PHP - Beyond the Basics

John Valance

division 1 systems

johnv@div1sys.com

www.div1sys.com

About John Valance

- **Independent Consultant for past 15 years**
- **Founder and CTO of Division 1 Systems (www.div1sys.com)**
  - Specialty is helping IBM shops develop web apps and related skills
  - Training, mentoring, project management, consultation and coding
- **30+ years IBM midrange experience (S/38 thru IBM i)**
- **12+ years of web development experience**
  - Web scripting language of choice = PHP
- **Frequent presenter on web development topics**
- **Trainer for Zend Technologies**
  - Teaches Intro to PHP for RPG programmers
  - Zend Certified Engineer

Goals of This Presentation

- **Cover tips and techniques useful to development of business applications in PHP on IBM i**

- **Topics:**
  - Database paging
  - File system processing / Writing CSV Content
  - Email
  - Session management / Cookies

Assumptions

- **You understand**
  - Basic PHP syntax
  - Basics of web application coding in PHP
  - Have done some basic DB2 applications with PHP
- **You are ready to go a little deeper**
  - Pick up a few ideas and “how-to” tips

Database Paging

Database Paging

- **Large Result Sets**
  - Don't load all records on screen at once
  - Show subset page by page
    - i.e. 20 to 50 records per page
- **Similar to Subfile**
  - But techniques very different, due to HTTP
- **Need mechanism / algorithm to allow user-controlled page access**
  - Involves PHP, HTML, and some JavaScript
- **Demo!**

<div1>

How do we get just the records we need?

- **Need to use a scrollable cursor**
  - Option on db2_connect() and db2_prepare() :
  - `array('cursor'=>DB2_SCROLLABLE);`
- **Allows us to fetch a specific row from result set**
  - High performance record-level access
- **DB2 only feature**
  - this won't work with MySQL or SQL Server

<div1>

3 Ways to Specify Scrollable Cursor in PHP

```
// Create options array to make DB2 cursor scrollable
$cursor_opt = array('cursor'=>DB2_SCROLLABLE);

// Can specify scrollable cursor on db2_connect()
// Affects all statements through this connection
$conn = db2_connect($srvr, $user, $pswd, $cursor_opt);

// Can also specify scrollable cursor on db2_prepare()
// Just affects one statement
$stmt = db2_prepare ( $conn, 'select * from mytable', $cursor_opt );

// Can also specify scroll cursor using db2_set_option()
// 3rd parameter indicates type of resource passed in 1st parm:
db2_set_option($conn, $cursor_opt, 1);  // 1=connection passed
db2_set_option($stmt, $cursor_opt, 2);  // 2=statement passed
```

<div1>

Fetching Specific Rows with Scroll Cursor

- **We can now specify the optional 2nd parameter on db2_fetch functions:**

```
$rowNumber = 21; // Start of page 2, if 20 rows per page
$row = db2_fetch_assoc( $stmt, $rowNumber );
```

From php.net: (http://php.net/manual/en/function.db2-fetch-assoc.php)

```
array db2_fetch_assoc ( resource $stmt [, int $row_number = -1 ] )
```

Parameters

stmt
A valid *stmt* resource containing a result set.

row_number
Requests a specific 1-indexed row from the result set. Passing this parameter results in a PHP warning if the result set uses a forward-only cursor.

<div1>

Retrieving a Specific Page with PHP

- **Retrieve $pageNum from HTML form field**

  $pageNum = $_REQUEST['pageToView'];

  - *If not present, default to page 1*


```
<input name="pageToView" />
```

- **Compute starting row number to retrieve as:**

```
define('PAGE_SIZE', 20);
$rowNum = (PAGE_SIZE * $pageNum) - PAGE_SIZE + 1;
```

- **DB Fetch Loop - while rows fetched <= PAGE_SIZE:**

```
$rows_fetched = 0;
while ( $row = db2_fetch_assoc( $stmt, $rowNum++ )) {
    if (++$rows_fetched > PAGE_SIZE ) break;
    // ... process DB row
}
```

⇐ Build HTML table rows filled with DB2 data

<div1>

Handling the View

- **We now have a PHP script that will accept a request variable named pageToView and it will produce HTML for that page of rows.**
  - Add error checking: If pageToView is not an integer, less than 1, or greater than number of pages, set pageToView = 1
- **Add buttons on screen to control pagination:**

<div1>

Button Functionality

- **Add buttons on screen to control pagination:**

Button Label	Effect	JavaScript onclick
Next Page	Add 1 to current page # and submit request	gotoPage(currentPage+1)
Previous Page	Subtract 1 from current page # and submit request	gotoPage(currentPage-1)
First Page	Request page 1	gotoPage(1)
Last Page	Request last page	gotoPage(numberOfPages)

- **JavaScript for gotoPage() function:**

```
<script>
function gotoPage(page) {
  document.objectList.pageToView.value = page;
  document.objectList.submit();
}
</script>
```

This changes the value in the page# input field

This submits the form to request the new page

<div1>

HTML for Paging Buttons in the View

First Page

```
<input type="button" value="First Page" <?= $prevState ?>
        onclick="gotoPage(1);" />
```

<< Previous

```
<input type="button" value="<< Previous"  <?= $prevState ?>
        onclick="gotoPage(<?= $currentPage - 1 ?>);" />
```

Next >>

```
<input type="button" value="Next >>" <?= $nextState ?>
        onclick="gotoPage(<?= $currentPage + 1 ?>);" />
```

Last Page

```
<input type="button" value="Last Page" <?= $nextState ?>
        onclick="gotoPage(<?= $numberOfPages ?>);" />
```

<div1>

Adding Buttons in the View

- **On first page, disable the "Previous" button**

```
if ($pageNum == 1) $prevState = "disabled";
```

- **On last page, disable the "Next" button**
  - *After exiting fetch loop, try to retrieve next record*
  - If no more, then that was last page. Disable the "Next" button

```
if (!$row = db2_fetch_assoc($stmt, $rowNum))
    $nextState = "disabled";
```

- **In HTML, echo $nextState and $prevState within button tags**

<div1>

Disabled Buttons on Page 1

First Page

```
<input type="button" value="First Page" disabled onclick="gotoPage(1);" />
```

<< Previous

```
<input type="button" value="<< Previous"  disabled onclick="gotoPage(0);" />
```

Next >>

```
<input type="button" value="Next >>" onclick="gotoPage(2);" />
```

Last Page

```
<input type="button" value="Last Page" onclick="gotoPage(6);" />
```

<div1>

Retrieving Number of Pages

- How do we know how many pages there are in query result, without reading every record?
- We need to do a separate SELECT to count number of rows in result

```
SELECT COUNT(*) as ROW_COUNT
FROM <same table>
JOIN <same inner joins>
WHERE <same filter conditions>
```

- We can then calculate total pages as:

```
$numberOfPages = ceil( (int)$row['ROW_COUNT'] / PAGE_SIZE);
```

- Use ceil() function to round page number up.

IFS Files and CSV Processing

File Processing

- **By file, we mean IFS files, not object type *FILE.**
  - E.g., text files, PDF, images, Excel, CSV
- **PHP core includes numerous file system functions**
  - http://us2.php.net/manual/en/ref.filesystem.php
- **Two types of file functions:**
  - File name-based
    - Dealing with the file as a whole
    - Receive a file name as parameter
  - Resource-based
    - Open a file to read/write portions of the file's contents
    - Receive a file "handle" resource as input

File Name-based Functions

- **Act upon a file as a whole – they do not open the file**
- **Some File Name-based Functions**
  - `bool file_exists($filename)`
  - `string file_get_contents($filename)`
  - `int file_put_contents($filename, $data)`
  - `int file_size($filename)`
  - `bool copy($source, $dest)`
  - `bool unlink($filename)  // delete a file`
- **Require a file name**
  - If no path specified, file is searched for in same folder as php script

<div1>

Resouce-based File Handling

- **First, open the file using fopen() function**
  - fopen() returns a file-handle resource

```
$fh = fopen('myfile.txt', 'a+');
```

- **Use resource returned by fopen() to call functions to read and write data**

```
$text = fread($fh, file_size('myfile.txt'));

fwrite($fh, "some new text\n");
```

- **Lastly, close the file resource using fclose()**

```
fclose($fh);
```

fopen() and Resource-based functions

- **resource fopen (string $filename , string $mode)**
  - $mode is open mode for file processing:
    - See table on next slide for fopen mode options
- **Some resource-based functions**
  - string **fread** (resource $handle , int $length)
  - int **fwrite** (resource $handle , string $string [, int $length])
  - int **fseek** (resource $handle , int $offset [, int $whence])
  - bool **feof** (resource $handle)
  - bool **fclose** (resource $handle)

<div1>

`fopen()` mode flags

Mode	Function	Read	Write	Pointer	Overwrite	Create
r	open	Yes	No	begin	No	No
r+	open	Yes	Yes	begin	No	No
w	open	No	Yes	begin	Yes	Yes
w+	open	Yes	Yes	begin	Yes	Yes
a	open	No	Yes	end	No	Yes
a+	open	Yes	Yes	end	No	Yes
x	create	No	Yes	begin	No	No
x+	create	Yes	Yes	begin	No	No
c	open/create	No	Yes	begin	No	Yes
c+	open/create	Yes	Yes	begin	No	Yes
b	binary file	Specify b along with above mode flags if binary file				

CSV file handling

- **PHP has built-in CSV handling functions**
  - Resource-based functions – need to use fopen() to get $handle
- `int fputcsv(resource $handle, array $fields)`
  - Parses the array $fields into a comma-separated string
  - Write the string to the end of the file denoted by $handle
    - Strings automatically quoted if contain blanks or commas
    - Includes newline at end of CSV string
  - Returns number of bytes written
- `array fgetcsv(resource $handle)`
  - Reads one line from $handle
  - Parses CSV content and returns an array containing an element for each value in the CSV string
  - Advances the file pointer to next line for looping

Building CSV files from DB2 content

- **Easy to create a CSV file from an SQL query**
- **Use db2_fetch_array()**
  - Returns an array of field values based on an SQL query
- **Pass returned array to fputcsv()**

```
$conn = db2_connect ( "*LOCAL", "USER", "PSWD" );
$stmt = db2_prepare( $conn, "SELECT * FROM MYTABLE" );
db2_execute( $stmt );
$fh = fopen('mytable.csv', 'w');
while ( $row = db2_fetch_array( $stmt ) ) {
     fputcsv($fh, $row);
}
fclose($fh);
db2_close($conn);
```

Adding Column Headings to CSV File

- Use db2_num_fields() and db2_field_name() functions
- Add the following before reading/writing data rows:

```
for ($col = 0; $col < db2_num_fields($stmt); $col++)
    $headings[] = db2_field_name( $stmt, $col );
fputcsv($handle, $headings); // first line of CSV file
```

- If cryptic DDS field names, create alias with 'AS' in SELECT
  - `SELECT  CSCNUM as "Customer Number",`
    `CSNAME as "Customer Name"`

Delivering a File to the Browser

- **Instead of writing to IFS, send it to user**
  - User will get a “File open/save” dialog

- **We can access the PHP output stream as a file resource**
  - php://output – Use this as filename in fopen()
  - Specify mode = ‘w’ (write)

```
$handle = fopen("php://output", 'w');
```

- **Need to do two other things:**
  - Buffer output
    - Sends whole file at once, and avoid problems with headers
  - Specify content type and file name
    - Use header() function to set values in HTTP headers sent to browser

Delivering a File to the Browser

```
ob_start(); // start output buffering

// set file type and name in HTTP header

header("Content-type: application/csv;");

header('Content-Disposition: attachment;
       filename="membership.csv"');

... do db2 query execute

$handle = fopen("php://output", 'w');

... write content to $handle as before

... after db2_close() and fclose():

// Flush output buffer - send entire file to browser

ob_end_flush();
```

Sending Email

Sending Email

- **PHP mail() function**
  - Built-in to PHP core
  - Simple, easy to use
  - Best suited for text-only messages

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

Example:

```
$to      = 'customer@gmail.com';
$subject = 'Test Email';
$message = 'Testing 1,2,3';
$headers = "From: custserv@ourcompany.com\n" .
           "Reply-To: custserv@ourcompany.com\n";
mail($to, $subject, $message, $headers);
```

- **SMTP server/port is set in php.ini**

<div1>

Adding Email Attachments

- **Underlying protocols are complex**
  - based on RFC822
- **Can be done with `mail()` function, but not easy**
  - Requires understanding MIME formats
    - MIME = Multipurpose Internet Mail Extensions
    - http://en.wikipedia.org/wiki/MIME
- **Best to use a package that makes it simple**
  - PEAR::Mail_Mime
    - http://pear.php.net/package/Mail_Mime
  - Zend Framework: Zend_Mail class 
    - http://framework.zend.com/manual/1.11/en/zend.mail.html
    - Zend Framework included with Zend Server (even CE)
    - Very simple interface
    - Great integration with other Zend products

Using Zend_Mail

- **To use Zend Framework classes in your code, add these two lines at top of your script:**

```
require_once 'Zend/Loader/Autoloader.php';
Zend_Loader_Autoloader::getInstance();
```

Note: path to Zend Framework library folder is already set in your include path by ZS installation

- **Example – sending plain text message:**

```
$mail = new Zend_Mail();
$mail->setFrom('johnv@jvalance.com', 'Our Company');
$mail->addTo('jvalance@sprynet.com', 'J. Valance');
$mail->addTo('john.valance@gmail.com', 'John V.');
$mail->setSubject('Test Order Confirm');
$mail->setBodyText('This is to confirm your recent order...');
$mail->send();
```

Add an Attachment

```
$pdf = file_get_contents('some_pdf_file.pdf');
$attach = $mail->createAttachment(
                        $pdf,
                        'application/pdf',
                        Zend_Mime::DISPOSITION_ATTACHMENT,
                        Zend_Mime::ENCODING_BASE64
                );
$attach->filename = 'brochure.pdf';
```

Note: `application/pdf` = Content-type

- Tells email client what program to open attachment with
- *Other examples:*
  - `application/csv` (Excel most likely)
  - `img/jpg`

HTML-formatted Emails

- **Use `$mail->setBodyHtml('…html content…')`**
- **Should also `setBodyText('…text content…')` for recipients that only receive text**
- **HTML emails can be tricky…**
  - Some email clients don't handle them well / the same
    - Web-based clients
    - PDAs / Smart-phones
  - A lot of companies stick to plain text notification emails
  - Rules of thumb for successful HTML emails:
    - Use <table>s for layout (vs. CSS positioning etc.)
    - Specify CSS attributes inline, vs, style sheet
      - i.e. - as <tag style="…"> attribute, no matter how redundant
  - Images in HTML:
    - better to use external file references for images (vs. image attachments)
    - i.e. <img src=http://www.mycompany.com/logo.gif>

Session Management and Cookies

Session Management

- **HTTP protocol is stateless**
  - There is no continuous connection to server
  - Each request/response is completely independent of the next
- **Web applications need a mechanism to simulate a user session**
- **PHP makes this easy with session functions and session variables**
- **Session variables are stored on the server by PHP**
  - Session variables are keyed by a session ID
  - Session variables are accessed via the $_SESSION array
- **Session ID is stored in a cookie on the client**
  - This is triggered by PHP's session_start() function
  - Cookies are automatically sent with request by browser

Using session variables in PHP

- **Must call session_start before accessing session variables**
  - This alters response headers, so it must be called before any response content (i.e. HTML text) is echoed.

```
session_start();
```

- **After session_start, access session vars using elements of $_SESSION array:**

```
$_SESSION['userid'] = 'JVALANCE';
```

```
$orderNum = $_SESSION['current_order_num'];
```

<div1>

PHP `session_start()` function

Was session cookie sent with request?

- **Yes:**
  - Retrieve session ID
  - Load $_SESSION array for session ID
- **No:**
  - Generate session ID
  - Initialize $_SESSION array
  - Send session cookie on response

Session Management

- **Login Script:**

```
session_start(); // must happen before any output
… validate user/pswd
$_SESSION['userid'] = $_POST['userid'];
… other processing
```

- **Application scripts include this at top:**

```
session_start();
if (! isset($_SESSION['userid'])
     header('Location: login.php'); // redirect to login
     exit;  // always exit after redirect
else
     echo “Hello “ . $_SESSION['userid'];
```

- **Logout:**

```
session_start();
session_destroy();
setcookie(session_name(),'',0,'/'); // expire cookie
header('Location: login.php'); // redirect to login
exit; // always exit after redirect
```

Session Persistance

Session variables persist until one of these happens:

- **session_destroy() is called**
- **browser windows are all closed**
  - all windows/tabs of the same browser brand
- **session cookie times out**
  - based on **session.cookie_lifetime** in php.ini
  - this is unreliable – better to code your own session timeout logic

Session Timeout Handling

- **Session timeout logic**

```
if ( isset($_SESSION['LAST_ACTIVITY'])
&& ( time() - $_SESSION['LAST_ACTIVITY'] > 1800) ) {
    // last request was more than 30 minutes ago
    session_destroy();      // destroy session data
    session_unset();        // unset $_SESSION vars
    setcookie(session_name(),'',0,'/'); // expire cookie
} else {
    // update last activity time stamp
    $_SESSION['LAST_ACTIVITY'] = time();
}
```

- **Add to start of every script**

Cookies

- **To persist user information beyond a session, set a cookie**

```
setcookie(
   'mycookie',          // name
   ‘Oreo’,              // value
   time()+(60*60*24*30) // expire in 30 days
);
```

- **Must call setcookie() before any browser output**
  - unless buffering is on
  - because cookies are set via response headers
- **Retrieve value via $_COOKIES array:**

```
$cookieValue = $_COOKIES[‘mycookie’];   // ‘Oreo’
```

<div1>

Summary

Summary

- **Database paging**
  - Scrollable cursor
  - Specify starting row on db2_fetch
- **File system processing / Writing CSV Content**
  - File-based functions vs. Resource-based functions
  - fputcsv() and fgetcsv()
  - $download = fopen("php://output", 'w');
    - Use buffering
    - Specifying file name and type with header() function

Summary

- **Email**
  - mail() – simple emails without attachments
  - Zend_Mail – attachments / HTML
- **Session management / Cookies**
  - HTTP = stateless protocol
  - session_start() / session_destroy()
  - $_SESSION array
  - setcookie() – persist information beyond session

Contact Information

John Valance

johnv@div1sys.com

802-355-4024

Division 1 Systems

<div1>

http://www.div1sys.com

Thank you for attending!

<div1>