Architectures et technologies WEB

Introduction aux technologies WEB

V01.00 17/08/2016 - MAURICE

Déroulement du cours

- *Chapitre 1 : Préambule*
- *Chapitre 1 : Les technologies WEB*
- *Chapitre 2 : La Programmation cliente*
- *Chapitre 3 : Le modèle MVC / Les templates*
- Chapitre 4 : La sécurité
- *Chapitre 5 : Les bases de PHP / Le Contrôleur*
- Chapitre 6 : **PHP 2ème partie / ORM**
- Chapitre 7 : SOA / ROA
- Chapitre 8 : Le référencement

Plan de la séance

- Chapitre 6 : PHP
  - COOKIE
  - SESSION
  - ORM

Les Cookies

Correspond à de petits fichiers envoyés par le serveur et stockés coté client (dans le navigateur)

- les cookies sont de simples fichiers textes contenant des chaines de la forme **nom=valeur** qui sont stockés dans un sous-repertoire d'un navigateur sur la machine client
- Une URL est aussi associée au cookie ce qui permet au navigateur de déterminer s'il doit associer un cookie avec une requête vers un serveur

- Les cookies sont soumis à un certain nombre de contraintes:
  - Leur nombre total est limité à 300 ;
  - La taille maximal d'un cookie est de 4 ko ;
  - Il ne peut exister au maximum que 20 cookies par domaine.
- À chaque visite de page dans un même site le client accompagne la requête HTTP du cookie
- Les cookies sont véhiculés dans les entetes HTTP
- Un cookie peut avoir une durée de vie instantanée ou très longue

Les Cookies

- Une page PHP peut installer un cookie par appel à la fonction SetCookie()
  - SetCookie("MonCookie", "valeur"); //détruit à la fermeture du navigateur
  - SetCookie("MonCookie", $valeur, time()+3600); /* expire dans une heure */

Exemple :

```
<?
$count++;
setCookie("count", $count);
… (a séparer en 2 dans un source PHP)
Echo "Bonjour, vous etes venu sur ce site $_COOKIE['count'] fois!";
?>
```

Syntaxe :

```
bool setcookie ( string $name [, string $value [, int $expire [, string $path [, string $domain [, bool $secure [, bool $httponly]]]]]] )
```

Les Cookies

- Expiration: détermine le moment à partir duquel le cookie sera effacé du disque du client.
  - Elle doit être passée sous forme d'un entier indiquant le nombre de secondes à compter du 1er janvier 1970 à partir desquelles le cookies n'est plus valide.
  - Il est généralement d'usage d'utiliser la fonction now() qui retourne le nombre de secondes de cette date à maintenant et d'y ajouter le nombre de secondes de validités que l'on désire.
  - Pour un cookie valide pendant un an ce sera now()+31536000.
- Chemin désigne le répertoire à partir de la racine de votre domaine pour lequel votre cookie est valide, c'est-à-dire que si vous ne voulez utiliser le cookie que dans des scripts stockés dans le répertoire /commerce il faudra définir le chemin /commerce/

Les Cookies

- Domaine il s'agit du domaine de votre serveur, (par exemple www.cmon-site.net). Par défaut (si vous omettez ce paramètre) le domaine qui a créé le cookie (donc le votre) sera automatiquement inséré
- Sécurisé lorsqu'il est mis à 1 indique que le cookie ne sera transmis que si la ligne est sécurisée (par SSL ou S-HTTP)

Les Cookies

- **La fonction setcookie() doit être utilisée avant tout envoi de données HTML vers le navigateur**, même si ces données sont envoyées avec echo, print ou équivalent (le script qui la contient doit donc être placé avant la balise <HTML> et avant toute fonction echo(),print ou printf()).
  - **Erreur classique : Headers already sent by monFichier.php**
- Le cookie n'est pas visible avant le prochain chargement de page.
- **Il est possible d'utiliser des tableaux dans un cookie**. Autant de cookies que d'éléments du tableau seront alors envoyés, mais tout se fait de façon transparente, puisque à la lecture un tableau sera créé. Il est quand même préférable d'utiliser les fonctions implode et explode pour envoyer ainsi qu'un seul cookie.
- Il faut savoir que **certains navigateurs plus anciens ne traitraient pas correctement les cookies**

Architecture et développement Web

Les Sessions

Les Sessions

- HTTP est sans état (stateless)
- Les sessions offrent un mécanisme pour créer un état au dessus de HTTP
  - Cela permet de réaliser des opérations au sein de transactions,
  - Cela veut dire permettre de retenir les valeurs de variables et de reconnaitrre un utilisateur d'une requete HTTP à l'autre
- Enregistrement de Session:
  - Cookies
  - Réécriture d'URL

Les Sessions

- Les Sessions sont à considérer
  - comme un espace de travail qui est rattaché à un utilisateur donné
  - Dans lequel il est possible de stocker un ensemble de variables
  - Et qui reste persistant tant que l'utilisateur visite les pages de notre site sans fermer le navigateur
- Une session commence :
  - Avec une commande particulière (debut de session) dans la page php
- Et termine via :
  - une commande particulière (fin de session)
  - ou alors en cas d'inactivité, après une délai d'expiration

Les Sessions

- Chaque session possède un identifiant unique - qu'on appelle "session id"
- Chaque requête HTTP du client est accompagnée du "session id"
- Les "Session id" sont indiqués via cookie ou via URL
- A ces sessions correspondent des fichiers temporaires sur le serveur Web

Les Sessions

- Peut être utilisée quand les cookies ont été interdits sur le navigateur
- Le “Session id” est ajouté aux URLs dans la page retournée vers le navigateur
- Par ex:

```
<a href=“carnet.php">
```

devient

```
<a href="carnet.php?PHPSESSID=F35AD32242DN43E">
```

Les Sessions

- Les sessions sont physiquement présentes sur le serveur web sous forme de fichier texte.
- Ils sont associés au « session id » grâce à leur nom
- Ce mécanisme est beaucoup plus fiable que les cookies, c’est la raison pour laquelle il est utilisé en majorité

Les Sessions

- Les principales:
  - session_start () : crée une session (ou restaure la session trouvée sur le serveur, via l'identifiant de session passé)
  - session_destroy (): détruit la session courante et toutes les données associées avec cette session
  - session_register (): Enregistre une variable dans une session
  - session_unregister (): Supprime une variable de la session
  - session_is_registered ():Vérifie si une variable est enregistrée dans la session
  - session_id (): identifiant courant de session

Les Sessions : Exemples

```
<?php
session_start();
if
(!isset($_SESSION['compteur']))
{
    $_SESSION['compteur'] = 0;
} else {
    $_SESSION['compteur']++;
}
?>
```

Sessions et sécurité

- Utiliser les sessions ne signifie pas que les données de session ne pourront être vue que par un seul utilisateur.
- Les sessions reposent sur un identifiant de session, ce qui signifie que quelqu'un peut voler cet identifiant, rien qu'en volant l'ID.

Sessions et sécurité

- Comment éviter les attaques par fixation de session ?
- S'il n'existe aucune session active associée à l'identifiant de session présenté par l'utilisateur, alors regénérez-le, juste pour être certain.

```
<?php
session_start();

if (!isset($_SESSION['initiated']))
{
    session_regenerate_id();
    $_SESSION['initiated'] = true;
}

?>
```

- Regénérer le SID à chaque requète

Architecture et développement Web

La couche modèle / ORM

Présentation

- MySQL est une base de données implémentant le **langage de requête SQL**. Cette partie suppose connue les principes des bases de données relationnelles.
- Il existe un outil libre et gratuit développé par la communauté des programmeurs libres : phpMyAdmin qui permet l'administration aisée des bases de données MySQL avec php. Il est disponible sur :

http://sourceforge.net/projects/phpmyadmin/ et http://www.phpmyadmin.net.

- Avec MySQL vous pouvez créer plusieurs bases de données sur un serveur. Une base est composée de tables contenant des enregistrements.
- Plus d'informations sont disponibles à http://www.mysql.com/.
- La documentation de MySQL est disponibles à http://www.mysql.com/documentation/, ainsi qu'en français chez nexen : http://dev.nexen.net/docs/mysql/.

Connexion (I)

Pour se connecter à une base depuis php, il faut spécifier un nom de serveur, un nom d'utilisateur, un mot de passe et un nom de base.

Les fonctions de connexion :

`mysql_connect($server,$user,$password)` : permet de se connecter au serveur **$server** en tant qu'utilisateur **$user** avec le mot de passe **$password**, retourne l'identifiant de connexion si succès, FALSE sinon

`mysql_select_db($base[,$id])` : permet de choisir la base **$base**, retourne TRUE en cas de succès, sinon FALSE

`mysql_close([$id])` : permet de fermer la connexion

`mysql_pconnect` : idem que **mysql_connect** sauf que la connection est persistante, il n'y a donc pas besoin de réouvrir la connexion à chaque script qui travaille sur la même base.

Les identifiants de connexion ne sont pas nécessaires si on ne se connecte qu'à une seule base à la fois, ils permettent seulement de lever toute ambiguïté en cas de connexions multiples.

Connexion (II)

Exemple 1 :

```
if( $id = mysql_connect("localhost","root","124") ) {
if( $id_db = mysql_select_db("librairie") ) {
echo "Succès de connexion.";
            /* code du script … */
      } else {
       die("Echec de connexion à la base.");
      }
      mysql_close($id);
} else {
     die("Echec de connexion au serveur de base de
données.");
}
```

Interrogation

- Pour envoyer une requête à une base de donnée, il existe la fonction : **`mysql_query($str)`** qui prend pour paramètre une chaîne de caractères qui contient la requête écrite en SQL et retourne un identificateur de résultat ou FALSE si échec.
- Les requêtes les plus couramment utilisées sont : **CREATE** (création d'une table), **SELECT** (sélection), **INSERT** (insertion), **UPDATE** (mise à jour des données), **DELETE** (suppression), **ALTER** (modification d'une table), etc.
- *Exemple :*

```
$result = mysql_query("SELECT addresse FROM client  WHERE nom=\"$name\"");
```

- Cet exemple recherche l'adresse de l'utilisateur portant pour nom la valeur de la chaîne **$name**. L'identificateur de résultat **$result** permettra à d'autres fonctions d'extraire ligne par ligne les données retournées par le serveur.
- Attention, contrairement à Oracle SQL, les requêtes MySQL ne se terminent pas par un point virgule '**;**' et n'autorisent pas les **SELECT** imbriqués.

Extraction des données (I)

- Une fois la requête effectuée et l'identificateur de résultat acquis, il ne reste plus qu'à extraire les données retournées par le serveur.
- Sous Oracle, l'affichage des résultats d'une requête se fait ligne par ligne, sous MySQL, c'est pareil. Une boucle permettra de recueillir chacune des lignes à partir de l'identifiant de résultat.

```
SQL > SELECT * FROM client;
 ID   NAME     ADDRESS
----------------------------
  1   Boris    Moscou
  2   Bill     Washington
  3   William  London
```

- 1ère ligne
- 2ème ligne
- 3è ligne

- Une ligne contient (sauf cas particulier) plusieurs valeurs correspondants aux différents attributs retournés par la requête. Ainsi, une ligne de résultat pourra être sous la forme d'un tableau, d'un tableau associatif, ou d'un objet.

Extraction des données (II)

`mysql_fetch_row($result)` : retourne une ligne de résultat sous la forme d'un tableau. Les éléments du tableau étant les valeurs des attributs de la ligne. Retourne FALSE s'il n'y a plus aucune ligne.

Ici, on accède aux valeurs de la ligne par leur indice dans le tableau.

```
while($ligne = mysql_fetch_row($result)) {
            $id = $ligne[0];
            $name = $ligne[1];
            $address = $ligne[2];
            echo ''$id - $name, $address <br />'';
      }
```

Extraction des données (III)

mysql_fetch_array($result) : retourne un tableau associatif. Les clés étant les noms des attributs et leurs valeurs associées leurs valeurs respectives. Retourne FALSE s'il n'y a plus aucune ligne.

Ici, on accède aux valeurs de la ligne par l'attribut dans le tableau associatif.

```
$requet = "SELECT * FROM client";
if($result = mysql_query($requet)) {
        while($ligne = mysql_fetch_array($result)) {
   $id = $ligne["id"];
                $name = $ligne["name"];
   $address = $ligne["address"];
   echo "$id - $name, $address <br />";
        }
} else {
   echo "Erreur de requête de base de données.";
}
```

PDO

PDO

- PDO (PHP Data Object) est une extension PHP proposant une abstraction de bases de données.
- Elle permet aux développeurs de créer des applications portables sur plusieurs moteurs de BDD.

PDO

- Pourquoi PDO :
  - Simplicité d’utilisation
  - Portabilité
  - Performance / Rapidité
  - Extensibilité

PDO

- Quels moteurs supportés :
  - Microsoft SQL Server / Sybase
  - Firebird / Interbase
  - DB2 / INFORMIX (IBM)
  - MySQL
  - OCI (Oracle Call Interface)
  - ODBC
  - PostgreSQL
  - SQLite

PDO

- Le DNS :

  drivername:<driver-specific-stuff>

  – mysql:host=name;dbname=dbname
  – odbc:odbc_dsn
  – oci:dbname=dbname;charset=charset
  – sqlite:/path/to/db/file
  – sqlite::memory:

PDO

- La création d'une connexion (Exemple sur MYSQL)

```
try {
        $dbh = new PDO("mysql:host=$hostname;dbname=$dbname", $user, $pass);
        echo "Connectée";
}
catch (PDOException $e) {
        echo $e->getMessage();
}
```

PDO

- La fermeture d'un connexion

```
try {
        $dbh = new PDO("mysql:host=$hostname;dbname=$dbname",
$user, $pass);
        echo "Connectée";
         $dbh = null;
}
catch (PDOException $e) {
        echo $e->getMessage();
}
```

PDO

- Les connexions persistantes entre les requêtes

```
$dbh = new PDO($dsn, $user, $pass,
 array(
  PDO_ATTR_PERSISTENT => true
 )
);
```

PDO

- L’exécution d’une requête d’insertion

```
try {
        $dbh = new PDO("mysql:host=$hostname;dbname=$dbname",
$user, $pass);
        echo "Connectée";

        $count = $dbh->exec ("INSERT INTO table ( col1 , col2)
VALUES ('v1' , 'v2') " ) ;

        // Nombre de lignes modifées
        echo $count;
}
catch (PDOException $e) {
        echo $e->getMessage();
}
```

PDO

- Traitement d'une requête SELECT

```
try {
        $dbh = new PDO("mysql:host=$hostname;dbname=$dbname", $user, $pass);
        echo "Connectée";

        $sql = "SELECT * from table";

        foreach ($dbh->query ($sql) as $row) {
                print $row["col1"] . " - " . $row["col2"];
        }
}
catch (PDOException $e) {
        echo $e->getMessage();
}
```

PDO

- Préparation d'une requête SELECT

```
$stmt = $dbh>prepare('select * from table where col1 = :col1 and
col2 = :col2');

$stmt->bindParam(':col1', $col1,PDO::PARAM_INT);
$stmt->bindParam(':col2', $col2, PDO::PARAM_STR, 5);

$stmt->execute ( )

$result = $stmt->fetchAll ();

foreach ($result as $row) {
       echo $row['col1'] . ' - ' . $row['col2'];
}
```

PDO

- Transactions

```
try {

	$dbh->beginTransaction ();

	// Maj donnéees

	$dbh->commit ();
}

catch (PDOException $e) {
	$dbh->rollback ();
}
```

ORM

ORM

- Un mapping objet-relationnel est un modèle de programmation permettant de faire la relation entre une Base de Données Relationnelle et un langage Orienté Objet.
- Un ORM génére des classes permettant de manipuler le schéma d'une BDD.
- Il fournit des API dans le language utilisé pour manipuler les données en BDD.

Les avantages

- Simplification du code :
  - L'ORM masque la problématique de la gestion des connexions aux serveurs de données
  - Simplifie la manipulation des données
  - Simplifie le code : évite le SQL au milieu du code
- Evolutivité :
  - Fournit une indépendance du code par rapport à la base de données
- Améliore les performances :
  - Optimise les accès aux moteurs de données
  - Offre une gestion de cache

Les Principaux ORM PHP

- Doctrine
- AgileToolKit
- Syrius
- pdoMap
- Kernel56
- PHPSIMPLEDB
- Propel
- FuelPHP,
- PersistentObject

PROPEL

Propel est un ORM pour PHP 5.

Crée en 2005

Licence MIT : À partir de la version 1.5

Il est possible de l'utiliser avec les frameworks Symfony et Symfony2.

http://propelorm.org/

Moteurs supportés

- Propel supporte les moteurs de base de données suivants :
  - MySQL,
  - PostgreSQL,
  - SQLite,
  - MSSQL,
  - Oracle.

Installation

- PHP 5.4 minimum,
- Avec le DOM (libxml2) module enabled
- Installation PEAR
  - sudo apt-get install php-pear
- Package Pear :
  - pear channel-discover pear.phing.info
  - pear install phing/phing
  - pear install Log

Installation

- Avec l'aide de composer

```
{  "require":
     {
     "propel/propel": "~2.0@dev"    }
}
```

Création du schéma

- Créer le fichier schema.xml
- Ce fichier permet la création automatique de la base de donnée
- Il permettra également de produire le modèle objet PHP associé.

Création du schéma

- La balise <database> permet de définir la base de données utilisée.

```
<?xml version="1.0" encoding="UTF-8"?>

<database name="librairie"
  defaultIdMethod="native">

  <!-- table definitions go here -->

</database>
```

Création du schéma

- La balise <table> permet de définir la structure de donnée de la table et de l'objet PHP correspondant.

```
<?xml version="1.0" encoding="UTF-8"?>

<database name="librairie"
  defaultIdMethod="native">

   <table name="livre" phpName="livre">

    <!- Définition de la table -->

  </table>

</database>
```

Création du schéma

- La balise `<column>` permet de définir les colonnes de la table et les attributs de l'objet défini.
- L'attribut `type` permet de préciser le type de la donnée
- L'attribut `primaryKey` permet de définir les clés primaires de la table
- L'attribut `required` permet de définir les champs obligatoires
- L'attibut `autoIncrement` permet de préciser les champs auto incrémentés

```
<column name="id" type="integer" required="true"
primaryKey="true" autoIncrement="true"/>


<column name="prenom" type="varchar" size="128" required="true"/>
```

Création du schéma

- L'ajout d'une clé étrangère s'effectue avec la balise `<foreign-key>`
- La balise reference permet de définir la colonne référencée

```
<foreign-key foreignTable="editeur"
      <reference local="editeur_id" foreign="id"/>
</foreign-key>


<foreign-key foreignTable="auteur">
      <reference local="auteur_id" foreign="id"/>
</foreign-key>
```

Création du schéma

- L'élément `<index>` permet de créer un index sur une ou plusieurs colonne

```
<index name="monindex">
    <index-column
    name="ColumnName" />
</index>
```

Création du schéma

- L'élément `<unique>` permet de créer un index unique sur une ou plusieurs colonne

```
<unique name="nomIndex"]> <unique-
column
    name="ColumnName"/>
</unique>
```

Construire le Modèle Objet

- Le fichier `propel.yaml` permet de définir les éléments de configuration nécessaires à la création du modèle de données et de la base associée.

```
propel:
  database:
      connections:
          librairie:
              adapter: mysql
              classname: Propel\Runtime\Connection\ConnectionWrapper
              dsn: "mysql:host=localhost;dbname=librairie"
              user: root
              password:
              attributes:
  runtime:
      defaultConnection: librairie
      connections:
          - librairie
  generator:
      defaultConnection: librairie
      connections:
          - librairie
```

Construire le Modèle Objet

- Définir un lien vers la commande

```
$ ln -s $HOME/workspace/vendor/propel/propel/bin/propel propel
```

- Générer le modèle objets :

```
$ ./propel model:build
```

- Générer le schéma de la BDD :

```
$ ./propel sql:build
```

Générer le fichier de configuration :

```
$ ./propel config:convert
```

Construire le Modèle Objet

- Gérer les dépendances vers le modèle objets produit par propel en complétant la directive `autoload` dans le fichier `composer.jon`
- La commande `config:convert` génère le script PHP de configuration `generated-conf/config.php`

```
{ ...
     "autoload":
               {
          "classmap":
               ["generated-classes/"]
          }
          }
```

- Compléter votre contrôleur pour inclure les dépendances

```
<?php

// setup the autoloading
require_once '/path/to/vendor/autoload.php';

// setup Propel
require_once '/generated-conf/config.php';
```

MEMENTO

Les principales commandes de PROPEL :

config

`config:convert-xml`	Génère le script de configuration

database

`database:reverse`	Génère le fichier schema.sql à partir d'une BDD source

model

`model:build`	Génère le modèle objet

sql

`sql:build`	Génère le script SQL de création de la BDD
`sql:insert`	Génère les ordres INSERT

C.R.U.D

- **CRUD** (pour *Create, Read, Update, Delete*) (ou SCRUD avec un "S" pour *Search*) désigne les quatre opérations de base pour la persistance en base de données.
  - **C**reate : créer
  - **R**ead : lire
  - **U**pdate : mettre à jour
  - **D**elete : supprimer
- Ce terme est aussi un jeu de mot en anglais sur l'adjectif **crude** (en français **brut** ou **rudimentaire**).

Ajout de lignes

- Exemple d'insertion d'une ligne dans la BDD via les objets du modèle objet

```
<?php
$auteur = new Auteur();
$auteur->setPrenom('Lucien');
$auteur->setNom('Martin');
$auteur->save();


// INSERT INTO auteur (prenom, nom) VALUES ('Lucien', 'Martin');
```

Lecture

- Exemple d'accès aux données via le modèle objet

```
<?php
echo $auteur->getId();          // 1
echo $auteur->getPrenom(); // 'Lucien'
echo $auteur->getNom();  // 'Martin';
```

SELECT

- Exemple de requête via le modèle objet pour récupérer un enregistrement par sa PK

```
<?php
$q = new AuteurQuery();
$firstAuteur = $q->findPK(1);
?>


SELECT auteur.id, auteur.prenom, auteur.nomFROM `auteur`
WHERE auteur.id = 1
LIMIT 1;
```

SELECT

- Exemple de factory permettant la sélection multiple

```
<?php

$selectedAuteurs = AuteurQuery::create()->findPKs(array(1,2,3,4,5,6,7));
```

- Requete par Colonne via la factory générée par Propel

```
<?php

    $auteurs = AuteurQuery::create()->find();

    foreach($auteurs as $auteur)  { echo $auteur->getPrenom(); }
```

- Exemple de requete en ligne

```
<?php

    $auteurs = AuteurQuery::create()

        ->orderByNom()  ->limit(10)  ->find();
```

- Exemple de filtre par la méthode filterBy***XXX***()

```
    $authors = AuteurQuery::create() ->filterByPrenom('Emma') ->find()
```

SELECT

- En utilisant du SQL

```
<?php
use Propel\Runtime\Propel;
$con=
Propel::getWriteConnection(\Map\BookTableMap::DATABASE_NAME);
$sql= "SELECT * FROM livre "
      ." WHERE auteur = :name)";
$stmt = $con->prepare($sql);
$stmt->execute(array(':name' => 'Emma'));
```

UPDATE

- Exemple de mise à jour d'un enregistrement via l'ORM

```
<?php

$author = AuteurQuery::create()-
  >findOneByPrenom('Lucien');

$author->setLastName('Martin');

$author->save();
```

DELETE

- Exemple de suppression d'un enregistrement via l'ORM

```
<?php
$auteur = AuteurQuery::create()-
  >findOneByPrenom('Lucien');
$auteur->delete();
```

Relations en cascade

- Exemple d'enregistrement mis en relations

```php
<?php

$auteur = new Author();
$auteur->setFirstName("Leo");
$auteur->setLastName("Tolstoy");

$editeur = new Publisher();
$editeur->setName("Viking Press");
```

```php
$livre = new Book();
$livre->setTitre("War & Peace");
$livre->setIsbn("0140444173");
$livre->setEditeur($editeur);
$livre->setAuteur($auteur);
$livre->save();
// Enregistre les 3 objets!
```

TRANSACTIONS

- Les propriétés ACID (atomicité, cohérence, isolation et durabilité) garantissent qu'une transaction informatique est exécutée de façon fiable.
  - Atomicité : Une transaction s'effectue au complet ou pas du tout ;
  - Cohérence : Une transaction passera le système d'un état valide à un autre état valide.
  - Isolation : Une transaction doit s'exécuter sans aucune dépendance avec d'autres transactions.
  - Durabilité : Lorsqu'une transaction a été confirmée, elle demeure enregistrée

```
use Propel\Runtime\Propel;
$con = Propel::getWriteConnection("librairie");
$con->beginTransaction();
try { …
            $con->commit(); }
catch (Exception $e) { …
            $con->rollback();}
```

Exemple (1/2)

- Accès à la couche modèle pour récupérer le catalogue :

```
<?php
$livres = LivreQuery::create()
    ->offset(0)
    ->limit(10)
    ->find()
    ->toArray();

echo $twig->render('catalogue.html',
array('livres' => $livres));
```

Exemple (2/2)

- Affichage de la vue catalogue

```
{% extends "masterPage.html" %}
{% block content %}
<table border="1">
{% for livre in livres %}
<tr>
        {% for key in livre %}
        <td>{{key}} {{livre[key]}}</td>
        {% endfor %}
        <td><a href="#" onclick ="ajouterPanier({{livre["Id"]}})">panier</a></td>
</tr>
{% endfor %}
</table>
{% endblock %}
```

DOCTRINE - Présentation

- L’ORM le plus connu sur la plateforme PHP
- Intégré à Symfony, Zend Framewotk
- Simple
- Puissant

DOCTRINE - Installation

```
{    "require": {
       "doctrine/orm" : "2.*",
       "symfony/yaml":"*"    },
 "autoload":
 {"classmap":["src/"]}
 }
```

```
Sudo ./composer.sh update
```

DOCTRINE - Configuration

- Un fichier bootstrap permettant de définir les paramètres de connexion à la BDD

```php
<?php
use Doctrine\ORM\Tools\Setup;
use Doctrine\ORM\EntityManager;
date_default_timezone_set('America/Lima');
require_once "vendor/autoload.php";

$isDevMode = true;
$config = Setup::createYAMLMetadataConfiguration(array(__DIR__ . "/yaml"), $isDevMode);

$conn = array(
  'driver' => 'pdo_mysql',
  'user' => 'root',
  'password' => '',
  'dbname' => 'librairie',
  'port' => '3306'
);

$entityManager = EntityManager::create($conn, $config);
```

Doctrine - Configuration

Le fichier `cli-config` permet de déclarer l'`EntityManager` à la console.

```
<?php
use Doctrine\ORM\Tools\Console\ConsoleRunner;

require_once 'bootstrap.php';

return ConsoleRunner::createHelperSet($entityManager);
```

DOCTRINE – La Console

- Doctrine met à disposition une console d'administration

Génération du fichier de mapping

```
php vendor/bin/doctrine orm:convert-mapping --namespace="" --force --
from-database yml ./config/yaml
```

Génération des classes entités

```
php vendor/bin/doctrine orm:generate-entities --generate-
annotations=false --update-entities=true --generate-methods=false ./src
```

Mise à jour du schéma

```
php vendor/bin/doctrine orm:schema-tool:update   --force
```

Validation du Schéma

```
php vendor/bin/doctrine orm:validate-schema
```

Vidage du cache

```
php vendor/bin/doctrine orm:clear-cache:metadata
```

DOCTRINE – Le manager

- L’EntityManager est le point d'accès central aux fonctionnalités ORM.
- C'est une façade pour toutes fonctions de l’ORM,
- L'instanciation se fait par la méthode static create ().

```
$clientRepository = $entityManager->getRepository('Client');
$clients = $clientRepository->findAll();
```

DOCTRINE : CRUD – La recherche

- La recherche s'effectue via la méthode findBy

```
$clientRepository = $entityManager->getRepository('Client');
$clients = $clientRepository->findBy(array('age' => 20, 'nom' =>
'Miller'));
$client = $clientRepository->findOneBy(array('prenom' =>
'Thomas'));
$client = $clientRepository->findOneByPrenom('Thomas');
$nb = $clientRepository- ->count(['prenom' => 'Thomas']);
```

- Ou une requete DQL

```
$q = $entityManager->createQuery("select u from
MyDomain\Model\Client c where c.age >= 20 and c.age <= 30");
$clients = $q->getResult();
```

DOCTRINE : CRUD – La Création

- L'instanciation permet la création d'une nouvelle occurrence qui sera enregistrée en BDD

```
<?php
$client = new Client;
$client->setNom('Dupond');
$entityManager->persist($client);
$entityManager->flush();
```

DOCTRINE : CRUD – La Mise à jour

```php
<?php
$cr = $entityManager->getRepository('Client');
$client = $cr->findOneBy(array('prenom'=>'Thomas'));
$client->setNom('Dupond');
$entityManager->persist($client);
$entityManager->flush();
```

DOCTRINE : CRUD – La Suppression

```
$entityManager->remove ($client);
$entityManager->flush ();
```

Doctrine le transactionnel

```php
<?php
$em->getConnection()->beginTransaction();
try {
    $client = new Client;
    $client->setNom('George');
    $em->persist($client);
    $em->flush();
    $em->getConnection()->commit();
} catch (Exception $e) {
    $em->getConnection()->rollBack();
    throw $e;
}
```

Doctrine : Relation one to one

```
$clientRepo = $entityManager->getRepository('Client');
$client = $clientRepo->find(1);
$adresse = new Adresse();
$adresse->setRue("Champ de Mars, 5 Avenue Anatole");
$adresse->setVille("Paris");
$adresse->setPays("France");

$client->setAdresse($adresse);

$entityManager->persist($client);
$entityManager->flush();
```

Références

- Sécurité des SESSIONS
  - http://php.net/manual/fr/session.security.php
- API Propel
  - http://propelorm.org/
- Doctrine
- https://openclassrooms.com/fr/courses/368049-utilisation-dun-orm-les-bases-de-doctrine