Programmation orientée objet et interfaces web en PHP

La programmation objet avec PHP5
Bases de données et interfaces web
Fonctionnement des sessions

Nicolas Moyroud

Cemagref - UMR TETIS

26 Juin 2008

Plan

1 Programmation orientée objet avec PHP5

2 Interaction avec les bases de données

3 Développement d'interfaces web pour les bases de données

4 Principes de fonctionnement des sessions

Principes généraux de la POO : rappels

- La programmation orientée objet (POO) est un style de programmation qui consiste à définir et assembler des briques de code appelées "objets", il s'oppose au style de programmation classique dit "procédural"
- L'intérêt est de favoriser l'évoluabilité du code en concevant une application non plus à partir de ses fonctionnalités, mais à partir de ses données qui sont généralement plus stables
- Un objet peut représenter un concept (fichier) ou une entité du monde physique (voiture, personne)
- Un objet est une structure qui regroupe des données (attributs) et les moyens de traiter ces données (des fonctions que l'on appelle "méthodes" en POO)

http://fr.wikipedia.org/wiki/Poo
http://hdd34.developpez.com/cours/artpoo/

http://www.larcher.com/eric/guides/javactivex/V.htm

Principes généraux de la POO : rappels

- La programmation orientée objet (POO) est un style de programmation qui consiste à définir et assembler des briques de code appelées "objets", il s'oppose au style de programmation classique dit "procédural"
- L'intérêt est de favoriser l'évoluabilité du code en concevant une application non plus à partir de ses fonctionnalités, mais à partir de ses données qui sont généralement plus stables
- Un objet peut représenter un concept (fichier) ou une entité du monde physique (voiture, personne)
- Un objet est une structure qui regroupe des données (attributs) et les moyens de traiter ces données (des fonctions que l'on appelle "méthodes" en POO)

http://fr.wikipedia.org/wiki/Poo
http://hdd34.developpez.com/cours/artpoo/

http://www.larcher.com/eric/guides/javactivex/V.htm

Principes généraux de la POO : rappels

- La programmation orientée objet (POO) est un style de programmation qui consiste à définir et assembler des briques de code appelées "objets", il s'oppose au style de programmation classique dit "procédural"
- L'intérêt est de favoriser l'évoluabilité du code en concevant une application non plus à partir de ses fonctionnalités, mais à partir de ses données qui sont généralement plus stables
- Un objet peut représenter un concept (fichier) ou une entité du monde physique (voiture, personne)
- Un objet est une structure qui regroupe des données (attributs) et les moyens de traiter ces données (des fonctions que l'on appelle "méthodes" en POO)

http://fr.wikipedia.org/wiki/Poo
http://hdd34.developpez.com/cours/artpoo/

http://www.larcher.com/eric/guides/javactivex/V.htm

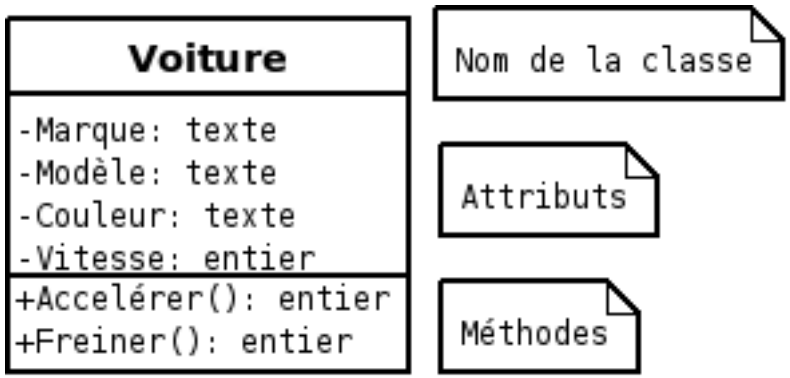
Principes généraux de la POO : rappels

- La programmation orientée objet (POO) est un style de programmation qui consiste à définir et assembler des briques de code appelées "objets", il s'oppose au style de programmation classique dit "procédural"
- L'intérêt est de favoriser l'évoluabilité du code en concevant une application non plus à partir de ses fonctionnalités, mais à partir de ses données qui sont généralement plus stables
- Un objet peut représenter un concept (fichier) ou une entité du monde physique (voiture, personne)
- Un objet est une structure qui regroupe des données (attributs) et les moyens de traiter ces données (des fonctions que l'on appelle "méthodes" en POO)

http://fr.wikipedia.org/wiki/Poo
http://hdd34.developpez.com/cours/artpoo/

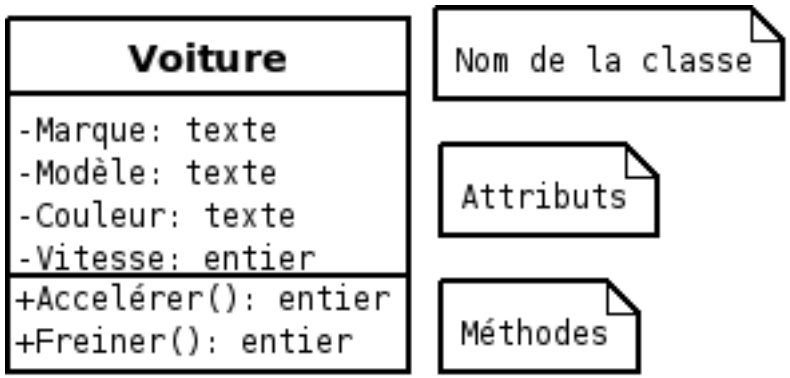
http://www.larcher.com/eric/guides/javactivex/V.htm

Principes généraux de la POO : concept de classe



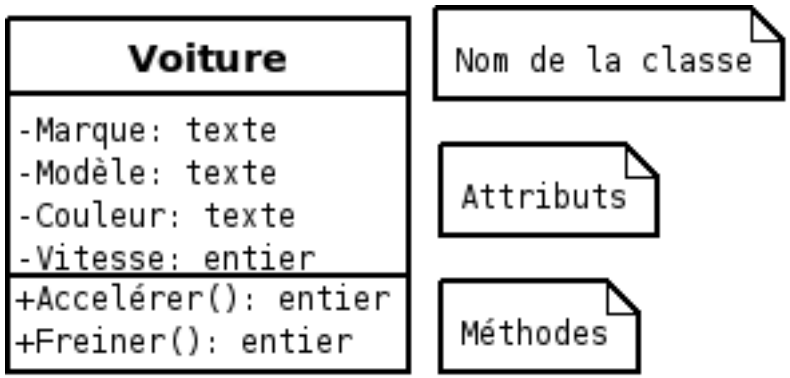
- En POO, l'élément de base est la classe qui est une représentation abstraite d'un objet, c'est une sorte de "moule à objets"
- Les objets concrets sont tous les instances d'une classe
- Une instance de la classe voiture serait par exemple un objet de marque "Renault", modèle "Clio", couleur "rouge" et vitesse "0"
- Les différents objets sont en interaction entre eux : on met en place un diagramme de classes qui traduit ces interactions

Principes généraux de la POO : concept de classe



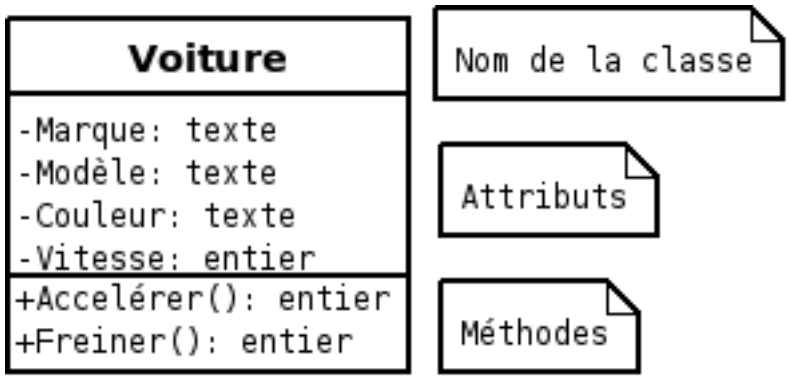
- En POO, l'élément de base est la classe qui est une représentation abstraite d'un objet, c'est une sorte de "moule à objets"
- Les objets concrets sont tous les instances d'une classe
- Une instance de la classe voiture serait par exemple un objet de marque "Renault", modèle "Clio", couleur "rouge" et vitesse "0"
- Les différents objets sont en interaction entre eux : on met en place un diagramme de classes qui traduit ces interactions

Principes généraux de la POO : concept de classe



- En POO, l'élément de base est la classe qui est une représentation abstraite d'un objet, c'est une sorte de "moule à objets"
- Les objets concrets sont tous les instances d'une classe
- Une instance de la classe voiture serait par exemple un objet de marque "Renault", modèle "Clio", couleur "rouge" et vitesse "0"
- Les différents objets sont en interaction entre eux : on met en place un diagramme de classes qui traduit ces interactions

Principes généraux de la POO : concept de classe



- En POO, l'élément de base est la classe qui est une représentation abstraite d'un objet, c'est une sorte de "moule à objets"
- Les objets concrets sont tous les instances d'une classe
- Une instance de la classe voiture serait par exemple un objet de marque "Renault", modèle "Clio", couleur "rouge" et vitesse "0"
- Les différents objets sont en interaction entre eux : on met en place un diagramme de classes qui traduit ces interactions

Principes généraux de la POO : l'encapsulation

- Les attributs d'un objet qui constituent sa structure interne ne sont en général pas accessibles aux autres objets, c'est le principe de l'encapsulation
- Par exemple, pour pouvoir définir la couleur d'une voiture, il faudra lui ajouter une méthode changerCouleur qui s'occupera de changer la valeur de son attribut couleur
- Les autres objets n'ont ainsi plus besoin de savoir comment changer la couleur de la voiture, ils se contentent d'appeler la méthode changerCouleur
- On garde ainsi une cohérence dans la gestion de l'objet et on assure l'intégrité de ses données

Principes généraux de la POO : l'encapsulation

- Les attributs d'un objet qui constituent sa structure interne ne sont en général pas accessibles aux autres objets, c'est le principe de l'encapsulation
- Par exemple, pour pouvoir définir la couleur d'une voiture, il faudra lui ajouter une méthode changerCouleur qui s'occupera de changer la valeur de son attribut couleur
- Les autres objets n'ont ainsi plus besoin de savoir comment changer la couleur de la voiture, ils se contentent d'appeler la méthode changerCouleur
- On garde ainsi une cohérence dans la gestion de l'objet et on assure l'intégrité de ses données

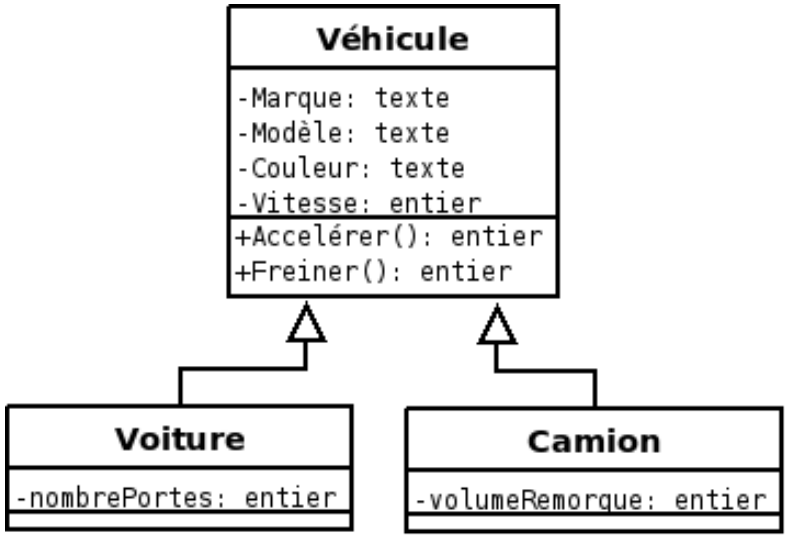
Principes généraux de la POO : l'encapsulation

- Les attributs d'un objet qui constituent sa structure interne ne sont en général pas accessibles aux autres objets, c'est le principe de l'encapsulation
- Par exemple, pour pouvoir définir la couleur d'une voiture, il faudra lui ajouter une méthode changerCouleur qui s'occupera de changer la valeur de son attribut couleur
- Les autres objets n'ont ainsi plus besoin de savoir comment changer la couleur de la voiture, ils se contentent d'appeler la méthode changerCouleur
- On garde ainsi une cohérence dans la gestion de l'objet et on assure l'intégrité de ses données

Principes généraux de la POO : l'encapsulation

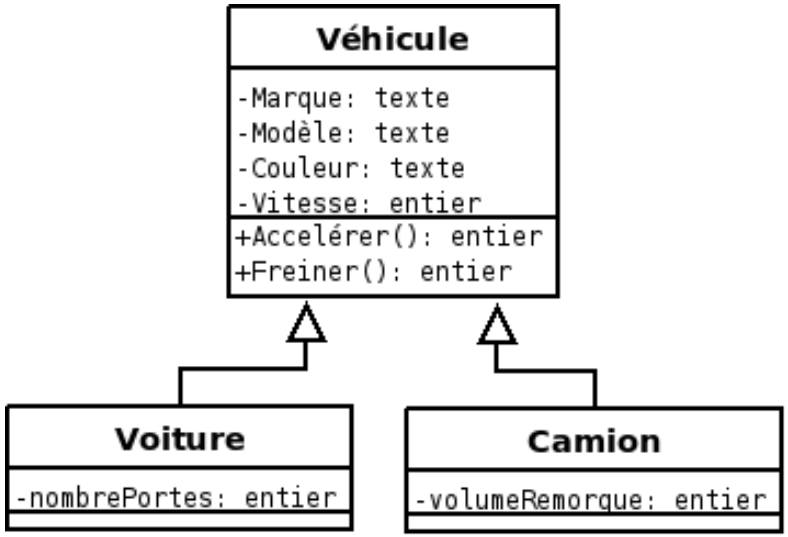
- Les attributs d'un objet qui constituent sa structure interne ne sont en général pas accessibles aux autres objets, c'est le principe de l'encapsulation
- Par exemple, pour pouvoir définir la couleur d'une voiture, il faudra lui ajouter une méthode changerCouleur qui s'occupera de changer la valeur de son attribut couleur
- Les autres objets n'ont ainsi plus besoin de savoir comment changer la couleur de la voiture, ils se contentent d'appeler la méthode changerCouleur
- On garde ainsi une cohérence dans la gestion de l'objet et on assure l'intégrité de ses données

Principes généraux de la POO : l'héritage



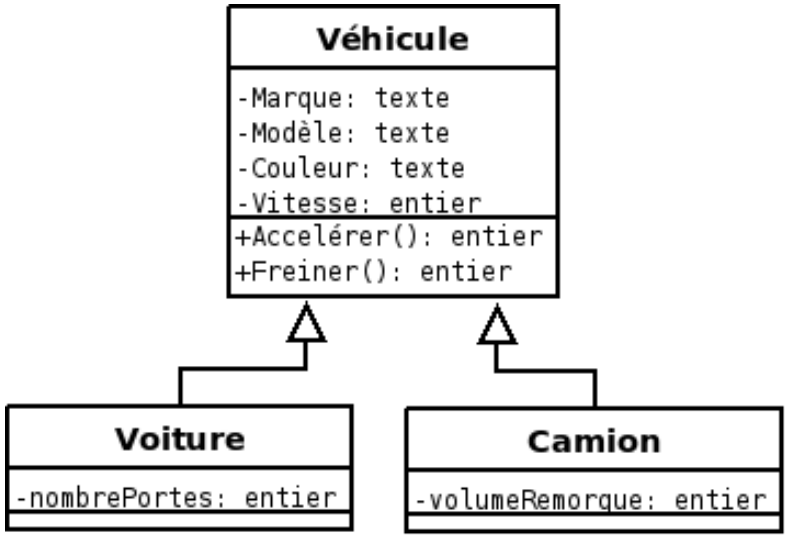
- Le principe de l'héritage est basé sur des classes filles qui héritent des caractéristiques (attributs et méthodes) d'une classe mère
- Une classe fille peut également définir ses propres caractéristiques
- Par exemple, on peut définir une classe "Véhicule" dont hérite deux classes filles "Voiture" et "Camion"
- Ce principe permet la réutilisabilité et l'adaptabilité des objets

Principes généraux de la POO : l'héritage



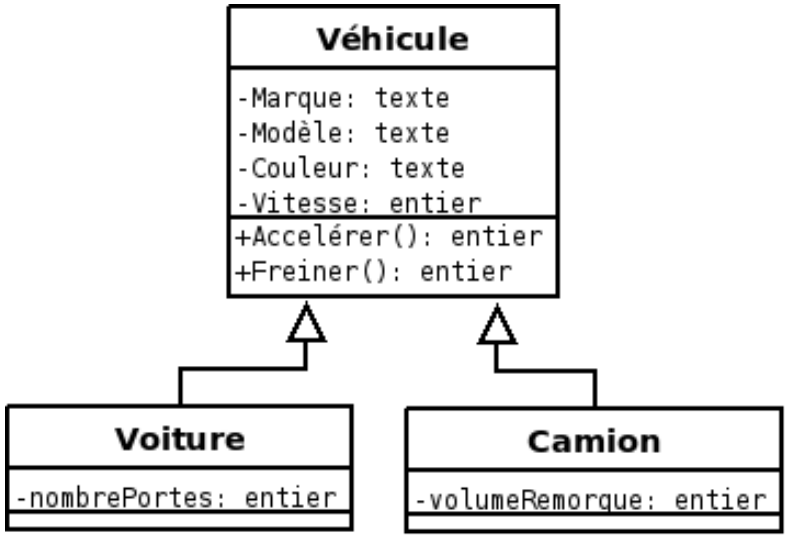
- Le principe de l'héritage est basé sur des classes filles qui héritent des caractéristiques (attributs et méthodes) d'une classe mère
- Une classe fille peut également définir ses propres caractéristiques
- Par exemple, on peut définir une classe "Véhicule" dont hérite deux classes filles "Voiture" et "Camion"
- Ce principe permet la réutilisabilité et l'adaptabilité des objets

Principes généraux de la POO : l'héritage



- Le principe de l'héritage est basé sur des classes filles qui héritent des caractéristiques (attributs et méthodes) d'une classe mère
- Une classe fille peut également définir ses propres caractéristiques
- Par exemple, on peut définir une classe "Véhicule" dont hérite deux classes filles "Voiture" et "Camion"
- Ce principe permet la réutilisabilité et l'adaptabilité des objets

Principes généraux de la POO : l'héritage



- Le principe de l'héritage est basé sur des classes filles qui héritent des caractéristiques (attributs et méthodes) d'une classe mère
- Une classe fille peut également définir ses propres caractéristiques
- Par exemple, on peut définir une classe "Véhicule" dont hérite deux classes filles "Voiture" et "Camion"
- Ce principe permet la réutilisabilité et l'adaptabilité des objets

La POO avec PHP5

- Contrairement des langages comme Java ou C++, les premières versions du langage PHP n'ont pas été conçues pour la POO
- Les premiers éléments de POO ont été intégrés dans la version 3
- PHP4 puis surtout PHP5 ont introduit de véritables concepts pour la POO, notamment la notion d'héritage et d'encapsulation
- Il est maintenant possible de programmer très efficacement en POO avec PHP
- La grande majorité des librairies de code sont disponibles sous forme de classes

`http://g-rossolini.developpez.com/tutoriels/php/cours/?page=poo`

La POO avec PHP5

- Contrairement des langages comme Java ou C++, les premières versions du langage PHP n'ont pas été conçues pour la POO
- Les premiers éléments de POO ont été intégrés dans la version 3
- PHP4 puis surtout PHP5 ont introduit de véritables concepts pour la POO, notamment la notion d'héritage et d'encapsulation
- Il est maintenant possible de programmer très efficacement en POO avec PHP
- La grande majorité des librairies de code sont disponibles sous forme de classes

`http://g-rossolini.developpez.com/tutoriels/php/cours/?page=poo`

La POO avec PHP5

- Contrairement des langages comme Java ou C++, les premières versions du langage PHP n'ont pas été conçues pour la POO
- Les premiers éléments de POO ont été intégrés dans la version 3
- PHP4 puis surtout PHP5 ont introduit de véritables concepts pour la POO, notamment la notion d'héritage et d'encapsulation
- Il est maintenant possible de programmer très efficacement en POO avec PHP
- La grande majorité des librairies de code sont disponibles sous forme de classes

`http://g-rossolini.developpez.com/tutoriels/php/cours/?page=poo`

La POO avec PHP5

- Contrairement des langages comme Java ou C++, les premières versions du langage PHP n'ont pas été conçues pour la POO
- Les premiers éléments de POO ont été intégrés dans la version 3
- PHP4 puis surtout PHP5 ont introduit de véritables concepts pour la POO, notamment la notion d'héritage et d'encapsulation
- Il est maintenant possible de programmer très efficacement en POO avec PHP
- La grande majorité des librairies de code sont disponibles sous forme de classes

`http://g-rossolini.developpez.com/tutoriels/php/cours/?page=poo`

La POO avec PHP5

- Contrairement des langages comme Java ou C++, les premières versions du langage PHP n'ont pas été conçues pour la POO
- Les premiers éléments de POO ont été intégrés dans la version 3
- PHP4 puis surtout PHP5 ont introduit de véritables concepts pour la POO, notamment la notion d'héritage et d'encapsulation
- Il est maintenant possible de programmer très efficacement en POO avec PHP
- La grande majorité des librairies de code sont disponibles sous forme de classes

`http://g-rossolini.developpez.com/tutoriels/php/cours/?page=poo`

La POO avec PHP5

Exemple de classe

```
class Vehicule {
    private $marque;
    private $vitesse;
    public function __construct($marque) {
        $this->marque = $marque;
        $this->vitesse = 0; //un nouveau vehicule a une vitesse nulle
    }
    public function accelerer($vit) {
        $this->vitesse += $vit; //ok dans la classe
    }
    public function afficherVitesse() {
        return $this->vitesse;
    }
}
```

- Mots réservés : class, function, private, public, protected, __construct, __destruct, extends, implements, self, parent, $this, ->

La POO avec PHP5

Héritage de classe

```
class Voiture extends Vehicule {
    private $nombrePortes;
    public function __construct($marque,$nbPortes) {
        parent::__construct($marque);
        $this->nombrePortes = $nbPortes;
    }
    public function afficherNbPortes() {
        return $this->nombrePortes;
    }
}
```

Utilisation d'une classe

```
$ma_voiture = new Voiture($marque,5);
$ma_voiture->accelerer(20);
$ma_voiture->accelerer(5);
echo 'nbPortes = '.$ma_voiture->afficherNbPortes();
echo 'vitesse = '.$ma_voiture->afficherVitesse();
```

Plan

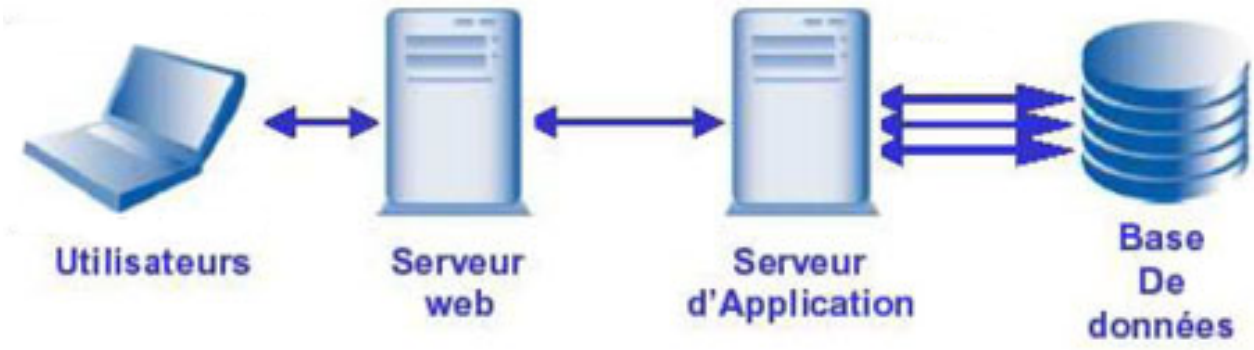
1 Programmation orientée objet avec PHP5

2 Interaction avec les bases de données

3 Développement d'interfaces web pour les bases de données

4 Principes de fonctionnement des sessions

PHP et les bases de données

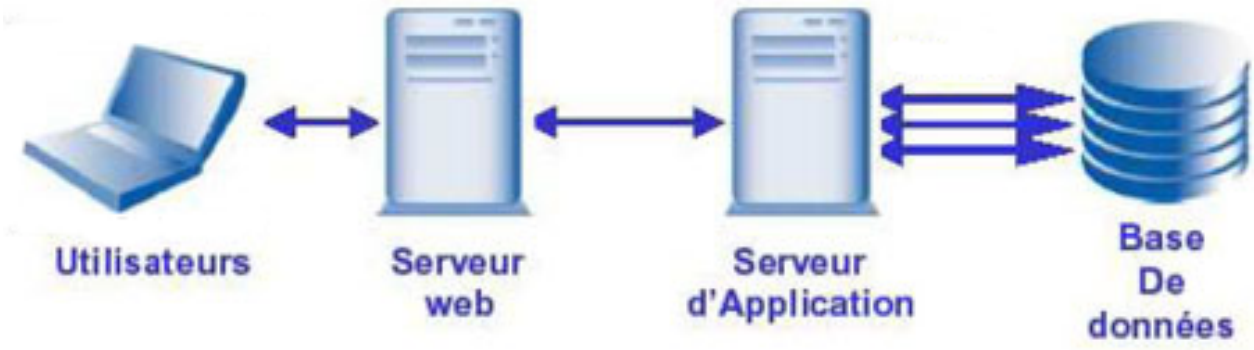


- Le langage PHP supporte l'accès à un grand nombre de systèmes de gestion de bases de données : Oracle, PostgreSQL, MySQL, ...
- Permet de développer des applications basées sur l'architecture client + serveur web + serveur de données
- Nativement, PHP propose des fonctions spécifiques à chaque SGBD
- Il existe des librairies pour l'abstraction de bases de données qui permettent d'utiliser un code PHP identique quel que soit le SGBD (PEAR_MDB2, PHP Data Objects)

PDO : `http://fr.php.net/pdo`

PEAR_MDB2 : `http://pear.php.net/package/MDB2`

PHP et les bases de données

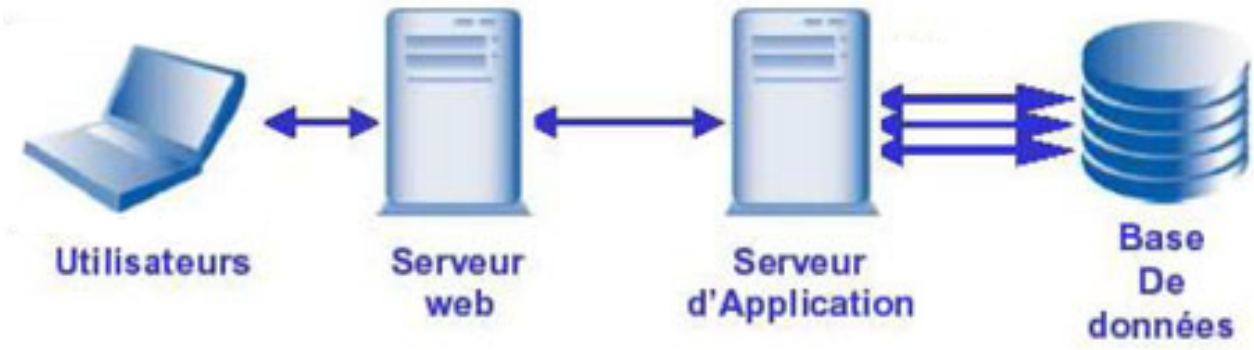


- Le langage PHP supporte l'accès à un grand nombre de systèmes de gestion de bases de données : Oracle, PostgreSQL, MySQL, ...
- Permet de développer des applications basées sur l'architecture client + serveur web + serveur de données
- Nativement, PHP propose des fonctions spécifiques à chaque SGBD
- Il existe des librairies pour l'abstraction de bases de données qui permettent d'utiliser un code PHP identique quel que soit le SGBD (PEAR_MDB2, PHP Data Objects)

PDO : `http://fr.php.net/pdo`

PEAR_MDB2 : `http://pear.php.net/package/MDB2`

PHP et les bases de données

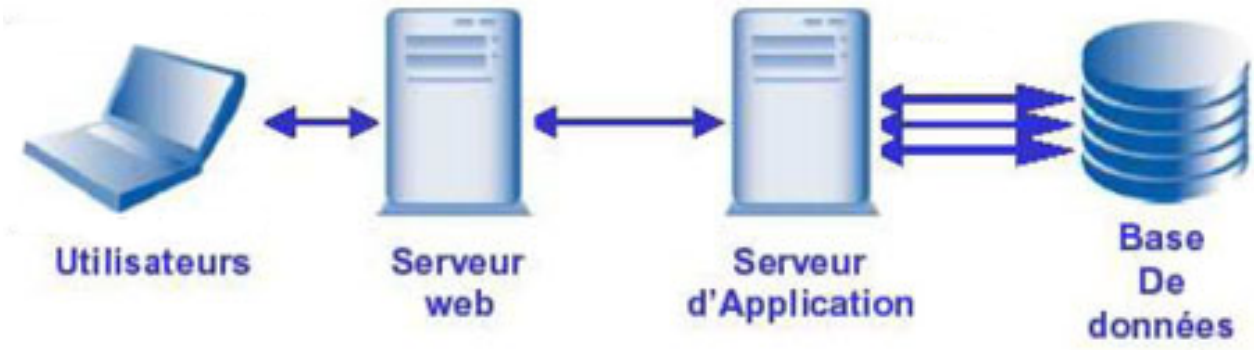


- Le langage PHP supporte l'accès à un grand nombre de systèmes de gestion de bases de données : Oracle, PostgreSQL, MySQL, ...
- Permet de développer des applications basées sur l'architecture client + serveur web + serveur de données
- Nativement, PHP propose des fonctions spécifiques à chaque SGBD
- Il existe des librairies pour l'abstraction de bases de données qui permettent d'utiliser un code PHP identique quel que soit le SGBD (PEAR_MDB2, PHP Data Objects)

PDO : `http://fr.php.net/pdo`

PEAR_MDB2 : `http://pear.php.net/package/MDB2`

PHP et les bases de données



- Le langage PHP supporte l'accès à un grand nombre de systèmes de gestion de bases de données : Oracle, PostgreSQL, MySQL, ...
- Permet de développer des applications basées sur l'architecture client + serveur web + serveur de données
- Nativement, PHP propose des fonctions spécifiques à chaque SGBD
- Il existe des librairies pour l'abstraction de bases de données qui permettent d'utiliser un code PHP identique quel que soit le SGBD (PEAR_MDB2, PHP Data Objects)

PDO : `http://fr.php.net/pdo`

PEAR_MDB2 : `http://pear.php.net/package/MDB2`

Les fonctions PHP spécifiques à PostgreSQL

- pg_connect → ouvre une connexion à une base PostgreSQL
- pg_query → exécute une requête SQL
- pg_query_params → exécute une requête SQL en passant des paramètres séparément du code SQL
- pg_fetch_row → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau numérique
- pg_fetch_assoc → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau associatif (les clés sont les noms des champs)
- pg_escape_string → protège une chaîne de caractères pour l'insérer dans un champ texte
- pg_field_name → retourne le nom d'un champ

Documentation : `http://fr.php.net/manual/fr/ref.pgsql.php`

Les fonctions PHP spécifiques à PostgreSQL

- pg_connect → ouvre une connexion à une base PostgreSQL
- pg_query → exécute une requête SQL
- pg_query_params → exécute une requête SQL en passant des paramètres séparément du code SQL
- pg_fetch_row → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau numérique
- pg_fetch_assoc → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau associatif (les clés sont les noms des champs)
- pg_escape_string → protège une chaîne de caractères pour l'insérer dans un champ texte
- pg_field_name → retourne le nom d'un champ

Documentation : `http://fr.php.net/manual/fr/ref.pgsql.php`

Les fonctions PHP spécifiques à PostgreSQL

- pg_connect → ouvre une connexion à une base PostgreSQL
- pg_query → exécute une requête SQL
- pg_query_params → exécute une requête SQL en passant des paramètres séparément du code SQL
- pg_fetch_row → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau numérique
- pg_fetch_assoc → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau associatif (les clés sont les noms des champs)
- pg_escape_string → protège une chaîne de caractères pour l'insérer dans un champ texte
- pg_field_name → retourne le nom d'un champ

Documentation : `http://fr.php.net/manual/fr/ref.pgsql.php`

Les fonctions PHP spécifiques à PostgreSQL

- pg_connect → ouvre une connexion à une base PostgreSQL
- pg_query → exécute une requête SQL
- pg_query_params → exécute une requête SQL en passant des paramètres séparément du code SQL
- pg_fetch_row → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau numérique
- pg_fetch_assoc → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau associatif (les clés sont les noms des champs)
- pg_escape_string → protège une chaîne de caractères pour l'insérer dans un champ texte
- pg_field_name → retourne le nom d'un champ

Documentation : `http://fr.php.net/manual/fr/ref.pgsql.php`

Les fonctions PHP spécifiques à PostgreSQL

- pg_connect → ouvre une connexion à une base PostgreSQL
- pg_query → exécute une requête SQL
- pg_query_params → exécute une requête SQL en passant des paramètres séparément du code SQL
- pg_fetch_row → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau numérique
- pg_fetch_assoc → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau associatif (les clés sont les noms des champs)
- pg_escape_string → protège une chaîne de caractères pour l'insérer dans un champ texte
- pg_field_name → retourne le nom d'un champ

Documentation : `http://fr.php.net/manual/fr/ref.pgsql.php`

Les fonctions PHP spécifiques à PostgreSQL

- pg_connect → ouvre une connexion à une base PostgreSQL
- pg_query → exécute une requête SQL
- pg_query_params → exécute une requête SQL en passant des paramètres séparément du code SQL
- pg_fetch_row → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau numérique
- pg_fetch_assoc → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau associatif (les clés sont les noms des champs)
- pg_escape_string → protège une chaîne de caractères pour l'insérer dans un champ texte
- pg_field_name → retourne le nom d'un champ

Documentation : `http://fr.php.net/manual/fr/ref.pgsql.php`

Les fonctions PHP spécifiques à PostgreSQL

- pg_connect → ouvre une connexion à une base PostgreSQL
- pg_query → exécute une requête SQL
- pg_query_params → exécute une requête SQL en passant des paramètres séparément du code SQL
- pg_fetch_row → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau numérique
- pg_fetch_assoc → lit un enregistrement depuis résultat d'une requête et stocke ses champs dans un tableau associatif (les clés sont les noms des champs)
- pg_escape_string → protège une chaîne de caractères pour l'insérer dans un champ texte
- pg_field_name → retourne le nom d'un champ

Documentation : `http://fr.php.net/manual/fr/ref.pgsql.php`

Exemple d'interaction PHP/PostgreSQL

Récupération d'enregistrements depuis une table

```
$dbconn = pg_connect ("host=localhost dbname=livres
                       user=nmoyroud password=toto");
if (!$dbconn) {echo 'Erreur'; exit;}

$result = pg_query($dbconn, "SELECT id, nom, email FROM auteurs");
if (!$result) {echo 'Erreur'; exit;}

echo '<table border="1" cellspacing="0">';
echo '<tr><td>Id</td><td>Auteur</td><td>Email</td></tr>';
while ($row = pg_fetch_assoc($result)) {
  echo '<tr><td>'.$row['id'].'</td>';
  echo '<td>'.$row['nom'].'</td>';
  echo '<td>'.$row['email'].'</td></tr>';
}
echo '</table>';
```

Exemple d'interaction PHP/PostgreSQL

Insertion d'enregistrements dans une table

```php
$dbconn = pg_connect ("host=localhost dbname=livres
                       user=nmoyroud password=toto");
if (!$dbconn) {echo 'Erreur'; exit;}

// Exemple en utilisant pg_query_params
$values = array($_POST['nom'],$_POST['email']));
$sql = 'INSERT INTO auteurs(nom,email) VALUES ($1,$2)';
$result = pg_query_params($dbconn, $sql, $values);
if (!$result) {echo "Erreur d'insertion"; exit;}

// Exemple en utilisant pg_query
$nom = pg_escape_string($_POST['nom']);
$email = pg_escape_string($_POST['email']);
$sql = "INSERT INTO auteurs(nom,email) VALUES ('$nom','$email')";
$result = pg_query ($dbconn, $sql);
if (!$result) {echo "Erreur d'insertion"; exit;}
```

Plan

1 Programmation orientée objet avec PHP5

2 Interaction avec les bases de données

3 Développement d'interfaces web pour les bases de données

4 Principes de fonctionnement des sessions

Développement d'interfaces web

- phpPgAdmin → interface web en PHP pour l'administration et la gestion des bases PostgreSQL, elle est adaptée pour ceux qui connaissent le fonctionnement des bases de données
- Pour des utilisateurs "non-informaticiens", il est nécessaire de présenter le contenu des bases de manière plus conviviale, en cachant la complexité des SGBD
- Avec PHP, on peut pré-écrire des requêtes SQL et les faire exécuter depuis une interface graphique accessible sur le web
- Les interfaces développées permettront aux clients non seulement de visualiser les données, mais également de les insérer / mettre à jour grâce à l'utilisation de formulaires HTML

Développement d'interfaces web

- phpPgAdmin → interface web en PHP pour l'administration et la gestion des bases PostgreSQL, elle est adaptée pour ceux qui connaissent le fonctionnement des bases de données
- Pour des utilisateurs "non-informaticiens", il est nécessaire de présenter le contenu des bases de manière plus conviviale, en cachant la complexité des SGBD
- Avec PHP, on peut pré-écrire des requêtes SQL et les faire exécuter depuis une interface graphique accessible sur le web
- Les interfaces développées permettront aux clients non seulement de visualiser les données, mais également de les insérer / mettre à jour grâce à l'utilisation de formulaires HTML

Développement d'interfaces web

- phpPgAdmin → interface web en PHP pour l'administration et la gestion des bases PostgreSQL, elle est adaptée pour ceux qui connaissent le fonctionnement des bases de données
- Pour des utilisateurs "non-informaticiens", il est nécessaire de présenter le contenu des bases de manière plus conviviale, en cachant la complexité des SGBD
- Avec PHP, on peut pré-écrire des requêtes SQL et les faire exécuter depuis une interface graphique accessible sur le web
- Les interfaces développées permettront aux clients non seulement de visualiser les données, mais également de les insérer / mettre à jour grâce à l'utilisation de formulaires HTML

Développement d'interfaces web

- phpPgAdmin → interface web en PHP pour l'administration et la gestion des bases PostgreSQL, elle est adaptée pour ceux qui connaissent le fonctionnement des bases de données
- Pour des utilisateurs "non-informaticiens", il est nécessaire de présenter le contenu des bases de manière plus conviviale, en cachant la complexité des SGBD
- Avec PHP, on peut pré-écrire des requêtes SQL et les faire exécuter depuis une interface graphique accessible sur le web
- Les interfaces développées permettront aux clients non seulement de visualiser les données, mais également de les insérer / mettre à jour grâce à l'utilisation de formulaires HTML

Exemple d'interface web

formAuthor.php : formulaire de saisie de valeurs

```
<form name="formAuthor" action="insertAuthor.php" method="POST">
Nom : <input type="text" name="nom" size ="20" /> <br />
Email : <input type="text" name="email" size="30" /> <br />
Pays : <select name="pays">
<option value="1">USA</option>
<option value="2">France</option>
<option value="3">Angleterre</option>
</select> <br />
<input type="submit" value="Envoyez" />
</form>
```

- Ce formulaire envoie les valeurs saisies au script PHP insertAuthor.php
- Deux champs texte permettent la saisie libre du nom et de l'email
- Un champ pays permet de choisir parmi une liste pré-définie : la valeur envoyée est celle précisée dans l'attribut value de l'option sélectionnée

Exemple d'interface web

formAuthor.php : formulaire de saisie de valeurs

```
<form name="formAuthor" action="insertAuthor.php" method="POST">
Nom : <input type="text" name="nom" size ="20" /> <br />
Email : <input type="text" name="email" size="30" /> <br />
Pays : <select name="pays">
<option value="1">USA</option>
<option value="2">France</option>
<option value="3">Angleterre</option>
</select> <br />
<input type="submit" value="Envoyez" />
</form>
```

- Ce formulaire envoie les valeurs saisies au script PHP insertAuthor.php
- Deux champs texte permettent la saisie libre du nom et de l'email
- Un champ pays permet de choisir parmi une liste pré-définie : la valeur envoyée est celle précisée dans l'attribut value de l'option sélectionnée

Exemple d'interface web

formAuthor.php : formulaire de saisie de valeurs

```
<form name="formAuthor" action="insertAuthor.php" method="POST">
Nom : <input type="text" name="nom" size ="20" /> <br />
Email : <input type="text" name="email" size="30" /> <br />
Pays : <select name="pays">
<option value="1">USA</option>
<option value="2">France</option>
<option value="3">Angleterre</option>
</select> <br />
<input type="submit" value="Envoyez" />
</form>
```

- Ce formulaire envoie les valeurs saisies au script PHP insertAuthor.php
- Deux champs texte permettent la saisie libre du nom et de l'email
- Un champ pays permet de choisir parmi une liste pré-définie : la valeur envoyée est celle précisée dans l'attribut value de l'option sélectionnée

Exemple d'interface web

Création dynamique de la liste pays

```
$result = pg_query($dbconn, "SELECT id,nom_pays FROM pays");
if (!$result) {echo 'Erreur'; exit;}

echo '<select name="pays">';
while ($row = pg_fetch_assoc($result)) {
  echo '<option value="'.$row['id'].'">';
  echo $row['nom_pays'];
  echo '</option>';
}
echo '</select>';
```

- Exemple d'extraction de la liste des pays depuis une table de la base
- PHP est utilisé ici en amont du formulaire de saisie

Exemple d'interface web

Création dynamique de la liste pays

```
$result = pg_query($dbconn, "SELECT id,nom_pays FROM pays");
if (!$result) {echo 'Erreur'; exit;}

echo '<select name="pays">';
while ($row = pg_fetch_assoc($result)) {
  echo '<option value="'.$row['id'].'">';
  echo $row['nom_pays'];
  echo '</option>';
}
echo '</select>';
```

- Exemple d'extraction de la liste des pays depuis une table de la base
- PHP est utilisé ici en amont du formulaire de saisie

Exemple d'interface web

insertAuthor.php : traitement des valeurs saisies

```
$dbconn = pg_connect ("host=localhost dbname=livres
                       user=nmoyroud password=toto");
if (!$dbconn) {echo 'Erreur'; exit;}

$values = array($_POST['nom'],$_POST['email'],$_POST['pays']);
$sql = 'INSERT INTO auteurs(nom,email,id_pays) VALUES ($1,$2,$3)';
$result = pg_query_params($dbconn, $sql, $values);
if (!$result) {echo "Erreur d'insertion"; exit;}

echo 'Informations ajoutees dans la base<br />';
echo '<a href="formAuthor.php">Ajouter un autre auteur</a>';
```

- Les valeurs saisies dans le formulaire sont stockées dans la base de données avec un message de confirmation
- Un lien permet de revenir au formulaire de saisie
- PHP est utilisé ici en aval du formulaire de saisie

Exemple d'interface web

insertAuthor.php : traitement des valeurs saisies

```
$dbconn = pg_connect ("host=localhost dbname=livres
                       user=nmoyroud password=toto");
if (!$dbconn) {echo 'Erreur'; exit;}

$values = array($_POST['nom'],$_POST['email'],$_POST['pays']);
$sql = 'INSERT INTO auteurs(nom,email,id_pays) VALUES ($1,$2,$3)';
$result = pg_query_params($dbconn, $sql, $values);
if (!$result) {echo "Erreur d'insertion"; exit;}

echo 'Informations ajoutees dans la base<br />';
echo '<a href="formAuthor.php">Ajouter un autre auteur</a>';
```

- Les valeurs saisies dans le formulaire sont stockées dans la base de données avec un message de confirmation
- Un lien permet de revenir au formulaire de saisie
- PHP est utilisé ici en aval du formulaire de saisie

Exemple d'interface web

insertAuthor.php : traitement des valeurs saisies

```
$dbconn = pg_connect ("host=localhost dbname=livres
                       user=nmoyroud password=toto");
if (!$dbconn) {echo 'Erreur'; exit;}

$values = array($_POST['nom'],$_POST['email'],$_POST['pays']);
$sql = 'INSERT INTO auteurs(nom,email,id_pays) VALUES ($1,$2,$3)';
$result = pg_query_params($dbconn, $sql, $values);
if (!$result) {echo "Erreur d'insertion"; exit;}

echo 'Informations ajoutees dans la base<br />';
echo '<a href="formAuthor.php">Ajouter un autre auteur</a>';
```

- Les valeurs saisies dans le formulaire sont stockées dans la base de données avec un message de confirmation
- Un lien permet de revenir au formulaire de saisie
- PHP est utilisé ici en aval du formulaire de saisie

Exemple d'interface web

insertAuthor.php : ajout du contrôle des valeurs saisies

```
$nom = trim($_POST['nom']);
$email = trim($_POST['email']);
if (empty($nom) || empty($email)) {
    echo 'Veuillez saisir un nom et un email.<br />';
    echo '<a href="formAuthor.php">Recommencer la saisie</a>';
}
else {... Traitement de la requete ...}
```

- On contrôle les valeurs nom et email qui sont obligatoires dans la base de données
- On pourrait ajouter le contrôle du format de l'email avec une fonction qui appelerait par exemple une expression régulière

Exemple d'interface web

insertAuthor.php : ajout du contrôle des valeurs saisies

```
$nom = trim($_POST['nom']);
$email = trim($_POST['email']);
if (empty($nom) || empty($email)) {
    echo 'Veuillez saisir un nom et un email.<br />';
    echo '<a href="formAuthor.php">Recommencer la saisie</a>';
}
else {... Traitement de la requete ...}
```

- On contrôle les valeurs nom et email qui sont obligatoires dans la base de données
- On pourrait ajouter le contrôle du format de l'email avec une fonction qui appelerait par exemple une expression régulière

Utilisation de la classe HTML_QuickForm

- Dans les exemples précédents, on a créé le formulaire directement en écrivant du HTML dans le code PHP, ce qui le rend peu lisible et difficile à débugguer
- On a également géré la vérification des valeurs, mais uniquement du côté du serveur : l'utilisateur est obligé d'envoyer le formulaire avant de voir ses erreurs
- Pour améliorer notre code et simplifier la création des interfaces, on peut utiliser une classe PHP qui est disponible dans le dépôt PEAR : HTML_QuickForm
- Elle permet de générer des formulaires et leur traitement avec la syntaxe objet de PHP

`http://pear.php.net/package/HTML_QuickForm`

`http://php.developpez.com/cours/pear/html-quickform/`

`http://www.midnighthax.com/quickform.php`

Utilisation de la classe HTML_QuickForm

- Dans les exemples précédents, on a créé le formulaire directement en écrivant du HTML dans le code PHP, ce qui le rend peu lisible et difficile à débugguer
- On a également géré la vérification des valeurs, mais uniquement du côté du serveur : l'utilisateur est obligé d'envoyer le formulaire avant de voir ses erreurs
- Pour améliorer notre code et simplifier la création des interfaces, on peut utiliser une classe PHP qui est disponible dans le dépôt PEAR : HTML_QuickForm
- Elle permet de générer des formulaires et leur traitement avec la syntaxe objet de PHP

`http://pear.php.net/package/HTML_QuickForm`

`http://php.developpez.com/cours/pear/html-quickform/`

`http://www.midnighthax.com/quickform.php`

Utilisation de la classe HTML_QuickForm

- Dans les exemples précédents, on a créé le formulaire directement en écrivant du HTML dans le code PHP, ce qui le rend peu lisible et difficile à débugguer
- On a également géré la vérification des valeurs, mais uniquement du côté du serveur : l'utilisateur est obligé d'envoyer le formulaire avant de voir ses erreurs
- Pour améliorer notre code et simplifier la création des interfaces, on peut utiliser une classe PHP qui est disponible dans le dépôt PEAR : HTML_QuickForm
- Elle permet de générer des formulaires et leur traitement avec la syntaxe objet de PHP

`http://pear.php.net/package/HTML_QuickForm`

`http://php.developpez.com/cours/pear/html-quickform/`

`http://www.midnighthax.com/quickform.php`

Utilisation de la classe HTML_QuickForm

- Dans les exemples précédents, on a créé le formulaire directement en écrivant du HTML dans le code PHP, ce qui le rend peu lisible et difficile à débugguer
- On a également géré la vérification des valeurs, mais uniquement du côté du serveur : l'utilisateur est obligé d'envoyer le formulaire avant de voir ses erreurs
- Pour améliorer notre code et simplifier la création des interfaces, on peut utiliser une classe PHP qui est disponible dans le dépôt PEAR : HTML_QuickForm
- Elle permet de générer des formulaires et leur traitement avec la syntaxe objet de PHP

`http://pear.php.net/package/HTML_QuickForm`

`http://php.developpez.com/cours/pear/html-quickform/`

`http://www.midnighthax.com/quickform.php`

Exemple d'utilisation de HTML_QuickForm

formAuthor.php : formulaire avec HTML_QuickForm

```php
require_once 'HTML/QuickForm.php';
$form = new HTML_QuickForm('formAuthor');
$form->setDefaults(array('nom' => 'Isaac Asimov',
                         'email' => 'asimov@sci-fi.pa',
                         'pays' => '1'));

$form->addElement('header', null, 'Auteur');
$form->addElement('text', 'nom', 'Nom :', array('size'=>20,'maxlength'=>30));
$form->addElement('text', 'email', 'Email :', array('size'=>30,'maxlength'=>50));
$form->addElement('select', 'pays', 'Pays :', $liste_pays);
$form->addElement('submit', null, 'Envoyer');

$form->applyFilter('__ALL__', 'trim');
$form->applyFilter('__ALL__', 'pg_escape_string');
$form->addRule('name', 'Saisissez un nom', 'required', null, 'client');
$form->addRule('email', 'Saisissez un email', 'required', null, 'client');
$form->addRule('email', 'Email non valide', 'regex',
                        '/^[a-z0-9_\.-]+@[a-z0-9_-]+\.[a-z]{2}$/', 'client');

if ($form->validate()) { // si les valeurs sont validees, on insere
    include 'insertAuthor.php';
    exit;
}
$form->display(); // affichage du formulaire
```

Utilisation de la classe HTML_QuickForm

- méthode setDefaults → affiche des valeurs par défaut dans les champs du formulaire
- méthode addElement → ajoute un champ dans le formulaire, avec le type précisé dans le 1er paramètre et le nom précisé dans le 2ème
- méthode applyFilter → applique une fonction de traitement sur le champ précisé (__ALL__ pour tous les champs)
- méthode addRule → ajoute une règle de vérification de la valeur saisie dans un champ (avec possibilité de validation côté client, sans envoi du formulaire)
- méthode validate → retourne vrai si les valeurs saisies sont validées
- méthode display → affiche le formulaire

Utilisation de la classe HTML_QuickForm

- méthode setDefaults → affiche des valeurs par défaut dans les champs du formulaire
- méthode addElement → ajoute un champ dans le formulaire, avec le type précisé dans le 1er paramètre et le nom précisé dans le 2ème
- méthode applyFilter → applique une fonction de traitement sur le champ précisé (__ALL__ pour tous les champs)
- méthode addRule → ajoute une règle de vérification de la valeur saisie dans un champ (avec possibilité de validation côté client, sans envoi du formulaire)
- méthode validate → retourne vrai si les valeurs saisies sont validées
- méthode display → affiche le formulaire

Utilisation de la classe HTML_QuickForm

- méthode setDefaults → affiche des valeurs par défaut dans les champs du formulaire
- méthode addElement → ajoute un champ dans le formulaire, avec le type précisé dans le 1er paramètre et le nom précisé dans le 2ème
- méthode applyFilter → applique une fonction de traitement sur le champ précisé (__ALL__ pour tous les champs)
- méthode addRule → ajoute une règle de vérification de la valeur saisie dans un champ (avec possibilité de validation côté client, sans envoi du formulaire)
- méthode validate → retourne vrai si les valeurs saisies sont validées
- méthode display → affiche le formulaire

Utilisation de la classe HTML_QuickForm

- méthode setDefaults → affiche des valeurs par défaut dans les champs du formulaire
- méthode addElement → ajoute un champ dans le formulaire, avec le type précisé dans le 1er paramètre et le nom précisé dans le 2ème
- méthode applyFilter → applique une fonction de traitement sur le champ précisé (__ALL__ pour tous les champs)
- méthode addRule → ajoute une règle de vérification de la valeur saisie dans un champ (avec possibilité de validation côté client, sans envoi du formulaire)
- méthode validate → retourne vrai si les valeurs saisies sont validées
- méthode display → affiche le formulaire

Utilisation de la classe HTML_QuickForm

- méthode setDefaults → affiche des valeurs par défaut dans les champs du formulaire
- méthode addElement → ajoute un champ dans le formulaire, avec le type précisé dans le 1er paramètre et le nom précisé dans le 2ème
- méthode applyFilter → applique une fonction de traitement sur le champ précisé (__ALL__ pour tous les champs)
- méthode addRule → ajoute une règle de vérification de la valeur saisie dans un champ (avec possibilité de validation côté client, sans envoi du formulaire)
- méthode validate → retourne vrai si les valeurs saisies sont validées
- méthode display → affiche le formulaire

Utilisation de la classe HTML_QuickForm

- méthode setDefaults → affiche des valeurs par défaut dans les champs du formulaire
- méthode addElement → ajoute un champ dans le formulaire, avec le type précisé dans le 1er paramètre et le nom précisé dans le 2ème
- méthode applyFilter → applique une fonction de traitement sur le champ précisé (__ALL__ pour tous les champs)
- méthode addRule → ajoute une règle de vérification de la valeur saisie dans un champ (avec possibilité de validation côté client, sans envoi du formulaire)
- méthode validate → retourne vrai si les valeurs saisies sont validées
- méthode display → affiche le formulaire

Utilisation de la classe HTML_QuickForm

Exemple de listes déroulantes liées avec hierselect

```
require_once 'HTML/QuickForm.php';
$form = new HTML_QuickForm('example');

$select1[0] = 'Science-fiction';
$select1[1] = 'Humour';
$select1[2] = 'Litterature francaise';
$select2[0][0] = $select2[1][0] = $select2[2][0] = '--- Auteur ---';
$select2[0][1] = 'Isaac Asimov';
$select2[0][2] = 'Douglas Adams';
$select2[1][1] = 'Pierre Desproges';
$select2[1][2] = 'Raymond Devos';
$select2[2][1] = 'Victor Hugo';
$select2[2][2] = 'Emile Zola';

$sel = $form->addElement('hierselect', 'auteurs', 'Choisissez un auteur');
$sel->setOptions(array($select1, $select2));
$form->display(); // affichage du formulaire
```

- HTML_QuickForm ajoute des éléments supplémentaires au HTML
- hierselect permet de réaliser des listes déroulantes liées : les valeurs affichées dans la liste du dessous dépendront de la valeur choisie dans celle du dessus (avec autant de niveaux que l'on souhaite)

Utilisation de la classe HTML_QuickForm

Exemple de listes déroulantes liées avec hierselect

```php
require_once 'HTML/QuickForm.php';
$form = new HTML_QuickForm('example');

$select1[0] = 'Science-fiction';
$select1[1] = 'Humour';
$select1[2] = 'Litterature francaise';
$select2[0][0] = $select2[1][0] = $select2[2][0] = '--- Auteur ---';
$select2[0][1] = 'Isaac Asimov';
$select2[0][2] = 'Douglas Adams';
$select2[1][1] = 'Pierre Desproges';
$select2[1][2] = 'Raymond Devos';
$select2[2][1] = 'Victor Hugo';
$select2[2][2] = 'Emile Zola';

$sel = $form->addElement('hierselect', 'auteurs', 'Choisissez un auteur');
$sel->setOptions(array($select1, $select2));
$form->display(); // affichage du formulaire
```

- HTML_QuickForm ajoute des éléments supplémentaires au HTML
- hierselect permet de réaliser des listes déroulantes liées : les valeurs affichées dans la liste du dessous dépendront de la valeur choisie dans celle du dessus (avec autant de niveaux que l'on souhaite)

Plan

1 Programmation orientée objet avec PHP5

2 Interaction avec les bases de données

3 Développement d'interfaces web pour les bases de données

4 Principes de fonctionnement des sessions

Les limites du protocole HTTP

- Le protocole HTTP est sans état ("stateless") : pour le serveur chaque requête reçue est indépendante de la précédente et de la suivante
- Quand on développe une application sur le web, on a besoin d'un mécanisme qui permet de se souvenir des actions réalisées précédemment par un utilisateur
- 1ère réponse : le cookie qui permet de stocker sur l'ordinateur du client un petit fichier texte contenant les informations
- Le cookie sera ensuite transmis dans l'en-tête de toutes les requêtes HTTP vers le domaine associé et ses valeurs accessibles en PHP par la variable $_COOKIE
- Limites : faible capacité de stockage, pas toujours accepté par les clients, transit de toutes les informations à chaque requête
- Problème de sécurité : le cookie est stocké en clair sur le client et peut donc être modifié facilement par celui-ci

Les limites du protocole HTTP

- Le protocole HTTP est sans état ("stateless") : pour le serveur chaque requête reçue est indépendante de la précédente et de la suivante
- Quand on développe une application sur le web, on a besoin d'un mécanisme qui permet de se souvenir des actions réalisées précédemment par un utilisateur
- 1ère réponse : le cookie qui permet de stocker sur l'ordinateur du client un petit fichier texte contenant les informations
- Le cookie sera ensuite transmis dans l'en-tête de toutes les requêtes HTTP vers le domaine associé et ses valeurs accessibles en PHP par la variable $_COOKIE
- Limites : faible capacité de stockage, pas toujours accepté par les clients, transit de toutes les informations à chaque requête
- Problème de sécurité : le cookie est stocké en clair sur le client et peut donc être modifié facilement par celui-ci

Les limites du protocole HTTP

- Le protocole HTTP est sans état ("stateless") : pour le serveur chaque requête reçue est indépendante de la précédente et de la suivante
- Quand on développe une application sur le web, on a besoin d'un mécanisme qui permet de se souvenir des actions réalisées précédemment par un utilisateur
- 1ère réponse : le cookie qui permet de stocker sur l'ordinateur du client un petit fichier texte contenant les informations
- Le cookie sera ensuite transmis dans l'en-tête de toutes les requêtes HTTP vers le domaine associé et ses valeurs accessibles en PHP par la variable $_COOKIE
- Limites : faible capacité de stockage, pas toujours accepté par les clients, transit de toutes les informations à chaque requête
- Problème de sécurité : le cookie est stocké en clair sur le client et peut donc être modifié facilement par celui-ci

Les limites du protocole HTTP

- Le protocole HTTP est sans état ("stateless") : pour le serveur chaque requête reçue est indépendante de la précédente et de la suivante
- Quand on développe une application sur le web, on a besoin d'un mécanisme qui permet de se souvenir des actions réalisées précédemment par un utilisateur
- 1ère réponse : le cookie qui permet de stocker sur l'ordinateur du client un petit fichier texte contenant les informations
- Le cookie sera ensuite transmis dans l'en-tête de toutes les requêtes HTTP vers le domaine associé et ses valeurs accessibles en PHP par la variable $_COOKIE
- Limites : faible capacité de stockage, pas toujours accepté par les clients, transit de toutes les informations à chaque requête
- Problème de sécurité : le cookie est stocké en clair sur le client et peut donc être modifié facilement par celui-ci

Les limites du protocole HTTP

- Le protocole HTTP est sans état ("stateless") : pour le serveur chaque requête reçue est indépendante de la précédente et de la suivante
- Quand on développe une application sur le web, on a besoin d'un mécanisme qui permet de se souvenir des actions réalisées précédemment par un utilisateur
- 1ère réponse : le cookie qui permet de stocker sur l'ordinateur du client un petit fichier texte contenant les informations
- Le cookie sera ensuite transmis dans l'en-tête de toutes les requêtes HTTP vers le domaine associé et ses valeurs accessibles en PHP par la variable $_COOKIE
- Limites : faible capacité de stockage, pas toujours accepté par les clients, transit de toutes les informations à chaque requête
- Problème de sécurité : le cookie est stocké en clair sur le client et peut donc être modifié facilement par celui-ci

Les limites du protocole HTTP

- Le protocole HTTP est sans état ("stateless") : pour le serveur chaque requête reçue est indépendante de la précédente et de la suivante
- Quand on développe une application sur le web, on a besoin d'un mécanisme qui permet de se souvenir des actions réalisées précédemment par un utilisateur
- 1ère réponse : le cookie qui permet de stocker sur l'ordinateur du client un petit fichier texte contenant les informations
- Le cookie sera ensuite transmis dans l'en-tête de toutes les requêtes HTTP vers le domaine associé et ses valeurs accessibles en PHP par la variable $_COOKIE
- Limites : faible capacité de stockage, pas toujours accepté par les clients, transit de toutes les informations à chaque requête
- Problème de sécurité : le cookie est stocké en clair sur le client et peut donc être modifié facilement par celui-ci

Le mécanisme des sessions avec PHP

- Pour pallier aux limites des cookies, PHP propose le mécanisme des sessions
- Les informations à conserver sont stockées sur le serveur et un identifiant de session de 32 caractères est généré aléatoirement
- Cet identifiant est ensuite le seul élément envoyé et stocké côté client
- Pour permettre l'accès aux informations, l'identifiant est transmis vers le serveur soit grâce à un cookie, soit dans l'URL des pages si les cookies ne sont pas acceptés par le client
- Tout le mécanisme est géré automatiquement par PHP, vous devez simplement démarrez une session pour l'utiliser
- Une session a une durée de vie limitée (par défaut 30 minutes) et se termine également quand l'utilisateur ferme son navigateur

`http://php.developpez.com/cours/sessions/`

`http://beaussier.developpez.com/articles/php/session/`

Le mécanisme des sessions avec PHP

- Pour pallier aux limites des cookies, PHP propose le mécanisme des sessions
- Les informations à conserver sont stockées sur le serveur et un identifiant de session de 32 caractères est généré aléatoirement
- Cet identifiant est ensuite le seul élément envoyé et stocké côté client
- Pour permettre l'accès aux informations, l'identifiant est transmis vers le serveur soit grâce à un cookie, soit dans l'URL des pages si les cookies ne sont pas acceptés par le client
- Tout le mécanisme est géré automatiquement par PHP, vous devez simplement démarrez une session pour l'utiliser
- Une session a une durée de vie limitée (par défaut 30 minutes) et se termine également quand l'utilisateur ferme son navigateur

`http://php.developpez.com/cours/sessions/`

`http://beaussier.developpez.com/articles/php/session/`

Le mécanisme des sessions avec PHP

- Pour pallier aux limites des cookies, PHP propose le mécanisme des sessions
- Les informations à conserver sont stockées sur le serveur et un identifiant de session de 32 caractères est généré aléatoirement
- Cet identifiant est ensuite le seul élément envoyé et stocké côté client
- Pour permettre l'accès aux informations, l'identifiant est transmis vers le serveur soit grâce à un cookie, soit dans l'URL des pages si les cookies ne sont pas acceptés par le client
- Tout le mécanisme est géré automatiquement par PHP, vous devez simplement démarrez une session pour l'utiliser
- Une session a une durée de vie limitée (par défaut 30 minutes) et se termine également quand l'utilisateur ferme son navigateur

`http://php.developpez.com/cours/sessions/`

`http://beaussier.developpez.com/articles/php/session/`

Le mécanisme des sessions avec PHP

- Pour pallier aux limites des cookies, PHP propose le mécanisme des sessions
- Les informations à conserver sont stockées sur le serveur et un identifiant de session de 32 caractères est généré aléatoirement
- Cet identifiant est ensuite le seul élément envoyé et stocké côté client
- Pour permettre l'accès aux informations, l'identifiant est transmis vers le serveur soit grâce à un cookie, soit dans l'URL des pages si les cookies ne sont pas acceptés par le client
- Tout le mécanisme est géré automatiquement par PHP, vous devez simplement démarrez une session pour l'utiliser
- Une session a une durée de vie limitée (par défaut 30 minutes) et se termine également quand l'utilisateur ferme son navigateur

http://php.developpez.com/cours/sessions/

http://beaussier.developpez.com/articles/php/session/

Le mécanisme des sessions avec PHP

- Pour pallier aux limites des cookies, PHP propose le mécanisme des sessions
- Les informations à conserver sont stockées sur le serveur et un identifiant de session de 32 caractères est généré aléatoirement
- Cet identifiant est ensuite le seul élément envoyé et stocké côté client
- Pour permettre l'accès aux informations, l'identifiant est transmis vers le serveur soit grâce à un cookie, soit dans l'URL des pages si les cookies ne sont pas acceptés par le client
- Tout le mécanisme est géré automatiquement par PHP, vous devez simplement démarrez une session pour l'utiliser
- Une session a une durée de vie limitée (par défaut 30 minutes) et se termine également quand l'utilisateur ferme son navigateur

http://php.developpez.com/cours/sessions/

http://beaussier.developpez.com/articles/php/session/

Le mécanisme des sessions avec PHP

- Pour pallier aux limites des cookies, PHP propose le mécanisme des sessions
- Les informations à conserver sont stockées sur le serveur et un identifiant de session de 32 caractères est généré aléatoirement
- Cet identifiant est ensuite le seul élément envoyé et stocké côté client
- Pour permettre l'accès aux informations, l'identifiant est transmis vers le serveur soit grâce à un cookie, soit dans l'URL des pages si les cookies ne sont pas acceptés par le client
- Tout le mécanisme est géré automatiquement par PHP, vous devez simplement démarrez une session pour l'utiliser
- Une session a une durée de vie limitée (par défaut 30 minutes) et se termine également quand l'utilisateur ferme son navigateur

`http://php.developpez.com/cours/sessions/`

`http://beaussier.developpez.com/articles/php/session/`

Exemple simple d'utilisation d'une session

sessionStart.php : création de session et enregistrement d'une variable

```php
session_start();
$_SESSION['prenom'] = 'Nicolas';
```

sessionRead.php : ouverture de session et récupération d'une variable

```php
session_start();
if (isset($_SESSION['prenom'])) {echo $_SESSION['prenom'];}
else {echo 'Aucun prenom enregistre.';}
```

- La session est créée ou ré-ouverte avec la fonction session_start()
- Les informations sont stockées dans le tableau $_SESSION

Exemple simple d'utilisation d'une session

sessionStart.php : création de session et enregistrement d'une variable

```
session_start();
$_SESSION['prenom'] = 'Nicolas';
```

sessionRead.php : ouverture de session et récupération d'une variable

```
session_start();
if (isset($_SESSION['prenom'])) {echo $_SESSION['prenom'];}
else {echo 'Aucun prenom enregistre.';}
```

- La session est créée ou ré-ouverte avec la fonction session_start()
- Les informations sont stockées dans le tableau $_SESSION

Exemple simple d'utilisation d'une session

sessionSuppr.php : suppression de variables et de la session

```php
session_start();
unset ($_SESSION['prenom']); // suppression d'une variable
$_SESSION = array(); // suppression de toutes les variables
session_destroy(); // destruction complete de la session
```

- Ne jamais faire unset($_SESSION) car cela rend impossible tout accès ultérieur aux variables de session jusqu'à sa destruction

Utilisation de sessions pour l'authentification

- Les sessions permettent de mettre en place un système d'authentification pour l'accès à des fonctionnalités sensibles (ajout, suppression et modification de données)
- On va utiliser pour le TP une classe AccessControl qui permet de gérer l'authentification d'un utilisateur grâce à un formulaire
- L'appel de la classe doit être inclus dans toutes les pages que l'on souhaite sécuriser
- Cette classe vérifie que l'utilisateur est connecté à chaque fois qu'il demande l'accès à une de ces pages
- La liste des utilisateurs autorisés, leur mot de passe et leur niveau d'accès sont stockés dans une base de données

Utilisation de sessions pour l'authentification

- Les sessions permettent de mettre en place un système d'authentification pour l'accès à des fonctionnalités sensibles (ajout, suppression et modification de données)
- On va utiliser pour le TP une classe AccessControl qui permet de gérer l'authentification d'un utilisateur grâce à un formulaire
- L'appel de la classe doit être inclus dans toutes les pages que l'on souhaite sécuriser
- Cette classe vérifie que l'utilisateur est connecté à chaque fois qu'il demande l'accès à une de ces pages
- La liste des utilisateurs autorisés, leur mot de passe et leur niveau d'accès sont stockés dans une base de données

Utilisation de sessions pour l'authentification

- Les sessions permettent de mettre en place un système d'authentification pour l'accès à des fonctionnalités sensibles (ajout, suppression et modification de données)
- On va utiliser pour le TP une classe AccessControl qui permet de gérer l'authentification d'un utilisateur grâce à un formulaire
- L'appel de la classe doit être inclus dans toutes les pages que l'on souhaite sécuriser
- Cette classe vérifie que l'utilisateur est connecté à chaque fois qu'il demande l'accès à une de ces pages
- La liste des utilisateurs autorisés, leur mot de passe et leur niveau d'accès sont stockés dans une base de données

Utilisation de sessions pour l'authentification

- Les sessions permettent de mettre en place un système d'authentification pour l'accès à des fonctionnalités sensibles (ajout, suppression et modification de données)
- On va utiliser pour le TP une classe AccessControl qui permet de gérer l'authentification d'un utilisateur grâce à un formulaire
- L'appel de la classe doit être inclus dans toutes les pages que l'on souhaite sécuriser
- Cette classe vérifie que l'utilisateur est connecté à chaque fois qu'il demande l'accès à une de ces pages
- La liste des utilisateurs autorisés, leur mot de passe et leur niveau d'accès sont stockés dans une base de données

Utilisation de sessions pour l'authentification

- Les sessions permettent de mettre en place un système d'authentification pour l'accès à des fonctionnalités sensibles (ajout, suppression et modification de données)
- On va utiliser pour le TP une classe AccessControl qui permet de gérer l'authentification d'un utilisateur grâce à un formulaire
- L'appel de la classe doit être inclus dans toutes les pages que l'on souhaite sécuriser
- Cette classe vérifie que l'utilisateur est connecté à chaque fois qu'il demande l'accès à une de ces pages
- La liste des utilisateurs autorisés, leur mot de passe et leur niveau d'accès sont stockés dans une base de données

Utilisation de la classe AccessControl

Code à inclure pour protéger l'accès à une page

```
require_once 'HTTP/AccessControl.class.php';
$acl = new AccessControl($dbconn,'utilisateurs',1);
$acl->run();
```

- 1er paramètre → variable qui contient la connexion à la base de données
- 2ème paramètre → nom de la table qui contient les utilisateurs (ses colonnes sont : id, login, password, userlevel)
- 3ème paramètre → niveau maximum que doit avoir l'utilisateur pour accéder à cette page (les utilisateurs de niveau 0 ont le maximum de droits)

Utilisation de la classe AccessControl

Code à inclure pour protéger l'accès à une page

```
require_once 'HTTP/AccessControl.class.php';
$acl = new AccessControl($dbconn,'utilisateurs',1);
$acl->run();
```

- 1er paramètre → variable qui contient la connexion à la base de données
- 2ème paramètre → nom de la table qui contient les utilisateurs (ses colonnes sont : id, login, password, userlevel)
- 3ème paramètre → niveau maximum que doit avoir l'utilisateur pour accéder à cette page (les utilisateurs de niveau 0 ont le maximum de droits)

Utilisation de la classe AccessControl

Code à inclure pour protéger l'accès à une page

```
require_once 'HTTP/AccessControl.class.php';
$acl = new AccessControl($dbconn,'utilisateurs',1);
$acl->run();
```

- 1er paramètre → variable qui contient la connexion à la base de données
- 2ème paramètre → nom de la table qui contient les utilisateurs (ses colonnes sont : id, login, password, userlevel)
- 3ème paramètre → niveau maximum que doit avoir l'utilisateur pour accéder à cette page (les utilisateurs de niveau 0 ont le maximum de droits)