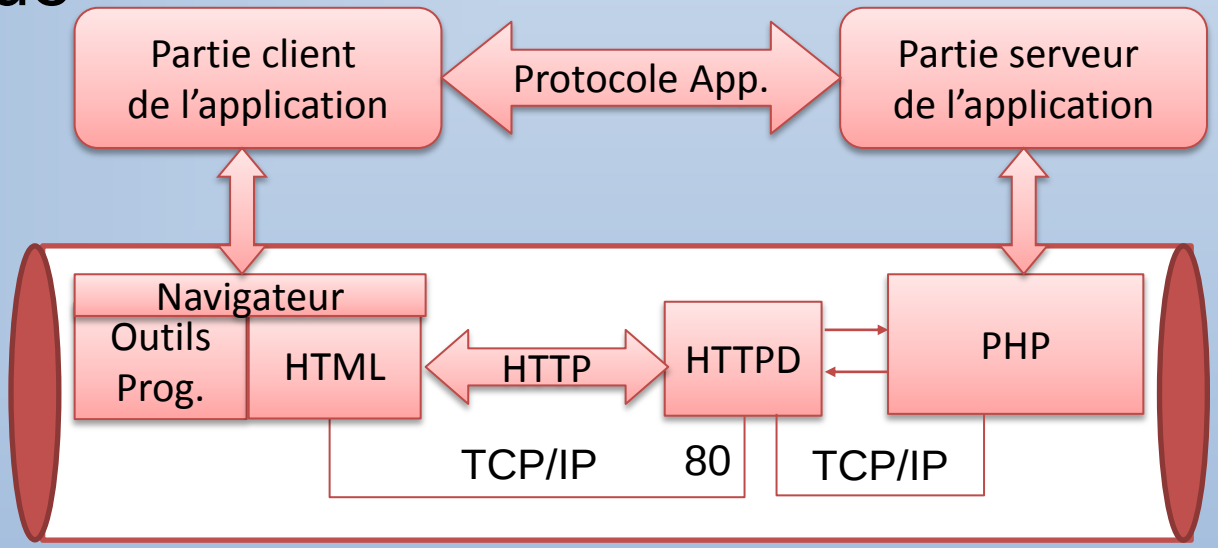
Programmation Web Avancée
php

Sur 3-4 séances
Objet, BD, Fichier, XML
« Modules »

Séance 1 : Objet, BD

Principes

- langage : PHP (html preprocessor)
- un pré processeur de HTTP
- typage dynamique
- orienté objet
- créé par Rasmus Lerdorf en 1994
- 5e version
- http://www.php.net/



Intégration du php dans les pages web

- intégration parfaite dans document HTML :
  - extension du fichier : .php (configuration du serveur web)
  - « balises de programmation »
    <?PHP
    // code php
    ?>
  - Génération de HTML retournée par echo (ou print)
- Méthode
  - façon « côté obscur » : echo (au fur et à mesure)
  - façon « côté lumineux » : calcul, mise en forme puis echo

Syntaxe

- Entre le C/Java et le Shell Script
- « ; » comme délimiteur
- commentaire
  - comme en java, entre les signes /* et */ ;
  - comme en java, en commençant une ligne par // :
  - comme en *shell* Unix, avec # .

Une exécution côté **SERVEUR**

- Le script PHP est **toujours**
  - stocké et exécuté sur le serveur web par l'interpréteur de PHP
- Sur demande d'un document (.php)
  - Le serveur identifie que c'est un fichier php (extension)
  - Recherche les « balises » <?php … ?>
  - Lance l'interpréteur pour ces balises
  - Retourne le document généré (réponse HTTP)

version côté obscur

```
<HTML> <HEAD>
<TITLE>HTML avec PHP</TITLE>
</HEAD>
<BODY>
<H1>HTML + PHP</H1>
<p>Nous sommes le <?php echo date
    ("j/m/Y"); ?></p>

</BODY></HTML>
```

```
<HTML> <HEAD>
<TITLE>HTML avec PHP</TITLE>
</HEAD>
<BODY>

<H1>HTML + PHP</H1>
<p>Nous sommes le 27/01/2011</p>

</BODY></HTML>
```

version côté lumineux

```
<?php
// calcul préalable
$date = "<p>Nous sommes le
     ".date("j/m/Y"). "</p>";
?>
<HTML> <HEAD>
<TITLE>HTML avec PHP</TITLE>
</HEAD>
<BODY>
<H1>HTML + PHP</H1>
<?php echo $date; ?>

</BODY></HTML>
```

```
<HTML> <HEAD>
<TITLE>HTML avec PHP</TITLE>
</HEAD>
<BODY>

<H1>HTML + PHP</H1>
<p>Nous sommes le 27/01/2011</p>

</BODY></HTML>
```

Éléments de Base de Php

Redirection, Session

Redirection

- Fonction **header**("Location: $url");
- $url : localisation de la page de redirection
  - Chemin local
  - Chemin absolu (http://...)
- Utilise un champ d'une réponse http
- **DONC A FAIRE AVANT D'EMETTRE UNE REPONSE**
  - i.e. avant d'être dans le document html

Notion de session

- Le support des sessions de PHP est un moyen de préserver des données entre plusieurs accès. Cela vous permet de créer des applications personnalisées.
- Chaque visiteur accédant à votre page web se voit assigner un identifiant unique, appelé "identifiant de session". Il peut être stocké soit dans un cookie, soit propagé dans l'URL.
- Lorsqu'un visiteur accède à votre site, PHP va vérifier sur demande explicite avec session_start( ) s'il existe une session du même nom. Si c'est le cas, l'environnement précédemment sauvé sera recréé.
- session_start( )
  - La gestion par défaut du numéro de session (identifiant) passe par les cookies…
  - donc session_start( ) doit être appelé avant toutes sorties

- Paramètre possible : n° de session
  - Passage dans l'Url (GET)

Session : $_SESSION

- $_SESSION : tableau contenant toutes les variables de session
- Affection = création ou mise à jour
  - $_SESSION["style"]="blue.css" : crée une variable de session « style » qui vaut « blue.css »
- Utilisation = utilisation de la variable
  - echo " <link …. href='{$_SESSION["style"]}'... ";
- Test d'existence : isset
  - if (isset($_SESSION["style"])) …
- Effacement : unset
  - unset(($_SESSION["style"])

Sessions sur un même serveur

- Foncton session_name
  - SANS PARAMETRE : permet de savoir le nom de la session courante
  - Avec une chaine de caractères (au moins une lettre) en paramètre ET AVANT session_start : permet de commencer une session spécifique
- Vous permet de faire des sessions distinctes sur www-mips…
- A mettre avant toutes ouvertures de session !
  - Sinon : une seule session pour tous les sites d'un serveur…

STRUCTURE DE CONTRÔLE

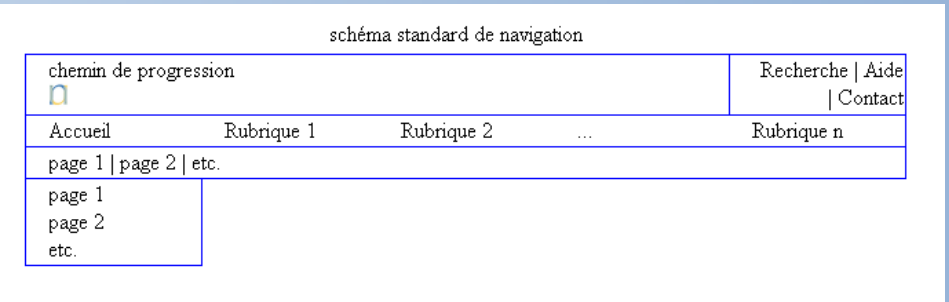
- require( ) et include( ) incluent et exécutent un fichier PHP.
  - La commande require() se remplace elle-même par le contenu du fichier spécifié
  - require( ) et include( ) sont identiques, sauf dans leur façon de gérer les erreurs. include( ) produite une Alerte (warning) tandis que require( ) génère une erreur fatale. Notamment lorsque le fichier manque.
- require_once( ) et include_once( )
  - La principale différence est qu'avec require_once( ), vous êtes assurés que ce code ne sera ajouté qu'une seule fois, évitant de ce fait les redéfinitions de variables ou de fonctions, génératrices d'alertes.

structures

- include, include_once, require, require_once
- ```
  <?php
   include("fichier.inc ");
   // suite…
   echo "bienvenu sur ma page";
  ?>
  ```
- Avec fichier.inc
  ```
  <?php
    $BD_serveur     = "localhost";
    $BD_utilisateur = "root";
  ?>
  ```
- Est équivalent à
  ```
   <?php
   // remplacement litéral
   $BD_serveur     = "localhost";
   $BD_utilisateur = "root";
   // suite…
   echo "bienvenu sur ma page";
  ?>
  ```
  ```

Mise en page et include

- Structure générale de page web : il est possible de trouver une partie des zones suivantes :
  - Entêtes en haut
  - Barre de navigation (menu) à gauche
  - Informations contextuelles à droite
  - Zone centrale pour la page
  - Pieds de page en bas
- Chacune des zones (si utilisées) correspondent à un fichier inclus
- exemple
  - Source : toutestfacile.com



Fonctions et Objets en Php

FONCTIONS

```
function NomFonction ([$arg1, $arg2, ...])
{
// Ici le code de la fonction
}
```

- arguments sont passés généralement *par valeur*
- par adresse : en préfixant la variable par & (avant le $). Il y a deux manières d'indiquer un passage par adresse :
  - au moment de l'appel, même pour une fonction qui a été prévue pour travailler sur des arguments passés par valeur (cette manière est déconseillée)
  - dans la définition de la fonction. C'est alors la règle par défaut pour tout appel de la fonction.

FONCTIONS

- Valeurs par défaut

```
function Connexion ($pNom, $pMotPass, $pBase = "clubvideo", $pServeur = "toto")
{  // Ici le code de la fonction }
// on peut donc appeler cette fonction sans citer les deux derniers arguments
$connexion1 = Connexion ("dupond", "passdupond");
$connexion2 = Connexion ("dupont", "passdupont", "Films");
```

- func_num_args( ) [pas de paramètre, retourne le nombre d'arguments passés à la fonction],
- func_get_arg( ) [un paremètre i : retourne le (i+1)ième élément de la liste des arguments]
- et func_get_args( ) [pas de paramètre, retourne les arguments d'une fonction sous forme de tableau]
- PHP propose trois types de variables :
  - Variables automatiques (pas de portée à l'extérieur)
  - Variables statiques (persistantes entre les appels)
  - Variables globales

Classes et Objets

- Une classe est une collection de variables et de fonctions qui fonctionnent avec ces variables. Une classe est définie en utilisant la syntaxe suivante :

```
class Panier {
        // Eléments de notre panier
        private $items;

        // Ajout de $num articles de type $artnr au panier
        function add_item ($artnr, $num) {
           $this->items[$artnr] += $num;
        }

        // Suppression de $num articles du type $artnr du panier
        function remove_item($artnr, $num) {
           if ($this->items[$artnr] > $num) {
              $this->items[$artnr] -= $num;
              return true;
           } else if ($this->items[$artnr] == $num) {
              unset($this->items[$artnr]);
              return true;
           } else {
              return false;
           }
        }
     }
```

Héritage et Constructeur

Fonction __construct (pour les versions résentes, function avec nom de classe)

```
<?php
class  LecteurTable {
public function __construct($s="euterpe.unice.fr", $l="visiteur", $p="toctoc", $bd="ihm") {
            // …
    }
}

class  UserManagement extends LecteurTable {
public function __construct() {
            parent::__construct(); // appel au constructeur de la classe parent
            // ….
    }
}
?>
```

initialisation et création

- Initialisation
  - Pour PHP 5 : utilisez les constructeurs pour les initialisations variables
  - function __construct
  - Ou function au nom de la classe
- Création
  - $cart = new Panier();
  - Le constructeur est la fonction qui est appelée automatiquement par la classe lorsque vous créez une nouvelle instance d'une classe a l'aide de l'opérateur *new*. La fonction constructeur a le même nom que la classe. Une fonction devient le constructeur si elle porte le même nom que la classe. Un constructeur peut avoir des paramètres avec ou sans des valeurs par défaut
- Accès aux membres d'une classe
  - $cart->add_item("diplome", 1);
    $cart->add_item("semestre", 6);
    print_r($cart->items);

Quelques mots-clefs

- this
  - La pseudo-variable *$this* n'est pas toujours définie si la méthode dans laquelle elle est présente est appelée statiquement. Cependant, ceci n'est pas une règle stricte : *$this* est définie si une méthode est appelée statiquement depuis un autre objet. Dans ce cas, sa valeur vaut l'objet qui fait l'appel static.
- extends
  - Une classe ne peut hériter que d'une seule autre classe
- implements
  - Une classe peut implémenter plusieurs interfaces
- ::
  - faire référence aux fonctions et variables d'une classe de base, ou bien d'utiliser des méthodes de classes qui n'ont pas encore d'objets créés
- parent
  - représente votre classe de base (celle indiqué par *extends*, dans la déclaration de votre classe)

TYPES : les chaînes de caractères

- **Traitement des variables dans les chaînes : cas des tableaux et des objets**

```
<?php
// Ces exemples sont spécifiques à l'utilisation de tableaux dans une chaîne  Lorsque vous êtes hors d'une chaîne,
// utilisez toujours des guillemets autour des index de tableau, et n'utilisez pas d'{accolades}.

// Affichons toutes les erreurs
error_reporting(E_ALL);

$fruits = array('fraise' => 'rouge', 'banane' => 'jaune');

// Fonctionne mais notez que cela fonctionne pas comme si cela était hors d'une chaîne
echo "Une banane est $fruits[banane].";

// Fonctionne
echo "Une banane est {$fruits['banane']}.";

// Fonctionne mais PHP cherche une constante appelée banane tel que décrit ci-dessous
echo "Une banane est {$fruits[banane]}.";

// Ne fonctionne pas, il manque les accolades. Cela donne une erreur d'analyse
echo "Une banane est  $fruits['banane'].";

// Fonctionne
echo "Une banane est " . $fruits['banane'] . ".";

// Fonctionne
echo "Ce carré a un coté de $square->width mètres de large.";

// Ne fonctionne pas. width00 pourrait être un champ.
echo "Ce carré a un coté de $square->width00 centimètres.";

// Fonctionne
echo "Ce carré a un coté de {$square->width}00 centimètres.";
?>
```

Fonctions liées aux objets

- is_a -- Retourne TRUE si un objet a pour parent une classe donnée

- class_exists -- Vérifie qu'une classe a été définie
- method_exists -- Vérifie que la méthode existe pour une classe

- De l'introspection…
  - get_class_methods -- Retourne les noms des méthodes d'une classe
  - get_class_vars -- Retourne les valeurs par défaut des propriétés d'une classe
  - get_class -- Retourne la classe d'un objet
  - get_declared_classes -- Liste toutes les classes définies
  - get_declared_interfaces -- Retourne un tableau avec toutes les interfaces déclarées
  - get_object_vars -- Retourne un tableau associatif des propriétés d'un objet
  - get_parent_class -- Retourne le nom de la classe d'un objet
  - is_subclass_of -- Détermine si un objet est une sous-classe
- *nterface_exists -- Vérifie si une interface a été définie*
- *property_exists -- Vérfie si un objet ou une classe a une propriété*

Typage de paramètre dans les fonctions

- Possible avec des interfaces ou des classes
- Exemple :
  - http://deptinfo.unice.fr/~renevier/calendrier/index2.php
  - Une interface :

```
<?php
interface CaseGenerateur {
   public function genereCase(Calendrier $cal);
   public function initialiseCases($auj) ; }
?>
```

  - Une classe : Calendrier…

Eléments **static**

Elements (champs ou méthodes) commun à toutes les instances de la classe
Accessibles sans instance
Accès via <nom de classe>::$<nom de champ> ou <nom de classe>:: <methode>( )

```
<?php
class Bandeau {

public static  $map = array(
                array("index.php", "connexion"),
                array("date.php", "manipulation de dates")
      );
}

echo "<a href='".Bandeau::$map[0][0]."'>='".Bandeau::$map[0][1]. " </a>" ;
?>
```

=> connexion

Chargement dynamique

Nécessaire en cas de désérialisation (session)

```
function __autoload($class_name) {
    include $class_name . '.inc'; // ou php
}
```

Utilisation du Php

connexion à un SGBD

Connexion à un SGBD

- La connexion se fait en trois temps :
  - La connexion
  - La sélection de la base de donnée
  - Exploitation de la bd
    - Selon les droits
    - Selon les besoins
- Support de plusieurs SGBD par php (mysql, postgres, etc.)
- Les fonctions varient selon la SGBD
  - Les noms
  - Certaines capacités particulières (rétro-conception)
  - Mais les principes restes…

Connexion à une base de donnée

- $connexion = mysql_connect('euterpe.unice.fr', 'visiteur', 'toctoc');
  - Connexion à un serveur mysql nommé euterpe.unice.fr
  - En tant que l'utilisateur nommé visiteur
  - Avec le mot de passe toctoc
- Si la connexion échoue, $connexion vaut « faux »
  - fonction exit() ou die() pour mettre fin au script (test ou "or die()")
  - Ou redirection
- La connexion au serveur sera fermée
  - À la fin du script
  - Par appel explicite à mysql_close($connexion).
- Suppression des messages d'erreur en cas d'échec avec @.

Sélection de la BD et requête SQL

- $bd = @mysql_select_db(test', $connexion);
  - Choix de la base de donnée nommée demo au travers de la connexion $connexion
  - Le second paramètre peut être omis, mais c'est la dernière connexion ouverte avec mysql_connect (si elle existe) qui est prise en compte
- $requete = mysql_query("select * from magasin;", $connexion);
  - Mysql_query permet de faire n'importe quel requête SQL
  - ici : sélection de tout ce qui est dans la table choisie
  - ATTENTION : $bd n'apparaît pas, mais IL FAUT faire le choix de BD
  - Le second paramètre peut être omis, mais c'est la dernière connexion ouverte avec mysql_connect (si elle existe) qui est prise en compte

Valeur de retour de mysql_query

- Pour les requêtes du type SELECT, SHOW, DESCRIBE ou EXPLAIN, mysql_query( ) retournera une ressource en cas de succès, ou FALSE en cas d'erreur.
- Pour les autres types de requêtes, UPDATE, DELETE, DROP, etc., mysql_query( ) retourne TRUE en cas de succès ou FALSE en cas d'erreur.
- mysql_query( ) échouera et retournera FALSE si l'utilisateur n'a pas les autorisations nécessaire pour accéder à la (aux) table(s) référencée(s) par la requête.
- Accéder aux erreurs
  - Les erreurs retournées par le serveur MySQL ne génèrent plus de message d'alerte.
  - mysql_error($connexion) retourne le message d'erreur généré par la dernière commande MySQL. Notez que cette fonction ne retourne que le texte de l'erreur la plus récente (n'incluant pas mysql_error() et mysql_errno())
  - mysql_errno($connexion) retourne le numéro d'erreur de la dernière commande MySQL.

Exploitation de la valeur de retour

- Utilisez **mysql_num_rows( )** pour trouver le nombre de lignes retournées pour une requête du type SELECT ou mysql_affected_rows( ) pour trouver le nombre de lignes affectées par les requêtes du type DELETE, INSERT, REPLACE, ou UPDATE.
- La ressource de résultat (cas d'un SELECT) retournée doit être passée par l'une des fonctions permettant d'explorer le résultat des tables, pour accéder aux données retournées.
  - **mysql_fetch_array($requete)** : retourne un tableau qui contient la ligne demandée et déplace le pointeur de données interne d'un cran ou FALSE s'il n'y a plus de lignes.
    - Le type de tableau retourné dépend d'un second paramètre facultatif paramètre result_type.
      - **MYSQL_BOTH (défaut), vous récupèrerez un tableau contenant des indices associatifs et numériques.**
      - **LES CLEFS SONT LES NOMS DES CHAMPS DANS LA TABLE DE LA BASE DE DONNEES**
      - En utilisant MYSQL_ASSOC, vous ne récupèrerez que les indices associatifs,
      - en utilisant MYSQL_NUM, vous ne récupèrerez que les indices numériques (indice = n° de colonne du champ dans la table)

Exploitation de la valeur de retour

- La ressource de résultat (cas d'un SELECT) retournée doit être passée par l'une des fonctions permettant d'explorer le résultat des tables, pour accéder aux données retournées.
  - Les noms des champs retournés par cette fonction sont sensibles à la casse.
- Tant qu'il y a des réponses, il faut les traiter :

```
while ($ligne = mysql_fetch_array( $requete ) )
    {
    // utilisation du tableau $ligne
    }
```

- Manipulation du pointeur interne à $requete :
  mysql_data_seek ($requete , $i)
  - i compris entre 0 et mysql_num_rows() – 1
  - Précise la ligne du prochain résultat (mysql_fetch_row, etc.)
  - Permet de remonter au début une fois la réponse exploitée

Des méta-informations

- mysql_num_fields($requete)
  - retourne le nombre de champs d'un jeu de résultat en cas de succès, ou FALSE si une erreur survient.
- mysql_num_rows($requete)
  - retourne le nombre de lignes dans un jeu de résultats en cas de succès, ou FALSE si une erreur survient.
- mysql_fetch_field($requete)
  - retourne un objet contenant les informations sur les champs. Cette fonction peut être utilisée pour obtenir des informations sur les champs de la requête fournie.
    - name - nom de la colonne
    - not_null - 1 si la colonne ne peut pas être NULL
    - primary_key - 1 si la colonne est une clé primaire
    - multiple_key - 1 si la colonne est une clé non unique
    - numeric - 1 si la colonne est numérique
    - type - le type de la colonne
    - unsigned - 1 si la colonne est non signée
    - zerofill - 1 si la colonne est complétée par des zéro
    - etc