EBook Gratis

APRENDIZAJE pagination

Free unaffiliated eBook created from
Stack Overflow contributors.

#pagination

Tabla de contenido

Acerca de 1

Capítulo 1: Empezando con la paginación 2

Observaciones 2

Examples 2

Instalación o configuración 2

Solución de paginación 2

index.php 2

Capítulo 2: Paginación con PHP y MySql. 5

Parámetros 5

Examples 5

Creando una Paginación Simple 5

Creación de paginación 5

Creando el archivo Paginator 6

Creditos 12

Acerca de

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

Capítulo 1: Empezando con la paginación

Observaciones

Esta sección proporciona una descripción general de qué es la paginación y por qué un desarrollador puede querer usarla.

También debe mencionar cualquier tema grande dentro de la paginación y vincular a los temas relacionados. Como la Documentación para paginación es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Instalación o configuración

Instrucciones detalladas para configurar o instalar la paginación.

Solución de paginación

La siguiente documentación describe las soluciones de paginación compatibles con MySQLi y PDO.

Vaya a https://github.com/rajdeeppaul/Pagination y descargue el archivo `pagination.php` en el directorio de su proyecto. Digamos que la estructura de su directorio se ve así:

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

Y supongamos que quieres que la URL se vea así:

```
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. Incluya el archivo `pagination.php` en la página `index.php` , como esto:

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

2. Crea una instancia de la clase de `Pagination` , como esta:

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

El método constructor toma los siguientes parámetros,

(1) `$databaseDriver` : controlador de `$databaseDriver` datos, los controladores actualmente admitidos son MySQLi y PDO
(2) `$hostname` : Nombre de host
(3) `$username` : `$username` usuario
(4) `$password` : contraseña
(4) `$databaseName` : nombre de la `$databaseName` datos

Ejemplo (s):

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

3. Establezca los parámetros de paginación utilizando el método `setPaginationParameters()` , como este:

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

El método `setPaginationParameters()` toma los siguientes parámetros,

(1) `$rowsPerPage` : Número de filas de tablas para mostrar por página
(2) `$numOfPaginationLinks` : Número de enlaces de paginación a mostrar por página

Ejemplo (s):

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

4. Llame al método `getResult()` para mostrar las filas de la tabla según la consulta de URL `?page=X` , como esto:

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

El método `getResult()` toma los siguientes parámetros,

(1) `$queryString` : cadena de consulta `SELECT`
(2) `$bindParameterArray` : Array que contiene variables o valores de enlace
(3) `$globalGetArray` : matriz `$_GET`
(4) `$keyFromURLQuery` : clave de la consulta de URL, es decir, `page`

Ejemplo (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');
```

Nota: No especifique ninguna cláusula `LIMIT` o `OFFSET` en la consulta, el script se encargará de esto.

Ahora recorra el conjunto de resultados, es decir, la matriz `$resultSet` para acceder / mostrar los detalles de las filas, como esto:

```
foreach($resultSet as $row){
    /* access/display row details */
    /* $row['column1'], $row['column2'] etc. */
}
```

Nota: Si desea ver la estructura de matriz completa, haga `var_dump($resultSet);` .

5. Mostrar enlaces de paginación utilizando el método `getPaginationLinks()` , como este:

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

El método `getPaginationLinks()` no toma ningún parámetro y devuelve una matriz de enlaces de paginación en el siguiente formato,

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

Ahora recorre la matriz `$pgLinks` para mostrar los enlaces de paginagtion, como esto:

```
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;';
}
```

Nota: Si desea ver la estructura completa de la matriz, haga `var_dump($pgLinks);` .

Nota (s) al pie: Puede aplicar estilo a las filas de resultados y a los enlaces de paginación según su elección.

Lea Empezando con la paginación en línea:
https://riptutorial.com/es/pagination/topic/7280/empezando-con-la-paginacion

Capítulo 2: Paginación con PHP y MySql.

Parámetros

Parámetro	Detalles
$ params = []	parámetros opcionales para `['table' => 'tableName', 'sort' => 'ASC', 'columns' => 'colId, name, etc']`
$ atributos = []	lista opcional de atributos `['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

Creando una Paginación Simple

`Pagination.php` incluir el `Pagination.php` en su página / 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';
```

Creación de paginación

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

Creando el archivo Paginator

```
<?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);
            }
        }
    }
}
```

ver este repositorio para más información

Lea Paginación con PHP y MySql. en línea:
https://riptutorial.com/es/pagination/topic/9914/paginacion-con-php-y-mysql-

Creditos

S. No	Capítulos	Contributors
1	Empezando con la paginación	Community, Rajdeep Paul
2	Paginación con PHP y MySql.	julekgwa