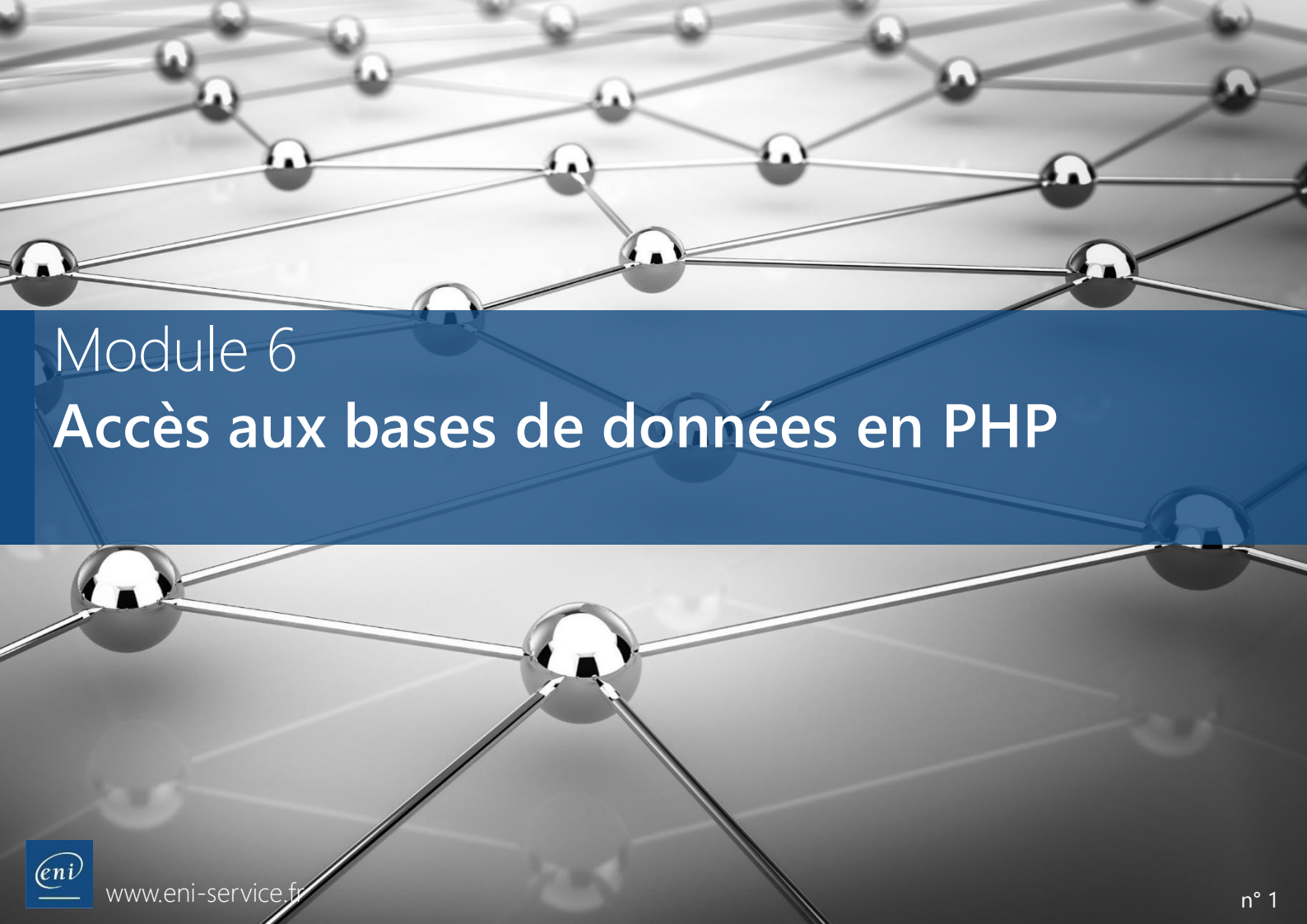


Module 6

Accès aux bases de données en PHP

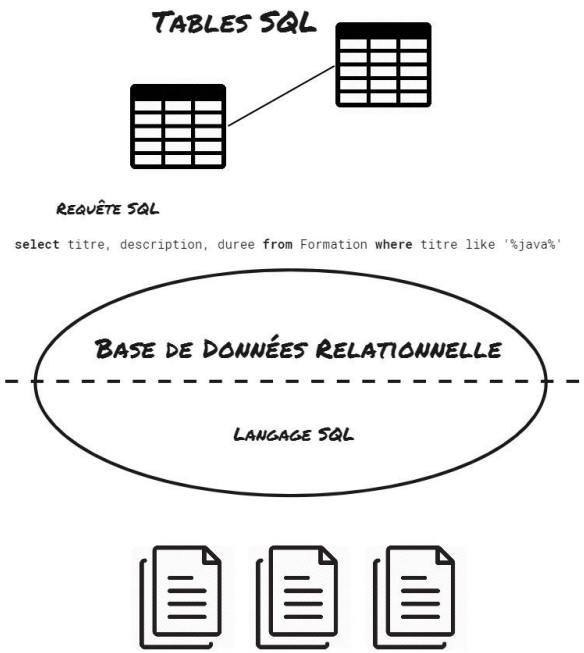
Contenu du module

- Présentation de la base de données MySQL.
  - Concepts fondamentaux et organisation du produit.
  - Les outils de gestion associés : MySQL Workbench et PHPMyAdmin.
  - L'organisation des données pour un site Web dynamique.
- Concepts d'accès aux données avec PHP.
  - Les fonctions natives.
  - La librairie PDO (PHP Data Object).
  - La configuration du moteur PHP.
- Modèle de programmation PDO
  - Démarrage
  - Requêtes
  - Résultats
  - Finalisation

Base de Données

- Application serveur qui **dissocie**
- les opérations de **manipulation des informations**
- des opérations de **lecture et d'écriture des fichiers**

- L'accès aux données se fait par le langage SQL



Présentation de la base de données MySQL

- MySQL est un système de gestion de base de données relationnelle (RDBMS) open source
  - C'est le système de base de données le plus populaire utilisé avec PHP
  - MySQL est développé, distribué et pris en charge par Oracle Corporation
- Historiquement développé par MySQL AB, l'entreprise a été rachetée par Sun Microsystems (inventeur de Java), elle-même ensuite rachetée par Oracle
  - De ces multiples rachats, est née la volonté de garder l'approche Libre et Open Source du projet, un « side-project » est né : MariaDB
- MariaDB et MySQL sont complétement compatibles
- Disposent d'une vaste bibliothèque de fonctions et d'API :
  - API pour C, C++, .NET, Eiffel, Java, Perl, PHP, Python, Ruby et Tcl sont disponibles
  - Les fonctions SQL sont mises en place en utilisant une bibliothèque de classes optimisées

Les outils de gestion associés

- Pour l'administration et la gestion du produit
- Des outils fournis avec MySQL :
  - MySQL Shell
    - Outils en ligne de commande
  - MySQL Workbench
    - Outil graphique pour créer ses structures de données, gérer le serveur, la sécurité, …
- Des outils tiers
  - phpMyAdmin
    - Application Web développée en PHP
    - Permet la gestion des schémas de données et l'administration du serveur
      - Orientée pour les développeurs PHP
    - phpMyAdmin est souvent intégré dans les packages de développement PHP
      - XAMPP, Wamp Server

Organisation des données pour un site Web dynamique

- Les données exploitées dans une application ou un site Web dynamique sont stockées dans des tables de base de données.
  - Une table correspond souvent à une structure organisée selon un concept métier
    - Un devis, une commande, un client, …
  - D'autres permettent de faire le lien entre ces objets
    - Une ligne de commande, …
- Les opérations SQL correspondent aux opérations CRUD classiques

Opération	CRUD	Instruction SQL
Ajout de données	Create	`INSERT`
Lecture de données	Read	`SELECT`
Mise à jour de données	Update	`UPDATE`
Suppression de données	Delete	`DELETE`

Concepts d'accès aux données

- Une application qui se connecte à une base de données relationnelle utilise la séquence d'étapes suivante :
  - Etablissement de la connexion
  - Préparation de la requête SQL à exécuter
  - Association des données de l'application à la requête
  - Exécution de la requête
  - Exploitation des résultats
  - Fermeture de la connexion
- Toutes ces opérations nécessitent qu'une gestion efficace des erreurs soient mise en place
  - Les exceptions levées seront systématiquement fatales pour les objets manipulés dans ce contexte.

Les fonctions natives historiques

- Historiquement, PHP possédait des fonctions natives pour les différentes bases de données supportées
  - Les fonctions `mysql_*` et `mysqli_*` pour MySQL
  - `oci_*` pour Oracle
  - …
- L'utilisation de ces fonctions n'assure aucune portabilité du code en cas de changement de base de données
  - Différence entre les environnements par exemple : dev, test, prod …

La librairie PDO (PHP Data Objects)

- PHP Data Objects
  - Extension native PHP qui définit une interface uniforme pour accéder aux bases de données en PHP
  - L'accès à une base de données à travers PDO grâce à un driver
    - Il expose les fonctionnalités de la base de données
- PDO ne fournit pas une couche d'abstraction de la base de données mais une couche d'abstraction de l'accès aux bases de données
  - Les requêtes que vous écrivez doivent respecter la syntaxe de la base de données que vous utilisez
  - PDO ne réécrit pas les requêtes SQL et n'émule pas les fonctionnalités manquantes
    - A l'exception des requêtes paramétrées si besoin
- De nombreuses bases de données disposent d'un driver PDO parmi lesquelles MySQL, et Oracle
  - Le support pour Oracle, MySQL, PostgreSQL et SQLLite est natif dans PHP7
  - L'utilisation d'autres bases de données nécessite de télécharger et installer un pilote PDO
- PDO est une extension orientée objet qui est constituée de trois classes :
  - `PDO` : Connexion entre PHP et la base de données
  - `PDOStatement` : Requête préparée, et, après exécution, résultat associé
  - `PDOException` : Exception levée par PDO

Configuration du moteur PHP

- L'utilisation de PDO peut nécessiter d'intervenir dans la configuration du moteur PHP
  - Fichier `php.ini`

```
extension=pdo_mysql
;extension=pdo_oci
;extension=pdo_odbc
;extension=pdo_pgsql
extension=pdo_sqlite
```

- Pour les autres bases de données du marché il sera nécessaire de :
  - Trouver et télécharger le pilote approprié
    - Souvent il suffit de chercher sur le site de l'éditeur
  - Installer le pilote
    - Une copie dans le répertoire des extensions PHP
  - Ajouter la ligne `extension=` dans le fichier `php.ini`

Modèle de programmation PDO – 1 - Démarrage

- Connexion à la base de données
  - Instanciation de la classe `PDO`
    - `public PDO::__construct(string $dsn, ?string $username = null, ?string $password = null, ?array $options = null)`
      - Le DSN est la chaine de connexion spécifique à chaque pilote PDO
      - Les options sont spécifiques à la base de données
    - L'objet retourné est une ressource
- Préparer la requête SQL
  - Création de la commande avec la classes `PDOStatement`
    - `public PDO::prepare(string $query, array $options = []): PDOStatement|false`
      - La méthode `prepare()` est invoquée sur l'objet PDO obtenu à la connexion
      - La requête SQL peut contenir des variables positionnelles ou nommées
        - » On parle de **requête préparées**
  - Exécuter la requête SQL
    - `public PDOStatement::execute(?array $params = null): bool`
      - Le tableau de paramètres correspond aux données remplaçant les variables de la requête

Les requêtes préparées

- Les requêtes préparées (ou requêtes précompilées) offrent plusieurs avantages :
  - Evitent les injections SQL.
  - Améliorent la lisibilité du code.
  - Améliorent l'échappement des caractères problématiques.
  - Améliorent les performances.
- Le principe est le suivant :
  - On définit la syntaxe générale de la requête en y introduisant des paramètres (des variables) pour les données qui viennent de l'application.
  - La requête est envoyée au serveur de base de données pour analyse et stockage en cache.
    - La syntaxe ne pourra donc plus être modifiée !
    - Les invocations de cette requête seront plus rapide car le serveur de base de données la connait déjà !
  - On remplace les paramètres par les données venant de l'application.
    - Données filtrées au préalable bien entendu !
  - On demande l'exécution de la requête au serveur.

Modèle de programmation PDO - 2 - Les requêtes préparées

- Avec des variables positionnelles :
  - Créer un objet PDOStatement :
    - `$stmt = $cnx->prepare("update article set prix=? where refart like ?);`
      - La première variable est à la position 1
  - Exécuter :
    - `$stmt->execute([1 => 500, 2 => "Vélo"]);`
- Avec des variables nommées :
  - Créer un objet PDOStatement :
    - `$stmt = $cnx->prepare("update article set prix=:prix where refart like :prd");`
  - Exécuter :
    - `$stmt->execute(['prix' => 500, 'prd' => "Vélo"]);`
      - Les variables sont nommées sans le préfix :

Modèle de programmation PDO - 3 - Résultats

- L'exploitation des résultats dépend de la requête exécutée
  - La méthode `execute()` renvoie un booléen selon que la requête est réussie ou non
- En cas de récupération de données, il est nécessaire de parcourir le jeu d'enregistrements résultant
  - Méthodes de l'objet `PDOStatement`
    - `fetch()`, `fetchAll()`, `fetchObject()`, `fetchColumn()`
      - `fetch()` peut être paramétrée pour fonctionner dans différents modes, équivalents aux autres fonctions `fetch*()`
    - Ces méthodes renvoient false quand il n'y a plus d'enregistrements dans le jeu
- Exemple :

```
$stmt = $cnx->prepare("SELECT ref,prd,prix FROM article");
$stmt->execute();
while($enreg = $stmt->fetch()) {
        echo $enreg['ref'] . " " . $enreg['prd'] . " " . $enreg['prix']
}
```

Modèle de programmation PDO - 4 - Finalisation

- Nécessiter de fermer la connexion
  - Positionner explicitement la référence de connexion à `null`
    - Ou attendre la perte de référence…
- Les opérations PDO sont susceptibles de lever des PDOException
  - Obligation de traiter les cas
    - La connexion ne peut être réutilisée dans ce cas
- Structure de code d'une méthode utilisant PDO :

```
$cnx = null;
try {
    // Connexion
    $cnx = new PDO(...);
    // Autres opérations PDO...

}
catch(PDOException $e) {
    // Gestion de l'erreur
}
finally {
    // Fermeture de la connexion
    $cnx = null;
}
```

Modèle de programmation PDO - 5 - Transactions

- Les pilotes PDO sont tous paramétrés par défaut pour utiliser le mode `autocommit`
  - Des transactions sont automatiquement gérées pour les opérations de modification de données
- Il est possible de piloter manuellement ces transactions
  - Démarrer une transaction explicitement
    - Annule le mode `autocommit`
    - `public PDO::beginTransaction(): bool`
  - Valider une transaction
    - En fin de bloc try
    - `public PDO::commit(): bool`
  - Annuler une transaction
    - Dans le bloc catch
    - `public PDO::rollBack(): bool`

Travaux pratiques