

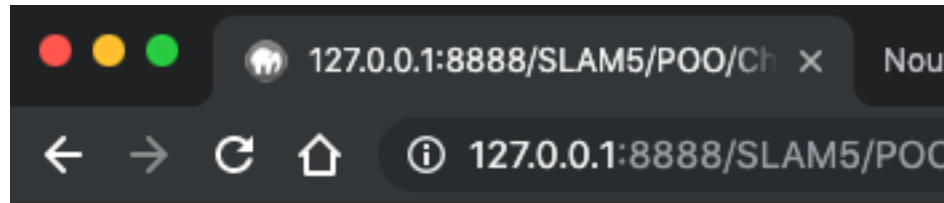
POO & BDD

BTS SIO :: SLAM

Une base

1 •	SELECT * FROM BD_AIRBASE.PILOT;				
100%	1:1				
Result Grid		Filter Rows:	Search	Edit:	Export/Import:
	NumP	NameP	Address	Salary	
▶ 1		Adémar Monoto	3 rue Alex Pyration 31000 Toulouse	3200.00	
2		Claire Hyère	12 boulevard Gaspard Alizan 1100...	2812.00	
3		Matéo Porte	12 avenue Yvon Lavallée 34000 M...	4200.00	
4		Tex Agère	34 rue Milène Micoton 75000 Paris	2770.70	

Exemple de la table PILOT



Rappel

Le pilote Adémar Monoto à pour salaire 3200.00
Le pilote Claire Hyère à pour salaire 2812.00
Le pilote Matéo Porte à pour salaire 4200.00
Le pilote Tex Agère à pour salaire 2770.70
Le pilote Larry Bambelle à pour salaire 3700.00
Le pilote Jerry Kan à pour salaire 2756.87
Le pilote Tangy à pour salaire 3244.50

- Si je souhaite afficher ce contenu en utilisant PHP je pourrais avoir ce code :

```
<?php
// Création de l'objet PDO
$db = new PDO('mysql:host=127.0.0.1:8889;dbname=BD_AIRBASE','root','root');
// Requête
$requete = $db->query('SELECT * FROM PILOT;');
while ($pilote = $requete->fetch(PDO::FETCH_ASSOC))
{
    echo "Le pilote ".$pilote['NameP'], " à pour salaire ", $pilote['Salary'], "<br>";
}
?>
```

Exercices

- Exercice 1.1 :
Construire la base avec le dump fournit sur le site
(structure et données)
Tester que le code précédent fonctionne (a vous
d'adapter les paramètres)

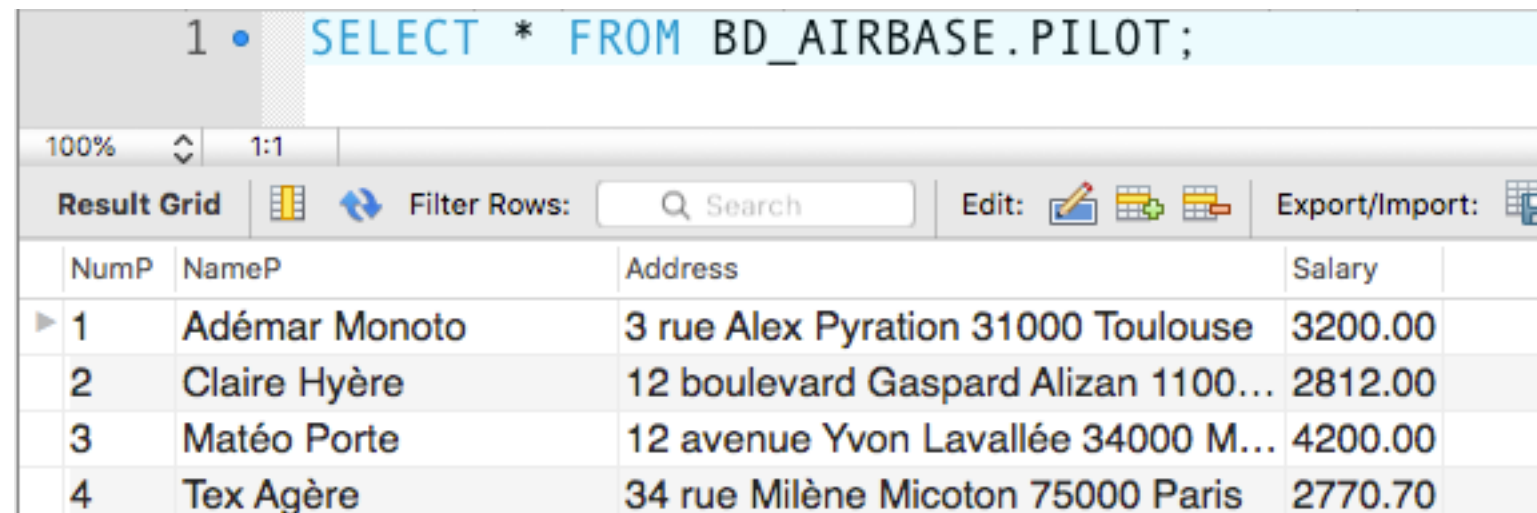
POO et BD

- Une entité = une classe
- On a donc besoin d'une classe par entité
- Quelles sont les propriétés ?
- Quelles sont les méthodes ?

POO et BD

- Les propriétés seront tout simplement les propriétés de l'entité.
- Mais il ne faudra pas les modifier directement mais passer par des setters et des getters. (déjà vu !)
- Le setter permettra de vérifier l'intégrité de la valeur assignée.

POO et BD



The screenshot shows a database query tool interface. At the top, a SQL query is entered: `SELECT * FROM BD_AIRBASE.PILOT;`. Below the query, the results are displayed in a table with the following columns: NumP, NameP, Address, and Salary. The results table contains four rows of data.

	NumP	NameP	Address	Salary
▶ 1	1	Adémar Monoto	3 rue Alex Pyration 31000 Toulouse	3200.00
2	2	Claire Hyère	12 boulevard Gaspard Alizan 1100...	2812.00
3	3	Matéo Porte	12 avenue Yvon Lavallée 34000 M...	4200.00
4	4	Tex Agère	34 rue Milène Micoton 75000 Paris	2770.70

Dans un premier temps on peut déjà écrire les attributs de la classe :

```
class Pilote
{
    private $_NumP;
    private $_NameP;
    private $_Address;
    private $_Salary;
```

POO et BD

Avec les getters

```
// Les getters

public function getNumP()
{
    return $this->_NumP;
}
public function getNameP()
{
    return $this->_NameP;
}
public function getAddress()
{
    return $this->_Address;
}
public function getSalary()
{
    return $this->_Salary;
}
```

Exercices

- *Exercice 1.2* : Ecrire la classe Pilote. Avec un constructeur. Attention le constructeur passera par les setters pour initialiser les valeurs des attributs.
- Ecrire les mutateurs de cette classe.
Le nom doit être en caractères, l'adresse également et le salaire compris entre 2000 et 30000 euros.
- Tester la création d'un pilote.

L'hydratation

- L'hydratation c'est le fait d'hydrater d'objet c'est à dire de lui apporter ce qu'il faut pour qu'il fonctionne ici les données pour les propriétés.
- On va donc assigner des valeurs aux attributs de l'entité. C'est ce que vous avez fait avec le constructeur mais dans le cas de classe lié à des tables on passe par cette fonction.
- Cette fonction s'appelle hydrate()
- Cette fonction prend en paramètre un tableau qui contient toutes les données.
- hydrate() passe également par les setters

L'hydratation

Comme il peut avoir beaucoup de données on va faire une boucle. On sait que le setter commence par set et après le nom de l'attribut.

```
public function hydrate(array $donnees)
{
    foreach ($donnees as $key => $value) {
        $method = 'set' . $key;
        if (method_exists($this, $method))
        {
            $this->$method($value);
        }
    }
}
```

Remarques:

- L'hydratation utilise les mutateurs.
- Il est courant d'implémenter un constructeur. Celui-ci prend un tableau en paramètre et appelle hydrate.

Exercices

- Exercice 1.3 : Modifier la classe pilote pour passer maintenant par la fonction hydrate.
- On gardera un constructeur qui prendra en paramètre un tableau pour appeler hydrate

La gestion de la BDD

- Reste les méthodes d'ajout, de modification, de suppression.
- C'est le CRUD (Create, Read, Update, Delete)
- On pourrait ajouter ces méthodes à notre classe ça serait même très tentant !!!
- Mais en POO une classe = un rôle
- La classe Pilote sert à représenter les Pilotes pas à gérer les occurrences dans la BDD ni le CRUD.

La gestion de la BDD

- On va donc créer une autre classe qui elle va gérer (manager) cela. A savoir l'ajout d'un pilote, la suppression, etc. dans la base de données.
- Même question à chaque fois : quels caractéristiques ? quelles fonctionnalités ?
- Ici une seule propriété l'objet PDO pour accéder à la base de données.
- Les fonctionnalités celles du CRUD

La gestion de la BDD

```
class PiloteManager
{
    private $_db;

    public function __construct($db)
    {
        $this->setDB($db);
    }

    // CRUD

    public function add(Pilote $pilote)
    {
        $q = $this->_db->prepare('INSERT INTO PILOT(NumP, NameP, Address, Salary) VALUES(:NumP, :NameP, :Address, :Salary)');
        $q->bindValue(':NumP', $pilote->getNumP());
        $q->bindValue(':NameP', $pilote->getNameP());
        $q->bindValue(':Address', $pilote->getAddress());
        $q->bindValue(':Salary', $pilote->getSalary());
        $q->execute();
    }

    public function delete(Pilote $pilote)
    {
    }

    public function update(Pilote $pilote)
    {
    }

    public function get($id)
    {
        // Retourne un objet Pilote en fonction de l'id
    }

    // Autres méthodes

    public function getList()
    {
        // Retourne tous les pilotes
    }

    public function setDb(PDO $db)
    {
        $this->_db = $db;
    }
}
```

Création d'un pilote

```
$piloteSuper = new Pilote([
    'NumP' => 1298,
    'NameP' => 'Tanguddy',
    'Address' => 'Rue des hirondelles 11000 Carcassonne',
    'Salary' => 3244.50]);

echo $piloteSuper;

try {
    $db = new PDO('mysql:host=127.0.0.1:8889;dbname=BD_AIRBASE','root','root');
}
catch (PDOException $e) {
    echo "Erreur : ".$e->getMessage();
    die();
}

$manager = new PiloteManager($db);
$manager->add($piloteSuper);
```

Exercices

- *Exercice 1.4* : Recopier le code PiloteManager et tester le code. En particulier l'ajout d'un pilote par le PiloteManager.
- Ecrire les méthodes manquantes à ce PiloteManager et les tester une à une.
 - add -> déjà faite
 - delete(Pilote \$pilote)
 - update(Pilote \$pilote)
 - get(\$NumP) :: retourne les informations sur le pilote dont le numéro est NumP
 - getList() :: retourne la liste des pilotes
- Vous avez fini ?
Bravo vous savez programmer les BDD en POO...