Simple. Secure. Login v3.4 Documentation

First I'd like to thank all of those who have bought **SSLv3** and any of my other products for that matter.

Help

If anyone one like any help with the use of **SSLv3** then contact me by the form on my website, this can be found here: http://phpcodemonkey.com/contact.html. Please read through the documentation first as it may just answer your question.

Changes

I have made many changes to SSLv3.4 I have rewritten most of the applications core and process files to help improve performance and fix some bugs.

1) Gravatar - *You can now access a users gravatar.*
2) Template Engine - *Makes it easier to send emails and customise them.*
3) SALTS - SSLv3.4 now uses password SALTS to help secure it's users.

Upcoming Changes

It's hard to release updates on a regular basis for this system due to my busy work load at university. I will however try to release incremental updates every 2 - 3 weeks that will add more functionality to SSLv3

1) Permission System - *Will allow the administrator to lock down pages.*

Installation

Installation is simple, follow the steps and you'll be fine. First thing is to ensure you have the required setup to install SSLv3.

Requirements!

I can’t stress the importance of this step, please ensure you have the following before continuing with installation.

Minimum PHP Version: 3.3.X

To check this create a blank document on the server you are installing it on and name it ***test.php.*** *Copy the following contents to the file...*

```
<?php
echo phpversion();
?>
```

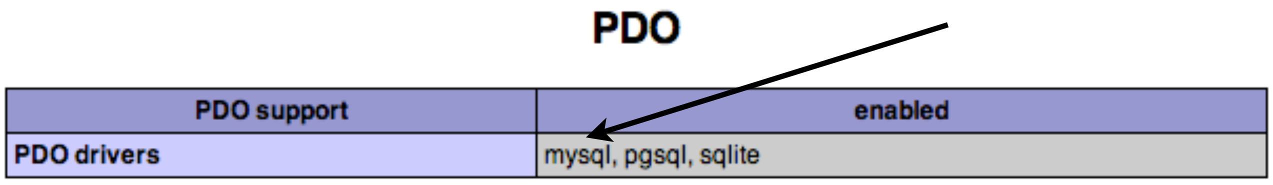
Open the page and you will see the version of php you have remember you are looking for version 3.3.x of higher!

Next...

We have to ensure you have PDO extension enabled and PDO MYSQL driver installed. To do this open up the **test.php** file we just created and add this:

```
<?php
phpinfo();
?>
```

Open the page in the browser and hit CMD + F (Mac) or CTRL + F (Windows/Linux) type in PDO and look for the following:



PDO

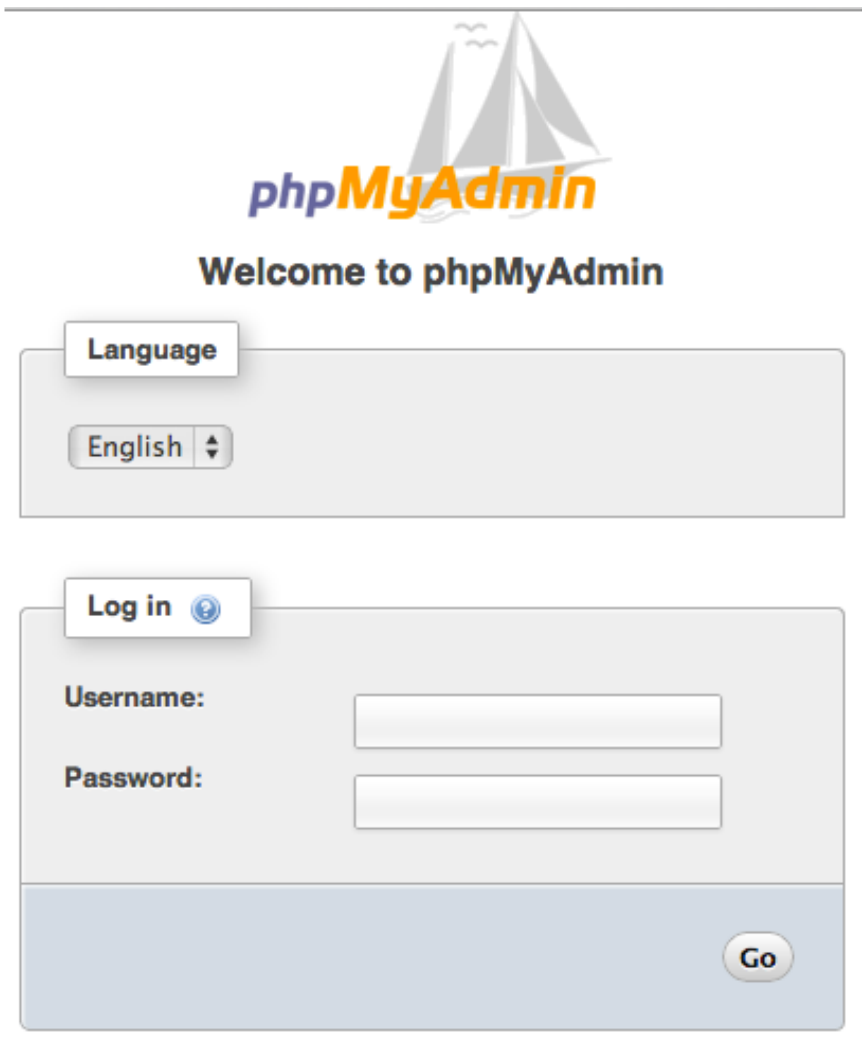
PDO support	enabled
PDO drivers	mysql, pgsql, sqlite

If you see this your all set! We should be able to install the application without a problem.

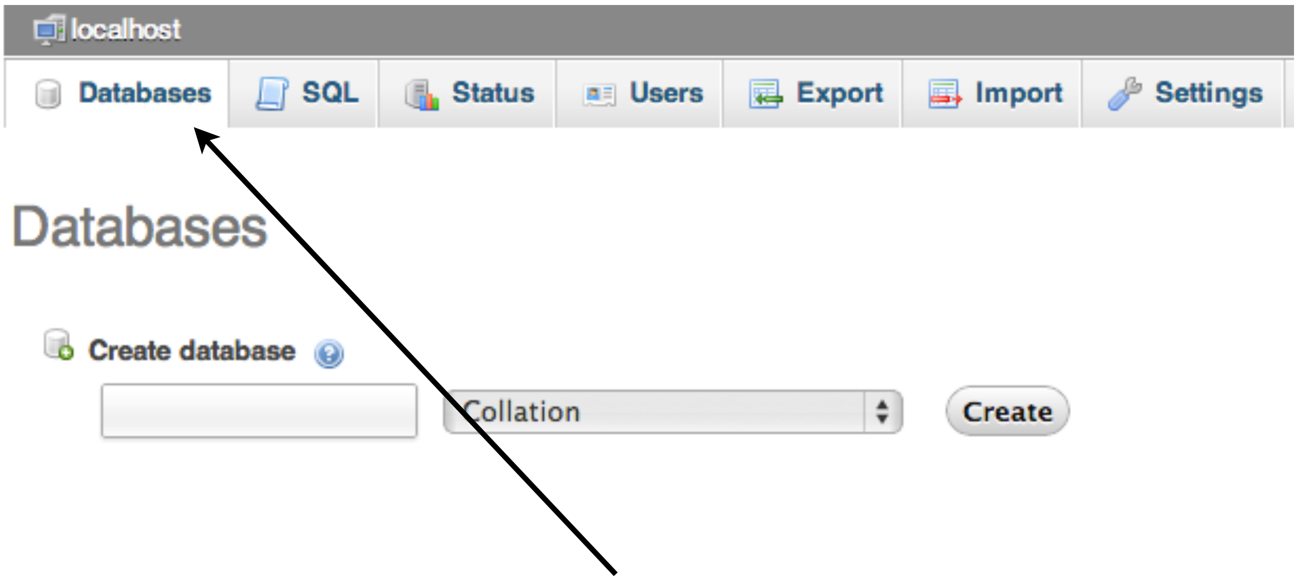
<u>Database</u>

We now need to setup and configure the database. To do this open up PHPMyAdmin or something equivalent like Sequel Pro (Mac) For the demo I will be using phpmyadmin as this is usually standard.

Step One – Login



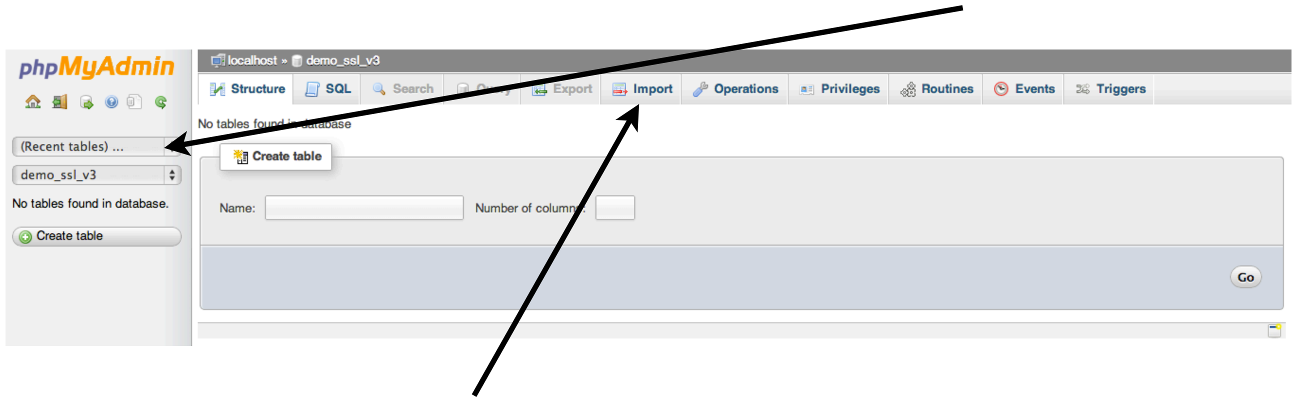
Locate your phpmyadmin installation and login, if you are unsure where to find this then please ask your host they will be able to point you in the right location.



Once logged in hit the database tab located in the top left of the phpmyadmin page.

Enter the name of the database this can be anything but make sure you remember what you enter. I'm going to call mine '*demo_ssl_v3*' When I have done this I can hit *create.*

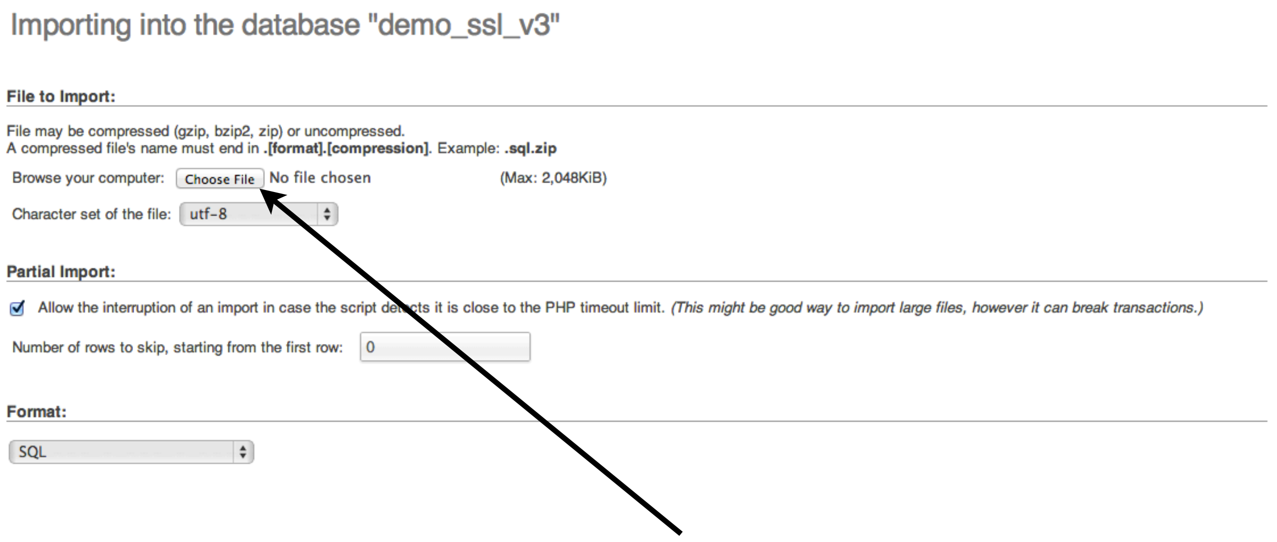
Next select the newly created database from the left hand side.



Then select the import tab located across the top of phpmyadmin.

Importing the SQL

Get ready to import the SQL file. Extract the contents of the SSLv3.4 zip and locate ‘*simple_secure_login_2013-02-03.sql*’ This is what we will be importing into phpmyadmin



Importing into the database "demo_ssl_v3"

File to Import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: Choose File No file chosen (Max: 2,048KiB)

Character set of the file: utf-8

Partial Import:

☑ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be good way to import large files, however it can break transactions.)*

Number of rows to skip, starting from the first row: 0

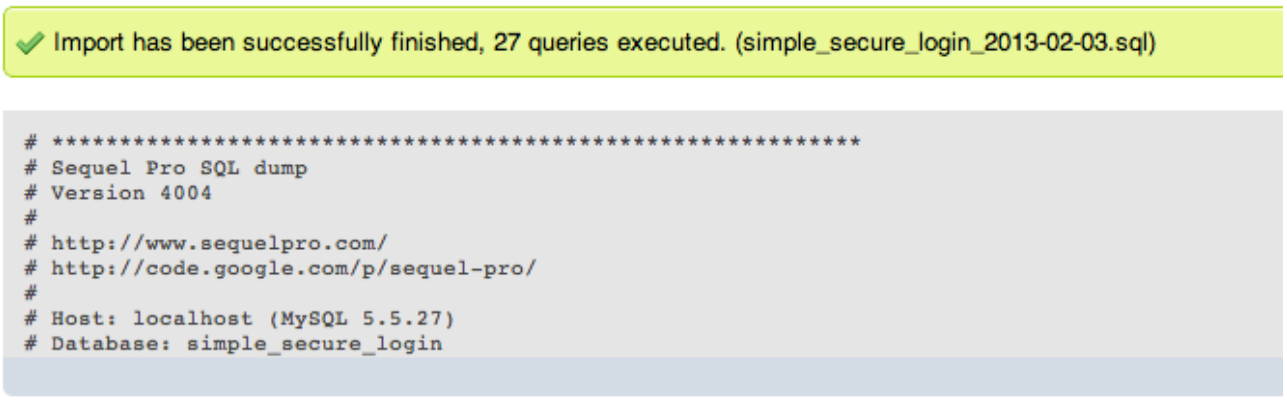
Format:

SQL

You will see a screen like above after selecting the import tab. Select the choose file button and import the SQL file we unzipped from the SSLv3.4 folder.

Hit **Go** located at the bottom of the page.

Once done you should see a success page like so:



✓ Import has been successfully finished, 27 queries executed. (simple_secure_login_2013-02-03.sql)

```
# ************************************************************
# Sequel Pro SQL dump
# Version 4004
#
# http://www.sequelpro.com/
# http://code.google.com/p/sequel-pro/
#
# Host: localhost (MySQL 5.5.27)
# Database: simple_secure_login
```

Hard part over =]

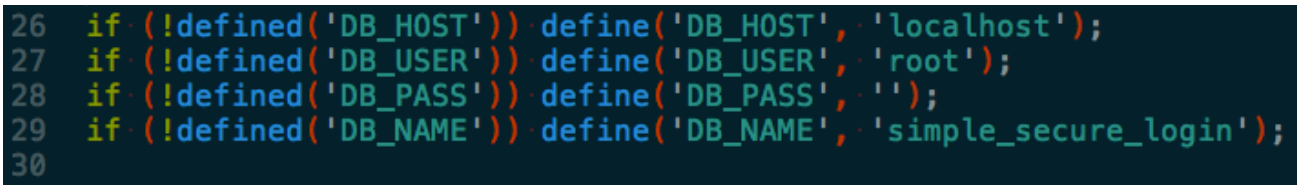
Configuration

Before we transfer to the server we need to make sure we can connect to the database.

Remember those credentials you used to access the database? You know the username and password you typed into phpmyadmin to login? Right well we need those credentials again...

We need to open up the *config.php* file. This is located in the ***inc*** folder.

Look for the following section of code located at line 26



```
26  if (!defined('DB_HOST')) define('DB_HOST', 'localhost');
27  if (!defined('DB_USER')) define('DB_USER', 'root');
28  if (!defined('DB_PASS')) define('DB_PASS', '');
29  if (!defined('DB_NAME')) define('DB_NAME', 'simple_secure_login');
30
```

Enter the credentials you used to connect to the database.

define('DB_USER', 'YOUR HOST'); *This is typically localhost if you are unsure contact your host and ask them for your database host.*

define('DB_USER', 'YOUR DATABASE USERNAME'); *The username you used to login to phpmyadmin.*

define('DB_PASS', 'YOUR DATABASE PASSWORD'); *The password you used to login to phpmyadmim*

define('DB_NAME', 'YOUR DATABASE NAME'); *The name we created in the database creation step I called mine 'demo_ssl_v3'*

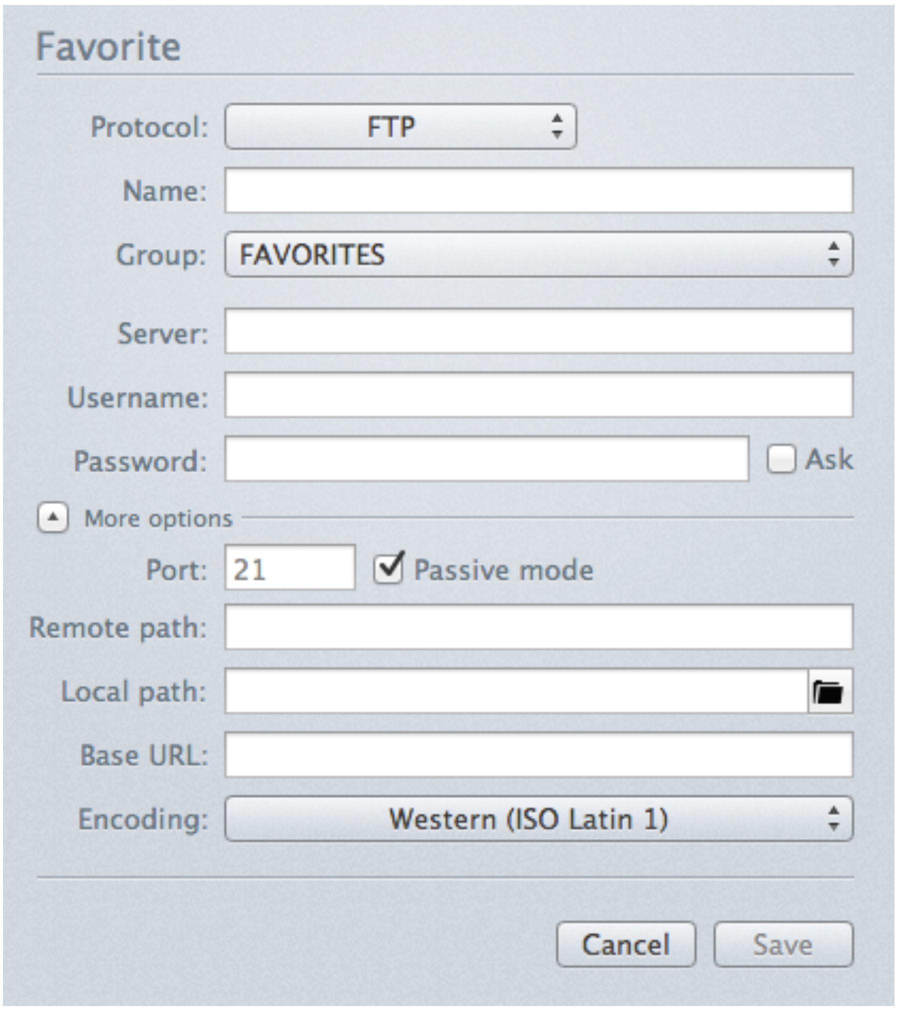
Little tip

If you are unsure about any of the information above be sure to contact your server host company people and ask them.

Uploading

To upload your application chose your favorite FTP client. I’m using Forklift for Mac. You can use an awesome free one that works on Mac, Linux and Windows it’s called Filezilla.

I’m not going to cover this step really because I’m going to assume you know how to do this.


Favorite

Protocol:	FTP
Name:	
Group:	FAVORITES
Server:	
Username:	
Password:	☐ Ask
Port:	21 ✓ Passive mode
Remote path:	
Local path:	
Base URL:	
Encoding:	Western (ISO Latin 1)

Save, connect and upload...

Using SSLv3.4

By default you have a user already in the database and their name is admin. **** SEE FOOTNOTE 1.0 && 1.1 ****

Username: **admin**
Password: **password**

Go ahead test it, head over to www.yoursite.com/sslv3.4/login.php

Login with the details above and if all went well and you followed my steps correctly you should be greeted with a new page.

Have a play about and get familiar with the control panel, add a few people and delete them.

Securing Pages

Simple. Secure. Login v3.4 makes it as easy as it's always been to secure pages. Simply add the following lines to any .php page.

```php
<?php
require_once 'bootstrap.php';
secure_page();
?>
```

If you are securing .php pages nested within folders make sure you require relatively the bootstrap.php file. For example:

```php
<?php
require_once '../../bootstrap.php'; // two folders deep
secure_page();
?>
```

This will allow any logged in user to view the page. If you would like to lock down a page to only allow administrators to view then use the following code.

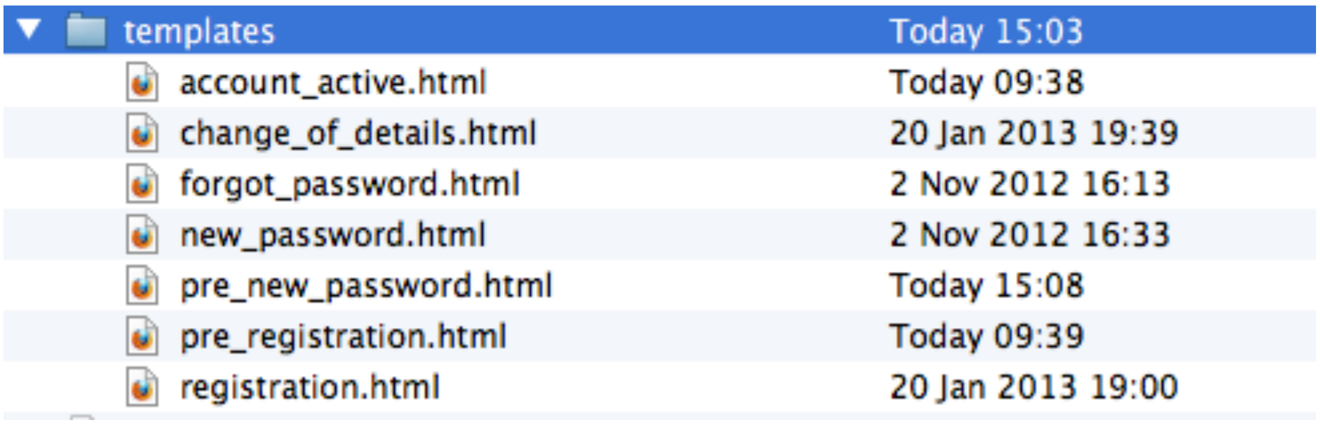
```
<?php
require_once 'bootstrap.php';
secure_page_admin();
?>
```

Remember! – *These pieces of code go at the very top of a .php file.*

Templates

Remember trying to modify the emails that get sent to users after certain tasks has been carried out? You had to go routing through the core.php file.

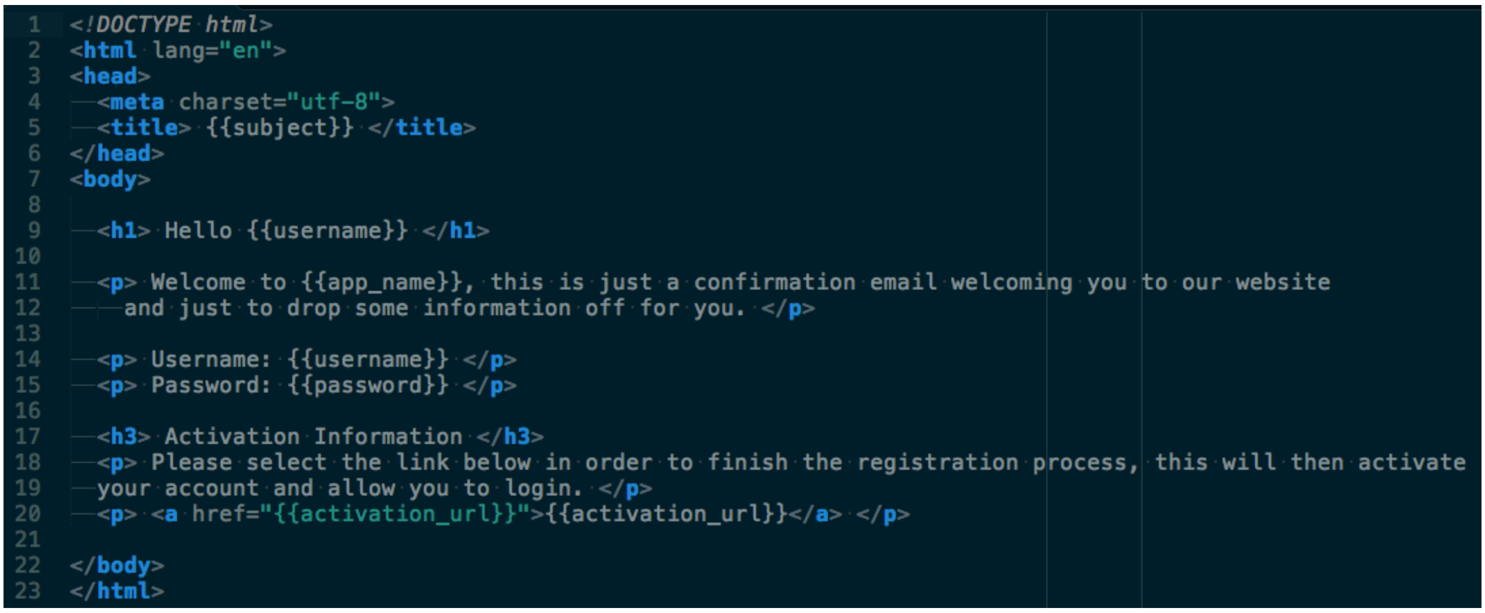
You don't have to anymore I have included a templates folder loaded with some HTML5 email templates for your to style.



templates	Today 15:03
account_active.html	Today 09:38
change_of_details.html	20 Jan 2013 19:39
forgot_password.html	2 Nov 2012 16:13
new_password.html	2 Nov 2012 16:33
pre_new_password.html	Today 15:08
pre_registration.html	Today 09:39
registration.html	20 Jan 2013 19:00

The names are self explanatory, but open these up in your favorite code editor and modify them to suite your needs.

Note the .html extension? Yes, they are dynamic open up the *registration.html* file and lets have a look.



```html
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<title> {{subject}} </title>
</head>
<body>

	<h1> Hello {{username}} </h1>

	<p> Welcome to {{app_name}}, this is just a confirmation email welcoming you to our website
		and just to drop some information off for you. </p>

	<p> Username: {{username}} </p>
	<p> Password: {{password}} </p>

	<h3> Activation Information </h3>
	<p> Please select the link below in order to finish the registration process, this will then activate
	your account and allow you to login. </p>
	<p> <a href="{{activation_url}}">{{activation_url}}</a> </p>

</body>
</html>
```

Note those weird placeholders like {{username}}? Whenever you see these in the default file is where you can replace the users data with.

For example in the *registration.html* file you have several placeholders these are:

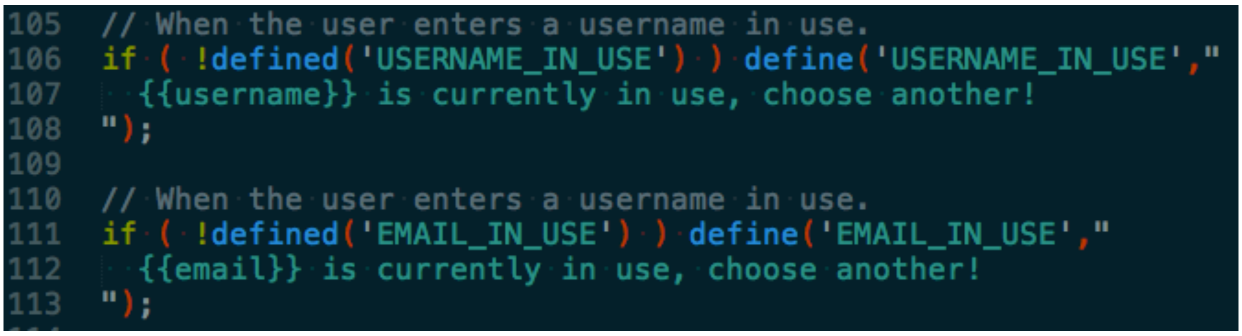
{{subject}}
{{username}}
{{password}}
{{activation_url}}

You can change the HTML to whatever you want and stick these placeholders anywhere you want. You can also use them more than once.

To style these template files either link to a stylesheet from the internet or add the styling to the top of the file.

Placeholders throughout SSLv3.4

You'll probably notice these weird placeholders throughout the SSLv3.4 application, for example open *lib/config.php*



```
105  // When the user enters a username in use.
106  if ( !defined('USERNAME_IN_USE') ) define('USERNAME_IN_USE',"
107    {{username}} is currently in use, choose another!
108  ");
109
110  // When the user enters a username in use.
111  if ( !defined('EMAIL_IN_USE') ) define('EMAIL_IN_USE',"
112    {{email}} is currently in use, choose another!
113  ");
```

Can you see {{username}} on line 107 and {{email}} on 112?

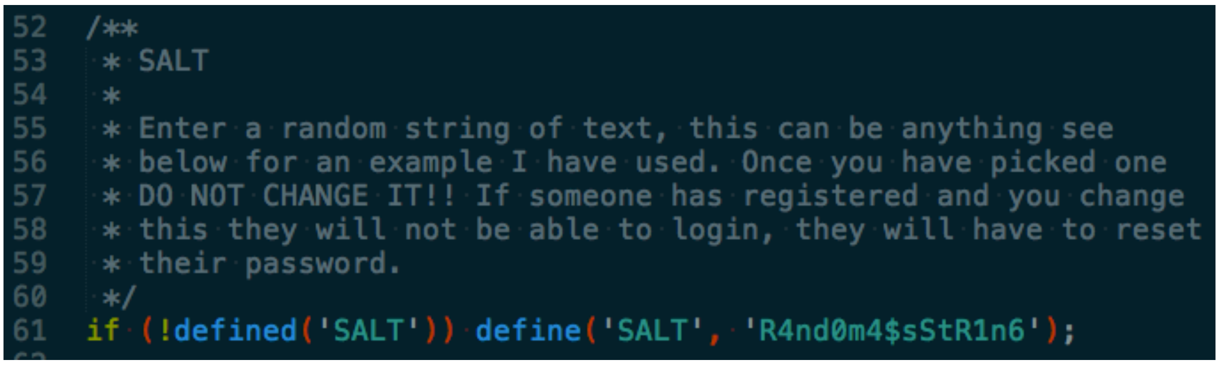
Well, you can change the text for the messages throughout the site whilst adding a little bit of personality and without the need to edit the core files.

For example you could change line 107 to read:
I'm sorry but {{username}} is currently in use... Choose another!

You get the idea right?

SALTS

We all like salt right? Well, SSLv3.4 now has password SALTS. To change your salt open up *lib/config.php* and find the following:



```
/**
 * SALT
 *
 * Enter a random string of text, this can be anything see
 * below for an example I have used. Once you have picked one
 * DO NOT CHANGE IT!! If someone has registered and you change
 * this they will not be able to login, they will have to reset
 * their password.
 */
if (!defined('SALT')) define('SALT', 'R4nd0m4$sStR1n6');
```

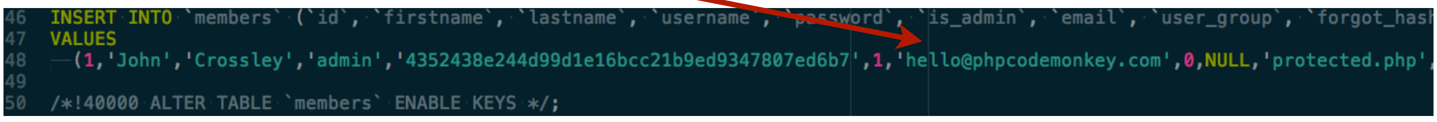
Enter a random string. **DO NOT CHANGE IT ONCE SET.** Change this before anyone even registers.

End

I think I have gotten everything for you. This is only the first revision of the SSLv3.4 documentation if there is something you feel I'm missing or don't explain well. Please let me know!

FOOTNOTE 1.0

Please open up the simple_secure_login_2013-02-03.sql file and edit the following.



```
INSERT INTO `members` (`id`, `firstname`, `lastname`, `username`, `password`, `is_admin`, `email`, `user_group`, `forgot_hash
VALUES
	(1,'John','Crossley','admin','4352438e244d99d1e16bcc21b9ed9347807ed6b7',1,'hello@phpcodemonkey.com',0,NULL,'protected.php',

/*!40000 ALTER TABLE `members` ENABLE KEYS */;
```

*Change the email from **hello@phpcodemonkey.com** to your email address.*

Continue with the importing of the SQL file.

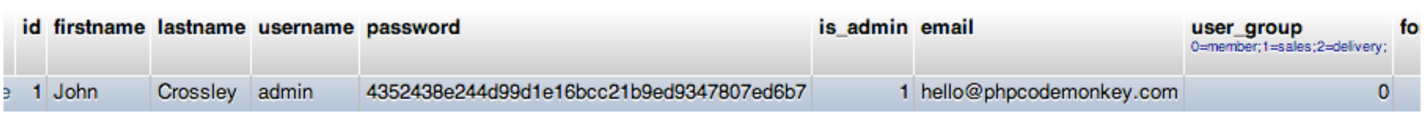
FOOTNOTE 1.1

Once you have change the email address and imported the file. Your login will not work if you have changed the default SALT.

This is because the password does not match the SALT. Just simply request a new password and everything should work great!

Troubleshooting

If you have tried the above and you are not receiving any emails it's all good we can still fix this. Open the admin up in phpmyadmin so you can see the following.

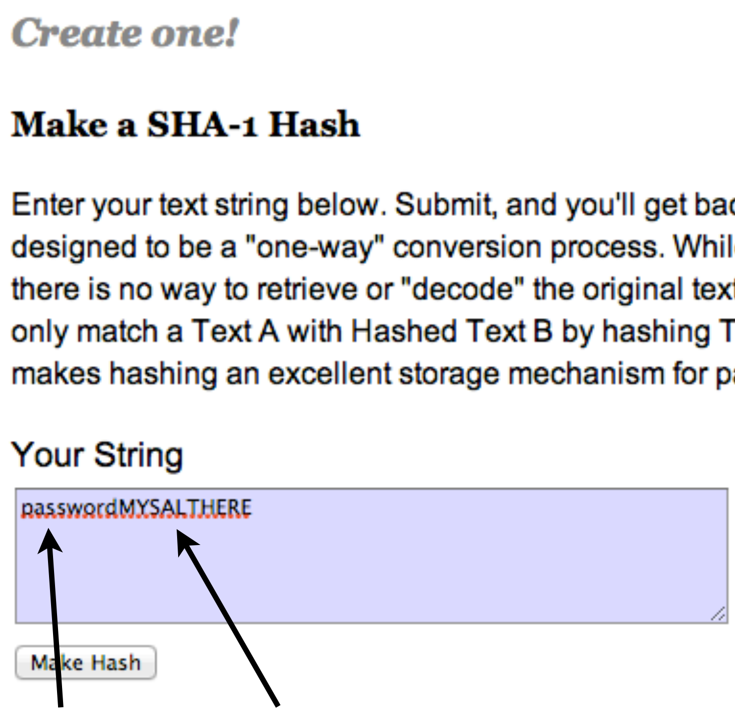


id	firstname	lastname	username	password	is_admin	email	user_group 0=member;1=sales;2=delivery;	fo
1	John	Crossley	admin	4352438e244d99d1e16bcc21b9ed9347807ed6b7	1	hello@phpcodemonkey.com	0	

Open the web browser and go to: http://ratfactor.com/sha1

Remember that SALT you chose...? Well we need it. We can change the password of the admin to ***password*** *(You choose what you like)*

Enter the information like so



Create one!

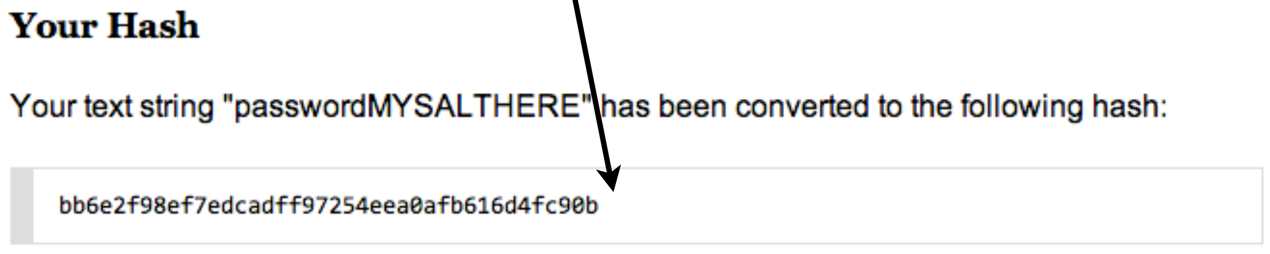
Make a SHA-1 Hash

Enter your text string below. Submit, and you'll get back a unique SHA-1 hash. Hashes are designed to be a "one-way" conversion process. While same-text "collisions" are possible, there is no way to retrieve or "decode" the original text string from a SHA-1 hash. You can only match a Text A with Hashed Text B by hashing Text A and comparing the hashes. This makes hashing an excellent storage mechanism for passwords.

Your String

passwordMYSALTHERE

password and **SALT** (NO SPACES) Then select Make Hash and you should see a weird string of text.



Your Hash

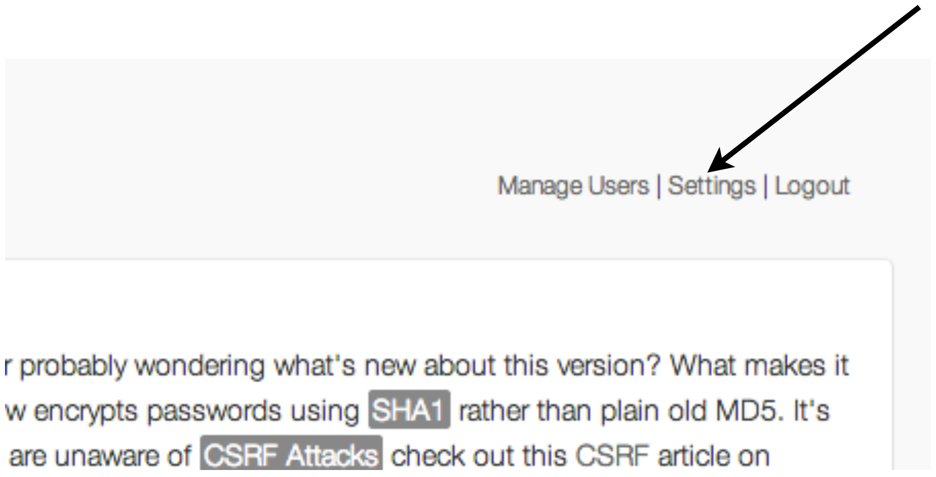
Your text string "passwordMYSALTHERE" has been converted to the following hash:

```
bb6e2f98ef7edcadff97254eea0afb616d4fc90b
```

Open up phpmyadmin and change the admin password with the new string we just generated.

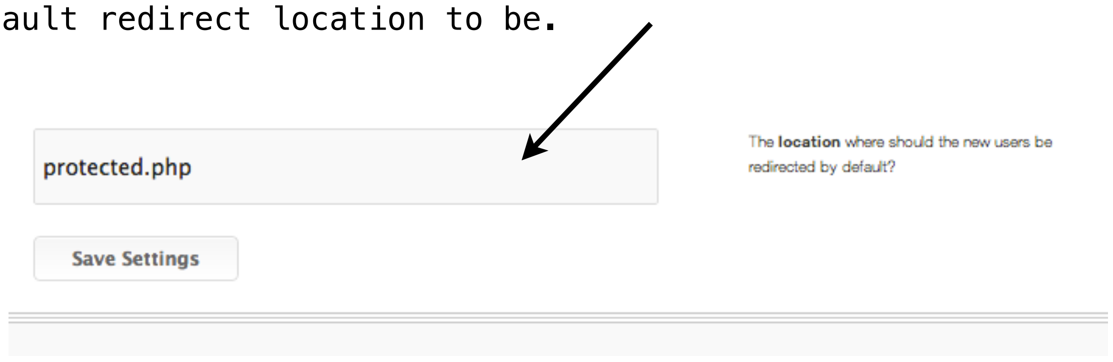
Changing the Default Redirect

If you'd like to set a default redirect location for your users, you can do this by navigating to the application settings.



r probably wondering what's new about this version? What makes it
w encrypts passwords using SHA1 rather than plain old MD5. It's
are unaware of CSRF Attacks check out this CSRF article on

Find the following field and change it to what ever you'd like the default redirect location to be.



protected.php

The **location** where should the new users be redirected by default?

Accessing your Users Data

When your users have logged in you can access their data like so:

```
<?php
require_once 'bootstrap.php';
$data = secure_page();
?>
```

Note how we have captured the secure_page() in the $data variable?

Doing this it returns the user once he/she is logged in. We can then do this.

```
<?php
// Add our generic block.
require_once 'bootstrap.php';
$data = secure_page();
?>
<!doctype html>
<html lang=”en”>
  <head>
     <meta charset=”utf-8”>
     <title> My Protected Page </title>
  </head>
  <body>
     <h1>Welcome, <?= $data->username ?></h1>
  </body>
</html>
```

Don’t copy this code, it’s been written using a word processor and will not work for your correctly. Write it out yourself! =]

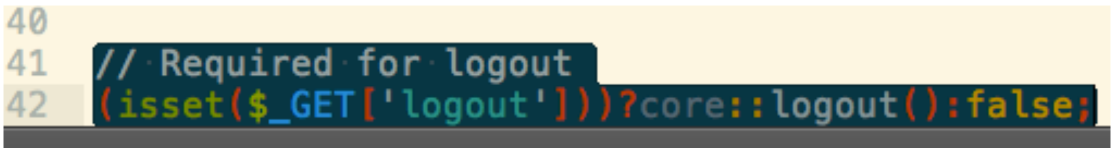
Here is a list of data that we have available about the user.

```
$data->id;
$data->firstname;
$data->lastname;
$data->username;
$data->is_admin;
$data->email;
$data->location; // Where are they redirected to?
$data->gravatar;
```

Note! The user must be logged in to access this data.

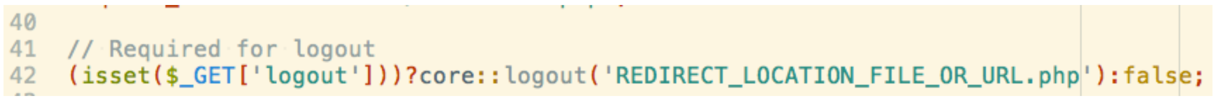
Change the Logout Redirect Location

Piece of PAI *(pie).* Open up the boostrap.php file located in the root of the application and fine line 42:



```
40
41 // Required for logout
42 (isset($_GET['logout']))?core::logout():false;
```

Modify it to look like this:



```
40
41 // Required for logout
42 (isset($_GET['logout']))?core::logout('REDIRECT_LOCATION_FILE_OR_URL.php'):false;
```

That's all folks...