FREE eBook

LEARNING pagination

Free unaffiliated eBook created from
Stack Overflow contributors.

#pagination

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: pagination

It is an unofficial and free pagination ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official pagination.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with pagination

Remarks

This section provides an overview of what pagination is, and why a developer might want to use it.

It should also mention any large subjects within pagination, and link out to the related topics. Since the Documentation for pagination is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting pagination set up or installed.

Pagination solution

The following documentation describes both MySQLi and PDO supported pagination solution.

Go to https://github.com/rajdeeppaul/Pagination and download `pagination.php` file into your project directory. Let's say your directory structure looks like this:

```
project directory
       |
       |--pagination.php (pagination script)
       |--index.php (file where you want to use this pagination script)
```

And suppose you want the URL to look like this:

```
http://example.com/index.php          // user is on page 1
http://example.com/index.php?page=1   // user is on page 1
http://example.com/index.php?page=5   // user is on page 5
// etc ...
```

index.php

1. Include `pagination.php` file in `index.php` page, like this:

   ```
   require_once('pagination.php');
   ```

2. Create an instance of `Pagination` class, like this:

   ```
   $pg = new Pagination($databaseDriver, $hostname, $username, $password, $databaseName);
   ```

The constructor method takes the following parameters,

(1)`$databaseDriver`: Database driver, currently supported drivers are MySQLi and PDO
(2)`$hostname`: Hostname
(3)`$username`: Username
(4)`$password`: Password
(4)`$databaseName`: Database name

Example(s):

```
$pg = new Pagination('mysqli', 'localhost', 'root', 'pass', 'pagination_db');
$pg = new Pagination('pdo', 'localhost', 'root', 'pass', 'pagination_db');
```

3. Set pagination parameters using `setPaginationParameters()` method, like this:

```
$pg->setPaginationParameters($rowsPerPage, $numOfPaginationLinks);
```

The `setPaginationParameters()` method takes the following parameters,

(1)`$rowsPerPage`: Number of table rows to display per page
(2)`$numOfPaginationLinks`: Number of pagination links to display per page

Example(s):

```
$pg->setPaginationParameters(10, 5);
```

4. Call `getResult()` method to display table rows based on the URL query `?page=X`, like this:

```
$resultSet = $pg->getResult($queryString, $bindParameterArray, $globalGetArray,
$keyFromURLQuery);
```

The `getResult()` method takes the following parameters,

(1)`$queryString`: `SELECT` query string
(2)`$bindParameterArray`: Array containing bind variables or values
(3)`$globalGetArray`: Superglobal array `$_GET`
(4)`$keyFromURLQuery`: Key from the URL query i.e. `page`

Example(s):

```
$resultSet = $pg->getResult('SELECT * FROM pagination', NULL, $_GET, 'page');
$resultSet = $pg->getResult('SELECT * FROM pagination WHERE column1 = ? AND column2 = ?',
array($value1, $value2), $_GET, 'page');
```

Note: Don't specify any `LIMIT` or `OFFSET` clause in the query, the script will take care of these.

Now loop through the result set i.e. `$resultSet` array to access/display row details, like this:

```
foreach($resultSet as $row){
```

```
    /* access/display row details */
    /* $row['column1'], $row['column2'] etc. */
}
```

Note: If you want to see the complete array structure, do `var_dump($resultSet);`.

5. Display pagination links using `getPaginationLinks()` method, like this:

```
$pgLinks = $pg->getPaginationLinks();
```

The `getPaginationLinks()` method doesn't take any parameter and returns an array of pagination links in the following format,

```
array (size=3)
  'prev' => @boolean
  'links' => @array
  'next' => @boolean
```

Now loop through the `$pgLinks` array to display paginagtion links, like this:

```
if(is_array($pgLinks) && count($pgLinks) && $pgLinks['prev']){
    /* previous pages are available */
    echo '&laquo; ';
}
if(is_array($pgLinks) && count($pgLinks) && count($pgLinks['links'])){
    /* show pagination links */
    foreach($pgLinks['links'] as $link){
        echo '<a href="example.php?page='.$link.'">'.$link.'</a> ';
    }
}
if(is_array($pgLinks) && count($pgLinks) && $pgLinks['next']){
    /* next pages are available */
    echo '&raquo;';
}
```

Note: If you want to see the complete array structure, do `var_dump($pgLinks);`.

Footnote(s): You can style the result rows and pagination links as per your choice.

Read Getting started with pagination online: https://riptutorial.com/pagination/topic/7280/getting-started-with-pagination

Chapter 2: Pagination with PHP and MySql

Parameters

Parameter	Details
$params = []	optional parameters for `['table' => 'tableName', 'sort' => 'ASC', 'columns' => 'colId, name, etc']`
$attributes = []	optional list of attributes `['ul-class': => 'space separated list of classes', 'ul-attr': 'id="someId" data-pre="pre"', 'li-class': 'space separated list of classes', 'li-attr': 'id="someid"']`

Examples

Creating a Simple Pagination

You need to include the `Pagination.php` in your page/s.

```
require_once 'Paginator.php';

$Paginator = new Paginator('mysql:host=localhost;dbname=ng_app', 'root', '000000');
$Paginator->setItemLimitPerPage(4);
$Paginator->setTable('comments');
$Paginator->createPages(); // this will create pages using PHP copy()
$Paginator->setCurrentPageClass('active'); // set the current page class
$Paginator->setUrlPattern('/php_paginator/');
$numPrevPage = 4; // number of pages to appear before the current page
$numNextPage = 4; // number of pages to appear after the current page
$paginationCssClass = 'pagination';
```

Creating Pagination

```
<div class="text-center">
    <!-- our pagination  using Bootstrap-->
    <hr>
    <?php
      $Paginator->pagination($Paginator->getPageNumber(), $numPrevPage, $numNextPage,
$paginationCssClass);
    ?>
</div
```

« Previous 1 2 3 Next »

Creating Paginator file

```
<?php

class Paginator
{
    private $_db;
    private $_table = null;
    private $_currentPageClass = '';
    private $_itemLimitPerPage;
    private $_rowOffset = 0;
    private $_urlPattern = '/';

    /**
     * @return string url pattern
     */
    public function getUrlPattern()
    {
        return $this->_urlPattern;
    }

    /**
     * @param string $urlPattern
     */
    public function setUrlPattern($urlPattern)
    {
        $this->_urlPattern = $urlPattern;
    }

    /**
     * @return int value of itemLimitPerPage
     */
    public function getItemLimitPerPage()
    {
        return $this->_itemLimitPerPage;
    }

    /**
     * @param int $limitItems number of items per page
     */
    public function setItemLimitPerPage($limitItems)
    {
        $this->_itemLimitPerPage = $limitItems;
    }

    /**
     * @return int value of rowOffset
     */
    public function getRowOffset()
    {
        return $this->_rowOffset;
    }

    /**
     * @param int $rowOffset number of row offset
     */
    public function setRowOffset($rowOffset)
    {
        $this->_rowOffset = $rowOffset;
```

```
    }


    /**
     * Paginator constructor.
     * @param string $dsn database host and database name
     * @param string $username database username
     * @param string $password user password
     */
    public function __construct($dsn, $username, $password)
    {
        try {
            $this->_db = new PDO($dsn, $username, $password);
        } catch (PDOException $e) {
            echo $e->getMessage();
        }
    }

    /**
     * Get the name of the table
     * @return string the name of the table
     */
    public function getTable()
    {
        return $this->_table;
    }

    /**
     * Set the name of the table
     * @param string $table the name of the table to be used
     */
    public function setTable($table)
    {
        $this->_table = $table;
    }

    /**
     * Get the class to be used on the current item/page
     * @return string the current page class
     */
    public function getCurrentPageClass()
    {
        return $this->_currentPageClass;
    }

    /**
     * Set the class to be used on the current item/page
     * @param string $currentPageClass set the class to be used for the current page
     */
    public function setCurrentPageClass($currentPageClass)
    {
        $this->_currentPageClass = $currentPageClass;
    }


    /**
     * Get the number of rows available
     * @param null $table optional table name
     * @return int the number of row count
     * @throws Exception when table is not set or provided
     */
```

```
    public function getRowCount($table = null)
    {
        if ($this->_table === null && $table === null) {
            throw new Exception("Table was not set");
        } else {
            if ($table !== null) {
                $stmt = $this->_db->prepare("SELECT * FROM $table");
                $stmt->execute();
                return $stmt->rowCount();
            } elseif ($this->_table !== null) {
                $stmt = $this->_db->prepare("SELECT * FROM $this->_table");
                $stmt->execute();
                return $stmt->rowCount();
            }
        }
    }

    /**
     * Get the number of rows left from the database
     * @param null $table optional table name
     * @return int number of rows left
     * @throws Exception when table is not set or provided
     */
    public function getRowsLeft($table = null)
    {
        if ($this->getCurrentPage() !== 'index.php') {
            $this->_rowOffset = ($this->_itemLimitPerPage * $this->getPageNumber());
        }
        if ($this->_table === null && $table === null) {
            throw new Exception("Table was not set");
        } else {
            if ($table !== null) {
                $stmt = $this->_db->prepare("SELECT * FROM $table LIMIT " . $this-
>getRowOffset() . "," . $this->getItemLimitPerPage());
                $stmt->execute();
                return $stmt->rowCount();
            } elseif ($this->_table !== null) {
                $stmt = $this->_db->prepare("SELECT * FROM $this->_table LIMIT " . $this-
>getRowOffset() . "," . $this->getItemLimitPerPage());
                $stmt->execute();
                return $stmt->rowCount();
            }
        }
    }

    /**
     * Get data to be used on the current page
     * @param int $colId column id
     * @param array $params optional parameters for ['table' => 'tableName', 'sort' => 'ASC',
'columns' => 'colId, name, etc']
     * @return array columns from database
     * @throws Exception when table is not set or provided
     */
    public function getPageData($colId, $params = [])
    {
        if ($this->_table === null && !isset($params['table'])) {
            throw new Exception("Table was not set");
        }
        $columns = isset($params['columns']) ? $params['columns'] : '*';
        $sort = isset($params['sort']) ? $params['sort'] : 'DESC';
        if (isset($params['table'])) {
```

```
            $table = $params['table'];
            $rowsLeft = $this->getRowsLeft($table);
            if ($rowsLeft < $this->_itemLimitPerPage) {
                $this->_itemLimitPerPage = $rowsLeft;
            }
            $select = "SELECT $columns FROM " . $table . " ORDER BY $colId $sort LIMIT ?,?";
            $prepare = $this->_db->prepare($select);
            $prepare->bindParam(1, $this->_rowOffset, PDO::PARAM_INT);
            $prepare->bindParam(2, $this->_itemLimitPerPage, PDO::PARAM_INT);
            $prepare->execute();
            $results = $prepare->fetchAll();
            return $results;
        } elseif ($this->_table !== null) {
            $rowsLeft = $this->getRowsLeft($this->_table);
            if ($rowsLeft < $this->_itemLimitPerPage) {
                $this->_itemLimitPerPage = $rowsLeft;
            }
            $prepare = $this->_db->prepare("SELECT * FROM $this->_table ORDER BY $colId $sort
LIMIT " . $this->getRowOffset() . "," . $this->getItemLimitPerPage());
            $prepare->execute();
            $results = $prepare->fetchAll();
            return $results;
        }
    }

    /**
     * Create pages that will appear before the current page
     * @param int $pageNumber the current page number
     * @param int $numPrevPages the number of pages to appear before the current page
     * @param $cssClass class set to the li list
     * @param $attr attribtes for li list
     * @return string list of pagination links
     */
    function prevPages($pageNumber, $numPrevPages, $cssClass, $attr)
    {
        $listItems = ''; // to save all list items.
        while ($numPrevPages >= 1) {
            $pageNumber -= 1;
            if ($pageNumber >= 1) {
                $page = $pageNumber . '.php';
                if (file_exists("$page")) {
                    $listItems = '<li class="' . $cssClass . '" ' . $attr . '><a href="' .
$this->getUrlPattern() . $pageNumber . '.php">' . $pageNumber . '</a></li>' . $listItems;
                }
            }
            $numPrevPages -= 1;
        }
        return $listItems;
    }

    /**
     * Create pages that will appear after the current page
     * @param $pageNumber the current page number
     * @param $numNextPages the number of pages to appear after the current page
     * @param $cssClass class set to the li list
     * @param $attr attribtes for li list
     * @return string list of pagination links
     */
    function nextPages($pageNumber, $numNextPages, $cssClass, $attr)
    {
        $listItems = ''; // to save list items.
```

```
        $count = 1;
        while ($count <= $numNextPages) {
            $pageNumber += 1;
            $page = $pageNumber . '.php';
            if (file_exists("$page")) {
                $listItems .= '<li class="' . $cssClass . '" ' . $attr . '><a href="' . $this-
>getUrlPattern() . $pageNumber . '.php">' . $pageNumber . '</a></li>';
            }
            $count += 1;
        }
        return $listItems;
    }

    /**
     * Create the pagination links
     * @param $pageNumber the current page number
     * @param $numPrevPages the number of pages to appear before the current page
     * @param $numNextPages  the number of pages to appear after the current page
     * @param array $attributes optional list of list attributes.
     * ['ul-class': => 'space separated list of classes', 'ul-attr': 'id="someId" data-
pre="pre"', 'li-class': 'space separated list of classes', 'li-attr': 'id="someid"']
     */
    function pagination($pageNumber, $numPrevPages, $numNextPages, $attributes = [])
    {
        $ulCssClass = isset($attributes['ul-class']) ? $attributes['ul-class'] : '';
        $ulAttr = isset($attributes['ul-attr']) ? $attributes['ul-attr'] : '';
        $liCssClass = isset($attributes['li-class']) ? $attributes['li-class'] : '';
        $liAttr = isset($attributes['li-attr']) ? $attributes['li-attr'] : '';
        $prevPagesList = '<ul class="' . $ulCssClass . '" ' . $ulAttr . '>' . $this-
>prevButton($pageNumber) . $this->prevPages($pageNumber, $numPrevPages, $liCssClass, $liAttr);
        $nextPageList = $this->nextPages($pageNumber, $numNextPages, $liCssClass, $liAttr) .
$this->nextButton($pageNumber) . '</ul>';
        if ($pageNumber == 'index') {
            $listItems = $prevPagesList . $nextPageList;
        } else {
            $listItems = $prevPagesList . '<li class="' . $this->getCurrentPageClass() . '"><a
href="">' . $pageNumber . '</a> </li>' . $nextPageList;
        }
        echo $listItems;
    }

    /**
     * Create a link for previous button
     * @param $pageNumber the current page number
     * @return string the previous link item list
     */
    function prevButton($pageNumber)
    {
        $prev = '';
        if ($pageNumber == 1) {
            $prev = '<li><a href="index.php">&laquo; Previous</a></li>';
        } elseif ($pageNumber > 1) {
            $prev = '<li><a href="' . $this->getUrlPattern() . ($pageNumber - 1) . '.php' .
'">&laquo; Previous</a></li>';
        }
        return $prev;
    }

    /**
     * Create a link for next button
     * @param $pageNumber the current page number
```

```
     * @return string the next link item list
     */
    function nextButton($pageNumber)
    {
        if ($pageNumber == 'index') {
            $page = '1.php';
        } else {
            $page = ($pageNumber + 1) . '.php';
        }
        if (file_exists($page)) {
            return '<li><a href="' . $this->getUrlPattern() . ($pageNumber + 1) . '.php">Next
&raquo; </a></li>';
        }
        return '';
    }

    /**
     * Get the current page number
     * @return int the current page number
     */
    function getPageNumber()
    {
        $currentPage = basename($_SERVER['SCRIPT_FILENAME']);
        $pageNumber = rtrim($currentPage, '.php');
        return $pageNumber;
    }

    /**
     * Get the current page
     * @return string return the current page
     */
    function getCurrentPage()
    {
        $currentPage = basename($_SERVER['SCRIPT_FILENAME']);
        return $currentPage;
    }

    /**
     * create the required pages
     */
    function createPages()
    {
        $last_page = ($this->getRowCount() / $this->getItemLimitPerPage()) - 1;
        if (!is_int($last_page)) {
            $last_page = (int)$last_page + 1;
        }
        for ($counter = 1; $counter <= $last_page; $counter++) {
            $page = $counter . '.php';
            if (!file_exists($page)) {
                copy('index.php', $page);
            }
        }
    }
}
```

see this repo for more info

Read Pagination with PHP and MySql online:
https://riptutorial.com/pagination/topic/9914/pagination-with-php-and-mysql

Credits

S. No	Chapters	Contributors
1	Getting started with pagination	Community, Rajdeep Paul
2	Pagination with PHP and MySql	julekgwa