Applications webs - Bases de données

Olivier Caron

Polytech Lille
Avenue Paul Langevin Cité Scientifique Lille 1
59655 Villeneuve d'Ascq cedex

http://ocaron.plil.fr
Olivier.Caron@polytech-lille.fr

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

Pourquoi la programmation ?

- Le langage SQL n'est pas un langage de programmation

Pourquoi la programmation ?

- Le langage SQL n'est pas un langage de programmation
- Ne résout pas toutes les requêtes (ex: clôture transitive)

Pourquoi la programmation ?

- Le langage SQL n'est pas un langage de programmation
- Ne résout pas toutes les requêtes (ex: clôture transitive)
- Permet de tester et d'expliquer toutes les contraintes avant d'effectuer une simple action sur la base.
  Exemple : emprunt d'un livre dans une bibliothèque

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 de ce module, architectures 2-tiers

Architectures 2-tiers

- les composants premier niveau : IHM

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 :

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

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 à base de balises :

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

<nomCommande />
```

- Des évolutions : de HTML 1 à HTML 5
- Les navigateurs sont plus permissifs que la norme
- `http://validator.w3.org/` (bookmark :-)

Premier fichier HTML 5

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"> <title>Essai</title>
  </head>
  <body>
    <h1>Hello les GIS 2A 3 </h1>
    jusqu'ici tout va <b>bien</b>
  </body>
</html>
```

- Seule la première ligne indique que c'est un fichier HTML 5

Déploiement

- Utilisation du serveur web installé sur la machine : `houplin.studserv.deule.net`
- Scrute les pages webs des utilisateurs Unix situés dans l'espace `public_html` (si les droits unix le permettent)

Configuration Polytech

Soit `fic.html` stocké dans le répertoire `public_html`
disponible à l'adresse `http:`
`//houplin.studserv.deule.net/~login/fic.html`
Donnez les droits UNIX au serveur web:
chmod o+x ~ ; chmod o+rx ~/public_html

Balises usuelles

- Les titres sont déclarés par la balise ’h’ (heading) avec un numéro de 1 à 6 : <h1> : titre de niveau 1, <h2> : titre de niveau 2,. . .

Balises usuelles

- Les titres sont déclarés par la balise ’h’ (heading) avec un numéro de 1 à 6 : <h1> : titre de niveau 1, <h2> : titre de niveau 2,. . .
- ’b’ (bold) : texte en gras, ’i’ (pour italic) : texte en italique

Balises usuelles

- Les titres sont déclarés par la balise ’h’ (heading) avec un numéro de 1 à 6 : <h1> : titre de niveau 1, <h2> : titre de niveau 2,. . .
- ’b’ (bold) : texte en gras, ’i’ (pour italic) : texte en italique
- ’ul’ (unumbered list), ’ol’ (ordered list) à utiliser avec ’li’ (list item)

Balises usuelles

- Les titres sont déclarés par la balise ’h’ (heading) avec un numéro de 1 à 6 : <h1> : titre de niveau 1, <h2> : titre de niveau 2,. . .
- ’b’ (bold) : texte en gras, ’i’ (pour italic) : texte en italique
- ’ul’ (unumbered list), ’ol’ (ordered list) à utiliser avec ’li’ (list item)
- Evolution norme HTML, tags obsolètes (deprecated) (ex: b et i).

La balise qui fait tout (1/2)

- La balise ’a’ (pour anchor) permet d’établir des hyperliens.

La balise qui fait tout (1/2)

- La balise ’a’ (pour anchor) permet d’établir des hyperliens.
- Un attribut `href` pour désigner l’adresse web cible.

La balise qui fait tout (1/2)

- La balise ’a’ (pour anchor) permet d’établir des hyperliens.
- Un attribut `href` pour désigner l’adresse web cible.
- `href` désigne une URL (Uniform Ressource Locator)

La balise qui fait tout (1/2)

- La balise 'a' (pour anchor) permet d'établir des hyperliens.
- Un attribut `href` pour désigner l'adresse web cible.
- `href` désigne une URL (Uniform Ressource Locator)
- Syntaxe simplifiée : `protocole://adresseIPMachine/chemin/nomFichier?parametres`

La balise qui fait tout (1/2)

- La balise 'a' (pour anchor) permet d'établir des hyperliens.
- Un attribut `href` pour désigner l'adresse web cible.
- `href` désigne une URL (Uniform Ressource Locator)
- Syntaxe simplifiée : `protocole://adresseIPMachine/chemin/nomFichier?parametres`
- Quelques exemples :

```
ftp://ftp.polytech-lille.fr
http://www.google.fr
http://www.polytech-lille.fr/genie-informatique-et-statistique-p117.html
http://www.bing.com/search?q=polytech
```

La balise qui fait tout (2/2)

- Possibilité de faire des liens relatifs (par rapport à l'adresse de la page qui contient l'hyperlien)

La balise qui fait tout (2/2)

- Possibilité de faire des liens relatifs (par rapport à l'adresse de la page qui contient l'hyperlien)
- Quelques exemples :
  `fic2.html` équivalent à `./fic2.html`
  `../rep1/fic3.html`

La balise qui fait tout (2/2)

- Possibilité de faire des liens relatifs (par rapport à l'adresse de la page qui contient l'hyperlien)
- Quelques exemples :
  `fic2.html` équivalent à `./fic2.html`
  `../rep1/fic3.html`
- Possibilité de faire des liens vers une partie précise d'une page grâce à '#': `http://ocaron.plil.fr/enseignement/index.html#GIS2A3`

Autres balises

- : permet de regrouper des éléments d'une page, permet de l'identifier grâce à l'attribut `id`

Autres balises

- : permet de regrouper des éléments d'une page, permet de l'identifier grâce à l'attribut `id`
- similaire pour une partie de texte

Autres balises

- : permet de regrouper des éléments d'une page, permet de l'identifier grâce à l'attribut `id`
- similaire pour une partie de texte
- <table> : Affichage de tableaux

Autres balises

- : permet de regrouper des éléments d'une page, permet de l'identifier grâce à l'attribut `id`
- similaire pour une partie de texte
- <table> : Affichage de tableaux
- et bien d'autres encore

La balise `table` (2/3)

- Idéale pour afficher des données structurées

La balise `table` (2/3)

- Idéale pour afficher des données structurées
- Les sous-balises de la balise `table` sont :

La balise `table` (2/3)

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

La balise `table` (2/3)

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

La balise `table` (2/3)

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

La balise `table` (2/3)

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

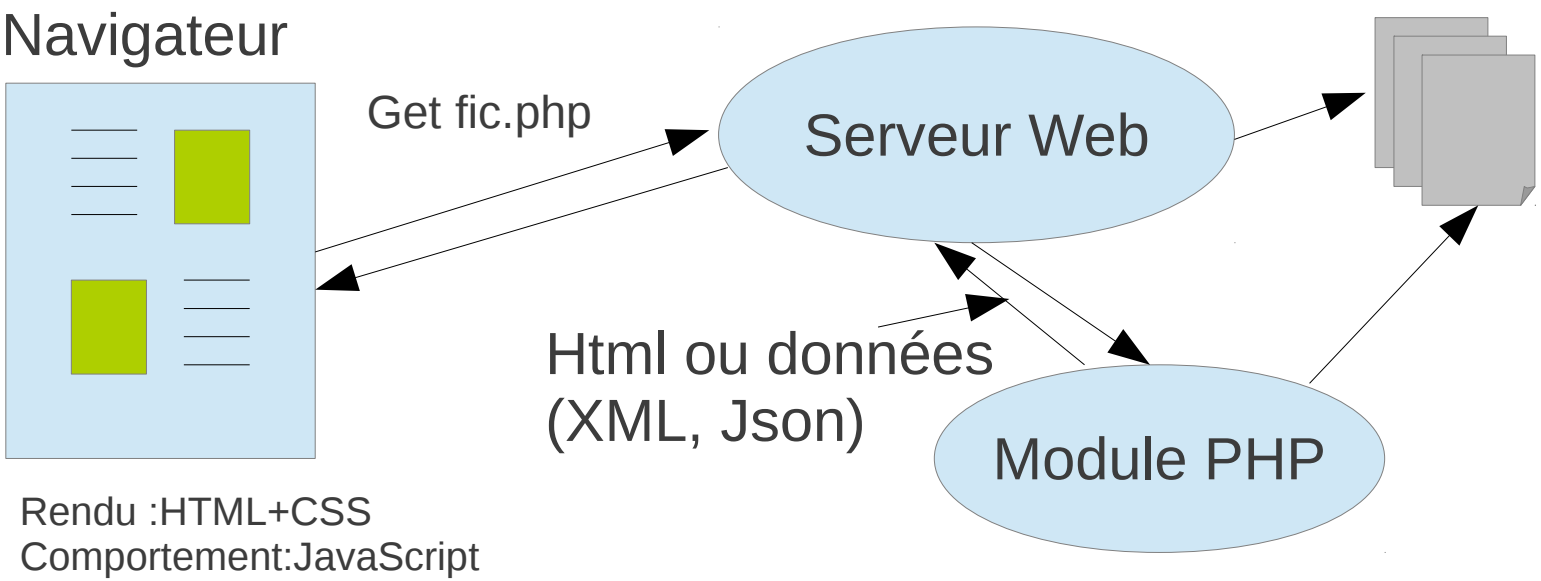
La balise `table` (2/3)

- Idéale pour afficher des données structurées
- Les sous-balises de la balise `table` sont :
  - `caption` : spécifie un nom de tableau.
  - `tr` : (table row) : spécifie une ligne du tableau.
  - `th` : (table heading) : spécifie un titre de colonne
  - `td` : (table data) : spécifie une valeur du tableau
  - `tbody`, `thead`, `tfoot`, désigne les différentes parties du tableau

La balise `table` - un exemple (3/3)

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

Architecture web, technologies associées



Envoi de données

- Envoi de données du navigateur au serveur web
- Ensemble de couples (nom,valeur)
- Données associées à l'appel d'un programme web distant
- Structure :`?nom1=valeur1&nom2=valeur2&...`
- Avantages méthode `post` :
  pas de limitation en taille des données envoyées, chiffrement (faible)
- Avantage méthode `get` :
  Utilisable sans formulaire !
- Deux types d'envoi de données :
  - méthode `get`: les données sont concaténées à l'URL (visibles), taille des données limitées à la taille max pour une URL
  - méthode `post`: les données sont envoyées à part, possibilité de crypter les données (niveau de sécurité faible),pas de limitation de taille

Les formulaires HTML

- Balise `form`, syntaxe :
  `<form action=url_program method=type_methode option>`
- url_program : adresse web du programme qui réceptionne les données du formulaire et exécute un traitement.
- type_methode : soit méthode `get`, soit méthode `post`
- option : `ENCRYPT`, valable uniquement pour la méthode `post`, crypte les données (niveau de sécurité faible).
- Remarque : on peut se passer de formulaire grâce à la méthode `get`

Structure générale d'un formulaire

```
<form action="traiter.php" method="get">
  code html
    <input type ="..." name="..."   value="..." ...>
  code html
    <input type ="..." name="..."   value="..." ...>
  <input type="submit" ...>
</form>
```

- Balise `input` : définition d'un composant graphique de saisie de données
- Deux champs requis : le champ `type` et le champ `name` .
- Le champ `value` parfois requis selon la valeur de `type`.
- Possibilité de donner une valeur par défaut (`value`)

Quelques attributs du composant `input`

required : vérification automatique que le champ est saisi avant envoi des données.

placeholder : Permet d'indiquer ce que l'on veut comme donnée...

pattern : expression régulière des données saisies qui sera vérifiée avant envoi.

```
nom :
<input type ="text" name="nom"  placeholder="votre nom...">
<br />code pays :
<input type="text" name="codePays" pattern="[A-Za-z]{2}" required>
<!-- code pays sur deux lettres (ex: FR, GB, de, ...) -->
```

Les types du composant `input` (1/3)

- `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.
- `date` : permet de sélectionner une date à partir d'un calendrier (norme HTML 5)
- d'autres types (HTML 5) : datetime, datetime-local, month , week, time, number, range, email, url, search, color,...

Les types du composant `input` (2/3)

- `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 ont la même valeur pour l'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 types du composant `input` (3/3)

- `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'attribut `accept` de la balise `form`.

Envoi des données du formulaire

- `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.openclassrooms.com`, `http://www.w3schools.com/html/html_forms.asp`, `https://www.w3.org/TR/html51/sec-forms.html`

Les ajouts HTML 5

- De nouveaux types d'entrée :
  datetime, datetime-local, date, month , week, time, number, range, email, url, search, color
- HTML 5, c'est aussi : la géolocalisation, le drag and drop
- Le composant `canvas`
- Les composants `audio` et `video`

La balise `textarea`

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

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 :

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)

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` (2/2)

- Un exemple :

```
Quel est votre systeme :
<select name="os">
  <option value="linux">Linux</option>
  <option value="win">Windows</option>
  <option value="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)
- Possibilité de générer des formulaires !
  Ex: les options d'une balise `select`, résultat d'une requête BD.

Le langage PHP

- Langage de script dédié au Web

Le langage PHP

- Langage de script dédié au Web
- S'insère dans du code html (balises `<?php ... ?>`).

Le langage PHP

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

Le langage PHP

- Langage de script dédié au Web
- S'insère dans du code html (balises `<?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 (balises `<?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 (balises `<?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`) :

```
<!DOCTYPE html>
<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>
```

PHP : les variables

- Utilisation de variables : $nomVariable
- C'est l'initialisation qui définit le type :

```
$i = 3 ; // integer
$trouve=true ;// booleen
$car = 'a' ; // char  quote (') ou guillemet (") : meme combat
$str = "select * from " ; // str : string
$str = $str . "personne" ; // string concatenation de chaines
$tab[0]= 3 ; // array of integer
$var = (string) $i ;// var : string , value "3"
```

- Visibilité de la variable: dans le bloc où la variable est utilisée la première fois
- Visibilité des variables globales : dans tout le fichier HTML (mot-clé `global`)

Les opérateurs de base

- opérateurs arithmétiques : $+$ (addition), $-$ (soustraction), * (multiplication), / (division), % (modulo)

Les opérateurs de base

- opérateurs arithmétiques : $+$ (addition), $-$ (soustraction), * (multiplication), / (division), % (modulo)
- = (affectation), l'affectation est un opérateur

Les opérateurs de base

- opérateurs arithmétiques : $+$ (addition), $-$ (soustraction), * (multiplication), / (division), % (modulo)
- = (affectation), l'affectation est un opérateur
- $++$ (incrémentation),$--$ (décrémentation)

Les opérateurs de base

- opérateurs arithmétiques : + (addition), − (soustraction), * (multiplication), / (division), % (modulo)
- = (affectation), l'affectation est un opérateur
- ++ (incrémentation),−− (décrémentation)
- opérateurs combinés (+=, -=, . . .)
  ex : $a += 3 signifie qu'on ajoute à la variable $a la valeur 3

Les opérateurs de base

- opérateurs arithmétiques : + (addition), − (soustraction), * (multiplication), / (division), % (modulo)
- = (affectation), l'affectation est un opérateur
- ++ (incrémentation),−− (décrémentation)
- opérateurs combinés (+=, -=, . . .)
  ex : $a += 3 signifie qu'on ajoute à la variable $a la valeur 3
- La concaténation de chaînes, opérateur '.',
  ex : $ch = "Hello "." World !" ;

Les opérateurs de base

- opérateurs arithmétiques : + (addition), − (soustraction), * (multiplication), / (division), % (modulo)
- = (affectation), l'affectation est un opérateur
- ++ (incrémentation),−− (décrémentation)
- opérateurs combinés (+=, -=, . . .)
  ex : $a += 3 signifie qu'on ajoute à la variable $a la valeur 3
- La concaténation de chaînes, opérateur '.',
  ex : $ch = "Hello "." World !" ;
- Une autre manière de concaténer des chaînes :
  ex : $ch=" World !" ; $ch2="Hello $ch" ;

Les opérateurs de comparaison

- == signifie égalité, renvoie true si égal

Les opérateurs de comparaison

- == signifie égalité, renvoie true si égal
- != signifie différent

Les opérateurs de comparaison

- == signifie égalité, renvoie true si égal
- != signifie différent
- opérateurs classiques : <, >, <=, >=

Les opérateurs de comparaison

- == signifie égalité, renvoie true si égal
- != signifie différent
- opérateurs classiques : <, >, <=, >=
- ===, signifie égal à, en type et en valeur
  ex : $a=3 ; $b="3", $a === $b (retourne faux), $a==$b (retourne vrai)

Les opérateurs de comparaison

- == signifie égalité, renvoie true si égal
- != signifie différent
- opérateurs classiques : <, >, <=, >=
- ===, signifie égal à, en type et en valeur
  ex : $a=3 ; $b="3", $a === $b (retourne faux), $a==$b (retourne vrai)
- !==, signifie différent en valeur ou différent en type

Les opérateurs logiques

- Opérateur ! représente la négation

Les opérateurs logiques

- Opérateur ! représente la négation
- Opérateurs && ou AND représentent le et logique

Les opérateurs logiques

- Opérateur ! représente la négation
- Opérateurs && ou AND représentent le et logique
- || ou OR représentent le ou logique

Les opérateurs logiques

- Opérateur ! représente la négation
- Opérateurs && ou AND représentent le et logique
- || ou OR représentent le ou logique
- XOR représente le ou exclusif logique

Structure de contrôle: if

- Egal à 0 ou false implique faux, différent de zéro ou true implique vrai
- Partie `else` facultative
- Définitions de blocs '{' et '}' si nécessaire
- Il existe aussi la clause `elseif` pour enchaîner les tests
- Attention au test d'égalité dans expression

```
<?php
   if ($v == "ensoleille")
      print("il fait beau") ;
   else {
      print("il ne fait pas beau") ;
      print("c'est le nord...") ;
   }
?>
```

Structure de contrôle: while

- Continue tant que l'expression est vraie

```
<?php
$i = 1 ;
while ($i <=10) {
   print($i."␣") ;
   $i++ ;
}
?>
```

Structure de contrôle: for

- 3 parties séparées par ";"
  - Partie 1 : initialisation variable
  - Partie 2 : test pour repasser dans la boucle
  - Partie 3 : expression effectuée à la fin de la boucle avant test de la partie 2
- L'instruction `for` est équivalente à une instruction `while`

```
<?php
print("<h1>affichage des nombres pairs </h1>"\n") ;
for ($pair=2; $pair <= 100; $pair=$pair+2)
  print($pair."<br />\n") ;
?>
```

Sections de code PHP

- Plusieurs sections de code PHP possibles dans un fichier HTML
- Possibilité d'un fichier pur PHP : bibliothèque de fonctions
- Obligation : tout fichier contenant au moins une section PHP doit avoir le suffixe "php"
- Le code le plus **utile** lors du développement PHP :

```
<?php
    error_reporting(E_ALL);
    ini_set('display_errors','On');
?>
```

Les fonctions PHP

- Syntaxe :

```
function nomFonction(argument1, argument2,
...)  {
  liste d'instructions
}
```

Les fonctions PHP

- Syntaxe :

```
function nomFonction(argument1, argument2,
...)  {
  liste d'instructions
}
```

- Pas de distinction fonctions, procédures. Instruction `return`.

Les fonctions PHP

- Syntaxe :

```
function nomFonction(argument1, argument2,
...)  {
  liste d'instructions
}
```

- Pas de distinction fonctions, procédures. Instruction `return`.
- Bibliothèques de fonctions :

```
require('rep1/rep2/bib.php') ;
```

Les fonctions PHP

- Syntaxe :

```
function nomFonction(argument1, argument2,
...)  {
  liste d'instructions
}
```

- Pas de distinction fonctions, procédures. Instruction `return`.
- Bibliothèques de fonctions :

```
require('rep1/rep2/bib.php') ;
```

- valeurs par défauts :

```
function f($a, $b = "ok")
```

Les tableaux en PHP (1/2)

- Initialisation d'un tableau :

```
$tableauVide=array() ;
$tableauEntiers=array(2,27,35,45) ;
$tableauCouleurs=array("bleu", "blanc", "rouge") ;
```

- Nbre d'éléments d'un tableau : fonction `count($tab)`

```
print("nbre d'elements :".count($tableauCouleurs)."<br/>\n") ;
// affiche 3
```

- Parcours d'un tableau (indice de 0 à taille-1) :

```
for ($i=0; $i<count($tableauEntiers);$i++)
  print("indice : $i vaut ".$tableauEntiers[$i]."<br/>\n") ;
```

Les tableaux en PHP (2/2)

- Saisie d'un tableau, indice explicite :

```
$tableauVide=array() ;
$tableau[0]="bonjour", $tableau[1]="ola" ;
```

- Saisie d'un tableau, auto-indentation :

```
$tableauVide=array() ;
$tableau[]="bonjour", $tableau[]="ola" ;
```

- Même résultat pour ces deux modes de saisie

Les tableaux associatifs en PHP

- Utilisation de noms pour indicer les valeurs :

```
$couleurs=array(
  "rouge" => "ff0000",
  "vert" => "00ff00",
  "bleu" => "0000ff"
) ;
```

- Accès à un élément d'un tableau associatif

```
print("code de bleu".$couleurs["bleu"]."<br />\n") ;
// affiche 0000ff
```

Parcours de tableaux

- Itération de tableaux : instruction `foreach`
- Fonctionne pour tableaux associatifs ou non.
- Deux syntaxes :

```
foreach($couleurs as $value)
  print("code : $value</br>\n")   ;
// ou bien
foreach($couleurs as $index => $value)
  print("code de $index : $value</br>")  ;
```

Les tableaux multi-dimensionnels

- On peut créer des tableaux à x dimensions

Les tableaux multi-dimensionnels

- On peut créer des tableaux à x dimensions
- Exemple : $tab[3][4] ;

Les tableaux multi-dimensionnels

- On peut créer des tableaux à x dimensions
- Exemple : $tab[3][4] ;
- On peut combiner des tableaux indicés numériquement avec des tableaux associatifs

Récupération de données issues d'un formulaire web

- Les données envoyées du formulaire sont stockées dans un tableau associatif `$_POST` ou `$_GET`
  L'indice du tableau correspond au nom de la variable
- Tester l'existence d'une donnée :
  ```
  if (isset($_POST["nom"])) ...
  ```
- `extract($_POST) ;` ou `extract($_GET) ;`
  `extract($tab)` : pour tous les éléments du tableau associatif `$tab`, génération d'une variable du nom de l'indice du tableau.
- Un moyen pour faire transiter des variables entre fichiers php :
  Exemple: `<a href="fic2.php?j=3&nom=dupont">`

Conclusion sur PHP

- Deux bonnes adresses :

Conclusion sur PHP

- Deux bonnes adresses :
- Ce cours PHP :`http://ocaron.plil.fr`

Conclusion sur PHP

- Deux bonnes adresses :
- Ce cours PHP :`http://ocaron.plil.fr`
- Le manuel PHP : `http://www.php.net/manual/fr/`

PHP et les bases de données

- Des bibliothèques (API) qui permettent de gérer des bases de données
  Un ensemble de fonctions pour chaque SGBD

PHP et les bases de données

- Des bibliothèques (API) qui permettent de gérer des bases de données
  Un ensemble de fonctions pour chaque SGBD
- Alternative : PDO (PHP Data Object):une API unique quelque soit le SGBD

PHP Postgres : connexion à une base de données

- Connexion :

```
<?php
// Connexion, sélection de la base de données
$db = pg_connect("host=localhost dbname=nomBase "
                 "user=www password=foo")
    or die('Connexion impossible : ' . pg_last_error());
?>
```

- Fonctions `pg_connect` et `pg_last_error`
- La fonction `die` permet de quitter l'exécution du programme.

PHP Postgres : Déconnexion à une base de données

- Déconnexion :

```
<?php
...
pg_close($db);
?>
```

PHP Postgres : instructions DDL

```
<?php
    $db = ...
    $requeteSQL="update employe set salaire=salaire*0.10" ;
    $result=pg_query($db,$requeteSQL)
            or die('Echec de la requête : ' . pg_last_error());
    echo "Mise à jour de ".pg_affected_rows($result).
        " employé(s)<br />" ;
?>
```

- Fonction `pg_affected_rows`, retourne le nombre de lignes concernées par l'opération de modification de la base.

PHP Postgres : consultation de la base (1/4)

```
<?php
$query = 'SELECT id, auteur FROM author';
$result = pg_query($db, $query)
          or die('Echec de la requête : ' . pg_last_error());
?>
```

- Méthode `pg_query` retourne une variable contenant les données résultat de la requête.

PHP Postgres : consultation de base (2/4)

- 1ière version: récupération des données ligne par ligne via la fonction `pg_fetch_assoc`

```
<?php
while ($row = pg_fetch_assoc($result)) {
  echo "id : ".$row['id']." " ;
  echo "auteur : ".$row['auteur']."<br />";
}
?>
```

- Fonction `pg_fetch_assoc` retourne un tableau associatif de la ligne en cours.

PHP Postgres : consultation de base (3/4)

- 2nde version: récupération des données ligne par ligne via la fonction `pg_fetch_row`

```
<?php
   while ($row = pg_fetch_row($result)) {
      echo "id: ".$row[0]." " ;
      echo "auteur: ".$row[1]."<br />";
   }
?>
```

- Fonction `pg_fetch_row` retourne un tableau de la ligne en cours, indice à partir de 0.

PHP Postgres : consultation de base (4/4)

- 3ième version: récupération des données dans un tableau via la fonction `pg_fetch_all`

```
<?php
  $nbrows = pg_num_rows($result);
  echo $nbrows . " ligne(s) retournée(s).\n";
  $rows = pg_fetch_all($result);
  foreach ($rows as $row) {
    echo "id: ".$row["id"]." " ;
    echo "auteur: ".$row["auteur"]."<br />";
  }
?>
```

- Fonction `pg_fetch_all` retourne un tableau de tableau associatif (indexé par noms des colonnes).
- La fonction `pg_num_rows` est utilisable pour toutes les versions de récupération de données

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires
  - Problèmes latence réseau :

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires
  - Problèmes latence réseau :
    - Un programme PHP renvoie une page complète

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires
  - Problèmes latence réseau :
    - Un programme PHP renvoie une page complète
    - Un programme PHP renvoie les données ET l'IHM

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires
  - Problèmes latence réseau :
    - Un programme PHP renvoie une page complète
    - Un programme PHP renvoie les données ET l'IHM
- Vers le web 2.0 (AJAX) :

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires
  - Problèmes latence réseau :
    - Un programme PHP renvoie une page complète
    - Un programme PHP renvoie les données ET l'IHM
- Vers le web 2.0 (AJAX) :
  - Technologie CSS : rendu graphique

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires
  - Problèmes latence réseau :
    - Un programme PHP renvoie une page complète
    - Un programme PHP renvoie les données ET l'IHM
- Vers le web 2.0 (AJAX) :
  - Technologie CSS : rendu graphique
  - Technologie JavaScript : comportement, vérifications des données

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires
  - Problèmes latence réseau :
    - Un programme PHP renvoie une page complète
    - Un programme PHP renvoie les données ET l'IHM
- Vers le web 2.0 (AJAX) :
  - Technologie CSS : rendu graphique
  - Technologie JavaScript : comportement, vérifications des données
  - Appel asynchrone

Conclusion

- Technologies PHP + HTML : OK mais insuffisant :
  - Composants graphiques sommaires
  - Problèmes latence réseau :
    - Un programme PHP renvoie une page complète
    - Un programme PHP renvoie les données ET l'IHM
- Vers le web 2.0 (AJAX) :
  - Technologie CSS : rendu graphique
  - Technologie JavaScript : comportement, vérifications des données
  - Appel asynchrone
  - Technologie XML : structuration d'une page (X)HTML, manipulation d'une page ou d'une partie d'une page