Cours de PHP

Cours n°2 : Présentation du langage

Complément du cours N°1

Classes

Include et Require

- **Ceux deux fonctions permettent d'inclure du code PHP au sein d'une autre page**
  - Include
    - Permet d'inclure un fichier au sein d'une page PHP. Le code PHP du fichier sera exécuté. Un warning est levé si le fichier n'existe pas
  - Include_once
    - Idem que Include, mais la fonction s'assure que le fichier n'a été inclus qu'une fois
  - Require
    - Idem que Include, mais lève une erreur si le fichier n'existe pas
  - Require_once
    - Idem que Include_once, mais lève une erreur si le fichier n'existe pas

```php
<?php
   include('include/file.php');
?>
```

Classes

Multimedia
Technologies de l'Information

Classes

Présentation du modèle objet en PHP

- **PHP dispose d'un vrai model objet depuis PHP 5**
  - Création de classe
  - Création d'un constructeur
  - Création d'un destructeur (non natif en PHP 4)
  - Gestion de la portée des propriétés, attributs et méthodes (impossible en PHP4)
  - Définition des interfaces (impossible en PHP4)
  - Définition de classes abstraites (impossible en PHP4)
  - Définition de classes par héritage
    - Etc…

Contrairement à PHP4, PHP5 offre l'ensemble des fonctions nécessaires à la POO

Multimedia
Technologies de l'Information

Classes

Définition 1/3

- **Création d'une classe**
  - Une classe doit être écrite sur **un seul fichier** et **un seul bloc de code**

```php
<?php
   class Myclass
   {
       public $myAttribute; // un attribut
       public function myMethod()  // une méthode
       {
           echo 'Je suis une méthode.';
       }
   }
?>
```

- **Instanciation d'une classe**
  - Une classe est instanciée avec le mot clef **new**

```php
<?php
    $aclass = new MyClass();
?>
```

Multimedia
Technologies de l'Information

Classes

Définition 2/3

- **Définition d'un attribut**
  - Un attribut se définit comme une variable sauf qu'il est précédé un élément de portée

```php
<?php
   class <className>
   {
       <portée> $<attributeName>; // un attribut
   }
   <className>->$<attributeName>;
?>
```

```php
<?php
   class MyClass
   {
       public $myAttribute = 'Toto'; // valeur par défaut
   }
   $test = new MyClass();
   echo $test->myAttribute; // affiche Toto
```

- **Définition d'une méthode**
  - Une méthode se définit comme une fonction sauf qu'elle est précédée un élément de portée

```php
<?php
   class <className>
   {
       <portée> function <functionName>(<args>)
       {
           <instructions>
       }
   }
?>
```

```php
<?php
   class MyClass
   {
       public function returnToto()
       {
           return 'Toto';
       }
   }
   $test = new MyClass();
   echo $test-> returnToto; // affiche Toto
```

Multimedia
Technologies de l'Information

Classes

Définition 3/3

Attributs de portées 1/2

- Les éléments de portées existant en PHP sont :
  - private : Le membre de la classe est privé
  - protected : le membre de la classe est protégé
  - public : le membre de la classe est public
  - interface : la classe est une interface
  - implements : la classe met en œuvre une interface

```php
<?php
   interface iClass
   {
       public function myMethod();  // une méthode
   }
?>
```

```php
<?php
   class MyClass implements iClass
   {
      public function myMethod() // obligatoire
      {
         echo 'Je suis une fonction d\'interface';
      }
   }
?>
```

  - abstract : la classe est abstraite
  - final : la classe ne peut être héritée
  - static : la classe ou le membre de la classe est accessible sans appel à un constructeur
  - extends : la classe hérite d'une autre classe

Bonus : Constantes de classe (marche aussi pour les interfaces)

- Les constantes de classe sont **implicitement publiques**

```php
<?php
   class <className>
   {
       const <constanteName> = <value>; // un attribut
   }
   <className>::<attributeName>;
?>
```

Multimedia
Technologies de l'Information

Classes

Définition : Exercice

- **Possible ou impossible?**

```php
<?php
   interface iDrivable
   {
       public function start();
       public function stop();
   }
   class Auto implements iDrivable
   {
       const NB_ROUES = 4;
       protected $tankVolume = 50;
       private $color = 'grise';
       public function start()
       {
           echo 'Je démarre!';
       }
       public function stop()
       {
           echo 'Je m\'arrête!';
       }
       private function brake()
       {
           echo 'Je freine!';
       }
       public function printColor()
       {
           echo $this->color;
       }
       public function printTankLevel()
       {
           echo $this->tankVolume;
       }
   }
?>
```

```php
<?php
    $test = new Auto();     // OK
    echo $test::NB_ROUES;   // affiche 4
    echo Auto::NB_ROUES;    // affiche 4
    echo $test->tankVolume;    // erreur
    echo $test->start();    // affiche 'Je démarre'
    echo $test->stop();     // affiche 'Je freine'
    echo $test->brake();    // erreur
    echo $test->color;      // erreur
    $test->printColor();    // affiche 'grise'
    $test->printTankLevel(); // affiche 50

    class Car extends Auto
    {
        const NB_ROUES = 5; // Il faut compter la roue de secours
        protected $tankVolume = 60;
        private $color = 'verte';
    }

    $test2 = new Car();     // OK
    $test2->printColor();   // affiche 'grise'
    $test2->printTankLevel();  // affiche 60
    echo Auto::NB_ROUES;    // affiche 4
    echo Car::NB_ROUES;     // affiche 5

?>
```

Multimedia
Technologies de l'Information

Classes

Utilisation

- **La variable $this**
  - Elle peut être utilisée uniquement dans les méthodes de la classe et représente une référence vers l'instance appelante.
- **Le mot clef self**
  - Contrairement à $this, **self** appelle une méthode ou un attribut correspondant à la classe où l'appel est fait.

```php
<?php
   class Person
   {
      private $_name= 'Bob';
      public function setName($name)
      {
         $this->$_name = $name;
      }
      public function getName()
      {
         return $this->$_name;
      }
      public function getTitle()
      {
         return 'Mr '.$this->getName();
      }
      public function sayHi()
      {
         echo 'Hi, I\'m '.$this->getTitle();
      }
      public function sayBye()
      {
         echo 'Bye, I was '.self::getTitle();
      }
   }
?>
```

```php
<?php
   class Teacher extends Person
   {
      public function getTitle()
      {
         return 'Teacher '.$this->getName();
      }
   }
   $test = new Teacher();
   $test->setName('Arnaud');
   $test->sayHi();  // affiche 'Hi, I'm Teacher Arnaud'
   $test->sayBye(); // affiche 'Bye, I was Mr Arnaud'
?>
```

- **Self ignore le contexte de l'appel. La fonction appelée est celle de la classe ou se situe l'appel.**
- **Parent a le même fonctionnement que Self mais appelle la classe mère**

Classes

Méthodes magiques 1/3

- **Présentation**
  - Les méthodes magiques sont des méthodes prédéfinies par PHP appelée dans des cas précis.
- **__construct([$ …])**
  - La méthode magique **__construct()** est appelée lorsque la classe est créé. Pour faire simple **c'est le constructeur de la classe**.
- **__destruct()**
  - La méthode magique **__destruct()** est appelée lorsque la classe est libérée. Pour faire simple **c'est le destructeur de la classe**.
- **__set($name, $value)**
  - La méthode **__set()** est appelée lorsque un **attribut inaccessible** de la classe est **écrit**
- **__get($name)**
  - La méthode **__get()** est appelée lorsque un **attribut inaccessible** de la classe est **lu**

```php
<?php
   class MyClass
   {
      public function __construct()
      {
         echo 'Salut!';
      }
      public function __destruct()
      {
         echo 'Au revoir!';
      }
      public function __get($name)
      {
         echo 'Aye! '.$name.' n\'existe pas!';
      }
      public function __set($name, $value)
      {
         echo 'Aye! '.$name.' n\'existe pas!';
         echo $value.' est perdu :(';
      }
   }
   $test = new MyClass(); // affiche Salut!
   $test->toto = 10; // affiche Aye! toto n'existe pas!
   $tmp = $test->titi; /* affiche Aye! titi n'existe pas!
10 est perdu :( */
?> // à la fin du script 'Au revoir!' est affiché
```

Classes

Méthodes magiques 2/3

- **__call($name, $arguments)**
  - La méthode **__call()** est appelée lorsque une **méthode inaccessible** de la classe est appelée.
    - $arguments est un tableau.
- **__callstatic($name, $arguments)**
  - La méthode **__callstatic()** est appelée lorsque une **méthode static inaccessible** de la classe est appelée.
    - $arguments est un tableau.
- **__isset($name)**
  - La méthode **__isset()** est appelée lorsque la fonction **isset() est appelée avec un attribut inaccessibles** de la classe est accédé.
- **__unset($name)**
  - La méthode **__unset()** est appelée lorsque la fonction **unset() est appelée avec un attribut inaccessibles** de la classe est accédé.

```php
<?php
   class MyClass
   {
      public function __call($name, $arguments)
      {
         echo  'La fonction :'.$name.' est introuvable! ';
         echo  'Voici ses arguments : '.implode(',', $arguments);
      }
      public function __callstatic ($name, $arguments)
      {
         echo  'La fonction statique :'.$name.' est introuvable! ';
         echo  'Voici ses arguments : '.implode(',', $arguments);
      }
      public function __isset($name)
      {
         echo 'Aye! '.$name.' n\'existe pas!';
      }
      public function __unset($name)
      {
         echo 'Aye! '.$name.' n\'existe pas!';
      }
   }
   $test = new MyClass();
   $test->Toto('titi','Tata'); /* affiche La fonction Toto est
introuvable! Voici ses arguments : titi, Tata */
   MyClass->Test42(42, 42); /* affiche La fonction Test 42 est
introuvable! Voici ses arguments : 42, 42 */
   isset($test->toto); // affiche Aye! toto n'existe pas!
   unset($test->titi); // affiche Aye! titi n'existe pas!
?>
```

Classes

Méthodes magiques 3/3

- **__sleep()**
  - La méthode **__sleep()** est appelée lorsque **serialize()** est utilisé sur un objet un PHP. Elle doit renvoyer un tableau avec le nom des attributs devant être sauvegardés par la sérialisation (idéal pour ne pas sauvegarder les attributs inutiles)
- **__wakeup()**
  - La méthode **__wakeup()** est appelée lorsque **unserialize()** est utilisé sur un objet un PHP. Le but de cette fonction est principalement la restauration des connexions au BDD et la remise en conformité de l'objet.
- **__toString()**
  - La méthode **__toString()** est appelée lorsque l'**objet est utilisée comme une chaine** (comme par exemple lors d'un echo $maclasse)
- **__invoke([$ …])**
  - La méthode **__invoke()** est appelée lorsque l'**objet est utilisée comme une fonction**(comme par exemple lors d'un $maclasse('42'))
- **__clone()**
  - La méthode **__clone()** est appelée après qu'un **objet ait été cloné** (via le mot clef clone)

Classes

Les exceptions

- **Fonctionnement**
  - Le fonctionnement des exceptions est similaire aux autres langages de programmation
    - Une exception est une instance de la **classe Exception** ou héritant de celle-ci.
    - Une **exception est levée** avec le mot clé **throw**
    - Une **exception est récupérée** avec un bloc **try** … **catch**
    - Une **exception non récupérée** entrainera une **erreur fatale** sur le script PHP
    - Depuis PHP 5.5, le bloc **finally** fait son apparition. Il permet de rajouter un bloc de code à un bloc try…catch qui sera **toujours exécuté** même si une exception a été levée

```php
<?php
    function inverse($x)
    {
       if (!$x)
          throw new Exception('Division par zéro.');
       else
          return 1/$x;
    }
?>
```

```php
<?php
    try
    {
       echo inverse(0);
    }
    catch (Exception $e)
    {
       echo 'Exception reçue : '.  $e->getMessage();
    }
    finally
    {
        echo 'Ce bloc est toujours exécuté.';
    }
?>
```

Classes

Bonus

- **spl_autoload_register**
  - spl_autoload_register permet de définir une fonction qui sera appelée lorsqu'une instance de classe doit être créée et que celle-ci n'est pas connue dans le contexte.
  - L'objectif principal de cette fonction est d'automatiser les appels à include.

```php
<?php
    function my_autoloader($classname) {
        include 'classes/'.$classname.'.class.php';
    }

    spl_autoload_register('my_autoloader');
?>
```

  - Il est possible d'avoir plusieurs fonctions enregistrées via spl_autoload_register. Elles seront appelées par ordre d'ajout.
- **spl_autoload_unregister**
  - Cette fonction sert à désenregistrer une fonction.

Une requête SQL va dans un bar. Elle marche jusqu'entre deux tables et demande : Puis-je me joindre à vous?

Anonyme (exécuté)

Bases de données

Bases de données

Présentation de PDO

- **PDO est une classe PHP qui permet de communiquer avec des serveurs de données (Php Data Object)**
- **PDO est une couche d'abstraction**
  - Il permet la communication avec MySQL, Oracle, Postgresql, etc… de façon transparente
- **PDO permet de sécuriser les requêtes et la réutilisation du code grâce aux requêtes préparées**

Pour résumer : PDO c'est cool!

Bases de données

Se connecter

- **Les éléments de connexion**
  - Pour se connecter 3 informations sont nécessaires
    - Le DSN (Data Source Name) est une chaine de connexion qui contient :
      - Le type de serveur de base de données (le nom du pilote PDO)
      - L'hostname du serveur
      - Le nom de la base
    - Le nom de l'utilisateur
    - Le mot de passe

```php
<?php
    $dsn = 'mysql:host=localhost;dbname=mti_db';
    $user = 'student_s';
    $password = '42';
    $connection = new PDO($dsn, $user, $password );
?>
```

Bases de données

Requêter la base 1/2

- **Sélectionner des données (SELECT)**
  - Une fois notre objet PDO initialisé et créé, il est possible pour l'utiliser et faire des requêtes via la méthode **query**

```php
<?php
    $select= $connection->query('SELECT * FROM students');
?>
```

- **$select** contient un objet de **type PDOStatement** pour l'instant non exploitable. Il faut définir la façon dont les données vont nous être retournées

```php
<?php
    $select->setFetchMode(PDO::FETCH_OBJ);
?>
```

- **PDO::FETCH_OBJ** indique que les enregistrements doivent être **retournés sous forme d'objets** (une colonne dans la base représentera un attribut)

```php
<?php
    while( $student = $select->fetch() )
    {
        // Affichage du nom et prenom
        echo $student->lastname.':'.$student->firstname.'<br />';
    }
?>
```

- Plus d'informations sur les PDOStatement à cette adresse : http://www.php.net/manual/fr/class.pdostatement.php

Bases de données

Requêter la base 2/2

- **Créer, Mettre à jour et supprimer des données (INSERT, UPDATE et DELETE)**
  - La méthode **exec** est à utiliser pour faire des requêtes modifiant la BDD. Elle retourne le nombre d'enregistrement affecté par la requête.

```php
<?php
    $nbDelete = $connection->exec('DELETE FROM students WHERE firstname=\'Toto\'');
    echo $nbDelete.' enregistrements ont été supprimés';
?>
```

```php
<?php
    $nbDelete = $connection->exec("INSERT INTO students (firstname, lastname) VALUES ('Simon', 'Radier')");
    echo $nbDelete.' enregistrement a été ajouté';
?>
```

```php
<?php
    $nbDelete = $connection->exec("UPDATE students SET firstname='toto'");
    echo $nbDelete.' enregistrements ont été modifiés';
?>
```

Bases de données

Les requêtes préparées 1/3

- **Les requêtes préparées sont la vraie force de PDO**
  - Elles permettent la réutilisation du code
  - Elles permettent de sécuriser les appels à la base et d'éviter les failles par Injection
  - La création d'une requête préparée se fait grâce à la méthode **prepare**
- **Les paramètres mystères**
  - C'est la première façon de définir une requête préparée. Les paramètres sont indiqués par des **?** dans la chaine de la requête.

```php
<?php
    $prepa = $connection->prepare('DELETE FROM students WHERE firstname=? AND lastname=?;');
    $nbDelete = $prepra->execute(array('Simon','Radier'));
    echo $nbDelete.' enregistrements a été supprimé.';
    $nbDelete = $prepra->execute(array('Toto','Titi'));
    echo $nbDelete.' enregistrements a été supprimé.';
?>
```

Bases de données

Les requêtes préparées 2/3

- **Les paramètres nommés**
  - C'est la seconde méthode, plus claire, elle permet de définir les paramètres par une **chaine précédée de :**

```php
<?php
    $prepa = $connection->prepare(SELECT FROM students WHERE firstname like :search OR lastname LIKE :search;');
    if(!$prepra->execute(array('search'=>'Totor')))
        // error
    while( $student = $prepa->fetch(PDO::FETCH_OBJ) )
    {
        // Affichage du nom et prenom
        echo $student->lastname.':'.$student->firstname.'<br />';
    }
?>
```

  - PDO gère automatique les chaines pour éviter les injections

Bases de données

Les requêtes préparées 3/3

- **Les paramètres bindés**
  - C'est la dernière méthode pour faire des requêtes préparés, c'est une évolution des paramètres nommés.
  - Le binding permet de binder une variable au paramètre de la requête
  - Attention! Si la variable est modifiée avant l'appel à la requête, le paramètre sera modifié aussi.

```php
<?php
    $firstname = 'Alexandre';
    $lastName = 'LeGrand';
    $insert = $connection->prepare(INSERT INTO students VALUE(NULL,:firstname,:lastname');
    $insert->bindParam(':firstname',$firstname, PDO::PARAM_STR, 20);
    $insert->bindParam(':lastname',$lastname, PDO::PARAM_STR, 15;
    if ($insert->execute()) // Alexandre LeGrand a été inséré dans la table students
       echo 'Enregistrement réussi';
    $firstname = 'Toto';
    if ($insert->execute()) // Toto LeGrand a été inséré dans la table students
       echo 'Enregistrement réussi';
?>
```

Bases de données

Les exceptions

- **PDO étant une implémentation POO, ses méthodes lèvent des exceptions.**
  - Seule les exceptions de connections sont lancées initialement, il faut activer les exceptions sur les requêtes.
  - Il faut ajouter des options lors de la création de l'objet PDO

```php
<?php
    $dsn = 'mysql:host=localhost;dbname=mti_db';
    $user = 'student_s';
    $password = '42';
    $option = array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION);
    $connection = new PDO($dns, $user, $password, $option);
?>
```

A partir de maintenant, toute requête devrait être dans un bloc try…catch

XML is a classic political compromise: it balances the needs of man and machine by being equally unreadable to both.

Utilisateur de Redit

XML

XML

Simple XML (DOM) 1/2

- **Simple XML est un parseur DOM natif à PHP5**
  - Voici un exemple de fichier XML

```
<students>
    <student id="1">
        <firstname>Simon</firstname>
        <lastname>Radier</lastname>
    </student>
    <student id="2">
        <firstname>Martin</firstname>
        <lastname>Durand</lastname>
    </student>
</students>
```

- **Ouvrir un fichier XML**
  - La fonction simplexml_load_file() permet de charger un fichier xml.
    - Elle retourne un objet de type SimpleXMLElement

```
<?php
    $studentsXML = simplexml_load_file('students.xml');
?>
```

XML

Simple XML (DOM) 2/2

- **Accéder à un élément**

```php
<?php
    $xml= simplexml_load_file('students.xml');
    echo $xml->student[0]->firstname; // Affiche Simon
?>
```

- **Accéder à plusieurs éléments**

```php
<?php
    $xml = simplexml_load_file('students.xml');
    foreach($xml->student as $student)
    {
        echo $student->lastname;
    } // Affiche Radier puis Durand
?>
```

- **Accéder aux attributs**

```php
<?php
    $xml = simplexml_load_file('students.xml');
    echo $xml->student[0]['id']; // Affiche 1
?>
```

Ces éléments sont accessibles aussi bien en lecture qu'en écriture!

XML

XPath 1/2

- **Présentation**
  - XPath est un langage permettant de localiser une portion de document XML.
  - XPath est devenu un standard de facto
- **SimpleXML supporte les accès en notation XPath**

```php
<?php
    $xml = simplexml_load_file('students.xml');

    foreach($xml->xpath('students') as $student)
    {
        echo $student->lastname;
    } // Affiche Radier puis Durand
?>
```

  - Besoin d'aide sur XPath? => http://www.w3schools.com/xpath/xpath_syntax.asp

Multimedia
Technologies de l'Information

Exercice

Exercice ~~500~~ 420px : 1/2

Recréer le site d'upload d'images 500px:

- Authentification : 4 points
  - Création des tables SQL (1 utilisateur peut avoir plusieurs images) : 1 point
  - Création de compte : 2 points
  - Identification pour l'accès : 1 point
- Gestion des images : 8 points
  - Formulaire d'ajout (mode authentifié) : 1 point
    - Chaque image doit être redimensionnée en 420x420 : 1 point
  - Suppression d'une image (mode authentifié, en base et sur le serveur) : 2 points
  - Voir les images d'un utilisateur (accès sans authentification)
    - Mode affichage normal : 2 points
    - Mode en flux RSS (XML) : 2 points
- Filtres de modification sur les images : 4 points
  - + ou - contraste : 0.5 point
  - + ou – de luminosité : 0.5 point
  - Sépia : 0.5 point
  - Niveau de gris : 0.5 point
  - Filtre de gauss : 1 point
  - Filtre de détection des contours : 1 point
- Recherche d'une photo par couleur (trouver les couleurs dominantes d'une image) : 2 points

Contraintes

- Le code doit respecter les préconisations suivantes
  - Utilisation de la POO
  - Gestion des erreurs via les Exceptions
  - Code clair et respect des bonnes pratiques

Exercice

420px : 2/2

- **A la fin du cours**
  - Envoyer un mail avec la bibliothèque d'image que vous comptez utiliser
- **Premier rendu**
  - Authentification et gestion des images : 4/05
    - Une note intermédiaire sur 12
- **Choix entre deux passages**
  - 11/05 : Notation sur 20 pour le site complet
  - 18/05 : Notation sur 42 (sur 20)
    - Le notation des éléments attendus est ramenée à **10 points**
    - 3 types de bonus possibles (non exhaustif)
      - Bonus simple : 1 point
        - Nouveau filtre (non similaire à ceux demandés)
      - Bonus moyen : 3 points
        - Export de l'ensemble des images au format ZIP
        - Amélioration de l'affichage (mise en place d'une CSS + design sympa)
      - Bonus complexe : 5 points
        - Création d'un GIF animé affichant l'ensemble des images d'un utilisateur
        - Mise en place d'une API REST pour la gestion des images

Le choix de votre passage doit être envoyé le 04/05 avec vos propositions de bonus (si vous comptez passer le 18 mai)