69/B Monwara Plaza (Level-5)
East Panthapath
Dhaka-1205
Cell: 01197 11 82 77

Lecture Sheet on “+MySQL”
Day-14: (PHP)

Pagination and others **Time: 2** hours

This course provides the learner with information about implementing pagination for a enormous length of data retrieved from database or other source.

Objectives	Topics
❖ Learn the portability of MySQL databases ❖ Restoring database from sql dumps. ❖ Sending mail from PHP ▪ **Retrieving mails from IMAP, POP3 mail servers.**	❖ Backing up MySQL Database in Different ways ❖ Restore from dump of sql. ❖ Understanding Mail functions. ❖ Understanding IMAP Functions. ❖ Registration confirmation email using mail function.

Using PHP To Backup MySQL Database

There are at least three ways to backup your MySQL Database :

1. Execute a database backup query from PHP file.
2. Run mysqldump using system() function.
3. Use phpMyAdmin to do the backup.
4. Use mysqldump command from commandline
5. Use MySQL QueryBrowser/ MySQLyog GUI Tool for backing up.

1. Execute a database backup query from PHP file

Below is an example of using SELECT INTO OUTFILE query for creating table backup :

```
<?php
$tableName   = 'mytable'; $backupFile = 'mytable.sql';
$query       = "SELECT * INTO OUTFILE '$backupFile' FROM $tableName";
$result = mysql_query($query);
?>
```

To restore the backup you just need to run LOAD DATA INFILE query like this :

```
<?php
$tableName   = 'mypet';   $backupFile = 'mypet.sql';
$query       = "LOAD DATA INFILE 'backupFile' INTO TABLE $tableName";
$result = mysql_query($query);
?>
```

It's a good idea to name the backup file as tablename.sql so you'll know from which table the backup file is

2. **Run mysqldump using system() function**

The system() function is used to execute an external program. Because MySQL already have built in tool for creating MySQL database backup (mysqldump) let's use it from our PHP script

```
<?php
$backupFile = $dbname . date("Y-m-d-H-i-s") . '.gz';
$command = "mysqldump --opt -h $dbhost -u $dbuser -p $dbpass $dbname | gzip > $backupFile";
system($command);
?>
```

3. **Use phpMyAdmin to do the backup**

This option as you may guessed doesn't involve any programming on your part. However I think i mention it anyway so you know more options to backup your database.

To backup your MySQL database using phpMyAdmin click on the "export" link on phpMyAdmin main page. Choose the database you wish to backup, check the appropriate SQL options and enter the name for the backup file.

4. Use mysqldump command to backup the database (Most Recommended Technique)

For dumping the database use the commandline from mysql’s bin folder ie. C:\xampp\mysql\bin

```
mysqldump -u username -p databasename >> filename.sql
```

Then import the dump from from .sql file using mysql. After logging in to mysql. Use the following command.

```
mysql -u username -p databasename < file.sql
```

For this method you may have to upload the .sql file to the webhost through FTP(File Transfer Protocol) or SSH(Secure Shell Host) methods. Study the parameters of mysqldump and mysql for more options to work on real jobs.

Notice: the >> and < for redirecting input or output.

5. Use MySQL Query Browser/ MySQLyog to Backup and Restore the Database.

Mail (PHP 4, PHP 5) mail — Send mail

Description Sends an email.

bool mail (string $to, string $subject, string $message [, string $additional_headers [, string $additional_parameters]])

PARAMETERS

to Receiver, or receivers of the mail.

subject Subject of the email to be sent.

Caution This must not contain any newline characters, or the mail may not be sent properly.

message Message to be sent.

Each line should be separated with a LF (\n). Lines should not be larger than 70 characters.

Caution

(Windows only) When PHP is talking to a SMTP server directly, if a full stop is found on the start of a line, it is removed. To counter-act this, replace these occurrences with a double dot.
```
<?php
$text = str_replace("\n.", "\n..", $text);
?>
```

additional_headers (optional) String to be inserted at the end of the email header.

This is typically used to add extra headers (From, Cc, and Bcc). Multiple extra headers should be separated with a CRLF (\r\n).

Note: When sending mail, the mail *must* contain a *From* header. This can be set with the *additional_headers* parameter, or a default can be set in *php.ini*.

Failing to do this will result in an error message similar to *Warning: mail(): "sendmail_from" not set in php.ini or custom "From:" header missing*. The *From* header sets also *Return-Path* under Windows.

Note: If messages are not received, try using a LF (\n) only. Some poor quality Unix mail transfer agents replace LF by CRLF automatically (which leads to doubling CR if CRLF is used). This should be a last resort, as it does not comply with » RFC 2822.

additional_parameters (optional)

The *additional_parameters* parameter can be used to pass an additional parameter to the program configured to use when sending mail using the *sendmail_path* configuration setting. For example, this can be used to set the envelope sender address when using sendmail with the *-f* sendmail option.

The user that the webserver runs as should be added as a trusted user to the sendmail configuration to prevent a 'X-Warning' header from being added to the message when the envelope sender (-f) is set using this method. For sendmail users, this file is */etc/mail/trusted-users*.

Return Values Returns TRUE if the mail was successfully accepted for delivery, FALSE otherwise.

It is important to note that just because the mail was accepted for delivery, it does NOT mean the mail will actually reach the intended destination.

Example: Sending mail.

```
<?php
// The message
$message = "Line 1\nLine 2\nLine 3";
//In case any of our lines are larger than 70 characters, we should use wordwrap()
$message = wordwrap($message, 70);
```

```
mail('caffinated@example.com', 'My Subject', $message);// Send
?>
```

Example: Sending mail with extra headers.

The addition of basic headers, telling the MUA the From and Reply-To addresses:

```
<?php
$to      = 'nobody@example.com';
$subject = 'the subject';
$message = 'hello';
$headers = 'From: webmaster@example.com' . "\r\n" .
    'Reply-To: webmaster@example.com' . "\r\n" .
    'X-Mailer: PHP/' . phpversion();
mail($to, $subject, $message, $headers);
?>
```

Example: Sending mail with an additional command line parameter.

The ***additional_parameters*** parameter can be used to pass an additional parameter to the program configured to use when sending mail using the *sendmail_path*.

```
<?php
mail('nobody@example.com', 'the subject', 'the message', null, '-fwebmaster@example.com');
?>
```

Example: Sending HTML email

```
<?php
// multiple recipients
$to  = 'aidan@example.com' . ', '; // note the comma
$to .= 'wez@example.com';

// subject
$subject = 'Birthday Reminders for August';

// message
$message = '
<html><head><title>Birthday Reminders for August</title></head><body>
  <p>Here are the birthdays upcoming in August!</p>
  <table>
    <tr>      <th>Person</th><th>Day</th><th>Month</th><th>Year</th>    </tr>
    <tr>      <td>Joe</td><td>3rd</td><td>August</td><td>1970</td>    </tr>
    <tr>      <td>Sally</td><td>17th</td><td>August</td><td>1973</td>    </tr>
  </table>
</body></html>';

// To send HTML mail, the Content-type header must be set
```

```
$headers  = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
// Additional headers
$headers .= 'To: Mary <mary@example.com>, Kelly <kelly@example.com>' . "\r\n";
$headers .= 'From: Birthday Reminder <birthday@example.com>' . "\r\n";
$headers .= 'Cc: birthdayarchive@example.com' . "\r\n";
$headers .= 'Bcc: birthdaycheck@example.com' . "\r\n";

// Mail it
mail($to, $subject, $message, $headers);
?>
```

Note: If intending to send HTML or otherwise Complex mails, it is recommended to use the PEAR package PEAR::Mail_Mime. *http://pear.php.net/package/Mail_Mime*

Note: The Windows implementation of **mail()** differs in many ways from the Unix implementation. First, it doesn't use a local binary for composing messages but only operates on direct sockets which means a *MTA* is needed listening on a network socket (which can either on the localhost or a remote machine).
Second, the custom headers like *From:*, *Cc:*, *Bcc:* and *Date:* are **not** interpreted by the *MTA* in the first place, but are parsed by PHP.
As such, the *to* parameter should not be an address in the form of "Something <someone@example.com>". The mail command may not parse this properly while talking with the MTA.
Note: Email with attachments and special types of content (e.g. HTML) can be sent using this function. This is accomplished via MIME-encoding - for more information, see this Zend article(http://www.zend.com/zend/spotlight/sendmimeemailpart1.php) or the PEAR Mime Classes(http://pear.php.net/package/Mail_Mime).
Note: It is worth noting that the **mail()** function is not suitable for larger volumes of email in a loop. This function opens and closes an SMTP socket for each email, which is not very efficient.
For the sending of large amounts of email, see the PEAR::Mail, and PEAR::Mail_Queue packages.
See Also imap_mail() PEAR::Mail PEAR::Mail_Mime

imap_open (PHP 4, PHP 5) imap_open — Open an IMAP stream to a mailbox

Description

resource **imap_open** (string $mailbox, string $username, string $password [, int $options [, int $n_retries]])
Opens an IMAP stream to a *mailbox*.
This function can also be used to open streams to POP3 and NNTP servers, but some

functions and features are only available on IMAP servers.

PARAMETERS

mailbox A mailbox name consists of a server and a mailbox path on this server. The special name *INBOX* stands for the current users personal mailbox. Mailbox names that contain international characters besides those in the printable ASCII space have to be encoded width imap_utf7_encode().

The server part, which is enclosed in '{' and '}', consists of the servers name or ip address, an optional port (prefixed by ':'), and an optional protocol specification (prefixed by '/').

The server part is mandatory in all mailbox parameters.

All names which start with *{* are remote names, and are in the form *"{" remote_system_name [":" port] [flags] "}" [mailbox_name]* where:

- *remote_system_name* - Internet domain name or bracketed IP address of server.
- *port* - optional TCP port number, default is the default port for that service
- *flags* - optional flags, see following table.
- *mailbox_name* - remote mailbox name, default is INBOX

Table Optional flags for names

Flag	Description
/service=service	mailbox access service, default is "imap"
/user=user	remote user name for login on the server
/authuser=user	remote authentication user; if specified this is the user name whose password is used (e.g. administrator)
/anonymous	remote access as anonymous user
/debug	record protocol telemetry in application's debug log
/secure	do not transmit a plaintext password over the network
/imap, /imap2, /imap2bis, /imap4, /imap4rev1	equivalent to /service=imap
/pop3	equivalent to /service=pop3
/nntp	equivalent to /service=nntp
/norsh	do not use rsh or ssh to establish a pre-authenticated IMAP session
/ssl	use the Secure Socket Layer to encrypt the session
/validate-cert	validate certificates from TLS/SSL server (this is the default behavior)
/novalidate-cert	do not validate certificates from TLS/SSL server, needed if server uses self-signed certificates
/tls	force use of start-TLS to encrypt the session, and reject connection to servers that do not support it
/notls	do not do start-TLS to encrypt the session, even with

Flag	Description
	servers that support it
/readonly	request read-only mailbox open (IMAP only; ignored on NNTP, and an error with SMTP and POP3)

username	The user name
password	The password associated with the *username*
options	The *options* are a bit mask with one or more of the following:

- **OP_READONLY** - Open mailbox read-only
- **OP_ANONYMOUS** - Don't use or update a *.newsrc* for news (NNTP only)
- **OP_HALFOPEN** - For IMAP and NNTP names, open a connection but don't open a mailbox.
- **CL_EXPUNGE** - Expunge mailbox automatically upon mailbox close (see also imap_delete() and imap_expunge())
- **OP_DEBUG** - Debug protocol negotiations
- **OP_SHORTCACHE** - Short (elt-only) caching
- **OP_SILENT** - Don't pass up events (internal use)
- **OP_PROTOTYPE** - Return driver prototype
- **OP_SECURE** - Don't do non-secure authentication

n_retries Number of maximum connect attempts

Return Values **Returns an IMAP stream on success or FALSE on error.**

Example Different use of imap_open()

```
<?php
// To connect to an IMAP server running on port 143 on the local machine,
// do the following:
$mbox = imap_open("{localhost:143}INBOX", "user_id", "password");

// To connect to a POP3 server on port 110 on the local server, use:
$mbox = imap_open ("{localhost:110/pop3}INBOX", "user_id", "password");

// To connect to an SSL IMAP or POP3 server, add /ssl after the protocol
// specification:
$mbox = imap_open ("{imap.gmail.com:993/imap/ssl}INBOX", "user_id", "password");

// To connect to an SSL IMAP or POP3 server with a self-signed certificate,
// add /ssl/novalidate-cert after the protocol specification:
$mbox = imap_open ("{pop.gmail.com:995/pop3/ssl/novalidate-cert}", "user_id", "password");

// To connect to an NNTP server on port 119 on the local server, use:
```

```
$nntp = imap_open ("{localhost:119/nntp}comp.test", "", "");
// To connect to a remote server replace "localhost" with the name or the
// IP address of the server you want to connect to.
?>
```

Example imap_open() example

```
<?php
$mbox = imap_open("{imap.example.org:143}", "username", "password");

echo "<h1>Mailboxes</h1>\n";
$folders = imap_listmailbox($mbox, "{imap.example.org:143}", "*");

if ($folders == false) {
   echo "Call failed<br />\n";
} else {
   foreach ($folders as $val) echo $val . "<br />\n";
}

echo "<h1>Headers in INBOX</h1>\n";
$headers = imap_headers($mbox);

if ($headers == false) {
   echo "Call failed<br />\n";
} else {
   foreach ($headers as $val)  echo $val . "<br />\n";
}

imap_close($mbox);
?>
```

Example Code for getting email addresses as unique from your gmail inbox:

```
Enter Your Gmail username and Password to get the inbox Addresses You have recieved mails
from<br />
<form id="form1" name="form1" method="post" action=""><input name="user" type="text" />
<br /><input name="pass" type="password" />
<br />
<input name="submit" type="submit" value="Get Addresses" />
</form>
<?php if(isset($_POST['submit'])){
$mbox = imap_open ("{imap.gmail.com:993/imap/ssl}INBOX", $_POST['user'], $_POST['pass']);


/*echo "<h1>Mailboxes</h1>\n";
$folders = imap_listmailbox($mbox, "{imap.example.org:143}", "*");


if ($folders == false) {
        echo "Call failed<br />\n";
} else {
        foreach ($folders as $val)              echo $val . "<br />\n";
} */
$MC = imap_check($mbox);

// Fetch an overview for all messages in INBOX
$result = imap_fetch_overview($mbox,"1:{$MC->Nmsgs}",0);

foreach ($result as $overview) { $addresses[]=$overview->from; }
$addresses=array_unique($addresses);
foreach($addresses as $from) echo $from.",";

//echo "<h1>Headers in INBOX</h1>\n";
/*$headers = imap_headers($mbox);

if ($headers == false) echo "Call failed<br />\n";
  else foreach ($headers as $val)       echo $val . "<br />\n";
} */
imap_close($mbox); }
?>
```