Lab 11: Sending an Email Message

There are two common methods to send an email message in PHP, i.e. using the server's built-in mail server and using an external mail server.

In this exercise, we will look at the implementation of these methods using a different technique in each method.

11.1 Using a Simple Technique

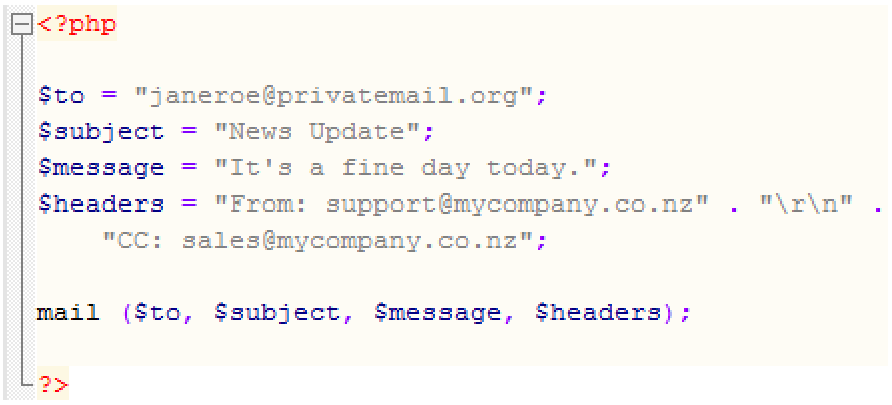
The simplest way to send an email in PHP is to use *mail ( )* function.

Format : mail (<to>, <subject>, <message>, [<headers>]);

Parameter	**Description**
to	Required. Specifies the receiver / receivers of the email.
subject	Required. Specifies the subject of the email. **Note:** This parameter cannot contain any newline characters.
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters.
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). **Note:** When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.

Figure 1 below shows an example of sending an email using *mail ( )* function. Note that this technique does not specify the outgoing mail server (and its associated authentication), hence, relies on the server's settings or php.ini file.

You may run into a difficulty in using this technique when the server does not allow sending of email, or when its settings were incomplete or incorrect. This is a likely situation when using a hosted web server.



```
<?php

$to = "janeroe@privatemail.org";
$subject = "News Update";
$message = "It's a fine day today.";
$headers = "From: support@mycompany.co.nz" . "\r\n" .
    "CC: sales@mycompany.co.nz";

mail ($to, $subject, $message, $headers);

?>
```

Figure 1: Sample code to send an email using *mail ( )* function.

It returns *TRUE* if the mail was successfully accepted for delivery and *FALSE* otherwise. Note that although the mail is accepted for delivery, it does not mean it will actually reach the intended destination.

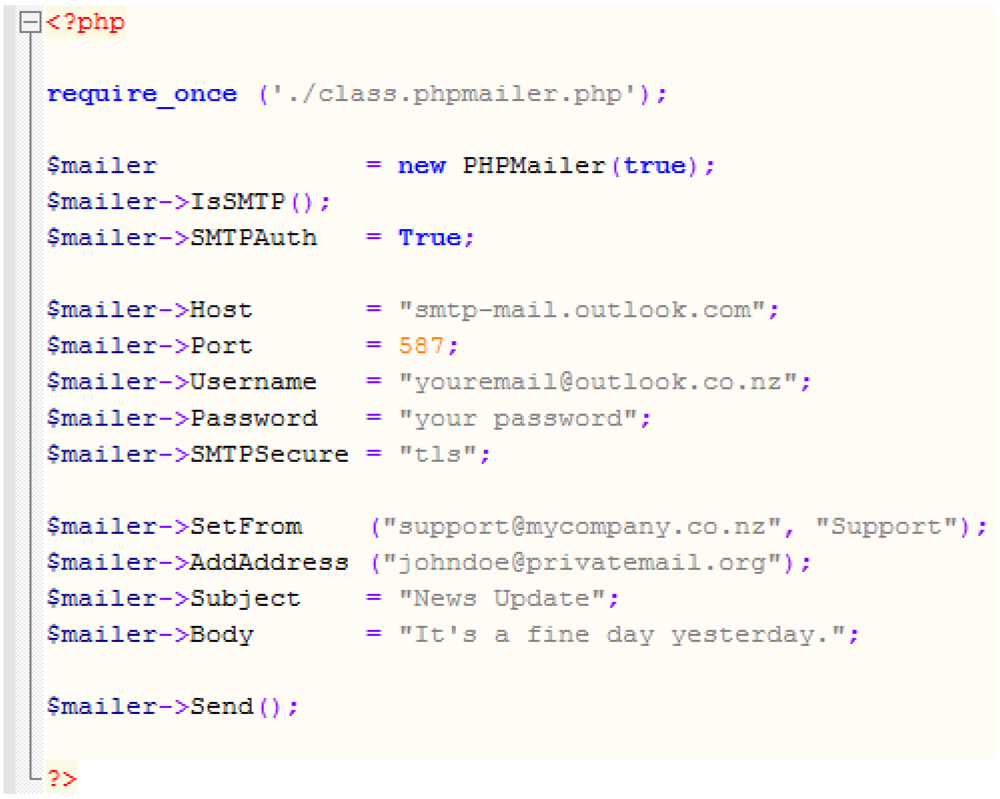
11.2 Using a Custom Technique

This custom technique uses an external mail server to relay the email message to the destination/recipients; hence, you will need an existing email account where its server supports access to its SMTP feature. This could be a corporate or a fee-free mail server.

This is a recommended technique for a hosted environment situation, or situation where a PHP programmer has no control over the server and its settings.

Figure 2 below shows an example of sending an email using *PHPMailer*. Note that it requires an additional file *class.phpmailer.php*.

Download this file from https://github.com/PHPMailer/PHPMailer/blob/master/class.phpmailer.php and include it in your web folder.



```php
<?php

require_once ('./class.phpmailer.php');

$mailer              = new PHPMailer(true);
$mailer->IsSMTP();
$mailer->SMTPAuth    = True;

$mailer->Host        = "smtp-mail.outlook.com";
$mailer->Port        = 587;
$mailer->Username    = "youremail@outlook.co.nz";
$mailer->Password    = "your password";
$mailer->SMTPSecure = "tls";

$mailer->SetFrom    ("support@mycompany.co.nz", "Support");
$mailer->AddAddress ("johndoe@privatemail.org");
$mailer->Subject     = "News Update";
$mailer->Body        = "It's a fine day yesterday.";

$mailer->Send();

?>
```

Figure 2: Sample code to send an email using *PHPMailer*.

Your tasks:

1. Write an HTML page that allow the user to submit a feedback.
2. Send the feedback via email to your email address, capturing the following information:
    a. Name
    b. Address
    c. Phone number
    d. Email address
    e. Feedback subject
    f. Feedback message
    g. IP address
    h. Time stamp
3. Auto-reply to the user with an acknowledgement email.

-- END --