



COMP4021 Lab 7

Server Setup & PHP

Nov. 8th, 2021

Outlines

- Installation of Server (Win10 & MacOS)
- PHP Basics
- Chatroom



Server Setup

with XAMPP

Installation

File structure

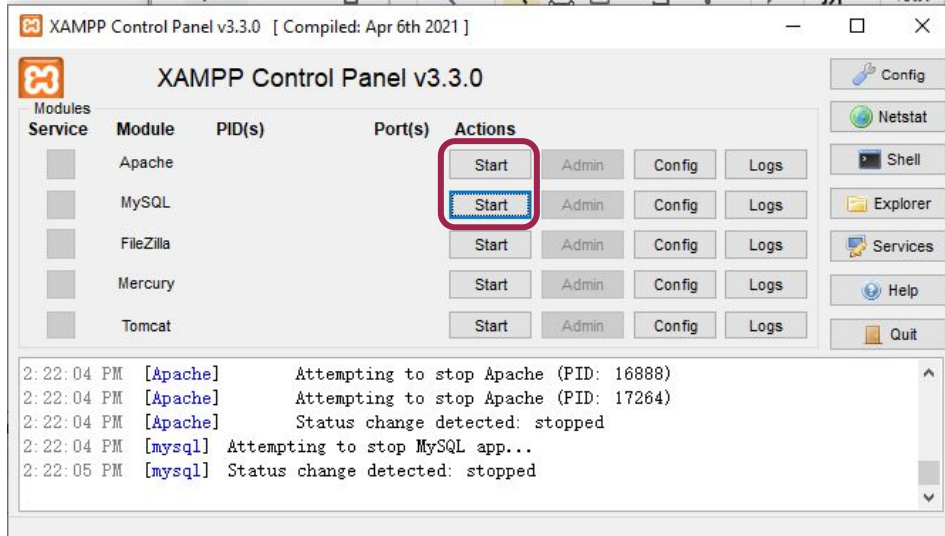
Start/stop the server

Server Setup

- We will use XAMPP (<https://www.apachefriends.org/index.html>) in this course.
- The source codes of the server can be found at the `“./htdocs/”` directory. This is the place where we should put *.php and *.html files.
- After starting the server, go to `“http://127.0.0.1/”` to access the server.



Control Panel (Win10)

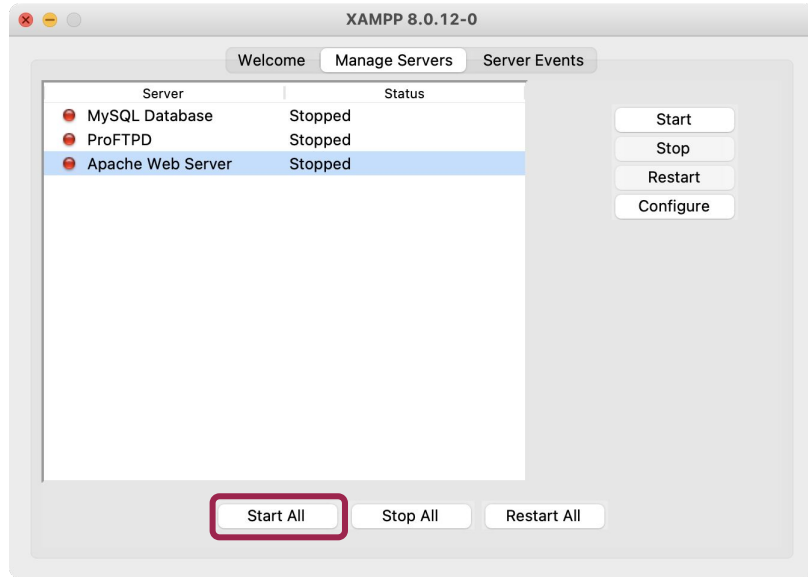


Start the Server

Codes Directory

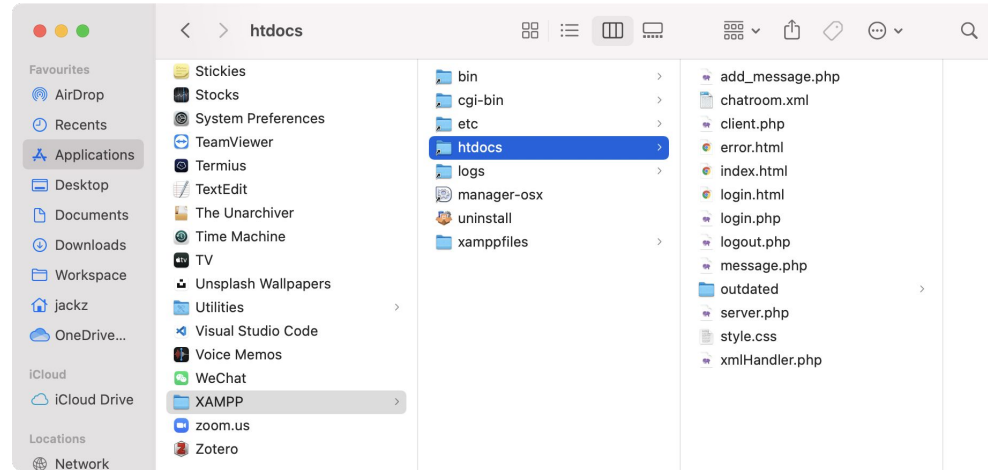
anonymous	11/3/2021 12:53 PM	File folder
apache	11/3/2021 12:53 PM	File folder
cgi-bin	11/3/2021 12:57 PM	File folder
contrib	11/3/2021 12:53 PM	File folder
FileZillaFTP	11/3/2021 12:57 PM	File folder
htdocs	11/3/2021 1:13 PM	File folder
img	11/3/2021 12:53 PM	File folder
install	11/3/2021 12:57 PM	File folder
licenses	11/3/2021 12:53 PM	File folder
locale	11/3/2021 12:53 PM	File folder
mailoutput	11/3/2021 12:53 PM	File folder
mailtodisk	11/3/2021 12:53 PM	File folder
MercuryMail	11/3/2021 1:11 PM	File folder
mysql	11/3/2021 12:53 PM	File folder
perl	11/3/2021 12:54 PM	File folder
php	11/3/2021 12:57 PM	File folder
phpMyAdmin	11/3/2021 12:55 PM	File folder
sendmail	11/3/2021 12:56 PM	File folder
src	11/3/2021 12:53 PM	File folder
tmp	11/3/2021 1:01 PM	File folder
tomcat	11/3/2021 12:53 PM	File folder
webalizer	11/3/2021 12:57 PM	File folder
...	11/3/2021 12:53 PM	File folder

Control Panel (MacOS)



Start the Server

Codes Directory



Example

- In this lab, we will use XAMPP on MacOS to demonstrate, but the operation is the same on Win10.
- Download the “[helloworld.php](#)” and place it in the “[./htdocs/](#)”, then go to “[http://127.0.0.1/helloworld.php](#)” to see the result.



PHP Basics

Basic Grammar

Conditional Statement

Loops

Functions

PHP Basics

- A PHP script starts with `<?php` and ends with `?>`
- Not case-sensitive for key words (e.g. `if`, `for`, `while`, `echo`), but case-sensitive for variable names
- PHP has several ways to comment (e.g. `#`, `//`, `/* */`)
- Each line of PHP code ends with `;`
- PHP variable starts with the `$` sign
- PHP use `echo` to output data to the screen

```
$txt = "PHP is hard to use";  
echo "I say $txt";
```

PHP Conditional Statement

- `if...elseif...else` statement: executes different codes for multiple conditions
- `switch` statement: selects one of many blocks of code to be executed

```
$n = 1;  
if ($n > 0) {  
    echo "Greater than zero";  
} else {  
    echo "Not greater than zero";  
}
```

PHP Loops

- **while**: repeat the block of code as long as the condition is true
- **do...while**: run the block of code once, and then repeat it as long as the condition is true
- **for**: repeat the block of code for a specified number of times
- **foreach**: repeat the block of code for each element in an array



PHP Functions

- Create a user defined function with the word **function**.

```
function printName($txt) {  
    echo "My name is $txt.<br>"  
}
```

```
// To use the function  
printName("Mark");
```



PHP Advanced

- The previous slides have summarized a few important parts of PHP, and you may learn advanced contents of PHP in the lecture.
- You may also refer to some free online PHP tutorials (e.g. <https://www.w3schools.com/php/>).



Chatroom

Overall Structure

Server Side

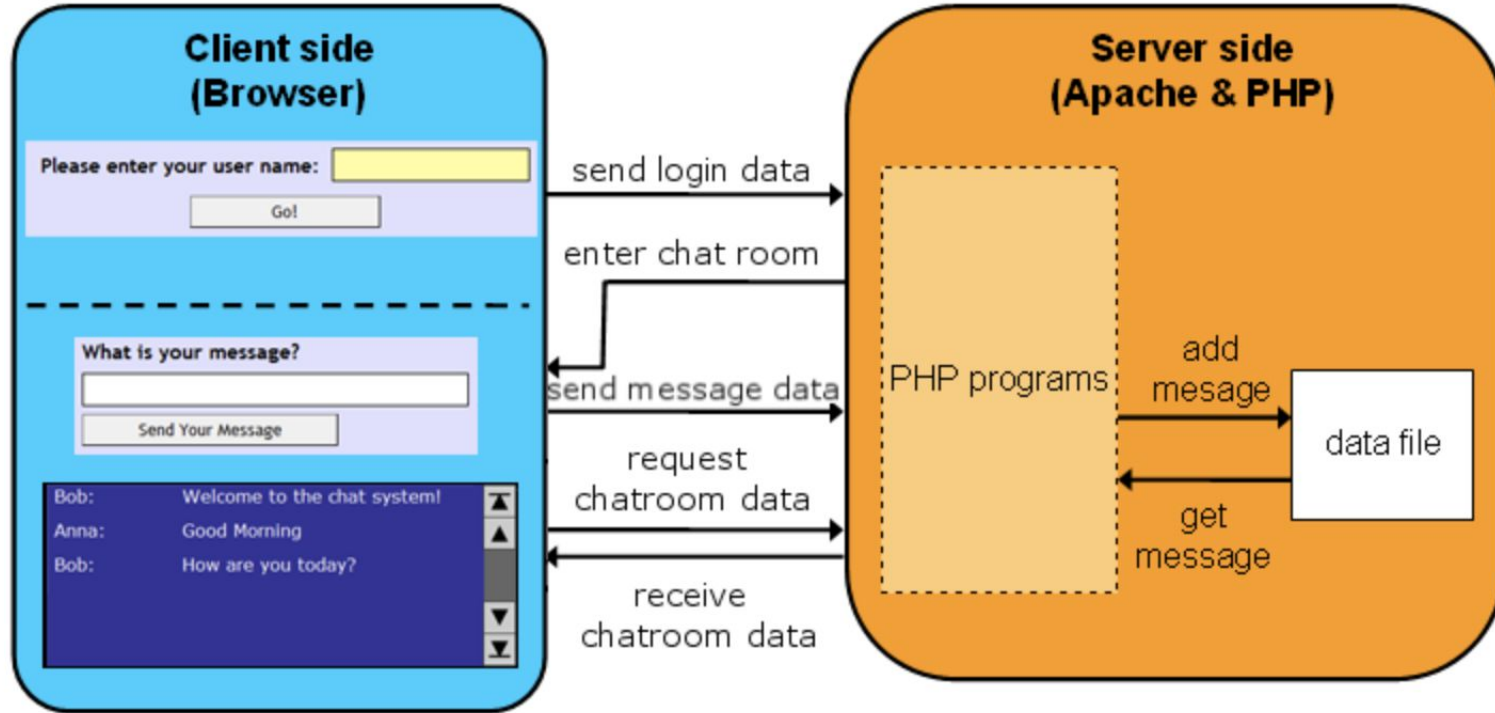
Client Side

Chatroom Project

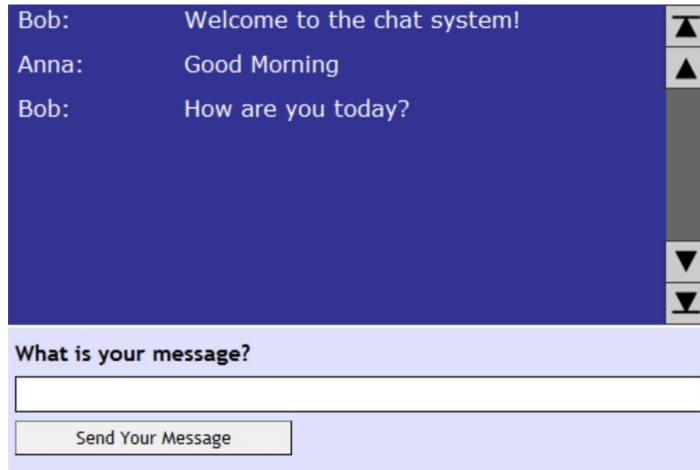
- In the last course project, we will build a real chatroom system and deploy it in a server.
- This lab will contain some basic parts of the chatroom, which can be kept for further development.
- You can find the start codes and the solution codes in the canvas.



Overall Structure



Overall Structure



Bob: Welcome to the chat system!

Anna: Good Morning

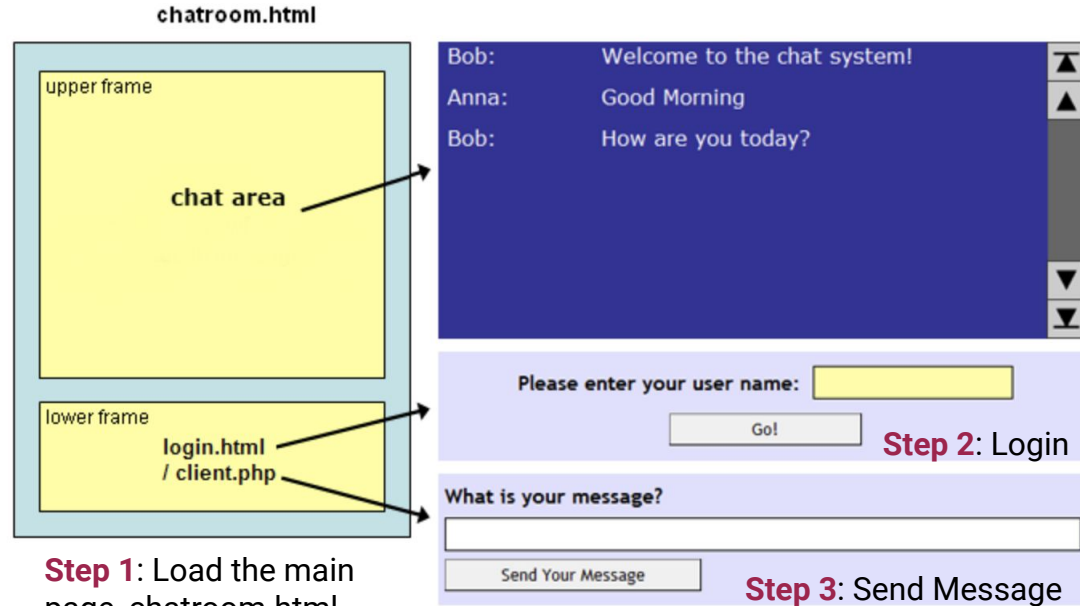
Bob: How are you today?

What is your message?

Send Your Message

Basic Interface

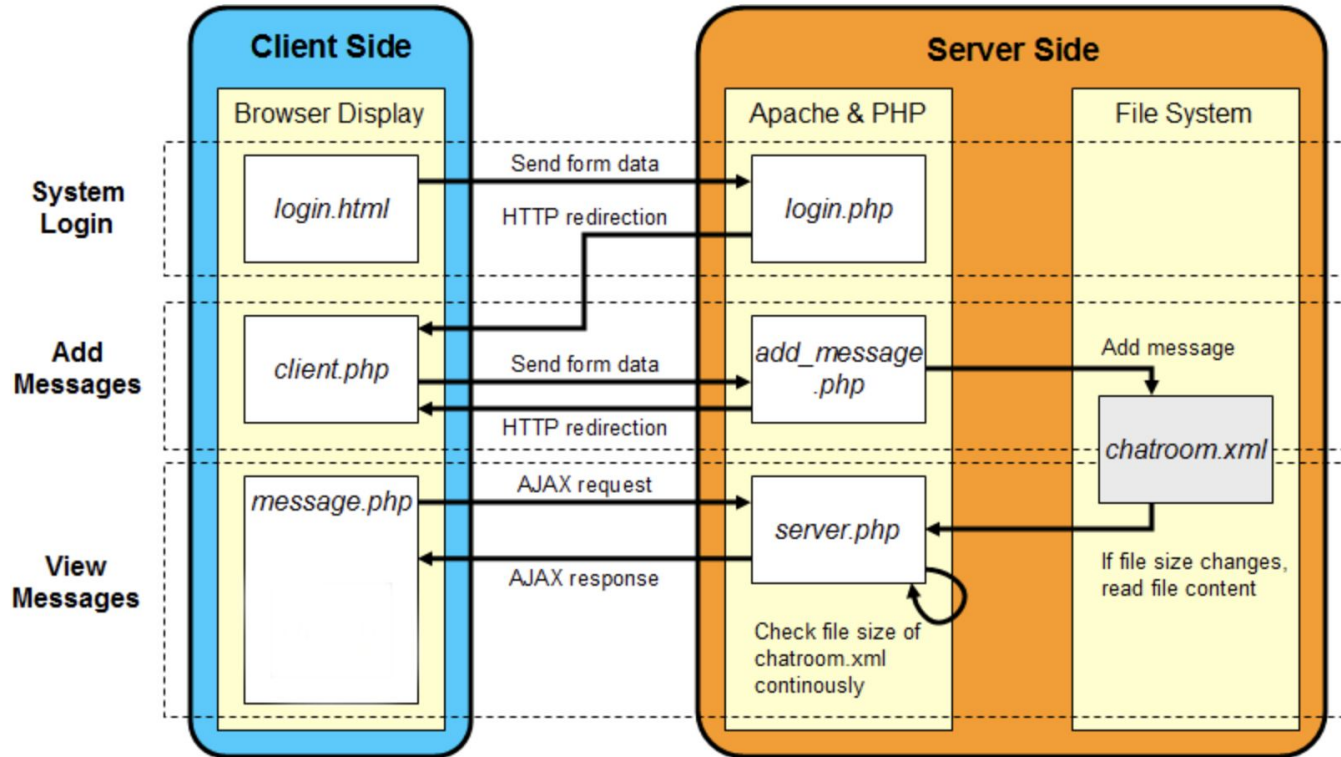
Frame Structure



Step 1: Load the main page, chatroom.html

Step 3: Send Message

Information Flow



Server Side

- There are three files mainly responsible for handling the operations in the server side: `login.php`, `add_message.php`, and `server.php`
- They are designed to be used for server side operations only. Note that we could have combined both the server operations and client display together in one file. To understand the system easily, we choose to use separate files for server-side operations and client-side display.



Handle User Login: login.php

- After the user enters a name in `login.html`, the name will be sent to this file as form data
- The purpose of this php script is to set the value of this user name into a **cookie**
- This can be easily done by calling the `setcookie` function in php, for example,

```
// Transfer from form data into cookie  
setcookie("name", $_POST["name"]);
```

- The above code sets the value from the form data "name" to a cookie with the same name
- After setting the cookie, every page requested by the client in the same session can also access this cookie, i.e. we can retrieve the user name in another php script, such as `add_message.php`
- After setting the cookie, the page is redirected to `client.php`, which is the message input page for the user
- This can be done by a `Location` HTTP header command, for example,

```
header("Location: client.php");
```

Add Message: add_message.php

- This php script is responsible for updating the content of our chat room
- The messages inside the chat room are stored in a single XML file called `chatroom.xml` located in the server
- When it receives the input coming from `client.php`, it will add the new message to the end of the XML file
- A new message consists of two values: name and message content
- The following format is used for a message in the XML file

```
<message name="... name here ...">... message content here ...</message>
```

- You need to add the code for adding a message to the XML file in this lab
- Here is the content of XML file `chatroom.xml` for the chat messages in the interface figure [above](#)

```
<?xml version="1.0" ?>
<chatroom>
  <users>
    <user name="Anna" />
    <user name="Bob" />
  </users>
  <messages>
    <message name="Bob">Welcome to the chat system!</message>
    <message name="Anna">Good Morning</message>
    <message name="Bob">How are you today?</message>
  </messages>
</chatroom>
```

Add Message: add_message.php

- The name is retrieved from the cookie set in login.php and the content is retrieved from form data
- To open an XML file, you can use the following:

```
$xmlh->openFile();
```

- Then we can add a new message element to the messages group by:

```
// Get the 'messages' element as the current element
$messages_element = $xmlh->getElement("messages");

// Create a 'message' element for each message
$message_element = $xmlh->addElement($messages_element, "message");
```

- Then you can add the name and the text message:

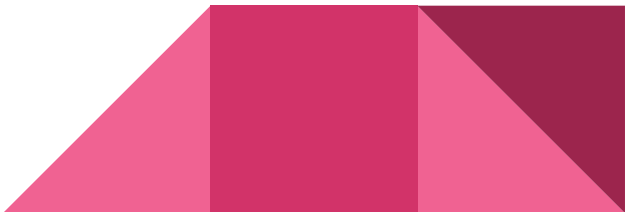
```
// Add the name
$xmlh->setAttribute($message_element, "name", $name);

// Add the content of the message
$xmlh->addText($message_element, $message);
```

- Remember to save it:

```
$xmlh->saveFile();
```

- Again, after updating the chatroom.xml, the page is redirected back to client.php for the next message input



Handle Requests: server.php

- This file is used as the server connection for `message.php`
- The main purpose of this file is to output the chatroom messages from `chatroom.xml` to `message.php`
- When this file is requested by the client, the output is not immediately sent
- Instead, the output will be sent only if there is any change in the file `chatroom.xml`
- Therefore, the php script may run for a long time if there is no change in the chat room
- We need to set the maximum execution time for the script because a php script cannot run forever in the server by default
- Using the following code, we can set the maximum execution time to be 60 seconds

```
set_time_limit(60);
```

- Then we let a php script to run at most 30 seconds before sending the output even if the content is not changed
- The file size of the XML file `chatroom.xml` will be obtained from the request from `message.php` (discussed later)
- After detecting a change of file size for `chatroom.xml`, we need to send the XML output
- The JavaScript code in `message.php` finds the new messages in the XML output and then displays the messages in the Flash area

Client Side

- The user can see the output of these three files: `login.html`, `client.php`, `message.php`



login.html

- This is a very simple form which accepts only one value, the user name



Please enter your user name:

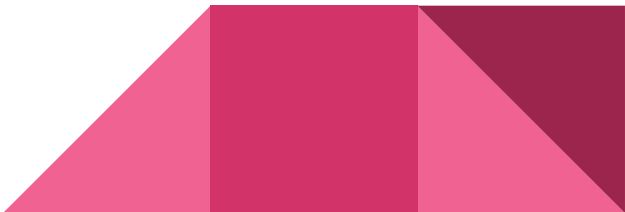
- This user name will be sent to `login.php` for processing in the server

client.php

- This is the main user input of the chat room

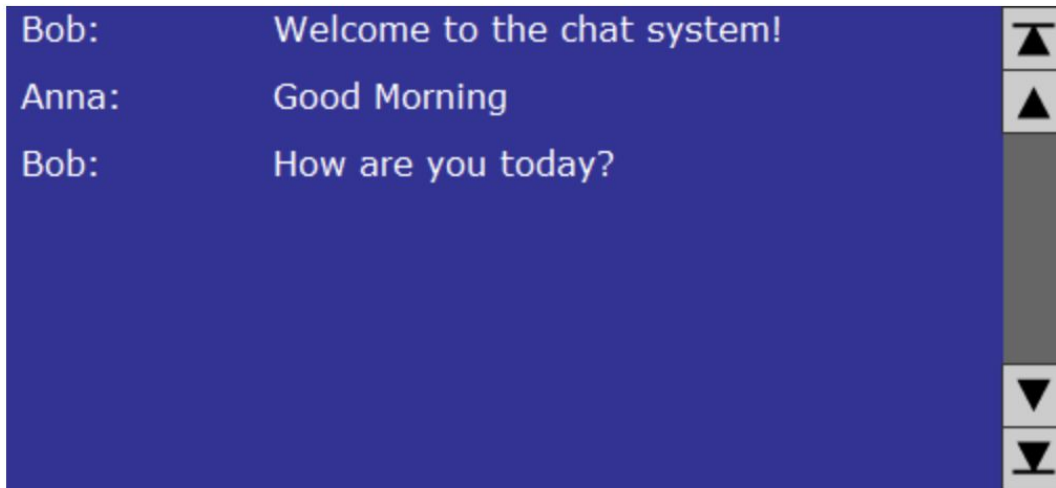
What is your message?

- The input from this form is sent to `add_message.php` so that the content of `chatroom.xml` can be updated



Get Messages from Server: message.php

- This is the chat room display of the system



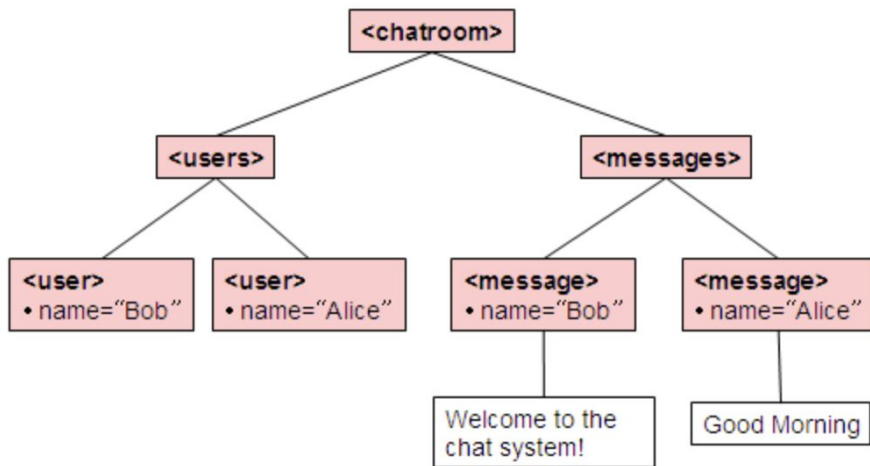
- In this lab, we use AJAX techniques to handle the request and response, i.e. we send request to the server and process the response using JavaScript

Get Messages from Server: message.php

- In function `updateChat()`, a for-loop is used to scan through all message nodes inside the DOM tree
- For example, if there are two new messages, like this:

```
Bob: Welcome to the chat system!  
Alice: Good Morning
```

- Then the DOM tree looks like this:





Thank you!