📓 Course: Pagination in PHP with PDO

Pagination is an essential technique when displaying large amounts of data, such as articles from an SQL database. It helps avoid overloading a web page and improves the user experience.

This course will guide you step by step to create a **classic pagination in PHP**, connected to a MySQL database using PDO.

📦 1. Preparing the Database

We will use the following table, already present in our beer database:

```sql
CREATE TABLE `article` (
  `ID_ARTICLE` int(11) NOT NULL,
  `NOM_ARTICLE` varchar(60) NOT NULL,
  `PRIX_ACHAT` double NOT NULL,
  `VOLUME` int(11) NOT NULL,
  `TITRAGE` double DEFAULT NULL,
  `ID_MARQUE` int(11) DEFAULT NULL,
  `ID_Couleur` int(11) DEFAULT NULL,
  `ID_TYPE` int(11) DEFAULT NULL
);
```

🧹 2. Connecting to the Database

First, let's connect to the database using PDO. We store the PDO object in a variable to use it for queries.

```php
// these connection credentials are obviously not the correct ones
$pdo = new PDO('mysql:host=localhost;dbname=ma_base;charset=utf8mb4', 'user',
'password');
```

🔢 3. Determine the Current Page

We will determine the current page by reading the `page` parameter in the URL. If not specified, we'll default to the first page.

```php
$page = isset($_GET['page']) ? (int)$_GET['page'] : 1;
// You might also check that the page is not negative.
```

Then, define the number of articles per page:

```
$articlesPerPage = 10;
```

And compute the **offset** for the SQL query:

```
$offset = ($page - 1) * $articlesPerPage;
// -1 is needed because page 1 should have an offset of 0 (start).
```

📊 4. Count Total Articles

Before fetching articles, we need the **total count** to calculate the number of pages:

```
$total = $pdo->query('SELECT COUNT(*) FROM article')->fetchColumn();
// fetchColumn retrieves a single column instead of a full row array.
$totalPages = ceil($total / $articlesPerPage);
```

🕹️ 5. Fetch Articles for the Current Page

Now that we have the offset and limit, we can fetch the articles for the requested page:

```
$stmt = $pdo->prepare('SELECT * FROM article LIMIT :limit OFFSET :offset');
$stmt->bindValue(':limit', $articlesPerPage, PDO::PARAM_INT);
$stmt->bindValue(':offset', $offset, PDO::PARAM_INT);
$stmt->execute();
// If FETCH_ASSOC is not the default, you can specify it here
$articles = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

🖥️ 6. Display Articles

Now loop through the results and display them in HTML:

```
foreach ($articles as $article) {
    echo '<p>' . htmlspecialchars($article['NOM_ARTICLE']) . ' - ' .
$article['PRIX_ACHAT'] . ' €</p>';
}
```

🔁 7. Display Pagination Links

Displaying 400 page links is not user-friendly. So, we only display a few pages **around the current one**.

Let’s create a function for that:

```php
/**
 * Generates HTML for pagination.
 *
 * @param integer $currentPage current page number
 * @param integer $totalPages total number of pages
 * @param integer $window number of pages to show before and after current page
 * @return string pagination HTML
 */
function renderPagination(int $currentPage, int $totalPages, int $window = 2):
string {

    $html = '<nav><ul class="pagination">';

    $start = max(1, $currentPage - $window);
    $end = min($totalPages, $currentPage + $window);

    if ($currentPage > 1) {
        $html .= '<li><a href="?page=' . ($currentPage - 1) . '">«</a></li>';
    }

    for ($i = $start; $i <= $end; $i++) {
        $class = $i === $currentPage ? 'class="active"' : '';
        $html .= "<li $class><a href='?page=$i'>$i</a></li>";
    }

    if ($currentPage < $totalPages) {
        $html .= '<li><a href="?page=' . ($currentPage + 1) . '">»</a></li>';
    }

    $html .= '</ul></nav>';
    return $html;
}
```

Then, display the pagination:

```php
echo renderPagination($page, $totalPages);
```

✅ Expected Result

A PHP page such as `index.php?page=3` will display:

- A **list of 10 articles** for page 3
- **Pagination links**: “Previous”, surrounding page numbers, “Next”

UX Tip

- You can style pagination using Bootstrap (`pagination`, `page-item`, `page-link`).
- You can add links to the **first and last** pages if needed.
- You can also disable the links if out of bounds.

🏁 Conclusion

This pagination technique is the foundation for any list display in PHP. It can be adapted for an API, search results, admin interfaces, and more.