Programmation web côté serveur

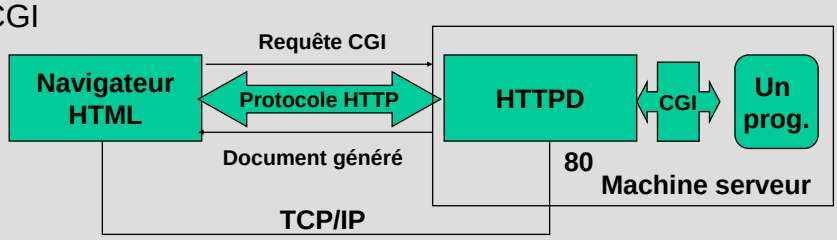
Nhan Le Thanh

Plan du cours
Programmation serveur

- MODELES CL/SV (VIA) WEB
  - MODELE AVEC CGI
  - MODELE AVEC PREPROCESSEURS
  - BASES DE DONNEES ET MODELE 3-TIERS
- PROGRAMMATION CGI
  - INTRODUCTION
  - COMMUNICATION CLIENT-SERVEUR
  - MISE EN ŒUVRE SUR SERVEUR
- ELEMENTS DU LANGAGES PHP
  - GENERALITES
  - VARIABLES, TYPES ET EXPRESSIONS
  - STRUCTURE DE CONTRÔLE
  - FONCTIONS
  - INTEGRATION AVEC HTML
- BASES DE DONNEES ET PHP
  - BASE DE DONNEES RELATIONNELLES
  - DEFINITION DE STRUCTURES ET DONNEES
  - MANIPULATION DE DONNEES
  - ACCES AU DONNEES DEPUIS D'UN PROGRAMME PHP
- PROGRAMMATION AVEC PHP
  - INTRODUCTION
  - PROGRAMMATION AVEC LES FONCTIONS
  - PROGRAMMATION AVEC LES OBJETS

PROGRAMMATION SERVEUR WEB
MODELES CL/SV via WEB

MODELE AVEC CGI : Notion de document dynamique avec la CGI

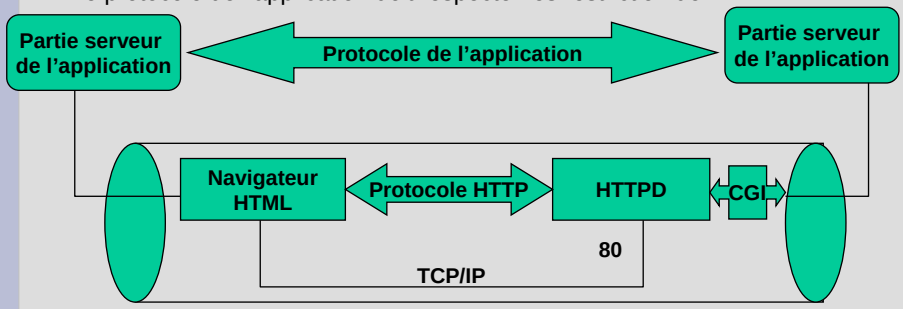


- Le serveur HTTP dispose une interface, dit CGI (Commun Gateway Interface), permettant d'invoquer depuis du client web, l'exécution d'un programme quelconque se trouvant sur la même machine serveur :
  - l'ordre d'exécution sera sous forme d'un hyperlien avec quelques conventions spécifiques
  - Le serveur HTTP reçoit cet ordre, il organisera ensuite l'exécution du programme demandé. La communication entre le programme et le serveur HTTPD est assurée par : les entrées/sorties standard (STDIN et STDOUT) et un ensemble de variables d'environnement du serveur HTTPD
  - Le résultat de l' exécution sera envoyé par le serveur HTTP au client web 'sous forme d'un document HTML généré depuis du prog. CGI)

PROGRAMMATION SERVEUR WEB
MODELES CL/SV via WEB

MODELE AVEC CGI (2) : Approche CL/SV par CGI :

- Utilisation l'interface CGI pour activer la partie Serveur de l'application
- Le programme Client est un script HTML (HyperText Markup Language)
- Le programme Serveur est un programme CGI
- Le protocole de l'application doit respecter les restriction de HTTP



PROGRAMMATION SERVEUR WEB
MODELES CL/SV WEB

- MODELE AVEC CGI (3): Passage de paramètres à un programme CGI :
  - Format de paramètre : en texte ASCII
    - chaque paramètre comprend 2 opérandes «Nom-de-variable » et «Valeur-de-variable » reliées par le symbole '='
    - les paramètres sont reliés par le symbole '&'
    - Convention : certaines règles de transformation automatique sont appliquées : le caractère d'espace (' ') est remplacé par '+', …

var1	=	val1	&	var2	=	val2	&	…	&	varn	=	valn

  - Modes de passage de paramètres :
    - GET : la chaîne de paramètres est envoyée avec l'URL après le caractère '?' et sera déposée dans une variable d'environnement, appelé QUERY_STRING du service HTTP (sur la machine serveur). Avantage : simple; Inconvénient : taille limitée à 200 caractères
    - POST : la chaîne de paramètre sera envoyée indépendamment de l'URL et dirigé vers le fichier STDIN (Standard INPUT) du programme CGI. Avantage : taille illimitée, traitement standard

PROGRAMMATION SERVEUR WEB
MODELES CL/SV WEB

- MODELE AVEC CGI (4) : Retour de données depuis d'un programme CGI au serveur HTTP et puis au Client web :
  - Format de données de retour : Texte HTML
  - Mode de passage du CGI vers le serveur HTTP : les données sorties du STDOUT (Standard OUTPUT) du programme CGI seront redirigées à l'entrée standard (stdin) du service HTTP qui les transmet au Client Web
  - Ce résultat peut être n'importe quel document multimédia, depuis le simple texte ascii jusqu'à la vidéo. Dans le cas où la requête d'un client se limite à demander au serveur de lui fournir un fichier, le serveur se base sur l'extension de ce fichier pour déterminer son type
  - Conformément au protocole HTTP, il faut alors transmettre ce type dans l'en-tête, avec la clause 'Content-type: *typeDocument*', pour que le navigateur sache comment décrypter les informations qui lui proviennent par la suite
  - Exemple : Pour un fichier HTML par exemple, l'extension est le plus souvent *.html*, et la valeur de *typeDocument* est 'text/html'

PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- MODELE AVEC CGI (5): Les variables d'environnement du HTTP :
  - Moyens de communication entre le Serveur http et le programme CGI : Le serveur, avant d'exécuter le programme CGI, initialise des variables d'environnement qui peuvent être récupérées par ce programme
  - Quand la méthode utilisée est GET, une de ces variables (QUERY STRING) contient la liste des paramètres issus du formulaire
  - Dans le cas d'une communication CGI, le serveur reste totalement neutre – il n'a pas d'extension de fichier sur laquelle s'appuyer – et transmet tel quel au client le résultat que lui fournit le programme CGI
  - Le programme CGI utilise les variables d'environnement du serveur HTTP pour connaître les caractéristiques du client web de la requête client ainsi que l'état de la transmission

PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- MODELE AVEC CGI (6) : Les variables d'environnement du HTTP :
  - REQUEST_METHOD : Méthode de transmission des paramètres (GET, POST, etc)
  - QUERY_STRING : Une chaîne de caractères contenant tous les paramètres de l'appel en cas de méthode GET. Cette chaîne doit être décodée, ce qui constitue l'aspect le plus fastidieux du traitement
  - CONTENT_LENGTH : Longueur de la chaîne transmise sur l'entrée standard, en cas de méthode POST
  - PATH_INFO : Informations sur les chemins d'accès menant par exemple vers des fichiers que l'on souhaite utiliser
  - HTTP_USER_AGENT : Type et version du navigateur utilisé par le client
  - REMOTE_ADDR : Adresse internet du client.
  - REMOTE_HOST : Nom de la machine du client.
  - REMOTE_USER Nom du client, pour les sites sécurisés avec htaccess
  - REMOTE_PASSWORD Mot de passe du client, pour les sites sécurisés

PROGRAMMATION SERVEUR WEB
MODELES CL/SV WEB

- MODELE AVEC CGI (7) : Limites de l'approche
  - Côté Client : absence des outils de programmation et les outils de présentation sont limités à la capacité du langage HTML
  - Côté Serveur : absence des outils de développement adaptés, la communication avec le serveur HTTP est à automatiser
  - Côté Protocole : les paramètres sont passés en mode texte avec un format imposé (absence de sécurité et capacité limitée) et deux modes de récupération fixes (GET et POST)

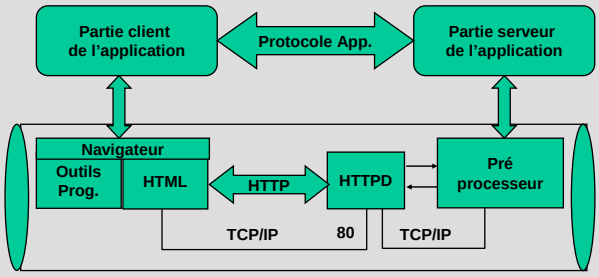
Renforcer les outils de programmation et de présentation du côté CLIENT

Automatiser le processus de communication Client/Serveur

Normaliser les outils de développement du côté SERVEUR

PROGRAMMATION SERVEUR WEB
MODELES CL/SV WEB

- MODELE AVEC PREPROCESSEUR de HTTP : Introduction
  - Tiers Client : Programme interface comprenant des outils de présentation et de programmation : HTML, Plug-in, JavaScript, Style, Applet Java
  - Tiers Serveur : Programme de services, appelé « objets de métiers », développé depuis un environnement de développement normalisé, dit pré processeur de HTTP, Par exemple : PHP, JSP, ASP



PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- MODELE AVEC PREPROCESSEUR (2) : principaux produits
  - ASP (ACTIVE SERVER PAGE) :
    - Caractéristiques :
      - Serveur intégré à l'environnement de IIS (httpd de Microsoft) permet de générer les documents HTML depuis d'un langage de programmation, appelé ASP, qui est une extension de syntaxe HTML avec des balises de programmation contenant les blocs de code Visual Basic
    - Avantage :
      - Environnement de programmation complet et intégré dans window
      - Bonne performance
      - Outils propriétaires bien suivis et fiables
    - Inconvénients :
      - Environnement fermé ne fonctionne qu'avec les système Windows

PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- MODELE AVEC PREPROCESSEUR (3): principaux produits
  - JSP (JAVA SERVER PAGE) :
    - Caractéristiques :
      - Un pré processeur logique de gestion des servlets (sous forme d'un service Internet) destinée à connecter au service HTTPD permettant de générer les documents HTML depuis d'un langage de programmation, appelé JSP, qui est une extension de syntaxe HTML avec des balises de programmation contenant les blocs de code JAVA
    - Avantage :
      - Environnement de programmation complet et intégré à tous les serveur HTTP (ouverture très large)
      - Très puissant et bonne performance
      - Outils propriétaires bien suivis et fiables (par SUN et APPACHE)
    - Inconvénients :
      - Complexe, exigence des connaissances de Java et de programmation à objets fortement typés

PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- MODELE AVEC PREPROCESSEUR (4) : principaux produits
  - PHP (PREPROCESSOR OF HTTP PROCESSOR) :
    - Caractéristiques :
      - Un pré processeur logique (sous forme d'un service) destinée à tous les service HTTPD permettant de générer les documents HTML depuis d'un langage de programmation, appelé PHP, qui est une extension de syntaxe HTML avec des balises de programmation contenant les blocs de code PHP
    - Avantage :
      - Environnement de programmation complet et intégré à tous les serveur HTTP (ouverture très large)
      - Puissance et performance acceptables
      - Langage PHP (typage dynamique, C simplifié) est simple à maîtriser
    - Inconvénients :
      - Produit libre, problème de maintenance et suivi
      - Problème de performance et de modulation

PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

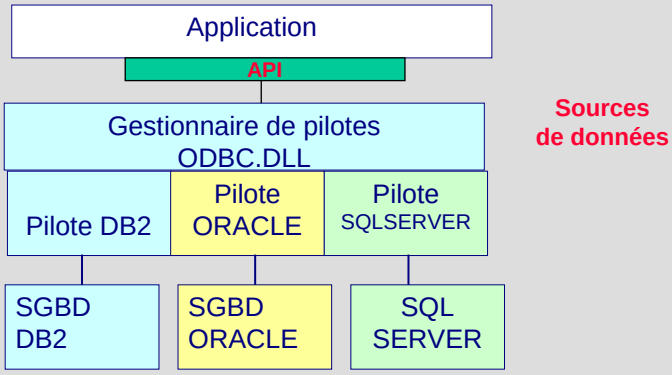
- BASES DE DONNEES ET MODELE 3-TIERS : Connexion au SGBD
  - Les SGBD jouent le rôle important dans les applications d'entreprise. L'intégration des possibilités de connexion aux SGBD dans le modèle de programmation CL/SV web est une nécessité
  - Deux manières de connexion :
    - Connexion par le biais d'un bibliothèque de procédure RPC (Remote Procedure Call) fournies par chaque constructeur de SGBD :
      - Avantage : performance, riche en fonctionnalités
      - Inconvénients : portabilité, complexe
    - Connexion via une plateforme intermédiaire (middleware) qui joue le rôle d'interface entre le programme et les SGBD
      - Avantage : portabilité, facile à programmer
      - Inconvénients : moins performance et fonctionnalités restant standard (CLI = Call Level Interface, la norme X/open définie par SAG (SQL Access Group)

PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- BASES DE DONNEES ET MODELES 3-TIERS(2) : Middlewares
  - ODBC (Open DataBase Connectivity)
    - Première implémentation du standard CLI réalisée par Microsoft
    - Accès normalisé à des SGBD relationnels différents (Oracle, DB2 ...)
    - Accès même à des pseudo-SGBD, ou des tableurs, ou encore des gestionnaires de fichiers
    - Interopérabilité avec des sources de données hétérogènes
    - Avec ODBC, il est possible de développer une application sans se soucier de la source de données qui sera utilisée en exploitation
    - API C (SDK ODBC) et classes C++ (MFC)
    - Utiliser pour les applications développées sous windows

PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- BASES DE DONNEES ET MODELES 3-TIERS(3) : Middlewares
  - ODBC (Open DataBase Connectivity)

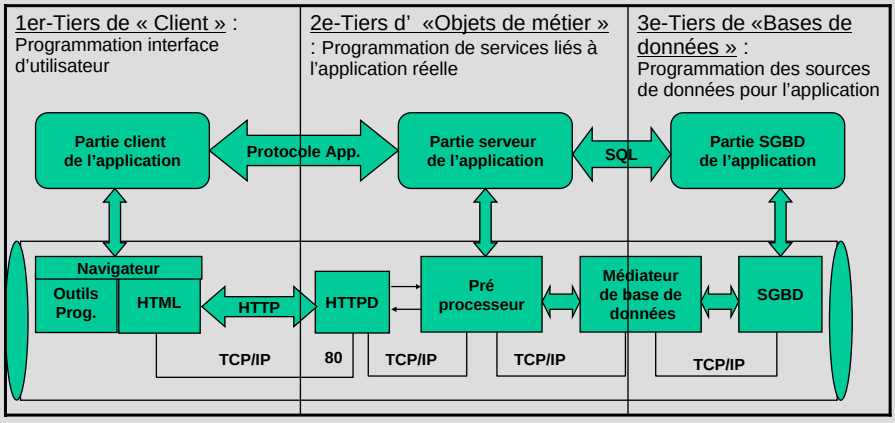


PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- BASES DE DONNEES ET MODELES 3-TIERS (4): Middlewares
  - JDBC (Java DataBase Connector)
    - Une plateforme logiciel permettant de connecter à un SGBD depuis d'un programme JAVA :
      - Un API en Java (standard CLI)
      - Des pilotes propriétaires pour accès aux SGBD
      - Un moteur de gestion des actions
    - Performante et sécurisée, mais spécifique pour le langage java

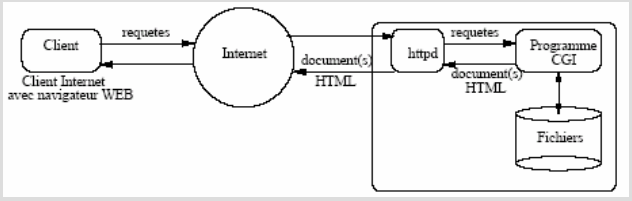
PROGRAMMATION SERVEUR WEB MODELES CL/SV WEB

- BASES DE DONNEES ET MODELES 3-TIERS (4): Architecture 3-tiers :



PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- INTRODUCTION
  - Le *Common Gateway Interface* (CGI) constitue la technique traditionnelle implanter une application CL/VR via web. L'idée de base est de produire les documents HTML par un programme qui est associé au serveur web. Ce programme reçoit en outre des paramètres saisis par l'utilisateur
  - Le CGI est la solution la plus ancienne, et sans doute encore la plus utilisée, pour la gestion de sites web dynamiques. La programmation de sites web avec PHP s'appuie d'ailleurs, pour tous les échanges client/serveur, sur le protocole CGI



PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- INTRODUCTION (2)
  - L'exécution du programme CGI par le serveur web se déroule en trois phases :
    - Le *Common Gateway Interface* (CGI) constitue la technique traditionnelle implanter une application CL/VR via web. L'idée de base est de produire les documents HTML par un programme qui est associé au serveur web. Ce programme reçoit en outre des paramètres saisis par l'utilisateur
    - *Exécution du programme CGI* : le serveur déclenche l'exécution du programme CGI, en lui fournissant les paramètres reçus ci-dessus ;
    - *Transmission du document HTML*: le programme CGI renvoie le résultat de son exécution au serveur sous la forme d'un fichier HTML, le serveur se contentant alors de faire suivre au client
  - Le programme CGI peut être écrit en n'importe quel langage (C, C++, Perl, script shell, python, etc) et est libre de faire toutes les actions (dans la limite de ses droits d'accès) pour satisfaire la demande. Il peut notamment rechercher et transmettre des fichiers ou des images, effectuer des contrôles, des calculs, créer des rapports, accéder à une base de donnée, etc.

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- COMMUNICATION CLIENT SERVEUR(3) : Appel d'un programme CGI
  - La communication entre la partie Client (feuille HTML) et la partie Serveur (programme CGI) de l'application se réalise à travers le protocole HTTP et quelques balises de HTML. Il s'agit d'outils d'appel d'un programme CGI, de saisie de paramètre, de transférer de paramètre au serveur
  - Appel d'un programme CGI :
    - Appel depuis un formulaire (balise FORM) avec l'attribut ACTION

      Exemples : **<FORM ACTION="/cgi-bin/calcu2l.cgi" METHOD="POST">**
    - Appel par une référence URL dans les autres balises possédant un attribut de référence à un URL (SRC, HREF, etc.), en particulier la balise A
    - Exemples :

      <A HREF="/cgi-bin/calcul.cgi">un programme cgi</A>

      <IMG SRC ="/cgi-bin/dessin1.cgi">

      <AREA SHAPE ="rect" COORS="122,12,23,28" HREF= "/cgi-bin/map1.cgi">

      <BODY BACKGROUND="/cgi-bin/dessin2.cgi">

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- COMMUNICATION CLIENT SERVEUR (2): Formation de paramètres
  - les paramètres d'un programme CGI pourront être formé dans une feuille HTML par deux manières suivantes :
    - dans une URL, on peut placer une chaîne de paramètres après le symbole « ? ». L'inconvénient de cette technique est qu'on ne peut pas, en général, modifier dynamiquement les valeurs de paramètres. La valeur de paramètres est dont déterminé par le concepteur de la feuille HTML mains non pas par les lecteurs de la feuille. Il maque donc de l'interactivité

      Exemples :

      <A HREF="/cgi-bin/calcul.cgi?val1=25&val2=39**">un programme cgi</A>**
    - Dans la balise <FORM> qui est conçue spécialement pour donner à l'utilisateur final la possibilité de saisir de la manière ad hoc la valeur des paramètres

      Exemples :

      <FORM ACTION="/cgi-bin/calcu2l.cgi" METHOD="POST">

      Hauteur : <INPUT Name="h" Type="text" Size="5">

      Largeur : <INPUT Name=« l" Type="text" Size="5">

      </FORM>

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- COMMUNICATION CLIENT SERVEUR (3): Format des paramètres
  - les paramètres d'un programme CGI sont envoyés sous forme d'une chaîne de caractères ASCI dont le format est le suivant:
    - chaque paramètre est exprimé par un couple d'opérandes «Nom-de-variable » et «Valeur-de-variable » reliées par le symbole '='
    - les paramètres sont reliés par le symbole '&'
    - deux règles de transformation automatique sont appliquées :
      - les caractères spéciaux (non alphanumérique) sont remplacés par +« %xx » où xx est le code hexadécimal du caractère concerné
    - Exemple :

      « val1=25&val2=39&texte=comment+%E7a+va%3F »

      signifie qu'on définie 3 champs (variables) val1, val2 et texte avec la valeur respective : '25', '39' et 'comment ça va? '

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- COMMUNICATION CLIENT SERVEUR (4): Deux modes de transmission de client au programme cgi via le serveur http :
  - GET : la chaîne de paramètres sera placée derrière l'URL concernée après le symbole '?', par le concepteur de la feuille ou par une procédure de transformation automatique du navigateur. Arrivée sur le serveur http, la chaîne de paramètre sera stockée dans la variable d'environnement QUERY_STRING. La longueur maximum d'une chaîne de paramètre dans le mode GET est 200 caractères
  - POST : La méthode POST permet d'envoyer des données au serveur et de les faire traiter par un programme qui les lit sur son entrée standard.
- Une seule manière de communication de programme CGI au client web via le serveur HTTP :
  - Le résultat de l'exécution d'un programme CGI, écrit sur sa sortie standard, sera transmis en réponse par le serveur HTTP au navigateur
  - Dans la plupart de cas, ce sont de données HTML. Le navigateur connaît ce type à travers de la valeur du champ « content-type » qui définie le type du document (content-type=text/html dans le cas d'un fichier html.

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- COMMUNICATION CLIENT SERVEUR (6): Formulaire avec POST
  - La balise formulaire (FORM) est seul moyen de base fourni dans le langage HTML pour rendre possible une interactivité entre l'utilisateur final et un programme CGI. On examine ce outil à travers un exemple
  - Considérons un document contenant le formulaire suivant, avec deux champs textuels simples et un bouton d'envoi :

    <FORM METHOD="post" ACTION="cgi-bin/prog.cgi">

    **Lettre : <INPUT NAME="L" TYPE="text" SIZE="1" maxlength="1">
**

    **Chiffre : <INPUT NAME="C" TYPE="text" SIZE="1">
**

    <INPUT NAME="Envoyer" TYPE="submit" VALUE="envoyer" />

    </FORM>

    Supposons que l'utilisateur écrit U dans le premier champ, 2 dans le second, puis presse le bouton d'envoi.

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- COMMUNICATION CLIENT SERVEUR (7): Formulaire avec POST
  - La chaîne de paramètre (sans limite de taille avec le mode POST) :
    - quand le bouton « Envoyer » est pressé le navigateur construit la chaîne de paramètres « **L=U&C=2 »** qui sera transmise au serveur http.
    - Ce dernier déclenche le programme CGI demandé et dirige la chaîne des paramètres vers l'entrée standard du programme CGI (STDIN).
    - de plus, ce programme a accès à des variables d'environnement, notamment aux deux suivantes :
      - **la variable CONTENT_LENGTH, dont la valeur est la chaîne de caractères représentant la longueur de la chaîne transmise (c'est la chaîne "7« dans le cas de "L=U&C=2")**
      - **la variable CONTENT_TYPE, dont la valeur est la chaîne "application/x-www-form-urlencoded"**

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- Mise en oeuvre sur le serveur : généralité
  - Le programme CGI représente la partie serveur de l'application considérée
  - Il est lancé par le serveur http selon l'URL communiquée par le navigateur
  - La chaîne de paramètres, est dirigée vers STANDARD INPUT du programme (si la méthode est GET) ou stockée dans la variable d'environnement QUERY-STRING du HTTP (si la méthode est GET)
  - Etant sous processus du HTTP, le programme CGI accès à tous les variables d'environnement du serveur HTTP

- **Structure : 3 parties en général**

Entête **Appel de l'interprète (si dans un langage interprété)** **- décode des paramètres**	**int main() {** **couple couples[MAX];** **int m;** **char *longue_chaine = extraire_couples(couples, &m);** **decoder_couples(m, couples);**
Traitement	**creer_document(m, couples);** **enregistrer(FICHIER, m, couples);** **free(longue_chaine);**
Format et envoi du résultat Format HTML - envoi sur STDOUT	**printf("Content-type: text/html\n\n");** **printf("<HTML>\n");** **...** **printf("</HTML>\n");** **return 0;** **}**

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- Mise en oeuvre sur le serveur (2): Exemple : Voir la date du serveur

Voici la date et l'heure locale

Nous sommes le Tue Apr 29 15:10:39 MET DST 1997

Partie client : test_date.html	Partie serveur : test_date.php
<HTML> **Voir la date sur le serveur** **</HTLM>**	**Header("Content-type: text/html");** **echo "<html>";** **echo "<head><title>Date et heure locale</title></head>";** **echo "<body> Voici la date et l'heure locale ";** **echo "<B>Nous sommes le".date('Y-m-d ');** **echo "</B></body></html>" ;**

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- Mise en oeuvre sur le serveur (3): Exemple : Recherche de films
  - Problème : consulter (par titre et par période) depuis du web les films stockés sur un fichier texte
  - Partie Client en HTML : (test_film.html)

```
<HTML><HEAD>
<TITLE>Formulaire recherche film</TITLE> </HEAD><BODY>
<H1>Formulaire de recherche de films</H1>
<FORM name="form1" ACTION='/cgi-bin/consuter_film.cgi' METHOD=POST>
Indiquer des paramètres de  recherche :
<P> Titre : <INPUT TYPE=TEXT SIZE=20 NAME = 'titre'></P>
<P> Année début : <INPUT TYPE=TEXT SIZE=4 NAME='anMin' VALUE=1900></P>
<P> Année fin : <INPUT TYPE=TEXT SIZE=4 NAME='anMax' VALUE=2100> </P>
<P> <B>Choix d'opération de combinaison </B>
ET <INPUT TYPE=RADIO NAME='comb' VALUE='ET' CHECKED></P>
OU <INPUT TYPE=RADIO NAME='comb' VALUE='OU'> ?
<P><INPUT TYPE=SUBMIT VALUE='Rechercher'></P>
</FORM></BODY></HTML>
```

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- Mise en oeuvre sur le serveur (4): Exemple : Recherche de films (suite)
  - Partie serveur : programme CGI en C : (test_film.cgi)

```
int main(int argc, char* argv[])
{
int nbFilms=0, anMin, anMax=0, anneeNaissance;
char titre[40], film[40], nom[40], prenom[40], comb[3];
int annee, lg;
FILE *films;
char ligne [MAXL], arg[MAXL];
short bonTitre=0, bonnePeriode=0;
/* On annonce qu'on va retourner un fichier HTML */
printf ("Content-type: text/html\n\n");
printf ("<HEAD><TITLE>R&eacute;sultat de la recherche</TITLE>");
printf ("</HEAD><BODY bgcolor=white>");
printf ("<H1><CENTER>R&eacute;sultat de la recherche</CENTER></H1>");
```

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- Mise en oeuvre sur le serveur (5): Exemple : Recherche de films (suite) : Partie serveur : programme CGI en C : (test_film.cgi)

```
/*************************************************
Phase 1: extraction des paramètres
*************************************************/
/* Dans la chaine 'ligne', on place les paramètres
provenant du formulaire */
lg = atoi (getenv("CONTENT_LENGTH"));
fgets (ligne, lg+1, stdin);
printf ("<B>Parametres :</B> %s<P>\n", ligne);
/* Avec la fonction ExtraitArg, on décrypte la chaine
pour en extraire les valeurs saisies par l'utilisateur */
strcpy (ligne, ExtraitArg (ligne, titre));
strcpy (ligne, ExtraitArg (ligne, arg)); anMin = atoi(arg);
strcpy (ligne, ExtraitArg (ligne, arg)); anMax = atoi(arg);
strcpy (ligne, ExtraitArg (ligne, comb));
printf ("<HR><B>Recherche:</B>Titre= %s %s année de %d `a
%d<br><HR>",
titre, comb, anMin, anMax);
```

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- Mise en oeuvre sur le serveur (6): Exemple : Recherche de films (suite) Partie serveur : programme CGI en C : (test_film.cgi)

```
/*************************************************
Phase 2: recherche et affichage
*************************************************/
/* Ouverture du fichier des films*/
films = fopen ("films.txt", "r");
while (fgets (ligne, MAXL, films))
{
/* Décryptage de chaque ligne */
sscanf (ligne, "%s %d %s %s %d",
film, &annee, nom, prenom, &anneeNaissance);
/* Test du film courant */
if (!strcmp (film, titre)) bonTitre = 1;
if (annee >= anMin && annee <= anMax) bonnePeriode=1;
/* Si oui : affichage du film dans la page HTML */
if ( (!strcmp ("ET", comb) && bonTitre && bonnePeriode)
|| (!strcmp ("OU", comb) && (bonTitre || bonnePeriode)))
{
nbFilms++;
printf ("<B>Film : </B> %s, %d, de %s %s <br>",
film, annee, prenom, nom);
}
bonTitre=0; bonnePeriode=0;
}
fclose (films);
if (!nbFilms)
printf ("<B>Désol´é : aucun film !</B>");
return 1;
}
/** Fin du programme */
```

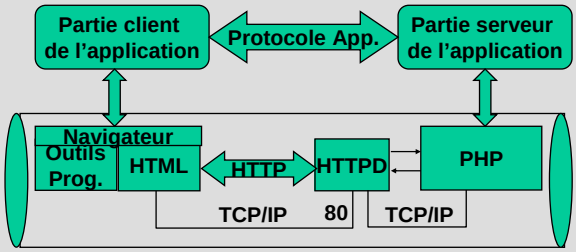
```
/*************************************************
Fonction ExtraitArg
*************************************************/
char* ExtraitArg (char* ligne, char *arg)
{
int pos = 0, posArg=0;
/* Recherche du signe '=', puis extraction jusqu''&' */
while (ligne[pos++] != '=');
while ( ligne[pos] != '&' && ligne[pos] != '\0')
arg[posArg++] = ligne[pos++];
arg[posArg] = '\0';
return &(ligne[pos+1]);
}
```

PROGRAMMATION SERVEUR WEB
Programmation avec CGI

- Mise en oeuvre sur le serveur (7): limites de l'approche CGI
  - Passage de paramètre en mode text avec un format contraignant. Idéalement, on souhaiterait disposer, directement dans le programme, des variables correspondant aux champs du formulaire HTML
  - Faible intégration avec HTML : le programme CGI peut-être écrit dans n'importe quel langage de programmation qui sont très différents de la syntaxe de HTML. On souhaite un rapprochement entre la programmation Client et la programmation serveur
  - Absence de notion session : le mode de programmation est par requête. Ce qui rendre difficile de développer des applications transactionnelles. Trois solutions utilisées :
    - Variables d'environnement. Les variables REMOTE USER et REMOTE PASSWORD peuvent être utilisées. Cette méthode est globalement insatisfaisante : les variables disponibles sont en nombre limité, fixées à l'avance, et en partie dépendantes du navigateur utilisé.
    - Paramètres dans l'URL. le contexte de la session peuvent être placé dans l'URL, après le caractère '?' qui indique le début des paramètres. Le programme peut ensuite les récupérer dans la variable QUERY STRING. Cette technique est limitée par la taille des informations qui peuvent être placées dans une URL, ainsi que par le manque de confidentialité.
    - Cookies. La technique la plus utilisée est de recourir aux *cookies*. Essentiellement, un *cookie* est une donnée, représentable sous la forme habituelle *nom=valeur*, que le navigateur conserve pour une période déterminée à la demande du serveur. Cette demande doit être effectuée dans un en-tête HTTP avec une instruction Set-Cookie. Par la suite, les cookies stockés par un navigateur sont envoyés au serveur dans une variable d'environnement HTTP COOKIE.

PROGRAMMATION SERVEUR WEB
ELEMENTS DU LANGAGE PHP

- GENERALITES
  - Un pré processeur de HTTP qui peut fonctionner avec la plupart des serveurs HTTP dans le marché
  - Un langage (v5 actuellement) de typage dynamique et orienté objet
  - Une intégration parfaite avec HTML : un fichier dans la syntaxe PHP est un fichier HTML intégrant des balises de programmation PHP. Avec cette définition, un fichier HTML est vue comme un programme php.



PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- GENERALITES (2)
  - Tout code PHP doit être inclus dans une balise <?php …. ?> . Des balises courtes <? … ?> sont parfois acceptées, mais elles risquent d'entrer en conflit avec d'autres langages (comme XML) et ne sont donc pas recommandées
  - Comme en C, le séparateur d'instructions est le point-virgule ';'. Noter qu'une instruction vide , marquée par un point-virgule est acceptée. La syntaxe suivante est donc correcte, bien que le second ';' ne serve à rien :

    ```
    echo "Ceci est une instruction inutile";
    ```
  - Il existe trois manières d'inclure des commentaires au sein du code PHP :
    - comme en C, entre les signes /* et */ ;
    - comme en C++, en commençant une ligne par // :
    - comme en *shell* Unix, avec # .

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- VARIABLES, TYPES ET EXPRESSIONS
  - **Les variables sont des symboles qui permettent de référencer des valeurs. Comme leur nom l'indique, les variables peuvent référencer des valeurs différentes au cours de l'exécution d'un script, ce qui les distingue**
  - **des *littéraux* ('0', '1.233', 'Ceci est une chaîne') qui représentent directement une valeur immuable**
  - **Un nom de variable commence toujours par un '$', suivi d'au moins un caractère non-numérique (le '_' est autorisé) puis de n'importe quelle combinaison de chiffres et de caractères.**
  - **Remarque : *PHP distingue les majuscules et minuscules dans le nom des variables : $mavariable et $maVariable désignent donc deux variables différentes. En revanche les noms de fonction sont insensibles à la casse*.**
  - **Le type de la valeur associée à une variable peut lui même changer (typage dynamique), contrairement `à des langages typés (compilés) comme le C, Java, C++, …**
  - **Au cours de la vie d'un script, une variable peut donc référencer un entier, puis une chaîne, puis un objet, etc. L'interpréteur se charge de gérer l'espace mémoire nécessaire pour stocker la valeur référencée par une variable**

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (2)
  - Déclaration : Il n'y a pas de déclaration de variable en PHP ! PHP crée automatiquement une variable dès qu'un nouveau symbole préfixé par '$' apparaît dans un script
  - Variable de variable : Le nom d'une variable peut lui-même être une variable. Le code ci-dessous affecte la valeur 10 à la variable $mavar, dont le nom est lui-même la valeur de la variable $v1. Cette construction assez exotique a une utilité douteuse : $v1 = "mavar"; $v1 = 10;
  - Constantes : Une constante est un symbole associé à une valeur mais, à la différence des variables, ce symbole ne peut jamais être modifié. Une constante peut être vue comme un littéral, mais désigné de manière symbolique, ce qui est préférable pour la clarté du code et son évolutivité. Les constantes sont définies avec la commande define.

    define(PI, 3.14116); define (MON SERVEUR, "cartier.cnam.fr:8080");
  - Par convention (mais ce n'est pas obligatoire) les constantes sont en majuscules. Une bonne règle est de ne *jamais* utiliser de valeur en dur dans un script, mais de définir une constante. Deux avantages :
    - le code est plus lisible
    - si on veut changer la valeur, on peut le faire en un seul endroit.

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (3)
  - **PHP distingue les *types scalaires* (entiers, flottants, chaînes) et les *types agrégats* (tableaux et classes). Les valeurs de types scalaires ne peuvent pas se décomposer, contrairement à celles des types agrégats.**
  - Types numériques et booléens
  - **Ils comprennent les entiers et les flottants, ces derniers ayant une partie décimale séparée de la partie entière par un «.»**

    $i = 1; // Entier en notation décimale

    $i = 011; // Notation octale (9 en décimal)

    $i = 0x11; // Notation hexadécimale (17 en décimal)

    $f = 3.14116 // Flottant

    $f = 0.3e-3 // Notation exponentielle (soit 0,0003)
  - **PHP n'a pas de type booléen explicite. Comme en C, la valeur *faux* est le 0, la chaîne vide ou la chaîne "0", et toute autre valeur est *vrai*, y compris un chiffre négatif par exemple. Les constantes TRUE et FALSE sont prédéfinies et peuvent être utilisées dans les structures de contrôles.**

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (4)
  - Chaînes de caractères : **Les chaînes de caractères peuvent êtres encadrées par des guillemets simples (') ou des guillemets doubles ("). Les deux types de chaîne ne sont cependant pas équivalents.**
    - Guillemets simples : **On ne peut y inclure ni variables, ni caractères d'échappement (comme '\n'). En revanche les sauts de lignes sont acceptés. Si on veut inclure un (') dans une telle chaîne, il faut le préfixer par '\'. Exemple : 'C\'est une chaîne avec guillemets simples et un saut de ligne.'**
    - Guillemets doubles : **Les chaînes peuvent inclure des noms de variables qui seront remplacées par leur valeur à l'exécution.**

      Exemple : $nom = "Hitchcock"; echo "Le réalisateur de Vertigo est $nom. ";

\n	Saut de ligne
\r	Retour chariot
\t	Tabulation
\\	Le signe '\'

\$	Le signe '$'
\"	Un guillemet double
\0nn	Une chaîne en octal
\xnn	Une chaîne en hexadécimal

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (5)
  - Tableaux
    - **Un tableau est une suite de valeurs référencées par une unique variable. PHP gère dynamiquement la taille des tableaux, ce qui permet d'ajouter ou de supprimer à volonté des valeurs sans se soucier de l'espace nécessaire. Les tableaux en PHP peuvent être soit *indicés* – les valeurs sont référencées par leur position en débutant à 0 – soit *associatifs*. Dans ce cas les valeurs sont référencées par des noms – ou *clés* – donnés explicitement par le programmeur.**
    - Tableaux indicés : **quelques exemples.**

      $tab[0] = « élément 1 "; $tab[1] = "élément 2 "; $tab[2] = 120;
    - **Notez bien que les indices commencent à 0, ce qui nécessite parfois un peu de réflexion quand on programme des itérations sur un tableau.**
    - **Une caractéristique importante et très utile : PHP affecte automatiquement un indice à un nouvel élément du tableau. Cet indice est le numéro de la première cellule vide. Donc le code ci-dessous est équivalent au précédent.**

      $tab[] = "élément 1 "; // $tab[0] ! ;
      $tab[] = "élément 2 "; // $tab[1] !
      $tab[] = 120; // $tab[2] !

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (6)
  - Tableaux
    - L'instruction array **: offre un moyen d'initialiser facilement un tableau..**

      Exemple $tab = array ("élément 1 ", "élément 1 ", 120);
    - Tableaux associatifs **permettent d'accéder à un élément par sa clé. Un tableau indicé est un cas particulier de tableau associatif, où les clés sont des entiers en séquence.**

      Exemple : tableau de films

Définition classique:
```
$mes["Vertigo"] = "Hitchcock";
$mes["Sacrifice"] = "Tarkovski";
$mes["Alien"] = "Scott"; !
```

Utilisation array:
```
$mes = array (
"Vertigo" = "Hitchcock",
"Sacrifice" = "Tarkovski",
"Alien" = "Scott");
```

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (7)
  - Tableaux
    - Tableaux multi-dimensionnels : **Les tableaux indicés et associatifs se généralisent aux tableaux multi-dimensionnels, pour lesquels l'indice, ou la clé, est constituée de plusieurs valeurs. Un tableau à deux dimensions peut être vu comme une table avec lignes et colonnes. Exemple :**

```
$tab[0][0] = "En haut gauche";
$tab[0][1] = "En haut droite";
$tab[1][0] = "En bas gauche";
$tab[1][1] = "En bas droite";
```

Utilisation array:
```
$mes = array (
"Vertigo" => array ( "Alfred", "Hitchcock"),
"Sacrifice" => array ( "Andrei", "Tarkovski"),
"Alien" => array ( "Ridley", "Scott"));
```

Note :
Dans l'exemple avec array les tableaux imbriqués sont indicés et contiennent chacun deux éléments. $mes["Vertigo"][1] est donc la chaîne "Hitchcock".

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (8)
  - Convention de typage dynamique
    - Le type d'une variable est déterminé par le contexte dans lequel elle est utilisée. Quand on implique par exemple une chaîne de caractères dans une addition, PHP essaiera d'en extraire un numérique : l'expression

      $r = 1 + "3 saucissons"; donne la valeur 4 à $r puisque la chaîne est convertie en 3.
    - Note : Si la conversion s'avère impossible, la valeur 0 sera utilisée
    - On peut convertir le type d'une variable en préfixant le nom de la variable par (type) où type est integer, double, string, etc.

      Exemple : $v = "3 petits poulets";

      $v = (integer) $v; // affecter 3 à $v

      $v = (double) $v; // affecter 3.0 à $v

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (9)
  - Expressions : **On désigne par *expression* toute construction du langage qui produit une valeur. Exemple :**
    - **Les variables, littéraux et constantes sont déjà des expressions : elles produisent leur propre valeur**
    - **Les 3 instructions : 10; $i = 10; $i; sont trois expressions produisent la valeur 10**
  - Opérateurs : **les *opérateurs* constituent le moyen le plus courant de créer des expressions. Un opérateur produit une valeur par manipulation – addition, soustraction, etc. – de valeurs fournies par d'autres expressions. Dans leur forme la plus simple, les opérateurs agissent sur des variables ou des littéraux.**

    Exemple : $a = 3; $a + 4; $b = $a + 2;
  - Remarque : **toute expression peut être interprétée comme une *expression booléenne* avec la valeur false si la valeur produite est égale à 0 (ou à une chaîne vide), true sinon. Donc toute expression peut être utilisée dans les structures de test**

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (10)
  - Opérateurs arithmétiques :
    - **$a + $b //Addition de $a et $b**
    - **$a - $b //Soustraction de $b `a $a**
    - **$a * $b //Multiplication de $a et $b**
    - **$a / $b //Division de $a par $b**
    - **$a % $b //$a modulo $b (reste de la division de $a par $b)**
    - **$i++; // incrémenter $i (qui vaut 5 si son ancienne valeur est 4)**
    - **$j = ++$i; // incrémenter $i puis affecter cette valeur à $j**
    - **$k = $i++; // affecter la valeur de $i à $k puis incrémenter $i**
    - **$k--; // décrémenter $k**
  - Opérateur de concaténation des chaînes:
    - **$c1 = "Bonjour "'; $c2 = " le monde";**
    - **$c = $c1 . " tout " . $c2 . " ! "; //donne « Bonjour tout le monde » dans $c**
    - **$c .= " ! "; //donne « Bonjour tout le monde !» dans $c (éviter de répéter $c)**

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (11)
  - Opérateurs de bits :
    - **$a & $b //*ET* binaire**
    - **$a | $b //*OU* binaire**
    - **$a ^$b //*OU EXCLUSIF* binaire**
    - **~$a //COMPLEMENT (INVERSER) des bits**
    - **$a << $b //Décale les bits de $a de $b positions vers la gauche.**
    - **$a >> $b //Décale les bits de $a de $b positions vers la droite.**
  - Opérateurs logiques
    - **$a && $b; $a and $b; //ET logique.**
    - **$a || $b ; $a or $b ; // OU logique**
    - **$a xor $b //Ou exclusif**
    - **!$a // NOT**

PROGRAMMATION SERVEUR WEB
ELEMENTS DU LANGAGE PHP

- TABLE VARIABLES, TYPES ET EXPRESSIONS (12)
  - Opérateurs de comparaison :
    - **$a == $b //Vrai si $a est égal à $b.**
    - **$a != $b //Vrai si $a est différent de $b.**
    - **$a $b //Vrai si $a est inférieur à $b.**
    - **$a $b //Vrai si $a est supérieur à $b.**
    - **$a $b //Vrai si $a est inférieur ou égal à $b.**
    - **$a $b //Vrai si $a est supérieur ou égal à $b.**
  - Note : **le test d'égalité s'écrit avec deux '=' ('==‘). Une erreur très courante est d'oublier un '=' dans un test d'égalité. L'interpréteur PHP ne signale pas cette erreur en affectant la valeur VRAI à l'expression**

    Exemple $i = 1; $j = 2;

    if ($i == $j) ... // Renvoie FALSE: i est différent de j

    if ($i = $j) ... // Renvoie TRUE !

PROGRAMMATION SERVEUR WEB
ELEMENTS DU LANGAGE PHP

- STRUCTURE DE CONTRÔLE
  - **Les structures de contrôles sont les *tests* et les *boucles*. Elles permettent de spécifier, en fonction de l'état d'un script à l'exécution (déterminé par la valeur de certaines variables), quelles sont les parties du script à effectuer (structures de tests), ou combien de fois on doit les effectuer (structures de boucle)**
  - Tests :
    - **La structure la plus courante est le** if … else**. Voici la syntaxe.**

```
if (expression)     {
// Bloc si expression est vraie.
}else{
// Bloc si expression est fausse.
      // Ici le script continue.
}
```

Notes :
- la clause ‘else’ est optionnelle
- la clause **if .. else** peuvent être imbriquées

PROGRAMMATION SERVEUR WEB
ELEMENTS DU LANGAGE PHP

- STRUCTURE DE CONTRÔLE (2)
  - Tests : switch ... case … default **:**

```
switch (expression)
case valeur1:
// expression vaut valeur1.
// Il faut sortir du switch !
break;
case valeur2:
// expression vaut valeur2.
break;
...
default:
// expression vaut autres valeurs
break;
```

Note :
il existe une syntaxe différente pour les **if-else** :
Le bloc après le if ou le else commence par ':', et la structure se termine par endif;.

```
if (expression):
// Bloc si expression est vraie.
else:
// Bloc si expression est fausse.
endif;
```

PROGRAMMATION SERVEUR WEB
ELEMENTS DU LANGAGE PHP

- STRUCTURE DE CONTRÔLE (3)
- Boucles :
  - Le while : **permet d'exécuter un bloc d'instructions tant qu'une condition est remplie. Deux structures sont équivalentes :**

```
while (expression)
{
// expression est vraie
}
```

```
while (expression):
// expression est vraie.
endwhile;
```

  - Le do-while : **est une variante du while qui effectue le bloc *avant* d'évaluer le test. Le bloc est toujours exécuté au moins une fois.**

```
do
{
// expression peut-être faut au premier passage
while (expression);
}
```

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- STRUCTURE DE CONTRÔLE (4)
  - Boucles :
    - Le for : **permet d'exécuter une itération sur une variable incrémentée (ou décrémentée) à chaque passage de la boucle. C'est la plus puissante des boucles, on peut y spécifier :**
      - l'initialisation des valeurs conditionnant l'itération ;
      - la ou les instructions faisant évoluer ces valeurs à chaque passage ;
      - la condition d'arrêt de la boucle.

```
for ($x=0; $x <10; $x++)
{
// Ici des instructions
}
```

```
$a=1; $b=6;
while ($a  $b)
{
$a++;
echo "$a = " . $a);
}
peut être remplacé par :
for ($a=1,$b=6; $a $b;
$a++, echo "$a = " . $a);
```

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- STRUCTURE DE CONTRÔLE (5)
  - Boucles :
    - Les instructions **break** et **continue** : **permet d'obtenir un comportement plus souple dans l'exécution d'une boucle :**
      - break déclenche la sortie forcée de la boucle ;
      - continue dirige l'exécution à la prochaine évaluation du test de continuation, en sautant les éventuelles instructions complétant le corps de la boucle.

```
$x = 0;
while (1)
{
if ($x == 10) break;          // $x vaut 10? On s'en va
$x++;                         // permet d'incrémenter $x de 1.
if ($x != 5) continue;        // $x différent de 5? On saute la suite
// Ici les instructions pour le cas où $x vaut 5
}
```

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- FONCTIONS
  - Définition : **Les fonctions en PHP doivent être définies avant leur appel. Le nom d'une fonction ne commence pas par '$', et n'est pas sensible à l'utilisation des majuscules/minuscules. Voici la syntaxe :**

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

  - Contraintes :
    - **Une fonction peut prendre un nombre arbitraire, mais *fixe* d'arguments. Les fonctions avec un nombre variable d'arguments n'existent pas en PHP**
    - **Une fonction peut renvoyer une valeur avec l'instruction return, mais ce n'est pas indiqué dans sa signature. On ne peut renvoyer qu'une seule valeur, mais cette valeur peut être un tableau.**

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- FONCTIONS (2)
  - Passage des arguments :
    - par valeur **: Les arguments sont passés généralement *par valeur*, ce qui signifie qu'une copie des variables est faite au moment de l'appel de la fonction, et que les éventuelles modifications faites dans le corps de la fonction sur les arguments n'ont qu'un effet local.**
    - par adresse : **Il est cependant possible de passer des variables *par adresse* en préfixant la variable par &. 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.

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- FONCTIONS (3)
  - Passage d'arguments :
    - Valeurs par défaut : **Il est possible de définir des valeurs par défaut pour un ou plusieurs arguments d'une fonction (comme en C++).**

      Exemple

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

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- FONCTIONS (4)
  - Fonctions et variables : **PHP propose trois types de variables (la même terminologie du C) :**
    - Variables automatiques : **qui sont créées dès que l'on entre dans leur espace de définition, qui est soit le script, soit une fonction. Elle disparaissent quand on sort de cet espace. Sa valeur n'est donc pas sauvegardée entre deux appels de la même fonction**
    - Variables statiques :**. qui sont persistantes entre les appels.**
    - Variables globales : **En principe le corps d'une fonction est une partie de code complètement isolée. En particulier les variables définies à l'extérieur de la fonction ne sont pas visibles. Une variable globale est au contraire visible partout**
    - Par défaut **les variables sont automatiques. Les autres types de variables sont beaucoup moins utilisés**

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- INTEGRATION AVEC HTML
  - PRINCIPE D'INTEGRATION :
    - Le langage PHP a été créé par Rasmus Lerdorf vers la fin de l'année 1994. Il est actuel en 4e version.
    - Contrairement à d'autres langages, PHP est exclusivement dédié à la production de pages HTML générées dynamiquement. Pour ce but, un programme PHP n'a pas une existence indépendante mais est dans un format intégré avec la syntaxe de HTML :

```
<HTML>
<!--
les balises HTML
et de code PHP
-->
</HTML>
```

```
<HTML> <HEAD> <TITLE>HTML avec PHP</TITLE>
</HEAD> <BODY> <H1>HTML + PHP</H1>
Nous sommes le <?php echo Date ("j/m/Y"); ?><P>
<?php
echo "Je suis $HTTP_USER_AGENT et je dialogue
avec $SERVER_NAME.";
?>
</BODY></HTML>
```

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

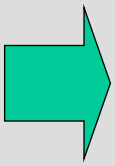
- INTEGRATION AVEC HTML (2)
  - PRINCIPE D'INTEGRATION :
    - Syntaxe d'un fichier php **: Un fichier PHP a le suffixe « .php ». Il est un fichier HTML (avec toute sa capacité) comprenant des balises « contenant-PHP ». Cette balise contenant-PHP est une balise vide de forme suivante :**

      <?php Code PHP ?>
    - Evaluation **: L'évaluation du fichier php sera fait en 3 phases :**
      - En analysant la demande d'exécution d'une URL du client web, le serveur HTTP détecte un fichier php (avec le suffixe .php), il forme une demande au processeur PHP d'évaluer le fichier et attend le retour du résultat de l'évaluation
      - Le processeur PHP évalue tous les balises contenant-PHP en replaçant chaque balise par le résultat de l'évaluation du code PHP contenu dans la balise. Il forme ainsi un document HTML qui sera renvoyé au serveur HTTP
      - Le serveur HTTP effectue l'envoi du document reçu au client web

PROGRAMMATION SERVEUR WEB ELEMENTS DU LANGAGE PHP

- INTEGRATION AVEC HTML (3)
  - PRINCIPE D'INTEGRATION :
    - Remarques :
      - Le script PHP est toujours stocké et exécuté du côté serveur par l'interpréter de PHP se trouvant sur la même machine que celle du serveur HTTP

```
<HTML> <HEAD>
<TITLE>HTML avec PHP</TITLE>
</HEAD>
<BODY>
<H1>HTML + PHP</H1>Nous sommes le
<?php echo Date ("j/m/Y"); ?>
<P> <?php
echo "Je suis $HTTP_USER_AGENT et
je dialogue avec $SERVER_NAME.";
?>
</BODY></HTML>
```



```
<HTML> <HEAD> <TITLE>
HTML avec PHP</TITLE>
</HEAD>
<BODY>
<H1>HTML + PHP</H1>
Nous sommes le 23/09/2004
<P> Je suis Mozilla/7.0 et je
dialogue avec nyx.unice.fr.
</BODY></HTML>
```

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- BASE DE DONNEES RELATIONNELLES
  - Concepts de base du modèle :
    - Structures logiques : relations (tables), attributs (colonnes), domaines (types), tuples (lignes)
    - Contraintes d'intégrités structurelles : Clés primaires, Clés étrangères (de référence)
    - Langage de requêtes : SQL = Structured Query Language
  - Système de gestion de bases de données relationnelles (SGBD)
    - Systèmes d'exploitation spécifiques permettant de gérer et d'exploiter les bases de données relationnelles
    - Caractéristiques : en réseaux, multi-utilisateurs, interface d'accès SQL, outils de programmation intégrés, sécurisant, etc.
    - Produits :
      - Propriétaire : Oracle, SQL/Server, SYBASE, ACCESS, etc.
      - Libre : MySQL, ?

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- BASE DE DONNEES RELATIONNELLES (2)
  - Exemple : schéma d'un club d'achat sur Internet
    - Produit (id-produit, titre, genre, prixU)
    - Membre (id-membre, nom, prénom, sexe, date-nais, ville, cp, adresse, email)
    - Commande (id-commande, id-membre, date-comm, etat-comm)
    - Detail-comm(id-comm, id-film, quantite, taux-remise)
  - Table Produit

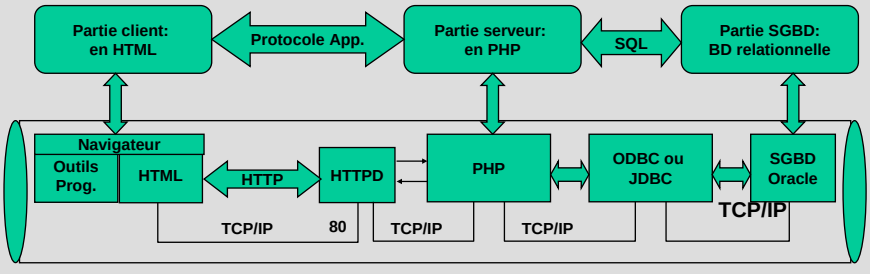
Id-produit	Titre	Genre	prixU
1	**Ma chanson**	**Chanson**	**7**
2	**Mon roman**	**Roman**	**15**
3	**Mon film**	**Film**	**20**
4	**Mon album**	**Musique**	**20**

Une requête SQL : *select * from produit where prixU > 10;*

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- BASE DE DONNEES RELATIONNELLES (4)
  Rappel de l'architecture 3-tiers : PHP et Oracle (ou mySQL)

1er-Tiers de « Client » : Programmation interface d'utilisateur	2e-Tiers d' «Objets de métier » : Programmation de services liés à l'application réelle	3e-Tiers de «Bases de données » : Programmation des sources de données pour l'application



PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- BASE DE DONNEES RELATIONNELLES (4)
  - Rappel de l'architecture 3-tiers : PHP et Oracle (ou mySQL) : Il s'agit d'une architecture à trois composantes, chacune réalisant une des trois tâches fondamentales d'une application d'entreprise :
    - le *navigateur* constitue *l'interface graphique de l'application* dont le rôle est de permettre à l'utilisateur de visualiser et d'interagir avec l'information ;
    - SGBDR est *le serveur de données de l'application* ;
    - enfin l'ensemble des fichiers PHP contenant le code d'extraction, traitement et mise en forme des données est le *serveur d'application*, associé à Apache qui se charge de transférer les documents produits entre le client et le serveur d'application.

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- DEFINITION DE STRUCTURES ET DE DONNEES
  - Création des tables : **Pour créer des tables, on utilise une partie de SQL dite Langage de Définition de Données (DDL) dont la commande principale est le CREATE TABLE :**
  - Exemple :

```
CREATE TABLE produit(
Id-produit VARCHAR (5) primary key,
titre VARCHAR (30) not null,
genre VARCHAR (5) not null,
prixU NUM (15,2) not null
);
```

  - Remarque : **On peut utiliser indifféremment les majuscules et les minuscules pour les mot-clé de SQL. De même, les sauts de ligne, les tabulations et les espaces successifs dans un ordre SQL équivalent à un seul espace pour l'interpréteur et peuvent donc être utilisés librement pour clarifier la commande**

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- DEFINITION DE STRUCTURES ET DE DONNEES (2)
  - Mise à jour d'une table : **Trois commandes de SQL permettant d'effectuer les mises à jours dans une table :**
    - INSERT : insertion d'une nouvelle ligne dans une table

      Syntaxe : INSERT INTO Table(attributs) VALUES (valeur d'une ligne);
    - DELETE : SUPRESSION d'une ou de plusieurs ligne(s) dans une table

      Syntaxe : DELETE FROM Table WHERE Qualification de lignes;
    - UPDATE : Modification d'une ou de plusieurs lignes dans une table

      Syntaxe : UPDATE Table
      liste de couples attribut=expression
      WHERE Qualification de lignes;
    - Exemple :

      INSERT Into Produit (id-produit, titre, genre, prixU) Values(‘’, ‘Ma chanson’, ‘chanson’, 10.0);

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- DEFINITION DE STRUCTURES ET DE DONNEES (3)
  - Connexion au SGBD (mySQL) : **La création des structures doit se faire par une interface spécifique de chaque SQBD (SQLPLUS sous ORACLE ou mysql sous MYSQL) :**

```
% mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 3.23.9
Type 'help' for help.
mysql> CREATE DATABASE Ecommerce;
mysql>
mysql> GRANT ALL PRIVILEGES ON Films.* TO admin@localhost
IDENTIFIED BY 'passadmin';
mysql> exit
```

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

• DEFINITION DE STRUCTURES ET DE DONNEES (4)

– Définir les tables (mySQL) : **la création des tables doit être faite dans le console mysql :**

```
% mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 10 to server version: 3.23.9
Type 'help' for help.
mysql> USE Ecommerce;
Database changed
mysql> CREATE TABLE produit (
-> (Id-produit VARCHAR (5) primary key,
-> titre VARCHAR (30) not null,
-> genre VARCHAR (5) not null,
-> prixU NUM (15,2) not null
-> );
Query OK, 0 rows affected (0.01 sec)
mysql> exit
```

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

• DEFINITION DE STRUCTURES ET DE DONNEES (5)

– Définir les tables (mySQL) : **la création des tables doit être faite dans le console mysql :**

– Remarques :

- **Vous pouvez consulter son schéma avec la commande DESC**
  mysql> DESC produit;
- **On peut configurer le SGBD pour pouvoir utiliser les accents dans les noms des tables, attributs, etc. mais cette pratique est à éviter car le schéma de données peut-être utilisé par les autres développeurs internationaux qui ne disposent pas de même configuration**
- **Pour détruire une table, on dispose de la commande DROP TABLE.**
- **mysql> DROP TABLE produit;**
- **Script SQL : On peut créer un fichier, dit script SQL, contenant les commandes et de l'exécuter. Un script SQL peut contenir un ensemble de commandes, chacune devant se terminer par un ';' Le fichier Script SQL a un suffixe '.sql'. On peut exécuter un script « unscript.sql » sous mysql de la manière suivante :** % mysql < unscript.sql

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- MANIPULATION DE DONNEES :
  - La manipulation de données relationnelles est faite à l'aide du langage de requête de SQL :

SELECT liste des colonnes figurées au résultat
FROM liste des tables participant à la requête
WHERE Expression de qualification des lignes
GROUP BY Liste des colonnes faisant critère du groupement
HAVING Expression de qualification des groupes
ORDER BY Expression de qualification des lignes

  - Remarques :
    - Select .. From … Where peuvent être imbriqué. Cela n'est pas le cas de Group by … Having et Order by qui sont apparu au maximum 1 fois dans la requête au premier niveau d'imbrication
    - Il est possible d'utiliser les requêtes prédéfinies comme des tables calculées

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- MANIPULATION DE DONNEES (2) :
  - Remarque :
    - **sous Unix, une table est stockée par MySQL dans un fichier de même nom. Comme Unix distingue les majuscules et les minuscules pour les noms de fichier, il faut absolument respecter la casse dans le nom des tables, sous peine d'obtenir le message :**
      '*Table does not exist*'
    - Quelques commandes utiles : **Enfin *mysql* fournit tout un ensemble de commandes pour inspecter les tables, donner la liste des tables d'une base de données, etc. Voici une sélection des commandes les plus utiles :**
      - SELECT DATABASE(); C'est une pseudo-requête SQL (il n'y a pas de FROM) qui affiche le nom de la base courante.
      - SELECT USER(); Idem, cette pseudo-requête affiche le nom de l'utilisateur courant.
      - SHOW DATABASES; Affiche la liste des bases de données.
      - SHOW TABLES; Affiche la liste des tables de la base courante.
      - SHOW COLUMNS FROM *NomTable*; Affiche la description de *NomTable*.

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- ACCES AUX DONNEES DEPUIS PHP :
  - Interface phpMyAdmin: **phpMyAdmin est un outil entièrement écrit en PHP qui fournit une interface simple et très complète pour administrer une base MySQL. La plupart des commandes de l'utilitaire *mysql* peuvent s'effectuer par l'intermédiaire de phpMyAdmin, les opérations possibles dépendant bien sûr des droits de l'utilisateur qui se connecte à la base. Voici une liste des principales possibilités :**
    - **Créer et détruire des bases de données (sous le compte root de MySQL).**
    - **Créer, détruire, modifier la description des tables.**
    - **Consulter le contenu des tables, modifier certaines lignes ou les détruire, etc.**
    - **Exécuter des requêtes SQL interactivement**
    - **Charger des fichiers dans des tables et, réciproquement, récupérer le contenu de tables dans des fichiers ASCII.**

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- ACCES AUX DONNEES DEPUIS PHP (2) :
  - Interface Mysql-PHP : **PHP communique avec MySQL par l'intermédiaire d'un ensemble de fonctions qui permettent de récupérer, modifier, ou créer à peu près toutes les informations relatives à une base de données. Parmi ces informations, il faut compter bien entendu le contenu des tables, mais également leur description (le *schéma* de la base). L'utilitaire phpMyAdmin utilise par exemple les fonctions permettant d'obtenir le schéma pour présenter une interface d'administration, générer à la volée des formulaires de saisie, etc.**
  - **Exemple :**

```
require ("Connect.php");
$connexion = mysql_pconnect (SERVEUR, NOM, PASSE);
if (!$connexion)
{
echo "Désolé, connexion " . SERVEUR . " impossible\n";
exit;
}
```

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- ACCES AUX DONNEES DEPUIS PHP (3) :
  - Interface Mysql-PHP :

Mysql_connect	**Pour établir une connexion avec MySQL, pour un compte utilisateur, et un serveur donné. Renvoie une valeur qui peut être utilisée ensuite pour dialoguer avec le serveur.**
Mysql_pconnect	**Idem, mais avec une connexion *persistante.* Cette deuxième version est plus performante.**
Mysql_select_db	**Permet de se placer dans une base de données**
Mysql_query	**Pour exécuter une requête SQL. Renvoie une variable représentant le résultat de la requête.**
Mysql_fetch_object	**Permet de récupérer une des lignes du résultat, et positionne le curseur sur la ligne suivante. La ligne est représentée sous forme d'un *objet* (un groupe de valeurs).**
Mysql_fetch_row	**Permet de récupérer une des lignes du résultat, et positionne le curseur sur la ligne suivante. La ligne est représentée sous forme d'un *tableau* (une liste de valeurs).**
Mysql_error	**Renvoie le message de la dernière erreur rencontrée**
Mysql_fetch_array	**Renvoi les résultats dans un tableau**

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- ACCES AUX DONNEES DEPUIS PHP (5):
  - Inclusion de fichiers – Constantes :**L'instruction require ("Connect.php"); permet d'inclure le contenu du fichier "Connect.php" dans le script. Cela évite de répéter systématiquement des informations. Ici on a placé dans le fichier *Connect.php* quelques informations de base sur le site : le nom du serveur, le nom de la base et le compte d'accès à la base (La commande define permet de définir des *constantes*):**

```
<?php
    define (U,"adminFilms);
    define (P, "mdpAdminFilms");
    define (S, "localhost");
    define (B, "Films");
    ?>
```

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- ACCES AUX DONNEES DEPUIS PHP(4) :
  - Interface Mysql-PHP : **Exemple**

```
<HTML><HEAD>
<TITLE>Connexion MySQL</TITLE>
<BODY><H1>
Interrogation de la table produit</H1>
<?php
require ("Connect.php");
$conn = mysql_connect (S, U, P)
or die("Désolé, connexion
" . S ." impossible\n« );

mysql_select_db (B, $conn)or
die("Désolé, accès la base "
. B . " impossible\n";')
```

```
$resultat = mysql_query
("SELECT * FROM produit", $conn);
if ($resultat)
{
while ($prod = mysql_fetch_object ($resultat))
{
echo  "Le prix du  $prod->titre est
$prod->prixU<BR>\n";
}
}
else
{
echo "<B>Erreur dans l'exécution de la
requête.</B><BR>";
echo "<B>Message de MySQL :</B> "
. mysql_error($conn);
}
?>
</BODY></HTML>
```

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- ACCES AUX DONNEES DEPUIS PHP (6) :
  - Connexion au serveur **Supposons que nous disposons avec la commande require des symboles de constantes U, P, B et S soit tous les paramètres nécessaires à la connexion à MySQL :**

$conn = mysql_pconnect (S, U, P);

- **La fonction *mysql_pconnect* essaie d'établir une connexion avec le serveur *mysqld*.**
- **En cas de succès une valeur positive est renvoyée, qui doit ensuite être utilisée pour dialoguer avec le serveur.**
- **En cas d'échec *mysql pconnect* affiche un message d'erreur et renvoie une valeur nulle**
- Remarque : **Si vous voulez éviter que MySQL envoie un message en cas d'échec à la connexion, vous pouvez préfixer le nom de la fonction par '@'. C'est à vous alors de tester si la connexion est établie et d'afficher un message selon vos propres normes de présentation.**

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- ACCES AUX DONNEES DEPUIS PHP (7) :
  - Exécution de la requête : **On utilise la fonction *mysql query*.**

    $resultat = mysql_query ("SELECT * FROM produit", $conn);

    - **Comme d'habitude, cette fonction renvoie une valeur positive si la fonction s'exécute correctement. En**
    - **cas de problème, erreur de syntaxe par exemple, le bloc associé au else est exécuté. Il affiche le message fourni par MySQL *via* la fonction *mysql error*.**

    **echo "<B>Erreur dans l'exécution de la requête.</B>
";**

    echo "<B>Message de MySQL :</B>" . mysql_error();
  - Note :
    - **l'utilisation de balises HTML dans les chaînes de caractères, ainsi que**
    - **l'utilisation de l'opérateur de concaténation de chaînes : '.'.**

PROGRAMMATION SERVEUR WEB BASES DE DONNEES PHP

- ACCES AUX DONNEES DEPUIS PHP (8) :
  - Affichage du résultat :
    - **Ici nous avons à résoudre un problème classique d'interaction entre une base de données et un langage de programmation. Le résultat est un ensemble, arbitrairement grand, de lignes dans une table, et le langage ne dispose pas de structure pratique pour représenter cet ensemble.**
    - **La technique habituellement utilisée est de parcourir les lignes une à une avec un *curseur* et d'appliquer le traitement à chaque ligne individuellement. Cela évite d'avoir à charger tout le résultat en même temps. Ici on utilise une des fonctions *fetch* qui correspondent à l'implantation de cette notion de curseur dans MySQL.**

      $film = mysql_fetch_object ($resultat);

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- INTRODUCTION
  - **on est amené souvent à programmer de manière répétitive des parties de code correspondant soit à des opérations routinières (connexion à la base, exécution d'une requête), soit à des tests (validations des champs de saisie, vérification que des instructions se sont exécutées correctement), soit enfin à du texte HTML. Cette répétition a de nombreux inconvénients :**
    - **L'objectif bien précis d'script se trouve dans un ensemble d'instructions dont beaucoup correspondent à des tâches annexes qui ne sont pas directement liées à cet objectif principal**
    - **la duplication de code est un gros obstacle pour la recherche des erreurs et l'organisation du développement**
    - **enfin, HTML est un langage qui n'est pas très économique et dont la syntaxe est nettement différente de celle de PHP, ce qui entraîne des scripts longs et difficiles à décrypter.**
  - **L'organisation du code sans redondance est une tâche importante :**
    - **Organisation par les fonctions qui sont appelées au besoin : on regroupe le code par les fonctionnalités de l'application**
    - **Organisation avec Objets : on regroupe les fonctionnalités autour des structures de données**

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION :
  - Définition : **Une fonction est une partie de code qui ne peut communiquer avec le script appelant que par l'intermédiaire d'un petit nombre de variables – les arguments de la fonction – bien identifiées. Toutes les données utilisées localement par la fonction pour accomplir sa tâche particulière ne sont pas accessibles au script appelant, et, réciproquement, la fonction ne peut pas manipuler les informations du script appelant**
  - Avantages de l'utilisation de fonctions : **La conception de cette structuration vise à deux buts principaux :**
    - ***déléguer* les tâches ingrates, les données secondaires, les contrôles d'erreur à des modules particuliers ;**
    - ***partager* le code : idéalement, on ne devrait *jamais é*crire deux fois la même instruction car cette instruction devrait être implantée par une fonction appelée partout où on en a besoin**

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION (2):
  - Fonction Connexion : **Une fonction prend simplement en entrée des paramètres et renvoie une valeur. Pour illustrer simplement la création d'une fonction, nous allons d´efinir une fonction *Connexion* qui se connecte à MySQL qui sert à traiter tous les cas d'échec de l'appel la fonction système *mysql_pconnect***

```
<?php
if (!isset ($FichierConnexion))
{
$FichierConnexion = 1;
// Fonction Connexion: connexion MySQL
function Connexion ($pU, $pP, $pB, $pS)
{
// Connexion au serveur
$conn = mysql_pconnect ($pS, $pU, $pP);
if (!$conn)
{
echo "Désolé, connexion au serveur
$pS impossible\n";
exit;
}
```

```
// Connexion la base
if (!mysql_select_db ($pB, $conn))
{
echo "Désolé, accès la base
        $pBase impossible\n";
echo "<B>Message de MySQL :</B> «
        . mysql_error($conn);
exit;
}
// On renvoie la variable de connexion
return $conn;
} // Fin de la fonction
} // Fin du test sur $FichierConnexion
?>
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION (3):
  - Fonction Connexion :
    - Test de l'inclusion multiple : **Cette technique permet de résoudre le problème qui pourrait surgir en cas d'inclusion multiple du fichier *Connexion.php* dans un script. Dans un tel cas l'interpréteur rencontrerait plusieurs fois la définition de la fonction et protesterait par une erreur. Avec le test sur *$FichierConnexion*, on ne passera sur la définition de la fonction qu'une seule fois. Si on rencontre à nouveau cette définition dans le même script, la variable *$FichierConnexion* existera et la seconde définition sera ignorée**

```
if (!isset ($FichierConnexion))
{
        $FichierConnexion = 1;
        ...
        // Définition de la fonction
        ...
        }
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION (4):
  - Fonction d'exécution d'une requête : **Selon le même principe, il est possible de définir des fonctions pour exécuter une requête avec MySQL.**
  - **Le fichier *ExecRequete.php* contient deux fonctions, l'une pour exécuter une requête, l'autre pour récupérer les lignes du résultat.**

```
<?php
if (!isset ($FichierExecRequete))
{
$FichierExecRequete = 1;
// Exécution d'une requête avec MySQL
function ExecRequete ($requete, $connexion)
{
$resultat = mysql_query ($requete, $connexion);
if ($resultat)
return $resultat;
else
```

```
{
echo "<B>Erreur dans l'exécution de la requête
        .</B><BR>";
echo "<B>Message de MySQL :</B> «
        . mysql_error($connexion);
exit;
}
} // Fin de la fonction ExecRequete
// Recherche de la ligne suivante
function LigneSuivante ($resultat)
{
return mysql_fetch_object ($resultat);
} // Fin de la fonction LigneSuivante
} // Fin du test
?>
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION (5):
  - Fonction de production de code HTML : **Les fonctions permettent également de coder une fois pour toutes les balises HTML les plus utilisées. La définition de fonctions les plus générales possibles implique d'analyser soigneusement quelles sont les parties paramétrables de ces balises. Par exemple :**
    - **les *ancres* sont définies par l'URL et le libellé placé entre les balises <A>**
    - **les *images* sont définies par le fichier contenant l'image, ainsi que par quelques attributs comme la taille de la bordure, la largeur et la hauteur.**

```
<?php
if (!isset ($FichierHTML))
{
$FichierHTML = 1;
// Fonctions produisant des balises HTML
function Ancre ($url, $libelle, $classe="")
{
return "<A HREF='$url' CLASS=
            '$classe'>$libelle</A>\n";
}
```

```
function Image ($fichier, $largeur=-1,
            $hauteur=-1, $bordure=0)
{
if ($largeur != -1) $attrLargeur = " WIDTH =
            '$largeur %' ";
if ($hauteur != -1) $attrHauteur =
            " HEIGHT = '$hauteur %' ";
return "<IMG SRC='$fichier'" . $attrLargeur
. $attrHauteur . " BORDER='$bordure'>\n";
}
}
?>
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION (6):
  - Fonction de recherche par clé : Une des requêtes les plus courantes effectuées dans une base de données est la recherche d'une ligne dans une table en fonction de la clé d'accès. Des fonctions extrêmement utiles sont donc celles qui, pour une table donnée, exécutent la requête et renvoient le résultat sous sa forme la plus pratique, un objet PHP. Voici par exemple la fonction *ChercheProd* qui prend en argument un identifiant de produit (ainsi que l'identifiant de connexion) et renvoie un objet $produit permettant d'accéder à tous les attributs de la ligne :

```
function ChercheProd ($idProduit, $conn)
{
$idProSain = addSlashes ($idProduit);
$requete = "SELECT * FROM produit
                    WHERE id-produit  = '$idProSain' " ;
$resultat = ExecRequete ($requete, $conn);
return LigneSuivante ($resultat);
}
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION (7):
  - Utilisation de fonctions : **Il existe deux instructions pour inclure du code dans un fichier,** require **et** include **:**
    - require**(*fichier*) se contente d'inclure le code de *fichier* dans le script courant, et tout se passe ensuite comme si l'instruction require avait été définitivement remplacée par le contenu de *fichier* ;**
    - include**(*fichier*), en revanche, correspond à une inclusion répétitive de *fichier*, chaque fois que l'instruction est rencontrée.**

```
<HTML><HEAD> <TITLE>Connexion MySQL</TITLE>
</HEAD> <BODY> <H1>Interrogation de la table FilmSimple</H1>
<?php
require ("Connect.php");
require ("Connexion.php");
require ("ExecRequete.php");
$connexion = Connexion (U, P, B, S);
$resultat = ExecRequete ("SELECT * FROM Produit", $conn);
while ($prod = LigneSuivante ($resultat))
        echo "<B>Le prix du  produit $prod->titre</B>, est $prod->prixU <BR>\n";
?> </BODY></HTML>
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION (8):
  - Module de production de tableau HTML :

```
<?php
if (!isset ($ModuleTable))
{
$ModuleTable = 1;
// Module de production de tableaux HTML
function TblDebut ($pBordure = '1', // La bordure
$pLargeur = '',
$pEspCell = '2', // CELLSPACING
$pRemplCell = '4', // CELLPADDING
$classe = "")
{
echo "<TABLE BORDER='$pBordure' CELLSPACING='$pEspCell' "
. "CELLPADDING='$pRemplCell' WIDTH='$pLargeur %' "
. "CLASS ='$classe'>\n";
} // Fin de la fonction
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION (10):
  - Module de production de tableau HTML (suite) :

```
function TblFin ()
{
echo "</TABLE>\n";
} // Fin de la fonction
function TblDebutLigne ($classe="")
{
echo "<TR CLASS='$classe'>\n";
} // Fin de la fonction
function TblFinLigne ()
{
echo "</TR>\n";
} // Fin de la fonction
function TblEntete ($contenu, $nbLig=1, $nbCol=1)
{
echo "<TH ROWSPAN='$nbLig' COLSPAN='$nbCol'>$contenu</TH>\n";
} // Fin de la fonction
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC FONCTION

```
function TblDebutCellule ($classe="")
{
echo "<TD CLASS='$classe'>\n";
} // Fin de la fonction
function TblFinCellule ()
{
echo "</TD>\n";
} // Fin de la fonction
function TblCellule ($contenu, $nbLig=1, $nbCol=1, $classe="")
{
echo "<TD ROWSPAN='$nbLig' COLSPAN='$nbCol' CLASS='$classe'>"
. "$contenu</TD>\n";
} // Fin de la fonction
} // Fin du module Table
?>
```

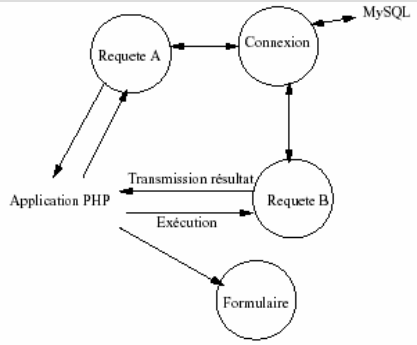
PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS :
  - GENERALITE
    - **PHP ne propose qu'un ensemble très limité de fonctionnalités objet. Cependant, les aspects objet de PHP sont cependant suffisants pour tirer parti de quelques-uns des principaux avantages de la programmation objet.**
    - **Le concept orienté-objet propose une intégration plus poussée des structures de données et des traitements qui leur sont appliqués. Un *objet* peut être simplement défini comme un sous-système, doté de ses propres informations et traitements, qui se charge de fournir des services au reste de l'application. Dans ce sens, l'objet peut-être vu comme l'extension du concept de module**
    - **Dans cette partie, on réduit les études sur quelques exemples illustrés les avantages de la programmation avec objets dans PHP**

PROGRAMMATION SERVEUR WEB
PROGRAMMATION AVEC PHP

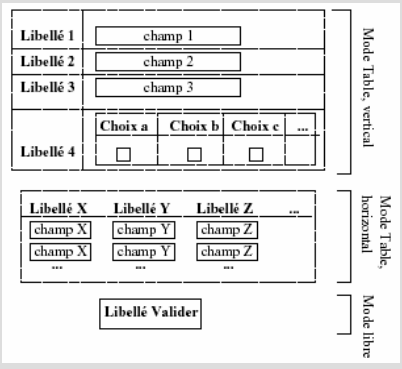
- PROGRAMMATION AVEC OBJETS (2):
  - GENERALITE
    - **Application à étudier : (extrait du cours CNAM Paris de Rivaux)**

Deux types d'objets :
-Les objets de type *Requête* (il peut y en avoir plusieurs simultanément) sont chargés de prendre une requête SQL, de l'exécuter avec MySQL
-Le deuxième type d'objet est le *Formulaire*. Il est chargé de tout ce qui concerne la production de formulaire HTML, et bien entendu il doit s'assurer que le formulaire produit est correct.



PROGRAMMATION SERVEUR WEB
PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (3):
  - Classe Formulaire : **une classe pour générer des formulaires HTML. Outre la création des champs d'un formulaire, cette classe permettra de soigner la présentation des formulaires en alignant les champs et les textes explicatifs à l'aide de tableaux HTML**
    - **La classe Formulaire doit respecter les principes suivants :**
      - tous les champs et boutons seront associés à un libellé qui indique leur utilité ;
      - le positionnement relatif d'un champ (ou d'un bouton) et de son libellé associé pourra être soit libre



PROGRAMMATION SERVEUR WEB
PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (4):
  - Classe Formulaire : **Définition**

```
class Formulaire
{
// ---- Partie privée : les variables
var $modeTable, $orientation;
var $entetes, $champs, $nbChamps, $nbLignes;
// ---- Partie privée : les méthodes
// Constructeur de la classe
function Formulaire ($pMethode, $pAction,
$pTransfertFichier=FALSE, $pNom="Form")
{
...
}
// Méthode pour créer un champ INPUT
général
function champINPUT ($pType, $pNom,
        $pVal, $pTaille)
{
...
}
```

```
...
// Partie publique
function champTexte ($pLibelle, $pNom,
        $pVal, $pTaille)
{
...
}
function champMotDePasse ($pLibelle,
        $pNom, $pVal, $pTaille)
{
...
}
...
}
```

PROGRAMMATION SERVEUR WEB
PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (5):
  - Classe Formulaire : **Constructeur**

```
// Constructeur de la classe
function Formulaire ($pMethode, $pAction,
$pTransfertFichier=FALSE, $pNom="Form")
{
$this->modeTable = FALSE;
// Mettre un attribut ENCTYPE si on transfère un fichier
if ($pTransfertFichier)
$encType = "ENCTYPE='multipart/form-data'";
// Ouverture de la balise
echo "<CENTER><FORM METHOD='$pMethode' " . $encType
. "ACTION='$pAction' NAME='$pNom'>\n";
}
```

```
// Exemple d'instanciation d'un objet

$f = new Formulaire ("POST", "InsertionFilm.php", TRUE);
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (6):
  - Classe Formulaire : **Méthodes de Production des champs de formulaire**

```
// méthode pour créer un champ INPUT général
function champINPUT ($pType, $pNom, $pVal, $pTaille, $pTailleMax)
{
$s = "<INPUT TYPE='$pType' NAME='$pNom' "  . "VALUE='$pVal'
        SIZE='$pTaille' MAXLENGTH='$pTailleMax'>\n";
return $s;
}
// Champ pour sélectionner dans une liste
function champSELECT ($pNom, $pListe, $pDefaut, $pTaille=1)
{
$s = "<SELECT NAME='$pNom' SIZE=$pTaille>\n";
while (list ($val, $libelle) = each ($pListe))
{
if ($val != $pDefaut) $s .= "<OPTION VALUE='$val'>$libelle</OPTION>\n";
else $s .= "<OPTION VALUE='$val' SELECTED>$libelle</OPTION>\n";
}
return $s . "</SELECT>\n";
}
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (7):
  - Classe Formulaire : **Méthodes de Production des champs de formulaire**

```
// Champ de formulaire
function champForm ($pType, $pNom,
        $pVal, $params, $pListe=array())
{
switch ($pType)
{
case "TEXT": case "PASSWORD": case "SUBMIT":
case "RESET": case "FILE":
$taille = $params["SIZE"];
$tailleMax = $params["MAXLENGTH"];
if ($tailleMax == 0) $tailleMax = $taille;
// Appel de la méthode champINPUT de l'objet courant
$champ = $this->champINPUT ($pType, $pNom,
$pVal, $taille, $tailleMax);
break;
case "TEXTAREA":
$lig = $params["ROWS"]; $col = $params["COLS"];
// Appel de la méthode champTEXTAREA
// de l'objet courant
$champ = $this->champTEXTAREA ($pNom,
$pVal, $lig, $col);
break;
```

```
case "SELECT":
$taille = $params["SIZE"];
// Appel de la méthode champSELECT
de l'objet courant
$champ = $this->champSelect ($pNom,
$pListe, $pVal, $taille);
break;
case "CHECKBOX": case "RADIO":
// Appel de la méthode champBUTTONS
de l'objet courant
$champ = $this->champBUTTONS ($pType,
 $pNom, $pListe, $pVal);
break;
default: echo "<B>ERREUR: $pType est un
type inconnu</B>\n";
break;
}
return $champ;
}
```

PROGRAMMATION SERVEUR WEB
PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (8):
  - Classe Formulaire : **Méthodes d'affichage des champs de formulaire**

```
// Affichage d'un champ avec son libellé
function champLibelle ($pLibelle, $pNom, $pVal,
$pType="TEXT", $params=array(), $pListe=array())
{
// Création du champ
$champHTML = $this->champForm ($pType,
$pNom, $pVal, $params, $pListe);
// Affichage du champ en tenant compte de la
 présentation
if ($this->modeTable)
{
if ($this->orientation == VERTICAL)
{
// Nouvelle ligne, avec libellé et champ dans
//deux cellules
TblDebutLigne();
TblCellule ("<B>" . $pLibelle . "</B>");
TblCellule ($champHTML);
TblFinLigne();
}
```

```
else
{
// On ne peut pas afficher maintenant :
//on stocke dans les tableaux
$this->entetes[$this->nbChamps] =
        "<B>" . $pLibelle . "</B>";
$this->champs[$this->nbChamps] =
          $champHTML;
$this->nbChamps++;
}
}
else
{
// Affichage simple
echo "$pLibelle ";
echo $champHTML;
}
}
```

PROGRAMMATION SERVEUR WEB
PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (9):
  - Classe Formulaire : Méthodes publiques

```
function champTexte ($pLibelle, $pNom,
$pVal, $pTaille, $pTailleMax=0)
{
$this->champLibelle ($pLibelle, $pNom,
$pVal,"TEXT", array ("SIZE"=>$pTaille,
"MAXLENGTH"=>$pTailleMax));
}
function champRadio ($pLibelle, $pNom,
$pVal, $pListe)
{
$this->champLibelle ($pLibelle, $pNom, $pVal,
"RADIO", array (), $pListe);
}
function champFenetre ($pLibelle, $pNom,
$pVal, $pLig, $pCol)
{
$this->champLibelle ($pLibelle, $pNom, $pVal,
"TEXTAREA",
array ("ROWS"=>$pLig,"COLS"=>$pCol));
}
```

```
// Début d'une table, mode horizontal ou vertical
function debutTable ($pOrientation=VERTICAL,
$pNbLignes=1)
{
// Pas de bordure
pOrientation == VERTICAL) TblDebut (0);
$this->modeTable = TRUE;
$this->orientation = $pOrientation;
$this->nbLignes = $pNbLignes;
$this->nbChamps = 0;
}
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (10):
  - Classe Formulaire : Méthodes publiques

```
// Fin d'un tableau
function finTable ()
{
if ($this->modeTable == TRUE)
{
if ($this->orientation == HORIZONTAL)
{
// Affichage des libelles
TblDebut(0); TblDebutLigne ();
// Les entêtes du tableau
for ($i=0; $i < $this->nbChamps; $i++)
TblCellule ($this->entetes[$i]);
TblFinLigne();
// Affichage des lignes et colonnes
for ($j=0; $j < $this->nbLignes; $j++)
{
TblDebutLigne ();
```

```
for ($i=0; $i < $this->nbChamps; $i++)
TblCellule ($this->champs[$i]);
TblFinLigne();
}
}
TblFin();
}
$this->modeTable = FALSE;
}
// Fin du formulaire
function fin ()
{
// Fin de la table, au cas o`
$this->finTable();
echo "</FORM></CENTER>\n";
}
```

PROGRAMMATION SERVEUR WEB PROGRAMMATION AVEC PHP

- PROGRAMMATION AVEC OBJETS (11):
  - Classe Formulaire : Utilisation de la classe

    Exemple

```
<?php
// Formulaire de saisie d'une requête SQL
require ("Formulaire.class");
function FormSQL ($requeteDefaut)
{
$form = new Formulaire ("POST", "ExecSQL.php");
$form->champFenetre ("", "requete", $requeteDefaut, 5, 50);
echo "<P>";
$form->champValider ("Exécuter la requête", "exesql");
$form->fin();
}
?>
```

TD6 : schéma de base de données

- schéma relationnel est le suivant :
  - EMPLOYE (id_emp, nom, prenom, poste, salaire , *id_dept*)
  - DEPT (id_dept, nom , loc)
- Les clés primaires d'EMPLOYE et DEPT sont respectivement id_emp et id_dept.
- Le champ id_dept dans EMPLOYE est une clé étrangère qui fait référence à la clé primaire id_dept de DEPT.

TD6 : Test de connexion

```
<?php
        $user="root";
        $pass="";
        $host="localhost";
        //connexion au serveur
        $id_connexion = mysql_connect($host,$user,$pass);
        if ( !$id_connexion )
        {
                echo "<script type=text/javascript>";
                echo "alert ('Connexion impossible à la base')</script>";
        }
        //sélection de la base du personnel, remplacez par le nom de votre base
        if ( !mysql_select_db("personnel"))
        {
                echo "<script type=text/javascript>";
                echo "alert ('Base introuvable')</script>";
        }
        echo "Connexion réussie";
?>
```

TD6 : Test de requête d'accès (1)

- Créez le script test_select.php permettant de lister les noms, prénoms, postes, département et les salaires des employés triés par noms.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test bd: selection</title>
</head><body>
<?php
        require("test_connexion.php");
        $requete= 'select employe . nom , employe . prenom , employe . poste , dept . nom , employe .
Salaire from employe , dept where employe . id_dept = dept . id_dept order by employe . nom ';
        $result=mysql_query($requete,$id_connexion);
        if(!$result)
        {
                echo "Lecture impossible";
        }
```

TD6 : Test de requête d'accès (2)

```
        else
        {
                echo "<h3>Voici la liste de nos employés triée par ordre
alphabétique:</h3>";
                echo "<table> <th>Nom</th>
<th>Prenom</th><th>Poste</th><th>Département</th><th>Salaire</th>";
                while($ligne=mysql_fetch_array($result,MYSQL_NUM))
                {
                        echo "<tr>";
                        foreach($ligne as $valeur){
                                echo "<td> $valeur </td>";      }
                        echo "</tr>";
                }
                echo "</table>";
                mysql_free_result($result);
                mysql_close($id_connexion);
        }
?></body></html>
```

TD6 : Test de requête d'accès (3)

- Créez le script test_update.php permettant d'augmenter de 10% le salaire des employés.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test bd: mise à jour</title>
</head><body>
<?php
        require("test_connexion.php");
        $requete= 'UPDATE employe set salaire=salaire*1.1';
        $result=mysql_query($requete,$id_connexion);
        mysql_close($id_connexion);
        if(!$result){
                echo "Erreur: ".mysql_error();        }
        else{
                echo "Mise à jour de la table terminée.";          }
?></body></html>
```

TD6 : Test de requête de MAJ

- Créez le script test_insert.php permettant d'ajouter une ligne dans la table EMP. Testez les cas d'erreurs suivants : l'employé est déjà présent, le no de département n'existe pas.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test bd: mise à jour</title>
</head><body>
<?php
        require("test_connexion.php");
        $requete= "INSERT INTO `dept` (`Nom`, `Loc`) VALUES ('Support', 'Cannes')";
        $result=mysql_query($requete,$id_connexion);
        mysql_close($id_connexion);
        if(!$result){
                echo "Erreur: ".mysql_error();        }
        else{
                echo "Insertion terminée.";
        }
?></body></html>
```

TD6 : Formulaire (0)

- Fichier form.php :
- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
- "http://www.w3.org/TR/REC-html40/strict.dtd">
- <html>
- <head>
- <title>Modifiez vos coordonnées</title>
- <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
- </head>
- <body>
- <form action="modif.php" method="post" enctype="application/x-www-form-urlencoded">
- <fieldset><legend>Saisissez votre code client pour modifier vos coordonnées</legend><table>
- <tr> <td>Code client : </td><td><input type="text" name="code" size="20" maxlength="10"/></td></tr>
- <tr><td>Modifier : </td> <td><input type="submit" value="Modifier"/></td></tr></table></fieldset></form>
-
- </body>
- </html>

TD6 : Formulaire (1)

- Créez un script qui permet de modifier un employé par l'intermédiaire d'un formulaire. A partir de l'id employé, on affichera les informations actuelles dans un formulaire, puis on enregistrera les modifications.
- <?php
- if(empty($_POST['code'])){header("Location:form.php");}
- ?>
- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
- "http://www.w3.org/TR/REC-html40/strict.dtd">
- <html>
- <head>
- <title>Modifiez vos coordonnées</title>
- <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

TD6 : Formulaire (2)

```
//Création du formulaire
 echo "<form action= \"". $_SERVER['PHP_SELF']."\" method=\"post\"enctype=\"application/x-www-form-urlencoded\">";
 echo "<fieldset>";
 echo "<legend><b>Modifiez vos coordonnées</b></legend>";
 echo "<table>";
 echo "<tr><td>Nom : </td><td><input type=\"text\" name=\"nom\" size=\"40\" maxlength=\"30\" value=\"$coord[1]\"/> </td></tr>";
 echo "<tr><td>Prénom : </td><td><input type=\"text\" name=\"prenom\" size=\"40\" maxlength=\"30\" value=\"$coord[2]\"/></td></tr>";
 echo "<tr><td>Poste : </td><td><input type=\"text\" name=\"poste\" size=\"40\" maxlength=\"30\" value=\"$coord[3]\"/></td></tr>";
 echo "<tr><td>Salaire : </td><td><input type=\"text\" name=\"salaire\" size=\"40\" maxlength=\"60\" value=\"$coord[4]\"/></td></tr>";
  echo "<tr><td><input type=\"reset\" value=\" Effacer \"></td> <td><input type=\"submit\" name=\"modif\" value=\"Enregistrer\"></td></tr></table>";
 echo "</fieldset>";
 echo "<input type=\"hidden\" name=\"code\"  value=\"$code\"/>";
 echo "</form>";
}
elseif(isset($_POST['nom'])&& isset($_POST['prenom'])&& isset($_POST['poste'])&& isset($_POST['salaire']))
```

TD6 : Formulaire (3)

```
//ENREGISTREMENT
 require('test_connexion.php');
 $nom=mysql_escape_string($_POST['nom']);
 $prenom=mysql_escape_string($_POST['prenom']);
 $poste=mysql_escape_string($_POST['poste']);
 $salaire=mysql_escape_string($_POST['salaire']);
 $code=mysql_escape_string($_POST['code']);
 //Requète SQL
        $requete="UPDATE employe SET nom='$nom',prenom='$prenom',poste='$poste',salaire='$salaire' WHERE id_emp='$code'";
        $result=mysql_query($requete,$id_connexion);
        mysql_close($id_connexion);
        if(!$result)
        { echo "<script type=\"text/javascript\">
         alert('Erreur : ".mysql_error()."')</script>";
         }
        else
        {  echo "<script type=\"text/javascript\"> alert('Vos modifications sont enregistrées')
window.location='test_select.php';</script>";
        }
}
?>
</body> </html>
```