# Exercise07: AJAX – PHP - JSON

## Objectives:

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

**Credits (for edits): Naresh Somisetty, Zahra Hosseini, Isaac Martin.**

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

# 1 Run UwAmp server

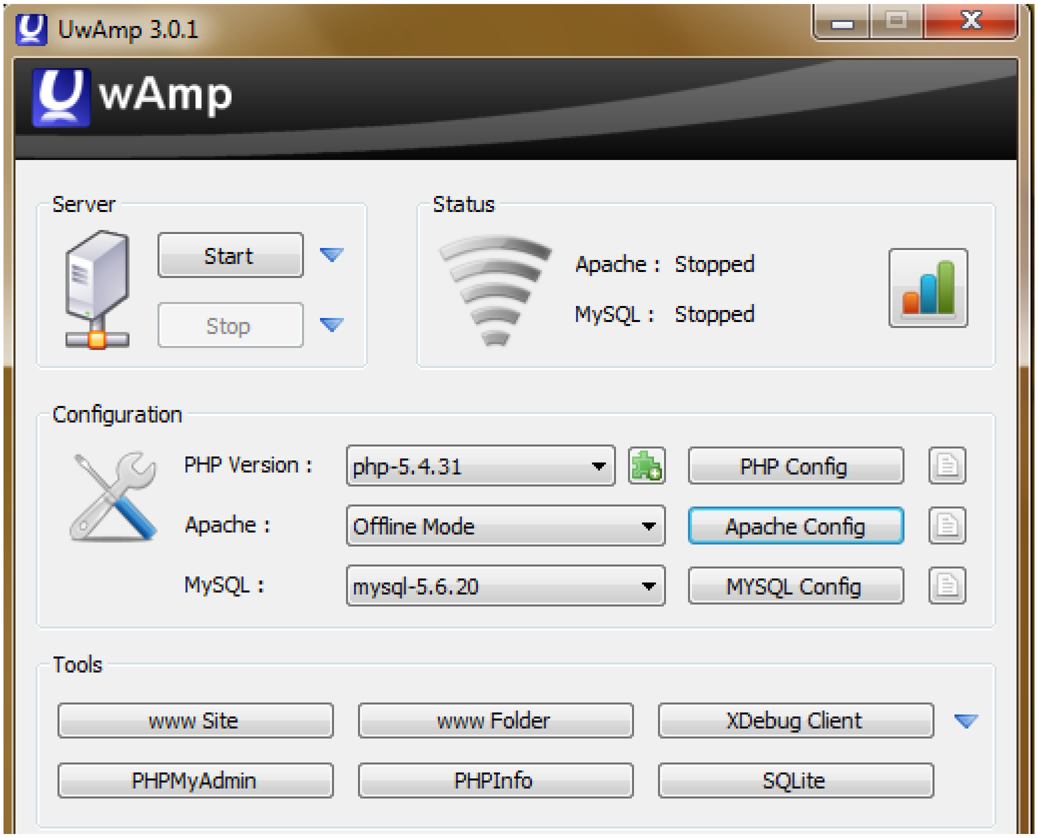
UwAmp is a WAMP server (Windows Apache MySQL, PHP). It does not need administrator privileges to install/run. You can download it by using the following steps. You can also run it off a USB stick.

## 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),



Apache : Stopped

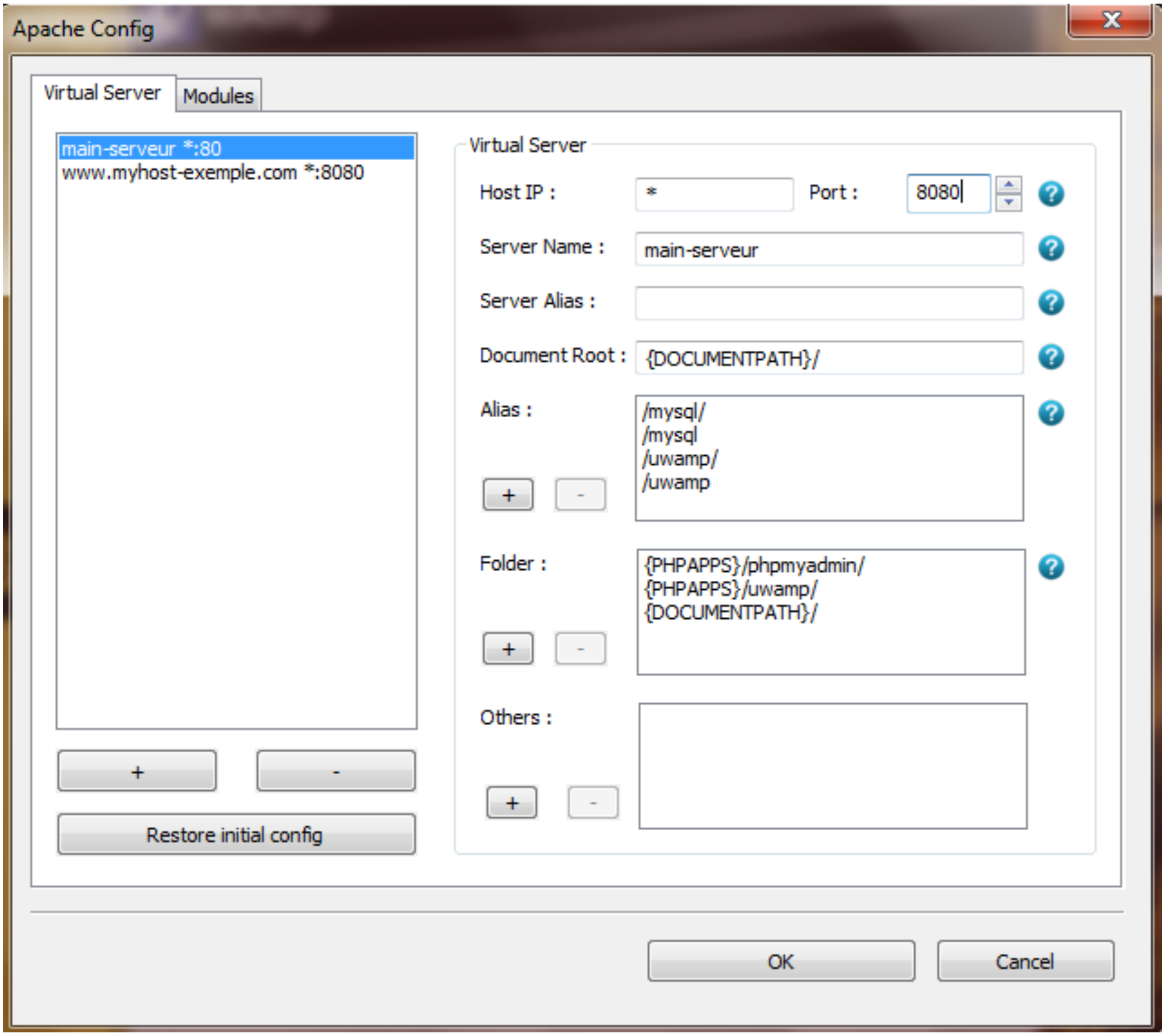
MySQL : Stopped

PHP Version : php-5.4.31

Apache : Offline Mode

MySQL : mysql-5.6.20

2) select the main-server (See below), change port, press ok, and start server.



main-serveur *:80
www.myhost-exemple.com *:8080

Host IP : *
Port : 8080
Server Name : main-serveur
Server Alias :
Document Root : {DOCUMENTPATH}/
Alias : /mysql/<br>/mysql<br>/uwamp/<br>/uwamp
Folder : {PHPAPPS}/phpmyadmin/<br>{PHPAPPS}/uwamp/<br>{DOCUMENTPATH}/
Others :

## 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" on browser to access your server code.

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

# 3 Client/Server Posts Application.

Create a SINGLE PAGE web application to create, view, and update posts on a webpage. Design the page as you wish.

## 3.1 Overview

UC1 View Posts: user Tom brings up the website. He is shown a login page. After successfully logging in, he is shown a list of posts (made by him 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.

## 3.2 Design

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

You can store the users info (assume pre-created users/passwords) in a text file "users.txt"

You can store posts info in a text file "posts.txt"

It might be a good idea to store the data in JSON format.

### 3.2.1 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

### 3.2.2 in viewPosts.php

- make sure to start session. store user name. (session_start)
- use file_get_contents from posts.txt; convert to php object (json_decode/json_encode)
- loop thru object and create html rows with a link to update post (that will use javascript to bring up a prompt to get the update 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.

### 3.2.3 in checkLogin.php

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

### 3.2.4 in updatePosts.php

- if the post information sent by the client is already in the posts
  - then modify it
  - else create a new entry
- use file_put_contents to store the posts back into the posts.txt file
- also update the sessions object if needed

### 3.2.5 in logout.php

- invalidate 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).

## 3.3 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.**

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

# Submission:

Zip your html and php files and submit 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.