PHP Data Objects

— PDO —

Plan

Créer un objet PDO (PHP Data Objects)

http://php.net/manual/en/book.pdo.php

```php
$DB_HOST = 'localhost';
$DB_NAME = 'test';
$DB_USER = 'root';
$DB_PWD = '';

$db_connection = null;
try{
   $db_connection = new PDO("mysql:host=$DB_HOST;dbname=$DB_NAME",
                            $DB_USER,
                            $DB_PWD,
                            array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"));
}
catch(Exception $e){
   die('Error while connecting to MySQL.\n');
}
```

Plan

PDO : Executer une requête

La fonction query — http://php.net/manual/en/pdo.query.php

```
public PDOStatement PDO::query ( string $statement )
```

PDOStatement — http://php.net/manual/en/class.pdostatement.php

Une requête préparée qui conserve son résultat si elle a été exécutée

```php
$sql = 'SELECT name, color, calories FROM fruit ORDER BY name';
foreach ($db_connection->query($sql) as $row) {
  print $row['name'] . "\t";
  print $row['color'] . "\t";
  print $row['calories'] . "\n";
}
```

Plan

http://php.net/manual/fr/pdostatement.fetch.php

Utilisation d'objets "anonymes" (PDO : :FETCH_OBJ)

Objet possédant des propriétés de même nom que les champs du résultat de la requête

```php
$sql = 'select email, name, surname from User';
$request = $db_connection->query($sql)
$data = $request->fetch(PDO::FETCH_OBJ);

echo '<h1>Users</h1><ul>';
while(!empty($data)) {
   echo '<li>'.$data->email.' : '.$data->name.', '.$data->surname.'</li>';
   $data = $request->fetch(PDO::FETCH_OBJ);
}
```

Plan

PDO : Résultats d'une requête comme instance d'une classe donnée

PDO::FETCH_CLASS http://php.net/manual/fr/pdostatement.fetch.php

retourne une nouvelle instance de la classe demandée, liant les colonnes du jeu de résultats aux noms des propriétés de la classe. Le constructeur de la classe ne sera appelé qu'après que les champs aient été affectés

```php
class User
{
    private $name;
    private $surname;
    private $email;

    public function toHtml()
    {
        echo "<ul><li>Name : " . $this->name . "</li>";
        echo "<li>Surname : ". $this->surname ."</li>";
        echo "<li>Email : " . $this->email . "</li></ul>";
    }
}
...

$sql = 'select email, name, surname from User';
$request = $db_connection->query($sql)

echo '<h1>Users</h1>';
$allUsers = $request->fetchAll(PDO::FETCH_CLASS,'User');
$allUsers[0]->toHtml();
```

Exemples

```php
class User {

   protected static function db(){
      $pdo = <code d'initialisation de pdo>;
      return $pdo;
   }

   // Exécute la requête $sql et retourne des objets modèles
   protected static function query($sql){
      $st = static::db()->query($sql) or die("sql query error ! request : " . $sql);
      $st->setFetchMode(PDO::FETCH_CLASS, get_called_class());
      return $st;
   }
   ...
}
```

PDO::FETCH_CLASS et get_called_class()

Chaque tuple résultat de la requête sera automatiquement une instance de la classe courante (User dans cet exemple). Exemple :

```php
$users = User::query("select * from User");
```

PDO::FETCH_PROPS_LATE

Permet que le constructeur de la classe soit exécuté avant que PDO n'affecte les valeurs du tuple résultat dans les variables d'instances de l'objet.

Méthode magiques PHP

https ://www.php.net/manual/en/language.oop5.magic.php `__get` et `__set` permettent de lire et écrire des propriétés inexistantes sur un objet

```php
class User {
    protected $props;

    public function __construct($props = array()) { $this->props = $props; }

    public function __get($prop) { return $this->props[$prop]; }
    public function __set($prop, $val) { $this->props[$prop] = $val; }

    protected static function query($sql){
        $st = static::db()->query($sql) or die("sql query error ! request : " . $sql);
        $st->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, get_called_class());
        return $st;
    }
}

$users = User::query("select * from User");
// all user object in $users store database infos into its own instance variable props (PDO::FETCH_PROPS_LATE)

$aUser = $users[0];
// accessing infos is easy thanks to magic methods
$aUser->name // name does not exist in $aUser so magic method executed $aUser->__get('name') which returns $this->props['name']
```

Plan

PDO : prepare() avec paramètres

http://php.net/manual/fr/pdo.prepare.php

```php
$stmt = $db_connection->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);

// insert one row
$name = 'one';
$value = 1;
$stmt->execute();

// insert another row with different values
$name = 'two';
$value = 2;
$stmt->execute();
```

Factoriser vos requêtes dans User.sql.php

```php
<?php
// SQL queries for User
// USER table structure (USER_ID, USER_LOGIN, USER_EMAIL, USER_PWD, USER_FIRSTNAME, USER_LASTNA

User::addSqlQuery('USER_LIST',
  'SELECT * FROM USER ORDER BY USER_LOGIN');

  User::addSqlQuery('USER_GET_WITH_LOGIN_AND_PASSWORD',
  'SELECT * FROM USER WHERE USER_LOGIN=:login');

  User::addSqlQuery('USER_CREATE',
  'INSERT INTO USER (USER_ID, USER_LOGIN, USER_EMAIL, USER_PWD, USER_FIRSTNAME, USER_LASTNAME)
(NULL, :login, :email, :pwd, :firstname, :lastname)');

User::addSqlQuery('USER_CONNECT',
  'SELECT * FROM USER WHERE USER_LOGIN=:login and USER_PWD=:password');
?>
```

Extraction des requêtes de la classe User

User.class.php

```php
class User {
   protected static $table_name = 'USER';

   // *** Queries ****
   protected static $requests = array();
   public static function addSqlQuery($key, $sql){ static::$requests[$key] = $sql; }
   public static function sqlQueryNamed($key){ return static::$requests[$key]; }

   protected static function exec($sqlKey,$values=array()){
    $sth = static::db()->prepare(static::sqlQueryNamed($sqlKey));
    $sth->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, get_called_class());
    $sth->execute($values);
    return $sth;
  }

   public static function getList() { return self::exec('USER_LIST'); }

   public static function create($login, $pwd, $mail, $prenom, $nom) {
      $sth = self::exec('USER_CREATE',
         array( ':login' => $login,
            ':pwd' => $pwd,
            ':email' => $mail,
            ':firstname' => $prenom,
            ':lastname' => $nom));
      return static::tryLogin($login, $pwd);
   }
   // ...
}
```


PHP
iss