Send email with php using PHPMailer

Monday, 17 July 2006
Last Updated Friday, 28 July 2006

Many PHP developers utilize email in their code. The only PHP function that supports this is the mail() function. However, it does not expose any of the popular features that many email clients use nowadays like HTML-based emails and attachments.

PHPMailer

PHPMailer is a PHP-Class for PHP providing a package of functions to send emails. Main Features are HTML-Mail and Emails with attachments. For mailing, PHPMailer supports nearly all ways to send emails: mail(), sendmail, qmail & smtp-server directly. You can use any feature of smtp based emails: multiple to's, cc's, bcc's etc.

Before continuing, ensure the PHPMailer class is installed correctly. You can check for the proper installation with this little script:

```
<?php

require("class.phpmailer.php");
$mail = new PHPMailer();

?>
```

If PHPMailer is not installed and errors are enabled in your server you would receive the appropriate error on executing the above-mentioned script.

The code shown above enables you to start working with PHPMailer. The require("class.phpmailer.php"); instantiates the class and $mail = new PHPMailer(); instances the class to $mail. It means, you can access all features, functions and methods of the class via the variable (object) $mail.

Let’s send out the first mail. For this you need:

• An email address where the mail is to be sent.
• The email address from where the mail has to be sent.
• A subject.
• The text of the mail, called mail-body or only body in short.

We will use SMTP in our forthcoming example to send an email. Although, you can use other methods such as sendmail or qmail to send emails. For SMTP, you need a valid smtp server. Ensure that your email server is SMTP enabled.

Now, let’s implement a full-fledged script to send email.

```
<?php

require("class.phpmailer.php");
$mail = new PHPMailer();
$mail->IsSMTP(); // telling the class to use SMTP
$mail->Host = "smtp.email.com"; // SMTP server
$mail->From = "from at email.com";
$mail->AddAddress("myfriend at site.com");
$mail->Subject = "first mail";
$mail->Body = "hi ! \n\n this is my first PHPMailer email !";
$mail->WordWrap = 50;

if(!$mail->Send())
{
   echo "Message was not sent";
   echo "Mailer Error: " . $mail->ErrorInfo;
} else {
   echo "Message has been sent";
}

?>
```

Save the above code in a php file and change the values to your requirements.

Now let’s look in detail the working of the above mentioned script:

$mail->IsSMTP();

This sets up STMP-Server as the desired method to send out email(s).

$mail->Host = "smtp.email.com";

Just replace smtp.email.com with your own SMTP-Server address. You can even specify more than one SMTP-Servers by separating them with a semicolon (;):

"smtp.email.com;smtp2.email.com"

If the first one fails, the second will be taken. This makes sense, if you know the mail server is sometimes down.

Next, put the address from where the email should be sent. Take any address that is valid for the SMTP-Server. The person who receives the email will know who has sent the email.

$mail->From = "from@email.com";

You can add another line of code to give the address a name. This will add 'Your Name' to the ‘from-address’, so the receiver will come to know, who has sent the mail.

$mail->FromName = "Your Name";

The following will add the ‘to-address’, the address where the email is sent. Ensure that this is a valid email address, so that you can check that your script worked. I would suggest you to take your own address. You can also add a name to the email. This is done in a different way as compared to the ‘from address’ as seen earlier:

$mail->AddAddress("myfriend@site.com","Friends name");
$mail->AddAddress("myfriend@site.com");

The next part is again very simple. Setting the subject and body is done there.

$mail->WordWrap = 50;

This feature of PHPMailer is used to word-wrap your message automatically.

This is followed by sending the email.

$return = $mail->Send();

This is combined with an error message. If Send() fails, it'll return false and you can catch it and display an error message. In case of success, it displays a kind of 'done' message. Hope your script works, if not, check out all syntax and logic errors.

Using attachments

There are two ways to send attachments along with your email: First, you can simply attach a file from the file system (remote won't work) and Second, you can attach (binary) data stored in a String variable, called Stringattachment.

File Attachments

In your code, a new command to attach a file can be placed anywhere between $mail = new PHPMailer(); and !$mail->Send(); and it's called

AddAttachment($path);

This single line will add the Attachment to your email $path is the path of the filename. It can be a relative one (from your script, not the PHPMailer class) or a full path to the file you want to attach.

If you want more options or you want to specify encoding and mime/type of the file, then you can use three more parameters, which are all optional:

AddAttachment($path,$name,$encoding,$type);

$name is an optional parameter for you to set the filename, which will occur in the mail you'll send out. The people who will receive your email will then only see this name instead of the original filename.

$encoding is a little more technical, with this parameter you can set the type of encoding of the attachment. Default is base64. Other types that are supported are: 7bit, 8bit, binary & quoted-printable.

$type is the mime-type of your attached file. On internet, you don't specify the file type with an ending dot and suffix, a so-called mime-type (MIME = Multipurpose Internet Mail Extensions) is used. This parameter gives you the possibility to change it from the default value of application/octet-stream (which works with every kind of file) to a more specific mime type like image/jpeg for a .jpg-photo for example.

String Attachments

String attachments have been supported since PHPMailer version 1.29. This method works quite like AddAttachment() and is called

AddStringAttachment($string,$filename,$encoding,$type)

Naturally, you have to pass your string data to the method and that's the first parameter $string. Because the string will lead into a standard file (that's what an attachment will be if you receive it via email), the $filename parameter is not optional now. Use it to give your data a filename.

The rest is just the same as described in detail above.

So, why should AddStringAttachment be used instead of AddAttachment? Is it for Text-Only files? - No, not at all. It's for Databases for example. Data stored in a Database is always stored as a string (often called blob). You can for example query your database and pass the returned string to the AddStringAttachment. Therefore you use it.
Inline Attachments

There is an additional way to add an attachment. If you want to make a HTML-message with Images, you have to make an attachment of the image and then link the tag to it. This is done with a so-called CID. For example, you add an image as Inline Attachment with the CID my-photo, you access it within the HTML-mail Part with

&ltimg src="cid:my-photo" alt="my photo" />

Here is the function to add the Inline Attachment in detail:

$mail->AddEmbeddedImage(filename, cid, name));

By using this function with this example's value above, leads into this code:

$mail->AddEmbeddedImage('my-photo.jpg', 'my-photo', 'my-photo.jpg '));

Handling Attachments

If you want to attach multiple files (or strings), just call AddAttachment() or AddStringAttachment() multiple times. It’s not possible to remove a particular attachment, but you can remove all attachments from your email by calling ClearAttachments(). This will remove all Attachments, this means all File-, String-, and Inline-attachments.

Using HTML Mail

Sending HTML emails depends on the receiving person's email client, which should be compatible with these HTML-mails. For those mailclients that are not able to display HTML formatting's, you can use an alternative (old+ alternate ??) body containing your message as plain text.

First we'll simply create a basic HTML message:

```
<?php

require("class.phpmailer.php");

$mail = new PHPMailer();
$mail->IsSMTP(); // telling the class to use SMTP
$mail->Host = "smtp.email.com"; // SMTP server
```

```
$mail->From = "from at email.com";
$mail->AddAddress("myfriend at site.com");
$mail->Subject = "look, it's a HTML message";
$mail->Body = "hi my friend! \n\n this message uses html entities !";

?>
```

Before sending this out, we have to tell the PHPMailer object, that we want to use a HTML message. In detail this means, the body has to be setup as multipart/alternative. The is done with:

```
$mail->IsHTML(true);
```

To make the HTML-message perfect, you can use $mail->AltBody="hi my friend! \n\n this message uses html entities, but you prefer plain text !"; to set the alternative (old+ alternate ??) body (AltBody in short). If you use this feature, the email will be automatically set as multipart/alternative and you do not need to use $mail->IsHTML(true); any more.

In this way you can send out HTML-messages. Just apply the send-command after it and the email is out. I would not recommend using HTML messages, because these messages are often used for viruses and exploits, they are bigger than simple plain-text emails and are not standardized in the way the plain-text email is. On the other hand, even attachments are used for viruses, but nobody would stop sending emails with attachments. It's your decision; PHPMailer is a solution for both, Emails with HTML or without HTML.