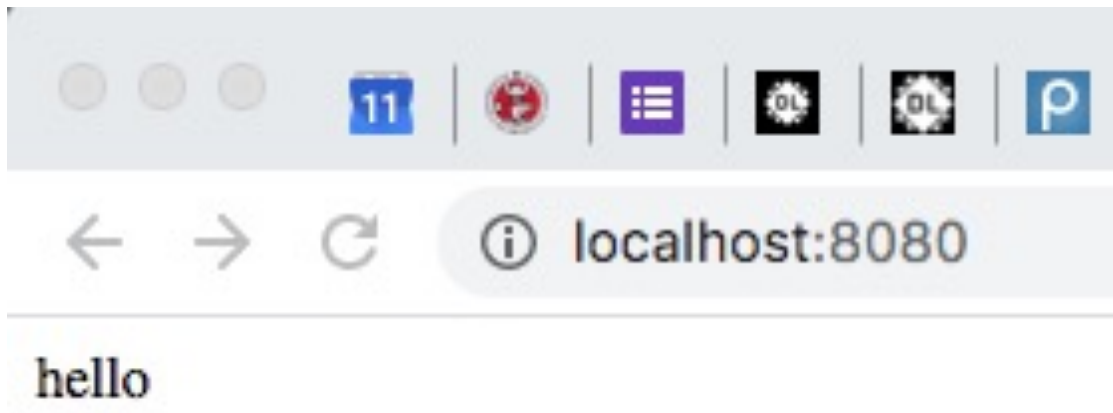
Main files e directories



Modifying server content

```
MR-MBP-14955:local ronchet$ cd /Applications/XAMPP/xamppfiles/htdocs
MR-MBP-14955:htdocs ronchet$ touch index.html
MR-MBP-14955:htdocs ronchet$ vi index.html
MR-MBP-14955:htdocs ronchet$ cat index.html
hello
```

Create empty fi
Edit file
Show content



APACHE DEFAULTS ARE: index.html, index.php

Apache configuration

See

http://www.cellbiol.com/bioinformatics_web_development/chapter-2-the-linux-operating-system-setting-up-a-linux-web-server/apache-web-server-configuration/

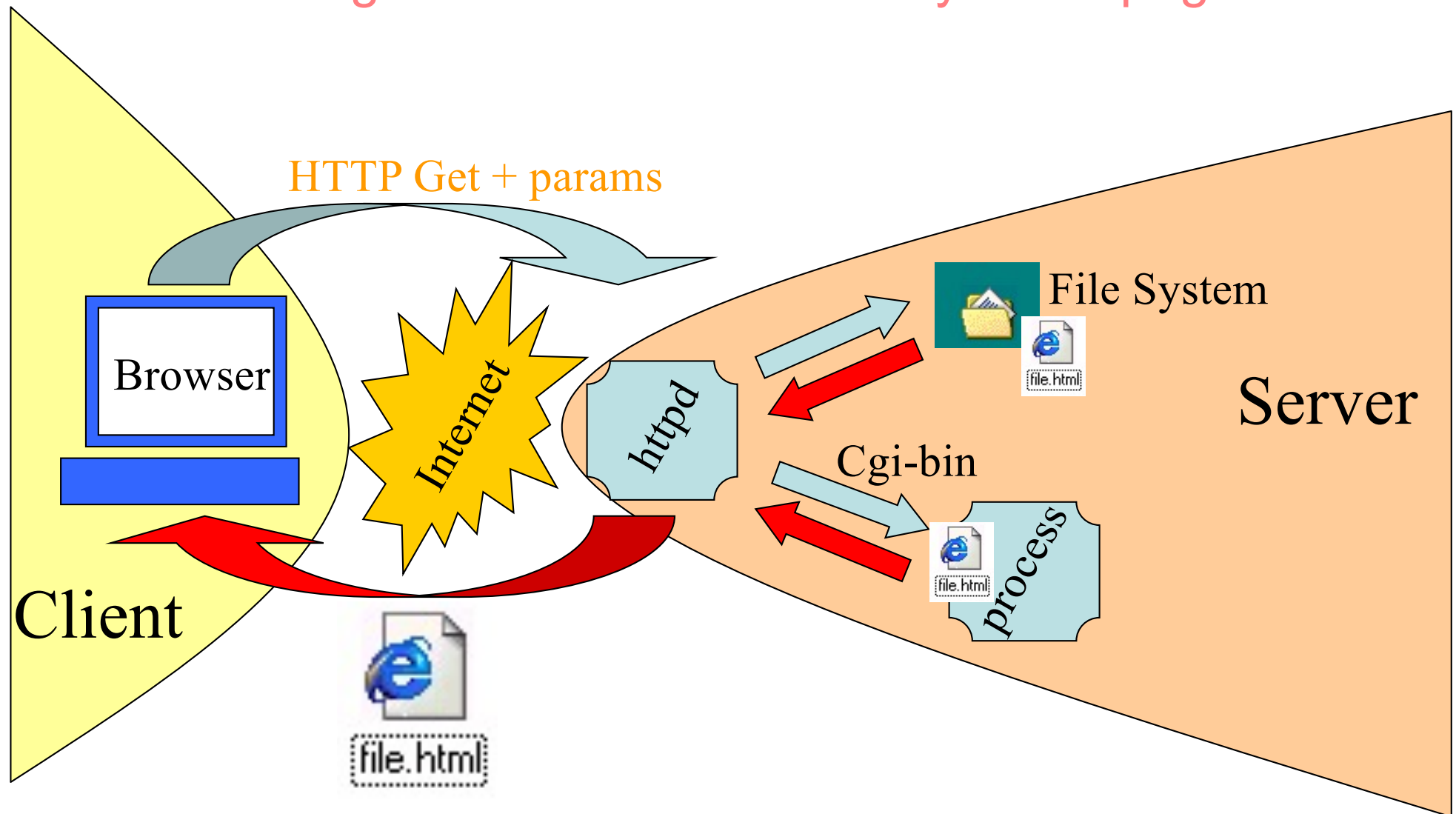
(has to be adapted to your locations)

STEP 2:
LET'S USE OUR WEB SERVER
TO GENERATE DYNAMIC PAGES

Dynamic pages: the main idea:
We want to obtain NON STATIC
information from the server. This implies
executing some code on it, and send the
results to the user.

e.g.: what's the time?

The original web architecture: dynamic pages



Evolution 1: dynamically create (interlinked) documents

The first implementation: CGI

The Common Gateway Interface was (is) a way to tell the server to spawn a process, get its results and send them as HTTP response.

Reading:

- <https://computer.howstuffworks.com/cgi.htm>
- https://en.wikipedia.org/wiki/Common_Gateway_Interface

Follower: FastCGI

- <https://en.wikipedia.org/wiki/FastCGI>

Creating dynamic pages

```
MR-MBP-14955:local ronchet$ cd /Applications/XAMPP/xamppfiles/cgi-bin
MR-MBP-14955:htdocs ronchet$ touch getTime.sh
MR-MBP-14955:htdocs ronchet$ vi getTime.sh
MR-MBP-14955:htdocs ronchet$ cat getTime.sh
#!/bin/sh
echo `date`
MR-MBP-14955:cgi-bin ronchet$ ls -la getTime.sh
-rw-r--r-- 1 ronchet admin 22 Feb 11 22:26 getTime.sh
MR-MBP-14955:cgi-bin ronchet$ chmod 755 getTime.sh
MR-MBP-14955:cgi-bin ronchet$ ls -la getTime.sh
-rwxr-xr-x 1 ronchet admin 22 Feb 11 22:26 getTime.sh
MR-MBP-14955:cgi-bin ronchet$ ./getTime.sh
Tue Feb 11 22:28:56 CET 2020
```

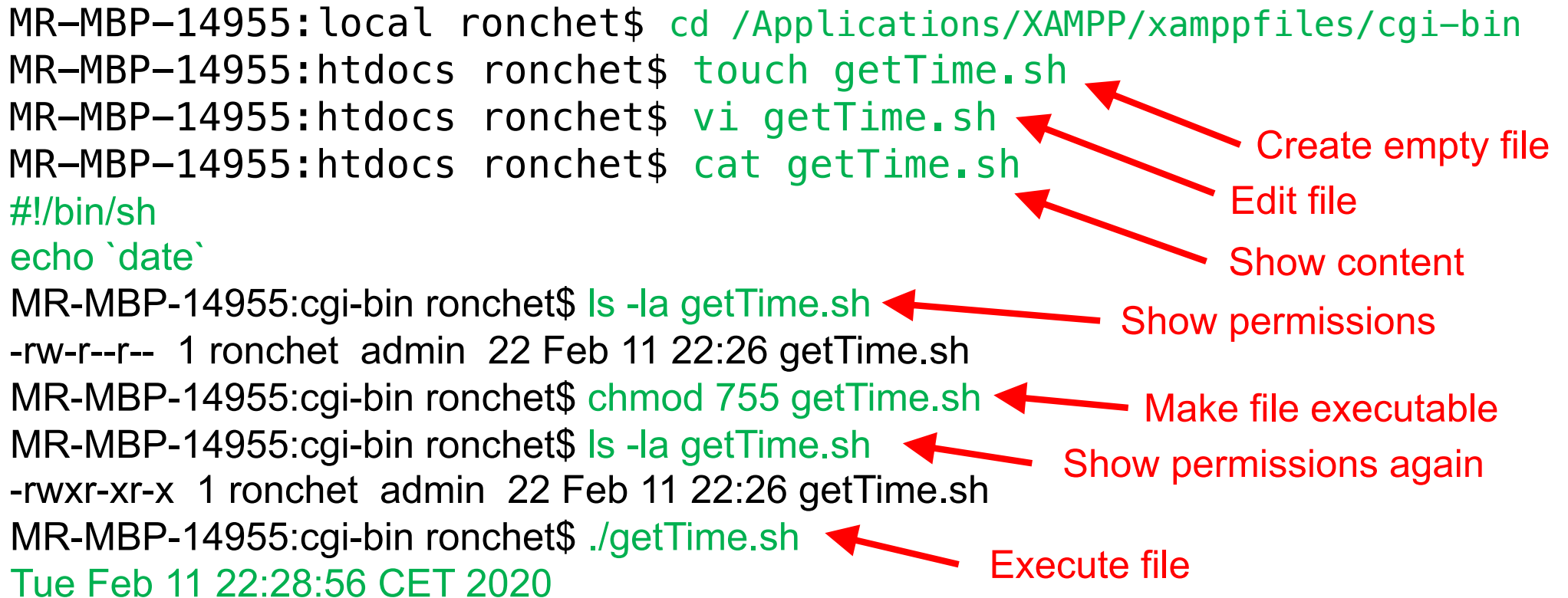


Diagram illustrating the steps to create and execute a dynamic page:

- Create empty file (points to `touch getTime.sh`)
- Edit file (points to `vi getTime.sh`)
- Show content (points to `cat getTime.sh`)
- Show permissions (points to `ls -la getTime.sh`)
- Make file executable (points to `chmod 755 getTime.sh`)
- Show permissions again (points to `ls -la getTime.sh`)
- Execute file (points to `./getTime.sh`)

← → ↻ ⓘ localhost:8080/cgi-bin/getTime.sh

Server error!

The server encountered an internal error and was unable to complete your request.

Error message:

Premature end of script headers: getTime.sh

If you think this is a server error, please contact the [webmaster](#).

Error 500

localhost

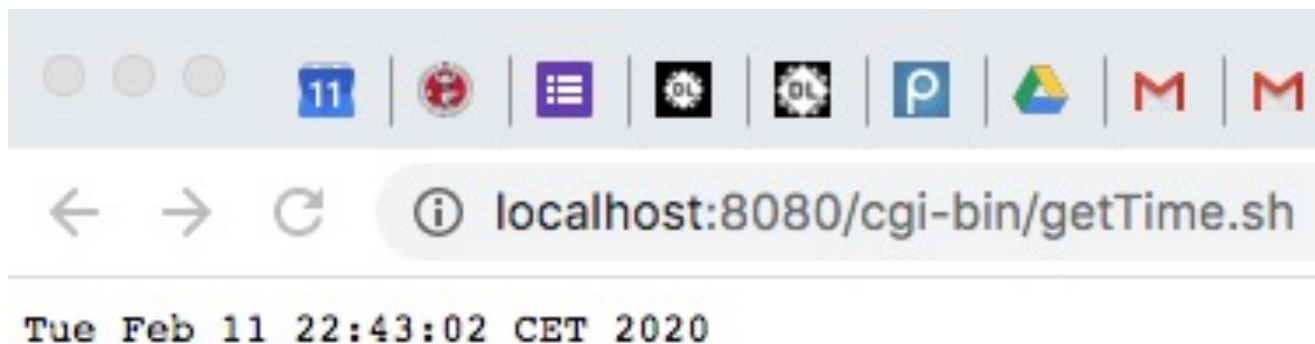
Apache/2.4.41 (Unix) OpenSSL/1.1.1d PHP/7.4.2 mod_perl/2.0.8-dev Perl/v5.16.3

Creating dynamic pages

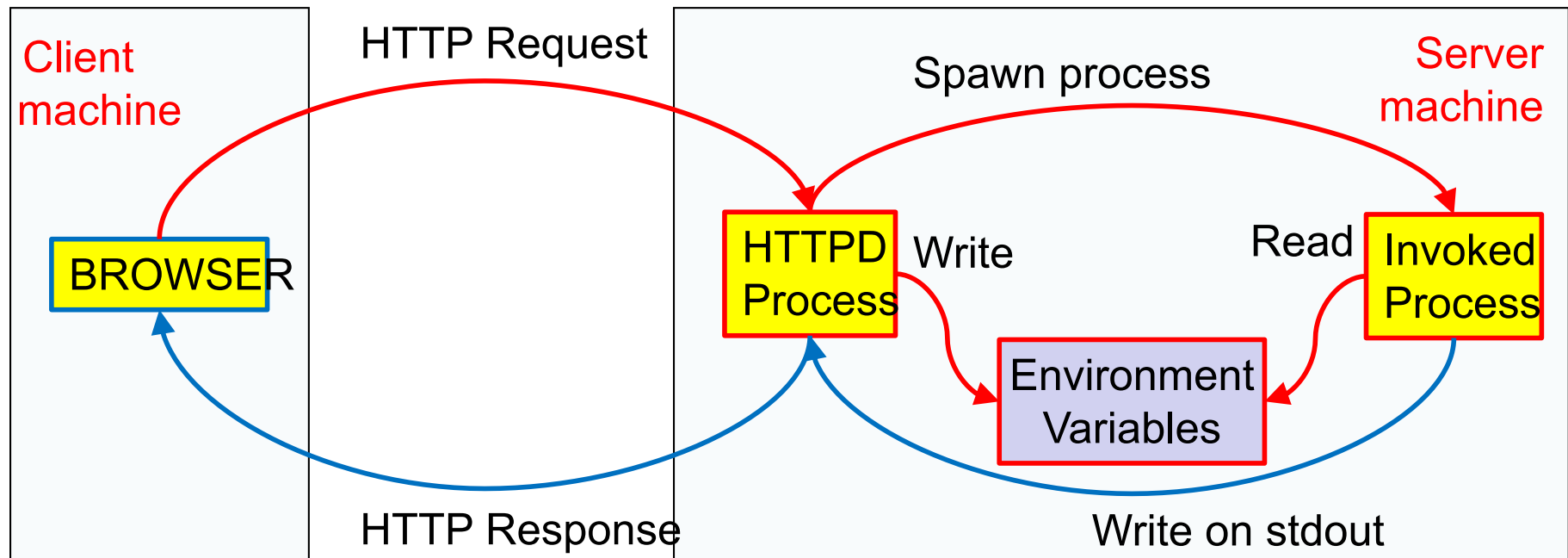
```
MR-MBP-14955:local ronchet$ cd /Applications/XAMPP/xamppfiles/cgi-bin
MR-MBP-14955:htdocs ronchet$ vi getTime.sh
MR-MBP-14955:htdocs ronchet$ cat getTime.sh
#!/bin/sh
echo "Content-type: text/plain; charset=iso-8859-1"
echo
echo `date`
```

Edit file

Show content



Getting info about the request



Access to environment vars is supported in all programming languages!

- **Server specific variables:**

- `SERVER_SOFTWARE` : *name/version* of HTTP server.
- `SERVER_NAME` : *host name* of the server, may be *dot-decimal* IP address.
- `GATEWAY_INTERFACE` : *CGI/version*.

Getting info about the request

- Request specific variables:

- `SERVER_PROTOCOL` : HTTP/*version*.
- `SERVER_PORT` : TCP port (decimal).
- `REQUEST_METHOD` : name of HTTP method (see above).
- `PATH_INFO` : path suffix, if appended to URL after program name and a slash.
- `PATH_TRANSLATED` : corresponding full path as supposed by server, if `PATH_INFO` is present.
- `SCRIPT_NAME` : relative path to the program, like `/cgi-bin/script.cgi`.
- `QUERY_STRING` : the part of URL after `?` character. The query string may be composed of **name=value* pairs separated with ampersands (such as `var1=val1&var2=val2...`) when used to submit form data transferred via GET method as defined by HTML application/x-www-form-urlencoded.
- `REMOTE_HOST` : host name of the client, unset if server did not perform such lookup.
- `REMOTE_ADDR` : IP address of the client (dot-decimal).
- `AUTH_TYPE` : identification type, if applicable.
- `REMOTE_USER` used for certain `AUTH_TYPE` s.
- `REMOTE_IDENT` : see `ident`, only if server performed such lookup.
- `CONTENT_TYPE` : Internet media type of input data if PUT or POST method are used, as provided via HTTP header.
- `CONTENT_LENGTH` : similarly, size of input data (decimal, in octets) if provided via HTTP header.
- Variables passed by user agent (`HTTP_ACCEPT` , `HTTP_ACCEPT_LANGUAGE` , `HTTP_USER_AGENT` , `HTTP_COOKIE` and possibly others) contain values of corresponding HTTP headers and therefore have the same sense.

Testing it...

You need to modify test-cgi as described in the file itself



The screenshot shows a web browser window with a toolbar at the top containing various icons like a calendar, email, and social media. The address bar shows the URL `localhost:8080/cgi-bin/test-cgi`. The main content area displays the output of a CGI script, which is a text-based report. Two red arrows point to the `REQUEST_METHOD = GET` and `QUERY_STRING =` lines in the report.

```
CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = Apache/2.4.41 (Unix) OpenSSL/1.1.1d PHP/7.4.2 mod_perl/2.0.8-dev Perl/v5.16.3
SERVER_NAME = localhost
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.1
SERVER_PORT = 8080
REQUEST_METHOD = GET
HTTP_ACCEPT = text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING =
REMOTE_HOST =
REMOTE_ADDR = ::1
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =
```

HTTP, HTTPS and TCP Networking in Java

Some reminders

Q

**Reminder: How is I/O managed
in Java?**

I/O in Java

	Byte Based		Character Based	
	<i>Input</i>	<i>Output</i>	<i>Input</i>	<i>Output</i>
Basic	InputStream	OutputStream	Reader	Writer
			InputStreamReader	OutputStreamWriter
Arrays	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
Files	FileInputStream	FileOutputStream	FileReader	FileWriter
	RandomAccessFile	RandomAccessFile		
Pipes	PipedInputStream	PipedOutputStream	PipedReader	PipedWriter
Buffering	BufferedInputStream	BufferedOutputStream	BufferedReader	BufferedWriter
Filtering	FilterInputStream	FilterOutputStream	FilterReader	FilterWriter
Parsing	PushbackInputStream		PushbackReader	
	StreamTokenizer		LineNumberReader	
Strings			StringReader	StringWriter
Data	DataInputStream	DataOutputStream		
Data - Formatted		PrintStream		PrintWriter
Objects	ObjectInputStream	ObjectOutputStream		
Utilities	SequenceInputStream			

There is also nio

Q

Reminder: What is a socket?

Sockets

The `java.net.Socket` class represents a side of connection (regardless if client or server).

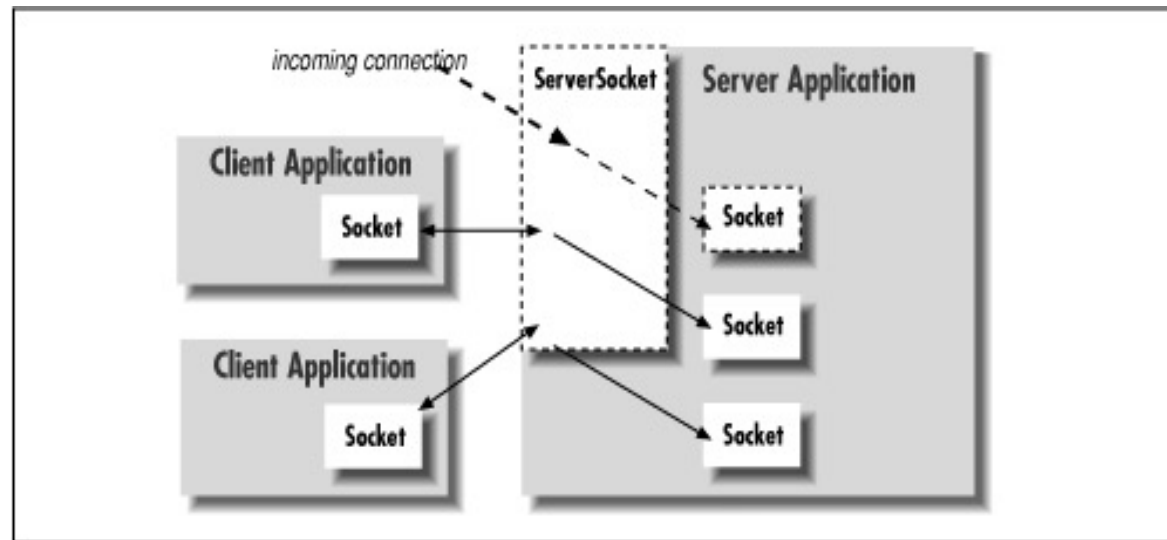
The server uses the `java.net.ServerSocket` class to wait for incoming conversations. It creates a `ServerSocket` object and waits, blocked on a `accept()` call until a connection comes. Then it creates a `Socket` object to be used to communicate with the client.

see <https://docs.oracle.com/javase/tutorial/networking/index.html>

Sockets

A server can maintain many conversations simultaneously.

There is only one ServerSocket, but one Socket for every client.



Server port

The client needs two pieces of info to establish a connection: a **hostname** (to get the server's address) and a **port number** (to identify a process on the server machine).

A server app listens on a predefined port while waiting for a connection.

Port numbers are coded in the RFC (Es. Telnet 23, FTP 21, ecc.), but they can be freely chosen for custom services.

Client port

The client's port number is generally assigned by the OS, and in general you do not care about it.

When the server responds it opens a new socket whose number is assigned by the OS. It then continues listening on the original port, and serves the particular clients on the new socket.

Sockets

The first choice is which protocol to use:

connection-oriented (TCP)

or

connectionless (UDP).

The Java Socket class uses TCP

java.net.Socket

This class implements a socket for interprocess communication over the network.

The constructors create the socket and connect it to the specified host on the specified port.

java.net.Socket - main methods

.

Once the socket is created, `getInputStream()` e `getOutputStream()` return `InputStream` e `OutputStream` objects (usable as I/O channels).

`getInetAddress()` e `getPort()` return address and port to which the socket is connected.

`close()` closes la socket.

java.net.ServerSocket

During creation you specify on which port to listen

The **accept()** starts listening and **blocks** until there is an incoming call.

At that point, **accept()** accepts the connection, creates and returns a **Socket** that the server can use to talk to the client.

E

Example: simple conversation on a socket.

(see the IntelliJ project on the web site)

Reading & Writing newline delimited strings – Server

```
public class SocketServerDemo {
    public static void main(String a[]){
        try {
            ServerSocket listener = new ServerSocket( 1234 );
            while ( true ) {
                Socket aClient = listener.accept(); // wait for connection
                // set up I/O
                InputStream in = aClient.getInputStream();
                InputStreamReader din = new InputStreamReader(in);
                BufferedReader bin = new BufferedReader(din);
                OutputStream out = aClient.getOutputStream();
                PrintStream pout = new PrintStream( out );
                // Read a string
                String request = bin.readLine();
                System.out.println("Got from the client "+request);

                // Write a string
                String greeting="Goodbye! you were on port "+aClient.getPort();
                pout.println(greeting);
                System.out.println("Said to the client "+greeting);

                aClient.close();
            }
            listener.close();
        } catch (IOException e ) {//... }
    }
}
```

Reading & Writing newline delimited strings – Server

```
public class SocketClientDemo {
    public static void main(String a[]) {
        try {
            Socket server = new Socket("localhost", 1234); // open connection
            // set up I/O
            InputStream in = server.getInputStream();
            InputStreamReader din = new InputStreamReader(in);
            BufferedReader bin = new BufferedReader(din);
            OutputStream out = server.getOutputStream();
            PrintStream pout = new PrintStream(out);

            // send newline delimited string
            String greeting="Hello";
            pout.println(greeting);
            System.out.println("Said to the server "+greeting);

            // Read a newline delimited string
            String response = bin.readLine();
            System.out.println("Got from the server "+response);

        } catch (IOException e) {
            e.printStackTrace();//...;
        }
    }
}
```

Q

How can we write a simple HTTP server?

(see the IntelliJ project on the web site)

A concurrent HTTP mini-server - Introduction

TinyHttpd listens on a specified port and services simple HTTP "get file" requests. They look something like this:

GET */path/filename [optional stuff]*


Your Web browser sends one or more as lines for each document it retrieves. Upon reading the request, the server tries to open the specified file and send its contents. If that document contains references to images or other items to be displayed inline, the browser continues with additional GET requests. For best performance (especially in a time-slicing environment), TinyHttpd services each request in its own thread. Therefore, TinyHttpd can service several requests concurrently.

A concurrent mini HTTP daemon

```
public class TinyHttpd {  
    public static void main( String argv[] )  
        throws IOException {  
        int port = 8888;  
        if (argv.length>0) port=Integer.parseInt(argv[0]);  
        ServerSocket ss = new ServerSocket(port);  
        System.out.println("Server is ready");  
        while ( true )  
            new TinyHttpdConnection(ss.accept());  
    }  
}
```

A concurrent mini HTTP daemon

```
class TinyHttpdConnection extends Thread {  
    Socket sock;  
  
    TinyHttpdConnection(Socket s) {  
        sock = s;  
        setPriority(NORM_PRIORITY - 1);  
        start();  
    }  
}
```



By lowering its priority to NORM_PRIORITY-1 (just below the default priority), we ensure that the threads servicing established connections won't block TinyHttpd's main thread from accepting new requests.

starts a new thread, and executes the method "run" in it.

The method "run" MUST be implemented by every subclass of Thread

A concurrent mini HTTP daemon

Set up I/O

```
public void run() {  
    System.out.println("=====");  
    System.out.println("Connected on port "+sock.getPort());  
    OutputStream out = null;  
    try {  
        // OPEN SOCKETS FOR READING AND WRITING  
        BufferedReader d =  
            new BufferedReader(new InputStreamReader(  
                sock.getInputStream()));  
        out = sock.getOutputStream();  
        PrintStream ps=new PrintStream(out);  
    }  
}
```

A concurrent mini HTTP daemon

Read and deal with request

```
// READ REQUEST
String req = d.readLine(); // first line contains request
if (req==null) return;
System.out.println("Request: " + req);
// READ REQUEST HEADERS
String head=null;
do {                                // following lines contain headers
    head = d.readLine();
    System.out.println("Header: " + head);
} while (head!=null && head.length()>0);
// PARSE REQUEST
StringTokenizer st = new StringTokenizer(req);
if ((st.countTokens() >= 2) && st.nextToken().equals("GET")) {
    if ((req = st.nextToken()).startsWith("/")) {
        req = req.substring(1);
    }
    if (req.endsWith("/") || req.equals("")) {
        req = req + "index.html";
    }
}
```


A concurrent mini HTTP daemon

Prepare and send the response

```
// OPEN REQUESTED FILE AND COPY IT TO CLIENT
    try {
//All our requested files must be in the "Documents" directory
        FileInputStream fis = new FileInputStream("Documents/"+req);
        int responseLength=fis.available();
        // LET'S SEND THE RESPONSE HEADERS
        ps.print("HTTP/1.1 200 OK\r\n");
        ps.print("Content-Length: "+responseLength+"\r\n");
        ps.print("Content-Type: text/html\r\n");
        ps.print("\r\n");
        // LET'S SEND THE CONTENT
        byte[] data = new byte[responseLength];
        fis.read(data);
        out.write(data);
        fis.close();
    } catch (FileNotFoundException e) {
        ps.print("HTTP/1.1 404 Not Found \r\n\r\n");
        System.out.println("404 Not Found: " + req);
    }
}
```

A concurrent mini HTTP daemon

Let's finish up

```
    } else {
        ps.print("HTTP/1.1 400 Bad Request\r\n\r\n");
        System.out.println("400 Bad Request: " + req);
    }
} catch (IOException e) { // Let's catch all generic I/O troubles
    System.out.println("Generic I/O error " + e);
} finally { // the following code is ALWAYS executed
    try {
        // let's close all channels
        out.close();
        sock.close();
    } catch (IOException ex) {
        System.out.println("I/O error on close" + ex);
    }
}
}
```

A concurrent mini HTTP daemon

```
} catch (IOException e) {  
    System.out.println("Generic I/O error " + e);  
} finally {  
    try {  
        out.close();  
    } catch (IOException ex) {  
        System.out.println("I/O error on close" + ex);  
    }  
}  
}  
}
```

A concurrent HTTP mini-server - usage

Compile TinyHttpd and place it in your class path. Go to a directory with some interesting documents and start the daemon, specifying an unused port number as an argument. For example:

```
% java TinyHttpd 1234
```

You should now be able to use your Web browser to retrieve files from your host. You'll have to specify the nonstandard port number in the URL. For example, if your hostname is foo.bar.com, and you started the server as above, you could reference a file as in:

```
http://foo.bar.com:1234/welcome.html
```

A concurrent HTTP mini-server - Problems

TinyHttpd still has room for improvement. First, it consumes a lot of memory by allocating a huge array to read the entire contents of the file all at once. A more realistic implementation would use a buffer and send large amounts of data in several passes.

A concurrent HTTP mini-server - Problems

TinyHttpd suffers from the limitations imposed by the fickleness of filesystem access.

It's important to remember that file pathnames are still architecture dependent--as is the concept of a filesystem to begin with. TinyHttpd works, as is, on UNIX systems. It needs adaptation for the Windows world.

E

Exercise: get ready for the first assignment:

- 1) Install Apache, follow the steps**
- 2) Install IntelliJ, open and run the distributed project**