Applications Webs - bases de données

© Olivier Caron

Polytech'Lille

Plan

√ Notion d'architectures logicielles

Plan

- √ Notion d'architectures logicielles
- √ Le langage HTML - formulaires

Plan

- √ Notion d'architectures logicielles
- √ Le langage HTML - formulaires
- √ Introduction au langage PHP - principes

Plan

- √ Notion d'architectures logicielles
- √ Le langage HTML - formulaires
- √ Introduction au langage PHP - principes
- √ PHP et postgres

Plan

- √ Notion d'architectures logicielles
- √ Le langage HTML - formulaires
- √ Introduction au langage PHP - principes
- √ PHP et postgres
- √ Présentation du projet BD

Les architectures logicielles

√ Au programme de la 2ième année

Les architectures logicielles

- √ Au programme de la 2ième année
- √ Découpage d'une application client-serveur en plusieurs étages, Architectures n-tiers

Les architectures logicielles

- √ Au programme de la 2ième année
- √ Découpage d'une application client-serveur en plusieurs étages, Architectures n-tiers
- √ Au programme, architectures 2-tiers

Architectures 2-tiers

- √ les composants premier niveau : IHM
  - Fichiers html, formulaires html, serveur web. . .

Architectures 2-tiers

√ les composants premier niveau : IHM

- Fichiers html, formulaires html, serveur web. . .
- De nombreuses autres technologies : applets, asp, jsp, . . .

Architectures 2-tiers

√ les composants premier niveau : IHM

- Fichiers html, formulaires html, serveur web. . .
- De nombreuses autres technologies : applets, asp, jsp, . . .
- Gère l'affichage, envoi et réception des données

Architectures 2-tiers

√ les composants premier niveau : IHM

- Fichiers html, formulaires html, serveur web. . .
- De nombreuses autres technologies : applets, asp, jsp, . . .
- Gère l'affichage, envoi et réception des données
- Peut vérifier la cohérence des données à envoyer (ex :javascript)

Architectures 2-tiers

- √ les composants premier niveau : IHM
  - Fichiers html, formulaires html, serveur web. . .
  - De nombreuses autres technologies : applets, asp, jsp, . . .
  - Gère l'affichage, envoi et réception des données
  - Peut vérifier la cohérence des données à envoyer (ex :javascript)
- √ Les composants second niveau :
  - Technologie : langage de script PHP
  - De nombreuses autres technologies (CGI, perl, C, servlets, EJB, DCOM,. . .)

Architectures 2-tiers

√ les composants premier niveau : IHM
- Fichiers html, formulaires html, serveur web. . .
- De nombreuses autres technologies : applets, asp, jsp, . . .
- Gère l'affichage, envoi et réception des données
- Peut vérifier la cohérence des données à envoyer (ex :javascript)

√ Les composants second niveau :
- Technologie : langage de script PHP
- De nombreuses autres technologies (CGI, perl, C, servlets, EJB, DCOM,. . .)
- Gère le code métier (code applicatif)

Architectures 2-tiers

√ les composants premier niveau : IHM
- Fichiers html, formulaires html, serveur web. . .
- De nombreuses autres technologies : applets, asp, jsp, . . .
- Gère l'affichage, envoi et réception des données
- Peut vérifier la cohérence des données à envoyer (ex :javascript)

√ Les composants second niveau :
- Technologie : langage de script PHP
- De nombreuses autres technologies (CGI, perl, C, servlets, EJB, DCOM,. . .)
- Gère le code métier (code applicatif)
- Gère les données issues des SGBD

Le langage HTML

- √ Simplification de SGML
- √ CERN de Genève , les Normes (`http://www.w3c.org`)
- √ Langage non rigoureux (ex : paragraphe, logiciel tidy)
- √ Langage à base de balise :

```
<nomCommande attribut1=valeur1 ... attributN=valeurN>
</nomCommande>
```

- √ Des évolutions : DHTML, XHTML,. . .
- √ De *l'habillage* : flash, javascript,. . .

Les formulaires HTML

√ Balise `FORM`, syntaxe :

```
<form action=url_program method=type_methode option>
```

Les formulaires HTML

- √ Balise `FORM`, syntaxe :
  `<form action=url_program method=type_methode option>`
- √ url_program : localisation web du programme qui réceptionne les données du formulaires et exécute un traitement.

Les formulaires HTML

- Balise `FORM`, syntaxe :
  `<form action=url_program method=type_methode option>`
- url_program : localisation web du programme qui réceptionne les données du formulaires et exécute un traitement.
- type_methode : soit méthode `GET`, soit méthode `POST`

Les formulaires HTML

- √ Balise `FORM`, syntaxe :
  `<form action=url_program method=type_methode option>`
- √ url_program : localisation web du programme qui réceptionne les données du formulaires et exécute un traitement.
- √ type_methode : soit méthode `GET`, soit méthode `POST`
- √ option : `ENCRYPT`, valable uniquement pour la méthode `POST`, crypte les données.

Les données d'un formulaire

√ Ensemble de couples (nom, valeurs)

Les données d'un formulaire

- √ Ensemble de couples (nom, valeurs)
- √ Envoi sous forme texte (sauf si `ENCRYPT`), structure :
  `?nom1=valeur1&nom2=valeur2&...&nomn=valeurn`
- √ Avantages méthode `POST` :
  pas de limitation en taille des données envoyées
  chiffrement

Les données d'un formulaire

- √ Ensemble de couples (nom, valeurs)
- √ Envoi sous forme texte (sauf si `ENCRYPT`), structure :
  `?nom1=valeur1&nom2=valeur2&...&nomn=valeurn`
- √ Avantages méthode `POST` :
  pas de limitation en taille des données envoyées
  chiffrement
- √ Avantage méthode `GET` :
  Utilisable sans formulaire !

Structure d'un formulaire

√ Balise `input` : définition d'un composant graphique de saisie

Structure d'un formulaire

- √ Balise `input` : définition d'un composant graphique de saisie
- √ Deux champs requis : le champ `name` et le champ `value`

Structure d'un formulaire

- √ Balise `input` : définition d'un composant graphique de saisie
- √ Deux champs requis : le champ `name` et le champ `value`
- √ Possibilité de donner des valeurs par défaut (`value`)

Les composants graphiques (1/4)

- `text` : il s'agit d'un champ de saisie permettant la saisie d'une ligne de texte. La taille du champ peut être définie à l'aide de l'attribut `size` et la taille maximale du texte saisi grâce à l'attribut `maxlength`

Les composants graphiques (1/4)

- √ `text` : il s'agit d'un champ de saisie permettant la saisie d'une ligne de texte. La taille du champ peut être définie à l'aide de l'attribut `size` et la taille maximale du texte saisi grâce à l'attribut `maxlength`
- √ `password` : il s'agit d'un champ de saisie, dans lequel les caractères saisis apparaissent sous forme d'astérisques afin de camoufler la saisie de l'utilisateur.

Les composants graphiques (2/4)

- `checkbox` : il s'agit de cases à cocher pouvant admettre deux états : checked (coché) et unchecked (non coché). Lorsque la case est cochée, la paire nom/valeur est envoyée au programme.

Les composants graphiques (2/4)

- √ `checkbox` : il s'agit de cases à cocher pouvant admettre deux états : checked (coché) et unchecked (non coché). Lorsque la case est cochée, la paire nom/valeur est envoyée au programme.
- √ `radio` : il s'agit d'un bouton permettant un choix parmi plusieurs proposés (l'ensemble des boutons radios devant porter le même attribut `name`. La paire nom/valeur du bouton radio sélectionné sera envoyé au programme. Un attribut `checked` pour un des boutons permet de préciser le bouton sélectionné par défaut.

Les composants graphiques (3/4)

- `hidden` : il s'agit d'un champ caché. Utile pour y définir des paramètres à envoyer au programme

Les composants graphiques (3/4)

- √ `hidden` : il s'agit d'un champ caché. Utile pour y définir des paramètres à envoyer au programme
- √ `file` : il s'agit d'un champ permettant à l'utilisateur de préciser l'emplacement d'un fichier qui sera envoyé avec le formulaire. Il faut dans ce cas préciser le type de données pouvant être envoyées grâce à l'option `ACCEPT` de la balise `FORM`.

Les composants graphiques (4/4)

- `submit` : il s'agit du bouton de soumission permettant l'envoi du formulaire. Le texte du bouton peut être précisé grâce à l'attribut `value`.

Les composants graphiques (4/4)

- √ `submit` : il s'agit du bouton de soumission permettant l'envoi du formulaire. Le texte du bouton peut être précisé grâce à l'attribut `value`.
- √ `image` : il s'agit d'un bouton de soumission personnalisé, dont l'apparence est l'image situé à l'emplacement précisé par son attribut `src`.

Les composants graphiques (4/4)

- √ `submit` : il s'agit du bouton de soumission permettant l'envoi du formulaire. Le texte du bouton peut être précisé grâce à l'attribut `value`.
- √ `image` : il s'agit d'un bouton de soumission personnalisé, dont l'apparence est l'image situé à l'emplacement précisé par son attribut `src`.
- √ `reset` : il s'agit d'un bouton de remise à zéro permettant uniquement de rétablir l'ensemble des éléments du formulaire à leurs valeurs par défaut.

Les composants graphiques (4/4)

- √ `submit` : il s'agit du bouton de soumission permettant l'envoi du formulaire. Le texte du bouton peut être précisé grâce à l'attribut `value`.
- √ `image` : il s'agit d'un bouton de soumission personnalisé, dont l'apparence est l'image situé à l'emplacement précisé par son attribut `src`.
- √ `reset` : il s'agit d'un bouton de remise à zéro permettant uniquement de rétablir l'ensemble des éléments du formulaire à leurs valeurs par défaut.
- √ En savoir plus :`http://www.commentcamarche.net/html/form.php3`

La balise `TEXTAREA`

√ La balise `TEXTAREA` permet de définir une **zone** de saisie. Cette balise possède les attributs suivants :

- `cols` : représente le nombre de caractères que peut contenir une ligne.

La balise `TEXTAREA`

√ La balise `TEXTAREA` permet de définir une **zone** de saisie. Cette balise possède les attributs suivants :

- `cols` : représente le nombre de caractères que peut contenir une ligne.
- `rows` : représente le nombre de lignes.

La balise `TEXTAREA`

√ La balise `TEXTAREA` permet de définir une **zone** de saisie. Cette balise possède les attributs suivants :

- `cols` : représente le nombre de caractères que peut contenir une ligne.
- `rows` : représente le nombre de lignes.
- `readonly` : permet d'empêcher l'utilisateur de modifier le texte entré par défaut dans le champ.

La balise `TEXTAREA`

√ La balise `TEXTAREA` permet de définir une **zone** de saisie. Cette balise possède les attributs suivants :

- `cols` : représente le nombre de caractères que peut contenir une ligne.
- `rows` : représente le nombre de lignes.
- `readonly` : permet d'empêcher l'utilisateur de modifier le texte entré par défaut dans le champ.
- Les champs `name` et `value`

La balise `SELECT` (1/2)

√ La balise `SELECT` permet de créer une liste déroulante d'éléments (précisés par des balises `OPTION` à l'intérieur de celle-ci). Les attributs de cette balise sont :

- Le champ `name`.

La balise `SELECT` (1/2)

√ La balise `SELECT` permet de créer une liste déroulante d'éléments (précisés par des balises `OPTION` à l'intérieur de celle-ci). Les attributs de cette balise sont :

- Le champ `name`.
- Le champ `disabled` : permet de créer une liste désactivée, c'est-à-dire affichée en grisée.

La balise `SELECT` (1/2)

√ La balise `SELECT` permet de créer une liste déroulante d'éléments (précisés par des balises `OPTION` à l'intérieur de celle-ci). Les attributs de cette balise sont :

- Le champ `name`.
- Le champ `disabled` : permet de créer une liste désactivée, c'est-à-dire affichée en grisée.
- Le champ `size` : représente le nombre de lignes dans la liste (cette valeur peut être plus grande que le nombre d'éléments effectifs dans la liste)
- Le champ `multiple` : marque la possibilité pour l'utilisateur de choisir plusieurs champs dans la liste.

La balise SELECT (1/2)

√ Un exemple :

```
Quel est votre système : <SELECT name="os">
                          <OPTION NAME="linux">Linux</OPTION>
                          <OPTION NAME="win">Windows</OPTION>
                          <OPTION NAME="sun">Solaris</OPTION>
                          <OPTION NAME="mac">Mac OS</OPTION>
                         </SELECT>
```

Commentaires :

√ Interface de saisie assez simple.

Commentaires :

- √ Interface de saisie assez simple.
- √ Graphismes corrects

Commentaires :

- √ Interface de saisie assez simple.
- √ Graphismes corrects
- √ Pas possible de tout faire, (ex : browser multiples)

Commentaires :

- √ Interface de saisie assez simple.
- √ Graphismes corrects
- √ Pas possible de tout faire, (ex : browser multiples)
- √ Possible de générer des formulaires !
  Ex : les options d'une balise `select`, résultat d'une requête BD.

La balise `TABLE` (1/2)

√ Idéal pour afficher des données structurées

La balise `TABLE` (1/2)

- √ Idéal pour afficher des données structurées
- √ Les sous-balises de la balise `table` sont :
  - `caption` : spécifie un nom de tableau.

La balise `TABLE` (1/2)

- √ Idéal pour afficher des données structurées
- √ Les sous-balises de la balise `table` sont :
  - `caption` : spécifie un nom de tableau.
  - `tr` : (table row) : spécifier une ligne du tableau.

La balise `TABLE` (1/2)

- √ Idéal pour afficher des données structurées
- √ Les sous-balises de la balise `table` sont :
  - `caption` : spécifie un nom de tableau.
  - `tr` : (table row) : spécifier une ligne du tableau.
  - `th` : (table heading) : spécifie un titre de colonne

La balise `TABLE` (1/2)

- √ Idéal pour afficher des données structurées
- √ Les sous-balises de la balise `table` sont :
  - `caption` : spécifie un nom de tableau.
  - `tr` : (table row) : spécifier une ligne du tableau.
  - `th` : (table heading) : spécifie un titre de colonne
  - `td` : (table data) : spécifie une valeur du tableau

La balise TABLE - un exemple (2/2)

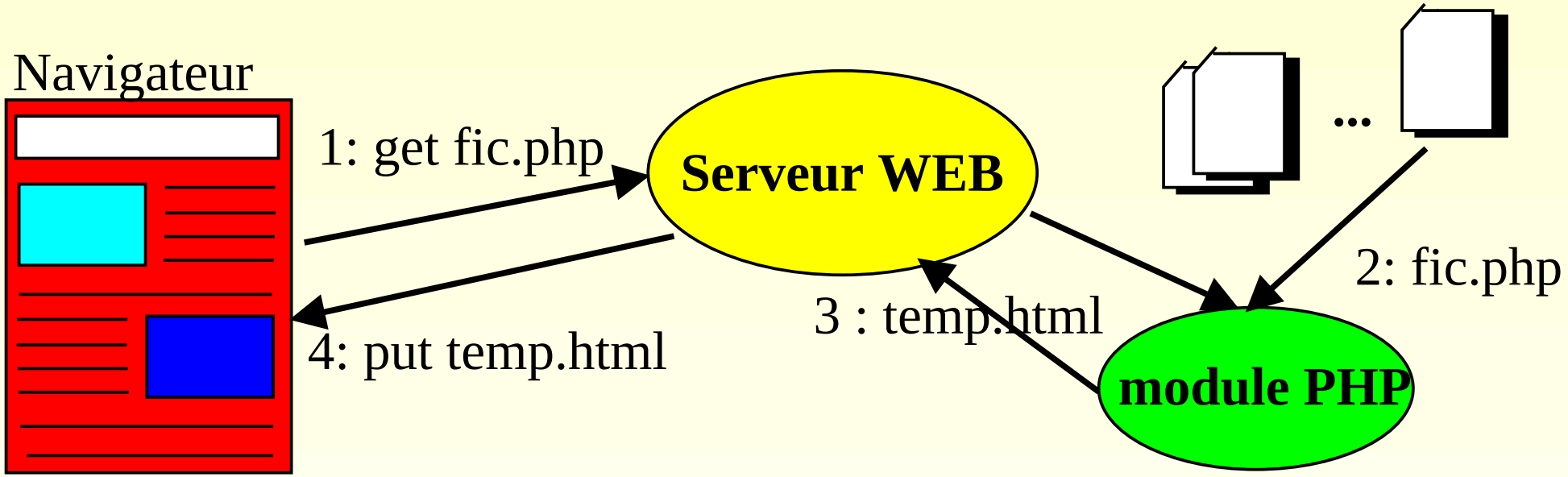
```
<table border=1>
  <caption>Plateformes reconnus</caption>
  <tr> <th> produits </th> <th> plateforme </th> </tr>
  <tr> <td> Java  </td> <td> linux, windows, mac os, ...</td> </tr>
  <tr>  <td> Word  </td  <td> mac os, windows </td> </tr>
</table>
```

donne le tableau suivant :

Plateformes reconnus

produits	**plateforme**
Java	linux, windows, mac os, . . .
Word	mac os, windows

Architecture web - PHP



Le langage PHP

√ Langage de script dédié au Web

Le langage PHP

√ Langage de script dédié au Web

√ S'insère dans du code html (balise `<?` ou `<?php`).

Le langage PHP

- √ Langage de script dédié au Web
- √ S'insère dans du code html (balise `<?` ou `<?php`).
- √ Pas de contrôle de type (du script, quoi. . .)

Le langage PHP

- √ Langage de script dédié au Web
- √ S'insère dans du code html (balise `<?` ou `<?php`).
- √ Pas de contrôle de type (du script, quoi. . .)
- √ Des structures de contrôle (if, while, for)

Le langage PHP

- √ Langage de script dédié au Web
- √ S'insère dans du code html (balise `<?` ou `<?php`).
- √ Pas de contrôle de type (du script, quoi. . .)
- √ Des structures de contrôle (if, while, for)
- √ Similaire au C

Le langage PHP

- √ Langage de script dédié au Web
- √ S'insère dans du code html (balise `<?` ou `<?php`).
- √ Pas de contrôle de type (du script, quoi. . .)
- √ Des structures de contrôle (if, while, for)
- √ Similaire au C
- √ De nombreuses fonctionnalités (évolue encore)

Un exemple (1/2)

Soit le fichier html-php suivant (nom du fichier : `test.php`) :

```
<html> <head> <title> Premier programme PHP</title> </head>
<body bgcolor=white>
<?php
  echo "  <h1> code php en action ! </h1>\n" ;
  $i=0 ;
  while ($i<5) {
    echo "    <h2> hello world ! </h2>\n" ;
    $i=$i+1 ;
  }
?>  <h1>Ce n'est plus du php !</H1>
</body></html>
```

Un exemple (2/2)

```
<html>
<head> <title> Premier programme PHP </title> </head>
<body bgcolor=white>
  <h1> code php en action ! </h1>
    <h2> hello world ! </h2>
    <h2> hello world ! </h2>
    <h2> hello world ! </h2>
    <h2> hello world ! </h2>
    <h2> hello world ! </h2>
    <h1>Ce n'est plus du php !</H1>
</body></html>
```

Le langage PHP - les bases

√ Utilisation de variables : $nom_variable

Le langage PHP - les bases

- √ Utilisation de variables : $nom_variable
- √ C'est l'affectation qui définit le type :

```
$i = 3 ; // i entier
$car = 'a' ; // caractère
$str = "select * from " ; // str : chaine de caracteres
$str = $str . "personne" ; // concatenation
$tab[0]= 3 ; // tab tableau à une dimension d'entiers
$var = (string) $i ;// var : chaine et vaut "3"
```

Récupération des données du formulaire

- √ Chaque donnée du formulaire est référencé par un nom issu de l'attribut `name`

  `extract($_POST) ;` ou `extract($_GET) ;`

Récupération des données du formulaire

- √ Chaque donnée du formulaire est référencé par un nom issu de l'attribut `name`
  `extract($_POST) ;` ou `extract($_GET) ;`
- √ PHP génère une variable du nom de la valeur de cet attribut

Récupération des données du formulaire

- √ Chaque donnée du formulaire est référencé par un nom issu de l'attribut `name`
  `extract($_POST) ;` ou `extract($_GET) ;`
- √ PHP génère une variable du nom de la valeur de cet attribut
- √ Possible par ce moyen de faire transiter des variables entre fichiers php :
  Exemple : `<a href=fic2.php?j=3>`

PHP et Postgres (1/3)

√ Connexion à la base :

```
$connect = pg_connect("host=weppes.studserv.deule.net  user=ocaron
                       password=postgres dbname=occhansons");
if (!$connect) {
  echo "Erreur connexion\n" ;
  exit ;
}
```

PHP et Postgres (1/3)

- Connexion à la base :

```
$connect = pg_connect("host=weppes.studserv.deule.net  user=ocaron
                       password=postgres dbname=occhansons");
if (!$connect) {
  echo "Erreur connexion\n" ;
  exit ;
}
```

- Déconnexion :

```
pg_close($connect) ;
```

- Préfixer les fonctions par @ pour éviter les messages d'erreurs

PHP et Postgres (2/3)

√ Executer une requête SQL :

```
$cmdeSQL = "select * from nom" ;
$cmdeSQL .= "order by n_nom" ;

$result=pg_exec($connect, $cmdeSQL) ;
if (!$result) {
  echo "Erreur SQL\n" ;
  exit ;
}
```

PHP et Postgres (3/3)

√ Exploiter le résultat d'une requête SQL :

```
// compter le nombre de lignes
$numLignes=pg_numrows($result);
$ligne=0 ;
while ($ligne<$numLignes) {
  echo pg_result($result,$ligne, "n_nom")
  echo <BR> ;
  $ligne++ ;
}
```

√ Les indices commencent à partir de zéro

PHP et Postgres (3/3)

- Exploiter le résultat d'une requête SQL :

```
// compter le nombre de lignes
$numLignes=pg_numrows($result);
$ligne=0 ;
while ($ligne<$numLignes) {
  echo pg_result($result,$ligne, "n_nom")
  echo <BR> ;
  $ligne++ ;
}
```

- Les indices commencent à partir de zéro
- `table pg_fetch_row($result,$ligne)` donne les valeurs des colonnes d'une ligne

Structuration du projet tutoré

- √ Durée du projet : 25h
- √ Thèmes du projet :
  - Base de données
    (analyse, conception, implantation modèle relationnel)
  - Introduction réseaux
    Serveur Web, PHP
- √ Réalisation du projet par binôme

Calendrier du projet

- √ Les deux premières semaines - 4 séances
  Analyse du problème et conception
  Le tuteur joue le rôle du **client** du futur système d'information et non celui de l'expert.
- √ Premier rapport : Un rapport d'analyse et de conception doit être délivré (analyse, schéma conceptuel UML, schéma relationnel).
- √ Phase de développement - 8 séances
  Développement HTML-PHP-Postgres
- √ Second rapport : Un rapport du projet doit être délivré :
  Adresse du site, description des fonctions du site, architecture globale de vos pages webs, principaux scripts php. . .

Spécificités des projets tutorés

- √ Sujets plus ambitieux que des sujets de T.P.
- √ Développer l'autonomie (et les initiatives ?)
- √ Auto-apprentissage :
  - `http://www.phpbuilder.com/`
  - `http://www.php.net/`
  - `http://www.w3c.org/`

Outils utilisés

- √ Serveur Web Apache installé sur `weppes.studserv.deule.net` :
  le fichier `~nomLogin/public_html/fic.html` est accessible à :
  `http://weppes.studserv.deule.net/~nomLogin/fic.html`
- √ Module Apache-PHP (vers. 4.0)
- √ Serveur Postgres (vers. 7.4.5) sur `weppes`

Sécurité de vos données

- √ La *sécurité 0* n'existe pas !
- √ Rendre accessible le répertoire web :

```
chmod 701 ~
chmod 701 ~/public_html
```

- √ Protéger vos fichiers :

```
cd ~/public_html
chmod -R 604 *
```

Remarques

- √ C'est avant tout un projet Base de données !
- √ Ne pas perdre son temps dans le graphisme !
- √ Privilégier l'aspect fonctionnel