EXERCISE05: AJAX – PHP - JSON

Objectives:

- **To learn to use jQuery/ajax/JSON**
- **To learn basics of PhP (and use of session variables).**
- **To learn/use basics of asymmetric cryptography.**

Credits (for edits): Naresh Somisetty, Zahra Hosseini, Isaac Martin, Sai Nemalikanti, Hosam Abdeltawab

Work with your group (or by yourself). Each group is to upload only one submission.

1 Run UwAmp server

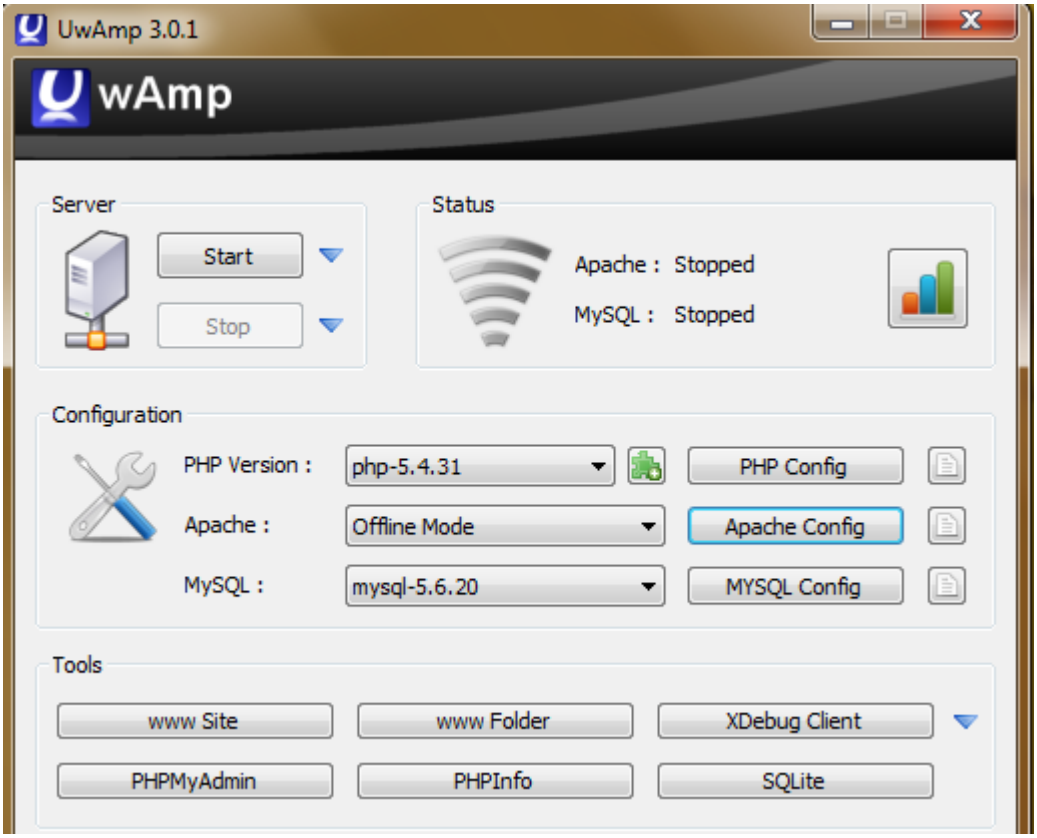
UwAmp is a WAMP server (Windows Apache MySQL, PHP). It does not need administrator privileges to install/run. You can download it by using the following steps. You can also run it off a USB stick.

1.1 Download UwAmp

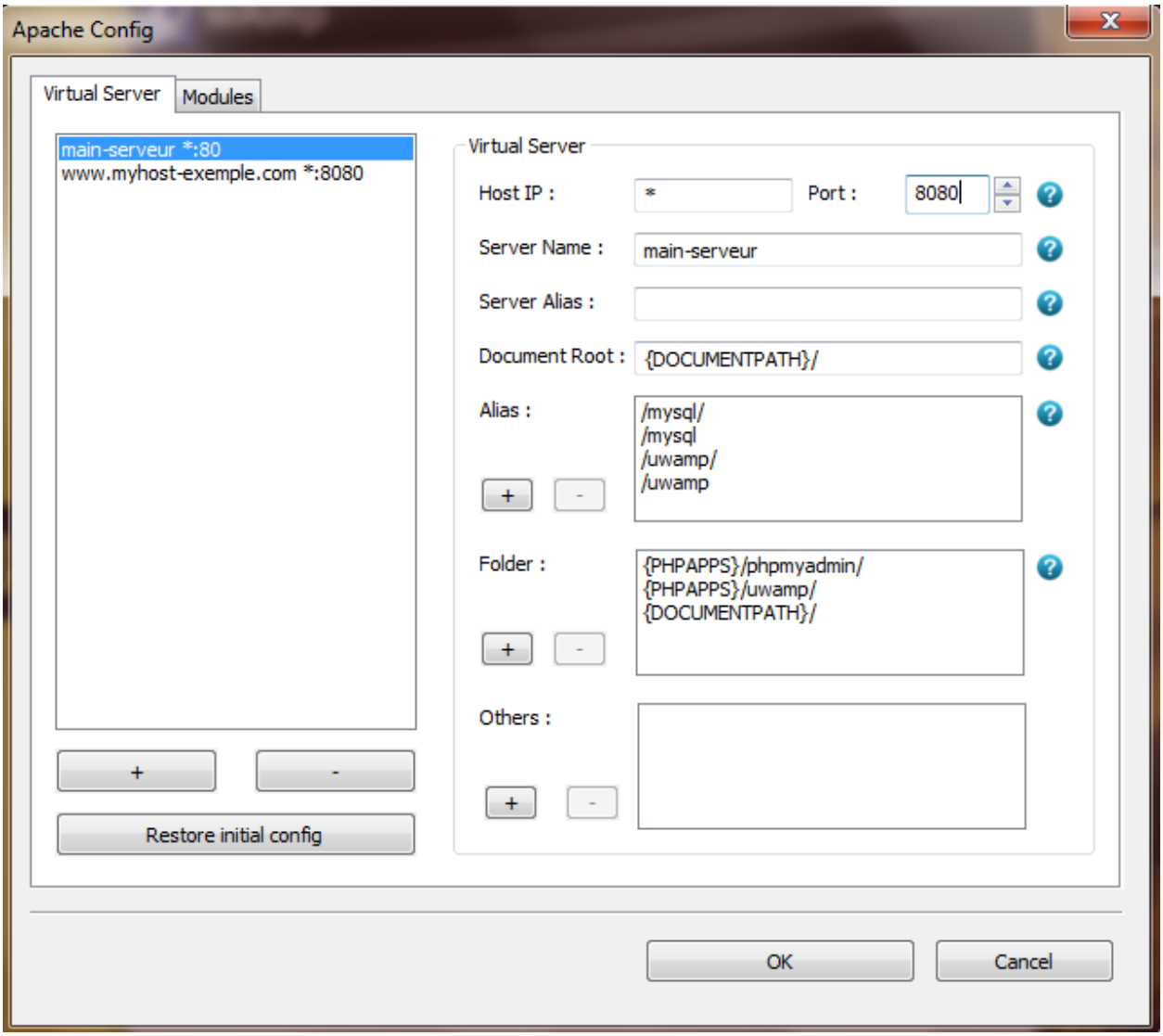
1) go to http://www.uwamp.com/en/?page=download
2) download zip file and extract the contents into a folder.
3) Please use UwAmp version (3.1.0) and use PHP Version 5.6.X.

1.2 Change port# to 8080 and start UwAmp

1) Open Uwamp and click on the Apache Config (see below highlighted in blue),



2) select the main-serveur Virtual Server as shown in the picture below, change the port (on the right side of the view) to 8080, and press ok. Then, start server.



1.3 Check to make sure it is working ok

1) Run UwAmp.
2) Select the first two check boxes you get.
3) Make sure that ALL the PHP code you want to run is in a folder INSIDE the www folder of UwAmp.

2 Warm Up: Try Some Examples

Download the examples from the assignment and try them out! The index.html file explains the purpose of each example.

a) Make sure that ALL the PHP code you want to run is in a folder INSIDE www folder of UwAmp.

b) type "localhost:8080" on browser to access your server code.

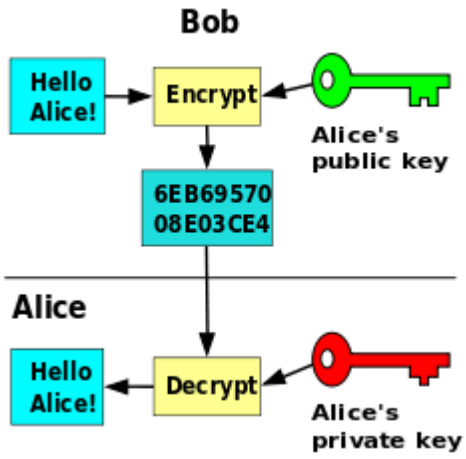
The index.html files acts as a README file and has links to the examples.

3 Information On Asymmetric cryptography

Public-key cryptography, or asymmetric cryptography, is any cryptographic system that uses pairs of keys: public keys that may be disseminated widely paired with private keys that are known only to the owner. There are two functions that can be achieved: using a public key to authenticate that a message originated with a holder of the paired private key; or encrypting a message with a public key to ensure that only the holder of the paired private key can decrypt it. More information related to this type of cryptography can be found on

https://en.wikipedia.org/wiki/Public-key_cryptography

The following picture shows the method of encryption and decryption needed for this assignment. Here Bob is using the public key to encrypt and Alice is using private key to decrypt.



The library **phpseclib** is provided to you (in file **phpseclib.zip** in the assignment folder). It works on PHP4+ (PHP4, assuming the use of PHP_Compat) and doesn't require any extensions. In this assignment we use this library for generating public key and private key, and for encryption and decryption.

An example use of this library has also been placed in the assignment folder. It is named **ExampleCryptography.php**.

4 Client/Server Posts Application.

Create a SINGLE PAGE web application to create, view**,** and update posts on a webpage (i.e. use ajax to update elements of the web page when you do some operation). Note that you will need separate pages for login and signup (but only one page for create/view/update posts). You can design the UI to look as you wish, however, the requirements below must be met.

4.1 Functionality (UC means Usecase)

UC0 Signup: User signs up to the website, and after hitting enter, webpage redirects to login page.

UC1 View Posts: After successfully logging in, the user is shown a list of posts (made by him/her and other users).

UC2 Make a Post: user Tom logs in. He clicks on the "make a post" button and enters his post with title and description. After he hits enter, the list of posts gets updated on the screen (with the latest post at the top).

UC3 Edit a Post: user Tom clicks on an existing post that he has created before. He enters a new message description that overwrites the old post description.

UC4 Logout: user Tom clicks on the "logout" button and is taken back to the login page.

UC5 Admin view: The administrator (user name is admin) logs in to the website. After successfully logging in, he is shown a list of all posts. (Admin uses the same html file as other users).

UC6 Admin Delete a post: Admin user can delete any post.

UC7 Send Message: A user clicks on “send message button”. Then the user enters a receiver user name and a message. After he/she hits enter, the message should be encrypted and sent to the receiver.

UC8 View Message: All users, after successfully logging in, are shown a list of messages (sent to them by other users).

4.2 Design

You will need to create files Signup.php (to show signup page), login.php (to show the login page), viewPosts.php (for create/view/update/delete messages), checkLogin.php (make ajax call to this), updatePosts.php (which will also be used to create a post; make ajax call to this), sendMessage.php (send ajax call to this), inbox.php, and a logout.php (redirect back to login page).

Note that you will need to use session_start(), session_destroy() methods as appropriate in your code. You can store posts info in a text file "posts.txt" It might be a good idea to store the data in JSON format.

Format of information in posts.txt: Title, description, and the time that it was posted.

4.2.1 Signup.php

- When a user goes to the signup page and enters his username and password, store the username and password and generate a public key and private key (using phplibsec library) and store the information (users/passwords/public key/private key) in a text file "users.txt". Here is an example entry in this file.
  smitra:testPassword:ddhfh9fi3f3i:94hredbb

- *Note: you can use JSON object to store information.*

4.2.2 In login.php

- use ajax call to /checkLogin.php file to check user/password
- on success go to viewPosts.php
- on failure – redirects to login.php.

4.2.3 in viewPosts.php

- use file_get_contents php function to get data from posts.txt; convert to php object (using phplibsec library)
- Create a table, in each row your program should shows title, description, time and update button.

- loop through this object and create html rows with a link to update post (that will use javascript to bring up a prompt to get the message from the user and then make an ajax call to updatePosts.php to do the update) and then refresh only the table (and not the rest of the page).
- The Posts can be stored in JSON format as an ARRAY of posts. Each post consists of an userID, postID, postTitle, and postTime.
- You can store the post object in the session object.

- Also, on this page, have a send button, which when pressed will bring up a simple UI with a "TO" input field, "FROM" input field, message BODY input field, and a SUBMIT button.
- Also, on this page, have a "inbox" button to direct to inbox page.

4.2.4 in checkLogin.php

- read the users.txt file and then check the user/password.
- return a JSON object with the success/failure info

- Example:

```
$data1 = "True";

$data2 = "False";

header('Content-type: application/json');

if ( login_succeed) echo json_encode( $data1 );

// prints zero-based JS array: ["a","b","c"]

// accessed in JS like: result[1]

(returns "b")

else echo json_encode( $data2 );
```

4.2.5 in updatePosts.php

- if the user is admin, he/she can delete any post (Session Object needs to store username and username can be retrieved from it)
- Posts can be stored in JSON format as an ARRAY of post objects. Each post consists of an userID, postID, postTitle, and postTime.

- if current user(except admin) posts some information which are stored in posts file
    - then creator of post can modify it

    - else create a new entry
- use file_put_contents() php function to store the posts back into the posts.txt file(and update time on posts.txt)
- also update the session object if needed.

4.2.6 in sendmessage.php

- When User clicks on SUBMIT button when sending message, the message should be encrypted by Receiver PublicKey and store information (contain Sender username, Receiver and body of message) in messages.txt file. Each item should contain the information as in this example: From: Alice; To: BOB; Body: dshgkfjsghfjksghjkf
- Make sure that you store the encrypted message in messages.txt file.

4.2.7 in inbox.php

- use file_get_contents to get JSON data from messages.txt. Next, convert JSON to PHP object (use json_decode or json_encode).
- loop through object: If the "TO" field is the same as the current username, then create html rows with after decrypting the body of the message to display the message.
- Have a "Posts" button to redirect to posts page.

4.2.8 in logout.php

- destroy the session
- re-direct the page to login.html

NOTE – we DO NOT really expect you to implement file-locking mechanisms to avoid concurrent read/write accesses to posts.txt. (so ignore this issue).

4.2.9 CheckList

[ ] All files and pages listed above exist, and work, as expected (satisfy use-cases).

[ ] Each post should contain at least a Title, description, and the time that it was posted.

[ ] Every post is editable and updatable.

[ ] Webpage has a working create post section.

[ ] Use only AJAX to get/create/update posts.

[ ] Each Message should contain at least a FROM, TO, and Body part.

[ ] Webpage has a working Send Message section.

[ ] Use only AJAX to get/create Message.

[ ] Please use relative paths to file names in your codes (do not use absolute paths)

EXTRA CREDIT: implement a "like" button implementation for each post.

5 Submission

Make sure your code is working. Then, Zip your html and PHP files, and participation file (i.e. who worked on which part or if you worked together). Then, submit this zip file on black board. Remember there is only one submission per group. Make sure to include all the files that are needed in order to run your program. DO NOT SUBMIT EXTRANEOUS FILES.

Participation file is a simple txt file, which clarifies the specific participation of two members.