📓 Cours : Pagination en PHP avec PDO

La pagination est une technique essentielle lorsqu'on affiche une grande quantité de données, comme des articles issus d'une base SQL. Elle permet de ne pas surcharger une page web et d'améliorer l'expérience utilisateur.

Ce cours va vous guider étape par étape pour créer une pagination **classique en PHP**, connectée à une base de données MySQL avec PDO.

📦 1. Préparation de la base de données

Nous allons utiliser la table suivante, déjà présente dans notre base de donnée bière :

```
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. Connexion à la base de données

Avant toute chose, connectons-nous à notre base via PDO. On stocke l'objet PDO dans une variable pour l'utiliser dans nos requêtes.

```
// ces informations de connexion ne sont pas les bonnes évidement
$pdo = new PDO('mysql:host=localhost;dbname=ma_base;charset=utf8mb4', 'user',
'password');
```

🔢 3. Déterminer la page actuelle

Nous allons déterminer la page actuelle en lisant le paramètre `page` passé dans l'URL. Si aucune page n'est précisée, on affichera la première.

```
$page = isset($_GET['page']) ? (int)$_GET['page'] : 1;
// On pourrait aussi vérifier que la page donnée n'est pas négative.
```

Ensuite, on définit le nombre d'articles à afficher par page :

```
$articlesParPage = 10;
```

Et on calcule le **décalage** (`OFFSET`) pour la requête SQL :

```
$offset = ($page - 1) * $articlesParPage;
// le -1 est là car à la page 1, on veut un décalage de 0 puisqu'on veut le début.
```

📊 4. Connaître le nombre total d'articles

Avant de récupérer les articles, on a besoin de connaître leur **nombre total** pour calculer le nombre de pages :

```
$total = $pdo->query('SELECT COUNT(*) FROM article')->fetchColumn();
// fetchColumn permet de récupérer le résultat d'un seule seule colonne plutôt
qu'un tableau de résultat.
$totalPages = ceil($total / $articlesParPage);
```

📥 5. Récupérer les articles de la page courante

Maintenant que nous avons l'offset et la limite, on peut récupérer les articles de la page demandée :

```
$stmt = $pdo->prepare('SELECT * FROM article LIMIT :limit OFFSET :offset');
$stmt->bindValue(':limit', $articlesParPage, PDO::PARAM_INT);
$stmt->bindValue(':offset', $offset, PDO::PARAM_INT);
$stmt->execute();
// Si on ne l'a pas précisé lors de la création de notre connexion PDO, on peut
indiquer le FETCH_ASSOC ici
$articles = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

💻 6. Afficher les articles

On parcourt maintenant les résultats pour les afficher en HTML. Par exemple :

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

🔁 7. Afficher les liens de pagination

Afficher toutes les pages quand on en a 400 serait peu ergonomique. On va donc n'afficher que quelques pages **autour de la page actuelle**.

Créons une fonction pour cela :

```php
/**
 * Crée le HTML pour la pagination et le retourne.
 *
 * @param integer $currentPage page actuelle
 * @param integer $totalPages nombre de page
 * @param integer $window nombres de page à afficher avant et après la page
actuelle
 * @return string la pagination à afficher en 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;
}
```

Et on affiche la pagination :

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

✅ Résultat attendu

Une page PHP accessible par exemple via `index.php?page=3` affichera :

- Une **liste de 10 articles** de la page 3
- Des **liens de pagination** : « Précédent », pages autour de la 3e, « Suivant »

🧠 Astuce UX

- Tu peux styliser la pagination avec Bootstrap (`pagination`, `page-item`, `page-link`).
- Tu peux ajouter un lien vers la **première et dernière** page si besoin.
- Tu peux aussi désactiver les liens si la page est en dehors des limites.

🏁 Conclusion

Cette technique de pagination est la base pour tout affichage de liste en PHP. Elle peut être adaptée pour une API, une recherche, une interface admin, etc.