tuts+

Advertisement

CODE > PHP

How to Implement Email Verification for New Members

by Philo Hermans 18 May 2009

Difficulty: Intermediate Length: Long Languages: English

PHP Web Development

Have you ever created an account with a website, and were required to check your email and click through a verification link sent by the company in order to activate it? Doing so highly reduces the number of spam accounts. In this lesson, we'll learn how to do this very thing!

Looking for a Shortcut?

This tutorial teaches you to build an email verification script from scratch, but if you want something that you can use on your website right away, check out Email Verify on Envato Market.

Email Verify is a PHP script that allows you to verify email addresses without storing anything in any databases. It will send users an email requiring them to click a link to verify their email before their email is added to whatever you want to add it to.

Email Verify on Envato Market

What Are We Going to Build?

We are going to build a nice PHP sign-up script where a user can create an account to gain access to a "members section" of a website.
After the user creates his account, the account will then be locked until the user clicks a verification link that he will receive in his email inbox.

Step 1 - Sign-up Page

We first need a simple page where our visitors can sign up their accounts; so that's the first thing we will build.
I would like to remind you that this is a PHP tutorial, and in my opinion, I think you need to know the basics of HTML before moving on with PHP. I'll add comments to the HTML & CSS to describe each line of code.

index.php - This is our sign up page with a basic form.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/T

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>NETTUTS > Sign up</title>
    <link href="css/style.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <!-- start header div -->
    <div id="header">
        <h3>NETTUTS > Sign up</h3>
    </div>
    <!-- end header div -->

    <!-- start wrap div -->
    <div id="wrap">

        <!-- start php code -->

        <!-- stop php code -->

        <!-- title and description -->
        <h3>Signup Form</h3>
        <p>Please enter your name and email addres to create your account</p>

        <!-- start sign up form -->
        <form action="" method="post">
            <label for="name">Name:</label>
            <input type="text" name="name" value="" />
            <label for="email">Email:</label>
            <input type="text" name="email" value="" />

            <input type="submit" class="submit_button" value="Sign up" />
        </form>
        <!-- end sign up form -->

    </div>
    <!-- end wrap div -->
</body>
</html>
```

css/style.css - This is stylesheet for index.php and further pages.

```
/* Global Styles */

*{
    padding: 0; /* Reset all padding to 0 */
    margin: 0; /* Reset all margin to 0 */
}

body{
    background: #F9F9F9; /* Set HTML background color */
    font: 14px "Lucida Grande";  /* Set global font size & family */
    color: #464646; /* Set global text color */
}

p{
    margin: 10px 0px 10px 0px; /* Add some padding to the top and bottom of t
}

/* Header */

#header{
    height: 45px; /* Set header height */
    background: #464646; /* Set header background color */
}

#header h3{
    color: #FFFFF3; /* Set header heading(top left title ) color */
    padding: 10px; /* Set padding, to center it within the header */
    font-weight: normal; /* Set font weight to normal, default it was set to
}

/* Wrap */

#wrap{
    background: #FFFFFF; /* Set content background to white */
    width: 615px; /* Set the width of our content area */
    margin: 0 auto; /* Center our content in our browser */
    margin-top: 50px; /* Margin top to make some space between the header and
    padding: 10px; /* Padding to make some more space for our text */
    border: 1px solid #DFDFDF; /* Small border for the finishing touch */
    text-align: center; /* Center our content text */
}

#wrap h3{
    font: italic 22px Georgia; /* Set font for our heading 2 that will be dis
}

/* Form & Input field styles */

form{
    margin-top: 10px; /* Make some more distance away from the description te
}

form .submit_button{
    background: #F9F9F9; /* Set button background */
    border: 1px solid #DFDFDF; /* Small border around our submit button */
    padding: 8px; /* Add some more space around our button text */
}

input{
    font: normal 16px Georgia; /* Set font for our input fields */
    border: 1px solid #DFDFDF; /* Small border around our input field */
    padding: 8px; /* Add some more space around our text */
}
```

As you can see, I have added a comment to each line that describes what they do. Also, you might have noticed the following comment in the index.php file:

```
<!-- start php code -->

<!-- stop php code -->
```

We are going to write our PHP between these 2 lines!

Step 2 - Input Validation

The first thing we are going to build is a piece of code that's going to validate the information. Here is a short list detailing what needs to be done.

- If the name field is not empty.
- If the name is not to short.
- If the email field is not empty.

- If the email address is valid xxx@xxx.xxx

So our first step is checking if the form is being submitted, and that the fields are not empty.

```
<!-- start PHP code -->
<?php

    if(isset($_POST['name']) && !empty($_POST['name']) AND isset($_POST['email
        // Form Submited
    }

?>
<!-- stop PHP Code -->
```

Time for a breakdown! We start of with an IF statement and we are first validating the name field:

```
if(  ){ // If statement is true run code between brackets

}

isset($_POST['name']) // Is the name field being posted; it does not matter w
&& // This is the same as the AND in our statement; it allows you to check mu
!empty($_POST['name']) // Verify if the field name is not empty

isset($_POST['email']) // Is the email field being posted; it does not matter
&& // This is the same as the AND in our statement; it allows you to check mu
!empty($_POST['email']) // Verify if the field email is not empty
```

So if you would submit the form now with empty fields, nothing happens. If you fill in both fields then our script will run the code between the brackets.
Now we are going to create a piece of code that will check if an email address is valid. If it's not, we will return a error. Also let's turn our post variables into local variables:

```
if(isset($_POST['name']) && !empty($_POST['name']) AND isset($_POST['email'])
    $name = mysql_escape_string($_POST['name']); // Turn our post into a local
    $email = mysql_escape_string($_POST['email']); // Turn our post into a loc
}
```

We can now reach our data via our local variables. As you can see, I also added a MySQL escape string to prevent MySQL injection when inserting the data into the MySQL database.

"The mysql_real_escape_string() function escapes special characters in a string for use in an SQL statement."

Regular Expressions

Next up is a small snippet that checks if the email address is valid.

```
$name = mysql_escape_string($_POST['name']);
$email = mysql_escape_string($_POST['email']);


if(!eregi("^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3}
    // Return Error - Invalid Email
}else{
    // Return Success - Valid Email
}
```

Please note that I did not personally write this regular expression, it's a small snippet from php.net.
Basically, it verifies if the email is written in the following format:

```
xxx@xxx.xxx
```

Now in the eregi, you can see that it checks if the email contains characters from the alphabet, if it has any numbers, or a phantom dash (_), and of course the basic requirements for an email (email)'@' and a (dot)'.' If none of these characters are found, the expression returns "false". Okay, so now we need to add some basic error messages.

```
if(!eregi("^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3}
    // Return Error - Invalid Email
    $msg = 'The email you have entered is invalid, please try again.';
}else{
    // Return Success - Valid Email
    $msg = 'Your account has been made, <br /> please verify it by clicking th
}
```

As you can see we have made a local variable "$msg", this allows us to show the error or the success message anywhere on the page.
And we are going to display it between the instruction text and the form.

```
<!-- title and description -->
<h3>Signup Form</h3>
<p>Please enter your name and email address to create your account</p>

<?php
    if(isset($msg)){  // Check if $msg is not empty
        echo '<div class="statusmsg">'.$msg.'</div>'; // Display our message
    }
?>

<!-- start sign up form -->
```

Add this to **style.css**, to style our status message a bit.

```
#wrap .statusmsg{
    font-size: 12px; /* Set message font size  */
    padding: 3px; /* Some padding to make some more space for our text  */
    background: #EDEDED; /* Add a background color to our status message   */
    border: 1px solid #DFDFDF; /* Add a border arround our status message   */
}
```

Step 3 - Creating the Database & Establishing a Connection

Now we need to establish a database connection and create a table to insert the account data. So let's go to PHPMyAdmin and create a new database with the name **registrations** and create a user account that has access to that database in order to insert and update data.

Let's create our **users** table, with 5 fields:

So now we must enter details for these fields:

For those who don't want to input this data manually, you can instead run the following SQL code.

```
CREATE TABLE `users` (
`id` INT( 10 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`username` VARCHAR( 32 ) NOT NULL ,
`password` VARCHAR( 32 ) NOT NULL ,
`email` TEXT NOT NULL ,
`hash` VARCHAR( 32 ) NOT NULL ,
`active` INT( 1 ) NOT NULL DEFAULT '0'
) ENGINE = MYISAM ;
```

Our database is created, now we need to establish a connection using PHP. We'll write the following code at the start of our script just below the following line:

```
<!-- start PHP code -->
<?php
// Establish database connection
```

We'll use the following code to connect to the database server and select the **registrations** database. (basic MySQL connection)

```
mysql_connect("localhost", "username", "password") or die(mysql_error()); // C
mysql_select_db("registrations") or die(mysql_error()); // Select registration
```

Now that we've established a connection to our database, we can move on to the next step and insert the account details.

Step 4 - Insert Account

Now it's time to enter the submitted account details to our database and generate an activation hash. Write the following code below this line:

```
// Return Success - Valid Email
$msg = 'Your account has been made, <br /> please verify it by clicking the ac
```

Activation Hash

In our database we made a field called hash, this hash is a 32 character string of text. We also send this code to the user's email address. They then can click the link (which contains the hash) and we will verify if it matches up with the one in the database. Let's create a local variable called $hash and generate a random md5 hash.

```
$hash = md5( rand(0,1000) ); // Generate random 32 character hash and assign i
// Example output: f4552671f8909587cf485ea990207f3b
```

What did we do? Well we are using the PHP function "rand" to generate a random number between 0 and 1000. Next our MD5 function will turn this number into a 32 character string of text which we will use in our activation email. My choice is to use MD5, because it generates a hash of 32 characters which is secure and, in this case, impossible to crack.

Creating a Random Password

The next thing we need to is to create a random password for our member:

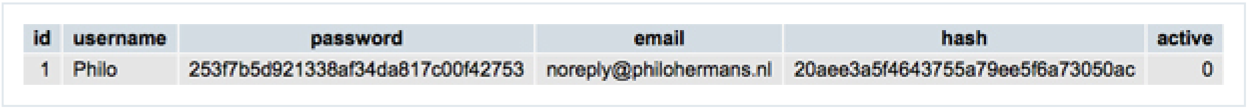
```
$password = rand(1000,5000); // Generate random number between 1000 and 5000 a
// Example output: 4568
```

Insert the following information into our database using a MySQL query

```
mysql_query("INSERT INTO users (username, password, email, hash) VALUES(
'". mysql_escape_string($name) ."',
'". mysql_escape_string(md5($password)) ."',
'". mysql_escape_string($email) ."',
'". mysql_escape_string($hash) ."') ") or die(mysql_error());
```

As you can see, we insert all data with a MySQL escape string around it to prevent any MySQL injection.
You also might notice that the MD5 function changes the random password into a secure hash for protection. Example: if an "evil" person gains access to the database, he can't read the passwords.

For testing, fill in the form and check if the data is being inserted into our database.



id	username	password	email	hash	active
1	Philo	253f7b5d921338af34da817c00f42753	noreply@philohermans.nl	20aee3a5f4643755a79ee5f6a73050ac	0

Step 5 - Send the Verification Email

Right after we have inserted the information into our database, we need to send an email to the user with the verification link. So let's use the PHP "mail" function to do just that.

```
$to      = $email; // Send email to our user
$subject = 'Signup | Verification'; // Give the email a subject
$message = '

Thanks for signing up!
Your account has been created, you can login with the following credentials a

------------------------
Username: '.$name.'
Password: '.$password.'
------------------------

Please click this link to activate your account:
http://www.yourwebsite.com/verify.php?email='.$email.'&hash='.$hash.'

'; // Our message above including the link

$headers = 'From:noreply@yourwebsite.com' . "\r\n"; // Set from headers
mail($to, $subject, $message, $headers); // Send our email
```

Now let's brake down the message:

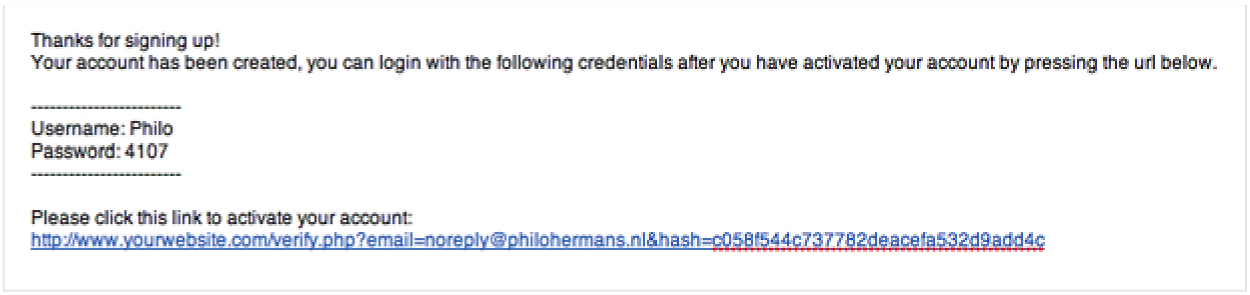
```
Thanks for signing up!
Your account has been created, you can login with the following credentials af

------------------------
Username: '.$name.'
Password: '.$password.'
------------------------
```

In the code above we send a short description to our user which contains the username and password -- using the local variables we created when the data was posted.

```
Please click this link to activate your account:
http://www.yourwebsite.com/verify.php?email='.$email.'&hash='.$hash.'
```

In this section of the code, we created a dynamic link. The result of this will look like this:


Thanks for signing up!
Your account has been created, you can login with the following credentials after you have activated your account by pressing the url below.

Username: Philo
Password: 4107

Please click this link to activate your account:
http://www.yourwebsite.com/verify.php?email=noreply@philohermans.nl&hash=c058f544c737782deacefa532d9add4c

As you can see, it creates a solid url, which is impossible to guess. This is a very secure way to verify the email address of a user.

Step 6 - Account Activation

As you can see, our url links to **verify.php** so let's create that, using the same basic template we used for index.php.
However, remove the form from the template.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/T

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>NETTUTS > Sign up</title>
    <link href="css/style.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <!-- start header div -->
    <div id="header">
        <h3>NETTUTS > Sign up</h3>
    </div>
    <!-- end header div -->

    <!-- start wrap div -->
    <div id="wrap">
        <!-- start PHP code -->
        <?php

            mysql_connect("localhost", "tutorial", "password") or die(mysql_e
            mysql_select_db("registrations") or die(mysql_error()); // Select

        ?>
        <!-- stop PHP Code -->


    </div>
    <!-- end wrap div -->
</body>
</html>
```

The first thing we need to do is check if we have our $_GET variables (email & hash)

```
if(isset($_GET['email']) && !empty($_GET['email']) AND isset($_GET['hash']) &&
    // Verify data
}else{
    // Invalid approach
}
```

To make things a bit easier, let's assign our local variables and add some MySQL injection prevention by, once again, using the MySQL escape string.

```
if(isset($_GET['email']) && !empty($_GET['email']) AND isset($_GET['hash']) &&
    // Verify data
    $email = mysql_escape_string($_GET['email']); // Set email variable
    $hash = mysql_escape_string($_GET['hash']); // Set hash variable
}
```

Next is to check the data from the url against the data in our database using a MySQL query.

```
$search = mysql_query("SELECT email, hash, active FROM users WHERE email='".$e
$match  = mysql_num_rows($search);
```

In the code above, we used a MySQL select statement, and checked if the email and hash matched. But beside that, we checked if the status of the account is "inactive". Finally, we use mysql_num_rows to determine how many matches have been found. So let's try this out. Simply use a simple echo to return the results.

```
$search = mysql_query("SELECT email, hash, active FROM users WHERE email='".$e
$match  = mysql_num_rows($search);

echo $match; // Display how many matches have been found -> remove this when d
```

We have a **MATCH**! To change the result, simply change the email and you'll see that the number returned is **0**.
We can use our **$match** variable to either activate the account or return a error when no match has been found.

```
if($match > 0){
    // We have a match, activate the account
}else{
    // No match -> invalid url or account has already been activated.
}
```

In order to activate the account, we must update the **active** field to 1 using a MySQL query.

```
// We have a match, activate the account
mysql_query("UPDATE users SET active='1' WHERE email='".$email."' AND hash='".
echo '<div class="statusmsg">Your account has been activated, you can now logi
```

So we use the same search terms for the update as we used in our MySQL select query. We change active to 1 where the email, hash and field active = 0 match up. We also return a message telling the user that his account has been activated. You can add a message like we did here to the "no match" part. So the final code should look similar to:

```
mysql_connect("localhost", "tutorial", "password") or die(mysql_error()); //
mysql_select_db("registrations") or die(mysql_error()); // Select registratio

if(isset($_GET['email']) && !empty($_GET['email']) AND isset($_GET['hash']) &
    // Verify data
    $email = mysql_escape_string($_GET['email']); // Set email variable
    $hash = mysql_escape_string($_GET['hash']); // Set hash variable

    $search = mysql_query("SELECT email, hash, active FROM users WHERE email=
    $match  = mysql_num_rows($search);

    if($match > 0){
        // We have a match, activate the account
        mysql_query("UPDATE users SET active='1' WHERE email='".$email."' AND
        echo '<div class="statusmsg">Your account has been activated, you can
    }else{
        // No match -> invalid url or account has already been activated.
        echo '<div class="statusmsg">The url is either invalid or you already
    }

}else{
    // Invalid approach
    echo '<div class="statusmsg">Invalid approach, please use the link that h
}
```

If you visit **verify.php** without any strings, the follow error will be shown:

Advertisement

Step 7 - Login

In this final step, I will show you how to create a basic login form and check if the account is activated. First create a new file called **login.php** with the basic template we used before, but this time I changed the form into a login form.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/T

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>NETTUTS > Sign up</title>
    <link href="css/style.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <!-- start header div -->
    <div id="header">
        <h3>NETTUTS > Sign up</h3>
    </div>
    <!-- end header div -->

    <!-- start wrap div -->
    <div id="wrap">
        <!-- start PHP code -->
        <?php

            mysql_connect("localhost", "tutorial", "password") or die(mysql_e
            mysql_select_db("registrations") or die(mysql_error()); // Select


        ?>
        <!-- stop PHP Code -->

        <!-- title and description -->
        <h3>Login Form</h3>
        <p>Please enter your name and password to login</p>

        <?php
            if(isset($msg)){ // Check if $msg is not empty
                echo '<div class="statusmsg">'.$msg.'</div>'; // Display our
            } ?>

        <!-- start sign up form -->
        <form action="" method="post">
            <label for="name">Name:</label>
            <input type="text" name="name" value="" />
            <label for="password">Password:</label>
            <input type="password" name="password" value="" />

            <input type="submit" class="submit_button" value="Sign up" />
        </form>
        <!-- end sign up form -->

    </div>
    <!-- end wrap div -->
</body>
</html>
```

The form is basic html, and almost the same as the signup form, no further explanation is needed. Now it's time to write the code for the login script, which we will write just below the MySQL connection code. We start with something we also did in the signup form.

```
if(isset($_POST['name']) && !empty($_POST['name']) AND isset($_POST['password'
    // Both fields are being posted and there not empty
}
```

So we first check to see if the data is being posted, and we make sure that it's not empty.
Next is to create some local variables for the post data:

```
if(isset($_POST['name']) && !empty($_POST['name']) AND isset($_POST['password'
    $username = mysql_escape_string($_POST['name']); // Set variable for the u
    $password = mysql_escape_string(md5($_POST['password'])); // Set variable
}
```

We created the local variables and changed the password into a md5 hash to match it with the password hash we have stored in the database.
Now, it's time to create the connection to our "users" table and verify if the entered data is correct.

```
if(isset($_POST['name']) && !empty($_POST['name']) AND isset($_POST['password'
    $username = mysql_escape_string($_POST['name']);
    $password = mysql_escape_string(md5($_POST['password']));

    $search = mysql_query("SELECT username, password, active FROM users WHERE
    $match  = mysql_num_rows($search);
            }
```

We wrote a MySQL query that will select the username, password and active information from our database, if the username and password match up.
AND active='1' is !IMPORTANT!, this makes sure that you can only login if your account is activated. We use the MySQL num rows again to see how many matches are found.

```
if($match > 0){
    $msg = 'Login Complete! Thanks';
    // Set cookie / Start Session / Start Download etc...
}else{
    $msg = 'Login Failed! Please make sure that you enter the correct details
}
```

In the code above we check if the login was a success or not.

The End

And that's the end of this tutorial! I hope you enjoyed it, and if you did please leave a comment below!

Follow us on Twitter, or subscribe to the NETTUTS RSS Feed for more daily web development tuts and articles.

Advertisement

B53a64c23546ffad133333ace1ca4074?
s=200&d=https%3a%2f%2fassets.tutsplus.com%2fimages%2fhub%2favatar
default

Philo Hermans

Philo Hermans is a freelance web developer from Utrecht, The Netherlands. He has over 9 years of experience and often works with designers who want to see their designs turned into fully coded websites or web applications. Philo is also a successful plugin developer on CodeCanyon, and has written frequently at Nettuts+. Find out more about him at his website.

FEED LIKE FOLLOW FOLLOW

Weekly email summary

Subscribe below and we'll send you a weekly email summary of all new Code tutorials. Never miss out on learning about the next big thing.

Email Address

Update me weekly

Advertisement

Translations

Envato Tuts+ tutorials are translated into other languages by our community members—you can be involved too!

Translate this post

Powered by native

Advertisement

QUICK LINKS - Explore popular categories

ENVATO TUTS+ +

JOIN OUR COMMUNITY +

HELP +

tuts+

26,782	1,187	32,362
Tutorials	Courses	Translations

Envato.com Our products Careers Sitemap

Follow Envato Tuts+