Développement PHP
Partie 4 : Bases de données

DENIS LOEUILLET – IFA - 2017

Partie 4 : Bases de données

- Présentation des bases de données
- phpMyAdmin
- Lire les données
- Écrire des données
- Les fonctions SQL
- Les dates en SQL
- Les jointures entre tables

Partie 4 : Bases de données
Lire les données

1. Se connecter à la base de données en PHP
2. Récupérer les données
3. Les critères de sélection
4. Construire des requêtes en fonction de variables
5. Traquer les erreurs

Partie 4 : Bases de données
Lire les données

- Nous allons apprendre à communiquer avec une base de données via PHP.
- Le langage utilisé est le langage SQL
- Nous allons nous entraîner à lire des données dans une table.
- Il est vivement conseillé d'avoir un peu manipulé phpMyAdmin au préalable : cet outil vous permettra de vérifier si les manipulations que vous faites en PHP ont bien l'impact que vous attendiez dans votre base de données.

Partie 4 : Bases de données
Lire les données

1. Se connecter à la base de données en PHP
   - Comme pour lire dans un fichier, pour travailler avec une BDD il faut d'abord s'y connecter
   - PHP fait l'intermédiaire entre vous et MySQL
     - PHP doit connaître :
       - ✓ User
       - ✓ Password
     - PHP pourra alors s'authentifier auprès de MySQL :
       - ✓ il établit une connexion avec MySQL
     - Une fois la connexion établit on pourra travailler avec la BDD

Partie 4 : Bases de données
Lire les données : se connecter à la BDD

- Comment se connecter à la BDD en PHP ?
  - PHP propose plusieurs solutions :
    - ✓ L'extension « mysql_ » :
      - This extension is deprecated as of PHP 5.5.0, and has been removed as of PHP 7.0.0.
      - ce sont des fonctions qui permettent d'accéder à une BDD MySQL et donc de communiquer avec MySQL.
      - leur nom commence toujours par « mysql_ ».
      - Toutefois, ces fonctions sont vieilles et on recommande de ne plus les utiliser aujourd'hui.
    - ✓ L'extension « mysqli_ » :
      - ce sont des fonctions améliorées d'accès à MySQL.
      - elles proposent plus de fonctionnalités et sont plus à jour.
    - ✓ L'extension « PDO » :
      - c'est un outil complet qui permet d'accéder à n'importe quel type de base de données.
      - on peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle.

Partie 4 : Bases de données
Lire les données : se connecter à la BDD

- Comment se connecter à la BDD en PHP ?
  - Quel moyen choisir parmi tous ceux-là ?
    - ✓ les fonctions « mysql_ » ne sont plus à utiliser : elles sont « obsolètes »
    - ✓ Il reste « mysqli » et « PDO »
      - nous allons ici utiliser « PDO » car c'est la solution d'avenir pour les prochaines versions de PHP
      - PDO présente l'avantage de fonctionner pour d'autres type de SGBD
      - on pourra donc utiliser ce qu'on va voir pour ORACLE ou PostgreSQL

Partie 4 : Bases de données
Lire les données : se connecter à la BDD

- Se connecter à MySQL avec PDO
  - 4 paramètres sont nécessaires :
    - ✓ le nom de l'hôte :
      - ❖ c'est l'adresse de l'ordinateur où MySQL est installé (comme une adresse IP).
      - ❖ si MySQL est installé sur le même ordinateur que PHP : mettez la valeur « localhost »
    - ✓ le nom de la base de données
    - ✓ Le login
    - ✓ Le password
  - Syntaxe :

```php
<?php

$bdd = new PDO('mysql:host=localhost;dbname=sta15;charset=utf8', 'sta15', 'xxxxxxxxxx');

?>

```

Partie 4 : Bases de données
Lire les données : se connecter à la BDD

- Se connecter à MySQL avec PDO

```php
<?php

$bdd = new PDO('mysql:host=localhost;dbname=sta15;charset=utf8', 'sta15', 'xxxxxxxxxx');

?>

```

  - PDO est une extension ***orientée objet***
    - ✓ C'est une façon de programmer un peu différente des fonctions classiques que nous étudierons plus tard
  - La ligne de code qu'on vient de voir crée ce qu'on appelle un objet $bdd :
    - ✓ Ce n'est pas vraiment une variable : c'est un objet qui représente la connexion à la base de données.
    - ✓ On crée la connexion en indiquant dans l'ordre dans les paramètres :
      - ❖ le nom d'hôte : localhost
      - ❖ la BDD : sta15
      - ❖ le login : sta15
      - ❖ le password : xxxxxx

Partie 4 : Bases de données
Lire les données : se connecter à la BDD

- Se connecter à MySQL avec PDO

```php
<?php

$bdd = new PDO('mysql:host=localhost;dbname=sta15;charset=utf8', 'sta15', 'xxxxxxxxxx');

?>

```

  - Le premier paramètre (qui commence par « mysql ») s'appelle le DSN : Data Source Name.
  - C'est généralement le seul qui change en fonction du type de base de données auquel on se connecte.

Partie 4 : Bases de données
Lire les données : se connecter à la BDD

- Tester la présence d'erreurs :
  - s'il y a une erreur (vous vous êtes trompés de mot de passe ou de nom de base de données, par exemple), PHP risque d'afficher toute la ligne qui pose l'erreur, ce qui inclut le mot de passe !
  - Il est préférable de traiter l'erreur :
    - ✓ En cas d'erreur, PDO renvoie ce qu'on appelle une ***exception*** qui permet de « capturer » l'erreur.
    - ✓ Voici comment je vous propose de faire :

```
7   try
8   {
9       // On se connecte à MySQL
10      $bdd = new PDO('mysql:host=localhost;dbname=sta15;charset=utf8', 'sta15', 'xxxxxxxxx');
11  }
12  catch(Exception $e)
13  {
14      // En cas d'erreur, on affiche un message et on arrête tout
15      die('Erreur : '.$e->getMessage());
16  }
17
```

Partie 4 : Bases de données
Lire les données : se connecter à la BDD

- Tester la présence d'erreurs :

```
7   try
8   {
9       // On se connecte à MySQL
10      $bdd = new PDO('mysql:host=localhost;dbname=sta15;charset=utf8', 'sta15', 'xxxxxxxxx');
11  }
12  catch(Exception $e)
13  {
14      // En cas d'erreur, on affiche un message et on arrête tout
15      die('Erreur : '.$e->getMessage());
16  }
17
```

  - PHP essaie d'exécuter les instructions à l'intérieur du bloc « ***try*** »
    http://php.net/manual/fr/language.exceptions.php
  - S'il y a une erreur, il exécute les instructions du bloc « ***catch*** », ici on arrête l'exécution de la page et on affiche un message décrivant l'erreur
  - Si tout se passe bien :
    - ✓ PHP poursuit l'exécution du code et ne lit pas ce qu'il y a dans le bloc « ***catch*** ».

Partie 4 : Bases de données
Lire les données : se connecter à la BDD

- Tester la présence d'erreurs :
  - PDO nous fait utiliser des fonctionnalités de PHP que l'on n'a pas étudiées jusqu'à présent (programmation orientée objet, exceptions…)
  - Contentez-vous pour le moment de réutiliser les codes que je vous propose
    - ✓ nous reviendrons sur ces codes-là plus tard pour les expliquer en détail.

Partie 4 : Bases de données
Lire les données

2. Récupérer les données
   - Dans un premier temps, nous allons apprendre à lire des informations dans la BDD
   - puis nous verrons dans le chapitre suivant comment ajouter et modifier des données.
   - Pour illustrer, il nous faut une table « toute prête » qui va nous servir de support :
     - cette table contient une liste d'une cinquantaine de jeux vidéo.
     - Pour cet exemple, plusieurs amis ont voulu répertorier tous les jeux vidéo qu'ils possèdent. La base de données est pour eux un moyen très pratique de classer et d'organiser tout cela

Partie 4 : Bases de données
Lire les données : récupérer les données

- Faire une requête :
  - Requête : demander à MySQL de faire des actions sur une table
    - ✓ Insérer des données
    - ✓ Supprimer des données
    - ✓ Modifier des données
    - ✓ Lister une partie des données
    - ✓ …
  - Pour récupérer des informations de la BDD, nous avons besoin de notre objet représentant la connexion à la base : $bdd.
    - ✓ Nous allons effectuer la requête comme ceci :

```
1  $reponse = $bdd->query('Tapez votre requête SQL ici');
```

      - ❖ On demande ainsi à effectuer une requête sur la BDD.
      - ❖ « ***query*** » signifie « requête »

Partie 4 : Bases de données
Lire les données : récupérer les données

- Faire une requête :

```
1  $reponse = $bdd->query('Tapez votre requête SQL ici');
```

  - On récupère ce que la BDD nous a renvoyé dans un autre objet que l'on a appelé ici « $reponse ».

- Exemple de requête :

```
1  SELECT * FROM jeux_video
```

  - Sélectionner tout ce qui figure dans la table « jeux_video » :
    - ✓ SELECT :
      - ❖ en langage SQL, le premier mot indique quel type d'opération doit effectuer MySQL.
      - ❖ Ce mot-clé demande à MySQL d'afficher ce que contient une table : récupérer des données
    - ✓ * :
      - ❖ après leSELECT, on doit indiquer quels champs MySQL doit récupérer dans la table.
      - ❖ Si on n'est intéressé que par les champs « nom » et « possesseur », il faudra taper :
        - SELECT nom, possesseur FROM jeux_video
      - ❖ pour sélectionner tous les champs on utilise « * »

Partie 4 : Bases de données
Lire les données : récupérer les données

- Exemple de requête :

```
1  SELECT * FROM jeux_video
```

  - Sélectionner tout ce qui figure dans la table « jeux_video » :
    - ✓ FROM :
      - ❖ c'est un mot de liaison qui se traduit par « dans ».
      - ❖ FROM fait la liaison entre le nom des champs et le nom de la table.
    - ✓ jeux_video :
      - ❖ c'est le nom de la table dans laquelle il faut aller piocher.
  - la requête avec PHP

```php
<?php
$reponse = $bdd->query('SELECT * FROM jeux_video');
?>
```

    - ✓ $reponse contient maintenant la réponse de MySQL.

Partie 4 : Bases de données
Lire les données : récupérer les données

- Afficher le résultat d'une requête
  - Le problème, c'est que $reponse contient quelque chose d'inexploitable.
  - MySQL nous renvoie beaucoup d'informations qu'il faut organiser.
  - Pour ne pas tout traiter d'un coup, on extrait cette réponse ligne par ligne, c'est-à-dire entrée par entrée.
  - Pour récupérer une entrée, on prend la réponse de MySQL et on y exécute « fetch() », ce qui nous renvoie la première ligne.
    http://php.net/manual/fr/pdostatement.fetch.php
    - ✓ fetch en anglais signifie « va chercher ».
    - ✓ $donnees est un array qui contient champ par champ les valeurs de la première entrée.
      - ❖ Par exemple, si vous vous intéressez au champ « console », vous utiliserez l'array $donnees['console'].
    - ✓ Il faut faire une boucle pour parcourir les entrées une à une :
      - ❖ Chaque fois que vous appelez $reponse->fetch(), vous passez à l'entrée suivante.
      - ❖ La boucle est donc répétée autant de fois qu'il y a d'entrées dans votre table.

```php
<?php
$donnees = $reponse->fetch();
?>
```

Partie 4 : Bases de données
Lire les données : récupérer les données

- Exemple (vois script)
  - On fait une boucle pour chaque entrée de la table.
  - On commence par l'entrée n°1, puis l'entrée n°2, etc.
  - Chaque fois qu'on fait une nouvelle boucle, on passe en revue une autre entrée.
  - Quelle est la différence entre $reponse et $donnees ?
    - ✓ $reponse contenait toute la réponse de MySQL en vrac, sous forme d'objet. (inexploitable)
    - ✓ $donnees est un array renvoyé par fetch().
      - ❖ Chaque fois qu'on fait une boucle,fetch va chercher dans $reponse l'entrée suivante et organise les champs dans l'array $donnees.
    - ✓ Ligne : while ($donnees = $reponse->fetch ()) :
      - ❖ Cette ligne fait deux choses à la fois :
        - elle récupère une nouvelle entrée et place son contenu dans $donnees ;
        - elle vérifie si $donnees vaut vrai ou faux.
      - ❖ Fetch renvoie faux (false) dans $donnees lorsqu'il est arrivé à la fin des données, c'est-à-dire que toutes les entrées ont été passées en revue :
        - Dans ce cas, la condition du while vaut faux et la boucle s'arrête.

Partie 4 : Bases de données
Lire les données : récupérer les données

- Exemple (vois script)
  - ✓ Ligne : $reponse->closeCursor(); : http://php.net/manual/fr/pdostatement.closecursor.php
    - ❖ Elle provoque la « fermeture du curseur d'analyse des résultats ».
      - vous devez effectuer cet appel à « closeCursor() » chaque fois que vous avez fini de traiter le retour d'une requête
      - afin d'éviter d'avoir des problèmes à la requête suivante.
      - Cela veut dire qu'on a terminé le travail sur la requête.

Partie 4 : Bases de données
Lire les données : récupérer les données

- Afficher seulement le contenu de quelques champs
  - si on avait voulu lister les noms des jeux, on aurait utilisé la requête SQL suivante :
  - Voir exemple (script)
    - ✓ Ce code est très semblable au précédent

```
1  SELECT nom FROM jeux_video
```

- À retenir :
  - la connexion à la base de données n'a besoin d'être faite qu'une seule fois, au début de la page ;
  - il faut fermer les résultats de recherche avec « closeCursor() » après avoir traité chaque requête.

Partie 4 : Bases de données
Lire les données

3. Les critères de sélection
   - Imaginons que je souhaite obtenir uniquement la liste des jeux disponibles de la console « Nintendo 64 » et les trier par prix croissants.
     - En modifiant nos requêtes SQL :
       - ✓ il est possible de filtrer et trier très facilement vos données.
       - ✓ Nous allons nous intéresser ici aux mots-clés suivants du langage SQL :
         - WHERE
         - ORDER BY
         - LIMIT

Partie 4 : Bases de données
Lire les données : les critères de sélection

- WHERE : trier vos données
  - Si on veut lister uniquement les jeux appartenant à Patrick :
    - ✓ La requête au début sera la même qu'avant, mais je rajouterai à la fin :
      - ❖ WHERE possesseur='Patrick'.
    - ✓ Cela nous donne la requête :

```
1  SELECT * FROM jeux_video WHERE possesseur='Patrick'
```

      - ❖ Traduction : « Sélectionner tous les champs de la table jeux_video lorsque le champ possesseur est égal à Patrick ».
        - pour délimiter les chaînes de caractères on doit les placer entre apostrophes
        - En revanche, elles ne sont pas nécessaires pour les nombres.
  - Exemple scripte where.php

Partie 4 : Bases de données
Lire les données : les critères de sélection

- WHERE : trier vos données
  - Exemple scripte where.php
    - ✓ Si on change le nom du possesseur
      - ❖ WHERE possesseur='Michel : par exemple
        - ○ ça n'affichera que les jeux appartenant à Michel.
  - Il est possible de combiner plusieurs conditions :
    - ✓ On veux lister les jeux de Patrick qu'il vend à moins de 20 euros :
      - ❖ je combinerai les critères de sélection à l'aide du mot-clé « AND » (qui signifie « et ») :

```
1  SELECT * FROM jeux_video WHERE possesseur='Patrick' AND prix < 20
```

        - ○ Traduction : « Sélectionner tous les champs de jeux_video lorsque le possesseur est Patrick ET lorsque le prix est inférieur à 20 ».
      - ❖ Remarque : Il existe aussi le mot-clé « OR » qui signifie « ou ».

Partie 4 : Bases de données
Lire les données : les critères de sélection

- ORDER BY : permet d'ordonner les résultats
  - On peut, par exemple, classer les résultats en fonction de leur prix :

```
1  SELECT * FROM jeux_video ORDER BY prix
```

    - ✓ Traduction : « Sélectionner tous les champs de jeux_video et ordonner les résultats par prix croissants ».
  - Exemple script order_by.php
  - Pour classer par ordre décroissant :
    - ✓ Ajouter le mot-clé « DESC » à la fin

```
1  SELECT * FROM jeux_video ORDER BY prix DESC
```

      - ❖ Traduction : « Sélectionner tous les champs de jeux_video, et ordonner les résultats par prix décroissants ».

Partie 4 : Bases de données
Lire les données : les critères de sélection

- LIMIT : permet de ne sélectionner qu'une partie des résultats
  - Utile pour faire de la pagination, par exemple
  - À la fin de la requête il faut ajouter :
    - ✓ le mot-clé « LIMIT »
    - ✓ Suivi de deux nombre séparés par une virgule :

```
1  SELECT * FROM jeux_video LIMIT 0, 20
```

      - ❖ On indique tout d'abord à partir de quelle entrée on commence à lire la table.
        - Dans notre exemple « 0 », ce qui correspond à la première entrée.
      - ❖ Ensuite, le deuxième nombre indique combien d'entrées on doit sélectionner.
        - Dans notre exemple « 20 », on prendra donc vingt entrées.
  - Exemples :
    - ✓ LIMIT 0, 20 : affiche les vingt premières entrées ;
    - ✓ LIMIT 5, 10 : affiche de la sixième à la quinzième entrée ;
    - ✓ LIMIT 10, 2 : affiche la onzième et la douzième entrée.

Partie 4 : Bases de données
Lire les données : les critères de sélection

- LIMIT : permet de ne sélectionner qu'une partie des résultats
  - Exemple script limit.php
- Exemple complet script criteres_de_selection.php :

```
1 SELECT nom, possesseur, console, prix FROM jeux_video WHERE console='Xbox' OR
  console='PS2' ORDER BY prix DESC LIMIT 0,10
```

  - Il faut utiliser les mots-clés dans l'ordre indiqué :
    - ✓ WHERE
    - ✓ Puis ORDER BY
    - ✓ puisLIMIT
    - ✓ sinon MySQL ne comprendra pas votre requête.

Partie 4 : Bases de données
Lire les données : les critères de sélection

- Exemple complet script criteres_de_selection.php :
  - Modifier la requête pour :
    - ✓ Classement par prix croissant
    - ✓ Classement par nom
    - ✓ Classement par nbre_joueurs_max
  - Créez une page qui affiche à chaque clic sur un lien les 10 enregistrements suivants de la première requête :
    - ✓ Au début les 10 premières
    - ✓ Ensuite le 10 suivantes
    - ✓ …

Partie 4 : Bases de données
Lire les données

4. Construire des requêtes en fonction des variables
   - Utiliser des variables PHP dans les requêtes
   - La mauvaise idée : concaténer une variable dans une requête

```php
<?php
$reponse = $bdd->query('SELECT nom FROM jeux_video WHERE possesseur=\'Patrick\'');
?>
```

   - cette requête récupère la liste des jeux appartenant à Patrick
   - Au lieu de toujours afficher les jeux de Patrick :
     - ✓ on aimerait que cette requête soit capable de s'adapter au nom de la personne défini dans une variable, par exemple $_GET['possesseur'].
     - ✓ Ainsi la requête pourrait s'adapter en fonction de la demande de l'utilisateur !

Partie 4 : Bases de données
Lire les données : requêtes en fonction variables PHP

- On pourrait concaténer la variable dans la requête

```php
<?php
$reponse = $bdd->query('SELECT nom FROM jeux_video WHERE possesseur=\'' . $_GET['possesseur'] . '\'');
?>
```

  - Il est nécessaire d'entourer la chaîne de caractères d'apostrophes d'où la présence des antislashs pour insérer les apostrophes :\'
  - ce code fonctionne
  - c'est l'illustration parfaite de ce qu'il ne faut pas faire
  - En effet, si la variable $_GET['possesseur'] a été modifiée par un visiteur
    - ✓ il y a un gros risque de faille de sécurité qu'on appelle injection SQL.
    - ✓ Un visiteur pourrait s'amuser à insérer une requête SQL au milieu de la vôtre et potentiellement lire tout le contenu de votre base de données, comme par exemple la liste des mots de passe de vos utilisateurs.
    - ✓ ***NE JAMAIS FAIRE CONFIANCE AUX DONNEES RECUES***

Partie 4 : Bases de données
Lire les données : requêtes en fonction variables PHP

- Les requêtes préparées
  - Le système de requêtes préparées a l'avantage :
    - ✓ d'être beaucoup plus sûr
    - ✓ mais aussi plus rapide pour la base de données si la requête est exécutée plusieurs fois.
    - ✓ À utiliser si vous voulez adapter une requête en fonction d'une ou plusieurs variables.
  - Avec des marqueurs « ? »
    - ✓ Dans un premier temps, on va « préparer » la requête sans sa partie variable, que l'on représentera avec un marqueur sous forme de point d'interrogation :

```php
<?php
$req = $bdd->prepare('SELECT nom FROM jeux_video WHERE possesseur = ?');
?>
```

❖ Au lieu d'exécuter la requête avec query() comme la dernière fois, on appelle ici prepare().

❖ La requête est alors prête, sans sa partie variable. Maintenant, nous allons exécuter la requête en appelant « execute » et en lui transmettant la liste des paramètres :

```php
<?php
$req = $bdd->prepare('SELECT nom FROM jeux_video WHERE possesseur = ?');
$req->execute(array($_GET['possesseur']));
?>
```

Partie 4 : Bases de données
Lire les données : requêtes en fonction variables PHP

- Les requêtes préparées
  - Avec des marqueurs « ? »

```php
<?php
$req = $bdd->prepare('SELECT nom FROM jeux_video WHERE possesseur = ?');
$req->execute(array($_GET['possesseur']));
?>
```

    - La requête est alors exécutée à l'aide des paramètres que l'on a indiqués sous forme d'array.
    - S'il y a plusieurs marqueurs, il faut indiquer les paramètres dans le bon ordre :

```php
<?php
$req = $bdd->prepare('SELECT nom FROM jeux_video WHERE possesseur = ? AND prix <= ?');
$req->execute(array($_GET['possesseur'], $_GET['prix_max']));
?>
```

        - Le premier point d'interrogation de la requête sera remplacé par le contenu de la variable$_GET['possesseur']
        - le second par le contenu de$_GET['prix_max'].
        - Le contenu de ces variables aura été automatiquement sécurisé pour prévenir les risques d'injection SQL.

Partie 4 : Bases de données
Lire les données : requêtes en fonction variables PHP

- Exemple : script requete_prepare_marqueurs.php
  - Lister les jeux :
    - ✓ appartenant à une personne
    - ✓ dont le prix ne dépasse pas une certaine somme
    - ✓ Bien que la requête soit « sécurisée » (ce qui élimine les risques d'injection SQL) :
      - il faudrait quand même vérifier que :
        - $_GET['prix_max'] contient bien un nombre
        - $_GET['prix_max'] est compris dans un intervalle correct.
      - Vous n'êtes pas dispensés d'effectuer des vérifications supplémentaires si vous estimez que cela est nécessaire.

Partie 4 : Bases de données
Lire les données : requêtes en fonction variables PHP

- Les requêtes préparées
  - Avec des marqueurs nominatifs
    - ✓ Si la requête contient beaucoup de parties variables, il peut être plus pratique de nommer les marqueurs plutôt que d'utiliser des points d'interrogation.

```php
<?php
$req = $bdd->prepare('SELECT nom, prix FROM jeux_video WHERE possesseur = :possesseur AND prix <= :prixmax');
$req->execute(array('possesseur' => $_GET['possesseur'], 'prixmax' => $_GET['prix_max']));
?>
```

      - Les points d'interrogation ont été remplacés par les marqueurs nominatifs
        - :possesseur
        - :prixmax
        - ils commencent par le symbole deux-points
      - Cette fois-ci, ces marqueurs sont remplacés par les variables à l'aide d'un array associatif.
      - Quand il y a beaucoup de paramètres, cela permet parfois d'avoir plus de clarté.
      - Contrairement aux points d'interrogation, nous ne sommes cette fois plus obligés d'envoyer les variables dans le même ordre que la requête.

Partie 4 : Bases de données
Lire les données

4. Traquer les erreurs
   - Lorsqu'une requête SQL « plante », bien souvent PHP vous dira qu'il y a eu une erreur à la ligne du fetch :

```
Fatal error: Call to a member function fetch() on a non-object in
```

   - Ce n'est pas la ligne du fetch qui est en cause :
     - ✓ c'est souvent la requête SQL quelques lignes plus haut.
     - ✓ Pour afficher des détails sur l'erreur, il faut activer les erreurs lors de la connexion à la base de données via PDO.

```php
<?php
$bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', '',
array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
?>
```

     - ❖ toutes les requêtes SQL qui comportent des erreurs les afficheront avec un message beaucoup plus clair.

Partie 4 : Bases de données
Lire les données : traquer les erreurs

- Exemple :
  - Supposons par exemple que j'écrive mal le nom du champ :

```php
<?php
$reponse = $bdd->query('SELECT champinconnu FROM jeux_video');
?>
```

  - L'erreur suivante s'affichera alors :

```
Unknown column 'champinconnu' in 'field list'
```

    - ✓ cela signifie : « La colonne champinconnu est introuvable dans la liste des champs ».
  - pensez toujours à activer les erreurs lors de la connexion à la base de données, cela vous permettra d'avoir un message d'erreur détaillé.

Partie 4 : Bases de données
Lire les données

- Résumé
  - Pour dialoguer avec MySQL depuis PHP, on fait appel à l'extension PDO de PHP.
  - Avant de dialoguer avec MySQL, il faut s'y connecter :
    - ✓ On a besoin :
      - ❖ de l'adresse IP de la machine où se trouve MySQL
      - ❖ du nom de la base de données ainsi
      - ❖ d'un login et d'un mot de passe.
  - Les requêtes SQL commençant par SELECT permettent de récupérer des informations dans une base de données.
  - Il faut faire une boucle en PHP pour récupérer ligne par ligne les données renvoyées par MySQL.
  - Le langage SQL propose de nombreux outils pour préciser nos requêtes, à l'aide notamment des mots-clés WHERE(filtre),ORDER BY(tri) et LIMIT(limitation du nombre de résultats).
  - Pour construire une requête en fonction de la valeur d'une variable, on passe par un système de requête préparée qui permet d'éviter les dangereuses failles d'injection SQL.

Partie 4 : Bases de données
Lire les données

- Exercice :
  - Modifier votre script criteres_de_selection.php en prenant en compte l'ensemble des informations du chapître,