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.

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

1 Run UwAmp server

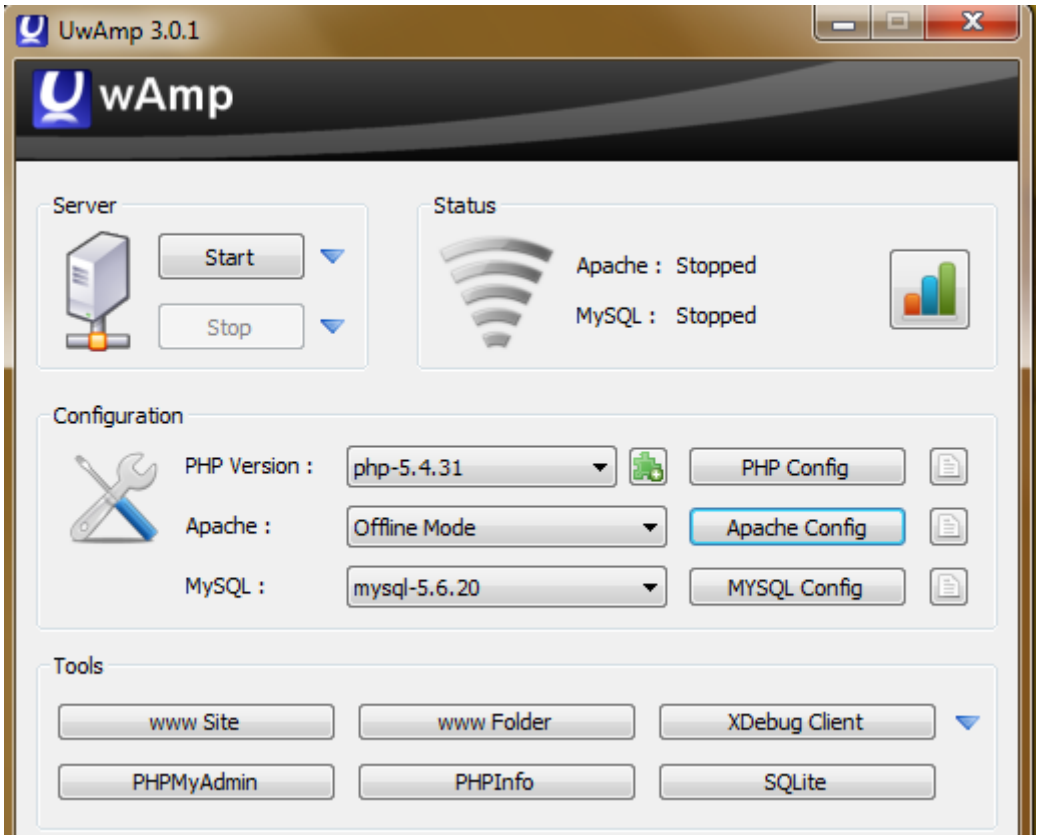
UwAmp is a WAMP server (Windows Apache MySQL, PHP). It does not need administrator privileges to install/run. You can download it by using the following steps. You can also run it off a USB stick.

1.1 Download UwAmp

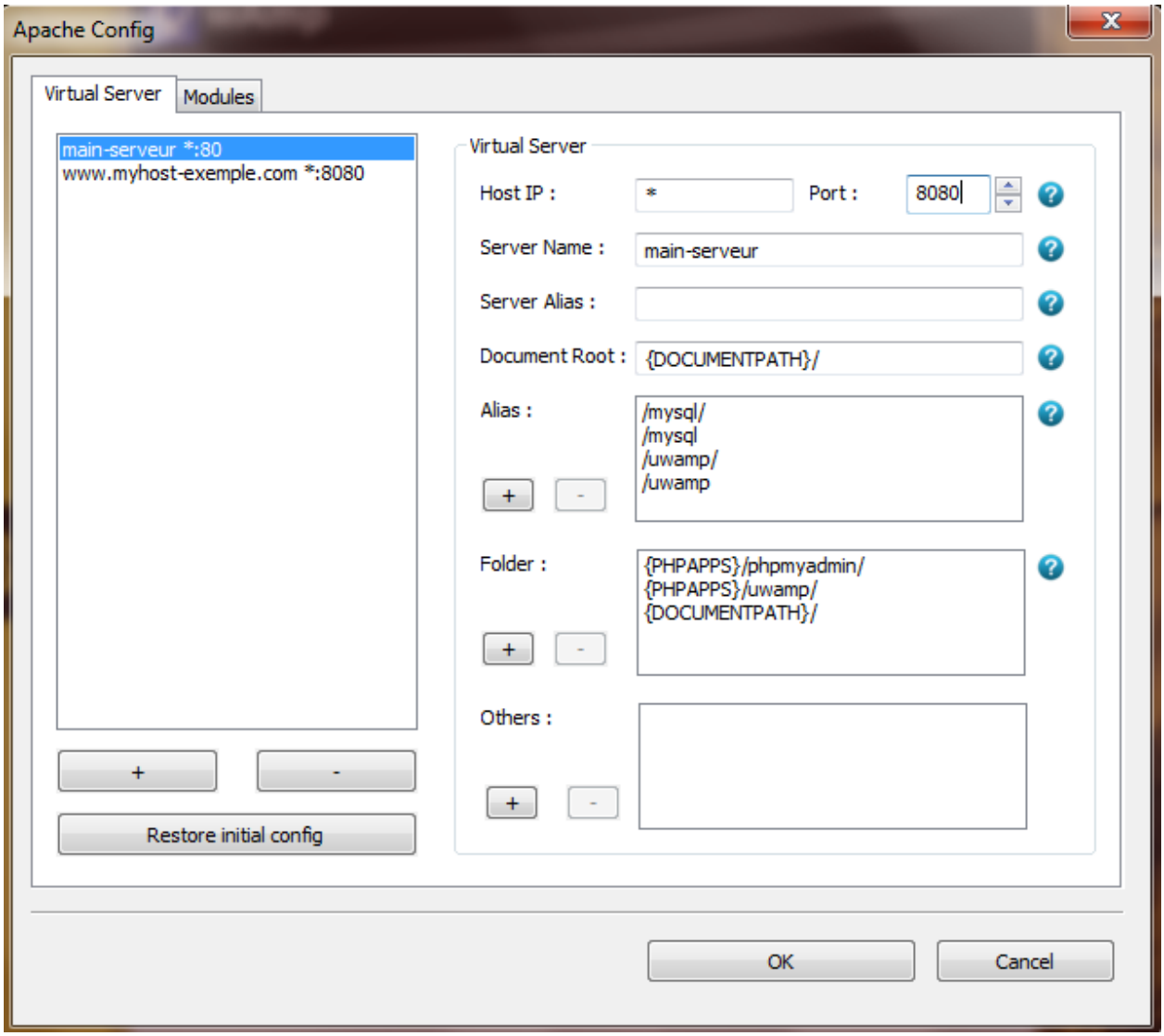
1) go to http://www.uwamp.com/en/?page=download
2) download zip file and extract the contents into a folder.

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 (See below), change port to 8080, press ok, and 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 www folder of UwAmp.

2 Warm Up: Try Some Examples

Download the examples from the assignment and try them out!

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.

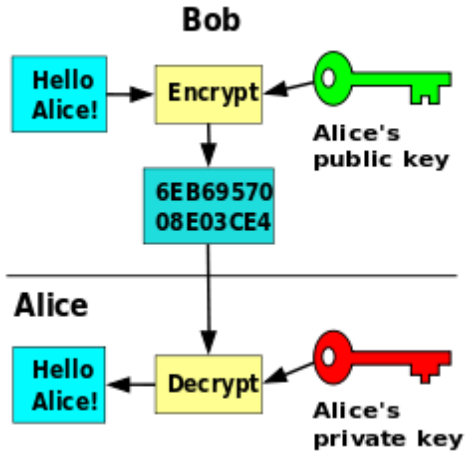
The index.html files acts as a README file and has links to the examples.

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.

Example of an use of this library (named **ExampleCryptography.php**) has also been placed in the assignment folder.

4 Client/Server Posts Application.

Create a SINGLE PAGE web application to create, view**,** and update posts on a webpage. You can design the UI to look as you wish, however, the requirements below must be met.

4.1 Functionality

UC0 Signup: User signup to the website, 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 message. 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 a post that he has made before. He enters a new message that overrides the old post.

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

UC5 Admin view: The administrator (user admin) logs in to the website. After successfully logging in, he is shown a list of all posts.

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

UC7 Send Message: All users (expect admin) clicks on “send message”. Then the user enters a receiver user name and a message. After he hits enter, the message should be encrypted and sent to the receiver.

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

4.2 Design

You will need to create files Signup.html, login.html (to show the login page), viewPosts.php, checkLogin.php, updatePosts.php (which will also be used to create a post), sendMessage.php, inbox.php and a logout.php.

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

4.2.1 Signup.html

- 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

4.2.2 In login.html

- use ajax call to /checkLogin.php file to check user/password
- on success go to viewPosts.php
- on failure – ask the user to correct the username and password

4.2.3 in viewPosts.php

- use file_get_contents php function to get data from posts.txt; convert to php object (using json_decode/json_encode php functions)
- loop thru 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 refreshes only the table (and not the rest of the page).
- can store the post object in the session object.

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

4.2.5 in updatePosts.php

- if the user is admin, he/she can delete any message.
- if the post information sent by the current user (except admin) is already in the posts file
    - then modify it
    - else create a new entry
- use file_put_contents php function to store the posts back into the posts.txt file
- also update the session object if needed.

4.2.6 in sendmessage.php

- Create a simple UI with a text file for Receiver part, User(fill with current username) and Body part, with submit button.
- When User clicks on submit, 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 User: Alice; Receiver: 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 (json_decode/json_encode)
- loop thru object: If the To part is related to the current username create html rows with after decrypting the body of the message.

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. (i.e. ignore this aspect).

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 Sender, Receiver, and Body part.

[ ] Webpage has a working Send Message section.

[ ] Use only AJAX to get/create Message.

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

5 Submission

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.

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