

Conception des bases de données III : Applications de bases de données

bdd3.pdf



STÉPHANE CROZAT

Table des matières

I - Implémentation de bases de données relationnelles avec PostgreSQL sous Linux	4
A. Cours.....	4
1. Introduction à Linux.....	4
2. Introduction à PostgreSQL : présentation, installation, manipulations de base.....	8
3. Éléments complémentaires indispensables à l'utilisation de PostgreSQL.....	10
B. Exercices.....	17
1. Découverte de la ligne de commande sous Linux.....	17
2. Découverte d'un SGBDR avec PostgreSQL.....	20
II - Application de bases de données, principes et exemples avec LAPP	24
A. Cours.....	24
1. Applications et bases de données.....	24
2. Architecture Web.....	31
B. Exercice.....	35
1. Tester un environnement LAPP sur son ordinateur personnel.....	35
C. Devoir.....	38
1. Déployez un site web sur Internet avec Filezilla.....	38
2. Déployez un site web sur un serveur de l'UTC.....	40
III - Introduction à HTML et PHP	41
A. Cours.....	41
1. Introduction à HTML et XHTML.....	41
2. Introduction à PHP.....	43
B. Exercices.....	49
1. Population.....	49
2. Hello PHP !.....	50
C. Devoirs.....	50
1. Deux fois deux.....	50
IV - Requêtes HTTP avec Apache sous Linux	51
A. Cours.....	51
1. Utiliser une machine Linux.....	51
2. Introduction au protocole HTTP.....	56
3. Requêtes HTTP avec une page web.....	58
4. Complément.....	60
B. Exercices.....	64
1. Linux.....	64

2. HTTP.....	65
3. XHTML, HTTP, PHP.....	66

V - Requêtes SQL avec PHP et PostgreSQL 68

A. Cours.....	68
1. Connexion d'une page PHP à une base de données PostgreSQL avec PDO.....	68
2. Complément.....	71
B. Exercices.....	77
1. Super-transferts.....	77
2. À l'école de musique.....	78
C. Devoirs.....	80
1. Recensement.....	80
2. Devoirs en ligne.....	81

Glossaire 84

Signification des abréviations 86

Références 87

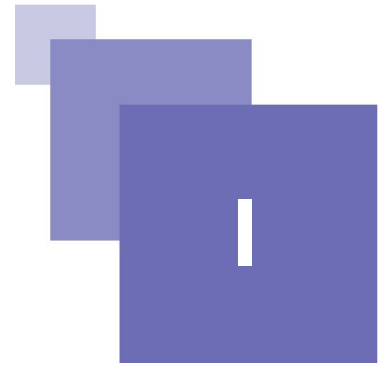
Bibliographie 88

Webographie 89

Index 90



Implémentation de bases de données relationnelles avec PostgreSQL sous Linux



A. Cours

1. Introduction à Linux

a) Linux en deux mots

- Linux est un système d'exploitation né en 1991.
- C'est **le premier système utilisé pour les serveur informatiques** (devant Windows et d'autres UNIX).
- C'est le troisième système utilisé pour les ordinateurs personnels (derrière Windows et MacOSX)
- C'est un système libre et gratuit.
- Il fait partie de la famille des Unix, à l'instar d'Android, MacOSX ou FreeBSD.

Définition : Distribution Linux

On appelle distribution Linux un ensemble de logiciels composé d'un système Linux et de logiciels complémentaires pré-installés et pré-paramétrés, typiquement : une interface graphique de gestion de fichiers, des suite bureautiques, des outils internet, des logiciels multimédia...

Exemple de distributions :

- Debian
- Ubuntu
- Xubuntu
- Fedora

- ...

Remarque : Linux ou GNU/Linux

GNU/Linux est la réunion de deux parties, le projet GNU de Richard Stallman et le projet Linux de Linus Torvalds.

Le nom Linux désigne en général le système d'exploitation dont le nom complet est GNU/Linux. On utilisera l'un pour l'autre dans le cadre de ce cours.

b) Utiliser Linux

Pour utiliser Linux, il y a plusieurs possibilités :

- Avoir accès à un PC sur lequel Linux est déjà installé (il suffit de disposer d'un compte utilisateur sur cet ordinateur).
- Installer Linux sur son ordinateur :
 - En téléchargeant et installant une distribution (il faut savoir préalablement graver un DVD ou créer une clé USB *bootable*) ;
 - En commandant un DVD ou une clé USB (coût de quelques euros) ;
 - En participant à une *install party* (organisées par des associations, elles permettent de se faire aider dans le processus d'installation et la prise en main initiale de l'environnement).
- Installer Linux sur son ordinateur en *double-boot* (à côté de son OS initial, on choisit au démarrage quel système on utilise).
- Installer Linux sur son ordinateur dans une machine virtuelle (par exemple : on peut utiliser Linux dans une fenêtre Windows).
- Installer Linux sur une clé en version *live USB persistant*.

Exemple : Installer Xubuntu sur son PC

<http://xubuntu.fr/>¹

Exemple : Installer Ubuntu dans une machine virtuelle sous VirtualBox sous Windows

<https://openclassrooms.com/courses/reprenez-le-controle-a-l-aide-de-linux/installez-linux-dans-une-machine-virtuelle>²

c) Le terminal

Les distributions Linux comportent souvent un mode graphique, pratique pour de nombreuses opérations.

Savoir utiliser un terminal en lignes de commandes n'est donc pas indispensable, mais cela présente des avantages comme :

- connaître des opérations qui seront reproductibles sur toutes les distributions (voire sur d'autres Unix) ;
- savoir utiliser un serveur à distance (via SSH) ;
- savoir échanger ou reproduire des procédures (sans avoir besoin de refaire une succession de manipulations à la souris).

Fondamental

Dès lors qu'on se connecte à un système Linux on peut ouvrir un terminal et faire de nombreuses opérations, telles que :

- **lancer des applications,**

1 - <http://xubuntu.fr/>

2 - <https://openclassrooms.com/courses/reprenez-le-controle-a-l-aide-de-linux/installez-linux-dans-une-machine-virtuelle>

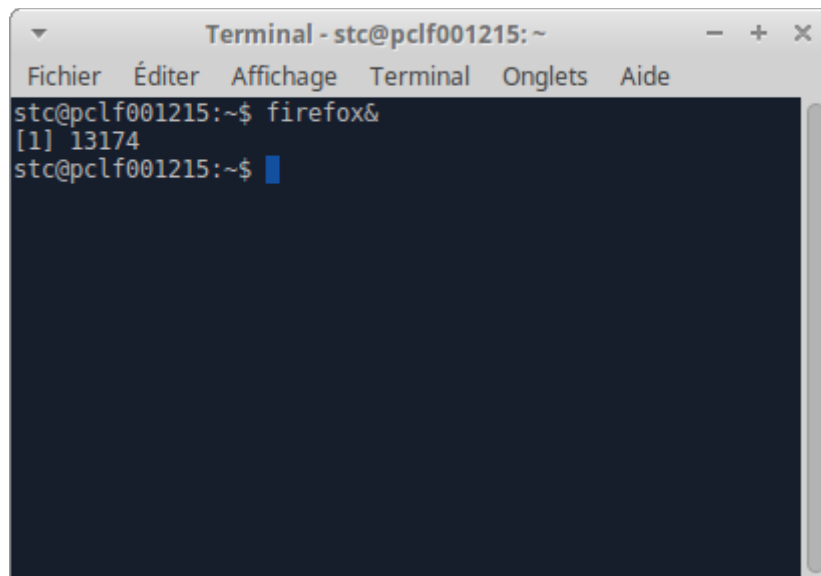
- **manipuler des fichiers,**
- **se connecter à un autre ordinateur.**

d) Lancer des applications sous Linux

Méthode

Le plus générique pour lancer une application sous Linux consiste à :

1. ouvrir un terminal,
2. écrire le nom de l'application, par exemple `firefox`,
3. saisir le caractère `&` et appuyer sur entrée.



```

Terminal - stc@pclf001215: ~
Fichier Éditer Affichage Terminal Onglets Aide
stc@pclf001215:~$ firefox&
[1] 13174
stc@pclf001215:~$

```

Conseil : La touche TAB

Il n'est pas en général nécessaire de saisir l'intégralité du nom du programme : on saisit le début, par exemple `fir` et on appuie sur la touche `TAB`. Dès que le système identifie l'application recherchée, il complète automatiquement avec les caractères manquants.

e) Commandes de bases sous Linux : `cd`, `ls`, `mkdir`, `rm`, `find`, `cat`, `nano`...

Le manuel !

La commande `man` permet d'afficher le manuel d'une autre commande.

Par exemple `man ls` permet d'afficher les options de la commande `ls`, et `man man` est l'affichage du manuel du manuel.

Organisation des fichiers

Les fichiers sont organisés sous Linux dans des dossiers (ou répertoires) arborescents (il n'y a pas de notion de disques).

Le premier de ces dossiers, appelé racine, est `/`.

Des fils courants de racines sont :

- `/bin` qui contient des programmes du système ;
- `/home` qui contient les données des utilisateurs ;
- `/tmp` qui contient des données volatiles accessibles à tous.

Gestion de fichiers

- `pwd` savoir où je me situe dans l'arborescence
- `cd` aller quelque part dans l'arborescence
 - `cd /home` aller dans /home (déplacement absolu)
 - `cd me` aller dans le répertoire me fils de mon répertoire courant (déplacement relatif)
 - `cd ..` remonter dans l'arborescence (déplacement vers son père)
 - `cd ~` permet de retourner dans son dossier initial (retour au domicile)
- `ls` voir les fichiers et dossiers dans mon dossier actuel (sauf les fichiers cachés commençant par un .)
- `ls -al` voir tous les fichiers et dossiers dans mon dossier actuel avec leurs informations associées (vue détaillée)
- `touch file` créer un fichier file
- `rm file` supprimer le fichier file dans mon dossier actuel
- `mkdir dir` créer un nouveau dossier dir dans mon dossier actuel
- `rm *` supprimer tous les fichiers de mon dossier actuel
- `rm -R dir` supprimer le dossier dir dans mon dossier actuel
- `cat file` afficher le contenu du fichier file
- `more` afficher le contenu du fichier file en mode paginé
- `less` afficher le contenu du fichier file en mode défilement

Édition de fichiers

- `nano file`
 - Éditeur dans le terminal (simple d'utilisation)
 - Les commandes sont indiqués en bas de l'éditeur
- `gedit file &`
 - Éditeur graphique
 - Utiliser une extension de fichier standard ou le menu `Affichage > Mode de coloration` pour obtenir une visualisation adapté au type de fichier édité (par exemple `.sql` pour un fichier SQL)

Rechercher un fichier

- `find / -name '*test*' >` permet de rechercher un fichier contenant la chaîne test sur tout le disque
- `find ~ -name '*test*' >` permet de rechercher un fichier contenant la chaîne test dans son espace personnel

Complément

Agir en tant que root (`su / sudo`)

Installer des applications sous Linux

f) Éteindre sa machine

Exemple

La commande `shutdown` permet d'éteindre sa machine en ligne de commande :

- `shutdown -h now` (éteindre tout de suite)
- `shutdown -h 23:00` (éteindre à 23h00)
- `shutdown -h +60` (éteindre dans 60 minutes)

2. Introduction à PostgreSQL : présentation, installation, manipulations de base

a) Présentation de PostgreSQL

PostgreSQL est :

- un SGBDR
- libre (licence BSD)
- multi-plate-formes (Unix, Linux, Windows, MacOS, ...)
- puissant
- très respectueux du standard
- très bien documenté

Fondamental : Documentation de PostgreSQL

<https://www.postgresql.org/docs/>³

(en français : <http://docs.postgresqlfr.org/>⁴)

Fonctionnement général

PostgreSQL comme la grande majorité des SGBD est un logiciel client-serveur. Il faut donc pour l'utiliser un serveur (programme *postgres* sous Linux) et un client.

Il existe plusieurs clients, les deux plus utilisés sont le client textuel *psql* et le client graphique *PgAdmin III*.

Complément

- <http://www.postgresql.org/>⁵
- <http://www.postgresql.fr/>⁶

b) Installation de PostgreSQL

Attention

Nous présentons ici une installation *a minima* de PostgreSQL uniquement à des fins de test et d'apprentissage, et pour un usage local. Pour mettre en production une base de données PostgreSQL sur le réseau, il faut suivre des directives supplémentaires, notamment pour assurer la sécurité du système, sa sauvegarde...

Ces thèmes ne sont pas abordés dans le cadre de ce cours.

PostgreSQL est à l'origine une base de données conçue pour Unix, son installation et son fonctionnement sont possibles aujourd'hui sur les trois OS, mais Linux reste son environnement de prédilection. C'est l'architecture PostgreSQL sur Linux qui est étudiée ici.

<https://www.postgresql.org/download/>⁷

Exemple : Installer PostgreSQL sur Ubuntu

Une installation sur Ubuntu est très facile :

1. Installer les paquets : `sudo apt-get install postgresql`
2. Activer l'utilisateur *postgres* : `sudo passwd postgres`
3. Se connecter en tant que *postgres* : `su postgres`
4. Lancer le client *psql* : `psql -h localhost -d postgres -U postgres`

3 - <https://www.postgresql.org/docs/>

4 - <http://docs.postgresqlfr.org/>

5 - <http://www.postgresql.org/>

6 - <http://www.postgresql.fr/>

7 - <https://www.postgresql.org/download/>

Complément

<https://doc.ubuntu-fr.org/postgresql>⁸

Complément : Base de données par défaut

L'installation crée une base de données par défaut qui s'appelle *postgres* et un utilisateur *postgres* qui possède cette base (OWNER).

Complément : Informations réseau

- Le port standard de communication de PostgreSQL est 5432.
- Si le client et le serveur sont sur la même machine, le client peut bien entendu se connecter au serveur *localhost* sur le port 5432.

Complément : Installer PostgreSQL sous Windows

1. Télécharger un *installer* depuis <http://www.enterprisedb.com/products-services-training/pgdownload#windows>⁹
2. Exécuter l'installation en validant les propositions par défaut (et sans installer les éventuels programmes complémentaires proposés à l'issue de l'installation)
3. Exécuter le client *psql* (également appelé *Shell SQL*)

Complément : SHOW ALL

La commande `SHOW ALL` permet de voir tous les paramètres du serveur PostgreSQL.

Par exemple `data_directory` permet de connaître le répertoire de stockage utilisé sur le disque dur.

c) Le client textuel "psql"

Définition : psql

`psql` est le client textuel de PostgreSQL.

Syntaxe : Connexion à un serveur PostgreSQL avec le client psql

```
1 psql -h serveraddress -d database -U user
```

Rappel : Connexion à la base "postgres" avec l'utilisateur "postgres" sur l'ordinateur local

```
1 psql -h localhost -d postgres -U postgres
```

Complément

La commande *psql* utilisée sans paramètre se connecte au serveur *localhost* avec pour nom de *database* **et** pour nom de *user* le nom de l'utilisateur du système qui invoque la commande (utilisateur Linux typiquement).

`me@mypc:~$ psql` équivaut à `: me@mypc:~$ psql -h localhost -d me -U me`

Syntaxe : Écrire une instruction SQL

```
1 dbnf17p015=> SELECT * FROM maTable ;
```

8 - <https://doc.ubuntu-fr.org/postgresql>

9 - <http://www.enterprisedb.com/products-services-training/pgdownload#windows>

Syntaxe : Écrire une instruction SQL sur plusieurs lignes

Une instruction SQL peut s'écrire sur une ou plusieurs lignes, le retour chariot n'a pas d'incidence sur la requête, c'est le ; qui marque la fin de l'instruction SQL et provoque son exécution.

```

1 dbnf17p015=> SELECT *
2 dbnf17p015-> FROM maTable
3 dbnf17p015-> ;

```

On notera dans `psql` la différence entre les caractères `=>` et `->` selon que l'on a ou pas effectué un retour chariot.

Fondamental : Commandes de base : aide

`\?` : Liste des commandes `psql`

`\h` : Liste des instructions SQL

`\h CREATE TABLE` : Description de l'instruction SQL `CREATE TABLE`

Fondamental : Commandes de base : catalogue

`\d` : Liste des relations (catalogue de données)

`\d maTable` : Description de la relation `maTable`

Fondamental : Commandes de base : quitter

`\q` : Quitter `psql`

Complément

<http://www.postgresql.org/docs/current/static/app-psql.html>¹⁰

d) Types de données

Types standards

- numériques : integer (int2, int4, int8), real (float4, float8)
- dates : date (time, timestamp)
- chaînes : char, varchar, text
- autres : boolean

Complément : Documentation

<http://docs.postgresqlfr.org/8.1/datatype.html>¹¹

3. Éléments complémentaires indispensables à l'utilisation de PostgreSQL

a) Exécuter des instructions SQL depuis un fichier

Il est souvent intéressant d'exécuter un fichier contenant une liste de commandes SQL, plutôt que de les entrer une par une dans le terminal. Cela permet en particulier de recréer une base de données à partir du script de création des tables.

10 - <http://www.postgresql.org/docs/current/static/app-psql.html>

11 - <http://docs.postgresqlfr.org/8.1/datatype.html>

Syntaxe

Pour exécuter un fichier contenant du code SQL utiliser la commande PostgreSQL `\i chemin/fichier.sql`

- `chemin` désigne le répertoire dans lequel est le fichier `fichier.sql`
- le dossier de travail de `psql` est le dossier dans lequel il a été lancé, le script peut être lancé à partir de son dossier `home` pour en être indépendant (`~/.../fichier.sql`)
- chaque commande doit être terminée par un `;`

```
1 dbnf17p015=> \i /home/me/bdd.sql
```

b) Importer un fichier CSV

Syntaxe

```
1 \copy nom_table (att1, att2, ...) FROM 'fichier.csv' WITH CSV DELIMITER ';'
   QUOTE ''
```

- `WITH` introduit les options de l'import
- `CSV` indique qu'il s'agit d'un fichier CSV
- `DELIMITER 'c'` indique que le caractère `c` est utilisé comme délimiteur de champ (en général `;` ou `,`)
- `QUOTE 'c'` indique que le caractère `c` est utilisé comme délimiteur de chaîne (en général `"`)

Remarque

- La table `nom_table` doit déjà exister
- Le nombre de colonnes spécifié doit correspondre au nombre de colonnes du fichier CSV
- Les types doivent être compatibles

Remarque

Ajouter l'option `HEADER` après `WITH CSV` si le fichier CSV contient une ligne s'entête.

```
1 \copy nom_table (att1, att2, ...) FROM 'fichier.csv' WITH CSV HEADER DELIMITER
   ';' QUOTE ''
```

Méthode : Localisation du fichier CSV depuis psql

Par défaut, la commande `\copy` prendra le chemin du répertoire courant au moment où la commande `psql` a été lancée.

Sous `psql`, vous pouvez utiliser les commandes :

- `dbnf17p007=> \! pwd`
Pour exécuter la commande `shell` `pwd` et obtenir le répertoire courant
- `dbnf17p007=> \cd directory`
Pour changer le répertoire courant

Complément

PostgreSQL sous Linux

Fichier CSV

c) Notion de schéma

Définition

« A PostgreSQL database cluster contains one or more named databases. Users and groups of users are shared across the entire cluster, but no other data is shared across databases. Any given client connection to the server can access only the data in a single database, the one specified in the connection request.

A database contains one or more named schemas, which in turn contain tables. Schemas also contain other kinds of named objects, including data types, functions, and operators. The same object name can be used in different schemas without conflict; for example, both `schema1` and `myschema` can contain tables named `mytable`. Unlike databases, schemas are not rigidly separated: a user can access objects in any of the schemas in the database he is connected to, if he has privileges to do so.

<http://www.postgresql.org/docs/8.4/static/ddl-schemas.html>¹²



Graphique 1 Organisation en cluster, base, schéma, table dans PostgreSQL

Syntaxe : Créer un schéma

```
1 CREATE SCHEMA myschema;
```

Syntaxe : Créer une table dans un schéma

```
1 CREATE TABLE myschema.mytable (  
2 ...  
3 );
```

Syntaxe : Requêter dans un schéma

```
1 SELECT ...
```

12 - <http://www.postgresql.org/docs/8.4/static/ddl-schemas.html>

```
2 FROM myschema.mytable
```

Exemple

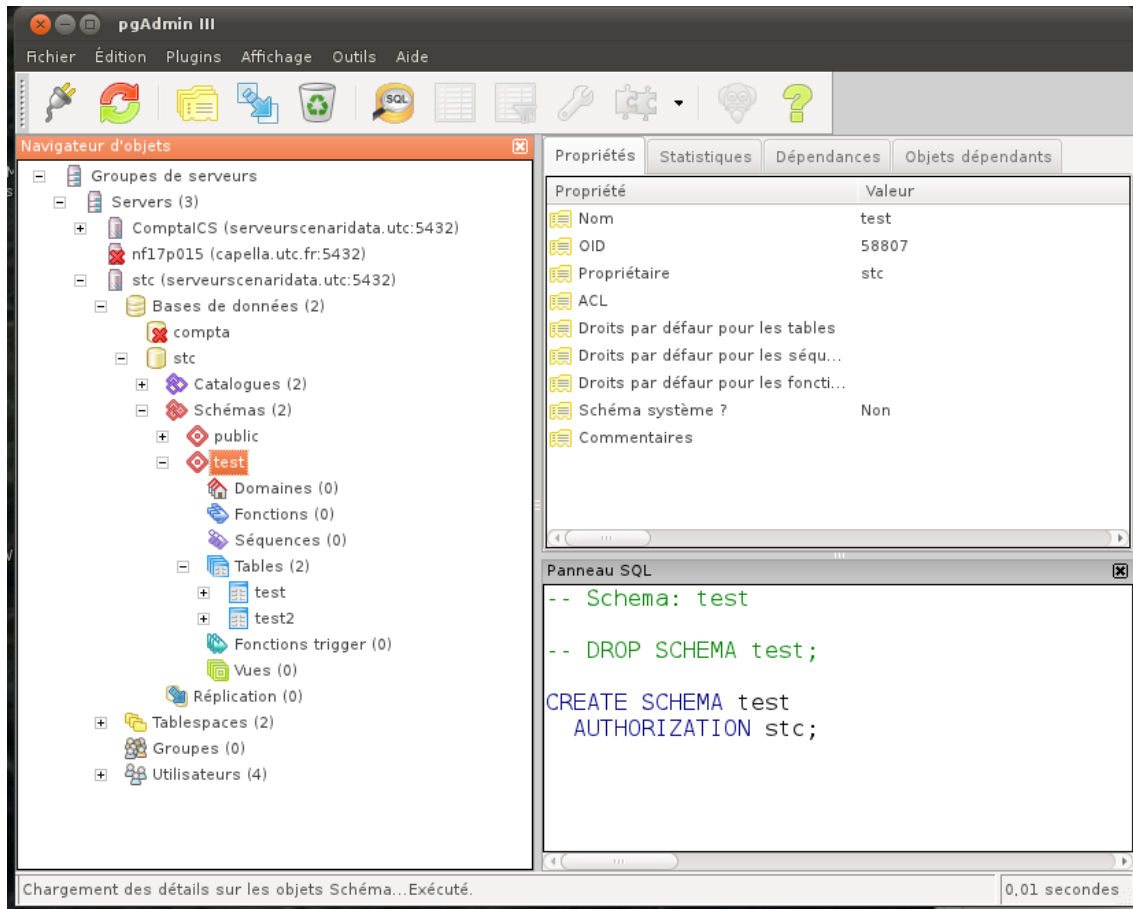


Image 1 Schéma sous PostgreSQL

Syntaxe : Catalogue : schémas

\dn : Liste des schémas de ma base de données

Complément : Schéma par défaut : search_path

Afin d'alléger la syntaxe il est possible de définir un schéma par défaut, dans lequel seront créés les tables non-préfixées et un ou plusieurs schémas par défaut dans lesquels seront requêtées les tables non-préfixées.

```
1 SET search_path TO myschema,public;
```

Cette instruction définit le schéma `myschema` comme schéma par défaut pour la création de table et le requêtage, puis `public` pour le requêtage, le premier étant prioritaire sur le second :

- `CREATE mytable` créera ma `mytable` dans le schéma `myschema`.
- `SELECT FROM mytable` cherchera la table dans le schéma `myschema`, puis dans le schéma `public` si la table n'existe pas dans le premier schéma.

Remarque : Schéma "public"

Le schéma `public` est un schéma créé par défaut à l'initialisation de la base, et qui sert de schéma par défaut en l'absence de toute autre spécification.

Remarque : Catalogue : \d

La liste des tables retournée par `\d` dépend du `search_path`. Pour que `\d` retourne les tables de `schema1`, il faut donc exécuter l'instruction :

```
SET search_path TO public, schema1
```

Complément : Pour aller plus loin

<http://www.postgresql.org/docs/8.4/static/ddl-schemas.html>¹³

d) Créer des bases de données et des utilisateurs

Syntaxe : Créer un utilisateur

```
1 CREATE USER user1 WITH ENCRYPTED PASSWORD 'password';
```

Syntaxe : Créer une base de données

```
1 CREATE DATABASE mydb WITH OWNER user1;
2
```

Complément : Supprimer des bases de données et des utilisateurs

```
1 DROP DATABASE mydb;
2 DROP USER user1;
```

Complément : Modifier le mot de passe d'un utilisateur

```
1 ALTER USER user1 WITH ENCRYPTED PASSWORD 'mypassword'
```

Syntaxe : Catalogue : utilisateurs et bases de données

- `\du` : liste des utilisateurs
- `\l` : liste des bases de données

Syntaxe : psql : changer d'utilisateur

`\c db user` : se connecter à la base `db` avec le compte `user`.

e) PostgreSQL sous Linux

Syntaxe : Exécuter une instruction PostgreSQL depuis Linux

```
1 psql -c "instruction psql"
```

Exemple : Exécuter un script PostgreSQL depuis Linux

```
1 #!/bin/sh
2 psql -c "DROP DATABASE mydb"
3 psql -c "CREATE DATABASE mydb"
4 psql -c "GRANT ALL PRIVILEGES ON DATABASE mydb TO user1"
```

Complément : psql : exécuter une commande Linux

- `\!` : permet d'exécuter certaines commandes du `shell` Linux depuis le client `psql`.

13 - <http://www.postgresql.org/docs/8.4/static/ddl-schemas.html>

f) Les clients graphiques pgAdminIII et phpPgAdmin

pgAdminIII

Un client graphique une interface graphique permettant d'effectuer les mêmes opérations qu'avec le client `psql`.

- Le client graphique pgAdminIII est un client lourd qui fonctionne très bien sous Linux et sous Windows.
- Le client graphique phpPgAdmin est un client léger (qui tourne dans un navigateur Web donc).

Déclarer une connexion dans pgAdminIII

1. Sélectionner `Fichier > Ajouter un serveur`
2. Saisissez les informations de connexion à un serveur PostgreSQL existant :
 - Hôte : `tuxa.sme.utc`
 - Port : `5432` (port standard de PostgreSQL)
 - Base : `dbnf17p...`
 - Nom : `nf17p...`
 - Mot de passe : ...

The image shows the 'Propriétés' (Properties) dialog box in pgAdmin III for configuring a new server connection. The fields are filled with the following values:

- Nom: nf17p119
- Hôte: host.utc
- Port TCP: 5432
- SSL: (empty dropdown)
- Base maintenance: dbnf17p119
- Nom utilisateur: nf17p119
- Mot de passe: (masked with 10 dots)
- Enregistrer le mot de passe:
- Restaurer l'env. ? :
- Restriction de la BD: (empty text area)
- Service: (empty text field)
- Se connecter:
- Couleur: (empty text field with a color picker button)

Buttons at the bottom: Aide, Valider, Annuler.

Image 2 Fenêtre de connexion pgAdmin III

Ouvrir un terminal SQL dans pgAdminIII

1. Sélectionner sa base de données dans la liste Bases de données
2. Sélectionner Outils > Éditeur de requêtes (ou CTRL+E)

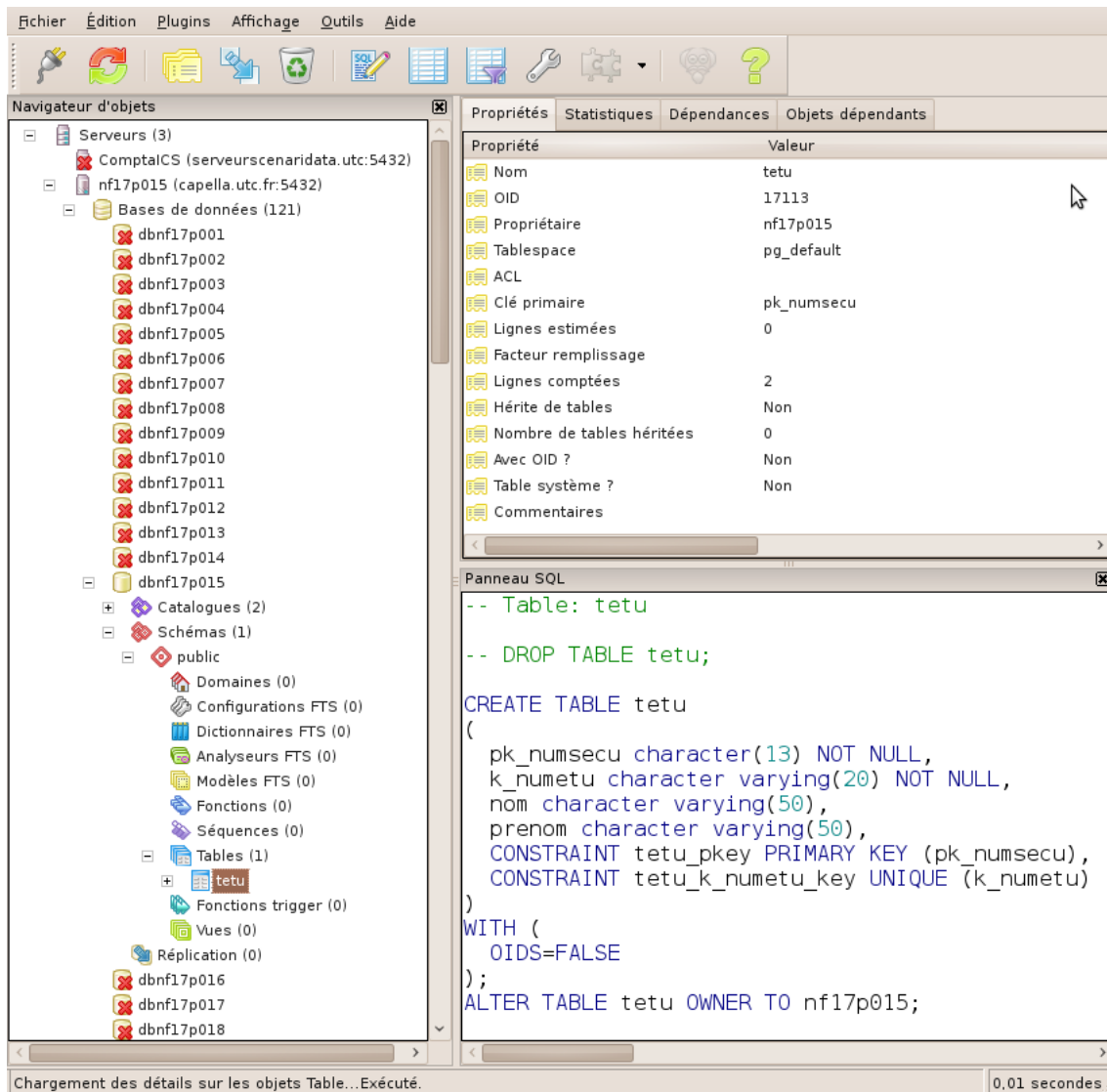


Image 3 pgAdminIII

Complément : phpPgAdmin

<http://phpPgAdmin.sourceforge.net>¹⁴

g) Compléments

Complément : Héritage (clause INHERITS)

<http://www.postgresql.org/docs/current/static/sql-createtable.html>¹⁵

Complément : PL/pgSQL

<http://www.postgresql.org/docs/current/static/plpgsql.html>¹⁶

14 - <http://phpPgAdmin.sourceforge.net>

15 - <http://www.postgresql.org/docs/current/static/sql-createtable.html>

16 - <http://www.postgresql.org/docs/current/static/plpgsql.html>

Complément : Autres langages procéduraux (PL)

- PL/Tcl
- PL/Perl
- PL/Python
- ...

<http://www.postgresql.org/docs/current/static/xplang.html>¹⁷

<http://www.postgresql.org/docs/current/static/server-programming.html>¹⁸

Complément : Triggers

<http://www.postgresql.org/docs/current/static/sql-createtrigger.html>¹⁹

(<http://www.postgresql.org/docs/current/static/triggers.html>²⁰)

B. Exercices

1. Découverte de la ligne de commande sous Linux

Cet exercice consiste à expérimenter quelques commandes de base sous Linux.

1. Connectez vous à une machine Linux ;
2. Ouvrez un terminal.

a) Exercice : man man

Exécutez la commande `man man`.

Exercice

Comment s'appelle la section qui résume la syntaxe d'une commande dans le manuel.

Exercice

Quelle touche doit-on utiliser pour convoquer l'aide de man ?

Exercice

Quelle touche doit-on utiliser pour sortir de man ?

17 - <http://www.postgresql.org/docs/current/static/xplang.html>

18 - <http://www.postgresql.org/docs/current/static/server-programming.html>

19 - <http://www.postgresql.org/docs/current/static/sql-createtrigger.html>

20 - <http://www.postgresql.org/docs/current/static/triggers.html>

b) Exercice

Trouvez ce que signifient dans le manuel :

- les termes en gras
- les termes soulignés ou italiques
- les termes entre crochets [...]
- la barre |
- les trois petits points ...

1 - italique ou souligné

2 - |

3 - []

4 - gras

5 - ...

à taper
exactement
comme indiqué

à remplacer par
l'argument
approprié

arguments
facultatifs

options qui ne
peuvent pas être
utilisées
simultanément

argument ou
expression qui
peut être répété

c) Exercice : ls

Exercice

Quelle commande devez-vous exécuter pour visualiser le manuel de la commande `ls` ?

Exercice

En lisant le manuel, trouvez comment afficher le contenu du répertoire courant en mode listing.

d) Exercice : cd pwd mkdir touch ...

L'exercice suivant consiste à manipuler le système de fichier. On vous demande de saisir la commande nécessaire pour répondre à chaque question posée.

Exercice

Déplacez-vous dans le répertoire `tmp` situé à la racine du système de fichier.

Exercice

Vérifiez que vous êtes bien au bon endroit.

Exercice

Créez un répertoire hello.

Exercice

Déplacez-vous dans ce répertoire.

Exercice

Créez un fichier world dans ce répertoire.

Exercice

Afficher la liste des fichiers de ce répertoire en mode listing.

Exercice

Éditer le fichier world avec l'éditeur nano, et saisissez le texte de votre choix.

Exercice

Après être sorti de l'éditeur, affichez le contenu de votre fichier (avec cat, more ou less)

Exercice

Rechercher dans votre répertoire courant tous les fichiers dont le nom contient orl.

Exercice

Supprimer le fichier *world*.

Exercice

Retournez dans votre dossier personnel initial.

Exercice

Supprimer le dossier *hello* que vous avez créé dans */tmp*.

2. Découverte d'un SGBDR avec PostgreSQL

a) Configuration technique pour réaliser les exercices

Fondamental : Serveur Linux + Client Linux

La meilleure configuration pour réaliser ces exercices est de disposer :

1. d'un serveur Linux hébergeant un serveur PostgreSQL ;
2. d'une base de données déjà créée (cela peut être la base par défaut *postgres*) ;
3. d'un utilisateur ayant les pleins droits sur cette base de données (cela peut être l'utilisateur par défaut *postgres*) ;
4. d'un client *psql* sous Linux connecté à la base de données.

Rappel

Présentation

Le client textuel "*psql*"

Installation (du serveur PostgreSQL et du client *psql*)

Serveur Linux + Client Windows

Si un serveur est disponible sous Linux mais que le client est sous Windows, il y a deux solutions :

- **Se connecter en SSH au serveur Linux avec Putty^{Putty} ↕ (on est ramené au cas précédent).**
- Installer le client *psql* sous Windows, mais cette solution est déconseillée :
 - le terminal est bien moins confortable que sous Linux,
 - une partie des questions sont relatives aux systèmes Linux et ne pourra être réalisée.

Complément : Serveur Windows + Client Windows

Cette configuration permettra de faire une partie des exercices, avec des adaptations.

b) Connexion à une base de données PostgreSQL

Cet exercice commence alors que vous êtes connecté à une base Oracle avec le client *psql*.

```
1 psql (9.x.x)
2 Saisissez « help » pour l'aide.
3
4 mydb=>
```

Demander de l'aide

Demander de l'aide en testant :

- help
- \?
- \h
- \h CREATE TABLE
- \h SELECT

c) Créer une base de données avec PostgreSQL

Créer une table

Exécuter les instructions suivantes.

```
1 CREATE TABLE etu (
2   pknumsecu CHAR(13) PRIMARY KEY,
3   knumetu VARCHAR(20) UNIQUE NOT NULL,
4   nom VARCHAR(50),
5   prenom VARCHAR(50));
```

```
1 INSERT INTO etu (pknumsecu, knumetu, nom, prenom)
2 VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3 INSERT INTO etu (pknumsecu, knumetu, nom, prenom)
4 VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');
```

```
1 CREATE TABLE uv (
2   pkcode CHAR(4) NOT NULL,
3   fketu CHAR(13) NOT NULL,
4   PRIMARY KEY (pkcode, fketu),
5   FOREIGN KEY (fketu) REFERENCES etu(pknumsecu));
```

```
1 INSERT INTO uv (pkcode, fketu)
2 VALUES ('NF17', '1800675001066');
3 INSERT INTO uv (pkcode, fketu)
4 VALUES ('NF26', '1800675001066');
5 INSERT INTO uv (pkcode, fketu)
6 VALUES ('NF29', '1800675001066');
```

Question 1

Utiliser le catalogue pour vérifier la création de la table.

Question 2

Utiliser deux instructions SELECT pour vérifier le contenu de la table.

d) Import de données depuis un fichier CSV

Nous allons à présent réinitialiser la base avec des données contenues dans un fichier.

Question 1

Exécuter les instructions nécessaires afin de **supprimer** les données existantes dans les tables (instruction DELETE du SQL LMD). Vérifier que les tables sont vides.

Question 2

Créer les deux fichiers de données suivants, respectivement *etus.csv* et *uvs.csv*. Regarder le contenu des fichiers. Pourquoi les appelle-t-on des fichiers CSV ?

```
1 1;A;Dupont;Pierre
2 2;B;Durand;Georges
3 3;C;Duchemin;Paul
4 4;D;Dugenou;Alain
5 5;E;Dupied;Albert
```

```
1 1;NF17
2 1;NF18
3 1;NF19
4 1;NF20
5 1;LA13
6 1;PH01
7 2;NF17
8 2;NF18
9 2;NF19
10 2;TN01
11 2;LA14
12 2;PH01
13 3;NF17
14 3;NF18
15 3;NF19
16 3;NF21
17 3;LA14
18 3;PH01
19 4;NF17
20 4;NF20
21 4;NF21
22 4;GE10
23 4;LA14
24 4;PH01
25 5;NF17
26 5;NF18
27 5;NF20
28 5;GE10
29 5;PH01
30 5;PH02
```

Indice :

Fichier CSV

Question 3

Insérer les données en complétant les instructions suivantes.

```
1 \copy ... (...) FROM '...' WITH CSV DELIMITER '...'
2 \copy ... (...) FROM '...' WITH CSV DELIMITER '...'
```

Indice :

Importer un fichier CSV

Question 4

Écrivez les requêtes permettant d'obtenir le nombre d'UV suivies par un étudiant et le nombre d'étudiants inscrits par UV.

e) Manipulation de schémas

Question 1

Visualiser la liste des schémas existants dans votre base de données.

Indice :

Notion de schéma

Question 2

Créer un second schéma *loisir*.

Question 3

Créer une table *sport* dans le schéma *loisir* ; l'objectif est d'indiquer pour chaque étudiant, la liste des sports qu'il pratique, par exemple : Tennis, Karaté, Aviron...

Vérifier votre création dans le dictionnaire.

f) Exécution de fichiers SQL et de scripts Linux

Reporter dans un fichier SQL l'ensemble des instruction permettant de supprimer, de créer et d'instancier les tables de la base de données.

Question 1

Exécuter ce fichier depuis *psql*.

Indice :

Exécuter des instructions SQL depuis un fichier

Question 2

Exécuter ce fichier depuis un script Linux.

Indice :

PostgreSQL sous Linux

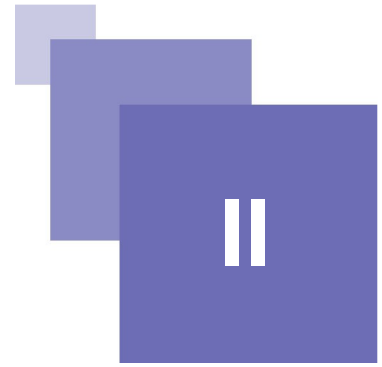
g) Test de pgAdminIII

Connexion depuis un PC avec pgAdmin III

Installer si nécessaire, puis lancer et tester le programme `pgAdminIII`.

Exécuter une ou deux requêtes permettant de se familiariser avec son fonctionnement.

Application de bases de données, principes et exemples avec LAPP



A. Cours

1. Applications et bases de données

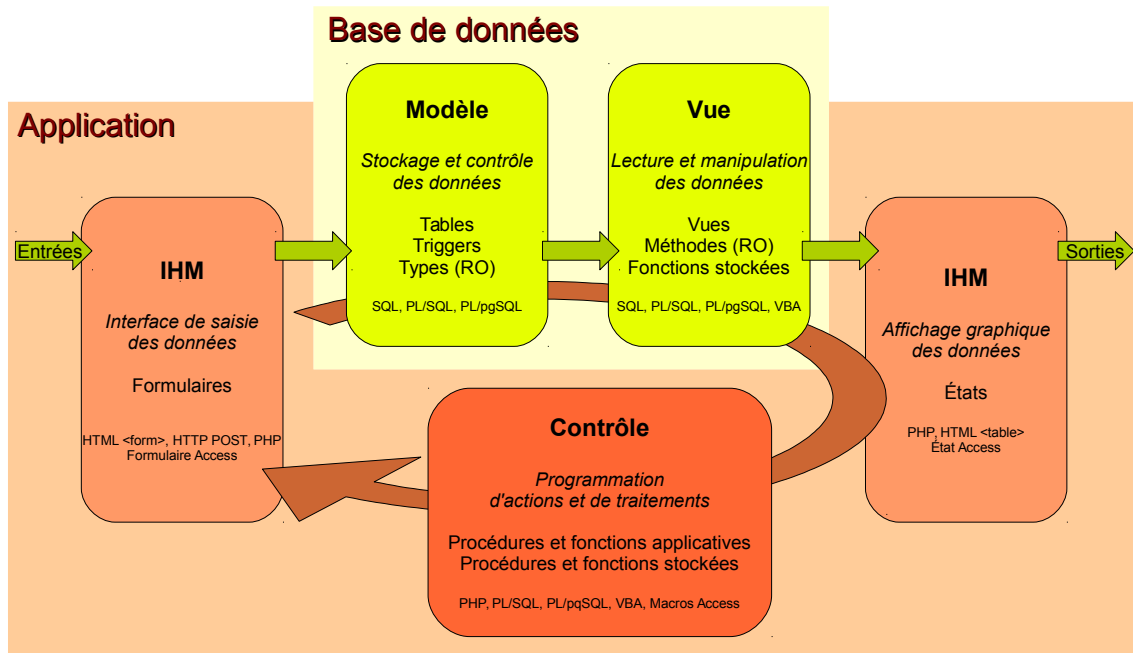
a) Architecture générale d'une application de base de données

Définition

Une application de base de données comporte :

1. Une base de données : Elle pose le modèle de données, stocke les données, définit des vues sur les données (préparation des modalités de lecture)
2. Une application : Elle définit les interfaces homme-machine pour écrire et lire les données et contient le programme informatique de traitement des données avant insertion dans la base ou présentation à l'utilisateur.

Fondamental



Graphique 2 Architecture générale d'une application de base de données

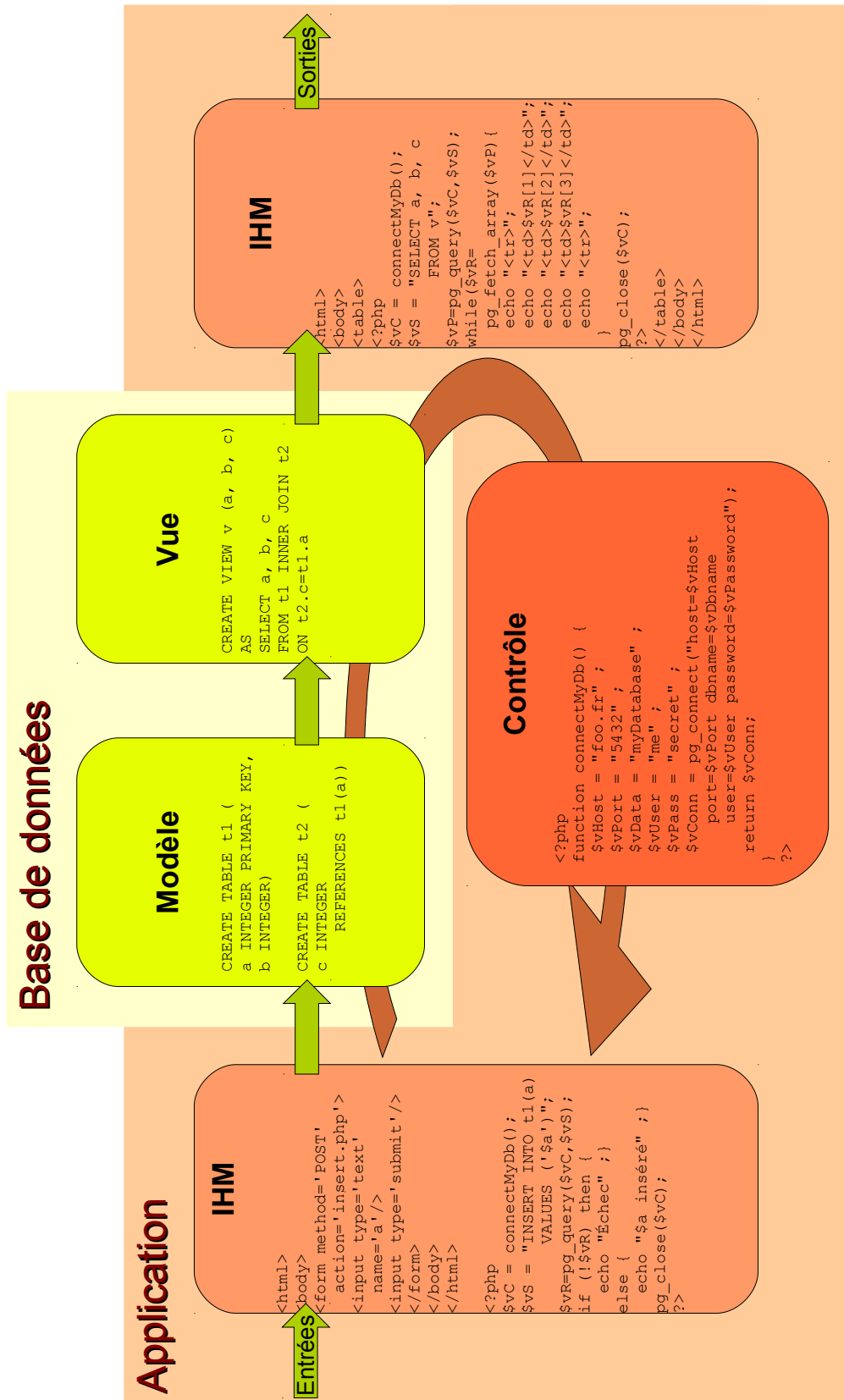
Méthode : Procédure générale de développement

- Développement de la BD★ : Conception en UML★ ou EA★, puis traduction en R★ ou RO★, puis implémentation SQL★
 - Déclarer les **types** (en RO)
 - Ajouter les **vues** utiles, pour simplifier l'accès aux données en lecture
 - Ajouter des **triggers**, pour implémenter les contraintes complexes non exprimables en SQL (lorsque le SGBD le permet, comme Oracle ou PostgreSQL)
 - Implémenter les **méthodes** d'accès aux données (en RO), ou à défaut les **fonctions stockées** (en R) permettant de renvoyer les valeurs calculées prévues par le modèle conceptuel
- Développement de l'application : traitements, formulaires et états
 - Les traitements proches des données sont réalisés dans le langage associé à la BD s'il existe (PL/SQL, PL/pgSQL, VBA)
 - Les traitements proches de l'application sont réalisés dans le langage applicatif choisi (PHP, Java, VBA & Macros sous Access)

Exemple : Exemple de technologies

- BD :
 - SQL pour les tables, les types et les vues
 - PL/SQL (Oracle), PL/pgSQL (PostgreSQL) pour les triggers
 - PL/SQL, PL/pgSQL, VBA (Access) pour les fonctions stockées
- Application
 - HTML avec la balise <form> et le protocole HTTP/POST (Web), formulaires Access
 - PHP (echo) et présentation HTML (<table> par exemple), état Access
 - PL/SQL, PL/pgSQL, VBA (Access) pour les procédures et fonctions stockées (traitements proches de la base de données)
 - PHP (Web), Macros & VBA (Access)

Exemple : Exemple PHP/PosgreSQL



Graphique 3 Exemple d'application de base de données

b) Méthode générale d'accès à une BD en écriture par un langage de programmation

Méthode

1. Connexion à la base de données et récupération d'un identifiant de connexion
2. Écriture de la requête d'insertion ou de mise à jour de données
3. Exécution de la requête sur la connexion ouverte et récupération d'un résultat d'exécution (ici TRUE ou FALSE selon que la requête s'est exécuté ou non avec succès)
4. Test du résultat et dialogue avec l'utilisateur ou remontée d'erreur en cas de résultat FALSE
5. Clôture de la connexion

Méthode : Pseudo-code

```

1 // Connexion à la base de données
2 $vHost = "foo.fr"
3 $vPort = "6666"
4 $vData = "myDatabase"
5 $vUser = "me"
6 $vPass = "secret"
7 $vConn = CONNECT ($vHost, $vPort, $vDb, $vUser, $vPass)

```

```

1 // Écriture de la requête
2 $vSql = "insert into t (a) values (1) ;"

```

```

1 // Exécution de la requête
2 $vResult = QUERY($vConn, $vSql)

```

```

1 // Test du résultat
2 IF (NOT $vResult) THEN MESSAGE("Échec de l'exécution")

```

```

1 // Clôture de la connexion
2 CLOSE ($vConn)

```

Remarque

La connexion est bien entendu inutile dans le cas de procédure stockées, qui se trouvent par définition déjà associées à une BD en particulier.

Exemple : Fonction PHP

```

1 <?php
2 function fInsert ($pValue) {
3 // Connexion à la base de données
4     $vHost = "foo.fr" ;
5     $vPort = "5432" ;
6     $vData = "myDatabase" ;
7     $vUser = "me" ;
8     $vPass = "secret" ;
9     $vConn = new PDO('pgsql:host=$vHost;port=$vPort;dbname=$vData', '$vUser',
    '$vPass');
10 // Écriture, exécution et test de la requête
11     $vSql = "INSERT INTO t (a) VALUES ($pValue)" ;
12     $vResult=$vConn->query($vSql);
13     if (! $vResult) then {
14         echo "Échec de l'insertion" ;
15         return 0 ;
16     }
17     else {
18         return 1 ;
19     }

```

```

20 // Clôture de la connexion
21 $vConn = null ;
22 }
23 ?>

```

Exemple : Procédure VBA

```

1 Sub fInsert(pValue As String)
2     vSql = "INSERT INTO t (a) VALUES ('" & pValue & "'"
3     CurrentDb.CreateQueryDef("", vSql).Execute
4 End Sub
5

```

Exemple : Programme Java

```

1 class fInsert {
2     public static void main(String[] args) {
3         try {
4             // Connexion
5             DriverManager.registerDriver (new OracleDriver());
6             Connection vCon =
7             DriverManager.getConnection("jdbc:oracle:thin:nf17/nf17@localhost:1521:test");
8             // Exécution de la requête
9             Statement vSt = vCon.createStatement();
10            vSt.execute("INSERT INTO t (a) VALUES ('" + args[0] + "'");
11        }
12        catch (Exception e) {
13            e.printStackTrace();
14        }
15    }

```

c) Méthode générale d'accès à une BD en lecture par un langage de programmation

Définition : Pointeur sur un résultat de requête

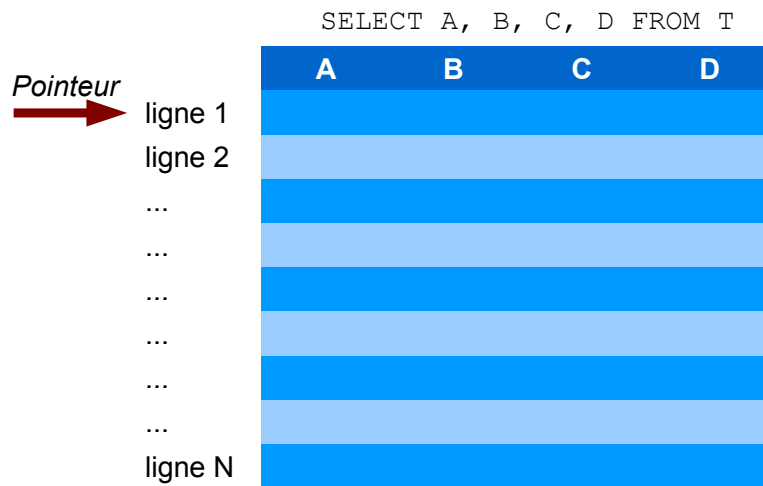
Lorsqu'un programme exécute une requête dans une BD, il récupère un pointeur sur un résultat de requête permettant de parcourir le tableau résultant de l'exécution de la requête ligne par ligne.

En effet :

- Une requête SQL retourne toujours un tableau (même si parfois il n'a qu'une ligne et une colonne), c'est le fondement du modèle relationnel
- En général il n'est pas souhaitable que ce tableau soit transmis directement sous la forme d'une structure en mémoire (*array*), car le résultat d'une requête peut être aussi volumineux que la BD complète stockée en mémoire secondaire
- La solution générale est donc que la BD fournisse le résultat ligne par ligne

Le pointeur permet de :

- Retourner la ligne pointée sous la forme d'un vecteur (tableau à une seule ligne)
- Passer à la ligne suivante
- Savoir si l'on a atteint la fin du tableau (ligne N)



Graphique 4 Pointeur sur un résultat de requête

Méthode

1. Connexion à la base de données et récupération d'un identifiant de connexion
2. Écriture de la requête de sélection
3. Exécution de la requête sur la connexion ouverte et récupération d'un pointeur sur un résultat de requête
4. Parcours du pointeur dans une boucle permettant rapporter (*fetch*) et traiter (afficher par exemple) le tableau ligne par ligne
5. Clôture de la connexion

Méthode : Pseudo-code

```

1 // Connexion à la base de données
2 $vHost = "foo.fr"
3 $vPort = "6666"
4 $vData = "myDatabase"
5 $vUser = "me"
6 $vPass = "secret"
7 $vConn = CONNECT ($vHost, $vPort, $vDb, $vUser, $vPass)

```

```

1 // Écriture de la requête
2 $vSql = "select a, b from t;"

```

```

1 // Exécution de la requête
2 $vPointeur = QUERY ($vConn, $vSql)

```

```

1 // Traitement du résultat
2 IF (NOT $vPointeur) THEN MESSAGE("Échec de l'exécution")
3 ELSE
4     WHILE ($vPointeur)
5         FETCH $vPointeur IN $vResult[]
6         PRINT $vResult[1]
7         PRINT $vResult[2]
8     NEXT $vPointeur
9     END WHILE
10 END IF

```

```

1 // Clôture de la connexion
2 CLOSE ($vConn)

```

Exemple : Fonction PHP

```

1 <?php
2 function fSelect () {
3 // Connexion à la base de données
4     $vHost = "foo.fr" ;
5     $vPort = "5432" ;
6     $vData = "myDatabase" ;
7     $vUser = "me" ;
8     $vPass = "secret" ;
9     $vConn = new PDO('pgsql:host=$vHost;port=$vPort;dbname=$vData', '$vUser',
    '$vPass');
10 // Écriture, préparation et exécution de la requête
11     $vSql = "SELECT a, b FROM t;"
12     $vResultSet = $vConn->prepare($vSql);
13     $vResultSet->execute();
14 // Traitement du résultat
15     while ($vRow = $vResultSet->fetch(PDO::FETCH_ASSOC)) {
16         echo $row['a'];
17         echo " ";
18         echo $row['b'];
19     }
20 // Clôture de la connexion
21     pg_close($vConn)
22 }
23 ?>

```

Exemple : Procédure VBA

```

1 Sub fSelect()
2     vSql = "select a, b from t"
3     Set vRs = CurrentDb.CreateQueryDef("", vSql).OpenRecordset
4     Do While Not vRs.EOF
5         Debug.Print vRs.Fields(0)
6         Debug.Print vRs.Fields(1)
7         vRs.MoveNext
8     Loop
9 End Sub

```

Exemple : Programme Java

```

1 class fInsert {
2     public static void main(String[] args) {
3         try {
4             // Connexion
5             DriverManager.registerDriver (new OracleDriver());
6             Connection vCon =
7             DriverManager.getConnection("jdbc:oracle:thin:nf17/nf17@localhost:1521:test");
8             // Exécution de la requête
9             Statement vSt = vCon.createStatement();
10            ResultSet vRs = vSt.executeQuery("SELECT a, b FROM t");
11            // Affichage du résultat
12            while(vRs.next()){
13                String vA = vRs.getString(1);
14                String vB = vRs.getString(2);
15                System.out.println(vA + " - " + vB));
16            }
17        } catch (Exception e) {
18            e.printStackTrace();
19        }
20    }
21 }

```

2. Architecture Web

Objectifs

Comprendre les principes des architectures d'application de bases de données (en particulier 3-tier et web)

Cette section a été réalisée à partir de contenus de www.commentcamarche.net²¹, © 2003 Jean-François Pillou (document soumis à la licence GNU FDL).

a) Notions d'architecture client-serveur

Présentation de l'architecture d'un système client/serveur

De nombreuses applications fonctionnent selon un environnement clients/serveur, cela signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en terme de capacités d'entrée-sortie, qui leur fournit des services. Ces services sont des programmes fournissant des données telles que l'heure, des fichiers, une connexion...

Les services sont exploités par des programmes, appelés programmes clients, s'exécutant sur les machines clientes. On parle ainsi de client FTP, client de messagerie...

Dans un environnement purement client/serveur, les ordinateurs du réseau (les clients) ne peuvent voir que le serveur, c'est un des principaux atouts de ce modèle.

Avantages de l'architecture client/serveur

Le modèle client/serveur est particulièrement recommandé pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont :

- **des ressources centralisées**
étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction
- **une meilleure sécurité**
car le nombre de points d'entrée permettant l'accès aux données est moins important
- **une administration au niveau serveur**
les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés
- **un réseau évolutif**
grâce à cette architecture on peut supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modifications majeures

Inconvénients du modèle client/serveur

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles :

- **un coût élevé**
dû à la technicité du serveur
- **un maillon faible**
le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui! Heureusement, le serveur a une grande tolérance aux pannes (notamment grâce au système RAID)

21 - www.commentcamarche.net

Fonctionnement d'un système client/serveur

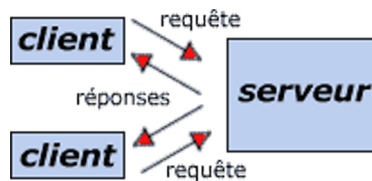


Image 4 Schéma de fonctionnement d'un système client/serveur (commentcamarche.net - © 2003 Pillou - GNU FDL)

Un système client/serveur fonctionne selon le schéma suivant:

- Le client émet une requête vers le serveur grâce à son **adresse** et à son **port**, qui désigne un service particulier du serveur
- Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine client (et de son port)

b) Notions d'architecture 3-tier

Présentation de l'architecture à deux niveaux

L'architecture à deux niveaux (aussi appelée architecture 2-tier, *tier* signifiant étage en anglais) caractérise les systèmes clients/serveurs dans lesquels le client demande une ressource et le serveur la lui fournit directement. Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir le service.

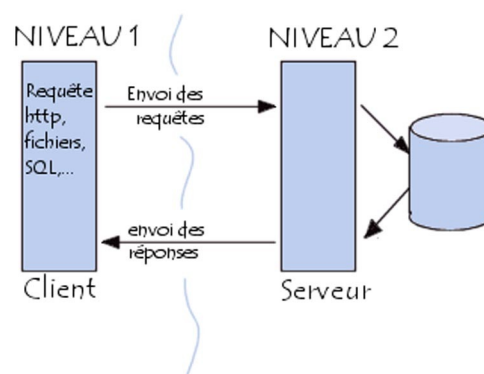


Image 5 Architecture 2-tier (commentcamarche.net - © 2003 Pillou - GNU FDL)

Présentation de l'architecture à trois niveaux

Dans l'architecture à 3 niveaux (appelée architecture 3-tier), il existe un niveau intermédiaire, c'est-à-dire que l'on a généralement une architecture partagée entre:

1. **Le client**
le demandeur de ressources
2. **Le serveur d'application**
(appelé aussi *middleware*) le serveur chargé de fournir la ressource mais faisant appel à un autre serveur
3. **Le serveur secondaire**
(généralement un serveur de base de données), fournissant un service au premier serveur

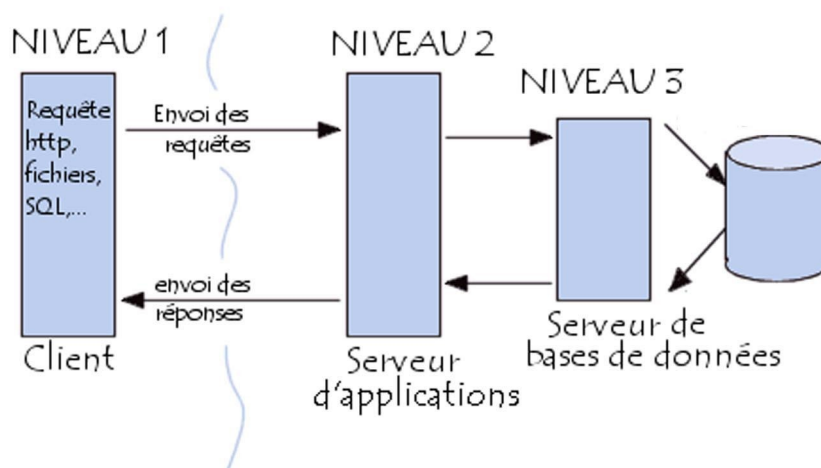


Image 6 Architecture 3-tier (commentcamarche.net - © 2003 Pillou - GNU FDL)

Remarque

Étant donné l'emploi massif du terme d'architecture à 3 niveaux, celui-ci peut parfois désigner aussi les architectures suivantes :

- Partage d'application entre client, serveur intermédiaire, et serveur d'entreprise
- Partage d'application entre client, base de données intermédiaire, et base de données d'entreprise

Comparaison des deux types d'architecture

L'architecture à deux niveaux est donc une architecture client/serveur dans laquelle le serveur est polyvalent, c'est-à-dire qu'il est capable de fournir directement l'ensemble des ressources demandées par le client.

Dans l'architecture à trois niveaux par contre, les applications au niveau serveur sont délocalisées, c'est-à-dire que chaque serveur est spécialisé dans une tâche (serveur web et serveur de base de données par exemple). Ainsi, l'architecture à trois niveaux permet :

- une plus grande flexibilité/souplesse
- une plus grande sécurité (la sécurité peut être définie pour chaque service)
- de meilleures performances (les tâches sont partagées)

Complément : L'architecture multi-niveaux

Dans l'architecture à 3 niveaux, chaque serveur (niveaux 1 et 2) effectue une tâche (un service) spécialisée. Ainsi, un serveur peut utiliser les services d'un ou plusieurs autres serveurs afin de fournir son propre service. Par conséquent, l'architecture à trois niveaux est potentiellement une architecture à N niveaux.

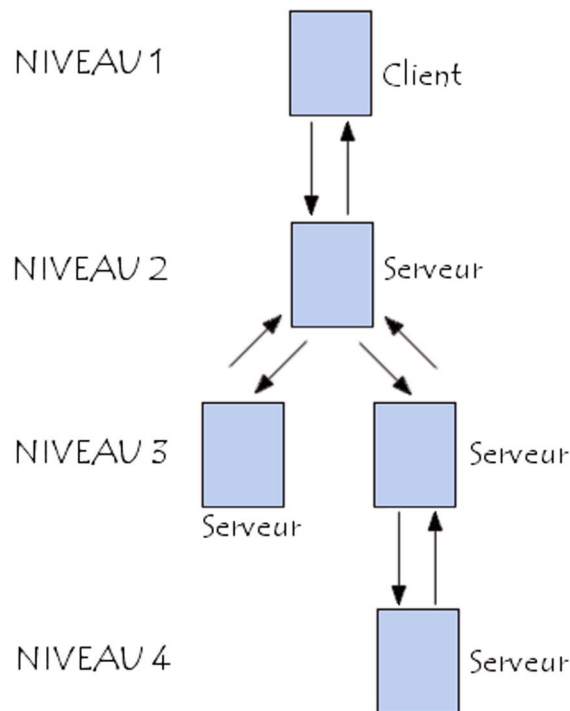


Image 7 Architecture N-tier (commentcamarche.net - © 2003 Pillou - GNU FDL)

c) Notions de serveur et de client web

Fondamental

Un serveur web sert à rendre accessibles des pages web sur internet via le protocole HTTP.

Définition : Serveur web

Un serveur web est un logiciel capable de répondre à des requêtes HTTP, c'est à dire de renvoyer des données (par exemple une page HTML), en réponse à des demandes écrites en HTTP (par exemple une requête GET).

Synonyme : serveur HTTP

Définition : Client web

Un client web est un logiciel capable d'envoyer des requêtes HTTP à un serveur web et d'afficher les résultats. Les navigateurs web sont les clients web les plus répandus.

Remarque

Un serveur web répond par défaut sur le port 80.

Exemple

- Apache : logiciel libre fondation Apache, la moitié des sites web au monde
- Nginx : logiciel libre BSD, en croissance
- IIS : logiciel propriétaire Microsoft

d) Notion d'architecture web

Fait à partir de www.commentcamarche.net²². Copyright 2003 Jean-François Pillou. Document soumis à la licence GNU FDL.

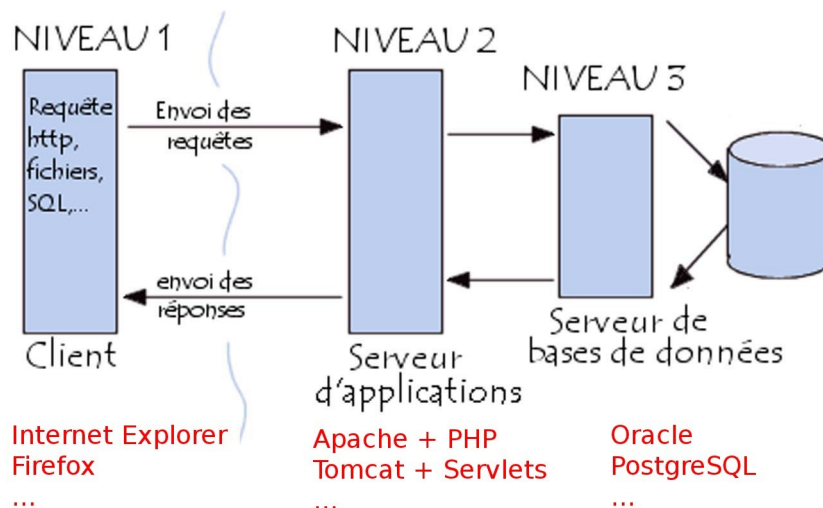


Image 8 Exemples d'architecture Web (commentcamarche.net - © 2003 Pillou - GNU FDL)

e) Architecture LAPP

Définition : Définition

On appelle une architecture LAPP une architecture qui s'appuie sur :

- **L**inux pour le système d'exploitation
- **A**pache pour le serveur Web
- **P**ostgreSQL pour la base de données
- **P**HP pour le langage applicatif

Complément : LAMP, WAMP, WAPP

- LAMP : Linux, Apache, MySQL, PHP
- WAMP : Windows, Apache, MySQL, PHP
- ...

B. Exercice

1. Tester un environnement LAPP sur son ordinateur personnel

Pré-requis

La réalisation de cet exercice nécessite un ordinateur sous Linux avec un accès *root* et un serveur PostgreSQL installé. Les commandes sont spécifiées pour une distribution Ubuntu.

Rappels Linux

- *Utiliser Linux*
- *Commandes de bases sous Linux : cd, ls, mkdir, rm, find, cat, nano...*

Rappels Postgres

- *Présentation de PostgreSQL*
- *Installation de PostgreSQL*
- *Le client textuel "psql"*

- *PostgreSQL sous Linux*

Serveur web

L'objectif de cette partie est d'avoir un serveur web capable de distribuer des page HTML. Installez le serveur web Apache.

```
1 sudo apt-get install apache2
```

Tester l'accès au serveur Web en entrant l'adresse web dans un navigateur : *http://localhost*²³

Question 1

L'installation par défaut du serveur web permet de servir les fichiers situés dans le dossier `/var/www/html`.

Déposez un fichier HTML de votre choix dans ce dossier et accédez-y via votre navigateur Web.

Indices :

Un exemple de fichier XHTML

Si votre fichier s'appelle `example.html` et qu'il se trouve à la racine du serveur web `/var/www/html` vous y accédez par l'adresse `http://localhost/example.html`²⁴.

PHP

L'objectif de cette partie est d'avoir un interpréteur PHP en mode développement, c'est à dire qui affiche les erreurs quand il y en a.

Installez l'interpréteur PHP pour Apache.

```
1 sudo apt-get install php
```

Question 2

Créez le fichier `example.php` ci-après et déposez-le sur votre serveur web. Faites un test d'accès.

```
1 <?php
2 echo "Hello world";
3 ?>
```

Question 3

Créez et testez le fichier `test.php` ci-après. Ce fichier permet de visualiser la configuration PHP du serveur.

```
1 <?php
2 phpinfo();
3 ?>
```

Question 4

Par défaut l'installation d'un interpréteur PHP est en mode *production* et il n'affiche pas les erreurs. Il faut donc changer la configuration pour passer en mode *développement*.

1. À l'aide du fichier `test.php` précédent trouvez l'endroit où est stocké sur votre ordinateur le fichier de configuration `php.ini`.
Par exemple : `/etc/php/7.0/apache2/php.ini`
2. Éditez ce fichier en tant que *root*: `sudo nano php.ini`
3. Remplacez le paramètre `display_errors = Off` par `display_errors = On`
4. Relancez le serveur web : `sudo service apache2 reload`

23 - `http://localhost`

24 - `http://localhost/example.html`

Question 5

Il est également nécessaire d'installer un complément à PHP pour qu'il se connecte à PostgreSQL.

```
1 sudo apt-get install php-pgsql
2 sudo service apache2 reload
```

PostgreSQL

L'objectif de cette partie est d'avoir une base PostgreSQL prête à être utilisée par un programme PHP.

Question 6

Connectez vous en tant qu'utilisateur *postgres* (su *postgres*).

Créez un utilisateur *test* et une base *test*. Connectez-vous à la base.

```
1 psql -c "CREATE USER test WITH ENCRYPTED PASSWORD 'test'";
2 psql -c "CREATE DATABASE test WITH OWNER test";
3 psql --host=localhost --dbname=test --username=test
```

Créez une table *medicament* et instanciez-la.

```
1 CREATE TABLE medicament (
2   nom varchar,
3   description varchar,
4   conditionnement integer,
5   PRIMARY KEY (nom)
6 );
7
8 INSERT INTO medicament (nom,description,conditionnement)
9 VALUES ('Chourix','Médicament contre la chute des choux',13);
10
11 INSERT INTO medicament (nom,description,conditionnement)
12 VALUES ('Tropas','Médicament contre les dysfonctionnements intellectuels',42);
```

L'objectif de cette partie est d'exécuter un programme PHP capable de lire le contenu de la table *medicament* de notre base de données *test*.

Question 7

Créez le fichier *select.php* suivant (avec l'éditeur *gedit* par exemple) et déposez-le sur votre serveur web. Accédez-y avec un client web pour exécuter le programme.

```
1 <?php
2
3 /** Connexion **/
4 $connexion = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test',
5   'test');
6
7 /** Préparation et exécution de la requête **/
8 $sql = "SELECT nom FROM medicament;";
9 $resultset = $connexion->prepare($sql);
10 $resultset->execute();
11
12 /** Traitement du résultat **/
13 while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
14   echo $row['nom'];
15   echo " ";
16 }
17
18 /** Déconnexion **/
19 $connexion=null;
20 ?>
```

L'objectif de cette partie est d'exécuter un programme PHP capable de modifier le contenu de la table `medicament` de notre base de données `test`.

Question 8

Créez le fichier `insert.php` et testez-le (exécutez-le au moins deux fois).

```

1  <?php
2
3  /** Connexion **/
4  $connexion = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test',
   'test');
5
6  /** Préparation et exécution de la requête **/
7  $sql = "INSERT INTO medicament (nom) VALUES ('Nouveau')";
8  $result = $connexion->prepare($sql);
9  $result->execute();
10
11 /** Traitement du résultat **/
12 if ($result) {
13     echo "'Nouveau' inséré";
14 }
15 else {
16     echo "Erreur lors de l'insertion";
17 }
18
19 /** Déconnexion **/
20 $connexion=null;
21
22 ?>
```

Question 9

Vérifiez que l'insertion a fonctionné en appelant `select.php`.

C. Devoir

1. Déployez un site web sur Internet avec Filezilla

Préambule

Ce devoir propose d'utiliser un site web gratuit chez *000webhost.com*, n'importe quel autre hébergeur peut être librement utilisé pour l'exercice.

Remarque : ce cours ne traite pas de sécurité informatique, mais notez que si, comme dans cet exercice, vous utilisez le protocole FTP pour transmettre des informations, tout passe en clair sur le réseau, y compris le mots de passe d'accès. Il est donc impératif de choisir un mot de passe unique pour ce compte.

Se connecter à un serveur Web

1. Créez un compte chez un hébergeur de site gratuit, par exemple : *000webhost.com*²⁵
2. Installez le client FTP Filezilla (<http://filezilla-project.org>²⁶) : `sudo apt-get install filezilla`
3. Ouvrez Filezilla et exécutez via le menu : File > Site manager (CTRL+S)
4. Cliquez sur le bouton : New site
5. Nommez votre site.
6. Renseignez les informations de connexion, par exemple :
 - Host : `files.000webhost.com`

25 - <https://000webhost.com>

26 - <http://filezilla-project.org/>

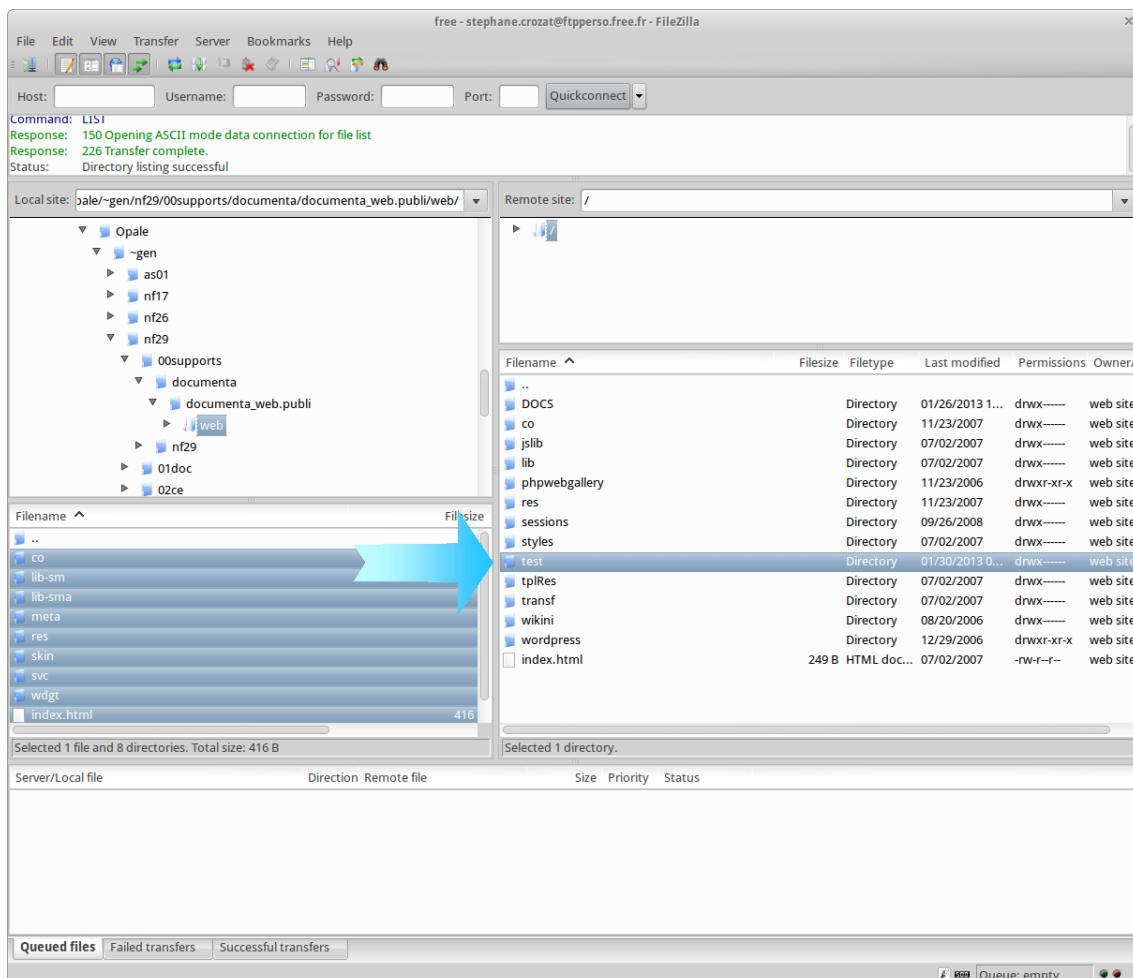
- Port :
 - Protocol : FTP
 - Encryption : Use plain FTP
 - Logon type : Normal
 - User : *votre compte*
 - Password : *votre mot de passe*
 - Account :
7. Cliquez sur : Connect

Publier des fichiers

Filezilla permet de naviguer à gauche dans les fichiers locaux situés sur l'ordinateur (*Local site*) et sur les fichiers distants situés sur le serveur (*Remote site*).

Pour mettre des fichiers sur le serveur Web, il suffit de les glisser-déposer depuis la fenêtre de gauche vers la fenêtre de droite.

1. Créez un dossier `test` dans le dossier `public_html` sur le serveur (clic droit `Create directory`).
2. Créez une ou plusieurs pages web HTML localement sur votre ordinateur.
3. Copiez l'adresse de ces fichiers dans le champ `Local site` de Filezilla pour vous déplacer directement à cette adresse locale.
4. Sélectionnez tous les fichiers et glissez-déposez-les dans votre répertoire distant `test`.
5. Attendez que tous les fichiers soient transférés.
6. Consultez votre site avec un navigateur web à l'adresse de votre site plus `/test`, par exemple : `https://compte.000webhostapp.com/test`



Publication de fichiers sur le Web avec Filezilla

2. Déployez un site web sur un serveur de l'UTC

Les étudiants de l'UTC disposent d'un compte personnel sur le serveur Linux *kappa.utc.fr* de l'UTC.

Utilisez ce compte pour déployer un site de test.

Introduction à HTML et PHP



A. Cours

1. Introduction à HTML et XHTML

Objectifs

Savoir écrire une page simple en HTML
Savoir créer des formulaires en HTML

a) HTML

Définition : HTML

HTML★ est un langage inventé à partir de 1989 pour coder des pages de contenu sur le Web. Il est standardisé par le W3C★.

Définition : Langage à balises

HTML est un langage à balises : il se fonde sur le mélange entre du contenu et des balises permettant de caractériser ce contenu. HTML utilise le formalisme SGML★ pour définir les balises et combinaisons de balises autorisées.

Exemple : Extrait de code HTML

```
1 <p>Ceci est un contenu, caractérisé par des <b>balises</b></p>
```

Les balises `p` et `b` ont une signification dans le langage HTML : Créer un paragraphe et mettre en gras.

Remarque : HTML5

La version courante de HTML et la version 4.01 de 1999. Le HTML5 en cours de spécification est déjà dans un état avancé de spécification et d'implémentation, il peut d'ors et déjà être employé et est prévu pour 2014.

b) XHTML

Définition : XHTML

XHTML est une réécriture du HTML : tandis que HTML est fondé sur SGML, XHTML est fondé sur XML, plus récent et plus rigoureux. XHTML et HTML ne présentent pas de différence fonctionnelle, uniquement des différences syntaxiques.

Exemple : Comparaison XHTML et HTML

```
1 <ul><li>Ceci est un extrait de contenu <i>HTML
```

```
1 <ul><li>Ceci est un extrait de contenu <i>XHTML</i></li></ul>
```

Dans le cas du HTML les balises fermantes sont optionnelles, en XHTML c'est obligatoire. Les deux exemples sont donc équivalents, mais dans l'exemple HTML, il existait en fait plusieurs interprétations possibles, par exemple :

```
<ul><li>Ceci est un extrait de contenu</li></ul><i>XHTML</i>
```

Remarque : XHTML5

La version actuelle de XHTML est la version 1, correspondant à HTML4. XHTML5 est le pendant de HTML5.

Complément

Définition du XML

Historique : de SGML à XML

Discussion : HTML et XML

c) Structure générale XHTML

Syntaxe : Structure générale

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2   <head> ... </head>
3   <body> ... </body>
4 </html>
```

Syntaxe : Entête

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2   <head>
3     <title>...</title>
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5   </head>
6   <body> ... </body>
7 </html>
```

Syntaxe : Corps

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2   <head>
3     <title>...</title>
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5   </head>
6   <body>
7     <h1>...</h1>
8     <h2>...</h2>
9     <p>...</p>
10  </body>
```

```
11 </html>
```

Complément

- Tutoriel XHTML : <http://fr.html.net/tutorials/html/>²⁷
- Brillant07, pp107-108 [Brillant07]

d) Balises de base XHTML

Syntaxe

```

1 <p>Un paragraphe de texte</p>
2 <p>Paragraphe contenant du texte, mot <b>gras</b> ou <i>italique</i>.</p>
3 <p><a href="page02.html">Un lien</a> vers une autre page</p>
4 
5 <h1>Titre de niveau 1</h1>
6 <h2>Titre de niveau 2</h2>
7 <h3>Titre de niveau 3</h3>
8 <table border="1">
9   <tr><th>Titre      colonne 1</th><th>Titre      colonne 2</th><th>...</th></tr>
10  <tr><td>Ligne 1 colonne 1</td><td>Ligne 1 colonne 2</td><td>...</td></tr>
11  <tr><td>Ligne 2 colonne 1</td><td>Ligne 2 colonne 2</td><td>...</td></tr>
12 </table>
13 <ul>
14   <li>Item de liste à puce</li>
15   <li>Item de liste à puce</li>
16 </ul>
17 <ol>
18   <li>Item de liste à ordonnée</li>
19   <li>Item de liste à ordonnée</li>
20 </ol>
```

Complément

Pour une description des balises de base : Brillant07, pp108-112 [Brillant07].

Complément

Introduction à CSS

2. Introduction à PHP

a) Présentation de PHP

PHP est un langage interprété (un langage de script) exécuté du côté serveur (comme les scripts CGI, ASP, ...) et non du côté client (un script écrit en JavaScript ou une applet Java s'exécute au contraire sur l'ordinateur où se trouve le navigateur). La syntaxe du langage provient de celles du langage C, du Perl et de Java.

Ses principaux atouts sont :

- La gratuité et la disponibilité du code source (PHP est distribué sous licence GNU GPL)
- La simplicité d'écriture de scripts
- La possibilité d'inclure le script PHP au sein d'une page HTML (contrairement aux scripts CGI, pour lesquels il faut écrire des lignes de code pour afficher chaque ligne en langage HTML)
- La simplicité d'interfaçage avec des bases de données (de nombreux SGBD sont supportés, le plus utilisé avec ce langage est MySQL).
- L'intégration au sein de nombreux serveurs web (Apache...)

27 - <http://fr.html.net/tutorials/html/>

Exemple : SGBD supportés par PHP

- MySQL
- Oracle
- PostgreSQL
- ...

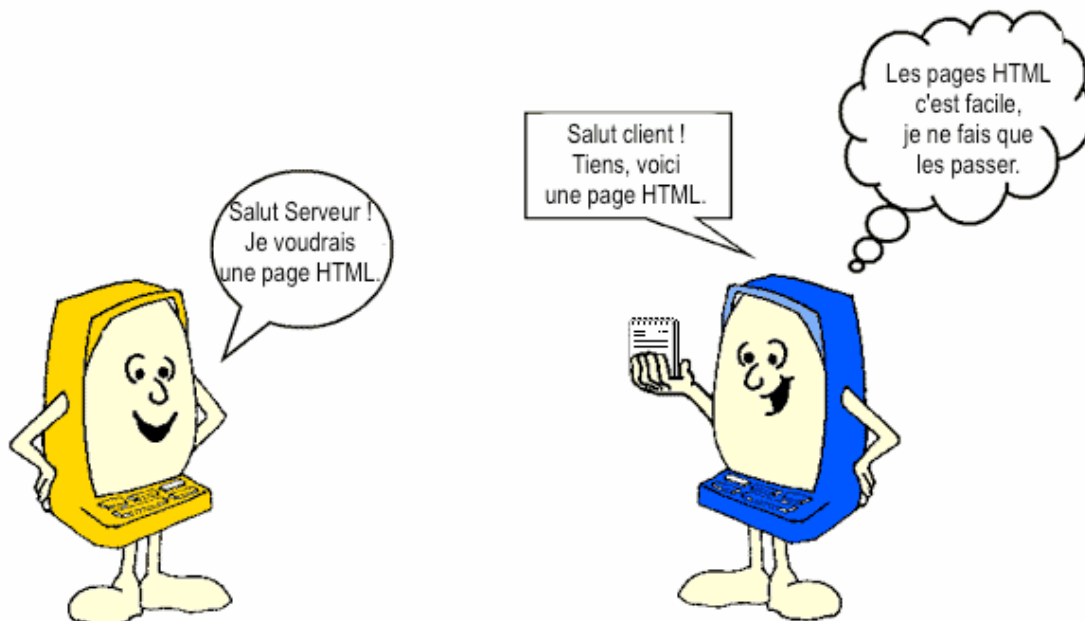
b) Principes de PHP

L'interprétation du code par le serveur

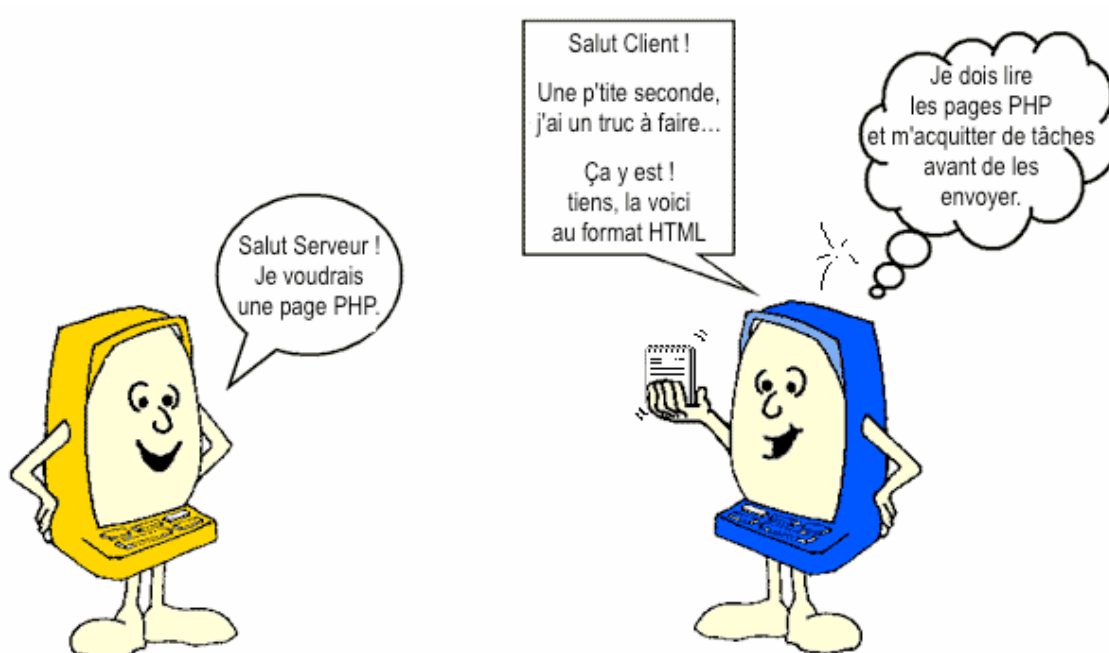
Un script PHP est un simple fichier texte contenant des instructions écrites à l'aide de caractères ASCII 7 bits (des caractères non accentués) incluses dans un code HTML à l'aide de balises spéciales et stocké sur le serveur. Ce fichier doit avoir une extension particulière (qui dépend de la configuration du serveur HTTP, en général ".php") pour pouvoir être interprété par le serveur.

Ainsi, lorsqu'un navigateur (le client) désire accéder à une page dynamique réalisée en php :

1. Le serveur reconnaît qu'il s'agit d'un fichier PHP
2. Il lit le fichier PHP
3. Dès que le serveur rencontre une balise indiquant que les lignes suivantes sont du code PHP, il "passe" en mode PHP, ce qui signifie qu'il ne lit plus les instructions: il les exécute.
4. Lorsque le serveur rencontre une instruction, il la transmet à l'interpréteur
5. L'interpréteur exécute l'instruction puis envoie les sorties éventuelles au serveur
6. A la fin du script, le serveur transmet le résultat au client (le navigateur)



Requête d'une page HTML (<http://fr.html.net/tutorials/php/lesson1.php>)



Requête d'une page PHP (<http://fr.html.net/tutorials/php/lesson1.php>)

Remarque : Code PHP et clients Web

Un script PHP est interprété par le serveur, les utilisateurs ne peuvent donc pas voir le code source.

Le code PHP stocké sur le serveur n'est donc jamais visible directement par le client puisque dès qu'il en demande l'accès, le serveur l'interprète ! De cette façon aucune modification n'est à apporter sur les navigateurs...

Exemple : Hello world

```
1 <?php
2 echo "Hello world";
3 ?>
```

Syntaxe

Pour que le script soit interprété par le serveur deux conditions sont nécessaires :

- Le fichier contenant le code doit avoir l'extension `.php` et non `.html` (selon la configuration du serveur Web)
- Le code PHP contenu dans le code HTML doit être délimité par les balises `<?php` et `?>`

Complément

Tutoriel PHP : <http://fr.html.net/tutorials/php/>²⁸

c) Envoi de texte au navigateur

Syntaxe

```
1 echo Expression;
```

28 - <http://fr.html.net/tutorials/php/>

Remarque : Print

La fonction `print` est iso-fonctionnelle avec `echo` et `printf` plus complexe permet en plus le formatage des données (peu utilisée).

d) Implantation du code PHP au sein du code HTML

Fondamental : L'importance de l'implantation du code PHP au sein du code HTML

Le code PHP peut être implanté au sein du code HTML. Cette caractéristique n'est pas à négliger car le fait d'écrire uniquement du code PHP là où il est nécessaire rend la programmation plus simple (il est plus simple d'écrire du code HTML que des fonctions `echo` ou `print`, dans lesquelles les caractères spéciaux doivent être précédés d'un antislash sous peine de voir des erreurs lors de l'exécution).

L'exemple le plus simple concerne les pages dynamiques dont l'en-tête est toujours le même: dans ce cas, le code PHP peut ne commencer qu'à partir de la balise `<body>`, au moment où la page peut s'afficher différemment selon une variable par exemple.

Mieux, il est possible d'écrire plusieurs portions de script en PHP, séparées par du code HTML statique car les variables/fonctions déclarées dans une portion de script seront accessibles dans les portions de scripts inférieures.

Exemple : Hello world

```

1 <html>
2 <head><title>Exemple</title></head>
3 <body>
4 <?php
5 echo "Hello world";
6 ?>
7 </body>
8 </html>

```

e) Syntaxe PHP

Fondamental : Manuel PHP en ligne

<http://php.net/manual/>²⁹

Exemple

```

1 <?php
2 $i=0 ;
3 while($i<6) {
4     echo $i ;
5     $i=rand(1,6) ;
6 }
7 ?>

```

Attention : Généralités

- Une instruction se termine par un ;
- Les espaces, retours chariot et tabulation ne sont pas pris en compte par l'interpréteur
- Les commentaires sont écrits entre les délimiteurs `/*` et `*/` ou `//` sur une seule ligne.
- Le langage est *case-sensitive* (sauf pour les fonctions).

29 - <http://php.net/manual/>

Complément : IDE

- Eclipse PDT (PHP Development Tools)
<http://www.zend.com/fr/community/pdt>³⁰
- Zend Studio
<http://www.zend.com/fr/products/studio/>³¹

f) Variables en PHP

- Les variables ne sont pas déclarées
- Les variables commencent pas un \$
- Les variables ne sont pas typées

Les variables en langage PHP peuvent être de trois types :

- Scalaires (entiers, chaîne, réels)
- Tableaux (un tableau pouvant être multidimensionnel et stocker des scalaires de types différents)
- Tableaux associatifs (indexés par des chaînes)

Exemple

```
1 $Entier=1;
2 $Reel=1.0;
3 $Chaine="1";
4 $Tableau[0]=1
5 $Tableau[1]="1"
6 $TableauMulti[0][0]="1.0"
7 $TableauAssoc[Age]=18
```

Complément : isset()

```
1 if (isset($var)) {
2     echo $var;
3 }
```

La fonction `isset()` permet de tester qu'une variable existe et est affectée.

Complément

Formulaires HTML et PHP

g) Structures de contrôle en PHP

Syntaxe : Alternative IF

```
1 if (condition réalisée) {
2     liste d'instructions
3 }
4 elseif (autre condition réalisée) {
5     autre série d'instructions
6 }
7 ...
8 else (dernière condition réalisée) {
9     série d'instructions
10 }
```

30 - <http://www.zend.com/fr/community/pdt>

31 - <http://www.zend.com/fr/products/studio/>

Syntaxe : Boucle FOR

```

1 for (compteur; condition; modification du compteur) {
2     liste d'instructions
3 }
```

Syntaxe : Boucle WHILE

```

1 while (condition réalisée) {
2     liste d'instructions
3 }
```

Complément : Autres structures de contrôle

<http://php.net/manual/fr/language.control-structures.php>³²

h) Formulaire HTML et PHP

Rappel

Requête GET ou POST par formulaire HTML (balise <form>)

Traiter les requêtes HTTP avec un serveur PHP

Exemple : Page d'appel

```

1 <html>
2 <body>
3 <form method="get" action="test.php">
4 <input type="text" size="20" name="MaVar"/>
5 <input type="submit"/>
6 </form>
7 </body>
8 </html>
```

Exemple : Page appelée (test.php)

```

1 <?php
2     echo $_GET["MaVar"];
3 ?>
```

Remarque : Code implanté

La page retournée dans l'exemple n'est pas du HTML (mais du simple texte qu'un navigateur peut néanmoins afficher). Pour retourner du HTML, il faut implanter ce même code au sein d'une page HTML.

```

1 <html>
2 <body>
3 <p>
4 <?php
5     echo $_GET["MaVar"];
6 ?>
7 </p>
8 </body>
9 </html>
```

Attention : Cache

Les navigateurs disposent d'un cache, c'est à dire d'une copie locale des fichiers qui leur évite de recharger plusieurs fois un fichier identique. Lorsque l'on développe

32 - <http://php.net/manual/fr/language.control-structures.php>

une application PHP, les fichiers changent fréquemment, il est alors nécessaire de vider le cache pour que le navigateur recharge bien la nouvelle version du code.

Sous Firefox, faire **CTRL+F5**.

B. Exercices

1. Population

[20 min]

Soit la page HTML suivante, visualisée sous le navigateur Web Firefox.



Population par département

Numéro	Nom	Population
01	Ain	529378
02	Aisne	552320
...

- Département le plus peuplé : **Paris** (2147857)
- Département le moins peuplé : **Hautes-Alpes** (126636)

Terminé

Image 9 Page HTML visualisée avec un navigateur Web

Question 1

Écrivez le code HTML source de cette page en utilisant un éditeur de texte, comme `notepad++` sous Windows ou `gedit` sous Linux.

Indice :

Procédez par étape et vérifiez au fur et à mesure le résultat dans un navigateur Web.

Question 2

Déposez le fichier HTML sur un serveur web une fois finalisé et accédez-y avec un navigateur pour vérifier.

Indice :

Mise en ligne d'un fichier HTML sur le serveur `tuxa.sme.utc` de l'UTC

2. Hello PHP !

Question 1

Créer un fichier `test.php` utilisant la fonction `phpinfo` et le tester (la fonction renvoie les informations techniques sur le module PHP installé sur le serveur).

Créer un fichier `hello.php` et le tester en accédant à la page.

```
1
2 <?php
3     echo "Hello world !";
4 ?>
5
```

Question 2

Transformer le fichier PHP pour qu'il renvoie une page HTML.

C. Devoirs

1. Deux fois deux

Question 1

Écrire un programme PHP qui permet d'afficher la table des 2 en HTML.

Indices :

Penser à implanter le code PHP dans le code HTML

On utilisera une boucle for

Question 2

Écrire une page HTML et une page PHP telles que :

1. la page HTML permet de saisir un nombre inférieur à 9 dans un formulaire ;
2. la page PHP affiche la table de ce nombre.

Indice :

Formulaires HTML et PHP

Requêtes HTTP avec Apache sous Linux



IV

A. Cours

1. Utiliser une machine Linux

Cette section a pour objectif de présenter une partie des commandes utiles pour utiliser un ordinateur sous Linux. Elle suppose l'usage du terminal et des commandes de base (cd, ls, cat, nano...) acquise.

On s'appuiera sur une distribution Ubuntu ou Debian.

<https://www.debian.org/doc/manuals/debian-reference/ch01.fr.html>³³

a) Connexion à un serveur Linux avec SSH

Rappel

Le terminal

L'usage de Linux en ligne de commande permet de travailler sur une machine à distance.

On utilise pour cela le protocole SSH. Il permet à un client SSH de se connecter à un serveur SSH.

Méthode : ssh : ce connecter à un ordinateur

`ssh user@server` permet de se connecter depuis n'importe quel terminal à une machine distante qui autorise les connexions SSH et sur laquelle on dispose d'un compte *user*.

Exemple

`ssh nf17@apollon.utc.fr` permet de se connecter à la machine *apollon.utc.fr* en tant qu'utilisateur *nf17*.

Complément : Syntaxe complète

`ssh user@server -p num_port` si la machine n'utilise pas le port standard (port 22).

33 - <https://www.debian.org/doc/manuals/debian-reference/ch01.fr.html>

Complément : hostname

Si l'on veut savoir sur quelle machine est actuellement connecté notre terminal, on utilise la commande `hostname`.

Complément : Putty sous Windows

Putty^{Putty} est un client SSH sous Windows, qui permet donc de se connecter à distance à une machine Linux

Méthode : scp : copie distante de fichier

```
scp /local/path/file user@server:/dist/path/file
scp -r /local/path/dir user@server:/dist/path/dir
scp user@server:/dist/path/file /local/path/file
scp -r user@server:/dist/path/dir /local/path/dir
```

Exemple

`scp monfichier.txt nf17@apollon.utc.fr:~/mondir` permet de copier le fichier `monfichier.txt` vers le répertoire `mondir` situé dans le dossier personnel de `nf17` sur la machine `apollon.utc.fr`.

Complément : sftp : transfert de fichier

SFTP est un protocole de transfert de fichiers alternatif à `scp` qui peut être utilisé avec un logiciel graphique comme *Filezilla*.

b) Affichage des droits sous Linux (ls -al)

Rappel

Commandes de bases sous Linux : `cd`, `ls`, `mkdir`, `rm`, `find`, `cat`, `nano`...

Sous Linux, chaque dossier et chaque fichier :

- appartient à un utilisateur **et** à un groupe ;
- possède des droits.

Ces droits se divisent en trois :

1. Les droits pour l'utilisateur propriétaire
2. Les droits pour les utilisateurs membres du groupe propriétaire
3. Les droits pour tous les autres utilisateurs

Fondamental : ls -al

La commande `ls -al` affiche tous les fichiers et dossiers d'un répertoire, avec leurs propriétaires et leurs droits.

```
1 drwxr-x--- 3 stc friends 4096 mars 31 19:11 .dir
```

- Le premier caractère indique sur l'élément est un dossier (*d*) ou un fichier (*-*).
- Les trois caractères suivants indiquent les droits de l'utilisateur propriétaire (ici *stc* a les droits *rwX* : lire, écrire, parcourir)
- Les trois caractères suivants indiquent les droits des utilisateur membres du groupe propriétaire (ici les membres de *friends* ont les droits *r-X* : lire et parcourir, mais pas écrire)
- Les trois caractères suivants indiquent les droits des autres utilisateurs (ici *---* : aucun droit)
- Le nombre suivant indique le nombre de sous-dossiers dans le dossier listé (ici *.dir* a 3 sous-dossiers, dont *.* et *..* donc un seul vrai sous-dossier)
- La chaîne suivante indique l'utilisateur propriétaire (ici l'utilisateur *stc*)

- **La chaîne suivante indique le groupe propriétaire (ici le groupe *friends*)**
- **Le nombre suivant indique la taille de l'élément en octets (ici 4096 octets)**
- **La chaîne suivante indique la date et l'heure de dernière modification du fichier (ici le 31 mars à 19h11)**
- **La dernière chaîne est le nom du fichier (ici *.dir*)**

Exemple

```
1 -rw----- 1 root  root  2048 mars  31 19:11 .config
2 drwx----- 2 stc   stc   4096 avril  3 20:25 firefox
3 -rw-rw-r-- 1 stc   www  31691 avril  2 10:11 test.jpg
```

- Le fichier *.config* appartient à l'utilisateur *root* et au groupe *root* (il y a souvent un utilisateur et un groupe de même nom) ; l'utilisateur *root* peut le lire et le modifier, personne d'autre ne peut y accéder.
- Le dossier *firefox* appartient à l'utilisateur *stc* et au groupe *stc* ; l'utilisateur *stc* peut y accéder, le lire et le modifier, personne d'autre ne peut y accéder.
- Le fichier *test.jpg* appartient à l'utilisateur *stc* et au groupe *www* ; l'utilisateur *stc* ainsi que les membres du groupe *www* peuvent le lire et le modifier, les autres utilisateurs peuvent seulement le lire.

Fondamental : Les droits *rwX*

- **r (read) : lire le contenu d'un fichier (exécuter *more* par exemple) ou d'un dossier (*ls*)**
- **w (write) : écrire un fichier (*nano*), écrire dans un dossier (*touch*)**
- **x (execute) : exécuter un fichier, entrer dans un dossier (*cd*)**

Complément

<http://www.linuxcertif.com/doc/keyword/ls/>³⁴

<http://www.linux-france.org/article/sys/fichiers/>³⁵

c) Gestion des droits sous Linux (chmod)

Rappel

Affichage des droits sous Linux

Fondamental : *chmod*

La commande `chmod` permet de modifier l'attribution des droits d'un fichier ou d'un dossier.

Pour cela on attribue 3 nombres de 0 à 7 correspondant aux 3 droits : utilisateur propriétaire (**u**), utilisateurs membres du groupe propriétaire (**g**), autres utilisateurs (**o**), sachant que :

1. **r vaut 4, w vaut 2, x vaut 1 ;**
2. **on somme les droits que l'on veut cumuler.**

Exemple

- `chmod 640 file` donne les droits **rw** (4+2) à *user*, **r** (4) à *group* et aucun droit (0) à *others*.
- `chmod 775 file` donne les droits **rwX** (4+2+1) à *user* et *group* et **rx** (4+1) à *others*.

34 - <http://www.linuxcertif.com/doc/keyword/ls/>

35 - <http://www.linux-france.org/article/sys/fichiers/>

Méthode : Droits courant pour un fichier (non exécutable)

- `chmod 666 file` (fichier non exécutable public)
- `chmod 644 file` (fichier non exécutable public, modifiable par soi uniquement)
- `chmod 600 file` (fichier privé)
- `chmod 400 file` (fichier privé en lecture seule)
- `chmod 444 file` (fichier public en lecture seule)

Droits courant pour un dossier (et de son contenu)

`chmod 777 dir` (dossier public)
`chmod 755 dir` (dossier public, modifiable par soi uniquement)
`chmod 700 dir` (dossier privé)
`chmod 500 dir` (dossier privé en lecture seule)
`chmod 555 dir` (dossier public en lecture seule)

Complément : `chmod -R` Changer les droits de tout un répertoire

`chmod -R XXX dir` affecte les droit XXX à *dir* et à tous les fichiers et sous-dossiers de *dir* (récursivement).

Complément : `chown` : Changer les propriétaires d'un fichier

`chown user:group target` permet de donner la propriété de *target* à l'utilisateur *user* et au groupe *group*.

Complément : `umask` : Changer les droits par défaut

- `umask -S` permet de voir les droits par défaut.
- `umask XYZ` enlève X, Y et Z à 666 pour les fichiers et 777 pour les dossiers lors de la création
- `umask 022` enlève les droits par défaut en écriture pour *group* et *others*;
- `umask 077` enlève tous les droits par défaut pour *group* et *others*.

d) Gestion des processus sous Linux (ps et kill)

Définition : Processus

Un processus est un programme en cours d'exécution.

Syntaxe : `ps` : afficher les processus

La commande `ps` permet d'afficher la liste des processus associés au terminal courant.

La commande `ps -aux` permet d'afficher la liste de tous les processus.

La commande `ps tree -p` permet d'afficher l'arbre de tous les processus.

Fondamental : `ps aux | grep xxx` : chercher un processus

La commande `ps aux | grep xxx` permet d'afficher tous les processus qui ont été lancé par *xxx* ou qui contiennent *xxx* dans leur nom.

Fondamental : `kill` : arrêter les processus

Pour arrêter un processus, trouver son numéro (*PID*) puis exécuter : `kill PID`.

Si le processus est bloqué et refuser de s'arrêter, exécuter `kill -9 PID` (cet arrêt forcé peut entraîner des perte de données).

Complément

`pkill xxx` (`pkill -9 xxx`) permet d'arrêter le processus de nom `xxx`.

`killall xxx` (`killall -9 xxx`) permet d'arrêter tous les processus de nom `xxx`.

Complément : top et htop

Les commandes `top` et `htop` permettent d'afficher les processus qui consomment le plus de ressources sur la machine.

e) Gestion de mémoires amovibles sous Linux (df -h et umount)

Définition : Montage et point de montage

Le **montage** d'une mémoire est le fait d'associer à cette mémoire un emplacement dans le système de fichier, appelé **point de montage**, pour pouvoir y accéder.

Montage automatique (/media)

Sous les Linux récents les mémoires amovibles (clés ou disques USB, CD-ROM...) se **montent** automatiquement. Les points de montage dépendent des distributions et des configurations, sous Ubuntu et Debian, il s'agit du dossier `/media` (on trouve aussi souvent le dossier `/mnt`).

Fondamental : Liste des disques et mémoires montées

Pour avoir des informations sur la liste des mémoires montées (et sur l'occupation de l'espace d'un disque ou d'une mémoire amovible) : `df -h`

Attention : Démontage

Une mémoire doit être démontée afin d'être débranchée, cela permet d'être certain que le système n'est plus en train d'agir sur la mémoire. Une déconnexion sans démontage peut entraîner la corruption des données stockées sur la mémoire : `umount point-de-montage`.

Par exemple : `umount /media/stc/MYUSB01`.

Complément : Montage manuel (et remontage)

Pour monter manuellement une mémoire qui ne l'aurait pas été automatiquement ou remonter une mémoire démontée (il faut être `root`) :

```
mount memoire point-de-montage
```

Par exemple : `sudo mount /dev/sdd1 /media/stc/MYUSB01`

Complément : Analyse d'occupation d'espace disque

- `du -h -d0 dir` espace occupé par `dir`.
- `du -h -d1 dir` espace occupé par chaque dossier de `dir`.
- `du -ha -d1 dir` espace occupé par chaque dossier et fichier de `dir`.
- `du -ha -d1 dir | sort -h` espace occupé par chaque dossier et fichier de `dir`, avec un tri par espace.

Complément : gnome-disks, baobab

gnome-disks est un utilitaire graphique qui permet de monter, démonter et formater des mémoires amovibles (ainsi que les disques durs du système).

baobab est un utilitaire graphique permettant de voir l'occupation des espaces sur les disques.

f) Quelques utilitaires sous Linux

tar et zip

tar est un format d'archivage qui permet de représenter une arborescence complète sous la forme d'un seul fichier.

gzip est un utilitaire de compression qui peut être utilisé pour compresser une archive Tar.

zip est un utilitaire d'archivage et de compression, équivalent fonctionnellement à Tar+Gzip ; il est moins utilisé sous Linux, mais plus sur Windows.

- `tar -cvf file.tar dir` crée un fichier *file.tar* avec le dossier *dir*.
- `tar -czvf file.tgz dir` crée un fichier *file.tgz* compressé avec *gzip*.
- `tar -tf file.tar` permet de voir le contenu du fichier archive *file.tar*.
- `tar -xvf file.tar` permet de décompresser le contenu du fichier *file.tar* dans l'emplacement courant.
- `zip file.zip dir` permet de zipper le dossier *dir* dans le fichier *file.zip*.
- `unzip file.zip -d destdir` permet de dézipper le fichier *file.zip* dans le dossier *destdir*.

cups-pdf et pdftk

pdftk permet de manipuler des fichiers PDF. Par exemple :

- `pdftk input.pdf cat 1-3 5 output output.pdf` permet de créer un fichier *output.pdf* avec les pages 1,2,3 et 5 du fichier *input.pdf*
- `pdftk t1.pdf t2.pdf output output.pdf` permet de créer un fichier *output.pdf* par concaténation des fichiers *t1.pdf* et *t2.pdf*.

cups-pdf permet d'imprimer dans un fichier PDF.

2. Introduction au protocole HTTP

a) HTTP

Définition

HTTP est un protocole de communication permettant à un client web d'interroger un serveur web.

Définition : Requêtes HTTP

HTTP repose un semble de requêtes : GET, POST, HEAD... Chacun permettant une demande particulière.

Exemple : Exemple de clients HTTP

- Les navigateurs web (Firefox, Chrome, Chromium, Internet Explorer, Safari...)
- Les clients en ligne de commande (wget, cURL...)
- Les clients programmés spécifiquement (la plupart des langages de programmation permettent de faire des requêtes HTTP)

Exemple : Exemple de serveurs HTTP

- Apache
- Nginx
- IIS
- ...

b) HTTP GET

Fondamental

La requête GET est la requête de base en HTTP, elle permet de demander une ressource au serveur.

C'est la requête que formule un navigateur web quand on tape une adresse web dans la barre d'adresse ou que l'on clic sur un lien .

Exemple

```

$ telnet www.perdu.com 80                               Connexion au serveur par telnet
Trying 208.97.177.124...
Connected to www.perdu.com.
Escape character is '^]'.

GET / http/1.1                                          Requête HTTP
Host: www.perdu.com

HTTP/1.1 200 OK                                        Réponse du serveur : headers
Date: Sat, 17 Aug 2013 11:59:04 GMT
Server: Apache
Accept-Ranges: bytes
X-Mod-Pagespeed: 1.1.23.1-2169
Vary: Accept-Encoding
Cache-Control: max-age=0, no-cache
Content-Length: 204
Content-Type: text/html

<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Interne
t ?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre> * <---- vous
&ccirc;tes ici</pre></strong></body></html>

```

<https://commons.wikimedia.org/w/index.php?curid=27753015>

c) Requête HTTP avec envoi de données au serveur

Méthode : Envoi de données au serveur avec GET

Il est possible d'envoyer des données au serveur avec la requête GET, on ajoute pour cela à l'URL demandé des couples clé-valeur en suivant la syntaxe : /adresse/de/la/ressource?cle1=valeur1&cle2=valeur2....

Exemple : Dans un navigateur web

<http://monsieur.fr/mapage.php?login=moi>

Méthode : Envoi de données au serveur avec POST

On préfère en général la méthode POST pour envoyer des données ; les couples clé-valeur sont alors envoyé dans le corps de la requête HTTP.

Exemple : Dans un navigateur web

```

1 POST /mapage.php
2 Host: monsieur.fr
3 login=moi&password=monsecret

```

Remarque : GET vs POST

- La taille des données envoyées au serveur est limitée avec GET. Les données envoyées sont visibles dans l'URL.
- La taille des données envoyées au serveur n'est pas limitée avec POST. Les données envoyées ne sont pas visibles dans l'URL, elles peuvent être chiffrées en HTTPS.

d) Traiter les requêtes HTTP avec un serveur PHP

Rappel

Requête GET ou POST par formulaire HTML (balise <form>)

Lorsqu'une requête HTTP envoie des données au serveur web, par exemple grâce à un lien <a> ou un formulaire <form> en HTML, les données envoyées doivent être traitées par un programme que l'on écrit spécifiquement sur le serveur.

Fondamental

Un serveur web/PHP peut gérer les données envoyées par une requête HTTP.

Lors de son chargement une page PHP contient un tableau de variables pour les données envoyées par la méthode GET et un autre pour les données envoyées par POST.

Syntaxe

On accède à ses données en utilisant la syntaxe :

```
$_GET["var1"]
```

```
$_GET["var2"]
```

ou

```
$_POST["var1"]
```

```
$_POST["var2"]
```

où `var1` et `var2` sont des noms de données dans la requête HTTP (par exemple le nom des contrôles dans le formulaire HTML à l'origine de la requête).

Exemple

```
1 <?php
2 echo 'Hello ' . $_POST["name"] ;
3 ?>
```

Complément

<http://php.net/manual/en/reserved.variables.get.php>³⁶

<http://php.net/manual/en/reserved.variables.post.php>³⁷

3. Requêtes HTTP avec une page web

a) Requête GET sans envoi de données (balise <a>)

Pour faire une requête GET en HTML, il suffit de faire un lien entre deux pages, avec la balise HTML <a>.

Exemple

```
1 <a href="page.html">Ceci est mon lien</a>
```

Lorsque l'utilisateur clique sur le lien Ceci est mon lien le navigateur envoie une requête GET pour récupérer `page.html` au serveur.

36 - <http://php.net/manual/en/reserved.variables.get.php>

37 - <http://php.net/manual/en/reserved.variables.post.php>

b) Requête GET avec envoi de données par l'URL (balise <a>)

Rappel

Requête HTTP avec envoi de données au serveur

Il est possible d'envoyer des données lors de la requête GET générée avec un lien <a>.

Exemple

```
1 <a href="page.php?d=mydata&x=otherdata">Ceci est mon lien</a>
```

Lorsque l'utilisateur clique sur le lien Ceci est mon lien le navigateur envoi une requête GET pour récupérer page.php au serveur et envoi les données d et x avec leurs valeurs respectives.

Attention

Les données qui sont envoyées sur le serveur doivent être traitées par une couche web applicative, par exemple en PHP.

Complément

Traiter les requêtes HTTP avec un serveur PHP

c) Requête GET ou POST par formulaire HTML (balise <form>)

Rappel

Un exemple de fichier XHTML

Définition : Formulaire

On appelle formulaire une interface permettant à un utilisateur humaine de saisir des données en vue dans une application informatique.

Définition : Contrôle

On appelle contrôle un élément d'un formulaire permettant d'effectuer une action : saisir une donnée, exécuter une requête...

La balise `form` du langage HTML permet de :

- créer un formulaire avec des contrôles,
- envoyer le contenu du formulaire à un serveur web grâce à une requête GET ou POST.

Exemple : Contrôle en HTML

- étiquette
- cases à cocher
- champs de saisie
- boutons radio
- listes à choix multiples
- ...

Exemple : Formulaire

Cf. "Exemple de formulaire HTML"

Animation 1 Exemple de formulaire HTML

```
1 <form metho="get" action="test.php">
2 <p><label>Nom</label> <input type="text" name="nom"></p>
3 <p><label>Prénom</label> <input type="text" name="prenom"></p>
4 <p><label>Age</label> <input type="text" name="age"></p>
```

```
5 <p><input type="submit"></p>
6 </form>
```

Complément

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/Form>³⁸

4. Complément

a) Administrer une machine Linux (introduction)

Cette section a pour objectif de présenter une partie des commandes utiles pour administrer un ordinateur sous Linux. Elle suppose l'accès en mode *root* à la machine.

i Agir en tant que root (su / sudo)

Un système Linux est généralement utilisé par plusieurs utilisateurs (ou *user*) identifiés par un compte.

Tous les systèmes ont en commun le premier de ces utilisateurs : *root*, l'administrateur du système qui possède tous les droits.

Syntaxe : su

Pour se connecter en tant qu'utilisateur *root* (et pouvoir faire les opérations qui lui sont réservées) : `su root`.

- Le mot de passe de l'utilisateur *root* est demandé par le système.

Syntaxe : sudo

Pour exécuter une commande en tant que *root* sans changer d'*user* : `sudo commande`.

- Le mot de passe de l'utilisateur lançant `sudo` est demandé par le système.
- Seuls certains utilisateurs appelés *sudoers* ont le droit d'exécuter la commande `sudo`.
- Pour ajouter un utilisateur dans la liste des *sudoers*, exécuté en tant que *root* (ou via un autre utilisateur *sudoers*) :
 - `adduser user sudo` (en tant que *root*)
 - `sudo adduser user sudo` (avec un utilisateur déjà membre des *sudoers*)

Attention : Debian (sudo)

Sous Debian la commande `sudo` n'est pas installée par défaut.

Attention : Ubuntu (su)

Sous Ubuntu l'utilisateur *root* n'est pas actif et il n'est donc pas possible d'exécuter la commande `su root`. On peut en revanche utiliser `sudo`, l'utilisateur créé à l'installation est membre des *sudoers*.

Pour activer le compte *root*, il faut lui attribuer un mot de passe avec la commande : `sudo passwd root`.

Complément : Agir en tant qu'un autre utilisateur

Pour agir en tant qu'utilisateur *user*, exécuter :

`su user`, ou `su - user` pour adopter l'environnement de *user*.

38 - <https://developer.mozilla.org/fr/docs/Web/HTML/Element/Form>

ii Installation d'applications sous Debian et Ubuntu (apt-get)

Pour installer une application exécuter la commande suivante en tant que *root* ou via *sudo* :

```
sudo apt-get install applications
```

Exemple : Installer virtualbox, lftp, cups-pdf, pdftk, dia, gimp et git

```
1 sudo apt-get install virtualbox lftp cups-pdf pdftk dia gimp git
```

Complément : Maintenir le système à jour

Pour mettre le système à jour, exécuter les deux commandes suivantes en tant que *root* ou via *sudo* :

1. sudo apt-get update
2. sudo apt-get upgrade

iii Gestion des utilisateurs sous Linux

Remarque

Seul l'utilisateur *root* (ou les *sudoers*) peuvent exécuter ces commandes.

Syntaxe : Utilisateurs

`adduser user` ajouter un utilisateur *user*.
`deluser user` supprimer un utilisateur *user*.
`passwd user` changer le mot de passe de *user*.
`cat /etc/passwd` lister les utilisateurs.

Syntaxe : Groupes

`addgroup group` ajouter un groupe *group*.
`usermod -aG group user` ajouter *user* à *group*.
`deluser user group` enlever *user* de *group*
`cat /etc/group` lister les utilisateurs avec leurs groupes.
`groups user` lister les groupes de *user* (cette commande est accessible aux utilisateurs non *root*)

b) Apache

i Installer Apache

Rappel

Notions de serveur et de client web

Méthode : Installer Apache sous Debian ou Ubuntu

```
1 sudo apt-get install apache2
```

- Tester l'accès au serveur Web en entrant l'adresse web dans un navigateur : `http://localhost`³⁹
- L'installation par défaut du serveur web permet de servir les fichiers situés dans le dossier `/var/www/html`.

Complément

<http://httpd.apache.org/docs>⁴⁰

39 - `http://localhost`

40 - `http://httpd.apache.org/docs`

<http://httpd.apache.org/docs/current/getting-started.html>⁴¹

ii Piloter Apache

Méthode

Le serveur web Apache sous Linux peut-être invoqué par la commande `apache2`, mais on utilise une commande plus haut niveau : `apache2ctl`.

Syntaxe

```
1 sudo apache2ctl start
2 sudo apache2ctl restart
3 sudo apache2ctl stop
```

Remarque

`apachectl` est un alias pour `apache2ctl` sur les systèmes actuels.

Remarque

Les processus lancés par le serveur web se nomment `apache2`.

Complément : Mode "graceful"

`sudo apache2ctl graceful` redémarre le serveur, mais après avoir attendu que les connexions en cours se terminent.

`sudo apache2ctl graceful-stop` éteint le serveur de la même façon.

Complément

<https://httpd.apache.org/docs/current/programs/apachectl.html>⁴²

iii Configurer Apache

La fichier de configuration standard d'Apache est : `httpd.conf`.

Sous Debian il s'agit du dossier `/etc/apache2` (lire le fichier `/etc/apache2/apache2.conf` en premier)

.htaccess

Outre la configuration centrale au niveau du serveur, Apache permet une configuration décentralisée par le biais de fichiers `.htaccess`.

Complément

<https://httpd.apache.org/docs/current/configuring.html>⁴³

<http://httpd.apache.org/docs/current/howto/htaccess.html>⁴⁴

Complément : VirtualHost (serveur virtuel)

<http://httpd.apache.org/docs/current/vhosts>⁴⁵

iv Installer PHP sur Apache

Méthode

```
1 sudo apt-get install php
```

41 - <http://httpd.apache.org/docs/current/getting-started.html>

42 - <https://httpd.apache.org/docs/current/programs/apachectl.html>

43 - <https://httpd.apache.org/docs/current/configuring.html>

44 - <http://httpd.apache.org/docs/current/howto/htaccess.html>

45 - <http://httpd.apache.org/docs/current/vhosts>

Tester l'accès au module PHP en déployant le fichier PHP contenant le code suivant :

```
1 <?php
2 phpinfo();
3 ?>
```

Attention

Par défaut l'installation d'un interpréteur PHP est en mode *production* et il n'affiche pas les erreurs. Il faut donc changer la configuration pour passer en mode *développement*.

1. **Le fichier de configuration de PHP est `php.ini`, situé par exemple dans :**
`/etc/php/7.0/apache2/php.ini`
2. **Éditez ce fichier et remplacez le paramètre `display_errors = Off` par `display_errors = On`**
3. **Relancez le serveur web :** `sudo service apache2 reload`

Complément

Il est également nécessaire d'installer un complément à PHP pour qu'il se connecte à PostgreSQL.

```
1 sudo apt-get install php-pgsql
2 sudo apache2ctl restart
```

Complément : Notes d'installation pour Debian

Les instructions `apt-get install php` et `apt-get install php-pgsql` permettent d'installer la dernière version de PHP sous Ubuntu. Sous Debian, ou pour installer une autre version, préciser la version de PHP :

```
1 apt-get install php5
2 apt-get install php5-pgsql
```

Complément : Notes d'installation pour Ubuntu 16.04

- `apache2ctl restart -> service apache2 reload`
- `apt-get install php -> apt-get install libapache2-mod-php`

c) Rappels système UTC

i Rappels architecture UTC

Les comptes d'UV sont des comptes sur des serveurs Linux de l'UTC (accès et espace disque réservé). Par exemple `nf17p001` est un compte sur le serveur `tuxa.sme.utc`.

Les comptes d'UV sont également des comptes d'accès sur des postes clients Linux ou Windows de l'UTC.

Méthode : Transférer des fichiers depuis un client vers le serveur auquel il est associé

- Les clients Windows de l'UTC sont configurés pour proposer un disque virtuel Z qui pointe sur l'espace du serveur Linux : les données déposées sur le disque Z sont donc physiquement stockées sur le serveur Linux.
- Les clients Linux de l'UTC sont configurés pour que le dossier `home` sur le client corresponde à un répertoire sur le serveur.
- Quelque soit le client il est possible d'utiliser un client SFTP (*Filezilla* par exemple) en se connectant au serveur (par exemple `tuxa.sme.utc`) avec son compte UTC (par exemple `nf17p001`).

Rappel

Connexion à un serveur Linux avec SSH

ii Mise en ligne d'un fichier HTML sur le serveur tuxa.sme.utc de l'UTC

- Les fichiers déposés dans un dossier `~/public_html` seront accessibles par le serveur web qui tourne sur *tuxa.sme.utc*.
- Les fichiers sont ensuite accessibles sur le Web via une adresse telle que : `http://tuxa.sme.utc/~nf17pXXX/monfichier.html`
- Il faut les droits en lecture sur les fichiers (`chmod 644`) et sur les dossiers (`chmod 755`).

B. Exercices

1. Linux

[20 min]

L'objectif de cet exercice est de se rappeler les commandes de base pour utiliser un compte Linux.

Question 1

Ouvrir une connexion SSH sur un serveur Linux de votre choix (proposant un accès SSH).

Indices :

Connexion à un serveur Linux avec SSH

À l'UTC vous pouvez vous connecter avec votre compte personnel au serveur `kappa.utc.fr` ou avec votre compte d'UV au serveur `tuxa.sme.utc`.

Question 2

Rendez-vous dans votre répertoire home.

Indice :

Commandes de bases sous Linux : `cd`, `ls`, `mkdir`, `rm`, `find`, `cat`, `nano`...

Question 3

Affichez le chemin de l'endroit où vous vous trouvez.

Indice :

Commandes de bases sous Linux : `cd`, `ls`, `mkdir`, `rm`, `find`, `cat`, `nano`...

Question 4

Créez un fichier texte README contenant votre nom dans votre répertoire home.

Indice :

Commandes de bases sous Linux : `cd`, `ls`, `mkdir`, `rm`, `find`, `cat`, `nano`...

Question 5

Affichez les droits qui ont été attribués par défaut à ce fichier.

Indice :

Affichage des droits sous Linux (`ls -al`)

Question 6

Changez les droits du fichier README pour les donner en lecture à tout le monde (en conservant les droits en modification de votre côté).

Indice :

Gestion des droits sous Linux (chmod)

Question 7

Vérifiez que les droits ont bien été attribués comme vous le souhaitez.

Indice :

Affichage des droits sous Linux (ls -al)

Question 8

Regardez le contenu du fichier README d'un autre utilisateur.

Indice :

Commandes de bases sous Linux : cd, ls, mkdir, rm, find, cat, nano...

Question 9

Ouvrez votre fichier README avec gedit.

Ajoutez une ligne, sans enregistrer la modification.

Indice :

Lancer des applications sous Linux

Question 10

Trouvez le PID du processus gedit

Indice :

Gestion des processus sous Linux (ps et kill)

Question 11

Arrêtez votre processus gedit.

Indice :

Gestion des processus sous Linux (ps et kill)

2. HTTP

Question 1

Créer un fichier *test.txt* à la racine de l'espace de publication de votre serveur HTTP.

Indice :

```
1 nano /dossier-web/test.txt
```

Question 2

Accédez à votre serveur web en vous connectant sur le port 80 avec le client *telnet* (*telnet* est un client TCP/IP généraliste fonctionnant en envoyant et recevant des lignes de texte).

Indice :

```
1 telnet nom-du-serveur 80
```

Question 3

Exécutez la requête HTTP 1.0 permettant de récupérer le fichier `test.txt`.

Indice :

```
1 GET /chemin/fichier HTTP/1.0
```

Question 4

Exécutez à nouveau la même requête en utilisant le client `wget`.

Indice :

```
1 wget serveur:port/fichier
```

3. XHTML, HTTP, PHP

Créez le fichier `action.php` ci-après, déployez-le sur un serveur web avec PHP.

```
1 <?php
2     $pName=$_GET['name'];
3     echo "Votre nom est " . $pName;
4 ?>
```

Question 1

Testez ce fichier en y accédant avec un navigateur web, qu'obtenez-vous ?

Indice :

Entrez l'adresse web du fichier dans votre navigateur, par exemple `http://monserveur.fr/action.php`.

Question 2

Testez ce fichier en envoyant la donnée `name=Nobody` avec un navigateur web en passant une requête HTTP GET.

Indice :

Requête HTTP avec envoi de données au serveur

Question 3

Créez le fichier `form.html` ci-après, déployez-le dans le même dossier que `action.php`. Testez-le en entrant une valeur dans le formulaire, afin de vérifier la mécanique du passage de variable.

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2     <head>
3         <title>Formulaire</title>
4     </head>
5     <body>
6         <form method="get" action="action.php">
7             <p><label>Entrez votre nom </label> <input type="text" name="name"
8             /></p>
9             <p><input type="submit" /></p>
```

```
9         </form>
10     </body>
11 </html>
```

Question 4

Transformez `form.html` et `action.php` afin que :

- la méthode utilisée soit POST et non GET,
- le fichier renvoyé soit un fichier XHTML et non une simple chaîne de caractère.

Indice :

Un exemple de fichier XHTML

Requêtes SQL avec PHP et PostgreSQL

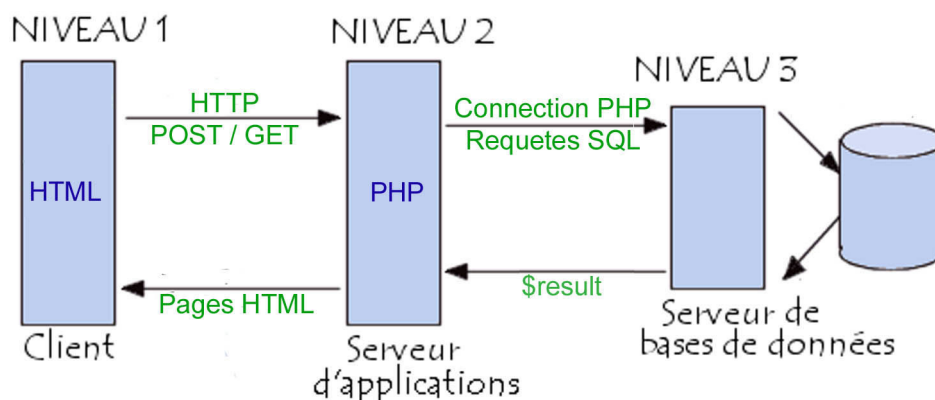


A. Cours

1. Connexion d'une page PHP à une base de données PostgreSQL avec PDO

a) Architecture PHP/BD

Exemple



Navigateur Web

Apache + PHP

Image 10 Exemple d'architecture 3-tiers : PHP/BD (commentcamarche.net - © 2003 Pillou - GNU FDL)

b) PHP Data Objects

Définition

« PDO fournit une interface d'abstraction à l'accès de données, ce qui signifie que vous utilisez les mêmes fonctions pour exécuter des requêtes ou récupérer les données quelque soit la base de données utilisée. »

<http://www.php.net/manual/fr/intro.pdo.php>⁴⁶

Syntaxe : Connexion à PostgreSQL avec PDO en PHP

```
1 $conn = new PDO('pgsql:host=hostname;port=5432;dbname=db', 'user', 'pass');
```

Syntaxe : Exécution de requête SQL

```
1 $sql = "...";
2 $resultset = $connexion->prepare($sql);
3 $resultset->execute();
```

Syntaxe : Traitement de résultat de requête SQL

```
1 while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
2     ... $row['...'];
3 }
4
```

Complément : Fixer le search_path à un schéma

```
1 $result = $Conn->exec('SET search_path TO ...');
```

Complément

<http://php.net/manual/fr/book.pdo.php>⁴⁷

c) Accès à une BD en écriture (INSERT, UPDATE, DELETE)

Exemple

```
1 <?php
2
3 /** Connexion **/
4 $connexion = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test',
5     'test');
6
7 /** Préparation et exécution de la requête **/
8 $sql = "INSERT INTO médicament (nom) VALUES ('Nouveau')";
9 $result = $connexion->prepare($sql);
10 $result->execute();
11
12 /** Traitement du résultat **/
13 if ($result) {
14     echo "'Nouveau' inséré";
15 }
16 else {
17     echo "Erreur lors de l'insertion";
18 }
19
20 /** Déconnexion **/
21 $connexion=null;
22 ?>
```

46 - <http://www.php.net/manual/fr/intro.pdo.php>

47 - <http://php.net/manual/fr/book.pdo.php>

d) Accès à une BD en lecture (SELECT)

Exemple

```

1 <?php
2
3 /** Connexion **/
4 $connexion = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test',
   'test');
5
6 /** Préparation et exécution de la requête **/
7 $sql = "SELECT nom FROM médicament;";
8 $resultset = $connexion->prepare($sql);
9 $resultset->execute();
10
11 /** Traitement du résultat **/
12 while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
13     echo $row['nom'];
14     echo " ";
15 }
16
17 /** Déconnexion **/
18 $connexion=null;
19
20 ?>

```

e) Requêtes préparées

L'étape de préparation de requête permet de précompiler une requête SQL côté serveur. Cela permet d'optimiser les performances lorsqu'une requête est utilisée plusieurs fois. Cela permet également de paramétrer les requêtes au niveau d'un langage applicatif (comme PHP).

Conseil

Utiliser des requêtes paramétrées au niveau des langages applicatifs.

Exemple : Requête INSERT non préparée en PHP

```
1 $result = $connexion->query("INSERT INTO médicament (nom) VALUES ('Nouveau')");
```

Exemple : Requête INSERT préparée en PHP

```
1 $result = $connexion->prepare("INSERT INTO médicament (nom) VALUES ('Nouveau')");
2 $result->execute();
```

Exemple : Requête INSERT préparée et paramétrée en PHP

```

1 $result = $connexion->prepare("INSERT INTO médicament (nom) VALUES (?)");
2 $result->bindParam(1, $name);
3
4 $name = "Nouveau1";
5 $result->execute();
6
7 $name = "Nouveau2";
8 $result->execute();

```

Complément : Requête préparée en PHP

<http://php.net/manual/fr/pdo.prepared-statements.php>⁴⁸

48 - <http://php.net/manual/fr/pdo.prepared-statements.php>

Complément : Requête préparée sous PostgreSQL (PREPARE)

<http://docs.postgresql.fr/current/sql-prepare.html>⁴⁹

2. Complément

a) Compléments pour la programmation PHP

i Fonctions en PHP

Syntaxe

```

1 function Nom_De_La_Fonction(argument1, argument2, ...) {
2     liste d'instructions
3     ...
4     return valeur_ou_variable;
5     ...
6 }
```

Complément

<http://www.php.net/manual/fr/language.functions.php>⁵⁰

ii Variables de session

Définition : Variable de session PHP

Une variable de session PHP est une variable stockée sur le serveur.

C'est une variable temporaire qui a une durée limitée et est détruite à la déconnexion (fermeture du navigateur).

Les variables de session sont **partagées** par toutes les pages PHP d'une session (accès depuis un même navigateur). Elles permettent donc le passage d'information entre pages.

Exemple : Exemple d'utilisation

- Sauvegarde d'identifiants de connexion
- Sauvegarde d'un panier d'achat
- ...

Syntaxe : Démarrer une session

```
1 <?php session_start(); ?>
```

Ce code permet de charger le fichier contenant les variables de session sur le serveur, ou s'il n'existe pas de la créer.

Remarque

Ce code est à placer au début de toutes les pages PHP qui souhaitent utiliser les variables de sessions, avant tout autre code PHP ou HTML.

Syntaxe : Utiliser les variables

```

1 <?php
2 ...
3 $_SESSION['variable'] = valeur ;
4 ...
```

49 - <http://docs.postgresql.fr/current/sql-prepare.html>

50 - <http://www.php.net/manual/fr/language.functions.php>

```
5 ?>
```

Un tableau association `$_SESSION` est alors mis à disposition pour gérer des variables.

Exemple

```
1 <?php
2 // page1.php
3 session_start();
4 ?>
5 <html>
6 <body>
7 <h1>Page 1</h1>
8 <?php
9 $_SESSION['login'] = 'me';
10 $_SESSION['mdp'] = 'secret';
11 ?>
12 <a href="page2.php">page 2</a>
13 </body>
14 </html>
```

```
1 <?php
2 // page2.php
3 session_start();
4 ?>
5 <html>
6 <body>
7 <h1>Page 2</h1>
8 <?php
9 echo $_SESSION['login'] ;
10 echo "<br/>" ;
11 echo $_SESSION['mdp'] ;
12 ?>
13 </body>
14 </html>
```

La page `page2.php` est en mesure d'afficher les informations de la page `page1.php`.

Syntaxe : Autres instructions

- Supprimer une variable : `unset($_SESSION['variable'])`
- Supprimer toutes les variables : `session_unset()`
- Supprimer la session : `session_destroy()`

Complément : Sources

<http://www.phpsources.org/tutoriel-SESSIONS.htm>⁵¹

<http://www.php.net/manual/fr/book.session.php>⁵²

Complément : Cookies

Les sessions s'appuient sur les cookies, fichiers de données gérés côté client par le navigateur Web, pour stocker l'identifiant de session. Il est possible d'utiliser des sessions sans cookie, en passant l'identifiant de session dans l'URL.

<http://www.phpsources.org/tutoriel-cookies.htm>⁵³

51 - <http://www.phpsources.org/tutoriel-SESSIONS.htm>

52 - <http://www.php.net/manual/fr/book.session.php>

53 - <http://www.phpsources.org/tutoriel-cookies.htm>

iii Objets en PHP

Syntaxe : Déclaration d'une classe

```

1 class NomClasse {
2     // Déclarations des attributs
3     public $donneeMembre1;
4     public $donneeMembre2;
5     ...
6     // Déclarations du constructeur
7     function __construct () {
8         ...
9     }
10    // Déclarations des méthodes
11    public function Nom_de_la_fonction_membre1(parametres) {
12        ...
13    }
14    ...
15 }

```

Remarque : This

Le mot clé `$this` permet d'accéder à l'objet en cours lors de la déclaration des méthodes.

Syntaxe : Instanciation d'objets

```

1 $Nom_de_l_objet = new Nom_de_la_classe;

```

Syntaxe : Accès aux propriété

```

1 $Nom_de_l_objet->Nom_de_la_propriété = Valeur;

```

Syntaxe : Accès aux méthodes

```

1 $Nom_de_l_objet->Nom_de_la_méthode (parametre1,parametre2,...);

```

Exemple : Classe de connexion à une base de données PostgreSQL

```

1 <?php
2 class Connect {
3     var $fHost;
4     var $fPort;
5     var $fDbname;
6     var $fUser;
7     var $fPassword;
8     var $fConn;
9     function __construct () {
10        $this->fHost="foo.fr";
11        $this->fPort="5432";
12        $this->fDbname="myDb";
13        $this->fUser="Me";
14        $this->fPassword="Secret";
15    }
16    function mConnect () {
17        $this->fConn = pg_connect("host=$this->fHost port=$this->fPort
18        dbname=$this->fDbname user=$this->fUser password=$this->fPassword") or die('Échec
19        de la connexion : ' . pg_last_error());
20    }
21    function mClose () {
22        pg_close($this->fConn);
23    }
24 }
25 ?>

```

Exemple : Utilisation de la classe de connexion

```

1 <?php
2     include "connect_class.php";
3     $vConnect = new Connect;
4     $vConnect->mConnect();
5 ?>

```

Complément

<http://www.php.net/manual/fr/language.oop5.php>⁵⁴

iv Include

b) Interfaces dédiées

i Interfaçage avec PostgreSQL

Syntaxe : Connexion à la BD

```

1 $vConn = pg_connect("host=$vHost port=$vPort dbname=$vDbname user=$vUser
password=$vPassword");

```

Syntaxe : Interrogation de la BD

```

1 $vSql = "SELECT ...";
2 $vQuery=pg_query($vConn, $vSql);
3 while ($vResult = pg_fetch_array($vQuery, null, PGSQL_ASSOC)) {
4     ... $vResult[nom_attribut]...
5 }

```

Syntaxe : Alimentation de la BD

```

1 $vSql="INSERT ...";
2 $vQuery=pg_query($vConn, $vSql);

```

Syntaxe : Déconnexion de la base de données

```

1 pg_close($conn)

```

ii Interfaçage PHP avec Oracle

Fait à partir de <http://www.php.net/manual/en/ref.oci8.php>⁵⁵.

Remarque

L'API OCI a remplacé l'ancienne API ORA (qui n'est plus supportée dans PHP).

Syntaxe : Connexion à la base de données

`oci_connect` établit une connexion entre le serveur PHP et un serveur Oracle.

```

1 connection_id oci_connect(string username, string password, string bd);

```

Exemple

```

1 if (! $conn=oci_connect($user, $passwd, $bd)) {

```

54 - <http://www.php.net/manual/fr/language.oop5.php>

55 - <http://www.php.net/manual/en/ref.oci8.php>

```

2   echo "Impossible d'établir la connexion ";
3   exit;
4 }

```

Syntaxe : Préparation d'une requête

`oci_parse` analyse une requête SQL et retourne un pointeur sur un *statement* (espace de requête).

```

1 statement_handle oci_parse(connection_id connection, string query);

```

Exemple

```

1 if(! $statement=oci_parse($conn, $sql)) {
2   echo "Impossible de préparer la requête";
3   exit;
4 }

```

Syntaxe : Exécution d'une requête

`oci_execute` exécute une commande déjà préparée avec `OCIParse`. Il est possible de spécifier le mode d'exécution des transactions (par défaut, il est en auto-commit, c'est à dire que l'ordre commit est passé automatiquement après chaque instruction SQL). Il est préférable d'utiliser le mode `OCI_DEFAULT` qui permet de contrôler les commits.

```

1 boolean oci_execute(statement_handle statement, int mode);

```

`$mode` permet de paramétrer le `commit` (par défaut, le commit est envoyé automatiquement si l'exécution est correcte).

Exemple

```

1 if(! oci_execute($statement, OCI_DEFAULT)) {
2   echo "Impossible d'exécuter la requête";
3   exit;
4 }

```

Syntaxe : Commit d'une transaction

`oci_commit` valide la transaction en cours sur une connexion.

```

1 boolean oci_commit(connection_id connection);

```

Exemple

```

1 if(! oci_commit($conn)) {
2   echo "Impossible de valider la transaction";
3   exit;
4 }

```

Remarque : Rollback

`oci_rollback` permet d'annuler une transaction.

Syntaxe : Récupération d'enregistrements

`oci_fetch_array` retourne la ligne suivante (pour une instruction `SELECT`) dans un tableau à une dimension (il écrasera le contenu du tableau s'il existe). Par défaut, le tableau sera un tableau à double index, numérique et associatif.

```
1 array oci_fetch_array(statement_handle statement)
```

Exemple

```
1 while ($row=oci_fetch_array($statement)) {
2     echo $results[0];
3     echo $results[1];
4     ...
5 }
```

Syntaxe : Déconnexion de la base de données

`oci_close` ferme une connexion Oracle.

```
1 boolean oci_close(connection_id connection);
```

Exemple

```
1 if (! oci_close($conn)) {
2     echo "Impossible de fermer la connexion ";
3     exit;
4 }
```

iii Interfaçage PHP avec MySQL

Fait à partir de www.commentcamarche.net⁵⁶. Copyright 2003 Jean-François Pillou. Document soumis à la licence GNU FDL.

Complété à partir de MySQL 4 : Installation, mise en oeuvre et programmation [Thibaud03].

Connexion au serveur

```
1 mysql_connect($host,$user,$passwd);
```

Connexion à la base de données

```
1 mysql_select_db($bdd);
```

Exécution de requête SQL

```
1 $result=mysql_query($query)
```

Traitement de résultat de requête SELECT

```
1 /* Test d'exécution de la requête */
2 if (! mysql_fetch_row($result)) {
3     echo "Aucun enregistrement ne correspond\n";
4 }
5 else {
6     while($row = mysql_fetch_row($result)) {
7         ... $row[1] ... $row[2] ...
8     }
9 }
```

Déconnexion de la base de données

```
1 mysql_close();
```

56 - www.commentcamarche.net

B. Exercices

1. Super-transferts

[30 minutes]

L'entreprise de ventes de figurines de super-héros GARVEL a monté un partenariat avec les deux sites de ventes en ligne *makemoney.com* et *dobusiness.com*. Chaque entreprise lui demande de mettre à disposition respectivement un fichier CSV et un fichier XML pour le transfert du catalogue, stocké dans une base de données PostgreSQL.

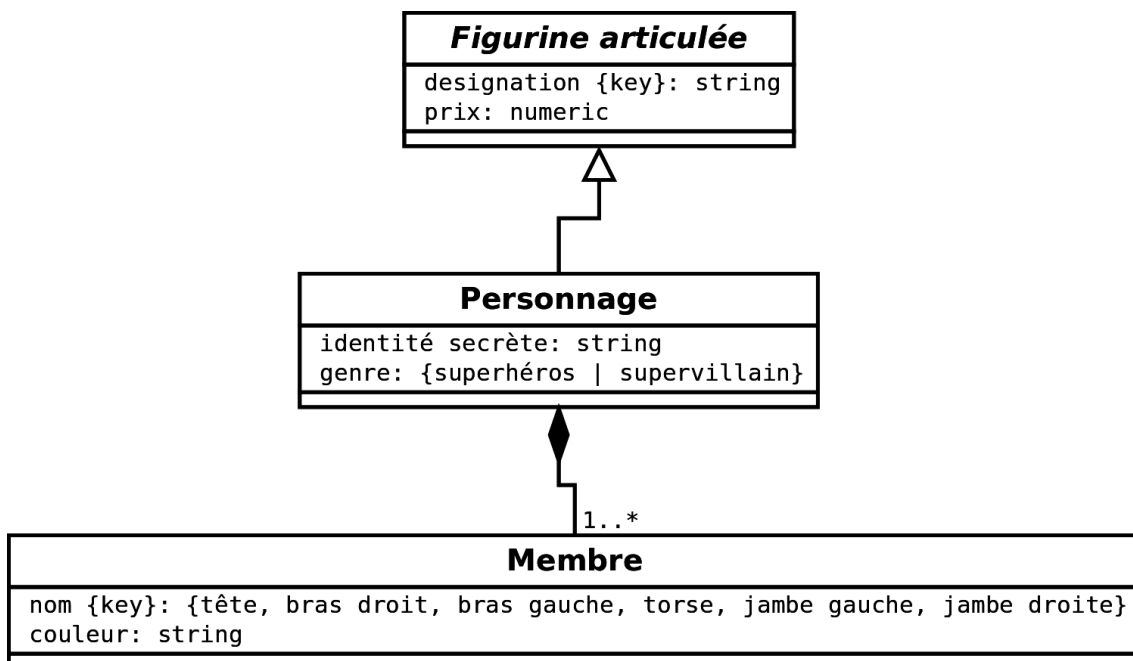
Le code devra être exécuté et testé.

Fichier CSV et fichier XML requis

```
1 Superman;15
2 Batman;12
3 Superchild;12
4 ...
```

```
1 <catalogue>
2   <figurine designation='Superman' prix='1555' />
3   <figurine designation='Batman' prix='12' />
4   <figurine designation='Superchild' prix='12' />
5   ...
6 </catalogue>
```

Modèle de la base de données



Modèle UML Figurines GARVEL (extrait)

Question 1

Créer la base de données correspondant à ce modèle.

Créer une vue `vfigurine` permettant de retourner les champs `designation` et `prix`.

Créer un utilisateur `customer` permettant de lire la vue figurine.

Question 2

Réaliser un script PHP *csv.php* permettant de se connecter à la base PostgreSQL et d'afficher la désignation et le prix au format CSV, en suivant l'exemple ci-après.

Indice :

Accès à une BD en lecture (*SELECT*)

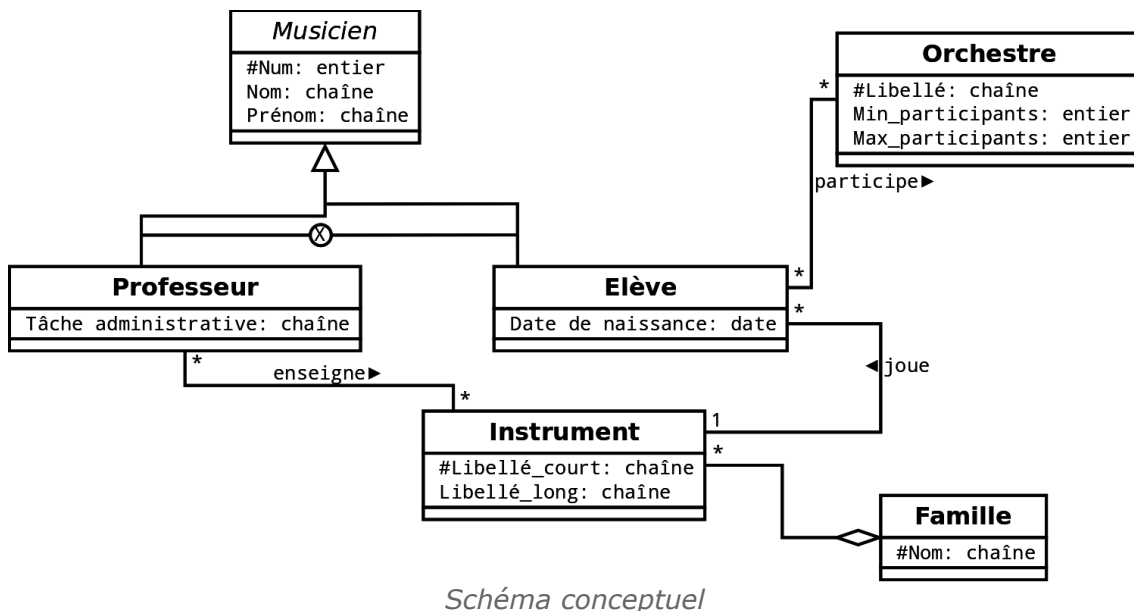
Question 3

Réaliser un script PHP permettant de se connecter à la base et d'afficher la désignation et le prix selon un schéma XML, en suivant l'exemple ci-après.

2. À l'école de musique

[30 min]

Une école de musique souhaite gérer les inscriptions aux différentes classes d'instrument et aux orchestres à l'aide d'une base de données.



```

1 Famille (#nom:chaîne)
2 Instrument (#lib:chaîne, lib_long:chaîne, famille=>Famille(nom))
3 Professeur (#num:entier, nom:chaîne, prénom:chaîne, tâche:chaîne)
4 Eleve (#num:entier, nom:chaîne, prénom:chaîne, date_naissance:date,
  inst=>Instrument(lib))
5 Orchestre (#lib:chaîne, min:entier, max:entier)
6 Enseigne (#num=>Professeur(num), #lib=>Instrument(lib))
7 Participe (#num=>Eleve(num), #lib=>Orchestre(lib))
  
```

Question 1

Créer la base de données permettant de gérer les **élèves** et les **instruments** (sans les familles). Insérer des données exemple.

Question 2

Écrire le code SQL qui permet d'afficher la liste des élèves (nom et prénom) triés par instrument.

Question 3

Écrire le code SQL qui permet d'afficher la liste des instruments avec le nombre d'élèves associés (on affichera même les instruments dont personne ne joue).

Question 4

Écrire le code SQL qui permet d'afficher la liste des instruments avec le nombre de places restantes par instruments, sachant qu'il y a 20 places disponibles par instrument.

Question 5

Compléter le code PHP ci-dessous pour afficher le nombre de places disponibles par instruments listés par leur libellé long.

On utilisera une requête paramétrée.

```

1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3 <title>École de musique</title>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5 </head>
6
7 <body>
8 <h2>Liste des places disponibles par instrument</h2>
9 <table border="1">
10 <tr> <td><b>Instruments</b></td> <td><b>Places disponibles</b></td> </tr>
11
12 <?php
13 /* Nombre maximum d'élèves autorisé par instrument */
14 $max_eleves = 20;
15
16 /* Nombre maximum d'élèves autorisé par instrument */
17 $connexion = new PDO('pgsql:host=localhost;port=5432;dbname=musique', 'me',
18 'mypassword');
19
20 /** Préparation et exécution de la requête **/
21 $sql = "SELECT I.lib AS lib, ? - COUNT(E.inst) AS dispo ...";
22 $resultset = $connexion->prepare($sql);
23 $resultset->bindParam(1, ...);
24 $resultset->execute();
25
26 while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
27     echo "<tr>";
28     echo "<td>" . $row[...] . "</td>";
29     echo "<td>" . $row[...] . "</td>";
30     echo "</tr>";
31 }
32
33 /** Déconnexion **/
34 $connexion=null;
35 ?>
36 </table>
37 </body>
38 </html>

```

Indice :

Requêtes préparées

C. Devoirs

1. Recensement

[45 min]

Soit le fichier CSV suivant :

```

1 "departement";"nbhabitants"
2 "01 - Ain";529378
3 "02 - Aisne";552320
4 "03 - Allier";357110
5 "04 - Alpes-de-Haute-Provence";144809
6 "05 - Hautes-Alpes";126636
7 "06 - Alpes-Maritimes";1022710
8 "07 - Ardèche";294522
9 "08 - Ardennes";299166
10 "09 - Ariège";142834
11 "10 - Aube";301388

```

Question 1

Établir un modèle relationnel et un code SQL permettant d'accueillir le contenu de ce fichier (on nommera la table `dpt1`).

Question 2

Implémenter ce modèle sous Postgres et importer le fichier CSV.

Indice :

Importer un fichier CSV

Question 3

En quelle forme normale est le modèle ? Normalisez le en 3NF (on nommera la table `dpt2`).

Question 4

Écrivez et exécutez la requête SQL permettant de migrer `dpt1` dans `dpt2`.

Indices :

```

1 INSERT INTO dpt2 (...)
2 SELECT ...
3 FROM dpt1

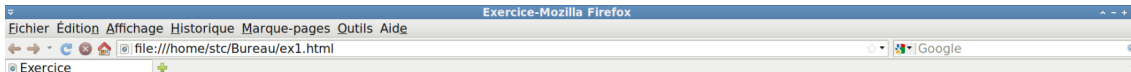
```

Utiliser la fonction `SUBSTR`.

Si vous avez défini `#num` comme entier, vous aurez besoin de la fonction `CAST(madonnée AS montype)`

Question 5

Écrivez le programme PHP permettant d'afficher le contenu de la table normalisée, tel que présenté ci-dessous.



Population par département

Numéro	Nom	Population
01	Ain	529378
02	Aisne	552320
...

- Département le plus peuplé : **Paris** (2147857)
- Département le moins peuplé : **Hautes-Alpes** (126636)

Terminé

Image 11 Page HTML visualisée avec un navigateur Web

2. Devoirs en ligne

[45 min]

Une université propose des formations pour ses étudiants via une plate-forme d'enseignement en ligne. Plusieurs devoirs sont proposés, chacun ayant une description et une même date de rendu pour tous les étudiants souhaitant le faire. Le but de cet exercice est de réaliser un site Web permettant aux étudiants de consulter leurs notes.

Base de données

On vous donne ci-dessous le code SQL LDD de la base de données sous PostgreSQL ainsi qu'un exemple de code SQL LMD pour l'insertion de données.

```

1 CREATE TABLE tEtudiant(
2   login CHAR(8) PRIMARY KEY,
3   nom VARCHAR(50),
4   prenom VARCHAR(50)
5 );
6 CREATE TABLE tDevoir(
7   num INTEGER PRIMARY KEY,
8   dateRendu DATE UNIQUE NOT NULL,
9   description VARCHAR(50) NOT NULL
10 );
11 CREATE TABLE tNote(
12   etudiant CHAR(8) REFERENCES tEtudiant(login),
13   devoir INTEGER REFERENCES tDevoir(num),
14   valeur INTEGER NOT NULL,
15   PRIMARY KEY(etudiant, devoir),
16   CHECK (valeur BETWEEN 0 AND 20)
17 );
18
19 INSERT INTO tEtudiant(login, nom, prenom) VALUES ('bfrankli', 'Franklin',
20   'Benjamin');
21 INSERT INTO tDevoir(num, dateRendu, description) VALUES (1, '10-05-
22   2013', 'Structures de donnees en C');
23 INSERT INTO tNote(etudiant, devoir, valeur) VALUES ('bfrankli', 1, 15);

```

Site Web

Le site Web sera composé :

- d'une page d'accueil (`accueil.html`) comportant un formulaire où l'étudiant doit entrer son login ;
- d'une page présentant les notes d'un étudiant pour chaque devoir qu'il a rendu ainsi que sa moyenne générale (`notes.php`).

La seconde page (`notes.php`) est appelée à partir de la première (`accueil.html`). On supposera dans le reste de l'exercice que tous les fichiers sont situés directement dans le répertoire `public_html` du serveur (sans sous-répertoires).

Notes de l'étudiant Benjamin Franklin (*bfrankli*)

Devoir	Date de rendu	Note
Structures de données en C	10-05-2013	15
...

Moyenne générale : **15.0**

[Retour à l'accueil](#)

Exemple d'affichage pour notes.php

Question 1

Écrivez le formulaire HTML de la page d'accueil (`accueil.html`). On utilisera la méthode HTTP POST.

Indice :

Requête GET ou POST par formulaire HTML (balise `<form>`)

```

1 <form method="post" action="notes.php">
2   <p>Login : </p>
3   <input type="text" name="login"/>
4   <input type="submit"/>
5 </form>
```

Dans les questions suivantes, on supposera que le login spécifié dans le formulaire de `accueil.html` existe effectivement dans la base de données (on ne fera donc pas de test pour le vérifier) et on se référera à l'exemple d'affichage pour `notes.php` pour le résultat souhaité.

Question 2

Écrivez le code PHP permettant de générer le code HTML du titre de la page "Notes de l'étudiant..." pour le login en question.

Indices :

Traiter les requêtes HTTP avec un serveur PHP

```

1 echo "<h1>Notes de l'étudiant " . $vPrenom . " " . $vNom . " (<i>" .
  $vLogin . "</i>)</h1>";
```

Question 3

Écrivez le code PHP permettant de générer le code HTML du tableau des notes de l'étudiant. Les devoirs seront affichés par ordre croissant de date de rendu.

Question 4

Écrivez le code PHP permettant de calculer la moyenne générale de l'étudiant.

Question 5

Écrivez le code HTML du lien hypertexte "Retour à l'accueil" et finalisez la page `notes.php`.

Indice :

```
1 <a href="accueil.html">Retour à l'accueil</a>
```

Glossaire



Client

Un client est un programme informatique qui a pour fonction d'envoyer des requêtes à un autre programme informatique, appelé serveur, d'attendre le résultat de cette requête et de traiter le résultat de la requête. Notons qu'un programme peut-être client vis à vis d'un programme et serveur vis à vis d'un autre. On ne prend pas ici le terme client dans son acception matérielle, qui signifie alors un ordinateur qui a pour fonction d'héberger des programmes clients.

Logiciel libre

Aujourd'hui, un logiciel est considéré comme libre, au sens de la *Free Software Foundation*, s'il confère à son utilisateur quatre libertés (numérotées de 0 à 3)⁷ :

- 0. la liberté d'exécuter le programme, pour tous les usages ;
- 1. la liberté d'étudier le fonctionnement du programme et de l'adapter à ses besoins ;
- 2. la liberté de redistribuer des copies du programme (ce qui implique la possibilité aussi bien de donner que de vendre des copies) ;
- 3. la liberté d'améliorer le programme et de distribuer ces améliorations au public, pour en faire profiter toute la communauté.

L'accès au code source est une condition d'exercice des libertés 1 et 3.

(Wikipédia ; fsf.org)

OS (Operating System, Système d'Exploitation)

Un ordinateur est une machine qui a besoin de programmes pour faire quelque chose. Lorsqu'un ordinateur démarre il exécute en général un programme interne, dit programme d'amorçage, puis le premier programme externe qu'il exécute est le **système d'exploitation**. C'est le système d'exploitation qui permet ensuite d'exécuter les applications (traitement de texte, navigateur web, client mail...).

Exemples de système d'exploitation :

- Windows
- MacOSX
- GNU/Linux
- Android
- iOS
- FreeBSD
- ...

RAID

La technologie RAID permet de répartir de l'information à stocker sur plusieurs "petits" disques, au lieu de la concentrer sur un seul "gros" disque. Cette technologie permet donc d'améliorer les performances (les accès disques pouvant être parallélisés) et d'améliorer la sûreté (en répartissant les risques de crash et en jouant sur une redondance des données). Il



existe plusieurs types d'architecture RAID, privilégiant ou combinant la parallélisation et la redondance.

Sérialisation

Processus consistant à enregistrer des données en mémoire vive (par exemple des objets) sous une forme permettant leur persistance, typiquement sur une mémoire secondaire.

Serveur

Un serveur est un programme informatique qui a pour fonction de recevoir des requêtes d'un autre programme, appelé client, de traiter ces requêtes et de renvoyer en retour une réponse. Notons qu'un programme peut-être serveur vis à vis d'un programme et client vis à vis d'un autre. On ne prend pas ici le terme serveur dans son acception matérielle, qui signifie alors un ordinateur qui a pour fonction d'héberger des programmes serveurs.

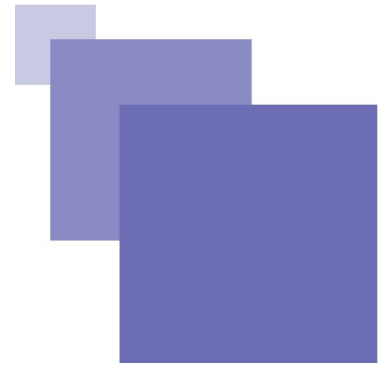
XSL-FO

XSL-FO (FO pour Formatting Objects) est la seconde partie du standard W3C « Extensible Stylesheet Language Family ». Il propose un langage XML de mise en forme de documents imprimables.

Exemple d'extrait de code FO :

- `<fo:block font-family="Times" font-size="12pt">Ceci est un paragraphe en police Times de taille 12pt</fo:block>`.

Signification des abréviations



- BD	Base de Données
- CSV	Comma Separated Values
- E-A	Entité-Association
- HTML	HyperText Markup Language
- OS	Operating Système (Système d'Exploitation)
- R	Relationnel
- RO	Relationnel-Objet
- SGBD	Système de Gestion de Bases de Données
- SGML	Standard Generalized Markup Language
- SQL	Structured Query Language
- UML	Unified Modeling Language
- W3C	World Wide Web Consortium
- XML	eXtensible Markup Language



Références



[Putty]

PuTTY est utile si vous êtes sur un système Windows, pour obtenir une connexion SSH et une console sur un serveur Linux.

C'est un exécutable qui ne nécessite pas d'installation, il peut donc être téléchargé sur n'importe quel disque et lancé directement.

<http://www.putty.org/>⁵⁷

<http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>⁵⁸

57 - <http://www.putty.org/>

58 - <http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Bibliographie



[(Dupoirier, 1995)] GÉRARD DUPOIRIER, *Technologie de la GED : Techniques et management des documents électroniques*, Hermes, 1995.

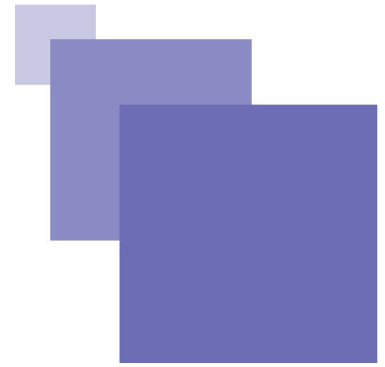
[André89] JACQUES ANDRÉ, RICHARD FURUTA, VINCENT QUINT, *Structured documents*, Cambridge University Press, 1989.

[Brillant07] ALEXANDRE BRILLANT, *XML : Cours et exercices*, Eyrolles, 2007 [ISBN 978-2212126914]

[Shea06] DAVE SHEA, MOLLY HOLZSCHLAG, *Le Zen des CSS*, Eyrolles, 2006.

[Thibaud03] THIBAUD CYRIL. *MySQL 4 : Installation, mise en oeuvre et programmation*. Editions ENI, 2003. Collection Ressources Informatiques.

Webographie



[w_w3c.org/XML] XML, <http://www.w3.org/XML> .

Index



<i>3-tier</i>	p.32, 34	<i>Form</i>	p.48	<i>Oracle</i>	p.68, 74
<i>Alternative</i>	p.47	<i>FORM</i>	p.59	<i>PHP</i> p.43, 44, 45, 46, 46, 47, 47, 48, 68,	
<i>Architecture</i>	p.31, 32, 34, 34, 68	<i>Formulaire</i>	p.48, 59	71, 73, 74, 74, 76	
<i>Balise</i>	p.	<i>Function</i>	p.71	<i>PostgreSQL</i>	p.74
<i>Boucle</i>	p.47	<i>HTML</i>	p.43, 44, 45, 46, 48, 59	<i>Serveur</i>	p.31, 32, 34, 34, 44, 46
<i>Classe</i>	p.73	<i>HTTP</i>	p.34	<i>SGML</i>	p.
<i>Client</i>	p.31, 32, 34	<i>IF</i>	p.47	<i>Variables</i>	p.47
<i>Echo</i>	p.45	<i>Méta-langage</i>	p.	<i>Web</i>	p.34, 34
<i>Fonction</i>	p.71	<i>MySQL</i>	p.76	<i>WHILE</i>	p.47
<i>FOR</i>	p.47	<i>Objet</i>	p.73	<i>XML</i>	p.
		<i>OCI</i>	p.74		