Zend
The *php* Company

Programmation orientée objet et usage avancé de PHP5 & MySQL5

Zend Technologies

France Télécom / Orange

Zend
The *php* Company

Introduction à PHP Avancé

Introduction

- Objectifs de la formation
  - La maîtrise de la programmation PHP 5
  - L'ensemble des nouveautés
  - Les bases de la POO
  - La maîtrise optimale du SGBD MySQL
- Comment profiter de la formation
  - Participer
  - Faire les exercices, mêmes triviaux
  - Innover ! Les TPs vous permettent de comprendre les concepts, faites-les à votre manière

Plan de formation (1/2)

- Rappels de MySQL
- La PHPDOC
- Indexation et MySQL
- Introduction à la POO
- Le connecteur PDO
- POO Avancée
- Gestion des exceptions
- Gestion des transactions SQL
- Flux XML
- Optimisation des bases de données

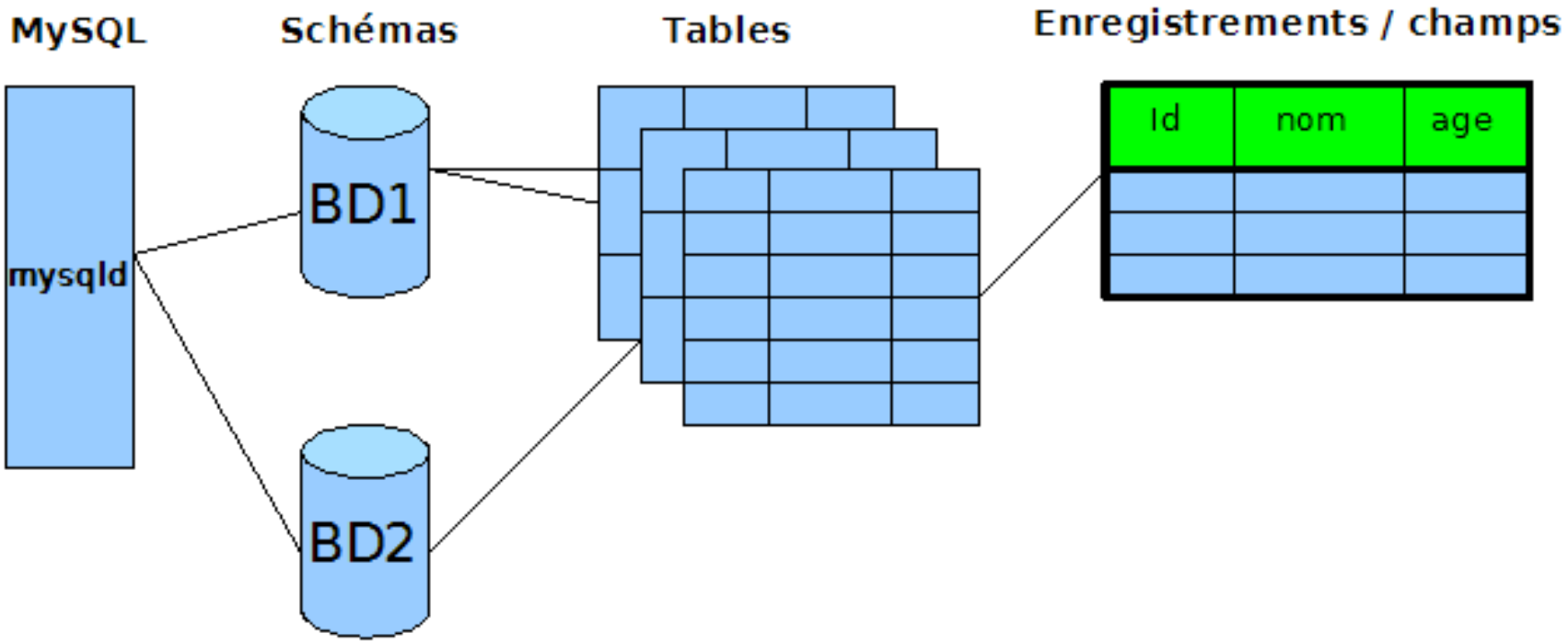
Plan de formation (2/2)

- Services Web
- Introduction à la SPL

Zend
The *php* Company

Rappel MySQL

Structure de la base de données



Structure de la base de données

Zend The php Company

- Un champ est une donnée, définie par un type, une longueur et des contraintes.
- Un enregistrement est une ensemble de champs.
- Une table contient des enregistrements et des champs.
- Un schéma (nom ANSI d'une base de données) est un ensemble de tables.
- Le serveur peut contenir plusieurs schémas.

L'extension mysqli

- Permet de profiter des fonctionnalités de MySQL 4.1 et +
- A partir de PHP 4.1.3
- PHP doit être compilé avec le support de l'extension mysqli (linux)
- L'extension mysqli doit être activée dans php.ini (windows)

Se connecter

- La fonction mysqli_connect(), possède 4 arguments principaux :
  - l'adresse du serveur
  - le nom d'utilisateur
  - le mot de passe pour l'authentification
  - la base de données à utiliser

Se connecter

- Connexion au serveur MySQL

```
<?php

$link = mysqli_connect('sql.zend.fr', 'monlogin', 'secret',
'mabase');
```

Créer un fichier de configuration

Zend The php Company

- L'accès aux bases de données se fait en plusieurs points de l'application
- Factorisez les informations de connexion dans un fichier de configuration

Créer un fichier de configuration

- Fichier de configuration

```
<?php

$mysql_host = 'sql.zend.fr';
$mysql_login = 'login';
$mysql_pass = 'secret';
$mysql_db = 'mabase';

$link = mysqli_connect($mysql_host, $mysql_login, $mysql_pass,
$mysql_db);
```

Envoyer une requête au serveur

- La fonction mysqli_query() permet d'envoyer une requête au serveur MySQL
- Elle prend 2 paramètres :
  - un identifiant de connexion vers le serveur
  - une requête SQL

Envoyer une requête au serveur

Zend The php Company

- Envoi d'une requête

```php
<?php

include_once 'configuration.php';

$sql = "SELECT nom, prenom FROM client WHERE ville = 'Paris'";

$resultat = mysqli_query( $link, $sql );
```

Récupérer le résultat

- 3 fonctions pour récupérer le résultat d'une requête :
  - mysqli_fetch_assoc() : Récupère le résultat sous forme de tableau associatif
  - mysqli_fetch_row() : Récupère le résultat sous forme de tableau indexé
  - mysqli_fetch_object() : Récupère le résultat sous forme d'objet

Récupérer le résultat

Zend The php Company

- Exploiter le résultat d'une requête SELECT

```php
<?php

include_once 'configuration.php';

$sql = "SELECT nom, prenom FROM client WHERE ville = 'Paris'";

$resultat = mysqli_query($link, $sql);

$enregistrement = mysqli_fetch_assoc($resultat);

while ($enregistrement) {
    // Affiche le champ - prenom -
    echo $enregistrement['prenom'], ' ';
    // Affiche le champ - nom -
    echo $enregistrement['nom'], '<br>';
}
```

Fermer une connexion

- La fonction mysqli_close() permet de fermer la connexion
- Elle prend en argument l'identifiant de connexion

Fermer une connexion

- mysqli_close() : ferme la connexion

```php
<?php

include_once('configuration.php');

$sql = "SELECT nom, prenom FROM client WHERE ville = 'Paris'";
$resultat = mysqli_query($link, $sql);
$enregistrement = mysqli_fetch_assoc($resultat);

while ($enregistrement) {
    // Affiche le champ - prenom -
    echo $enregistrement['prenom'], ' ';
    // Affiche le champ - nom -
    echo $enregistrement['nom'], '<br>';
}
//ferme la connexion
mysqli_close($link)
```

Rappels MySQL (1/3)

- A l'aide de phpMyAdmin, créez une base 'magasin'
- Dans cette base, créez une table 'produits' avec 5 champs :
  - 'id' de type 'INT', clé primaire, auto-incrémenté
  - 'nom' de type VARCHAR(250)
  - 'description' de type TEXT
  - 'prix' de type DECIMAL
  - 'image' de type BLOB

Vous pouvez aussi utiliser d'autres outils (Mysql Query Browser, SQL Front, ...).

Rappels MySQL (2/3)

- Créez un fichier 01_configuration.php
  - Définissez les variables $mysql_host = 'localhost', $mysql_db = 'magasin', $mysql_login = 'root', $mysql_pass = ''.
  - Il s'agit des paramètres de connexion à la base de données MySQL
  - Mettez des commentaires sur ces variables

C'est ce fichier de configuration que nous allons utiliser pour nous connecter à la base de données.

Rappels MySQL (3/3)

- Créez un fichier 02_rappels.php
- Dans ce fichier
  - Connectez-vous à la base avec mysqli
  - Exécutez une requête qui rapatrie les produits dans l'ordre croissant des prix
  - Affichez ces produits avec une boucle while
  - Fermez la connexion à la base de données

Ces rappels sont également valables avec l'extension mysql (PHP4) en mode procédural.

Zend
The *php* Company

La PHPDOC

Introduction

- La phpdoc est un standard d'écriture pour la documentation d'un code source en PHP.
- On reconnait la phpdoc à sa structure.
- La phpdoc permet de documenter en particulier :
  - Les fichiers php
  - Le classes
  - Les fonctions et les méthodes de classe
  - Les constantes et les propriétés de classe

Importance de la documentation

- La documentation permet de rendre compréhensible un code source.
- La phpdoc est un standard !
  - Qui permet de travailler à plusieurs.
  - Qui permet d'échanger le code source.
  - Qui rend possible une génération universelle de documentation en ligne ou papier.

Structure de la PHPDOC

- La PHPDOC est reconnaissable à sa structure

```
/**
 * Je suis un commentaire PHPDOC !
 */
```

Les balises de la PHPDOC

- La PHPDOC possède des balises d'identification

```
/**
 * Je suis le titre de mon commentaire.
 *
 * Je suis le contenu de mon commentaire.
 *
 * @author Steven Van Poeck <steven@zend.com>
 * @copyright Zend 2006
 * @package taskmanager
 * @version 0.1
 */
```

Les tags utiles (1)

- @access
  - Désigne la visibilité d'une propriété ou d'une méthode (public, protected, private static).
- @author
  - Auteur de l'élément (page, classe, méthode, fonction, propriété)
- @copyright
  - Copyright du code source ou de la classe.

Les tags utiles (2)

- @deprecated
  - La présence de ce tag informe le développeur que cette fonctionnalité est dépréciée et sera retirée à terme.
- @example
  - Lien vers un exemple. Il est aussi possible de mettre en place des exemples "inline".
- @ignore
  - Demande au parseur phpdoc d'ignorer un élément (pour un doublon par exemple)

Les tags utiles (3)

- @internal
  - Introduit une documentation "privée" qui peut être ou ne pas être générée à la demande.
- @link
  - Introduit un lien en rapport avec l'élément (exemple ou documentation supplémentaire). Cette balise n'accepte qu'un lien http.
- @see
  - Fait référence à une ressource (classe, fonction, etc. ou lien).

Les tags utiles (4)

- @since
  - Informe depuis quelle version de l'application une fonction existe.
- @tutorial
  - Lien vers la documentation étendue de l'élément. Pour d'autres liens, utiliser plutôt @link ou @see.
- @version
  - Version de l'élément.

Complétion dans l'éditeur

- Les tags @var et @return permettent d'expliciter le type de donnée qui sera contenu dans une propriété ou renvoyé par une fonction ou une méthode.
- Dans l'exemple du slide suivant, l'éditeur sait que la propriété $statement contient une valeur de type PDOStatement.

La PHPDOC
Complétion dans Zend Studio

```
1  <?php
2
3  class DB
4  {
5      /**
6       * @var PDOStatement
7       */
8      public $statement;
9
10 }
11
12 $db = new DB();
13 $db->statement->
```

```
PDOStatement::__sleep()                                                                                    int
PDOStatement::__wakeup()                                                                                   int
PDOStatement::bindColumn(mixed $column, mixed &$param, int[optional] $type = null, int[optional] $maxlen = null, mixed[option...  bool
PDOStatement::bindParam(mixed $paramno, mixed &$param, int[optional] $type = null, int[optional] $maxlen = null, mixed[option...  bool
PDOStatement::bindValue(mixed $paramno, mixed $param, int[optional] $type = null)                          bool
PDOStatement::closeCursor()                                                                                bool
PDOStatement::columnCount()                                                                                int
PDOStatement::debugDumpParams()                                                                            void
```

class PDOStatement

function **bindParam**(mixed $paramno,
mixed $param,
int[optional] $type,
int[optional] $maxlen,
mixed[optional] $driverdata)

bind a parameter to a PHP variable.

Parameters:

paramno mixed
param mixed
type int[optional]
maxlen int[optional]
driverdata mixed[optional]

Returns:

bool

Documentation d'un fichier

- Dans le dernier fichier PHP que vous avez créé
  - Ajoutez un commentaire PHPDOC en haut du fichier
  - Ajoutez un titre et une description
  - Repérez les tags utiles à la description du fichier et ajoutez-les
- A l'aide de l'assistant PHPDocumentor de Zend Studio, générez une documentation sur ce fichier

IMPORTANT : Documentez votre code au fur et à mesure. A la fin de la formation, nous générerons la documentation de l'ensemble de vos exercices.

Zend
The *php* Company

Gestion des Index

Les index

- Les index accélèrent les lectures (SELECT)
- Mais peuvent ralentir les écritures (INSERT, UPDATE, DELETE)
- 3 syntaxes pour la création
  - CREATE TABLE...
  - CREATE [UNIQUE | FULLTEXT] INDEX nom_index ON nom_table (colonne_à_indexer,...)
  - ALTER TABLE nom_table ADD {INDEX | KEY | UNIQUE | PRIMARY KEY} [nom_index] (colonne_à_indexer,...)

Les index

- Créer un index

```
/* Ajout d'un index unique sur le champ ine */
CREATE unique INDEX idx_ine ON etudiant (ine);

/* Ajout d'un index fulltext sur le champ cv */
CREATE fulltext INDEX idx_ft_cv ON etudiant (cv);

/* Ajout d'une clé primaire sur le champ id_etudiant */
ALTER TABLE etudiant ADD primary key(id_etudiant);

/* Ajout d'un index sur le couple de champs nom,prenom */
ALTER TABLE idx_nom_prenom ADD key (nom, prenom);
```

Les types d'index

- PRIMARY KEY
  - identifiant unique d'un enregistrement
  - Interdit les doublons
  - NOT NULL: interdit la valeur NULL
  - Un seul par table, mais peut regrouper plusieurs champs
- UNIQUE
  - Interdit les doublons
  - Peut prendre des valeurs NULL
  - Possibilité d'en avoir plusieurs

Les types d'index

- KEY / INDEX
  - Permet d'indexer des champs avec des valeurs identiques
- FULLTEXT
  - Optimise la recherche de mots dans du texte
  - Pour les champs de type CHAR, VARCHAR ou TEXT

Les types d'index

- Créer une table avec des index

```
CREATE TABLE etudiant (
    id_etudiant int unsigned NOT NULL auto_increment,
    ine char(11),
    nom char(30),
    prenom char(30),
    cv text,
    PRIMARY KEY (id_etudiant),
    UNIQUE (ine),
    KEY (nom, prenom),
    FULLTEXT (cv)
)engine=MyISAM;
```

Les types d'index (1/2)

Zend The php Company

- Recherche FULLTEXT

```
/* Recherche des étudiants ayant le mot clé php dans leur cv */
SELECT ine FROM etudiant
  WHERE MATCH(cv) AGAINST('php');

/* Recherche des étudiants ayant les mots clé php et certification
   dans leur cv */
SELECT ine FROM etudiant
  WHERE MATCH(cv) AGAINST('php certification');
```

Les types d'index (2/2)

- Recherche FULLTEXT

```
/* Recherche FULLTEXT booléenne (avec opérateurs) */

/* Recherche des étudiants ayant les mots clé php et mysql
   mais pas asp dans leur cv */
SELECT ine FROM etudiant
  WHERE MATCH(cv) AGAINST('+php +mysql -asp' IN BOOLEAN MODE);

/* Recherche d'une expression */
SELECT ine FROM etudiant
  WHERE MATCH(cv) AGAINST(' "J'aime le PHP" ' IN BOOLEAN MODE);
```

Table avec indexes (1/2)

- Si ce n'est pas déjà fait, à l'aide de phpMyAdmin créez une base 'magasin'
- Dans cette base, créez une table 'produits' avec 5 champs :
  - 'id' de type 'INT', clé primaire, auto-incrémenté
  - 'nom' de type VARCHAR(250)
  - 'description' de type TEXT
  - 'prix' de type DECIMAL
  - 'image' de type BLOB

Vous pouvez aussi utiliser d'autres outils (Mysql Query Browser, SQL Front, ...).

Table avec indexes (2/2)

Zend The php Company

- Mettez des indexes sur les champs suivants
  - nom
  - prix
- Faites en sorte que les champs suivants soient uniques
  - nom

Cette étape est généralement importante pour l'efficacité future de vos applications.

Zend
The *php* Company

Les bases de la POO

Introduction

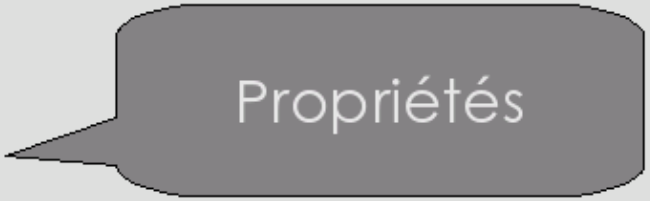
- Une classe / un objet
- Constructeur / destructeur
- Visibilité
- Différences entre PHP 4 et PHP 5
- Classes abstraites et interfaces
- Héritage / surcharge
- Sérialisation
- Clonage

Qu'est-ce qu'une classe ?

- Une classe peut être vue comme une abstraction.
- Elle contient des propriétés et des méthodes :
  - Propriété = variable de classe
  - Méthode = fonction de classe
- L'environnement de la classe est protégé :
  - Les propriétés et les méthodes peuvent être déclarées comme non accessibles en dehors de la classe.
  - Les propriétés ne se mélangent pas aux variables globales.

Exemple de classe


```
class TaskManager
{
    public $taskTitle;

    private $xmlData;

    public function displayTasks()
    {
        // ...
    }
    private function cleanTasks()
    {
        // ...
    }
}
```



Qu'est-ce qu'un objet ?

- Un objet est une instance de classe.
- Avec une classe, on peut créer autant d'objets qu'on veut (sauf si dans l'implémentation on décide de contrôler l'instanciation).
- Une classe est une déclaration, alors qu'un objet est contenu dans une variable, créé à partir de notre classe avec le mot clé « new ».
- Un objet est une instance de ma classe.

Exemple de création d'objets


```php
// Une classe Task
class Task
{
    public $title;
    public $description;
}
// Un objet "Task".
$task1 = new Task();
// Un deuxième objet "Task".
$task2 = new Task();
// Accès à un objet
$task1->title = "Comprendre la POO";
$task1->description = "La POO est magnifique car ...";
```

Une classe et ses objets

Une classe :

Des objets

Composition d'une classe

- La déclaration
  - Le 'moule' de la classe
- Les propriétés
  - Les 'variables' de la classe
- Les méthodes
  - Les 'fonctions' de la classe
- La classe mère, les interfaces
  - Elles font évoluer les possibilités et les fonctionnalités de la classe

Exemple de classe (1/2)

Zend The php Company

- Un exemple simple de classe

```php
<?php

// Déclaration de la classe
class User
{

    // Une propriété (variable de classe)
    var $name;

    // Une méthode (fonction de classe)
    function getName()
    {
        return ucfirst($this->name);
    }

}
```

Exemple de classe (2/2)

Zend The php Company

- Un exemple simple de classe

```
// Création d'un objet à partir de la classe
$pierre = new User();

// Modification d'une propriété
$pierre->name = 'pierre';

// Accès à une méthode
echo $pierre->getName();

// Affiche 'Pierre'
```

Création d'une classe

- Créez un fichier 03_Rectangle_creation.php
- Dans ce fichier, créez une classe 'Rectangle' (avec un R majuscule) avec les propriétés suivantes
  - $longueur
  - $largeur
- Créez deux objets 'Rectangle' à partir de cette classe
  - Remplissez leurs propriétés ($longueur et $largeur)
  - Affichez-les avec la fonction print_r()

Cette classe est destinée à contenir des données formatées qui sont plus faciles à manipuler sous forme d'objets que sous forme de tableaux.

Le constructeur

- Comment marche un constructeur ?
  - Le constructeur est une méthode dite "magique" : elle est appelée automatiquement.
  - Le constructeur est appelé lors de l'instanciation d'une classe, c'est à dire la création d'un objet à partir d'une classe.
- A quoi ça sert ?
  - On appelle souvent le constructeur pour charger des propriétés (par exemple une connexion à la base de données).

En PHP4, le constructeur est la méthode qui porte le nom de la classe. Cette variante est encore valable en PHP5 bien que dépréciée.

Le destructeur

- Comment marche un destructeur ?
  - De la même manière que le constructeur, le destructeur est appelé à la fin de la vie d'un objet.
  - Le destructeur sert généralement à faire des nettoyages.
- A quoi cela peut-il servir ?
  - Fermeture de la connexion à la base de données.
  - Nettoyage du contexte d'un objet (suppression de fichiers temporaires, vidage cache, etc.)

La fin de vie d'un objet peut subvenir à la fin du script, lorsque l'on supprime la variable qui contient l'objet (unset) ou lorsque l'on remplace son contenu (affectation).

Constructeur (1/2)

- Créez un fichier 04_Rectangle_constructeur.php
- Dans ce fichier, créez une classe 'Rectangle' (R majuscule) avec les propriétés $longueur et $largeur
- Créez un constructeur qui prend en paramètre deux variables $long et $larg
  - Vérifiez que les valeurs des paramètres sont bien numériques (is_numeric())
  - Si c'est le cas, transférez ces valeurs sur les propriétés $longueur et $largeur

Ici, l'utilisation d'un constructeur simplifie le remplissage des propriétés du rectangle.

Constructeur (2/2)

- Créez deux objets de type 'Rectangle' à partir de cette classe
  - Remplissez leurs propriétés ($longueur et $largeur) au moment de les créer
  - Affichez-les avec la fonction print_r()

La création d'un objet et le remplissage des propriétés sont effectués par la même instruction PHP.

Exemple (construct, destruct)

Zend The php Company

- Un constructeur et un destructeur

```
class DB
{

    var $dbConnexion = null;

    var function __construct()
    {
        // Connexion à la base de données
    }

    function __destruct()
    {
        // Déconnexion à la base de données
    }

}
```

La visibilité

- Application
  - La visibilité s'applique à une propriété ou une méthode
- Privé (private)
  - N'est accessible que dans la classe courante.
- Protégé (protected)
  - Est accessible dans la classe courante et dans les classes dérivées.
- Publique (public)
  - Est accessible partout, même à l'extérieur de la classe.

Visibilité (1/2)

- Créez un fichier 05_Rectangle_visibilite.php
- Dans ce fichier, créez une classe 'Rectangle' (avec un R majuscule) avec les propriétés privées $longueur et $largeur
- Puis créez :
  - Deux méthodes publiques getLongueur() et getLargeur() qui retournent $this->longueur et $this->largeur
  - Deux méthodes publiques setLongueur() et setLargeur() qui remplissent la longueur et la largeur

Les méthodes qui commencent par 'get' et 'set' sont appelées 'accesseurs' parce qu'elles permettent l'accès (filtré et sécurisé) aux propriétés d'un objet.

Visibilité (2/2)

- Créez un constructeur qui prend deux paramètres optionnels $long et $larg qui font appel à setLongueur() et setLargeur()
- Créez deux objets 'Rectangle' à partir de cette classe
  - Remplissez leurs propriétés avec les accesseurs 'set'
  - Affichez leurs propriétés avec les accesseurs 'get'

La création d'un objet et le remplissage des propriétés sont effectués par la même instruction PHP.

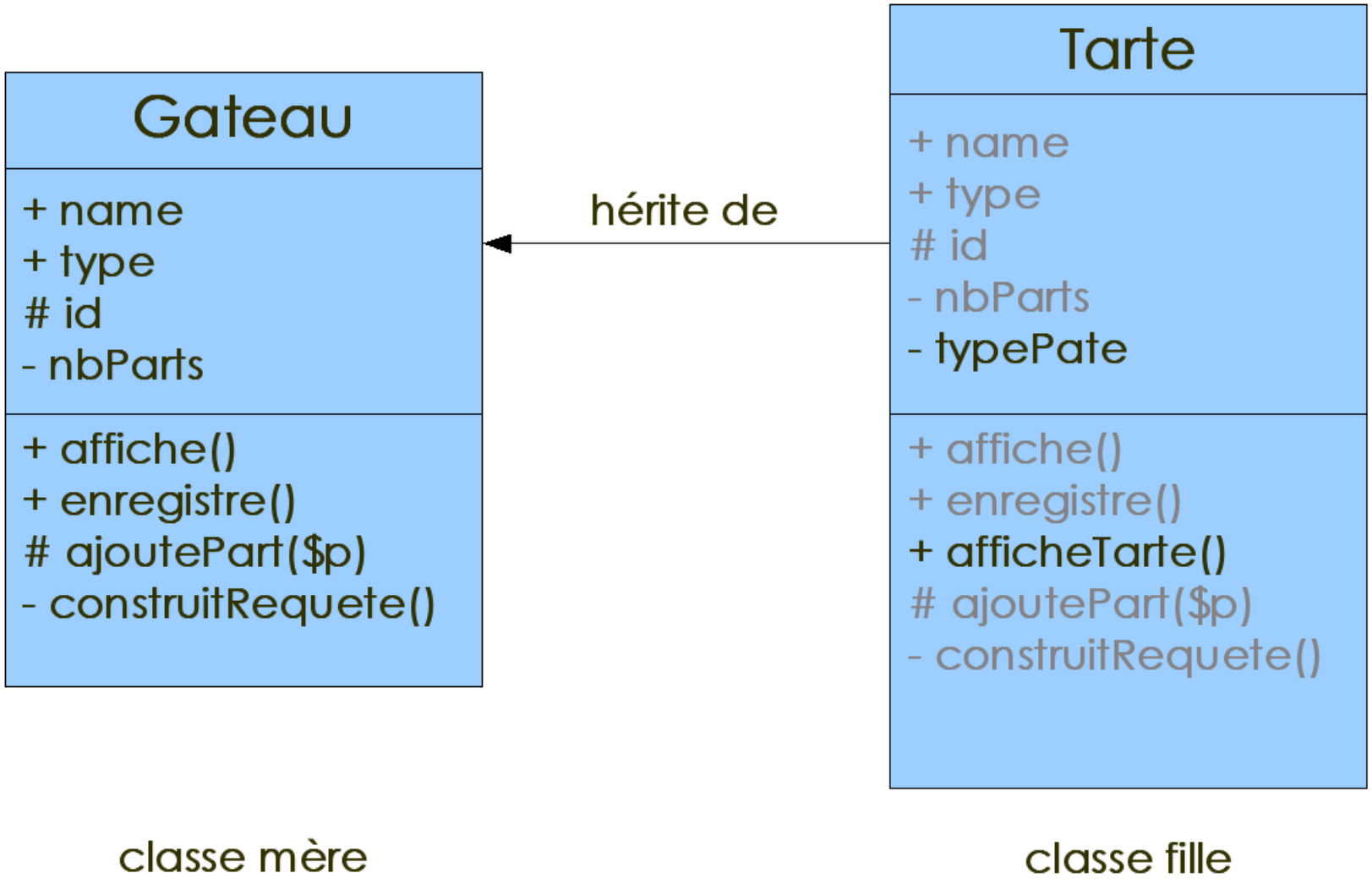
Différences entre PHP4 et PHP5

- En PHP 5, les passages d'objets se font par défaut par référence. (cf clone)
- En PHP 5, intervient la notion de visibilité.
- L'objet PHP 4 est limité, PHP 5 permet de faire de la POO digne de ce nom.

L'héritage

- L'héritage permet de hiérarchiser les classes.
- Les classes mères sont génériques, elles mettent en place des fonctionnalités communes à une ou plusieurs classes.
- Les classes filles sont spécifiques, elles héritent des fonctionnalités des classes mères.

L'héritage (UML)



Héritage (1/2)

- Exemple d'héritage

```php
<?php

class User // Une classe Utilisateur
{
    private $name;

    public function setName($name)
    {
        $this->name = $name;
    }

    public function getName()
    {
        return ucfirst($this->name);
    }
}
```

Héritage (2/2)

- Exemple d'héritage

```
// Une classe 'Administrator' qui hérite de 'User'
class Administrator extends User
{
    private $level = 'administrator';
}

// Administrator hérite de User donc on peut
// s'en servir comme d'un User

$admin = new Administrator();
$admin->setName('jean');
echo $admin->getName();
```

Héritage (1/3)

- Créez un fichier 06_heritage_Rectangle.php
- Dans ce fichier :
  - Créez une classe 'Rectangle' avec des propriétés privées 'longueur' et 'largeur' et leurs accesseurs (set, get)
  - Des méthodes 'getPerimetre' et 'getAire' de la classe 'Rectangle' qui renvoient respectivement le périmètre et l'aire du rectangle.
  - Créez un constructeur __construct($long, $larg) destiné à simplifier la création de l'objet 'Rectangle'

Note : dans la plupart des conventions à votre disposition, un fichier ne contient qu'une classe et le nom du fichier porte le nom de la classe.

Héritage (2/3)

- Créez un fichier 06_heritage_test.php
  - Dans ce fichier, incluez le précédent
  - Créez un objet $rectangle et affichez son périmètre et son aire

Evitez de créer les objets dans le même fichier que la classe. Cela surcharge le fichier et impose l'instanciation d'un objet qui n'est pas forcément utile.

Héritage (3/3)

- Créez un fichier 06_heritage_Carre.php
- Dans ce fichier :
  - Créez une classe 'Carre' qui hérite de la classe 'Rectangle'
  - Surchargez le constructeur : déclarez un constructeur __construct($cote) qui fait appel à celui de rectangle grâce au mot clé parent::__construct($long, $larg)
- Dans le fichier de test utilisé pour 'Rectangle'
  - Créez un objet $carre et affichez son périmètre et son aire. Ce mécanisme doit fonctionner car 'Carre' hérite de 'Rectangle'

Rappel : dans la plupart des cas, la classe mère est générique et la classe fille est spécifique.

La sérialisation

- Principe
  - Les objets en tant que tel ne peuvent être transmis d'un programme à l'autre via un réseau, car ils contiennent des caractères non imprimables
  - Sérialisation = transformation d'une donnée complexe (objet, tableau) en une chaîne de caractère imprimable
  - Désérialisation = transformation d'une chaîne sérialisée en sa donnée d'origine
- Utilité
  - Transmission de données complexes
  - Stockage de ces données en base ou dans des fichiers

Sérialisation

- Exemple de sérialisation, désérialisation

```php
<?php

class User
{
    public $name;
}

$user = new User();
$user->name = "Seth";

$serializedUser = serialize($user);
echo $serializedUser . "\n\n";

$user2 = unserialize($serializedUser);
print_r($user2);
```

Sérialisation (affichage)

- Résultat du script précédent

```
O:4:"User":1:{s:4:"name";s:4:"Seth";}

User Object
(
    [name] => Seth
)
```

Zend
The *php* Company

PDO

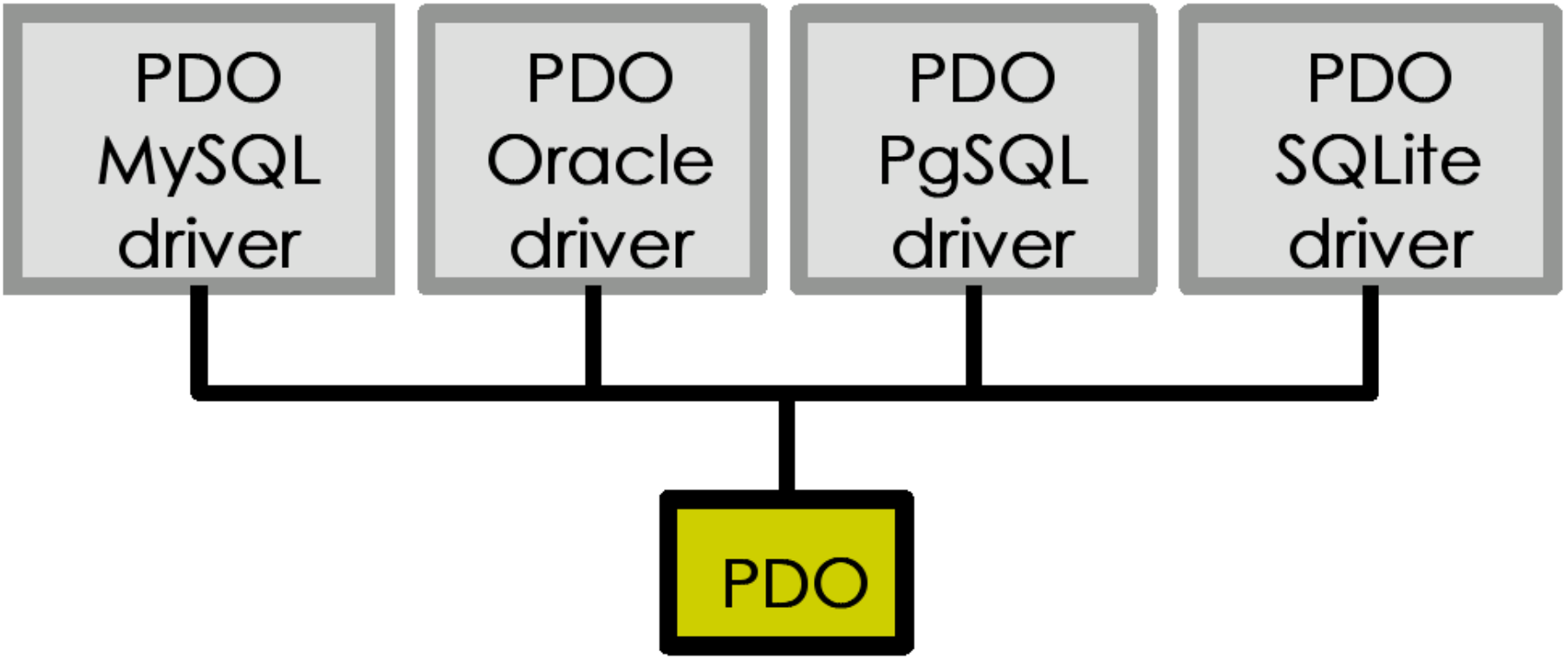
Introduction

- PDO est une interface d'accès aux bases de données.
- PDO n'est pas un système d'abstraction complet
  - Il ne manipule pas les requêtes.
  - Il met à disposition des classes et des fonctions communes à un ensemble de SGBD.
  - Si certains SGBD ne disposent pas de certaines fonctionnalités (gestion des transactions, requêtes paramétrées, etc.), il les simule.

Architecture / principe

- PDO fonctionne avec un ensemble d'extensions
  - Une extension PDO de base qui définit l'interface commune.
  - Une extension Driver PDO relative à chaque SGBD que l'on veut gérer.

Architecture / principe



Les classes de l'extension

Zend
The php Company

- La classe PDO gère l'accès aux SGBD ainsi que les fonctionnalités de base :
  - Connexions
  - Lancement des requêtes
- La classe PDOStatement gère une liste de résultats.
- La classe PDOException est une classe d'exception personnalisée interne pour PDO.

Connexion simple à PDO

Zend The php Company

- Déclaration du DSN, connexion et requête

```php
<?php

// DSN pour se connecter à MySQL
$dsn = 'mysql:host=localhost;dbname=mybase';

// Création d'un objet pour manipuler des requêtes
$dbh = new PDO($dsn, 'login', 'pass');

// Execution d'une requête SELECT
$result = $dbh->query('SELECT * from FOO');

// Itération sur les résultats d'une requête
foreach ($result as $row) {
    print_r($row);
}
```

Bases de PDO

- Créez un fichier 07_pdo_connexion.php
- Dans ce fichier
  - Créez une variable $dsn qui contient le dsn permettant de se connecter à la table 'produits' de la base 'magasin'
  - Connectez-vous en créant un objet $pdo
  - A l'aide de l'exemple précédent, faites une requête SELECT sur la table et affichez les données dans un tableau HTML

Pour changer de base de données ou de SGBD, il suffit de changer le DSN.

Requêtes préparées

- Les requêtes préparées sont un modèle de requête enregistré sur le serveur le temps de l'exécution du script (par opposition aux procédures stockées qui sont stockées de manière permanente).
- Avec les requêtes paramétrées il n'est plus nécessaire de se protéger des attaques par injection SQL car le serveur sait ce qu'il attend.

Requêtes préparées

- Exemple de requête préparée avec PDO

```
// On déclare la requête avec la méthode prepare
$stmt = $dbh->prepare('insert into ttt values (:name)');

// Puis on lie les paramètres à des variables
$stmt->bindParam(':name', $func);

// Enfin, on appelle la méthode execute
$funcs = get_extension_funcs('xml');
foreach ($funcs as $func) {
    $stmt->execute();
}
```

Gestion des transactions

- Gestion d'une transaction avec PDO

```
try {
    $dbh = new PDO('mysql:host=localhost;dbname=mybase', 'root', '',
    array(PDO::ATTR_PERSISTENT => true));
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $dbh->beginTransaction();
    $dbh->exec("insert into staff (id, first, last) values (23,
'Joe', 'Bloggs')");
    $dbh->exec("insert into salarychange (id, amount, changedate)
values (23, 50000, NOW())");
    $dbh->commit();

} catch (Exception $e) {
    $dbh->rollBack();
    echo "Échec : " . $e->getMessage();
}
```

Création classe d'accès (1/2)

- Créez un fichier 08_Db.php
- Créez une classe Db destinée à gérer les accès à la base de données. Cette classe comporte une méthode "execute" qui prend en paramètre une requête et des paramètres, et retourne un résultat sous forme de tableau.
- Si le paramètre est une requête, exécutez la requête et renvoyez le résultat. S'il y a un ou plusieurs paramètres, effectuez une requête préparée.

Vous pouvez définir plusieurs politiques de gestion d'erreurs avec PDO : pas d'erreurs, exceptions ou erreurs PHP.

Création classe d'accès (2/2)

- Si le paramètre est un tableau, exécutez les requêtes contenues dans ce tableau et renvoyez un tableau à plusieurs dimentions contenant les résultats.
- Si terminé : créez une méthode 'getResult' qui renvoit l'objet résultat (PDOStatement) donné par la méthode 'query' de PDO

Pour augmenter les performances, évitez d'itérer plusieurs fois sur des collections (tableau intermédiaire par exemple).

Zend
The *php* Company

POO Avancée

POO avancée

- Les types d'objets en UML
  - Entité, contrôle, dialogue
- Les interfaces
- Les classes abstraites
- Les méthodes et constantes magiques
- Les motifs de conception (design patterns)
  - Singleton, fabrique, façade, observateur, MVC

Types d'objets

- En UML, on peut différencier trois manières différentes d'utiliser un objet :
  - Entité : contient des données
  - Contrôle : renferme des fonctionnalités
  - Dialogue : propose une API simple à utiliser

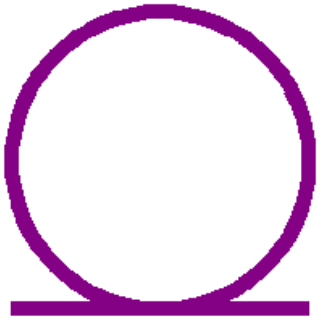
Une entitée

Zend
The php Company

- L'entité contient des données.

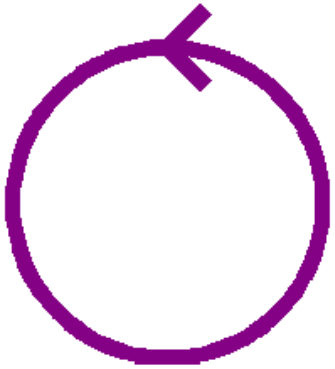
User
+ firstName + lastName # id - administrator
+ isAdmin() + setAdmin() # setId($p) - checkValidity()

```
class User
{
    public    $firstName;
    public    $lastName;
    protected $id;
    private   $administrator = false;

    public function isAdmin()
    {
        return $this->administrator;
    }

    public function setAdmin()
    {
        $this->administrator = true;
    }

    public function setId($id)
    {
        $this->id = (integer) $id;
    }

    public function checkValidity()
    {
        //...
    }
}
```



Un contrôle

- Le contrôle met en place des fonctionnalités.

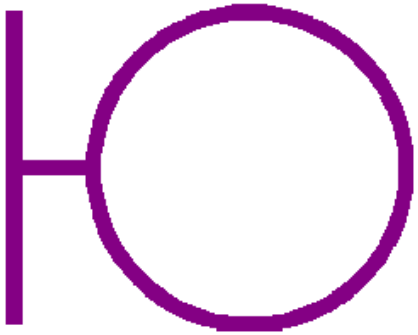


UserManager
+ addUser($user) + getUser($id) + getUsers($params) + isAdmin($user) + setAdmin($user) # cleanUsers($p) - deleteUser($user) - updateUser($user)

```
class UserManager
{
    public function addUser(User $user)
    {
        // ...
    }

    public function getUser($id)
    {
        // ...
    }

    // ...
}
```

Un dialogue

- Un dialogue permet un accès simple à un système complexe.



WebSiteAPI
nbPages # nbNews # title # description
+ getLastNews() + getLastPages() + getStats() + addPage($page) # update()

```
class WebSiteAPI
{
    protected $nbPages;
    protected $nbNews;
    protected $title;
    protected $description;

    public function getLastNews()
    {
        // ...
    }

    public function getLastPages()
    {
        // ...
    }
}
```

Les interfaces : en théorie

- Un héritage permet de regrouper des objets de manière verticale
  - Une classe mère regroupe plusieurs classes filles
  - Chaque classe fille hérite de fonctionnalités de la classe mère
- Une interface permet de regrouper des objets de manière transversale
  - Un groupe d'objets peut implémenter une interface
  - Une interface impose la présence de fonctionnalités dans ces objets
  - Si une classe mère implémente une interface, ses classes filles l'implémentent aussi par héritage

Les interfaces : en pratique

- Une interface se déclare comme une classe, avec le mot clé 'interface'
- Une interface ne contient que des déclarations de méthodes, sans contenu
  - Ces méthodes ainsi déclarées doivent oblitagoirement être redéclarées dans les classes qui implémentent l'interface

Exemple d'interface (1/3)

- Définit un objet de type 'géométrique'

```php
<?php

// Une inteface 'Surface_Interface' qui impose 'getAire()'
interface Surface_Interface
{
    public function getAire();
}

class Rectangle implements Surface_Interface
{
    // ...
    public function getAire()
    {
        // ...
    }
}
```

Exemple d'interface (2/3)

- Définit un objet de type 'géométrique'

```
// Implémente Geometric_Interface par héritage
// avec Rectangle
class Carre extends Rectangle
{
    // ...
}

// Une interface 'Volume' qui impose 'getVolume()'
interface Volume_Interface
{
    public function getVolume();
}
```

Exemple d'interface (3/3)

- Définit un objet de type 'géométrique'

```
// Rectangle implémente Geometric_Interface
class Cube implements Surface_Interface, Volume_Interface
{
    // ...
    public function getAire()
    {
        // ...
    }

    public function getVolume()
    {
        // ...
    }

}
```

Interfaces

- Créez un fichier 09_interfaces.php
- Reprenez les interfaces et les classes de l'exemple précédent
- Remplissez les blancs pour que les objets renvoient les aires et volumes voulus
- A la fin du fichier, créez un objet $rectangle et un objet $carre, affichez leur aires et le volume du cube

L'utilisation d'interfaces ne permet pas l'implémentation de fonctionnalités en soi. Cela permet juste de forcer le respect de certaines contraintes.

Les classe abstraite

- Une classe abstraite ne peut être instanciée
  - En d'autres termes : on ne peut pas créer un objet avec une classe abstraite
- Une classe abstraite est destinée à être utilisée comme classe mère d'autres classes
- La déclaration d'une classe abstraite se fait avec le mot clé 'abstract'
  - abstract class Parallelogramme { ... }

Les méthodes magiques

Zend The php Company

- Sont appelées automatiquement :
  - Quand un objet se crée : __construct()
  - Quand un objet meurt : __destruct()
  - Quand un objet est dupliqué : __clone()
  - Quand on appelle une méthode inconnue : __call()
  - Quand on affecte une propriété inconnue : __set()
  - Quand on veut la valeur d'une prop. inc. : __get()
  - Quand on met un 'echo' devant l'objet : __toString()
  - Quand on sérialise un objet : __sleep()
  - Quand on désérialise un objet : __wakeup()

Les constantes magiques

- Donnent des informations sur :
  - Le chemin du fichier PHP courant : __FILE__
  - Le nom de la fonction courante : __FUNCTION__
  - Le nom de la classe courante : __CLASS__
  - Le numéro de ligne dans le fichier PHP : __LINE__

Constantes magiques

- Utilisation fréquente de __FILE__

```php
<?php

// Trouver le chemin vers le fichier PHP courant
$path = dirname(__FILE__);

// Pour éviter d'appeller dirname à chaque fois,
// vous pouvez créer une fonction ou une méthode
function getPath()
{
    static $path = null;

    if ($path === null) {
        $path = dirname(__FILE__);
    }
    return $path;
}
```

Les motifs de conception

- Origine
  - A force de programmer en objet, les développeurs se retrouvaient souvent avec les mêmes constructions
  - Ils ont décidé de mettre un nom sur ces constructions et de les répertorier pour mieux les étudier
- Implémentation avec PHP
  - La POO de PHP5 permet de construire toute sorte de motif de conception
  - Mais PHP est un langage interprété, gare aux multiplications d'objets !

Les motifs usuels (1/2)

- Le singleton
  - Une classe qui ne peut être instanciée qu'une seule fois
- La fabrique
  - Un objet qui 'fabrique' d'autres objets pour vous
- Le proxy
  - Un objet qui contrôle l'accès à une ressource
- La façade
  - Un objet qui fournit une interface simple pour accéder à un système complexe

Les motifs usuels (2/2)

- L'observateur
  - Un objet qui associe des événements à des actions
- M.V.C. (Modèle, Vue, Contrôleur)
  - Un motif qui gère l'interaction entre une application et un utilisateur (très utilisé pour le web)

Construire un singleton

Zend The php Company

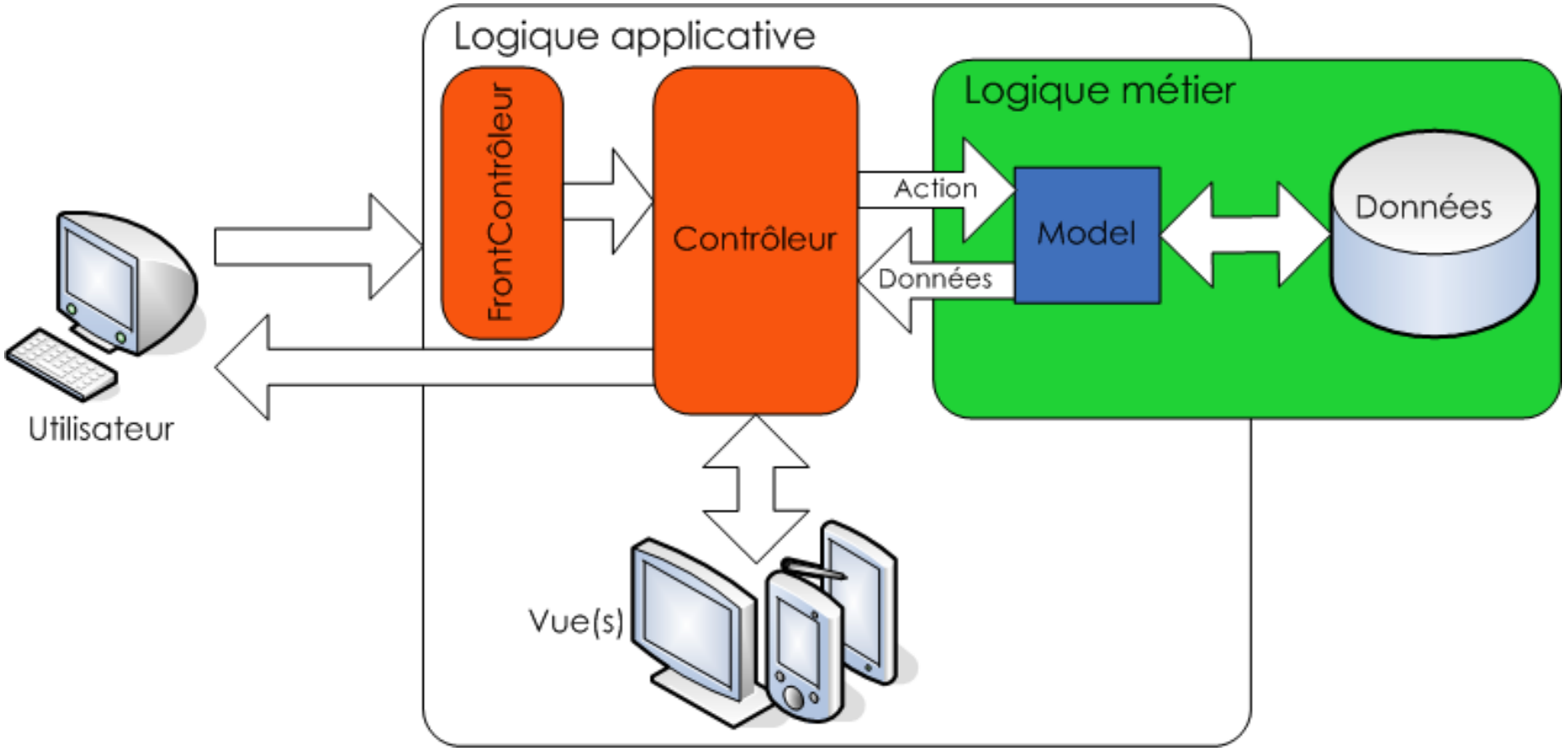
- Créez un fichier 10_singleton.php
  - Dans ce fichier, créer une classe 'User_Singleton' contenant une propriété privée 'name' et ses deux accesseurs
- Transformation en singleton
  - En utilisant __construct, faites en sorte que l'on ne puisse pas instancier la classe
  - En utilisant __clone, faites en sorte qu'on ne puisse pas cloner l'objet
  - Déclarez une méthode publique 'getInstance()' qui renvoie l'objet unique de la classe 'User_Singleton'

Cet exercice nécessite une réflexion préalable. Une bonne utilisation des motifs de conception nécessite un bon niveau et une pratique courante de la POO.

Introduction à MVC

- MVC est très utilisé dans les applications web
- Il est décomposé en 3 parties
  - Le contrôleur : qui gère la navigation
  - Le modèle : qui gère la structuration et l'accès aux données
  - La vue : qui gère la partie 'présentation' (visible)

Principe de MVC



MVC et formulaires

- La vue
  - Elle gère l'aspect du formulaire : son design
- Le contrôleur
  - Il récupère les données du formulaire pour les traiter
- Le modèle
  - Il fait l'interface entre la base de données d'une part, le contrôleur et la vue d'autre part

Attention : certaines implémentations de MVC ne permettent pas au modèle d'accéder aux données, c'est le contrôleur qui le fait. Le véritable rôle du modèle est de STRUCTURER les données.

Précisions sur MVC

- La plupart des frameworks PHP utilisent MVC
  - Zend Framework, Symfony, ...
- MVC définit la structure globale de l'application
- La partie MVC est spécifique à l'application, contrairement aux librairies
- Les modèles et les contrôleurs sont généralement des classes
- Les vues sont généralement des templates HTML ou PDF

Zend
The *php* Company

La gestion des exceptions

Principe des exceptions

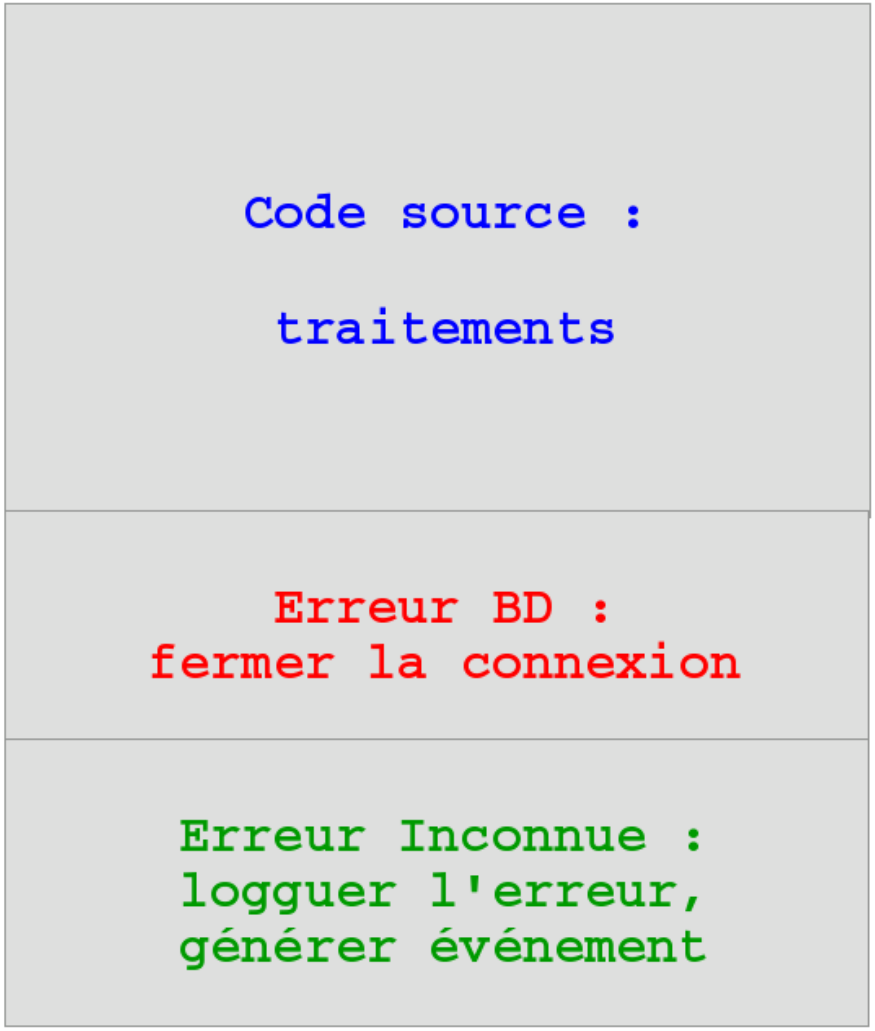
- Les exceptions sont un moyen élégant de gérer les erreurs d'exécution.
- Une exception définit un espace dans lequel les erreurs sont détectées.
- La gestion des erreurs peut être capitalisé dans des processus communs à l'ensemble des erreurs ou à des catégories d'erreurs.

Utilité

- Les exceptions permettent de mieux organiser la gestion des erreurs et des événements.
- Les exceptions contribuent à augmenter la qualité d'un code source.

Illustration

- En bleu : on effectue les traitements
- En rouge : lors des traitements, une erreur BD est apparue ? On la traite.
- En vert : lors des traitements, une erreur inconnue est apparue ? On la traite.



Implémentation des exceptions

Zend The php Company

- Une implémentation classique en PHP5

```php
<?php

try {
    // traitement
} catch (PDOException $e) {
    // traitement d'une erreur base de données
} catch (Exception $e) {
    // Traitement d'une erreur inconnue
}
```

Contexte d'une exception

- Données disponibles à travers une exception

```
object(Exception)[2]
  protected 'message' => 'Impossible de lire les tâches.' (length=30)
  private 'string' => '' (length=0)
  protected 'code' => 0
  protected 'file' => '/var/www/html/tools/zend/TaskManager.php'
(length=40)
  protected 'line' => 10
  private 'trace' =>
    array
      0 =>
        array
          'file' => '/var/www/html/tools/zend/functions.php'
(length=38)
          'line' => 8
          'function' => 'getTaskList' (length=11)
          'class' => 'TaskManager' (length=11)
```

Contexte d'une exception (suite)

- Données disponibles à travers une exception

```
          'type' => '->' (length=2)
          'args' =>
            array
              empty
      1 =>
        array
          'file' => '/var/www/html/tools/zend/exception.php'
(length=38)
          'line' => 6
          'function' => 'getMyTaskList' (length=13)
          'args' =>
            array
```

Exceptions base (1/2)

- Créez un fichier 11_display_exception.php
- Dans ce fichier
  - Créez une fonction "display" qui prend en paramètre une variable et l'affiche avec echo.
  - Faites en sorte que si la variable n'est pas une chaîne de caractères, une exception soit lancée (throw) avec un message d'erreur.

Les exceptions sont une manière élégante de gérer les erreurs de manière globale.

Exceptions base (2/2)

- Testez l'appel de display avec un mauvais paramètre.
- Testez l'appel de display avec un mauvais paramètre avec try et catch. Détectez l'erreur et affichez un message personnalisé dans le bloc catch.

N'hésitez pas à faire beaucoup de tests sur les exceptions et à regarder les exemples en ligne.

Exceptions personnalisées

- Une exception personnalisée permet d'avoir la main sur le traitement des erreurs ou des événements traités par les exceptions.
- La première étape consiste à créer une classe d'exception personnalisée.
- Dans un deuxième temps, on peut séparer la gestion des exceptions dans des blocs "catch" différents.

Implémentation

Zend
The php Company

```php
/**
 * Définition d'une classe d'exception personnalisée.
 */
class TaskException extends Exception
{
    public function __construct($message, $code = 0) {
        // ... traitement personnalisé lié à la gestion des tâches ...
        parent::__construct($message, $code);
    }

    public function __toString() {
        return __CLASS__ . ": [{$this->code}]: {$this->message}\n";
    }

    public function getTaskStatus() {
        echo "Une fonction personnalisée pour ce type d'exception\n";

    }
```

```php
try {
    throw new TaskException('Impossible d\'ajouter la tâche.', 1);
} catch (TaskException $t) {
    echo $t;
}
```

```
TaskException: [1]: Impossible d'ajouter la tâche.
```

Séparer les traitements

Zend The php Company

- Séparation à l'aide de classes d'exceptions

```
try {
    // ... traitement ...
} catch (TaskException $t) {
    echo $t;
} catch (DBException $db) {
    $db->closeConnexion();
    echo $db->getMessage();
} catch (Exception $e) { // Toujours intercepter Exception
    echo $e->getMessage();
}
```

Exceptions avancé (1/2)

Zend The php Company

- Créez un fichier 12_display_exception_pers.php avec:
  - une classe 'Display_Exception' qui hérite de 'Exception'
  - une fonction "display" qui lance une exception 'Display_Exception' en cas d'erreur.
- Testez display avec un bloc try et deux blocs catch
  - catch (Display_Exception $e) : récupération des exceptions personnalisées
  - catch (Exception $e) : récupération des autres exceptions

La création d'exceptions personnalisées permet de gérer plus finement les erreurs, c'est à dire de les trier et de traiter différents types d'erreurs séparément.

Exceptions avancé (2/2) (+)

- Testez avec display et en lançant manuellement une exception de type "Exception".
- Imbriquez un bloc "try/catch" dans un bloc "try" de manière à ce que l'exception "Display_Exception" génère une exception "Exception".

Pour plus d'informations sur les exceptions, rendez-vous sur la documentation en ligne :
http://fr.php.net/manual/fr/language.exceptions.php

Zend
The *php* Company

InnoDB : Moteur transactionnel

Présentation

- Edité par Innobase Oy (Oracle)
- Distribué sous licence GPL
- Moteur transactionnel
  - Début transaction: START TRANSACTION
  - Fin transaction: COMMIT ou ROLLBACK
- Support des clés étrangères (FOREIGN KEY)
- Support de l'intégrité référentielle
- Ne supporte pas les index fulltext

Clé étrangère

- Permet de déclarer des relations entre les colonnes dans différentes tables
- L'intégrité est maintenue par l'interdiction d'actions qui pourraient violer ces relations
- Les INSERT, UPDATE ou DELETE qui violent l'intégrité ne sont pas autorisées
- Syntaxe:
  - CONSTRAINT nom_clé_étrangère FOREIGN KEY (nom_colonne_enfant, ...) REFERENCES nom_table_parent (nom_colonne_parent, ...)

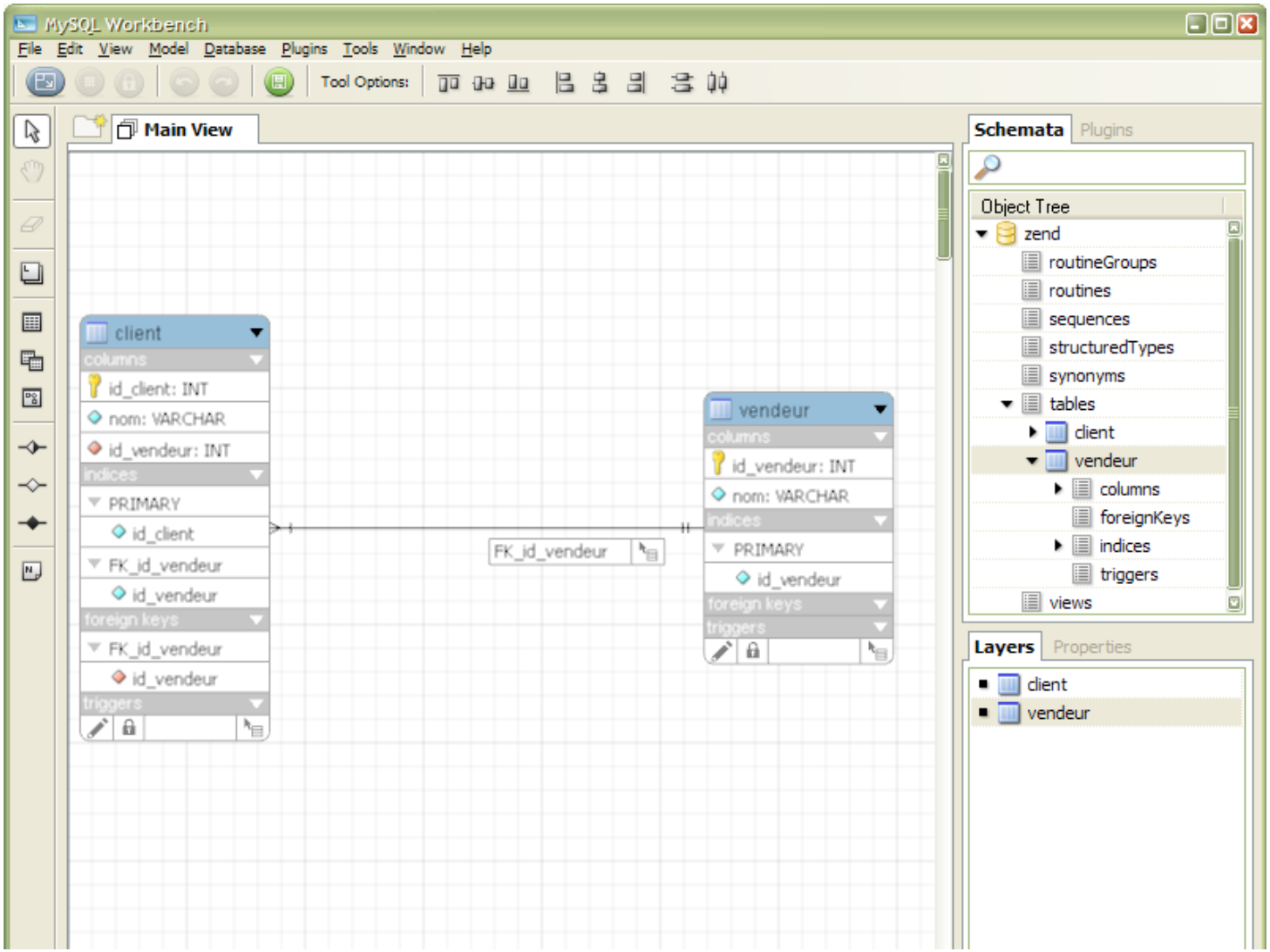
Intégrité référentielle

- Lors d'un DELETE ou d'un UPDATE dans la table parente, l'intégrité est vérifiée dans la table enfant, grâce aux clauses ON UPDATE/ON DELETE
- Les valeurs possibles sont:
  - CASCADE: Le DELETE ou l'UPDATE aura lieu également dans la table enfant
  - SET NULL: Les enregistrements liés aux lignes parentes prendront la valeur NULL
  - NO ACTION/RESTRICT: Le DELETE ou l'UPDATE ne sont pas autorisés lorsqu'il existe des enregistrements enfants. C'est la valeur par défaut

Intégrité référentielle

- Syntaxes:
  - [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
  - [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]

Relation entre deux tables



Clé étrangère

- Relation entre deux tables

```
CREATE TABLE vendeur (
    id_vendeur int(10) unsigned NOT NULL auto_increment,
    nom varchar(45) NOT NULL default '',
    PRIMARY KEY (id_vendeur)
) ENGINE=InnoDB;
-- La table client est liée à vendeur par le champ id_vendeur:
CREATE TABLE client (
    id_client int(10) unsigned NOT NULL auto_increment,
    nom varchar(45) NOT NULL default '',
    id_vendeur int(10) unsigned default NULL,
    PRIMARY KEY (id_client),
    KEY FK_id_vendeur (id_vendeur),
    CONSTRAINT FK_id_vendeur FOREIGN KEY (id_vendeur)
      REFERENCES vendeur (id_vendeur)
    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Clé étrangère (1/3)

- Si ce n'est pas déjà fait, à l'aide de phpMyAdmin créez une base 'magasin'
- Dans cette base, créez une table InnoDB 'produits' avec 5 champs :
  - 'id' de type 'INT', clé primaire, auto-incrémenté
  - 'nom' de type VARCHAR(250)
  - 'description' de type TEXT
  - 'prix' de type DECIMAL
  - 'image' de type BLOB

Si vous possédez déjà cette table en MyISAM ou tout autre moteur de stockage, vous pouvez effectuer le transfert en InnoDB directement avec phpMyAdmin.

Clé étrangère (2/3)

- A l'aide de phpMyAdmin créez une base 'categories'
- Dans la base 'magasin', créez une table 'categories' avec 2 champs :
  - 'id' de type 'INT', clé primaire, auto-incrémenté
  - 'categorie' de type VARCHAR(250)
- Ajoutez deux catégories (par exemple : livre, cd-rom)

Pour optimiser vos futures requêtes, vous pouvez également créer un index sur 'categorie' et définir ce champ comme unique.

Clé étrangère (3/3)

- Dans la table 'produits', ajoutez un champ 'id_categorie' qui fait référence au champ 'id' de la table 'categories'
- Ajoutez plusieurs produits.
- Tentez également d'ajouter un produit sans catégorie ou avec une catégorie inexistante.

La contrainte d'intégrité doit être forte. En d'autres termes, le remplissage d'un produit qui n'appartient pas à une catégorie connue doit être impossible.

Les transactions

- Groupe de requêtes SQL rassemblées en une opération atomique
- InnoDB supporte la norme ACID
  - **A**tomicité (toutes les requêtes de la transaction sont executées ou aucune ne l'est)
  - **C**ohérence (la cohérence de la base est préservée)
  - **I**solation (les transactions sont isolées les unes des autres)
  - **D**urable (mécanisme de récupération automatique des transactions validées en cas de crash)

Les transactions

- Une transaction débute avec START TRANSACTION ou BEGIN
- Elle se termine par :
  - COMMIT: validation de la transaction
  - ROLLBACK: annulation de la transaction
  - Ou alors elle est validée implicitement avec, par exemple, une instruction DDL (create table,...)
- InnoDB prend également en charge les points de sauvegarde (SAVEPOINT)

- Transaction validée

```
/* Début de la transaction */
START TRANSACTION;

/* suite de reqêtes */
UPDATE produit SET qte=qte-1 WHERE id_produit = 4126;

INSERT INTO log (date, id) VALUES (now(), 4126);

/* Validation: Fin de la transaction */
COMMIT;
```

Zend
The php Company

- Transaction annulée partiellement

```
START TRANSACTION; /* Début de la transaction */

INSERT INTO vendeur (nom) VALUES ('Zend');

INSERT INTO vendeur (nom) VALUES ('Steven');

SAVEPOINT S1; /* Point de sauvegarde */

INSERT INTO vendeur (nom) VALUES ('Guillaume');

INSERT INTO vendeur (nom) VALUES ('Olivier');

ROLLBACK TO SAVEPOINT S1; /* Annulation partielle de la transaction
jusqu'au point de sauvegarde */

SELECT nom FROM vendeur; /* Affiche Zend & Steven */
```

Zend
The *php* Company

Flux XML

Qu'est-ce que XML ?

- XML est un standard permettant d'organiser / stocker des données.
- La plupart des structures de données peuvent être traduites en XML.

Exemple de fichier XML

- Un fichier XML simple

```
<document id="12">
    <author>Guillaume Ponçon</author>
    <title>Best practices PHP 5</title>
    <chapter>
        <title>Les flux XML</title>
        <paragraph type="introduction">...</paragraph>
    </chapter>
</document>
```

Les extensions XML

- Il existe 3 implémentations natives liées à la manipulation de documents XML en PHP
- L'extension SimpleXML
  - Simple à utiliser, permet la lecture, la modification et la suppression de données.
- L'extension DOMXML
  - Verbeux, permet la lecture, la modification et la suppression de données.
- Support XML natif : SAX
  - Permet la lecture rapide mais pas l'écriture.

L'extension SimpleXML

- SimpleXML est une extension de manipulation de flux XML introduite dans PHP 5.
- Elle est facile à mettre en oeuvre et se base sur le même backend que DOMXML : le passage de DOM à SimpleXML et inversement n'a quasiment aucun coût d'exécution.
- SimpleXML permet de manipuler un flux à travers des objets créés à partir des noms de balises XML.

Avantages & inconvénients

- Avantages
  - Simplicité d'utilisation
  - Permet toutes les opérations (lecture, écriture, suppression, création, enregistrement).
  - Gère XPath et permet le passage à DOM en une seule instruction.
- Inconvénients
  - Nécessité de charger le flux XML en entier avant de l'éditer.
  - Moins permissif que SAX (XML strict requis).

Exemple de lecture


```
$xml = utf8_encode("<document id=\"12\">
    <author>Guillaume Ponçon</author>
    <title>Best practices PHP 5</title>
    <chapter>
        <title>Les flux XML</title>
        <paragraph>paragraphe 1</paragraph>
        <paragraph>paragraphe 2</paragraph>
    </chapter>
    <chapter>
        <title>Les tableaux</title>
        <paragraph>paragraphe 3</paragraph>
    </chapter>
</document>");

$simpleXml = simplexml_load_string($xml);
echo '+ ' . $simpleXml->title . "\n";
echo '  by ' . $simpleXml->author . "\n\n";
foreach ($simpleXml->chapter as $chapter) {
    echo '- ' . $chapter->title . "\n";
    foreach ($chapter->paragraph as $paragraph) {
        echo '  -> ' . $paragraph . "\n";
    }
    echo "\n";
}
```

```
+ Best practices PHP 5
  by Guillaume Ponçon

- Les flux XML
  -> paragraphe 1
  -> paragraphe 2

- Les tableaux
  -> paragraphe 3
```

Exemple de modification

```php
$simpleXml = simplexml_load_string($xml);
$simpleXml->title = "Bonnes pratiques PHP 5";
$newChapter = $simpleXml->addChild('chapter');
$newChapter->addChild('title', 'Programmation objet');
$newChapter->addChild('paragraph', 'La POO est magnifique ...');
```

```
+ Bonnes pratiques PHP 5
  by Guillaume Ponçon
- Les flux XML
  -> paragraphe 1
  -> paragraphe 2
- Les tableaux
  -> paragraphe 3
- Programmation objet
  -> La POO est magnifique ...
```

Exemple avec Xpath


```php
$result = $simpleXml->xpath('/document/chapter/paragraph');
foreach ($result as $paragraph) {
    echo "-> $paragraph\n";
}
```

Résultat :

```
-> paragraphe 1
-> paragraphe 2
-> paragraphe 3
-> La POO est magnifique ...
```

Exemple de lecture RSS


```php
$xml = simplexml_load_file('http://www.afup.org/backend.php3');
foreach ($xml->channel->item as $item) {
    echo '- ' . $item->date . ' : ' . utf8_decode($item->title) . "\n";
}
```

```
- 2006-05-12 11:06:38 : Appel à conférenciers
- 2006-05-10 17:24:48 : Evaluation de la certification PHP de Zend par l'AFUP
- 2006-04-24 12:36:54 : L'AFUP sera présent au salon Webmasters Expo
- 2006-04-20 22:12:15 : Revue de Presse / PHP Avril 2006
- 2006-03-23 11:55:38 : 06/04/2006 : Programmation Orientée Aspect
- 2006-03-20 11:39:42 : 29/03/2006 : Le Stack LAMP dans les Entreprises Modernes
- 2006-03-19 16:40:55 : Résumé de la conférence "Clients riches avec XUL"
- 2006-03-13 11:16:30 : Flickr, le service de partage de photo de Yahoo ! utilise PHP
- 2006-03-06 09:16:58 : PHP Québec annonce la Conférence PHP Québec du 29 au 31 Mars 2006
- 2006-02-09 11:00:17 : 02/03/2006 : clients riches avec XUL
```

L'extension DOMXML

- DOMXML est une extension stable et efficace qui permet de gérer la lecture, l'écriture, la suppression, l'enregistrement.
- Cette extension est disponible dans PHP 4 et PHP 5. En revanche, elle a été stabilisée dans PHP 5.
- Gère les requêtes XPath et les transformations XSLT.
- Son utilisation est verbeuse et pas toujours très intuitive au début.

DOMXML : lecture


```php
$xml = utf8_encode("<document id=\"12\">
    <author>Guillaume Ponçon</author>
    <title>Best practices PHP 5</title>
    <chapter>
        <title>Les flux XML</title>
        <paragraph>paragraphe 1</paragraph>
        <paragraph>paragraphe 2</paragraph>
    </chapter>
    <chapter>
        <title>Les tableaux</title>
        <paragraph>paragraphe 3</paragraph>
    </chapter>
</document>");

$dom = new DOMDocument();
$dom->LoadXML($xml);
$title = $dom->getElementsByTagName('title');
echo "Chapitres de " . $title->item(0)->nodeValue . " : \n";
foreach ($dom->getElementsByTagName('chapter') as $element) {
    $titles = $element->getElementsByTagName('title');
    echo "\n- " . $titles->item(0)->nodeValue . "\n";
    $paragraphs = $element->getElementsByTagName('paragraph');
    foreach ($paragraphs as $paragraph) {
        echo '  * ' . $paragraph->nodeValue . "\n";
    }
}
```

```
Chapitres de Best practices PHP 5 :

- Les flux XML
  * paragraphe 1
  * paragraphe 2

- Les tableaux
  * paragraphe 3
```

DOMXML : écriture


```php
$dom = new DOMDocument();
$dom->LoadXML($xml);
$title = $dom->getElementsByTagName('title');
$title->item(0)->nodeValue = "Bonnes pratiques PHP 5";
$element = $dom->createElement('category', 'PHP');
$rootElement = $dom->getElementsByTagName('document');
$rootElement->item(0)->appendchild($element);
$dom->save('/tmp/document.xml');
echo file_get_contents('/tmp/document.xml');
```

```xml
<?xml version="1.0"?>
<document id="12">
    <author>Guillaume Ponçon</author>
    <title>Bonnes pratiques PHP 5</title>
    <chapter>
        <title>Les flux XML</title>
        <paragraph>paragraphe 1</paragraph>
        <paragraph>paragraphe 2</paragraph>
    </chapter>
    <chapter>
        <title>Les tableaux</title>
        <paragraph>paragraphe 3</paragraph>
    </chapter>
    <category>PHP</category>
</document>
```

L'extension SAX

- L'extension SAX permet la lecture partielle ou totale d'un fichier.
- SAX est le parseur le plus efficace en lecture :
  - Il consomme peu de ressources.
  - Il effectue une lecture linéaire.
- SAX est très permissif, il est capable de lire des fichiers HTML mal formés.
- Il n'est pas possible d'enregistrer avec SAX, à moins de reproduire par programmation le document à partir d'un contenu lu avec les fonctions de l'extension.

SAX : lecture

- Transformation d'un flux XML en tableau

```
$parser = xml_parser_create('utf-8');
xml_parse_into_struct($parser, $xml, $table);
unset($parser);
$level = 0;
foreach ($table as $v) {
  $v['value'] = utf8_decode($v['value']);
  if ($v['type'] == 'open')
    echo str_repeat(' ', $level++) . '[' . $v['tag'] . "]\n";
  if ($v['type'] == 'close')
    echo str_repeat(' ', $level--) . '[/' . $v['tag'] . "]\n";
  if ($v['type'] == 'complete') {
    echo str_repeat(' ', $level + 1) . '[' . $v['tag'];
    echo ']' . $v['value'] . '[/' . $v['tag'] . "]\n";
  }
}
```

SAX : lecture (résultat)

- Transformation d'un flux XML en tableau

```
[DOCUMENT]
    [AUTHOR]Guillaume Ponçon[/AUTHOR]
    [TITLE]Best practices PHP 5[/TITLE]
  [CHAPTER]
      [TITLE]Les flux XML[/TITLE]
      [PARAGRAPH]paragraphe 1[/PARAGRAPH]
      [PARAGRAPH]paragraphe 2[/PARAGRAPH]
  [/CHAPTER]
  [CHAPTER]
      [TITLE]Les tableaux[/TITLE]
      [PARAGRAPH]paragraphe 3[/PARAGRAPH]
  [/CHAPTER]
[/DOCUMENT]
```

SAX : lecture avec handlers

- Un fichier XML peut être parcouru via des handlers, c'est à dire des fonctions qui sont automatiquement appelées au fur et à mesure du parsing :
  - une fonction lorsque le parseur rencontre une balise ouvrante ;
  - une fonction lorsque le parseur rencontre une balise fermante ;
  - une fonction lorsque le parseur détecte des données.
- Vous devez vous-même définir les fonctions appelées.

Création d'une application (1/3)

- Nous allons créer ici une petite application qui affiche des informations sur les résultats des matchs de foot (ou toute autre information contenue dans un flux rss : actualités, blog, etc.) en HTML
- Proposer dans un premier temps une idée d'architecture pour cette application
  - Classe(s) à mettre en oeuvre ?
  - Fichiers à créer ?
- Les règles à respecter sont mentionnées en 2/3

Bien penser quels fichiers, quelles fonctions, quelles classes vont être créés avant de développer est important.

Création d'une application (2/3)

Zend The php Company

- Les différentes parties de votre application seront les suivantes
  - La partie 'métier' s'occupera de récupérer les données à traiter
  - La partie 'présentation' s'occupera de formatter les données dans du code HTML pour l'affichage
  - Pour la partie 'métier', on utilisera SimpleXML dans un premier temps
  - Pour la partie 'présentation', on utilisera un fichier php qui contient le HTML et très peu de PHP

Tentez de séparer un maximum le code HTML du code PHP. Si vous avez terminé, vous pouvez essayer de faire la même chose avec SAX ou DOM.

Création d'une app. (3/3) (+)

- Partie avancée
  - Créez une classe personnalisée 'Rss_Exception' qui gèrera toutes les erreurs. L'affichage du message de l'exception doit se faire en HTML.
  - Remaniez votre partie 'métier' dans une classe 'RssManager' si nécessaire
  - Créez une classe 'RdfManager' qui fait la même chose que 'RssManager' pour les flux Rdf
  - Créez une classe 'FeedBuilder' qui renvoit un objet RdfManager ou RssManager en fonction des caractéristiques du flux passé en paramètre

SimpleXML permet de gérer 95% des applications. Lorsque votre fichier XML devient plus complexe (espaces de noms), l'utilisation de DOMXML est recommandée.

Zend
The *php* Company

Optimisation SQL & Architecture

Optimisation

- Plusieurs axes d'optimisation des performances pour une base de données
- L'optimisation des requêtes sql offre souvent le plus grand gain
- Un gain appréciable est également possible en optimisant la structure de la base de données
- Le plus souvent, les réponses ne sont pas spécifiques à MySQL

MySQL ne peut utiliser d'index

- Clause LIKE qui commence par un joker
  - SELECT ville FROM client WHERE nom LIKE '%nd';
- En n'utilisant pas la première partie de l'index:
  - SELECT * FROM table WHERE idx_part2=123;
- Si MySQL décide que la lecture complète de la table est plus rapide que l'utilisation de l'index

MySQL ne peut utiliser d'index

- Si la partie de gauche ne contient pas d'expression et si la partie de droite est une constante, l'exécution sera plus rapide:
  - SELECT id FROM produit WHERE prix*1.5 > 1000; => n'utilise pas l'index
  - SELECT id FROM produit WHERE prix > 1000/1.5; => peut utiliser l'index

Optimisation de l'index

- Un index sur la colonne entière n'est pas toujours nécessaire
- Indexer le préfixe de la colonne: gain de place et de performance
  - CREATE INDEX idx_nom ON client (nom(10)); /* index sur les 10 premiers caractères */
- Utiliser les premières colonnes des index composés
  - KEY(a,b) servira pour:
  - SELECT ... WHERE a='x' OR b='y'
  - SELECT ... WHERE a='z'
- Index FULLTEXT pour la recherche textuelle

EXPLAIN: optimiser sa requête

- EXPLAIN affiche le plan d'exécution d'une requête SELECT
  - Ordre de lecture des tables
  - Opérations de lecture effectuées
  - Quels index auraient pu être utilisés
  - Quels index sont utilisés
  - Références entre les tables
  - Estimation du nombre de lignes récupéré par l'optimiseur

EXPLAIN: optimiser sa requête

Zend The php Company

```
D:\MySQL\MySQL50\bin\mysql.exe
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 5.0.24a-community-nt-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

11:21 root$(none)> use zend;
Database changed
11:21 root$zend> desc vendeur;
+------------+------------------+------+-----+---------+----------------+
| Field      | Type             | Null | Key | Default | Extra          |
+------------+------------------+------+-----+---------+----------------+
| id_vendeur | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| nom        | varchar(45)      | NO   |     |         |                |
+------------+------------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)

11:21 root$zend> EXPLAIN SELECT nom FROM vendeur WHERE id_vendeur=2;
+----+-------------+---------+-------+---------------+---------+---------+-------+------+-------+
| id | select_type | table   | type  | possible_keys | key     | key_len | ref   | rows | Extra |
+----+-------------+---------+-------+---------------+---------+---------+-------+------+-------+
|  1 | SIMPLE      | vendeur | const | PRIMARY       | PRIMARY | 4       | const |    1 |       |
+----+-------------+---------+-------+---------------+---------+---------+-------+------+-------+
1 row in set (0.00 sec)

11:21 root$zend>
```

Influencer l'optimiseur

- Forcer MySQL à utiliser ou à ignorer les index (si possible)
- SELECT ... FROM table_name [[USE|FORCE|IGNORE] INDEX(nom_colonne_indexée)]
  - USE: essaye d'utiliser l'index
  - FORCE: force l'utilisation de l'index
  - IGNORE: ignore l'index
- SELECT STRAIGHT_JOIN
  - Lors d'une jointure, force l'optimiseur à joindre les tables dans l'ordre indiqué

Influencer l'optimiseur

- SELECT SQL_BUFFER_RESULT
  - Le résultat est stocké dans une table temporaire (pour des requêtes couteuses en temps)
- SELECT SQL_BIG_RESULT
  - Indique à MySQL que la taille du résultat sera importante (s'utilise avec GROUP BY, DISTINCT)

Les requêtes préparées

- Valable seulement pour la session MySQL en cours
- API natives disponibles pour PHP (mysqli_stmt_prepare,...)
  - PREPARE stmt_client FROM 'SELECT nom, prenom FROM client WHERE id_client = ? AND ville = ?';
  - SET @parm1 = 123, @param2 = 'Paris';
  - EXECUTE stmt_client USING @parm1, @param2;

Quelques conseils

- Les règles d'optimisation des index du SELECT s'appliquent aussi aux requêtes d'écritures(clause WHERE
- Utiliser l'INSERT multi-lignes pour écrire plusieurs enregistrements à la fois
  - INSERT INTO table (champ)VALUES (valeur1),(valeur2),(valeur3);
- Import massif de données texte avec la commande LOAD DATA INFILE
  - plus rapide que des INSERT équivalents

Quelques conseils

- En général les requêtes ont toutes une clause WHERE (réduit le nombre d'enregistrements)
- Utiliser la clause LIMIT
- Evitez la commande SELECT *
- La clé primaire (PRIMARY KEY) doit être la plus petite possible
  - Utiliser auto_increment
- Utiliser les requêtes préparées

Optimisation du schéma

- Choisir les bons types de champs (PROCEDURE ANALYSE())
- Choisir la bonne longueur de champs
- Attention particulière aux clés étrangères
- Normalisez votre base de données (3ème forme normale)
- Choisir les moteurs de stockages adaptés

PROCEDURE ANALYSE()

- Procédure d'analyse de la structure de la table
- Propose le type optimale de chaque champ analysé dans le SELECT
  - nom
  - valeur minimale
  - valeur maximlale
  - taille maximlale
  - nombre de valeur vides ou égales à zéro
  - nombre de valeur NULL
  - valeur ou taille moyenne
  - écart type
  - type optimal suggéré pour le champ analysé

EXPLAIN: optimiser sa requête

Zend The php Company

D:\MySQL\MySQL50\bin\mysql.exe

```
17:11 root$world> select continent, capital, population from country procedure analyse()\G
*************************** 1. row ***************************
             Field_name: world.country.Continent
              Min_value: Africa
              Max_value: South America
             Min_length: 4
             Max_length: 13
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 7.2678
                    Std: NULL
      Optimal_fieldtype: ENUM('Africa','Antarctica','Asia','Europe','North America','Oceania','South America') NOT NULL
*************************** 2. row ***************************
             Field_name: world.country.Capital
              Min_value: 1
              Max_value: 4074
             Min_length: 1
             Max_length: 4
       Empties_or_zeros: 0
                  Nulls: 7
Avg_value_or_avg_length: 2071.3060
                    Std: 1181.5409
      Optimal_fieldtype: SMALLINT(4) UNSIGNED
*************************** 3. row ***************************
             Field_name: world.country.Population
              Min_value: 50
              Max_value: 1277558000
             Min_length: 2
             Max_length: 10
       Empties_or_zeros: 7
                  Nulls: 0
Avg_value_or_avg_length: 25434098.1172
                    Std: 112031499.9648
      Optimal_fieldtype: INT(10) UNSIGNED NOT NULL
3 rows in set (0.00 sec)
```

Longueur des champs

- Bien dimensionner sa base
- Pas de VARCHAR(255) quand VARCHAR(40) suffit
- Pas de BIGINT quand INT suffit
- UNSIGNED pour un champ positif (auto_increment)
- Pensez aux type ENUM & SET
- Évitez l'utilisation de valeurs NULL (surtout pour les index)

Normalisation

- Organiser ses données pour
  - limiter les redondances
  - diviser une table en plusieurs et les relier entre elles
- afin d'éviter
  - les anomalies de mise à jour
  - les redondances

Moteur de stockage

- Pluggable storage engine architecture
- Plusieurs moteurs de stockage peuvent être utilisés sur le même serveur
- Le moteur de stockage est défini au niveau des tables
- Possibilité de changer de moteur
  - ALTER TABLE ma_table ENGINE = MyISAM
  - ALTER TABLE ma_table ENGINE = InnoDB
- Chaque moteurs optimisés pour un besoin

Moteur de stockage

- Caractéristiques indépendentes du moteur de stockage
  - les requêtes sql
  - le plan d'exécution
  - la sécurité
  - les fonctions
  - les logs
  - ...

Quel moteur de stockage ?

- Dépend des besoins suivant les critères:
  - de vitesse
  - de transactions
  - fréquence des écritures
  - volume de données a stocker
- Choisir moteur de stockage adapté est important

MyISAM

- Très rapide pour les applications exigeantes en lecture
- Prend en charge les index FULLTEXT
- Prend en charge les types de données spatiales (GIS)
- Adapté pour le reporting ou le data warehouse
- Cadre d'utilisation:
  - lectures fréquentes
  - mises à jour avec des index

InnoDB

- Moteur de stockage transactionnel (norme ACID)
- Prend en charge les clés étrangères et l'intégrité référentielle
- Cadre d'utilisation:
  - besoin transactionnel
  - besoin intégrité référentielle
  - tables avec un fréquence élevée de commandes UPDATE et DELETE

MEMORY

- Stocke les données uniquement en mémoire
- Très rapide en lecture et en écriture
- Non transactionnel
- Les données sont perdues quand le serveur s'arrete
- La structure de la table est persistante
- Pas de prise en charge de BLOB ni de TEXT
- Cadre d'utilisation:
  - tables temporaires

ARCHIVE

- Stocke de grandes quantités de données en les compressant
- Ne permet que les requêtes INSERT et SELECT
- Ne permet pas:
  - la clause ORDER BY
  - les colonnes BLOB
- Cadre d'utilisation:
  - stocker de grandes quantitées de données (archives, logs)

Zend
The *php* Company

Services Web

Principe des Services Web

- Les services web répondent à un besoin d'intéropérabilité.
- Ils permettent l'échange d'informations et l'accès à des services dynamiques entre plusieurs systèmes hétérogènes.
- Ils définissent un protocole commun et standard pour cela.

- REST signifie Representational State Transfer.
- L'objectif de REST est de permettre la mise en place de services légers et faciles à utiliser.
- REST n'est pas un standard, mais une méthode.
- Il existe de plus en plus de services REST disponibles (yahoo, google, amazon, etc.). C'est une solution très utilisée en B2C.
- REST ne définit pas de protocole ni de format de données, une documentation développeur est souvent nécessaire pour faire un client REST.

Principe de REST

- L'interrogation d'un service REST se fait via l'URL (requête GET).
- Le format du retour d'information peut prendre plusieurs formes possibles :
  - Un flux XML
  - Une chaîne sérialisée

REST : exemple de comportement

- Appel du serveur REST

```
Appel du serveur à vide : http://localhost/server.php

<response type="error">
  <message>Requête incorrecte.</message>
</response>

Appel de l'opération 'add' pour ajouter des éléments :
http://localhost/server.php?operation=add&name=Guillaume&tel=4678
http://localhost/server.php?operation=add&name=Steven&tel=47859

<response type="info">
  <operation>add</operation>
  <message>Enregistrement ajouté.</message>
</response>
```

REST : exemple de comportement

- Appel du serveur REST

```
Appel de l'opération 'list' :
http://localhost/server.php?operation=list

<response type="info">
  <operation>list</operation>
  <content>
    <item><name>Guillaume</name><tel>4678</tel></item>
    <item><name>David</name><tel>47859</tel></item>
  </content>
</response>
```

Serveur REST (1/2)

- Créez un fichier 13_rest_server.php
- Dans ce fichier
  - Créez une classe 'MathServer' et une méthode 'getDouble($number)' qui renvoit le double d'un nombre.
  - L'utilisateur doit appeler votre serveur avec un paramètre (GET) 'operation' qui vaut 'getDouble' et un paramètre 'number' qui contient un nombre
  - Le serveur affiche un flux XML comme décrit dans les slides précédents. Une réponse de type 'erreur' doit être renvoyée si les paramètres ne sont pas corrects

Libre à vous d'utiliser ou non une extension tel que SimpleXML pour générer votre flux XML.

Client REST (2/2)

- Créez un fichier 13_rest_client.php
- Dans ce fichier
  - Créez une classe 'MathManager' avec une méthode '__call'
  - Le paramètre 'operation' est le nom de la méthode demandée et le paramètre 'number' le premier paramètre passé à la méthode
  - '__call' fait appel à votre serveur rest, renvoit le résultat si tout s'est bien passé et lance une exception dans le cas contraire
  - Exemple d'utilisation : echo $mathManager->getDouble(3);

Pour le développeur, l'utilisation de votre classe doit être transparente. Il n'y a pas besoin de savoir qu'il y a un service derrière pour pouvoir s'en servir.

REST par la sérialisation

- Il est possible de faire la même chose en envoyant des données sérialisées plutôt que du xml.
  - Avantage : les données sérialisées sont facile à manipuler avec un client PHP.
  - Inconvénient : seul un client PHP peut lire les données sérialisées, ce qui réduit la couverture des clients potentiels.
- Les services Yahoo ! proposent des services REST sérialisés pour PHP.

- XMLRPC est un protocole d'échange encore utilisé.
- Il permet l'échange de données et la manipulation de fonctions à distance.
- Ces services sont souvent remplacés par les standards SOAP / WSDL

SOAP & WSDL

- SOAP et WSDL sont les standards les plus utilisés pour mettre en place des services web.
- SOAP
  - Est le protocole d'échange de données : il assure la mise en place d'un flux à travers lequel les données et leurs structures sont entièrement conservés.
- WSDL
  - Est le protocole de description du service. Il s'agit d'un fichier XML "mode d'emploi" du service SOAP.

SOAP et PHP

- PHP permet de créer un serveur SOAP
- PHP permet de créer un client SOAP
- PHP ne permet pas (encore) de générer un document WSDL.
- L'extension SOAP pour PHP 5 est native, stable et efficace.

SOAP : implémentation du serveur

- Un serveur qui exporte la fonction double

```
class mySoapService {
    /**
     * Obtenir la valeur doublée
     *
     * @param integer $num
     * @return integer
     */
    function double($num) {
        return $num * 2;
    }
}

$soap = new SoapServer(dirname(__FILE__).'/soap_server.wsdl');
$soap->setClass('mySoapService');
$soap->handle();
```

Générer le fichier WSDL

- Avec Zend Studio
  - Aller dans outils -> générateur WSDL
  - Sélectionner l'emplacement du fichier WSDL
  - Sélectionner les classes qu'il faut intégrer au WSDL
  - Valider
- Avec UML2PHP5
  - http://uml2php5.zpmag.com/SOAP_server.php

Le client SOAP associé

- Implémentation du client SOAP

```
$client = new SoapClient(dirname(__FILE__) . '/soap_server.wsdl');
echo $client->double(4);
```

Serveur SOAP (1/2)

- Créez un fichier 14_soap_server.php
- Dans ce fichier
  - Créez une classe 'MathServer' et une méthode 'getDouble($number)' qui renvoit le double d'un nombre.
  - N'oubliez pas de mentionner le type des paramètres et valeurs de retours dans la phpdoc !
  - Générez le fichier WSDL de MathServer à l'aide de Zend Studio
  - Le serveur propose l'utilisation de MathServer à travers un serveur SOAP

Avec PHP5, utilisez l'extension native SOAP qui est de loin la plus efficace.

Client SOAP (2/2)

- Créez un fichier 14_soap_client.php
- Dans ce fichier
  - Créez une classe 'MathManager' avec une méthode '__call'
  - Le paramètre 'operation' est le nom de la méthode demandée et le paramètre 'number' le premier paramètre passé à la méthode
  - '__call' fait appel à votre serveur soap, renvoit le résultat si tout s'est bien passé et lance une exception dans le cas contraire
  - Exemple d'utilisation : echo $mathManager->getDouble(3);

Pour le développeur, l'utilisation de votre classe doit être transparente. Il n'y a pas besoin de savoir qu'il y a un service derrière pour pouvoir s'en servir.

Création d'un service SOAP (+)

- Créez un fichier 15_soap_authentification.php
- Créer un service qui 'exporte' une fonction 'authentificate'.
- Cette fonction 'authentificate' prend en paramètre un login et un mot de passe.
  - Mettez en dur dans la fonction 'authentificate' le login / mot de passe valide.
  - Cette fonction est similaire à la fonction 'double' de l'exemple précédent.
- Implémentez le serveur et un client.

Aidez-vous de l'implémentation décrite dans les slides pour mettre en place votre service.

Exemple de client Google

- Service de recherche avec Google

```
$words = $_GET['query'] ? $_GET['query'] : "afup";
array('namespace' => 'urn:GoogleSearch', 'trace' => 0);
try {
    $client = new SoapClient(dirname(__FILE__).
              "/GoogleSearch.wsdl", array(
'namespace' => 'urn:GoogleSearch'));
    $key = '<votre clé ici !>';
    $result = $client->doGoogleSearch($key, $words, 0, 10, false, '',
false, 'lang_fr', '', '');
    var_dump($result);
} catch (Exception $e) {
    echo $e->getMessage();
}
```

Zend
The *php* Company

Introduction à la SPL

Qu'est-ce que la SPL ?

- La SPL est une extension contenant un ensemble de classes implémentées nativement dans PHP.
- Ces classes sont utilisées par :
  - Les ressources natives de PHP
  - Le développeur PHP, qui peut s'en servir pour ses propres besoins.
- La SPL est documentée :
  - http://www.php.net/~helly/php/ext/spl/
  - http://www.php.net/spl

Concepts / utilité

- Les classes de la SPL permettent d'attribuer des fonctionnalités complémentaires aux classes que l'on manipule.
- Elles permettent aussi d'obtenir un squelette pour un pattern.
- Elles gèrent de nombreuses fonctionnalités telles que par exemple :
  - La gestion des exceptions
  - Les collections (facultés d'itération)

Hiérarchie des classes

- Les classes et interfaces de la SPL :
  - http://www.php.net/~helly/php/ext/spl/hierarchy.html
- Quelques classes - racine
  - ArrayAccess
  - Coutable
  - Exception
  - SplFileInfo, SplObserver, SplSubject
  - Traversable

Classes SPL Disponibles

- Pour liste les classes SPL disponibles...

```
print_r(spl_classes());

Résultat :
Array
(
    [AppendIterator] => AppendIterator
    [ArrayIterator] => ArrayIterator
    (...)
    [UnderflowException] => UnderflowException
    [UnexpectedValueException] => UnexpectedValueException
)
```

Les itérateurs (Traversable)

- Un itérateur permet à un objet de se comporter comme une collection, c'est à dire d'être parcouru.
- Le parcours d'un objet qui implémente un itérateur peut être géré par l'instruction "foreach".
- Il existe de nombreuses interfaces d'itération :
  - Dba, Directory, Append (simplexml), Caching, Filter, Recursive, ArrayObject, etc.

Parcours de répertoire

- Sans itérateur...

```
if ($handle = opendir(".")) {
   while (false !== ($file = readdir($handle))) {
       if ($file != "." && $file != "..") {
           print "$file<br />\n";
       }
   }
   closedir($handle);
}
```

Parcours de répertoire

- Avec itérateur...

```
$dir = new DirectoryIterator(".");
while($dir->valid()) {
   if(!$dir->isDot()) {
       print $dir->current() . "<br />\n";
   }
   $dir->next();
}
```

Iterateur pour tableaux

- Manipuler un tableau avec arrayObject

```
$array = array('1' => 'un',
               '2' => 'deux',
               '3' => 'trois');

$arrayobject = new ArrayObject($array);
$iterator = $arrayobject->getIterator();
while($iterator->valid()) {
   echo $iterator->key() . ' => ' .
        $iterator->current() . "\n";
   $iterator->next();
}
```

Implémenter un itérateur

- Vos propres classes peuvent implémenter un itérateur afin d'être parcourues via foreach ou while !
- Quelques exemples d'utilisation :
  - Parcourir le résultat d'une requête SQL
  - Orienter le parcours d'un tableau ou simuler un parcours récursif à travers un parcours simple.

Itérateur : principe

- Un intérateur possède des méthodes spéciales permettant son parcours :
  - rewind : élément précédent
  - current : élément courant
  - key : clé de l'élément courant
  - next : élément suivant
  - valid : vérifie s'il y a d'autres éléments
- Le principe du parcours est le même que celui du tableau.

Optimisation BD avec SPL (1/2) (+)

- A l'aide de phpMyAdmin créez une base 'magasin'
- Dans cette base, créez une table 'produits' avec 5 champs :
  - 'id' de type 'INT', clé primaire, auto-incrémenté
  - 'nom' de type VARCHAR(250)
  - 'description' de type TEXT
  - 'prix' de type DECIMAL
  - 'image' de type BLOB
- Ajoutez quelques enregistrements à cette table

Vous pouvez aussi utiliser d'autres outils (Mysql Query Browser, SQL Front, ...).

Optimisation BD avec SPL (2/2) (+)

- Créez un fichier 16_spl_speed_display.php
- Dans ce fichier
  - Créez une classe 'Product_Iterator' qui implémente l'interface native 'Iterator'
  - Le constructeur se connecte à la base avec PDO et effectue une requête SELECT
  - Implémentez les méthodes 'next', 'preview', 'current', ... demandées par l'interface 'Iterator'
  - Créez un objet avec cette classe, que vous utiliserez pour afficher les résultats avec 'foreach'

Pour cet exercice, le formateur doit vous guider. Il s'agit d'une optimisation élégante bien que pas nécessaire pour vos développements.

Zend
The *php* Company

Merci de votre attention !

Avez-vous des questions ?

- D'ordre général sur notre formation ?
- Sur un point précis qui pose problème ?
- Sur un point précis à approfondir ?
- Sur un tout autre sujet ?

Utilisez ce slide pour prendre des notes (1/3)

Utilisez ce slide pour prendre des notes (2/3)

Utilisez ce slide pour prendre des notes (3/3)