PHP

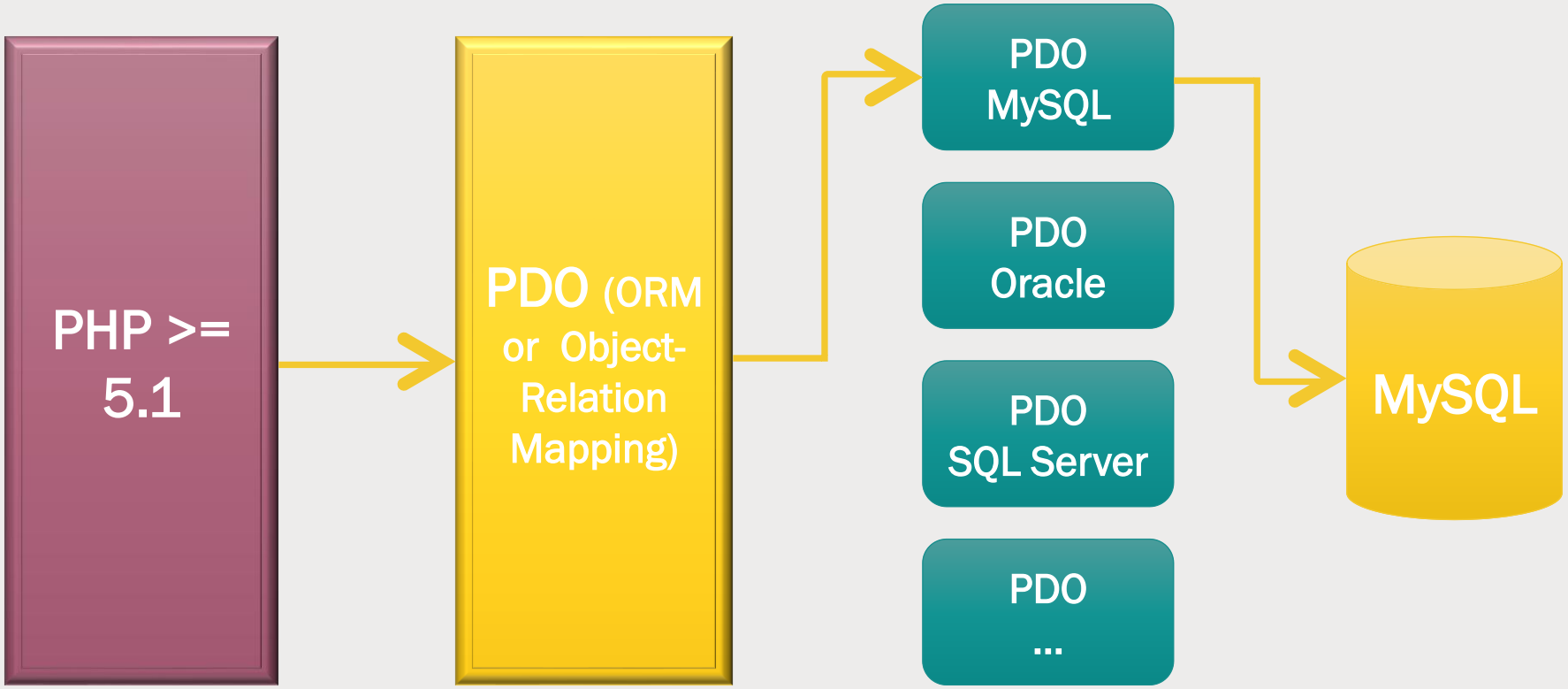
Introduction à PDO

Dans ce module

- PDO ?
- PDO Class – Introduction aux classes
- PDO Exception
- PDO Query (PDOStatement)
- PDO FetchAll
- PDO Closing connection
- Pratique

PDO ?

- PDO ou PHP Data Objects apporte une interface de code simplifiée pour les échanges entre PHP et une BDD.
- PDO supporte uniquement du relationnel : MySQL, PostgreSQL, SQL Server, Oracle, SQLite, ...



PDO ?

- PDO existe nativement dans PHP mais n'est pas toujours activé par défaut. Pour l'utiliser, il faut vérifier l'activation de l'extension "php_pdo" dans php.ini
- PDO a aussi besoin d'une extension à installer ou à activer dans les configurations de php.ini
- Extensions PHP PDO :
  - *Mysql/MariaDB => php_pdo_mysql*
  - *PostgreSQL => php_pdo_pgsql*
  - *...*
- PDO sera utilisé suivant les étapes :
  1. *Connection à la BDD*
  2. *Utiliser la connexion pour lancer une requête Query (SELECT) ou Exec (INSERT, UPDATE, ...)*
  3. *Retraitement et utilisation du résultat*
  4. *Fermeture de la connexion*

PDO Class – Introduction aux classes

- PDO est une classe. Pour le moment, nous nous contenterons des informations suivante :
  - *Une classe à un nom unique*
  - *Une classe est soit global (ex: PDO) soit doit être "importé"*
  - *Une classe doit être initialisé/construite pour être utilisé*
  - *Une classe peut posséder des propriétés qui sont des variables ou des fonctions*
  - *Pour utiliser ces propriétés, il faut les appeler par leur nom précédé du symbole "-> (tiret + chevron droit).*

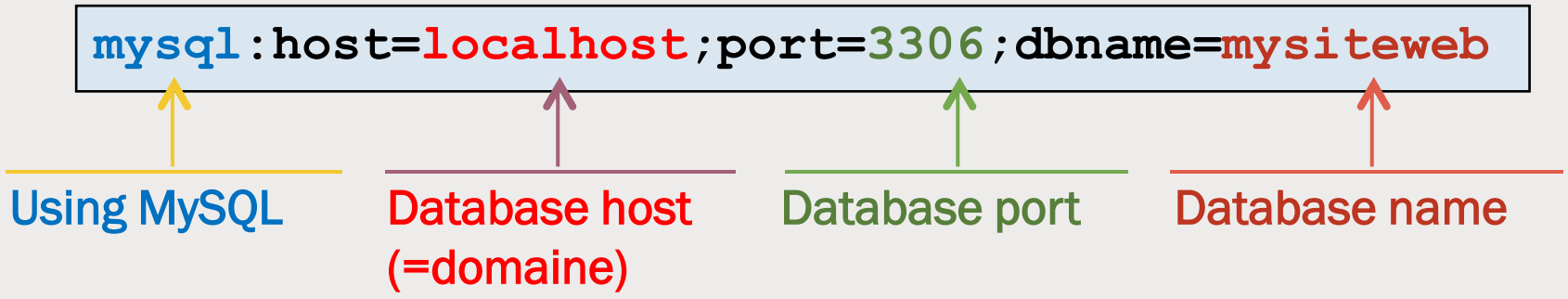
```php
// Construction de la classe
$varClass = new CLASSNAME(PARAM, PARAM, ...);

$varClass->PROTERTYNAME; // Variable de la classe
$varClass->PROTERTYNAME(); // Fonction de la classe
```

PDO Class

- L'étape de connexion à la BDD correspond en fait à la construction de la classe PDO avec des paramètres comme suit :

```php
$db = new PDO(
    'mysql:host=localhost;port=3306;dbname=mysiteweb', // DSN
    'admin', // DB user name
    'password', // DB user password
);
```



PDO Exception

- PDO étant une classe complexe servant d'interface entre PHP et la BDD, elle génère des erreurs lorsque quelques choses de va pas : erreur de connexion à la BDD, accès refusé, table introuvable, ...
- Il convient alors de contrôler les erreurs pour les identifier et les résoudre.
  - *Lorsque d'un problème/erreur survient, PHP l'intercepte et déclenche une **Exception**. Ces exceptions génèrent alors le message d'erreur que l'on voit : PHP Syntax Error, Array key out of range, ...*
- **Nous pouvons capturer les Exceptions est contrôler leur comportement** comme suit :

```php
try {
    // CODE TO TRY
} catch (Exception $e) {
    // CODE IF EXCEPTION
}
```

```php
$dsn = 'mysql:host=localhost;port=3306;dbname=mysiteweb';
try {
    $db = new PDO($dsn, 'admin', 'password');
} catch (PDOException $e) {
    echo $e->getMessage() . '<br>';
    exit();
}
```

→ *Pour PDO, "try catch" nous servira à contrôler les erreurs lors de la connections*

PDO query (PDOStatement)

- Après la connexion réussie avec PDO, il convient de préparer la requête qui sera vérifiée par la BDD. Nous avons 2 choix suivant le type de requête :
  - *query : SELECT, SHOW, ...*
  - *exec : INSERT, UPDATE, DELETE, ...*
- Il faut savoir que "query()" et "exec()" renvoie en fait soit "false" soit une autre classe qui s'appel (PDOStatement)

```php
// -- Select -- \\
$result = $db->query('SELECT * FROM users');
    // $result n'est pas utilisable tel quel
    // Si "$result === false" alors la requête a échouée

// -- Insert -- \\
$result = $db->exec("INSERT INTO users (id, email) VALUES (19, 'user19@gmail.com')");
    // $result est le nombre de row affectée
    // Si "$result === false" alors la requête a échouée
```

PDO Query

- Il faudra aussi vérifier le résultat de la base de données. Si le résultat est faux (false), c'est que la BDD ne peut pas traiter la requête. Les messages d'erreurs de la BDD sont récupérable avec "$db->errorInfo()". La fonction fournira un tableau dont le 3ème élément est le message d'erreur.

```php
// -- Select -- \\
$result = $db->query('SELECT * FROM users');
    // $result n'est pas utilisable tel quel
    // Si "$result === false" alors la requête a échouée

if ($result === false) {
    echo $db->errorInfo()[2];
    exit();
}

// -- Insert -- \\
$result = $db->exec("INSERT INTO users (id, email) VALUES (19, 'user19@gmail.com')");
    // $result est le nombre de row affectée
    // Si "$result == false" alors la requête a échouée

if ($result === false) {
    echo $db->errorInfo()[2];
    exit();
}
```

PDO FetchAll

- Dans le cas d'une query (select) et après vérification du résultat, il convient de retraiter la réponse pour obtenir des données utilisables.

```php
$result = $db->query('SELECT * FROM users');

if ($result === false) {
    echo $db->errorInfo()[2];
    exit();
}

// Récupération des données de la table users
$users = $result->fetchAll(PDO::FETCH_ASSOC);
var_dump($users);
```

```
array (size=3)
  0 =>
    array (size=2)
      'id' => string '1'
      'email' => string 'user1@gmail.com'
  1 =>
    array (size=2)
      'id' => string '2'
      'email' => string 'user2@gmail.com'
  2 =>
    array (size=2)
      'id' => string '19'
      'email' => string 'user19@gmail.com'
```

PDO Closing

- Il convient de fermer les connexions à la BDD si on ne sent sert plus. Sinon par défaut les connexions sont arrêtées à la fin de l'exécution de l'instance PHP.

```php
// Connexion à la BDD
$dsn = 'mysql:host=localhost;port=3306;dbname=my_site_web';
try {
    $db = new PDO($dsn, 'admin', 'password');
} catch (PDOException $e) {
    echo $e->getMessage() . '<br>';
    exit();
}

$resultUsersQuery = $db->query('SELECT * FROM users'); // Préparation
// --> Check $resultUsersQuery
$users = $resultUsersQuery->fetchAll(PDO::FETCH_ASSOC); // Récupération
$resultUsersQuery = null; // Closing : PDOStatement
var_dump($users); // Utilisation

$resultOrdersQuery = $db->query('SELECT * FROM orders'); // Préparation
// --> Check $resultOrdersQuery
$orders = $resultOrdersQuery->fetchAll(PDO::FETCH_ASSOC); // Récupération
$resultOrdersQuery = null; // Closing : PDOStatement
var_dump($orders); // Utilisation

// Fermeture des connections PDO
$db = null; // Closing : PDO
// $users et $orders ne sont pas des connexions
```

Pratique

- ✓ *Quand on parle de **page**, c'est pour un fichier php qui affiche de l'**HTML mais contient du php pour un affichage dynamique ou pour des droits d'accès à la page.***
- ✓ *Quand on parle de **script**, c'est pour un fichier php qui **traite des données et redirige** l'utilisateur sur une page après traitement.*
- ✓ *Pensez à utiliser des fonctions et des fichiers organisés pour tout code qui serait souvent répété.*

- ❖ Créez une nouvelle base de données "mywebsite" et une table "users" avec les champs "id", "email", "password" et "name". Ajoutez-y un utilisateur pour la suite
- ❖ Créez un mini-site avec une authentification reliée à votre base de données. Vous devrez utiliser les exemples et les connaissances du cours pour réaliser ce mini-site. Le but est d'avoir une page home.php qui n'est accessible que par un utilisateur que l'on considère connecté. L'authentification est un concept de droit d'accès, il y a donc plusieurs solutions à l'application de ce concept avec les éléments des cours (Session).
- ❖ Vous devez obligatoirement stocker vos utilisateurs dans une base de données.
- ❖ Il est demandé les fichiers suivants :
  - ➢ 1 page d'accueil simpliste avec les liens vers les autres pages
  - ➢ 1 page de login à part avec le formulaire (email + password)
  - ➢ 1 script de traitement du formulaire de login avec les redirections nécessaires
  - ➢ 1 page home pour les utilisateurs connectés. Sur cette page home, il serait intéressant de réfléchir à un bouton (lien) de déconnexion (logout) pour déconnecter l'utilisateur et de le rediriger sur la page d'accueil. Il est conseillé de créer un nouveau script pour le logout.

Pratique

- Les liens affichés dans le menu ne peuvent pas être les mêmes entre un utilisateur déconnecté et un utilisateur connecté. Faites le nécessaire en affichant que des liens correspondant à l'état de l'utilisateur.
- Ajoutez maintenant :
  - 1 page register pour l'inscription de nouveau utilisateur.
  - 1 script traitant le formulaire du register. Si les données du formulaire de register répondent à vos attentes pour l'inscription, enregistrer ce nouvel utilisateur en base de données puis connecter-le directement avant de le renvoyer sur la page home.
- Il peut, ensuite, être intéressant d'améliorer les comportements du site. Par exemple, un utilisateur connecté ne devrait plus pouvoir accéder à la page de connexion (login) et d'inscription (register).
- Si ça n'a pas été fait, vous pouvez afficher l'email et le nom de l'utilisateur connecté sur la page home. Pour cela, vous avez peut-être pensé à utiliser la session pour stoker ces informations ? Mais il est plutôt conseillé d'utiliser une requête BDD à chaque appel de la page. *Ce n'est pas optimisé ?* Oui mais pourquoi à votre avis on fait un appel BDD à chaque page d'un utilisateur connecté ?
- Le mot de passe est en clair dans votre BDD. Trouvez une solution pour améliorer le stockage des mots de passe. Aide de recherche : « hash de mot de passe en php »

PHP

Introduction à PDO

Fin du module