

# Internet et bases de données

Clara Bertolissi

SIL-NTI

Université de Provence

# Plan

- Les fonctions dynamiques
- Les expressions régulières
- Les en-têtes html
- Les Urls
- Les mails
- Les librairies php
- Les extensions php

# Register globals

- Dans php.ini on a un paramètre `register_globals` qui permet de manipuler les variables dites globales.
- Les variables passées par le biais des formulaires, cookies, urls, sessions sont alors directement accessibles par le script qui les reçoit.
- Problèmes : on peut pas avoir deux variables avec le même nom.

# Super-global arrays

- Le paramètre `register_globals` est à off dans `php.ini`
- Les variables ne sont plus immédiatement importées, mais sont stockées dans le tableau globale correspondant:  
`$_GET`, `$_POST`, `$_SESSIONS`, `$_FILES`,  
`$_COOKIE`
- Diminue le risque de manipulation des variables par un utilisateur mal intentionné

# Retour sur les fonctions

Il est possible de créer dynamiquement des fonctions.  
Pour les déclarer, on affecte à une variable chaîne de caractères le nom de la fonction à dupliquer.  
Puis on passe en argument à cette variable les paramètres de la fonction de départ.

*Exemple :*

```
function divers($toto) {  
    echo $toto;  
}  
$mafonction = "divers";  
$mafonction("bonjour !");
```

```
// affiche 'bonjour !' par  
// appel de divers()
```

# Les fonctions

Quelques fonctions permettant de travailler sur des fonctions utilisateur:  
**call\_user\_func\_array**, **call\_user\_func**, **create\_function**,  
**func\_get\_arg**, **func\_get\_args**, **func\_num\_args**, **function\_exists**,  
**get\_defined\_functions**, ...

**call\_user\_func\_array**(\$str [, \$tab]) : Appelle une fonction utilisateur  
\$str avec les paramètres rassemblés dans un tableau \$tab.

*Exemple :*

```
function essai($user, $pass) {           // fonction à appeler
    echo "USER: $user PASSWORD: $pass";
}
$params = array("hugo", "0478");        // création du tableau de
    paramètres
call_user_func_array("essai", $params); // appel de la fonction
```

Le nom de la fonction à appeler doit être dans une chaîne de caractères.

# Les fonctions

**call\_user\_func**(\$str [, \$param1, \$param2, ...]) : Appelle une fonction utilisateur éventuellement avec des paramètres.

*Exemple :*

```
function essai($user, $pass) {  
    echo "USER: $user PASSWORD: $pass";  
}  
call_user_func("essai", "hugo", "0478");
```

Similaire à **call\_user\_func\_array** à la différence qu'ici les arguments à envoyer à la fonction à appeler ne sont pas dans un tableau mais envoyés directement en paramètre à **call\_user\_func**.

# Les fonctions

- **create\_function(\$params,\$code)** : Crée une fonction anonyme.
- Prend en argument la liste **\$params** des arguments ainsi que le code **\$code** de la fonction sous la forme de chaînes de caractères.
- Utiliser les simples quotes afin de protéger les noms de variables **'\$var'**, ou alors échapper ces noms de variables **\\$var**.
- Le nom de la fonction créée sera de la forme : *lambda\_x* où x est l'ordre de création de la fonction.

*Exemple :*

```
$newfunc = create_function('$a,$b','return \$a+\$b;');  
echo "Nouvelle fonction anonyme : $newfunc <br />";  
echo $newfunc(5,12)."<br />";
```

- Cet exemple est équivalent à : **function lambda\_1(\$a,\$b) { return \$a+\$b; }**



# Les fonctions

**func\_num\_args()** : Retourne le nombre d'arguments passés à la fonction en cours.

**func\_get\_arg(\$nbr)** : Retourne un élément de la liste des arguments envoyés à une fonction. L'indice \$nbr commence à zéro et renvoie le \$nbr-1 ème paramètre.

*Exemple :*

```
function foobar() {  
    $numargs = func_num_args();  
    echo "Nombre d'arguments: $numargs<br />";  
    if ($numargs >= 2) {  
        echo "Le 2ème param : ".func_get_arg(1)."<br />";  
    }  
}  
foobar("foo", 'bar', 10.5);
```

Cet exemple affiche la chaîne : 'Le 2ème param : bar'

# Les fonctions

**func\_get\_args()** : Retourne les arguments de la fonction en cours sous la forme d'un tableau.

*Exemple :*

```
function somme() {  
    $params = func_get_args();  
    $nbr = 0;  
    foreach($params as $elem) {  
        $nbr += $elem;  
    }  
    return $nbr;  
}  
echo somme(50,20,3,98,50);
```

Cette fonction **somme** retourne la somme de tous ses arguments quel qu'en soit le nombre.

# Les fonctions

- **function\_exists(\$str)** : Retourne TRUE si la fonction **\$str** a déjà été définie.
- **get\_defined\_functions()** : Retourne un tableau associatif contenant la liste de toutes les fonctions définies. A la clé **"internal"** est associé un tableau ayant pour éléments les noms des fonctions internes à PHP. A la clé **"user"**, ce sont les fonctions utilisateurs.

# Expressions régulières

Les expressions régulières permettent la recherche de motifs dans une chaîne de caractères.

*Fonctions :*

- **ereg**(\$motif, \$str) : teste l'existence du motif \$motif dans la chaîne \$str
- **ereg\_replace**(\$motif, \$newstr, \$str) : remplace les occurrences de \$motif dans \$str par la chaîne \$newstr
- **split**(\$motif, \$str) : retourne un tableau des sous-chaînes de \$str délimitées par les occurrences de \$motif

Les fonctions **eregi**, **eregi\_replace** et **spliti** sont insensibles à la casse (c'est-à-dire ne différencient pas les majuscules et minuscules).

*Exemple :*

```
if (ereg("Paris", $adresse))  
    echo "Vous habitez Paris.";
```

# Expressions régulières

- Les motifs peuvent contenir des caractères spéciaux :
- `[abcdef]` : intervalle de caractères, teste si l'un d'eux est présent
- `[a-f]` : plage de caractères : teste la présence de tous les caractères minuscules entre 'a' et 'f'
- `[^0-9]` : exclusion des caractères de '0' à '9'
- `\^` : recherche du caractère '^' que l'on déspecialise par l'antislash \
- `.` : remplace un caractère
- `?` : rend facultatif le caractère qu'il précède
- `+` : indique que le caractère précédent peut apparaître une ou plusieurs fois
- `*` : pareil que + Mais le caractère précédent peut ne pas apparaître du tout
- `{i,j}` : retrouve une chaîne contenant entre au minimum i et au maximum j fois le motif qu'il précède
- `{i,}` : idem mais pas de limite maximum
- `{i}` : retrouve une séquence d'exactly i fois le motif qu'il précède
- `^` : le motif suivant doit apparaître en début de chaîne
- `$` : le motif suivant doit apparaître en fin de chaîne

# Expressions régulières

- *Exemples de motifs :*
- “[A-Z]” : recherche toutes les majuscules
- “[a-zA-Z]” : recherche toutes les lettres de l’alphabet minuscules ou majuscules
- “[^aeuyio]” : exclu les voyelles
- “^Le ” : toute chaîne commençant par le mot “Le “ suivi d’un espace
- “\$\.com” : toute chaîne se terminant par “.com” (déspecialise le point)
- *Exemples :*
- ```
if ( ereg("^.*@wanadoo\.fr", $email) ) {
```
- ```
    echo “Vous êtes chez Wanadoo de France Télécom.”;
```
- ```
}
```
- ```
$email = eregi_replace("@", "-nospam@", $email);
```
- Ce dernier exemple remplace “moi@ici.fr” en “moi-nospam@ici.fr”.

# Expressions régulières

- Il existe des séquences types :
- `[:alnum:]` : [A-Za-z0-9] – caractères alphanumériques
- `[:alpha:]` : [A-Za-z] – caractères alphabétiques
- `[:digit:]` : [0-9] – caractères numériques
- `[:blank:]` : espaces ou tabulation
- `[:xdigit:]` : [0-9a-fA-F] – caractères hexadécimaux
- `[:graph:]` : caractères affichables et imprimables
- `[:lower:]` : [a-z] – caractères minuscules
- `[:upper:]` : [A-Z] – caractères majuscules
- `[:punct:]` : caractères de ponctuation
- `[:space:]` : tout type d'espace

# Entêtes HTTP

- Il est possible d'envoyer des entêtes particuliers du protocole HTTP grâce à la commande **header**.
- Syntaxe : **header(\$str);**
- Exemples :  
**header("Content-type: image/gif");** // spécifie le type d'image gif  
**header("Location: ailleurs.php");** /\* redirection vers une autre page \*/  
**header("Last-Modified: ".date("D, d M Y H:i:s")." GMT");**
- Les entêtes doivent obligatoirement être envoyées avant l'affichage de tout caractère dans la page en cours. Car l'affichage force l'envoi des entêtes de base.
- **headers\_sent()** : Retourne TRUE si les entêtes ont déjà été envoyées, FALSE sinon.



# Entêtes HTTP

- Le rôle des entêtes est d'échanger des méta informations entre serveur et client à propos du document, de la connexion, etc.

Voici quelques entêtes HTTP :

- HTTP/1.0 301 Moved Permanently
- Date: Sun, 07 Apr 2002 14:39:29 GMT
- Server: Apache/1.3.9 (Unix) Debian/GNU
- Last-Modified: Sun, 07 Apr 2002 14:39:29 GMT
- Connection: keep-Alive
- Keep-Alive: timeout=15, max=100
- Content-type: text/html
- Content-length: 1078
- Transfert-Encoding: chunked
- Pragma: no-cache
- WWW-Authenticate: Basic realm="Domaine sécurisé"
- Location: home.html

# Entêtes HTTP

```
<?php  
header("Location: home2.php");  
exit();  
?>
```

Ce script effectue une redirection vers une autre page sans regarder la suite du script. La fonction **exit** est là pour parer au cas impossible où le script continuerait son exécution.

Note: en règle générale, le format d'un entête est le suivant

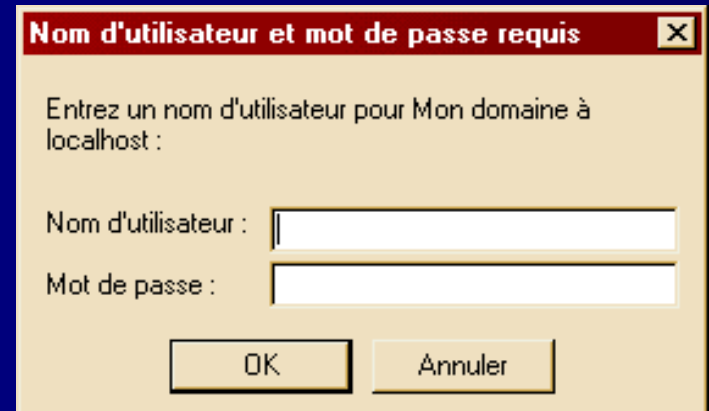
**Champ: valeur**

Avec un espace entre les deux points ':' et la '**valeur**'.

# Entêtes HTTP

*Exemple :*

```
<?php
if(!isset($PHP_AUTH_USER)) {
    header("WWW-Authenticate: Basic
    realm=\"Mon domaine\"");
    header("HTTP/1.0 401
    Unauthorized");
    echo "Echec de l'identification.";
    exit();
} else {
    echo "Bonjour
    $PHP_AUTH_USER.<br />";
}
?>
```



Nom d'utilisateur et mot de passe requis

Entrez un nom d'utilisateur pour Mon domaine à localhost :

Nom d'utilisateur :

Mot de passe :

OK Annuler

# Mail

- La fonction **mail** envoie un message électronique.
- Syntaxe :
- **mail(\$destinataire, \$subject, \$message[, \$headers, \$params]);**
- Exemple :
- **\$message = "Pour tout savoir sur le Php, visitez le site officiel";**
- **mail("vous@labas.fr", "Aide sur PHP", \$message);**
- Note: cette fonction ne marche que si un programme de messagerie électronique ("mailer") est préalablement installé sur le serveur.

# Mail

Quelques entêtes :

- **From:** Hugo Blanc <theboss@php-help.com>\n
- **X-Mailer:** PHP\n // mailleur
- **X-Priority:** 1\n // Message urgent!
- **X-Files:** Truth is out there\n // entête fantaisiste !
- **Return-Path:** <daemon@php-help.com>\n // @ retour pour erreurs
- **Content-Type:** text/html; charset=iso-8859-1\n // Type MIME
- **Cc:** archives@php-help.com\n // Champs CC
- **Bcc:** bill@php.net, tony@phpinfo.net\n // Champs BCC
- **Reply-To:** <hugo@php-help.com> // @ de retour
- Format général des entêtes :
- **Nom-Entete:** valeur\n

# Mail

*Exemple :*

```
<?php
$recipient = "Tony <tony@labas.com>, ";
$recipient .= "Peter <peter@pwet.net>";
$subject = "Notre rendez-vous";
$message = "Je vous propose le samedi 15 juin \n";
$message .= "--\r\n";           // Délimiteur de signature
$message .= "Hugo";
$headers = "From: Hugo Blanc <cyberzoide@multimania.com>\n";
$headers .= "Content-Type: text/html; charset=iso-8859-1\n" ;
$headers .= "Cc: bruno@ici.fr\n";
mail($recipient, $subject, $message, $headers);
?>
```

# URL

- `http://www.google.fr/?q=cours+php`
- `http://cyberzoide.developpez.com/php/php4_mysql.ppt`
- `ftp://foo:0478@ftp.download.net`
- Leur format spécifique leur interdit de comporter n'importe quel caractère (comme l'espace par exemple).
- Une URL est une chaîne de caractères composée uniquement de caractères alphanumériques incluant des lettres, des chiffres et les caractères : - (tirêt), \_ (souligné), . (point).
- Tous les autres caractères doivent être codés. On utilise le code suivant : `%xx`. Où % introduit le code qui le suit et `xx` est le numéro hexadécimal du caractère codé.

# URL

- Le passage de valeur d'un script à l'autre se fait soit par les formulaires, soit par les sessions, ou encore par l'URL.

*Exemple par l'URL :*

- `<a href="index.php?imprim=yes&user_id=75">Version imprimable</a>`
- Dans cet exemple on transmet deux variables au script **index.php**. Les valeurs sont des chaînes de caractères qui pourront être castées implicitement en entier.
- Le caractère **?** Indique que la suite de l'URL sont des paramètres et ne font pas partie du nom de fichier. Le caractère **=** sépare un nom de paramètre et sa valeur transmise. Le caractère **&** séparer deux paramètres.
- Pour faire face au cas général d'un paramètre dont la valeur contient des caractères interdits, on utilise les fonction de codage.



# URL : codage

Quelques fonctions de codage sur l'URL :

- *Codage de base :*
- **urlencode** : Encode une chaîne en URL.
- **urldecode** : Décode une chaîne encodée URL.
- *Codage complet :*
- **rawurlencode** : Encode une chaîne en URL, selon la RFC1738.
- **rawurldecode** : Décode une chaîne URL, selon la RFC1738.
- *Codage plus évolué :*
- **base64\_encode** : Encode une chaîne en MIME base64.
- **base64\_decode** : Décode une chaîne en MIME base64

# URL : codage

- **urlencode(\$str)** : code la chaîne \$str. Les espaces sont remplacés par des signes plus (+). Ce codage est celui qui est utilisé pour poster des informations dans les formulaires HTML. Le type MIME utilisé est *application/x-www-form-urlencoded*.

*Exemple 1 :*

```
echo <a href=\"$PHP_SELF?foo=\".urlencode($foo).\"\">Foo</a>;
```

- **rawurlencode(\$str)** : code la chaîne \$str. Remplace tous les caractères interdits par leur codage équivalent hexadécimal.

*Exemple 2 :*

```
echo <a href=\"$PHP_SELF?foo=\".rawurlencode($foo).\"\">Foo</a>;
```

- Pour être accessible, la valeur du paramètre devra par la suite être décodée dans le script d'arrivée par la fonction réciproque adéquate.

# URL : codage

- **base64\_encode(\$str)** : code la chaîne \$str en base 64.  
Cet encodage permet à des informations binaires d'être manipulées par les systèmes qui ne gèrent pas correctement les codes 8 bits (code ASCII 7 bit étendu aux accents européens).
- Une chaîne encodée en base 64 a une taille d'environ 33% supérieure à celle des données initiales.

*Exemple 3 :*

```
echo <a href=\"'$PHP_SELF?foo='.base64_encode($foo).'\">>Foo</a>";
```

- Comparatif des trois encodages :  
*Sans codage :* René & Cie : 30%-5\*20  
*urlencode :* Ren%E9+%26+Cie+%3A+30%25-5%2A20  
*rawurlencode :* Ren%E9%20%26%20Cie%20%3A%2030%25-5%2A20  
*base64\_encode :* UmVu6SAmIENpZSA6IDMwJS01KjIw

# URL

- **parse\_url(\$str)** : retourne un tableau associatif contenant les différents éléments de l'URL passée en paramètre. Les champs sont les suivants : "scheme" (protocol), "host" (domaine), "port" (n° de port), "user" (nom d'utilisateur ftp), "pass" (mot de passe ftp), "path" (chemin de la ressource), "query" (paramètres et valeurs).

*Exemple :*

```
$tab = parse_url("http://www.cia.gov:8080/form.php?var=val");
```

Cet exemple correspond au tableau suivant :

- |          |    |             |
|----------|----|-------------|
| ■ Scheme | => | http        |
| ■ host   | => | www.cia.gov |
| ■ port   | => | 8080        |
| ■ path   | => | form.php    |
| ■ Query  | => | var=val     |

# URL

- `parse_str($str)` : analyse la chaîne `$str` comme si c'était une URL et en extrait les variables et valeurs respectives qui seront alors connues dans la suite du script.
- Cette fonction évite d'avoir à créer ses propres fonctions d'analyse de champs de base de données où l'on aurait sauvegardé une url.

*Exemple :*

- `$str = "nom=jean+pierre&email[]=moi@ici.fr&email[]=moi@labas.com";`
- `parse_str($str);`
- `echo $nom, $email[0], $email[1];`

# Les librairies Php

- Librairie FPDF : pour générer des documents PDF
- Librairie graphique JPgraph : pour générer des graphes relativement complexes

# FPDF

Se compose :

- d'un fichier de classe `fpdf.php` (qu'il faudra inclure dans votre script)
- et des fichiers de définition des polices.
- La première étape consiste à créer un objet FPDF.

# Objet FPDF

Un objet dépend de trois paramètres :

- L'orientation de la page (P pour portrait ou L pour landscape)
- L'unité de mesure (mm, cm, in, pt)
- Format de la page (A3, A4, ...)

EXEMPLE : Création de l'objet et ajout d'une page

- `$pdf1 = new FPDF('P', 'mm', 'A4');`
- `$pdf = new FPDF();`
- `$pdf -> AddPage();`



# Choix de la police

Méthode `SetFont()` avec comme paramètres :

- Le type de police (Arial, Times, ...)
- Le style de police (normal, B gras, I italique, ...)
- La taille donnée en points

Exemple

- `$pdf->SetFont('Arial', 'B', 16)`

# L'impression

Méthode `Cell()` avec comme paramètres :

- La longueur et la largeur de la cellule
- Le texte
- L'encadrement (1 ou 0) (optionnel)
- Le retour à la ligne (1 ou 0) (optionnel)
- Le positionnement (C centre, L gauche, R droite) (optionnel)

Exemple

- `$pdf -> Cell(150, 10, 'mon texte', 1, 0, 'C')`

# Affichage

Methode **Output()**

- Sans paramètre : l'affichage se fait à l'écran
- Avec un nom de fichier en paramètre : sauvegarde du document dans le fichier

# Autres méthodes

- `AddLink()`, `SetLink()` : ancre interne au document
- `Image(nom, coord_x, coord_y, [largeur, hauteur, type, link])`
- `MultiCell(largeur, hauteur, text, [bordure, align, couleur] )`
- `PageNo()` : numero de la page courante
- `SetDrawColor(r,g,b)`, `SetFillColor(r,g,b)`, `SetTextColor(r,g,b)`

# Héritage

On peut définir un document plus complexe en utilisant le mécanisme d'héritage entre classes

Exemple :

```
Class Pdf extends FPDF {
```

```
Var $titre;
```

```
...
```

```
Function Header { }
```

```
Function Footer { }
```

```
...
```

```
}
```

# JPGraph

- Librairie orientée objet libre
- Permet de créer des graphiques complexes avec un minimum de code
- Elle s'appuie sur le module graphique de Php GD

# JPGraph

- Différents types de graphics : (diagrammes en lignes, barres, ronds) grâce aux librairies `jpgraph_line`, `jpgraph_bar`, `jpgraph_pie` ...
- Support de formats gif, jpeg, png.
- Les images générées seront incluses dans le document html dans une balise `<IMG src = "image.php"/>` ou `image.php` est le code generateur

# JPGraph

- Création de l'objet graphe avec sa longueur et hauteur

Exemple :

```
$graph = new Graph(300, 200)
```

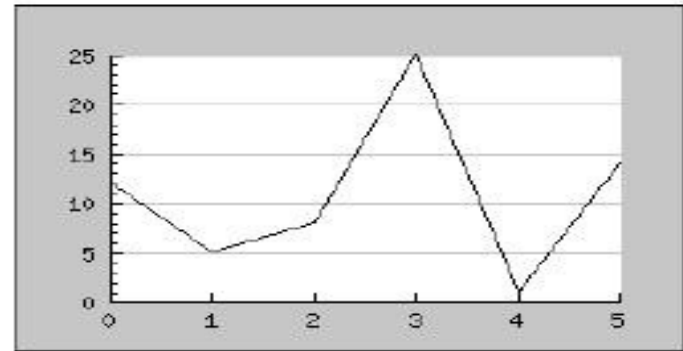
- Calcul du graphique
- Affichage avec la méthode Stroke()

Pour pouvoir l'afficher il faudra deux fichiers : le script générant le graphe et le fichier html (ou php) contenant la balise `<IMG... />`

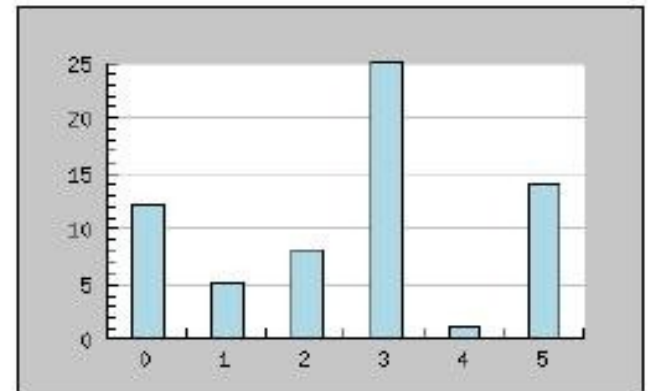


# Exemple

- Diagramme de lignes

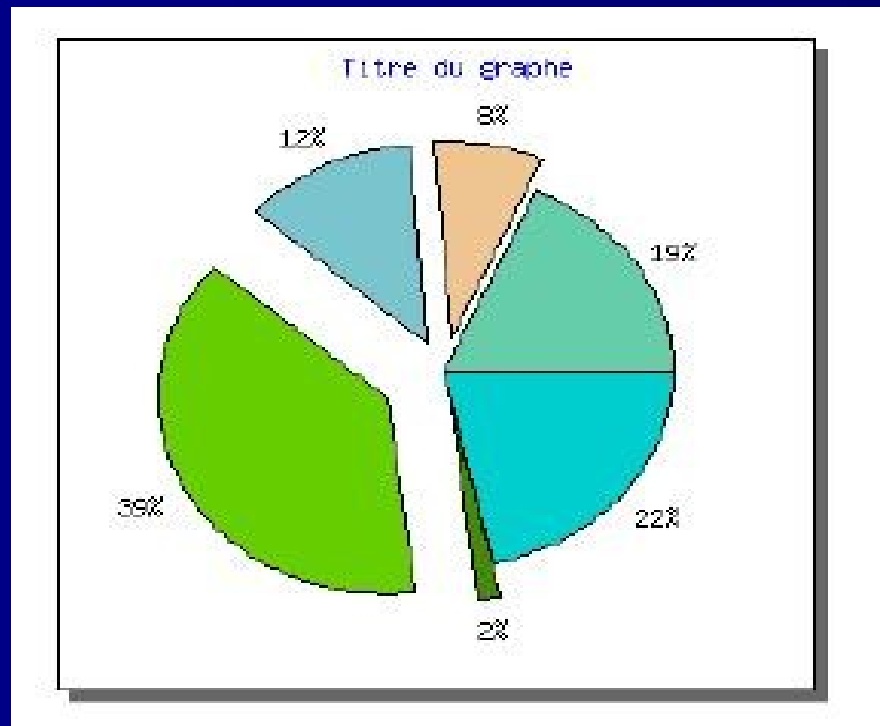


- Diagramme à barres



# Exemple

- Diagramme “à amembert”



# Les extensions PHP

- PEAR : Php Extension and Application Repository

<http://pear.php.net>

- PECL : Php Extension Community Library

<http://pecl.php.net>

# Pear

- Pear est composé d'un ensemble de packages en accès libre
- Les packages sont construits à partir des fonctions Php standard et souvent écrits en style orienté objets
- Ils peuvent être inclus dans les scripts Php avec `include()` ou `require()` comme n'importe quel autre fonction ou classe Php

# Pear

Exemple de packages :

- DB – connexion à une base de donnée
- HTTP – manipulation du protocole http
- Mail – interaction avec Pop, Imap et Smtplib
- Calendar – fonctions et objets concernant le calendrier
- Archive\_Zip – interaction avec des fichiers compactés
- ...

# PecI

- PecI est conceptuellement très similaire à Pear
- Il contient des extensions de Php écrites en C
- Plus rapides à l'exécution que les extensions contenues dans Pear.
- Les packages PecI peuvent être installés via le logiciel PEAR
- Des nouvelles extensions écrites par les utilisateurs peuvent être ajoutées à PECL.

# Aujourd'hui on a vu

- Les fonctions dynamiques
- Les expressions régulières
- Les en-têtes html
- Les mails
- Les urls
- Les librairies php : FPDF, JPDFGraph
- Les extensions php : PEAR, PECL