Cours HTML/PHP

E.Coquery

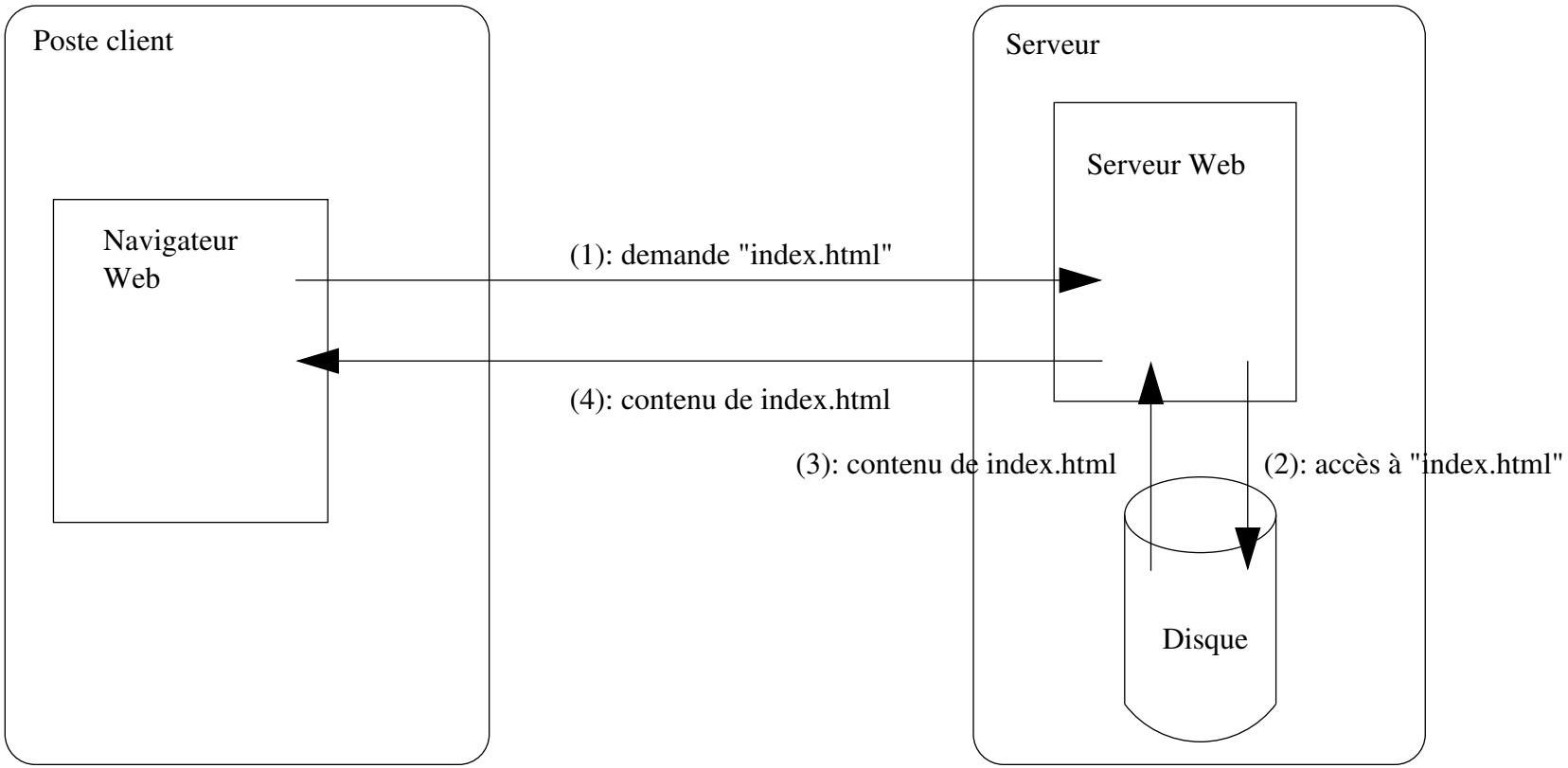
emmanuel.coquery@liris.cnrs.fr

Pages Web statiques

Principe de fonctionnement :

- L'utilisateur demande l'accès à une page Web depuis son navigateur.
  - Adresse tapée, clic sur un lien, utilisation d'un signet, etc ...
- Le navigateur envoie une demande à un serveur Web.
- Le serveur Web lit le fichier demandé sur le disque dur.
- Le serveur Web envoie le contenu du fichier au navigateur.
- Le navigateur affiche le contenu de la page.
  - Pour l'affichage, il peut également demander le contenu d'autres fichier au serveur (ex : images).

Illustration



Pages Web dynamiques

Pages statiques :

- Pour changer le contenu, il faut éditer le fichier.

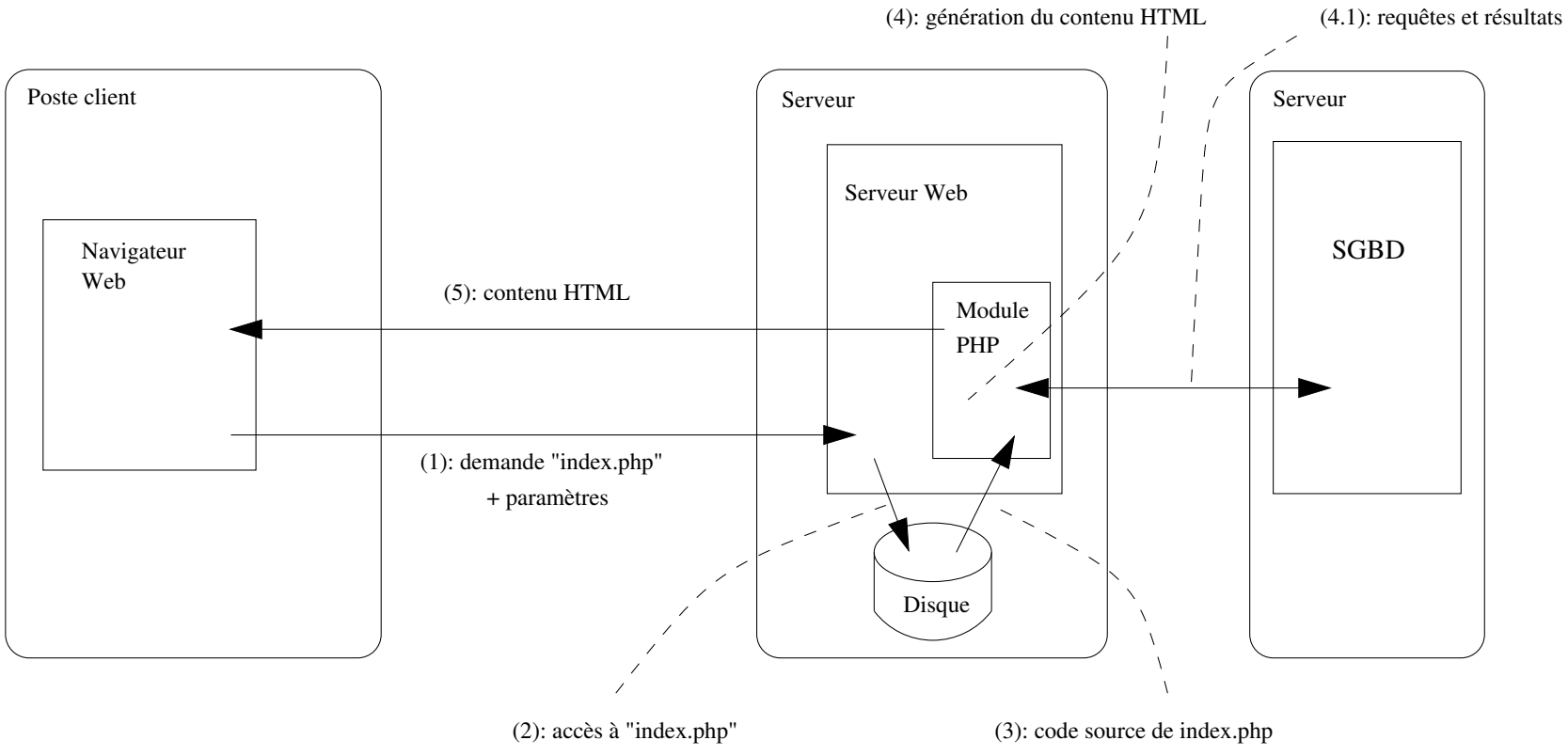
Idée : générer le contenu de la page au moment où elle est demandée, en fonction :

- du contenu d'une base de données ;
- de paramètres fournis avec la demande (ex : formulaires).

Pages Web Dynamiques : fonctionnement

- L'utilisateur demande l'accès à une page Web depuis son navigateur.
- Le navigateur transmet envoie une demande au serveur web comprenant :
  - le nom de la page, qui correspond en fait à un programme ;
  - éventuellement un ensemble de paramètres.
- Le serveur web va chercher sur le disque le code source d'un programme.
- Le serveur web exécute ce programme qui peut :
  - utiliser les paramètres transmis avec la demande du navigateur ;
  - aller chercher des données dans une base de données ou sur le disque.
- Durant l'exécution, le programme génère un contenu HTML.
- Le contenu HTML est envoyé au navigateur.
- Le navigateur affiche le résultat.

Illustration



Quelques langages

Langages utilisés :

- Pour le contenu des pages :
  - Données à afficher
  - Mise en forme

  $\Rightarrow$ HTML, CSS, JavaScript, ...)
- Pour générer le contenu des page :
  - Aller chercher les données
  - Fabriquer un document (en général un document HTML)

  $\Rightarrow$ PHP (ou bien Perl, Python, Java, ...)
- Pour interroger la base :
  - Requêtes à effectuer sur la base.

  $\Rightarrow$ SQL

Création de pages dynamiques

Dans notre cadre, pour créer une page dynamique, il faut :

- Créer un programme PHP correspondant à cette page.
  - Plus précisément, écrire le code source de ce programme.
- Le but de ce programme est de générer le contenu d'une page Web, *i.e.* du HTML.
  - Cette génération se fait via des `print`, un peu comme pour écrire dans un fichier.
- Ce programme peut éventuellement utiliser des requêtes SQL.
  - Ces requêtes sont passées sous forme de chaînes de caractères à des fonctions spéciales qui vont les transmettre au SGBD et récupérer les résultats.

Remarque : Bien que le but principal d'un programme PHP soit de produire un page HTML, cela reste un programme qui peut par conséquent avoir d'autres effets :

- Ex : insérer des valeurs dans une base de données

HTML : Principe

- Fichier texte contenant des informations de structuration.
- La structure est indiquée à l'aide de balises :
  - Le nom de la balise indique le type de mise en forme à appliquer.
  - On met une balise *ouvrante* au début du morceau de texte concerné :
    - `<nom_balise>début du texte`
  - On met une balise *fermante* à la fin du texte concerné :
    - `fin du texte</nom_balise>` (! au /)
  - Une balise ouvrante peut contenir des *attributs* de la forme `nom="valeur"`
    - `<nom_balise nom1="val1" nom2="val2" ...>texte`
    - Les attributs permettent de préciser des informations concernant la mise en forme.
  - Les balises peuvent être imbriquées.
- C'est le navigateur qui interprète les balises pour faire l'affichage.

Structure d'une page HTML

Une page HTML a la structure suivante :

```
<html>
  <head>
    <title>titre de la page</title>
  </head>
  <body>
    Partie affichée dans le navigateur.
  </body>
</html>
```

Les commentaires (non affichés) sont délimités par <!-- et -->.

Structure : paragraphes et titres

- `<p>texte</p>`
  `texte` forme un paragraphe (saut de ligne avant et après).
- `<h1>texte</h1>`
  `texte` est un titre important (paragraphe avec un affichage plus gros, en gras).
- `<h2>texte</h2>`
  `texte` est un titre moins important (affichage un peu moins gros).
- ...
- `<h6>texte</h6>`
  `texte` est un petit titre.

Exemple de document HTML

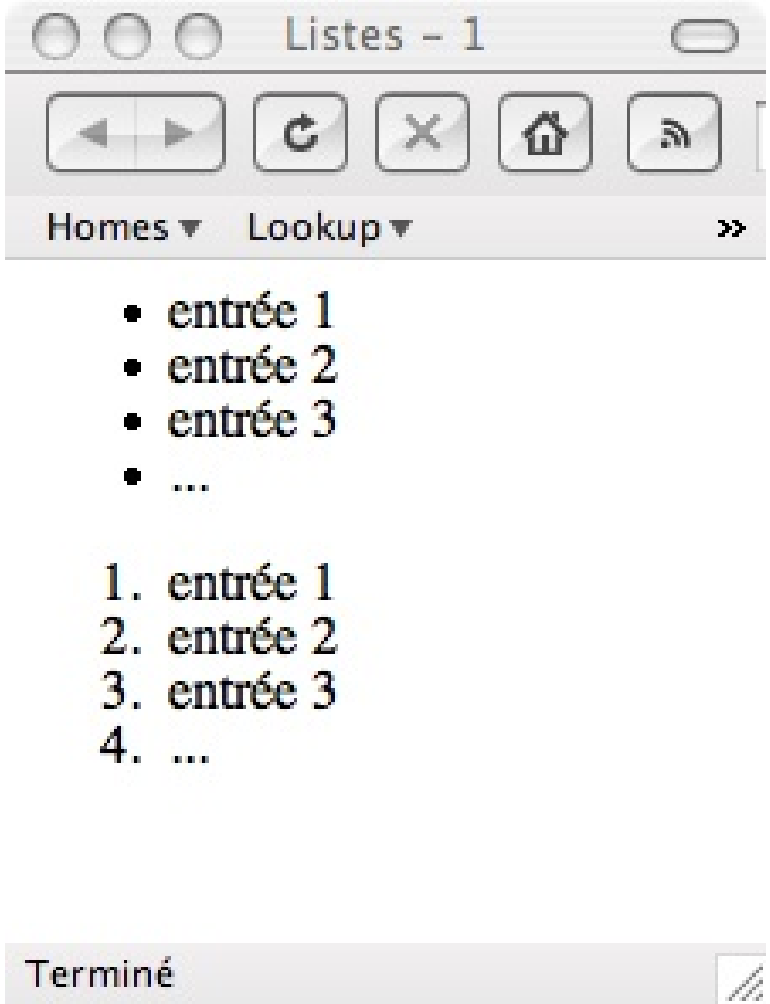
```
<html>
  <head>
    <title>La page de Toto</title>
  </head>
  <body>
    <h1>Toto</h1>
    <h2>L'histoire de Toto</h2>
    <p>Il était une
      fois ...</p>
    <p>Chemin
      faisant ...
    </p>
    <h2>Les amis de Toto</h2>
    <p>Ouioui</p>
    <p>Casimir</p>
  </body>
</html>
```

Structure : styles simples

- `<i>texte</i>` ou `<em>texte</em>` :
  mettre `texte` en italique.
- `<b>texte</b>` ou `<strong>texte</strong>` :
  mettre `texte` en gras.
- `<u>texte</u>` :
  souligner `texte`.
- `<big>texte</big>` :
  mettre `texte` en plus grand.
- `<small>texte</small>` :
  mettre `texte` en plus petit.

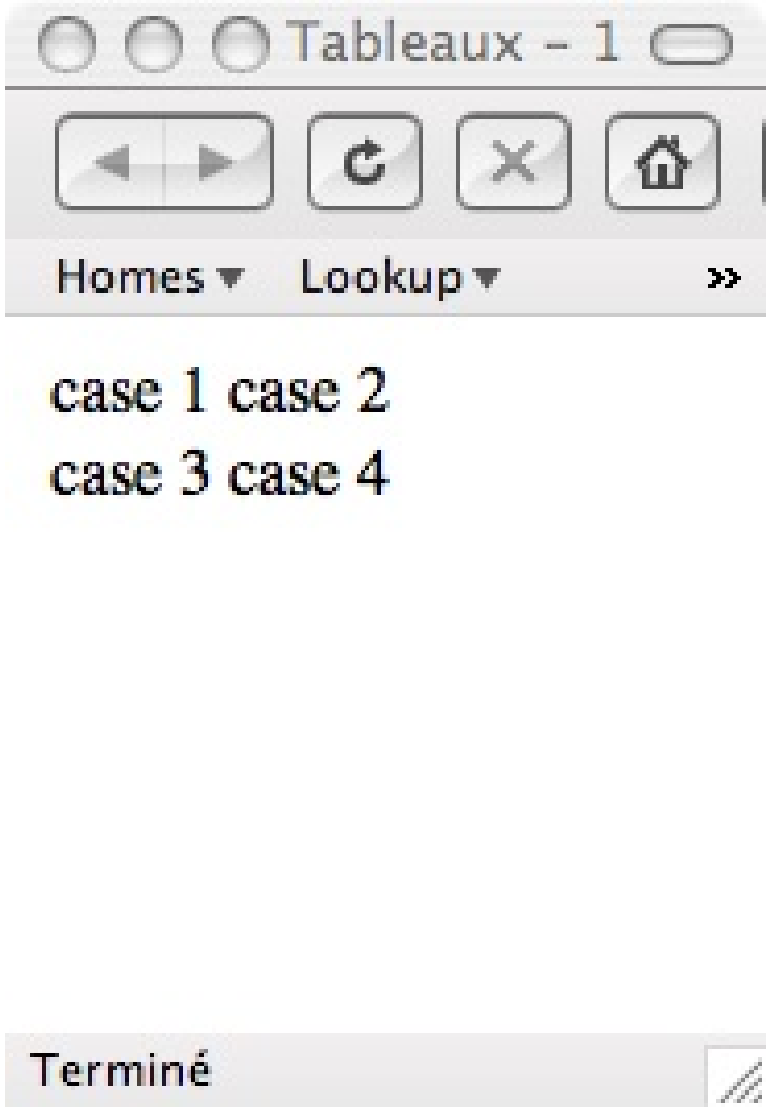
Structure : listes

```
<ul>
  <li>entrée 1</li>
  <li>entrée 2</li>
  <li>entrée 3</li>
  <li>...</li>
</ul>
<ol>
  <li>entrée 1</li>
  <li>entrée 2</li>
  <li>entrée 3</li>
  <li>...</li>
</ol>
```



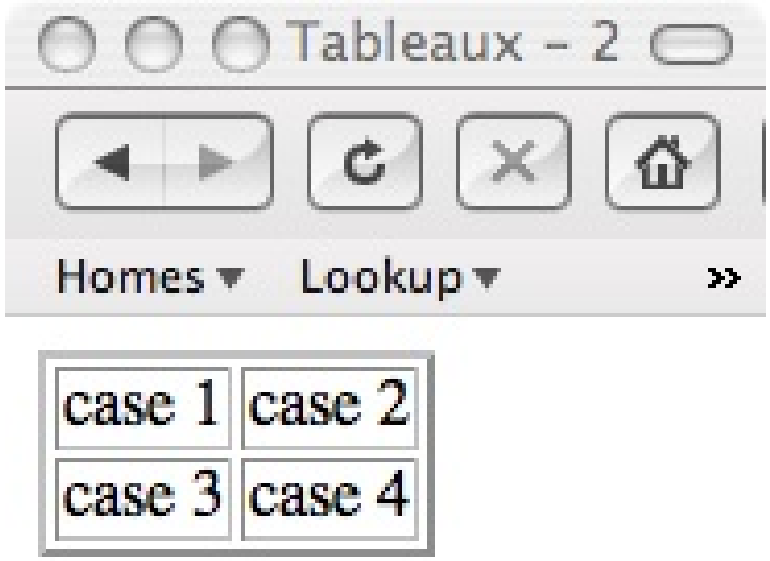
Structure : tableaux

```
<table>
  <tr>
    <td>case 1</td>
    <td>case 2</td>
  </tr>
  <tr>
    <td>case 3</td>
    <td>case 4</td>
  </tr>
</table>
```



Structure : tableaux - 2

```
<table border="2">
  <tr>
    <td>case 1</td>
    <td>case 2</td>
  </tr>
  <tr>
    <td>case 3</td>
    <td>case 4</td>
  </tr>
</table>
```



Liens hypertextes

```
<a href="adresse_web">texte</a>
```

- `texte` devient un lien cliquable.
- Lorsque l'on clique sur `texte`, on va à l'adresse (URL) `adresse_web`.

```
<a href="http://www.w3.org/TR/html401/">Documentation HTML</a>
```

- Le texte `Documentation HTML` est un lien vers la page web dont l'adresse est `http://www.w3.org/TR/html401/`

```
<a href="truc.html">Des trucs</a>
```

- Le texte `Des trucs` est un lien vers la page web `truc.html` située au même endroit que la page courante.

Images

Inclure une image :

- `<img src="adresse_web_image">`
- Pas de balise fermante pour `<img>`
- `adresse_web_image` : adresse web où chercher l'image.

Formulaires : principe

- Objectifs :
  - accéder à une page (dynamique) en spécifiant des *paramètres* ;
  - permettre à l'utilisateur de saisir ces paramètres.
- Comment :
  - en utilisant des champs textuels, des listes déroulantes, des cases à cocher ;
  - à chacun de des composants de saisie correspond un paramètre.

Formulaires : balise principale

```
<form action="adresse_web" method="POST" name="nom">
  contenu_formulaire
</form>
```

- `contenu_formulaire` : du texte contenant en particulier des balises représentant les différents composants de saisie ;
- `adresse_web` : l'adresse de la page web dynamique à laquelle on souhaite accéder via ce formulaire ;
- `POST` : la méthode de transmission des paramètres (on peut également mettre GET) ;
- `nom` : le nom du formulaire (optionnel).

Formulaires : saisie de texte

```
<input type="text" name="nom">
```

- Créée un champ de saisie pour une ligne de texte.
- `nom` est le nom du paramètre correspondant à ce composant.
- On peut ajouter les attributs suivants :
  - `size="un_nombre"` : la taille du champ en caractères ;
  - `value="une_valeur"` : texte pré-saisi
    - utile pour modifier des informations.
- On peut remplacer `type="text"` par `type="password"` si on veut afficher des * au lieu des lettres lors de la saisie.
- Pas de balise fermante.

Formulaires : saisie d'un grand texte

```
<textarea name="nom" rows="h" cols="l">
  contenu pré saisi
</textarea>
```

- Créée un champ de saisie pour du texte sur plusieurs lignes.
- `nom` est le nom du paramètre correspondant à ce composant.
- `h` est la hauteur du composant en nombre de lignes.
- `l` est la largeur du composant en nombre de caractères.
- le contenu pré saisi peut être vide et ne contient de balise.

Formulaires : paramètres cachés

```
<input type="hidden" name="nom" value="val">
```

- Permet de donner la valeur `val` au paramètre `nom`.
- Ce composant n'est pas affiché.
  - Utile pour spécifier un identifiant dans un formulaire de modification des informations de la base.
- Pas de balise fermante

Formulaires : liste déroulante

```
<select name="nom">
  <option value="val1">Texte 1</option>
  <option value="val2">Texte 2</option>
  ...
</select>
```

- Créée une liste déroulante ayant comme sélection possible `Texte 1`, `Texte 2`,...
- La valeur du paramètre `nom` est donnée par la sélection choisie par l'utilisateur :
  - `val1` pour `Texte 1`
  - `val2` pour `Texte 2`
  - ...
- L'attribut `value` est optionnel.
  - Par défaut c'est le texte dans la balise `<option></option>`
- On peut ajouter `selected="true"` dans une des balises options pour pré sélectionner cette option.

Formulaires : boutons de soumission

`<input type="submit" value="texte">`

- Créée un bouton déclenchant le chargement de la page de destination (attribut `action` de la balise `<form>`).
- `texte` est un texte qui sera affiché sur le bouton.

`<input type="reset" value="texte">`

- Créée un bouton déclenchant la réinitialisation du formulaire, en utilisant les valeurs pré saisies lorsqu'elles existent.
- `texte` est un texte qui sera affiché sur le bouton.

HTML : méthodes de transmission de paramètres

Deux méthodes de transmission des paramètres

- GET
  - Les paramètres sont encodés avec l'adresse de la page :
    - à la fin de l'adresse, on ajoute le caractère ?
    - puis pour chaque paramètre on ajoute `nom=val`
    - les paramètres sont séparés par le caractère `&`
  - Utile pour spécifier des paramètres dans un lien hypertexte.
- POST
  - Les paramètres sont encodés séparément de l'adresse web.
  - Plus pratique pour les formulaires.

PHP

- Un fichier PHP est le code source d'un programme.
- Ce programme a pour but de générer une page HTML.
- PHP est un langage impératif proche du C.

PHP : deux types de "zones"

- Délimitées par <? et ?>
- Zones entre <? et ?> : code PHP à exécuter (similaire à du code C).
- Zones à l'extérieur de <? et ?> : texte et balises qui seront recopiés directement dans le contenu HTML généré.

PHP : variables

- Le nom d'une variable commence par un $
  - `$i`, `$utilisateur`, `$id`, ...
- Affectation comme en C :
  - `$i =` *`valeur`* ;
- Les variables ne sont pas explicitement déclarées comme en C.
  - Une variable existe dès que l'on a fait une affectation dessus.
- Une variable peut contenir un nombre, une chaîne de caractères, un booléen (en réalité un entier comme en C) ou un tableau.

PHP : génération du contenu

Deux méthodes :

- Mettre du texte à l'extérieur de `<?` et `?>`.
- Utiliser l'instruction `print` :
  - `print` *`valeur`* ;
  - *`valeur`* est évalué puis transformé en texte.
  - Ce texte est ajouté à la suite du contenu HTML déjà généré.
  - On peut utiliser `echo` à la place de `print`.

On peut considérer que le texte mis à l'extérieur de `<?` et `?>` est passé en argument à un `print`.

PHP : chaînes de caractères

- Délimitées par des guillemets simples `('chaîne')`ou doubles (`"chaîne"`).
- Si une variable apparaît dans une chaîne avec guillemets doubles, elle est remplacée par sa valeur (convertie en chaîne de caractères).
  - `print "<p>Mon nom est $nom</p>" ;`
- Les variables apparaissant dans des guillemets simples ne sont pas remplacées.
- Un `.` entre deux chaînes les concatène.
  - `print "<p>Mon nom".$nom."</p>" ;`

PHP : opérateurs courants

Arithmétiques :

- + (addition), - (soustraction), * (multiplié), / (divisé), % (modulo), ++ (incrément), -- (décrément).

De comparaison :

- == (égalité), < (inférieur strict), <= (inférieur large), >, >=, != (différence)

Logiques :

- `and`, && (et), `or`, || (ou), `xor` (ou exclusif), ! (non)

PHP : tableaux

- Syntaxe similaire au C :
  - `$mon_tableau[2] = "coucou" ;`
    - Range `"coucou"` dans la case numéro 2.
  - `print $mon_tableau[2] ;`
    - Génère le texte coucou.
- `$mon_tableau[] =` *`valeur`* `;`
  - Ajoute une case au tableau `$mon_tableau` et y range `valeur`.
- La première case d'un tableau porte le numéro 0.
- Pour créer un tableau vide, on peut utiliser :
  - `$mon_tableau = array() ;`

PHP : tableaux associatifs

- Tableau associant une valeur à une chaîne de caractères.
- Syntaxe :
  - `$personne['Prenom'] = 'Toto' ;`
    - Associe la valeur `'Toto'` à la chaîne `'Prenom'`.
  - `print $personne['Prenom'] ;`
    - Génère le texte `Toto`.

PHP : contrôles

Structures de contrôles similaires à celles de C :

- `if (...) { ...} else { ...}`
- `for(... ;... ;...) { ...}`
- `while (...) { ...}`

Inclusion d'un autre fichier PHP :

- include("nom_fichier.php") ;

PHP : fonctions

Définition de fonction :

```
function nom_fonction($param1, $param2, ...) {
   ...
}
```

- Pour renvoyer un résultat dans une fonction :
  - `return` *`valeur`* `;`

PHP : récupération des paramètres

Principe :

- L'interpréteur PHP initialise un tableau associatif qui associe à chaque nom de paramètre sa valeur.

Le nom du tableau initialisé dépend de la méthode :

- `$_POST` pour la méthode POST
- `$_GET` pour la méthode GET

Exemple : afficher la valeur du paramètre `nom`, s'il est transmis par la méthode POST.

```
print $_POST['nom'] ;
```

Accès à une base de donnée MySQL

Cinq étapes :

1. Connexion au SGBD.
2. Sélection d'une base.
3. Envoi d'une requête.
4. Récupération et utilisation du résultat.
5. Fermeture de la connexion.

On peut itérer les étapes 3 et 4 autant de fois que l'on veut avant de fermer la connexion à l'étape 5.

Connexion

- Connexion au SGBD :

```
$user = 'toto' ;
$passwd = 'mdptoto' ;
$machine = 'localhost' ;          machine où tourne PHP
$connect = mysql_connect($machine,$user,$passwd)
            or die('Echec de connexion au SGBD') ;
```

- Choix de la base :

```
$bd = 'entreprise' ;
mysql_select_db($bd,$connect)
 or die('Echec lors de la selection de la base') ;
```

- Fermeture (après les requêtes) :

```
mysql_close($connect) ;
```

Envoi de la requête

```
$requete = 'une requete SQL' ;
$resultat = mysql_query($requete,$connect)
    or die('Erreur durant l'exécution de la requête') ;
```

Exemple :

```
$salaire_max = 20000 ;
$requete = "SELECT nom FROM employe "
                    ." WHERE salaire <= $salaire_max" ;
$resultat = mysql_query($requete,$connect)
  or die('Erreur durant l'exécution de la requête') ;
```

Durant la phase de développement, il peut être utile d'afficher $requete avant son envoi au SGBD.

Exploitation du résultat d'une requête

Code type pour parcourir le résultat :

```
while ($nuplet = mysql_fetch_assoc($resultat)) {
   ...
}
```

- La boucle `while` permet de parcourir les n-uplets qui forment le résultat (un n-uplet par tour de boucle).
- `$nuplet` est un tableau associatif qui associe à chaque attribut du résultat sa valeur pour le n-uplet courant.

Si la requête est une mise à jour, il est inutile de parcourir le résultat.

Exemple

```
$salaire_max = 20000 ;
$requete = "SELECT nom,salaire FROM employe "
                    ." WHERE salaire <= $salaire_max" ;
$resultat = mysql_query($requete,$connect)
    or die('Erreur durant l'exécution de la requête') ;
print "<h3>Employés gagnant moins de "
                    ."$salaire_max euros par an</h3>" ;
while ($nuplet = mysql_fetch_assoc($resultat)) {
  $nom = $nuplet['nom'] ;
  $sal = $nuplet['salaire'] ;
  print "<p>$nom gagne $salaire euros par ans.</p>" ;
}
```

Sessions : pourquoi ?

Il peut être utile de conserver des informations d'une page sur l'autre. Par exemple pour :

- se souvenir du login de l'utilisateur
- se souvenir des références indiquant à quoi l'utilisateur s'intéresse
- se souvenir des dernières pages vistées par l'utilisateur
- etc

Jusqu'ici, un seul moyen : utiliser des paramètres et penser à les remettre à chaque lien et dans chaque formulaire

$\Rightarrow$ Programmation fastidieuse et source de problèmes.

Sessions

Une session peut être vue comme un ensemble d'informations concernant un utilisateur d'un site.

- par utilisateur, on entend un navigateur sur une machine
- les informations sont conservées entre deux pages
- une page PHP peut ajouter ou modifier des informations

En PHP, la session est vue comme une variable spéciale appelée `$_SESSION` :

- c'est un tableau associatif
- sa valeur est conservée d'une page sur l'autre

Utilisation des sessions en PHP

- Une page PHP utilisant une session doit *obligatoirement*, *avant même d'afficher quoi que ce soit*, commencer par l'instruction :

```
session_start() ;
```

- Cette instruction crée la variable `$_SESSION` et la remplit avec les valeurs qu'elle avait dans la page PHP précédente.
- La variable `$_SESSION` se manipule ensuite comme un tableau associatif classique.

Déconnexion

- Lorque l'utilisateur se déconnecte, il est important de détruire la session
  - par exemple pour éviter qu'une seconde personne utilisant le même ordinateur ne se fasse passer pour la première personne
- pour détruire une session :

```
$_SESSION = array();
session_write_close();
```