Programmation événementielle - HAI203I

Michel Meynard

UM

Univ. Montpellier

Table des matières I

Table des matières II

Plan

Plan

Prérequis I

Les étudiants doivent avoir des connaissances de base sur :

- l'utilisation d'un navigateur Web (Chrome(ium), Firefox, ...)
- l'utilisation d'un éditeur de code simplifiant l'écriture du HTML grâce à la complétion interactive : on préconise **fortement** l'utilisation de `vscode(ium)` présent dans les salles machines
- connaissances de base en programmation : variables, types, structures de contrôles, fonctions
- la recherche de références concernant HTML, PHP, Javascript, les feuilles de style CSS `https://www.w3schools.com/` ou `https://developer.mozilla.org/fr`
- références biblio. :[1]

Objectifs

Les objectifs de ce cours sont nombreux :

- Séance 1 : HTTP+HTML, CSS, bootstrap
- Séance 2 : apprentissage d'un langage de programmation côté serveur PHP avec requêtes effectuant des calculs et des rendus sans mémoire
- Séance 3 : sessions PHP pour mémoriser côté serveur, téléchargement, cookies pour mémoriser côté client
- Séance 4 : modèle objet de PHP, PDO pour accéder aux BDs, MVC

Plan

HTTP et architecture du Web I

Quelques définitions de base sont nécessaires :

HTTP `HyperText Transfer Protocol` est un protocole de communication **client-serveur** développé pour le World Wide Web (www)

HTTP inventé par Tim Berners-Lee avec les adresses web (URL) et le langage HTML

port utilisé par défaut le port 80

HTTP est un protocole **non orienté connexion**

client HTTP (navigateur) envoie une requête au serveur qui lui retourne une réponse généralement sous la forme d'un fichier HTML. Le client affiche alors la réponse qui contient généralement des **liens** hypertextes ou des formulaires, qui une fois cliqués ou soumis génèrent une requête au serveur etc.

HTTP et architecture du Web II

serveur HTTP Apache, Nginx, IIS, ... sont des serveurs logiciels installés sur des machines appelées également “serveur” (matériel)

client HTTP appelé aussi client léger ou navigateur Web, les plus connus sont Firefox, Internet Explorer, Opera, ...

URL *Uniform Resource Locator* est une chaîne de caractères codée en ASCII pour adresser les ressources du World Wide Web : document HTML, image, son, forum Usenet, boîte aux lettres électroniques, etc.

exemple d'URL appelée aussi **adresse web** : `http://www.google.fr/`, `mailto:toto@titi.fr`, `http://www.lirmm.fr/~meynard/Ens2/rubrique.php3?id_rubrique=36&x=1`, ...

HTTP et architecture du Web III

URL relative à l'intérieur d'une page HTML, des liens vers d'autres pages du même site peuvent être définis avec une écriture relative au répertoire courant de la page : par exemple, `../images/toto.jpg` est une URL relative.

Plan

Application Web I

- HTTP n'est pas orienté connexion : le serveur ne mémorise aucune information à propos du client
- entre 2 requêtes du client, il peut se passer 2 secondes ou un temps infini !
- Donc l'écriture des applications côté serveur doit prendre en compte cette absence de mémoire
- une part non négligeable du traitement dans une application peut être effectué côté client afin d'effectuer des vérifications (contrôles) mais aussi des calculs métiers !
- éviter de surcharger le serveur
- des règles de sécurité interdisent aux scripts côté client d'accéder au ressources du poste client (fichiers, imprimante, ...) sauf si l'utilisateur le demande !

Application Web II

- Actuellement, avec les Single Page Application (SPA), l'application est située majoritairement sur le client qui la charge au début puis cette appli. fait appel à des Web Services (API Rest) quand cela est nécessaire.

Côté Serveur I

L'application côté serveur utilise plusieurs technologies possibles :

cgi script (python, bash) ou binaire (compilé par g++) chargé dans un processus externe au serveur http (fork). La sortie standard de ce processus est redirigé dans la réponse envoyée par le serveur au client.

inconvénient principal des cgi perte de temps nécessitée par la création d'un processus pour chaque nouvelle requête ;

module le serveur intégre des modules interprétant des langages de programmation tels que Perl, Python, PHP. L'interprétation de script est beaucoup plus rapide car dans un Thread

Node.js un serveur Node.js est un démon JavaScript qui tourne sur le serveur et qui sert chaque requête sans créer de processus léger ou lourd en utilisant le caractère asynchrone de Js !

Côté Client I

L'application côté client peut utiliser plusieurs technologies possibles :

javascript langage de script interprété par le navigateur et permettant de manipuler l'arborescence du document(DOM) et de nombreuses autres API ;

applet Java mini application Java permettant d'utiliser toute la puissance du langage Java (API, structures de données, ...)

ActionScript langage de script compatible avec JavaScript et permettant de réaliser des animations Flash ou Flex ...

Plan

Plan

Histoire de HTML, XHTML, HTML5 I

- “HyperText Markup Language”, HTML, est le langage conçu pour représenter les pages web.
- août 1991 : Tim Berners-Lee (CERN) annonce publiquement le web sur Usenet, en donnant l’URL d’un document de suffixe .html
- HTML basé sur SGML jugé trop complexe
- Différentes versions de HTML jusqu’à la version 4.01 puis ...
- “eXtensible HyperText Markup Language”, XHTML 1.0, destiné à remplacer HTML 4 est une application XML.
- HTML5 apparue depuis 2006 sous la houlette du World Wide Web Consortium (W3C) et du Web Hypertext Application Technology Working Group (WHATWG)
- Beaucoup des recommandations de cette norme sont implémentées sur les navigateurs récents mais pas toutes ...

Histoire de HTML, XHTML, HTML5 II

- Nous utiliserons le HTML5 sans examiner toutes les capacités de ce langage.

Plan

Caractéristiques de HTML I

- A l'origine, contenu et format de document étaient mélangés dans un même fichier HTML
- éléments de structuration du contenu tels que `div`, `h1`, `h2`, `table`, ...
- éléments de mise en forme tels que `b` (bold), `i` (italic),`center`, ...
- Certaines mises en page utilisaient les tableaux, ce qui est maintenant blâmé
- Actuellement, feuilles de style (en cascade) "Cascading Style Sheets", CSS, pour le format
- La feuille de style est souvent externe au fichier HTML : fichier .css
- W3C (*World Wide Web Consortium*) chargé de rédiger des recommandations (sorte de norme) concernant les technologies du web `http://www.w3.org/`.

Caractéristiques de HTML II

- valider ses documents en ligne à l'adresse `http://validator.w3.org/`
- WHATWG (`https://whatwg.org/`) rédige lui le HTML living standard
- autre site sans lien avec le W3C `http://www.w3schools.com/` qui est un site pédagogique très pratique même s'il est commercial.

Plan

Vocabulaire I

- document : l'ensemble du texte composé du fichier principal et des fichiers inclus ou référencés (script, feuille de style, image, ...) ;
- Document Object Model (DOM) est le modèle arborescent d'une page Web : ses noeuds sont appelés `Node` ou pour les noeuds correspondant à deux balises `Element` ;
- langage à balisage : `<balise>contenu ...</balise>` ;
- élément : composé d'une balise ouvrante puis d'un contenu (possédant éventuellement d'autres éléments) puis de la balise fermante correspondante ;
- élément vide : ne possédant pas de contenu, la balise ouvrante est également fermante comme dans `<br />` ;
- attribut : information de la forme `nom='valeur'` présente uniquement dans une balise ouvrante ;

Vocabulaire II

- validité : un document HTML est valide s'il correspond aux règles édicté par le "World Wide Web Consortium" (w3c) ;
- URI : *Uniform Ressource Identifier* adresse d'une ressource Internet composé d'un protocole, d'un nom de domaine complètement qualifié (FQDN), d'un chemin, et d'autres paramètres possibles ; Par exemple, `http://www.lirmm.fr/~meynard/ProjetInfoL3/index.php?rubrique=Projet&action=liste` est une URI. Autre exemple, `mailto:toto@tutu.fr` désigne une adresse email ;

Plan

Syntaxe de HTML5 I

- HTML5 extrêmement laxiste vis-à-vis de XHTML 1.0 : les balises `html, head, body` sont optionnelles, les valeurs d'attribut ne sont pas forcément entourées de guillemets, les éléments vides ne sont pas forcément fermés ...
- Pour des raisons de lisibilité du code, nous continuerons à utiliser une syntaxe stricte héritée de xhtml
- les noms d'attribut sont en minuscules
- les valeurs d'attribut doivent être entre apostophes ou bien entre guillemets
- les éléments vides sont définis par une seule balise, comme dans `<br />` et dans `<img src="toto.jpg" />` ;
- les attributs présentiels n'ont pas de valeur puisque leur présence suffit (pseudo valeur booléenne) ; voir l'exemple suivant avec autofocus ou multiple.

Syntaxe de HTML5 II

```
<input type="text" name="prenom" autofocus />
<select multiple name="ues" ...
```

Exemple de document HTML minimum : modele.html I

```
<!doctype html>
<html lang="fr">
<head>
<meta charset="utf-8" />
<title>mon premier site</title>
</head>
<body>
<header>
<h1>Bandeau haut du site</h1>
</header>
<div>
<h2>Partie principale</h2>
<p>blabla ...
<h2>Titre de niveau 2</h2>
```

Exemple de document HTML minimum : modele.html II

```
<p>hello <b>World</b> ! éàè</p>
</div>
<footer>
<a href="contact.html">Nous contacter</a>
<a href="cgv.html">Conditions générales de vente</a>
</footer>
</body>
</html>
```

doctype DTD (Document Type Definition) définit la syntaxe d'un type de document : grammaire formelle indiquant les imbrications possibles d'éléments, les attributs obligatoires, optionels, ...

Exemple de document HTML minimum : modele.html III

historique La déclaration de la DTD `<!DOCTYPE ...` était auparavant complexe : URI du fichier de DTD. Il y avait plusieurs DTD pour XHTML 1 (strict, transitional, frameset)

élément html la racine du document qui possède l'attibut spécifiant la langue du site (fr ançais)

head l'en-tête du document non visible

body le corps du document qui sera affiché sur la page

Plan

L'en-tête du document I

L'élément `head` peut et **doit** contenir d'autres éléments que le titre du document `title` :

- meta fournit des méta-informations (ou métadonnées) sur la page en cours qui pourront être utilisées par les moteurs de recherche ou par le navigateur. Ses attributs :
  - charset indique l'encodage du fichier HTML. Pour le français les 2 encodages les plus fréquents sont l'`iso-latin-1` (1 octet par lettre accentuée), ou l'`utf-8` (codage multi-octets mais universel)

L'en-tête du document II

http-equiv permet d'envoyer des en-têtes HTTP au navigateur pour l'aider à interpréter le document. Par exemple,
`<meta http-equiv="refresh" content="5" />`
permet de rafraîchir la page au bout de 5 secondes. La valeur associée à `http-equiv` sera dans l'attribut `content`.

name est optionel et peux prendre les valeurs `author`, `description`, `keywords`. La valeur associée à ce nom sera dans l'attribut `content`

content définit la valeur de la méta-information. Par exemple,

```
<meta name="keywords" content="fromage,
↪  camembert, produit frais" />
```

L'en-tête du document III

script permet de référencer un fichier script externe et/ou de définir des fonctions JavaScript locales. Par exemple :

```
<script src="monscript.js"></script>
```

Attention à ne pas créer un élément vide `<script ... />` lorsque tout le code est dans un fichier externe. Cela ne fonctionne pas !

link permet de lier un autre document. Par exemple, pour définir la feuille de style associée :

```
<link rel="stylesheet" href="messtyles.css" />
```

style permet de définire des styles en ligne sans utiliser de fichier externe

Plan

Le corps du document I

Le corps (*body*) du document HTML constitue ce qui va être affiché sur l'écran. Par défaut, les éléments sont affichés selon leur type :

- les éléments en ligne (*inline*) sont placés de gauche à droite jusqu'à ce qu'il n'y ait plus de place dans leur bloc et un retour à la ligne automatique est géré par le navigateur ; ainsi le texte d'un paragraphe est il affiché avec des coupures de lignes dépendant de la taille de la fenêtre du navigateur ; de même les ancres sont des éléments en ligne ;
- les éléments de type `bloc` sont affichés sur **toute la largeur de leur parent** avec une coupure de ligne avant et après eux ; les paragraphes sont de type bloc ainsi que les titres (h1, h2, . . .)

Un grand nombre de balises *sémantiques* ont été ajoutées en HTML5 afin d'éviter la multiplication des blocs (`div`) pour des usages différents

Plan

Attributs de base I

Un certain nombre d'attributs de base communs à quasiment tous les éléments (*core attributes*) peuvent être utilisés.

class classe CSS de l'élément. Cela permet de formatter l'élément selon un style défini dans une classe déjà définie ; par exemple, `class='monvert'` où monvert est une classe de style ;

style style en ligne ; par exemple, `style='color:green;'`

id identifiant **unique** de l'élément permettant sa manipulation en JavaScript grâce à la méthode : `document.getElementById(id)` ;

title *tooltip* affiché lors du survol de l'élément ;

Attributs de base II

De plus, une gestion des événements souris et clavier permettent d'associer des scripts JavaScript. Cette association est réalisée grâce à des attributs tels que `onclick` ou `onkeypress` qui peuvent être associés à un gestionnaire d'événements. Attention à ne pas confondre l'identifiant (id) qui est utilisé pour la manipulation côté client et le nom (name) des champs de formulaire qui lui caractérise les paramètres des requêtes passées au serveur lors de la soumission.

Plan

Eléments indispensables I

a *anchor* les ancres mettent en oeuvre l'hypertexte et peuvent être utilisées de 2 façons :

1. lien hypertexte référençant une autre page web grâce à l'attribut `href`. Par exemple, pour aller sur le site du `w3schools` :

```
<a href='http://www.w3schools.com/'>CLIQUEZ
↪  ICI</a>
```

2. marque-page indiquant une cible potentielle d'un lien :

```
<a id='toto'>Texte cible</a>
```

Plus loin dans le même document, on pourra référencer cette cible par le lien suivant :

```
<a href='#toto'>aller au texte cible</a>
```

Bien entendu, on peut faire une référence externe sur une cible en suffixant le chemin de l'url par `#toto` ;

Eléments indispensables II

Enfin, la valeur d'un href peut-être une pseudo-URL contenant du code JavaScript qui sera exécuté lors de l'activation du lien comme dans l'exemple suivant :

```
<a href="javascript:alert('Bonjour le monde !')">Mon
↪  Lien</a>
```

Remarquons que le code est préfixé par `javascript:` afin d'indiquer le langage

script exécution de code `javascript` dans le corps de page ; par exemple :

```
<script>document.writeln('Bonjour le monde
↪  !');</script>
```

p paragraphe contenant une suite de phrases ;

Eléments indispensables III

br *break* élément vide qui permet de passer à la ligne suivante ; en HTML les espaces, tabulations et retour ligne (CR et/ou LF) multiples ne sont perçus que comme un espace séparateur lorsqu'ils sont situés entre deux mots.

h1 *header* de niveau 1 est un titre de section ; on trouve également `h2` ...

ul *unordered list* liste d'items non numérotés ;

ol *ordered list* liste d'items numérotés ;

li *list item* élément d'une liste numérotée ou non ;

div *division* est une section logique du document permettant de regrouper plusieurs blocs dans un même formatage ; on l'utilise souvent dans la mise en page du document pour scinder les différentes parties (bandeau haut, menu de gauche, page centrale, bandeau du bas) ;

Eléments indispensables IV

table les tables HTML sont un moyen d'afficher les tableaux mais pas de mettre en page un document (utiliser plutôt les div). Les tables peuvent ou doivent contenir :

- caption la légende de la table
- tr *table row* une ligne
- td *table delimiter* une case ; `colspan` et `rowspan` sont des attributs indiquant le nombre de cases à fusionner avec la case courante vers la droite et vers le bas
- th *table header* une case de titre

hr *horizontal rule* ligne de séparation horizontale ;

img *image* en format gif, png ou jpg ; les attributs indispensables :

- src *source* chemin du fichier image

Eléments indispensables V

alt *alternative* texte utilisé si le navigateur ne sait pas afficher les images

width largeur en pixels

height hauteur en pixels

form les formulaires sont décrits dans la section suivante

header HTML5 à ne pas confondre avec `head`, cet élément contient l'entête visuel d'un document ou d'une section (appelé aussi bandeau haut). Il inclut souvent une barre de menus (nav).

footer HTML5 contient le pied de page visuel d'un document ou d'une section (appelé aussi bandeau bas)

nav HTML5 contient une liste de liens externes ou internes. Il est typiquement utilisé pour implémenter un menu ou un fil d'Ariane :

```
rayon informatique > clé USB > par marque
```

Eléments indispensables VI

article HTML5 portion de page indépendante du reste de la page

section HTML5 groupement de contenu incluant généralement un titre

aside HTML5 groupement de contenu non primordial ;

time HTML5 pour spécifier une date compréhensible par javascript

figure, figcaption HTML5 pour insérer un **flottant** composé d'images de textes et d'une légende,

Plan

Les formulaires de base I

- Les formulaires constituent des éléments indispensables dans la saisie d'informations à destination d'une application web
- Un ou plusieurs formulaires peuvent être présents dans une même page, seul le formulaire soumis sera envoyé
- L'attribut `name` de chaque champ de saisie correspond au nom du "paramètre" qui contiendra l'information
- Plusieurs champs peuvent avoir le même nom, dans ce cas le paramètre sera de type tableau si le nom est suffixé par `[]`
- L'attribut `tabindex` permet d'ordonner les champs de saisie quand l'utilisateur appuiera sur la touche de tabulation

Eléments de formulaires I

Dans HTML5, de nouveaux éléments de saisie sémantiques sont apparus associés à des contrôles de validité automatiques. Ils sont plus ou moins bien implémentés selon l'âge du navigateur.

form élément racine d'un formulaire contenant les attributs suivants :

- **action** est l'URI où sera soumis le formulaire (généralement un script php ou un programme cgi). Cette URI peut être absolue ou relative au répertoire courant. La norme recquiert cet attribut qui doit être une URI, or une URI ne peut être vide mais, la norme HTML4.0 admet l'exception d'attribut vide pour désigner le document courant !

Eléments de formulaires II

method soit get, soit post ; la première (get) insère les champs saisis par l'utilisateur à la fin de l'url après un point d'interrogation sous la forme `http://localhost/index.php?nom1=val1&nom2=val2` ; les noms sont les valeurs des attributs `name` des champs tandis que les valeurs sont les contenus saisis par l'utilisateur. La seconde méthode post "cache" les contenus saisis

target avec la valeur `_blank`, une nouvelle fenêtre sera ouverte, avec `_self` (par défaut) on recouvrira la fenêtre actuelle.

onsubmit code javaScript à exécuter avant la soumission ; Si le code JavaScript retourne faux, la soumission **est annulée** !

Eléments de formulaires III

name ou id : nom ou id du formulaire pouvant être utile pour le référencement des champs de saisie dans le code javaScript de vérification

label : texte préfixant le champ de saisie et lié à lui par l'attribut `for`. Indispensable pour l'accessibilité

input : élément **vide** définissant un champ de saisie ayant :

- un attribut **name**, le nom du champ evoyé comme nom de paramètre lors de la soumission
- un attribut **type** permettant d'indiquer la forme du champ de saisie parmi :

  text : champ de type texte libre sur une ligne ; autres attributs : `size` la taille du champ de saisie à l'écran, `maxlength` le nombre maximum de caractères à saisir, `value` valeur par défaut

Eléments de formulaires IV

password champ de mot de passe caché

button bouton permettant de déclencher un script javascript ; autres attributs : `value` valeur affichée à l'écran, `onclick` code javaScript à déclencher

checkbox case à cocher ; autres attributs : `value` valeur affectée au nom dans le cas où cette case est cochée, `checked="checked"` si on veut que la case soit cochée par défaut

Eléments de formulaires V

radio case à cocher **exclusive** ; tous les boutons radio mutuellement exclusifs doivent avoir le même attribut `name` ; autres attributs : `value` valeur affectée au nom dans le cas où cette case est cochée, `checked="checked"` si on veut que la case soit cochée par défaut

hidden champ caché n'apparaissant pas dans le formulaire ; historiquement, les champs cachés permettaient de faire transiter l'information passée de proche en proche lors de la navigation ; autres attributs : `value` ; Actuellement, le mécanisme de session est plus pratique à utiliser

Eléments de formulaires VI

submit bouton de soumission permettant d'exécuter l'`action` du formulaire ; plusieurs boutons de soumission peuvent coexister dans un même formulaire

Plan

Plan

Introduction I

Le langage CSS (Cascading Style Sheets : feuilles de style en cascade) sert à décrire la présentation des documents HTML et XML

- standards définissant CSS publiés par le World Wide Web Consortium (W3C)
- introduit au milieu des années 1990
- depuis CSS3, la nouvelle norme est rétrocompatible. Elle est découpée en modules d'un niveau :
  - Selectors :
  - Box Model ;
  - Backgrounds and Borders ;
  - Text Effects ;
  - 2D/3D Transformations ;
  - Animations ;
  - Multiple Column Layout ;
  - User Interface ;

Introduction II

- CSS 4 (2017) n'existera pas : il n'y a que des niveaux (level) d'un module : CSS Selectors Level 3, CSS Cascading and Inheritance Level 4, ... : `http://www.w3.org/TR/`

Plan

Objectifs I

- séparer la structure d'un document de ses styles de présentation ;
- définir le rendu d'un document en fonction du média de restitution et de ses capacités (type de moniteur ou de dispositif vocal), de celles du navigateur textuel, ...
- permettre la cascade des styles, c'est-à-dire un héritage multiple entre les origines (auteur, utilisateur), le média, la localisation des définitions de style (externe, en ligne, ...) ;

Plan

Syntaxe I

Un style est défini par :

```
selecteur {prop1:val1; prop2:val2 val3 val4; prop3: v1, v2;}
```

Le **sélecteur** définit l'**ensemble** des balises affectées par le style. Chacune des 53 **propriétés** est répartie dans un groupe parmi les 6 suivants :

font contenant font-family, font-size, font-weight, ...

color, background contenant color, background-color, background-repeat, background-image, ...

text contenant text-align, text-indent, ...

box, layout contenant padding, border, margin, width, height, display, float, ...

list contenant list-style-type, list-style-position, ...

table contenant caption-side, padding, border-collapse, ...

Les **valeurs** de propriétés sont réparties dans les catégories suivantes :

Syntaxe II

mot-clé non sensible à la classe tel que : red, underline, bold, top, collapse ...

longueur nombre décimal absolu ou relatif (démarrant par un signe) suivi par une unité sur 2 lettres. Les unités suivantes sont possibles :

- **cm, mm** centimètre, millimètre ;
- **in** inch (2.54cm) ;
- **pt** point (1/72 inch) ;
- **pc** pica (12pt ou 4.2mm) ;
- **px** pixel (dépend de la résolution de l'écran) ;
- **em** hauteur de la police courante ;
- **ex** hauteur d'un x de la police courante ;

Syntaxe III

Par exemple, +0.25mm, -3em, 100px, sont des longueurs correctes dont les deux premières indiquent un agrandissement et un rétrécissement par rapport à la valeur par défaut ;

pourcentage longueur relative telle que width :50%, line-height :120%, ...

couleur exprimée en notation RGB (Red, Green, Blue) par :

- **6 chiffres hexadécimaux** tels que #FFFFFF pour le blanc ;
- **3 chiffres hexadécimaux** tels que #000 pour le noir ; #F00 pour le rouge (chaque chiffre est répété) ;
- **rgb(128,0,128)** pour le pourpre (également rgb(50%,0,50%)) ;

url avec la notation `url(http://toto.fr/chemin)` ;

Plan

Exemple avec styles dans l'en-tête I

- styles définis dans l'en-tête html du document
- pour p, polices séparées par des virgules : **un seul** des éléments de la liste sera choisi (le premier possible)
- pour border de div, **concaténation** (sans virgule) de valeur (color, width, style).

```
<style type="text/css">
<!--
p {
  font-family : cursive, fantasy, monospace;
  line-height : 100%;
}
h1 {
  text-align: center;
  font-variant : small-caps;
}
```

Exemple avec styles dans l'en-tête II

```
div {
  float: left;
  width: 30%;
  border: blue 1px solid;
}
-->
</style>
</head>
<body> ...
```

- commentaires html entourant la définition des styles : navigateurs ne comprenant pas les styles CSS
- valider sa feuille de style : validateur du W3C : `http://jigsaw.w3.org/css-validator/`

Exemple avec styles dans l'en-tête III

- Les divisions flottantes permettent de placer les divisions dans le flot des boîtes affichées. Ici, trois divisions (3*30%<100%) pourront être placées horizontalement dans la page.

positionnement fixe du bandeau haut I

- ensemble de styles permettant de définir un bandeau haut (div) fixée en haut du navigateur
- une division principale qui lorsqu'elle défilera, passera sous le bandeau
- la propriété `z-index` définit l'empilement des boîtes à l'affichage (par défaut 0)

```
<style type="text/css">
<!--
h1 {
  text-align: center;
  color: rgb(20,152,12);
}
div.entete {
  position: fixed;
  z-index: 1;
  background-color:aquamarine;
```

positionnement fixe du bandeau haut II

```
  top : 0px;
  height: 100px;
  width: 100%;
  border: red 1px solid
}
div.principale {
  position: absolute;
  top : 100px; /* sous l'entete */
  border: red 1px solid
}
div {
  float: left;
  border: blue 1px solid
}
-->
</style></head><body>
```

positionnement fixe du bandeau haut III

Dans le corps du document, on aura une structure en division telle que ce qui suit :

```
<div class="entete">
<h1>Titre de niveau 1</h1>
</div>

<div class="principale">

<div>
<p>hello World ! abcde abcde abcde abcde abcde
</div>

<div>
<p>hello World ! abcde abcde abcde abcde abcde
</div>
...
</div> <!-- principale -->
```

positionnement fixe du bandeau haut IV

Plan

Localisation des styles I

Les styles peuvent être définis de différentes façons et avec une priorité **décroissante** :

en ligne à éviter car le style est un attribut de balise et la séparation contenu et forme n'est plus assurée ;

dans l'en-tête comme dans le premier exemple, l'élément style peut être placé dans l'élément `head` du document html ; sa portée concerne **seulement** le document où il est défini ;

dans un autre fichier de style d'extension `.css` qui sera lié aux documents html par un élément `link`. C'est la meilleure méthode ! On peut également lié le document html à plusieurs fichiers de styles : en cas de conflit, c'est le dernier fichier lié qui est prépondérant ;

Exemple de liaison à deux fichiers de style

Localisation des styles II

```
<!doctype html>
<html lang="fr"><head><meta charset="utf-8" />
<title>Fichier CSS externe</title>
<link rel="stylesheet" type="text/css" href="css3.css">
<link rel="stylesheet" type="text/css" href="css32.css">
</head>
```

Supposons que dans le fichier `css3.css`, on a `h1 {color: red;}` et que dans le fichier `css32.css`, on a `h1 {color: blue;}`. Alors, les éléments `h1` seront bleu.

Plan

Sélecteur I

- Le sélecteur définit les éléments affectés par les propriétés
- les sélecteurs simples tels que `p` ou `div` ou `h1` permettent de désigner tous les éléments de cette catégorie
- on peut raffiner la sélection en utilisant les nombreux procédés suivants

Correspondance de modèle (pattern matching) I

- Des expressions régulières permettent de sélectionner certains éléments du DOM
- Ces expressions régulières utilisent des opérandes (éléments notés E, F tels que `div` ou `p` ou des sélecteurs complexes)
- et les nombreux opérateurs suivants :

  `*` correspond à tous les éléments de n'importe quel type ;

  E F correspond aux élément F descendant de E ; par exemple : `h1 a {color : red}` correspond aux liens situés dans un titre de niveau 1 qui seront en rouge ;

  E > F correspond aux éléments F fils de E ;
  `ol > li {font-weight: bolder;}` met en gras les listes numérotées mais pas les autres (ul) ;

Correspondance de modèle (pattern matching) II

E + F correspond aux F qui sont un frère suivant d'un E ; par exemple : `h2 + p {color: yellow}` met en jaune le paragraphe qui suit un titre de niveau 2 ;

E [*nom* = "*val*"] correspond aux éléments E ayant un attribut `nom` de valeur `val` ; par exemple :
`input[type="text"] {color: blue}` met les champs de saisie en bleu ; On peut ne pas mentionner la valeur afin de sélectionner tous les éléments ayant un attribut : `*[id]` sélectionne tous les éléments possédant l'attribut `id`.

Sélecteur multiple ,

Pour avoir tous les éléments de titre centrés :

```
h1, h2, h3 {text-align: center}
```

Plan

classes de style . I

Une classe de style permet d'affecter différents styles à un même élément HTML selon le contexte où il est utilisé. Par exemple, un paragraphe de résumé pourra être écrit en italique alors que les paragraphes "normaux" ne le seront pas. Pour cela, il suffit de définir un attribut `class` de l'élément à styliser.

```
p {
  font-family : monospace;
}
p.resume {
  font-style : italic;
  line-height : 80%;
}
```

Remarquons que les paragraphes de résumé héritent de la famille de police monospace. Voici l'utilisation des deux styles.

classes de style . II

```
<p class="resume">
  hello World ! abcde abcde abcde abcde abcde abcde abcde abcde abc
</p>
<p>abcde abcde abcde abcde abcde abcde abcde abcde abcde abcde
abcde abcde abcde abcde abcde abcde abcde abcde abcde abcde abcde al
</p>
```

Cette façon de procéder est très fréquente pour définir des divisions sémantiques du document avec des `div` en leur affectant des classes différentes

Classe générique I

On peut créer une classe sans l'associer à un élément html. Dans ce cas, le sélecteur est composé d'un point suivi du nom de classe. Cette classe peut être affectée à des éléments de différents types :

```
.italique {font-style: italic;}
...
<p class="italique">bla bla </p>
<h4 class="italique">titre</h4>
```

Plan

Identifiant d'élément avec le caractère dièse

`#monid` permet de sélectionner l'unique élément html qui possède cet identifiant (`<div id="monid" ...`)

```
#valider { color : yellow;}
h1#premier { font-style : italic;}
```

Plan

Pseudo-classe : I

Les pseudo-classes et pseudo-éléments permettent de sélectionner des objets ou des classes qui ne sont pas des noeuds du DOM :

pseudo-élément la première ligne d'un paragraphe, ou un élément généré par la feuille de style ;

pseudo-classe classe acquise dynamiquement ou déduite d'un parcours du DOM ;

:first-child sélectionne seulement les paragraphes premiers fils :

```
p:first-child { text-indent: 0 }
```

:link sélectionne les liens avant qu'ils aient été visités :

```
a:link { color: blue }
```

:visited sélectionne les liens après qu'ils aient été visités :

```
a:visited { color: blue }
```

Pseudo-classe : II

:focus sélectionne l'élément ayant le focus : `input[type="text"]:focus {color: red}`. Durant la frappe du texte, celui-ci est rouge ;

:hover sélectionne l'élément ayant le pointeur dessus (souvent utilisé pour les liens) : `input[type="text"]:hover {color: yellow}`. Durant le survol, le texte est jaune ;

:active sélectionne l'élément en train d'être activé (clic souris) : `input[type="text"]:active {color: black}`.

:first-line première ligne d'un paragraphe par exemple ;

:first-letter première lettre ;

:before pour générer du texte avant un certain élément :

:after pour générer un contenu (texte, image, ...) après un élément ;

Pseudo-classe : III

A noter que les 4 derniers types de sélecteurs sont des pseudo-éléments et non des pseudo-classes et qu'ils doivent donc être à la fin d'un sélecteur.

Exemples

- Le style suivant permettra de précéder chaque titre de niveau 1 d'un "Chapitre 12. " et chaque titre de niveau 2 d'un "Section 12.3. ".

```
body {
  counter-reset: chapter;  /* Crée un compteur de chapitre
  ↪  */
}
h1:before {
  content: "Chapitre " counter(chapter) ". "; /* contenu
  ↪  généré */
  counter-increment: chapter; /* chapter++ */
  counter-reset: section;   /* Crée un compteur de section
  ↪  */
}
```

Pseudo-classe : IV

```
h2:before {
  content: counter(chapter) "." counter(section) ". ";
  counter-increment: section;
}
```

- Dans cette autre exemple, le texte “Fin du body” sera affiché en fin de page.

```
body:after {
  content: "Fin du body";
  display: block;
  margin-top: 2em;
  text-align: center;
}
```

Plan

Modèle de visualisation I

Le flux des éléments du document est affiché selon un modèle de visualisation utilisant le principe des “boîtes” TeX. Les éléments peuvent être classés en deux familles :

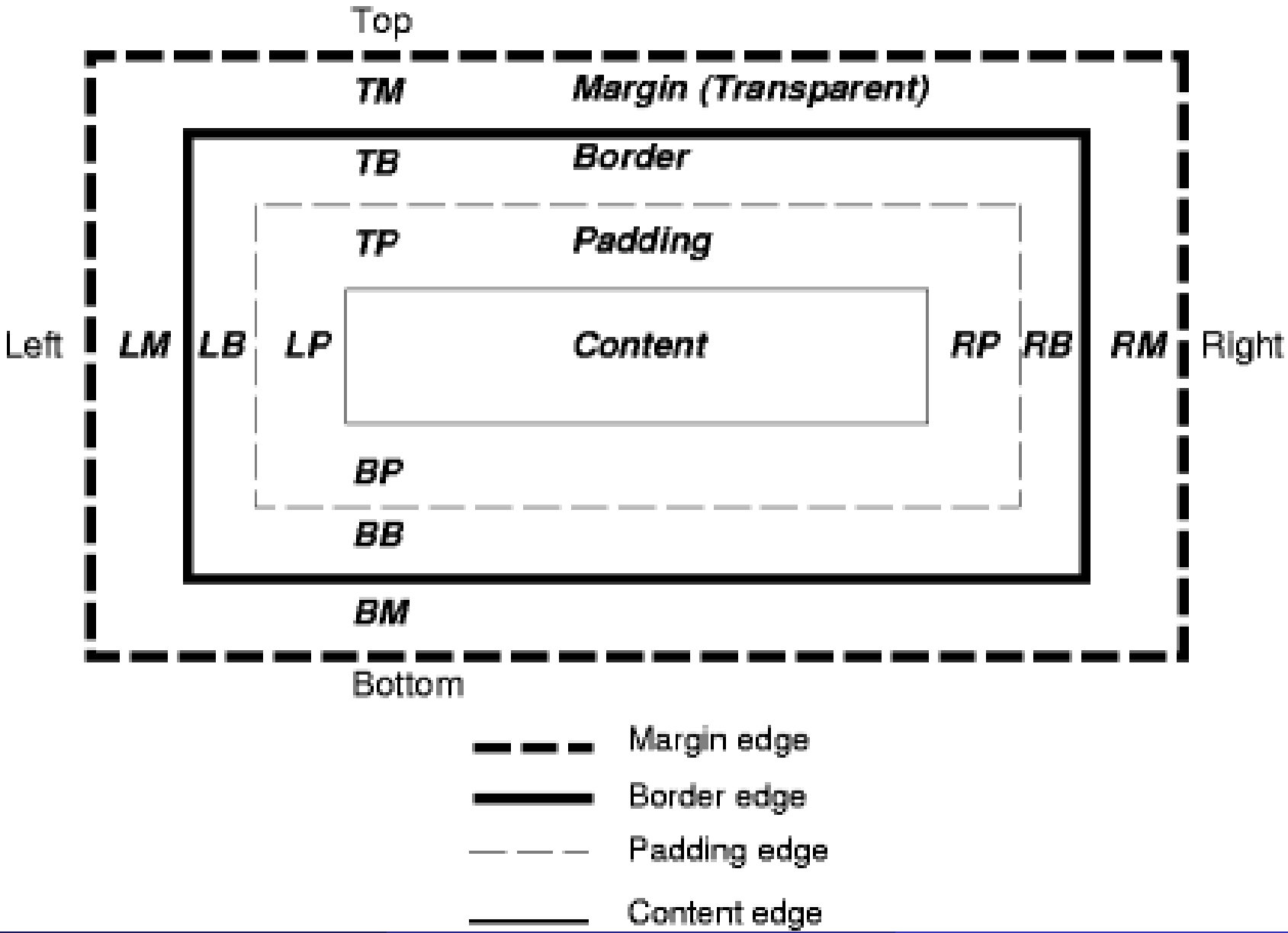
- Les éléments de type block (h1, p, ul, ol, dl, table, blockquote, etc.) ;
- Les éléments de type inline (a, img, strong, abbr, etc.) ;

Un élément de type block se différencie des éléments de type en ligne sur différents points :

- Il occupe l’entiereté de la largeur de son conteneur ;
- Il permet l’attribution de marges verticales ;
- Il permet la modification de sa hauteur et largeur ;

Tout élément peut être “reclassé” dans la famille opposée grâce à la propriété `display`. Le calcul des dimensions et de la position de ces boîtes est dynamique au fur et à mesure du chargement de la page et/ou de la dynamique interactive.

Modèle de visualisation II



Plan

La propriété css position I

- On peut sortir un élément bloc du flux lorsqu'on envisage une mise en page un peu sophistiquée en utilisant la propriété **position**
- Une fois le type de `position` fixée, ce sont les propriétés `left, right, top, bottom` qui définiront les décalages.
- **La position statique** (`position:static`) correspond simplement à la valeur par défaut d'un élément du flux. Il n'y a que peu d'intérêt à la préciser, si ce n'est dans le but de rétablir dans le flux un élément en particulier parmi une série qui serait positionnée hors du flux

La propriété css position II

- **La position fixe** (`position:fixed`) positionne l'élément par rapport à la fenêtre du navigateur. L'élément est fixé à un endroit et ne pourra se mouvoir, même lors de la présence d'une barre de défilement. En d'autres termes, la position intiale est fixée au chargement de la page, le fait qu'une éventuelle scrollbar puisse être utilisée n'a aucune influence sur le positionnement de l'élément : il ne bouge plus de la position initialement définie.
- **La position absolue** (`position:absolute`) ressemble à la position fixe sauf qu'elle positionne l'élément par rapport à son premier ancêtre ayant une position autre que statique (body s'il n'en existe pas). `fixed, absolute` permettent la superposition. Un élément bénéficiant d'une position absolue ne bougera pas de sa position initiale tant que l'une des propriétés top, bottom, left ou right n'a pas été précisée ; il s'agit d'ailleurs là d'un comportement appliquable à toutes les positions. Dans le cas où un élément est en position absolue

La propriété css position III

mais n'a pas de coordonnées (left, top, ...), il est placé à la suite comme s'il était dans le flux, mais ses sucesseurs qui sont dans le flux viennent l'écraser (superposition).

- **La position relative** (`position:relative`) permet de décaler un élément par rapport à une position de référence : celle qu'il avait dans le flux. Les éléments qui le suivent et le précèdent ne sont pas influencés par ce décalage puisqu'ils considèrent que l'élément est toujours dans le flux à sa position initiale. Attribuer à un élément une position relative peut être pratique dans les situations suivantes :
  - Servir de référent à un élément enfant positionné en absolu (rappelons qu'un élément positionné absolument grâce aux propriétés top, left, ... le fera par rapport à la fenêtre du navigateur à défaut d'avoir un parent lui-même positionné) ;
  - Bénéficier de la possibilité d'utiliser la propriété z-index pour gérer des superpositions d'éléments (propriété inopérante pour des éléments du flux) ;

Plan

La propriété float I

La propriété float existe avant tout pour répondre à un besoin typographique précis : la création d'habillages. Un habillage est une pratique courante dans le média print consistant à "enrouler" un texte autour d'un élément (graphique ou texte).
À l'instar du positionnement absolu, un élément flottant adopte par défaut la largeur qu'occupe son contenu. Le principe de base est simple : un élément flottant est ôté partiellement du flux et placé à l'extrême gauche (`float:left`) ou droite (`float:right`) de son conteneur, forçant par la même occasion tout contenu du flux qui suit à l'envelopper. Deux objets flottants dans la même direction se rangeront côte à côte, seul un contenu demeuré dans le flux qui les succède *immédiatement* initiera l'habillage.

Plan

Type de media et résolution d'écran I

Depuis CSS2, on peut spécifier une feuille de style différente selon le media d'affichage de la page web :

```
<link rel="stylesheet" media="screen" href="screen.css" type="text/
<link rel="stylesheet" media="print" href="print.css" type="text/cs
```

On peut également spécifier des propriétés à l'intérieur d'une même feuille :

```
@media print {
  #menu, #footer, aside {
    display:none;
  }
  body {
    font-size:120%;
    color:black;
  }
}
```

Type de media et résolution d'écran II

Les différents media sont : screen, handheld (mobiles), print, aural (Synthèse vocale), braille, projection, tty (police à pas fixe), tv, all.
Afin de prendre en compte, les résolutions d'écran différentes (moniteur, tablette, smartphone), CSS3 a introduit les *media queries* qui permettent de spécifier un style conditionnel en fonction de la taille de la fenêtre. Un exemple suit :

```
.precisions span {
        display: none; /* par défaut n'apparait pas */
}
@media screen and (min-width: 1024px) and (max-width: 1280px) {
        .wide {
                background: #f11a57;    /* change couleur et fond
                ↪  */
                color:#fff;
        }
        .precisions span.regle3 {
```

Type de media et résolution d'écran III

```
                display: block;          /* sauf si grand écran
                ↪   */
        }
}
```

D'autres éléments de CSS3 telles que les grilles fluides dépassent l'objectif de ce cours !

Plan

Responsive web design I

Les tailles d'écrans de visualisation se démultipliant (mobile, tablette, TV, ...), une conception adaptée consiste à prendre en compte ces différents supports afin de permettre à chaque utilisateur de pouvoir utiliser le site web. Une solution simple consiste à définir son site pour l'écran le moins disant au niveau de sa définition, par exemple en définissant de manière fixe les éléments importants de votre blog :

```
article {
  width: 300px;
  height: 600px;
}
```

Une autre solution, bien meilleure, consiste à utiliser un framework CSS tel que bootstrap qui redéfinit le css en fonction de la largeur de l'écran grâce à l'élément méta à insérer dans l'entête (head) de la page :

```
<meta name="viewport" content="width=device-width,
↪  initial-scale=1">
```

Plan

Plan

Introduction I

- Bootstrap est un framework CSS donc côté client (*frontend*) créé pour Twitter
- Il embarque également des composants HTML, JavaScript, JQuery
- Il comporte un système de **grille** simple pour organiser l'aspect visuel d'une page web "responsive"
- Il embarque des thèmes visuels, des icônes ...
- Il apporte du style pour les boutons, les formulaires, la navigation ...
- Il permet ainsi de concevoir un site web rapidement et avec peu de lignes de code (hum hum)
- Il utilise le "responsive web design" qui consiste à adapter la page Web au média utilisé.

Plan

De l'utilisation d'un framework I

Avantages

- les navigateurs ont des comportements très différents malgré leur lente convergence vers les standards. Les frameworks assurent une présentation similaire quel que soit le navigateur utilisé et une parfaite compatibilité ;
- bootsrap assure une accessibilité pour les utilisateurs handicapés ;
- les frameworks CSS font gagner du temps de développement **une fois qu'on les maîtrise** ;
- ils proposent un ensemble homogène de styles ;
- ils proposent en général une grille pour faciliter le positionnement des éléments ;
- ils offrent souvent des éléments complémentaires : boutons esthétiques, fil d'ariane, etc...

De l'utilisation d'un framework II

- la grande variété des nouveaux moyens de visualisation du web (smartphones, tablettes. . .) impose désormais la prise en compte de tailles d'écran très variées ; les frameworks CSS prennent généralement en compte cette contrainte.

Inconvénients

- temps d'apprentissage ;
- version 3 non compatible avec version 2

Plan

Caractéristiques I

- système de grille de 12 colonnes ;
- sous licence Apache, actuellement à la version 4.5 orientée mobile ;
- du code fondé sur HTML5 et CSS3 ;
- une bonne documentation ;
- une architecture basée sur Sass, un outil qui étend les possibilités de CSS ;

Plan

Installation I

- se rendre à l'url `http://getbootstrap.com` ;
- télécharger bootstrap sous forme d'un fichier zip contenant le code css compilé et minimisé ainsi que du javascript et des polices ;
- décompresser l'archive à la racine du site et renommer le répertoire en `bootstrap` (supprimer le numéro de version) ;
- insérer dans l'entête (head) des pages html ou php conforme au langage HTML5 le code suivant :

```
<link href="bootstrap/css/bootstrap.min.css" rel="styleshe
<script src="bootstrap/js/jquery.js"></script>
<script src="bootstrap/js/bootstrap.min.js"></script>
```

Il faudra bien entendu télécharger le fichier `jquery.js` dans le répertoire concerné ! En effet, la bibliothèque javascript`jquery` est utilisée par le code javascript de bootstrap. Attention à télécharger une version compatible !

Installation II

Remarques :

- Les fichiers minimisés (.min.cs ou .min.js) sont des versions compressées illisibles dans lesquelles les commentaires ont été supprimés
- Ils sont destinés à être chargés plus rapidement en mode production
- En mode développement, il est conseillé d'utiliser les versions non minimisées.
- Les CDN (*Content delivery network*) sont des serveurs qui mettent à disposition des librairies
- Il devient ainsi inutile de stocker ces librairies sur son propre serveur, il suffit de "pointer" vers eux
- L'avantage à utiliser un CDN est de diminuer l'utilisation de ressources (stockage et bande passante) sur son propre serveur mais surtout de factoriser ces ressources dans le cache du navigateur ce qui accélère le chargement des pages

Plan

La grille Bootstrap I

- découpage de la page Web en 12 colonnes de **même largeur**
- Les différents éléments de la page (banière, barre de menu, ...) seront disposés sur un nombre donné de ces colonnes
- La hauteur d'une ligne est déterminée par la hauteur de son plus gros élément
- Le nombre de colonnes affecté à un bloc (div) peut varier en fonction de la taille de l'écran
- 5 classes CSS génériques (i.e. `col-md-5`) permettent d'indiquer le nombre de colonnes (5) utilisées pour cet élément dans la grille **pour tous les écrans de largeur supérieure ou égale à la taille** (md), sauf si une autre spécification existe pour ces écrans (lg ou xl) !

Tailles d'écran :

rien pour les largeurs d'écrans $<$ 576 px : **Mobile First**

La grille Bootstrap II

-sm 576 <= largeur pour les mobiles et plus ;

-md 768 px <= largeur pour les tablettes et plus ;

-lg 992 px <= largeur pour ordinateurs portables et plus ;

-xl 1200 px <= largeur pour les grands écrans ;

- définir une mise en page de la grille spécifique pour un type d'écran, par exemple `md`
- Si l'utilisateur possède un écran `md` ou `lg`, la disposition prévue sera respectée
- Par contre, sur les écrans plus petits, un empilement vertical des éléments aura lieu afin de permettre leur lisibilité sur des mobiles par exemple
- On peut combiner différentes largeurs pour plusieurs types d'écran pour chaque élément afin de prévoir différentes disposition selon le type d'écran

2 dispositions distinctes de blocs I

Le code source HTML

```
<body>
<div class="container-fluid">
<h3>Responsive design</h3>
<p>combinaisons de dispositions : rien pour les mobiles et étendu
aux tablettes, md pour les écrans de portables et d'ordinateurs
de bureau.</p>
<div class="row">
  <div class="col-12 col-md-8">(col-12 col-md-8) plein sur
  ↪  mobile,
2/3 sur écran</div>
  <div class="col-6 col-md-4">(col-6 col-md-4) moitié sur mobile,
1/3 sur écran</div>
</div>
<div class="row">
  <div class="col-6 col-md-4">(col-6 col-md-4) moitié sur mobile,
```

2 dispositions distinctes de blocs II

```
1/3 sur écran</div>
  <div class="col-6 col-md-4">(col-6 col-md-4) moitié sur mobile,
1/3 sur écran</div>
  <div class="col-6 col-md-4">(col-6 col-md-4) moitié sur mobile,
1/3 sur écran</div>
</div>
<div class="row">
  <div class="col-6">(col-6) moitié toujours</div>
  <div class="col-4 offset-2">(col-4 avec décalage. de 2)
1/3 toujours</div>
</div></div>
```

2 dispositions distinctes de blocs III

Affichage sur grand écran

Responsive design

combinaisons de dispositions : rien pour les mobiles et étendu aux tablettes, md pour les écrans de portables et d'ordinateurs de bureau.

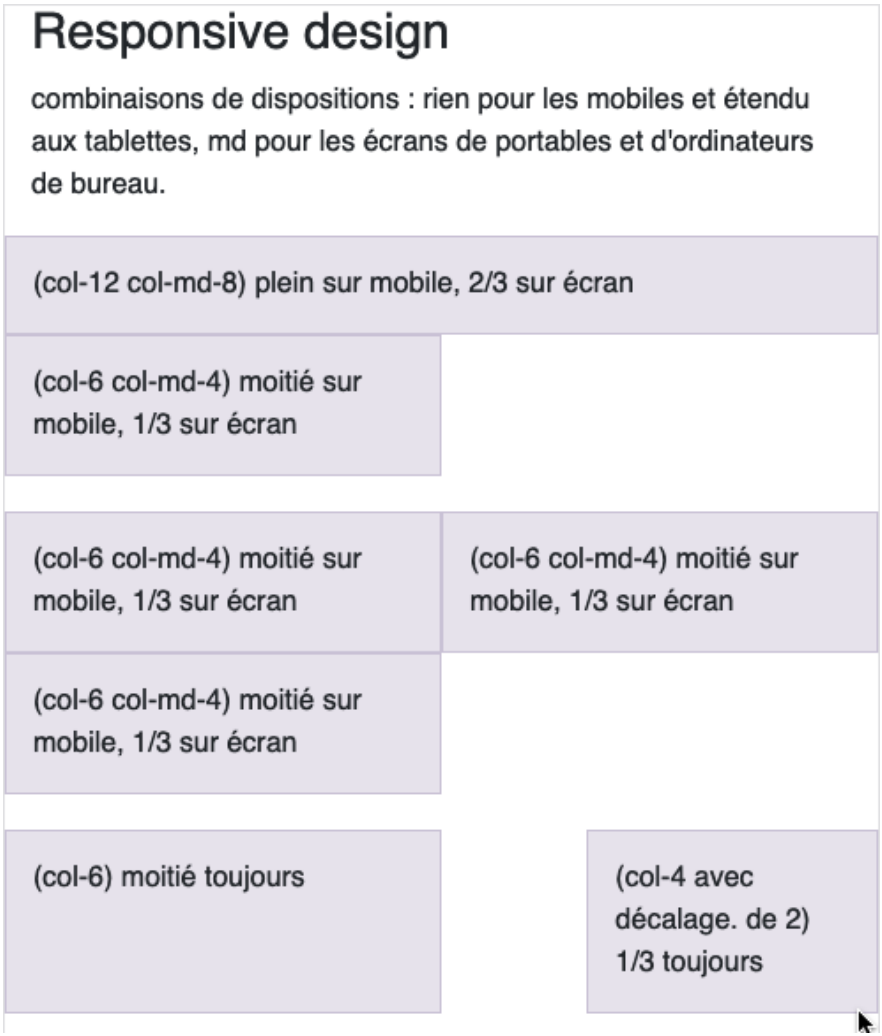
(col-12 col-md-8) plein sur mobile, 2/3 sur écran	(col-6 col-md-4) moitié sur mobile, 1/3 sur écran

(col-6 col-md-4) moitié sur mobile, 1/3 sur écran	(col-6 col-md-4) moitié sur mobile, 1/3 sur écran	(col-6 col-md-4) moitié sur mobile, 1/3 sur écran

(col-6) moitié toujours	(col-4 avec décalage. de 2) 1/3 toujours

2 dispositions distinctes de blocs IV

Affichage sur mobile



2 dispositions distinctes de blocs V

- le principe de la grille de 12 est récursif, ce qui fait qu'ont peut emboîter des blocs à l'intérieur
- Bootstrap propose sur son site des exemples de dispositions de pages
- Le site `http://www.codeply.com/` propose un éditeur graphique en ligne afin de prototyper l'apparence de son projet

Plan

Eléments de base I

Afin de créer un rendu visuel cohérent pour que tous les éléments cohabitent de façon esthétique, Bootstrap impose une certains nombre de principes :

- Tous les éléments sont englobés dans une racine `<div class="container">` fille de `body` ; Pour les applications (sans marges latérales) la classe `container-fluid` est plus adaptée ;
- l'emboîtement (row in col), le déplacement (offset), le changement d'ordre des colonnes (push) sont possibles ;
- des titres de h1 à h6 sont disponibles avec un sous-titre en ligne grâce à la classe `small` ;
- la taille de police par défaut du body est de 14px, la hauteur de ligne de 1,428 soit près de 20px ;

Eléments de base II

- diverses balises sont stylées pour les listes ul, ol, dl (description list=dictionnaire), les abbréviations (abbr), les citations (blocquote), les alignements gauche droite justifié,
- les tables html possèdent différentes classes intéressantes :

  .table-bordered une bordure simple encadre chaque cellule ;

  .table-hover la ligne est surlignée lorsque la souris survole ;

  .table-responsive emboîter une table dans une table-responsive ajoutera à ascenseur horizontal sur les petits écrans ;
- les formulaires sont divisés en plusieurs classes :

  par défaut les contrôles (input, select,) ont une largeur de 100% (formulaires destinés au téléphones mobiles) ; label et input sont emboîtés verticalement dans une div de classe `form-group` ;

  form-inline permet de placer les labels et leur contrôle en ligne ;

Eléments de base III

form-horizontal permet d'utiliser le principe de la grille afin de placer le label dans la colonne de gauche, et le contrôle dans la colonne de droite ; en cas d'utilisation avec petit écran, l'empilement reprend le dessus et l'aspect redevient celui de la classe par défaut.

Les différents contrôles HTML doivent posséder la class `form-control` et doivent être emboîtés dans une div de class `form-group`. De plus, une autre classe peut être ajoutée (en javascript ou PHP) pour indiquer la qualité de la validation des données saisies : `has-success`, `has-warning`, `has-error`. De nombreux types de boutons existent (forme, couleur) et l'on peut représenter différents éléments html tels que les liens (a href), les button, les input de type button et les input de type submit avec la même représentation graphique !

- Des menus dropdown, des groupes de boutons, ...

Eléments de base IV

- Une navigation est une liste non numérotée (ul) possédant la classe nav et une sous-classe précisant la visualisation des li : nav-tabs pour des onglets, nav-pills pour des boutons accolés. Chaque item de liste peut contenir un simple lien ou un un dropdown-toggle permettant de basculer un menu ;
- Une barre de navigation est un bloc `<nav>` possédant la classe `navbar` et offre un widget réactif (responsive) pour réaliser le menu principal d'un site. Il peut réunir un logo, des dropdown, un formulaire de recherche , un autre de connexion, ...
- un fil d'ariane (breadcrumb) affiche la hiérarchie de navigation en séparant les liens par des slash. Il est réalisé avec une liste ordonnée (ol) possédant la classe breadcrumb.

Plan

Barre de navigation, menu et formulaire I

```
<meta name="viewport" content="width=device-width,
↪  initial-scale=1.0">
<!-- Bootstrap -->
<link rel="stylesheet"
↪  href="https://stackpath.bootstrapcdn.com/...>
</head><body>
<!-- Fixed navbar -->
<div class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    <div class="navbar-header"><!-- réduction sur mobile -->
      <button type="button" class="navbar-toggle"
      ↪  data-toggle="collapse"
        data-target=".navbar-collapse"> <!-- bouton d'expansion
        ↪  du menu -->
        <span class="icon-bar"></span>  <!-- 3 tirets horizontaux
        ↪  -->
```

Barre de navigation, menu et formulaire II

```
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
    <a class="navbar-brand" href="#">Tournoi</a>  <!-- titre
    ↪  -->
</div>
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Home</a></li>
      <li><a href="#about">About</a></li>
      <li><a href="#contact">Contact</a></li>
      <li class="dropdown">
      <a href="#" class="dropdown-toggle"
      ↪  data-toggle="dropdown">Equipe <b
      ↪  class="caret"></b></a>
      <ul class="dropdown-menu">
        <li><a href="#">Créer</a></li>
```

Barre de navigation, menu et formulaire III

```
            <li><a href="#">Supprimer</a></li>
            <li><a href="#">Mettre à jour</a></li>
            <li class="divider"></li>
            <li class="dropdown-header">Inscription</li>
            <li><a href="#">Inscrire</a></li>
            <li><a href="#">Désinscrire</a></li>
          </ul>
        </li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
          <li class="active"><a href="./">Lien à droite</a></li>
        </ul>
      </div>  <!--/.nav-collapse -->
    </div>
  </div>
  <div class="container">
    <!-- Main component -->
```

Barre de navigation, menu et formulaire IV

```
  <div class="jumbotron">
    <h1>Exemple de navbar</h1>
    <p>Cet exemple de barre de menu horizontale fixe en haut de
    ↪  page permet de tester les navbar.
    <p>
      <a class="btn btn-lg btn-primary" href="#">Bouton lien
      ↪  &raquo;</a>
    </p>
  </div>
</div> <!-- /container -->
<div class="container">
<form class="form-horizontal" role="form">
  <div class="form-group">
    <label for="inputEmail1" class="col-lg-2
    ↪  control-label">Email</label>
    <div class="col-lg-10">
      <input type="email" class="form-control" id="inputEmail1"
      ↪  placeholder="Email">
```

Barre de navigation, menu et formulaire V

```
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword1" class="col-lg-2
    ↪  control-label">Password</label>
    <div class="col-lg-10">
      <input type="password" class="form-control"
      ↪  id="inputPassword1" placeholder="Password">
    </div>
  </div>
  <div class="form-group">
    <div class="col-lg-offset-2 col-lg-10">
      <div class="checkbox">
        <label>
          <input type="checkbox"> Remember me
        </label>
      </div>
```

Barre de navigation, menu et formulaire VI

```
    </div>
  </div>
  <div class="form-group">
    <div class="col-lg-offset-2 col-lg-10">
      <button type="submit" class="btn btn-default">Sign
      ↪  in</button>
    </div>
  </div>
</form>
  <!-- at the end of the document so the pages load faster -->
  <!-- jQuery ! (necessary for Bootstrap's js plugins) -->
<script src="https://code..../jquery.min.js" ...></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js..."></script>
<script src="https://stackpath.../bootstrap.min.js" ...></script>
</body></html>
```

Barre de navigation, menu et formulaire VII

FIGURE – barre de navigation fixe et formulaire horizontal sur grand écran

Barre de navigation, menu et formulaire VIII

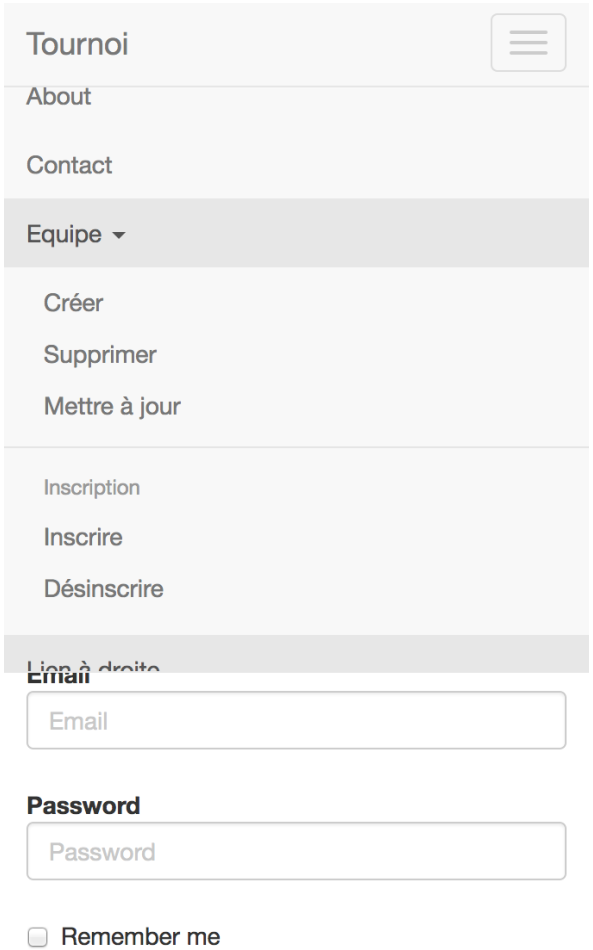


FIGURE – barre de navigation fixe et formulaire sur petit écran

Barre de navigation, menu et formulaire IX

Remarquons que dans cet exemple, la modification de la taille de l'écran induit de grandes différences d'interface du menu et du formulaire.

Plan

Plan

Introduction à PHP I

- PHP : acronyme récursif pour “PHP Hypertext Preprocessor”
- langage de script (interprété) et Open Source, spécialement conçu pour le développement d’applications web
- le script PHP est **intégré** au HTML
- la réponse à la requête est constitué du code HTML dans lequel les affichages (echo, print) effectués par le code PHP sont insérés
- La référence du langage est sur un site Web [5] qu’il faut toujours avoir en face des yeux quand on programme

Introduction à PHP II

Exemple (hello.php)

```
<html><head><title>premier prg php</title></head><body>
<?php
  echo "Bonjour le monde !";
?>
</body></html>
```

- différence avec les autres scripts CGI (Perl, C, ...) : page HTML avec du code inclus à l'intérieur :
  - pas besoin de générer le code HTML et ses entêtes
  - prototypage du squelette du site par des spécialistes graphiques
- différence avec Javascript : le code est exécuté sur le serveur
- langage extrêmement simple et fonctionnalités avancées
- l'interprète PHP peut être appelé par :
  - le serveur Web comme module ou comme CGI

Introduction à PHP III

- l'interpréteur de commandes de votre système : `php hello.php`.
- en mode **interactif** (`php -a`), une session d'interprétation s'ouvre et vous pouvez tester vos instructions

Caractéristiques I

- Différentes versions : PHP2FI, PHP3, PH4, PHP5, PHP7, PHP8
- Nous traiterons de **PHP7** (PHP8 en version Beta)
- L'interprète PHP fonctionne sur le fichier interprété comme un **filtre** laissant passer de l'entrée standard vers la sortie standard tout ce qui n'est pas un bloc php (entre)
- Les sorties des blocs php (`echo`, `print()`, `var_dump()`) sont insérés dans le code HTML à la place du bloc
- si plusieurs *blocs* php existent, les définitions du premier bloc sont utilisables dans tous les autres (unique contexte d'exécution)
- On peut utiliser `<?= $i ?>` quand on veut juste afficher la valeur d'une variable

Caractéristiques II

Exemple

```
<html><head><title>deuxieme prg php</title></head><body>
<?php
$i=123;
?>
bla bla bla
<?php
echo "la variable i vaut : $i";
?>
</body></html>
```

Le fichier d'extension `.php` doit être lisible par le serveur HTTP (`chmod a+r hello.php`). Le droit d'exécution n'est pas nécessaire.

Plan

Structure du langage I

- Au niveau syntaxique, le langage PHP ressemble fortement au langage C++
- Différence essentielle : il est typé dynamiquement
- Toutes les variables doivent être **préfixées par un $** : c'est la cause principale d'erreur syntaxique. Il faut donc répéter que TOUTES LES VARIABLES DOIVENT ÊTRE PRÉFIXÉES PAR UN $!
- Originellement, la structure d'un script était un ensemble de fonctions
- Depuis PHP5 un modèle objet à classe permet de programmer par objets

Types scalaires I

Les types scalaires sont :

- integer avec des littéraux tels que : 10, -454, 0777, 0xABCD ;
- float avec des littéraux tels que : 1.24, -1.2e3, 5E-6 ;
- string avec des littéraux tels que : 'hello world', `"hello $i world"` ; Entre guillemets, les variables scalaires sont substituées par leur valeur (pas entre apostrophes) ! De nombreuses fonctions strxxx existent !
- boolean : true et false (insensibles à la casse) ;
- null : la valeur spéciale NULL représente l'absence de valeur (insensible à la casse).

Typage dynamique I

Une variable n'est pas typée statiquement mais dynamiquement au fur et à mesure de ses affectations successives. La conversion de type est automatique et dépend du contexte.

Exemple

```php
<?php
$x = "0";  // $x est une chaîne de caractères (ASCII 48)
$x += 2;   // $x est maintenant du type entier (2)
$x = $x + 1.3;  // $x est maintenant du type double (3.3)
$x = 5 + "10 roses"; // $x est du type entier (15)
?>
```

On peut forcer le type d'une variable avec :

- la fonction settype() ;
- ou le transtypage à la C : `$x = (int) 5.6;`

Typage dynamique II

On peut tester le type d'une variable avec :

- `bool is_int(mixed $var)`
- `bool is_string(mixed $var)`
- ...

le type `mixed` signifie un type quelconque (string ou array ou object)

Les variables I

- Les variables sont créées dynamiquement dans le contexte d'exécution au fur et à mesure de l'exécution du script PHP
- La portée d'une variable peut être **locale** à la fonction où elle est définie ou **globale** lorsqu'elle définie dans le script en dehors de toute fonction
- Les variables globales ne sont pas accessibles simplement dans les fonctions (différent de C++) : Il faut, dans la fonction, utiliser la notation : `$GLOBALS['glob']` ou la déclaration : `global $glob,$g2;` pour y accéder en lecture et écriture
- `$GLOBALS` est le tableau **superglobal** des variables globales

Les variables II

- Un tableau superglobal est une variable globale accessible directement dans les fonctions (comme en C++). Un petit nombre de tableaux superglobaux sont **prédéfinis** en PHP : `$_GET`, `$_POST`, `$_SESSION`, `$_COOKIE`, `$_ENV`, ... Il n'est pas possible d'en définir d'autres ;
- La fonction `isset($x)` permet de tester l'existence d'une variable ce qui s'avère primordial dans le cadre de la soumission de formulaire HTML pour savoir si un paramètre a été fourni
- On peut aussi utiliser `empty($x)` qui teste la non-existence ou la vacuité de `$x`.
- Pour supprimer une variable de son contexte, il faut utiliser la fonction `unset($x)`

Évaluation et comparaison I

- Dans les structures de contrôle (if, while, ...), les comparaisons entre variables de différents types sont réalisées par des coercitions (cast) automatiques nombreuses et parfois déconcertantes : `false==0`, `""==0`.
- On peut utiliser l'opérateur `===` qui teste l'égalité de type ET de valeur
- L'opérateur `!==` teste la différence de type ou de valeur
- En cas de doute, il est préférable d'utiliser l'égalité ou la différence triple

Constantes I

Comme en C, les constantes sont des macros, elles sont définies par la fonction `define()` et utilisées **sans $** !

```
define("REP_TELECHARGEMENT", "Telechargement/");
define("TAILLE_MAXI_FICHIER", 1000000);
echo TAILLE_MAXI_FICHIER;
```

Le type tableau associatif I

- le type tableau associatif (array) est constamment utilisé, il faut donc bien comprendre son fonctionnement
- Un tableau PHP est une association (Map) de couples (clé, valeur)
- Les clés sont uniques et sont des entiers ou des chaînes
- Les valeurs sont quelconques (type et valeur)
- Si l'on a besoin d'un tableau à la C indicé par des entiers débutant à 0, il faut considérer ces indices comme des clés.
- Lorsqu'on ajoute des valeurs sans clé dans le tableau, la clé est calculée comme un entier immédiatement supérieur à la plus grande clé entière du tableau !
- **Attention**, l'ordre chronologique des ajouts est conservée lorsqu'on parcourt (`foreach`) le tableau.
- Création d'un tableau vide : `$t=array();` ou `$t2=[];`

Le type tableau associatif II

- Ajout d'une valeur en fin de tableau : `$t[]="push me !";`

Exemple (tableau.php)

```
<html><head><title>tableaux php</title></head><body>
<?php
$tab=array("un"=>123, 2=>"deux", "abc"=>12.3);
$tab[]="toto";$tab[1]="un";
echo "taille tab : ".count($tab)."<br/>\n";
foreach($tab as $c=>$v){echo "$c => $v <br/>\n";}
?>
</body></html>
```

Le type tableau associatif III

Exemple (Affichage de tableau.php)

```
<html><head><title>tableaux php</title></head><body>
taille tab : 5<br/>
un => 123 <br/>
2 => deux <br/>
abc => 12.3 <br/>
3 => toto <br/>
1 => un <br/>
</body></html>
```

Quelques fonctions utiles sur les tableaux :

Le type tableau associatif IV

`count($t)`	retourne le nombre de couples
`unset($t['deux'])`	supprime la clé et la valeur
`isset($t['deux'])`	teste si défini
`empty($_POST['valider'])`	teste si non défini ou vide ou 0
`$t[]='nouveau'`	ajoute un couple à la fin (push)
`var_dump($t)`	affiche récursivement le tableau
`$compact=array_values($t)`	compacte t dans compact indicé de 0 à n-1
`foreach($t as $cle=>$valeur){}`	parcours du tableau
`bool in_array ("toto",$t)`	recherche de valeur
$t=array(array(1,"toto"=>3), array("tutu",1=>2))	multi-dimensionnel
`bool array_xxx($t)`	innombrables fonctions avec xxx valant sort, merge, diff, ...

Le type tableau associatif V

Lorsque l'on veut insérer dans une chaîne, une valeur présente dans un tableau, il faut utiliser les accolades :

```
echo "bonjour {$tab['michel'][5]}";
```

Expressions et Opérateurs I

Les opérateurs ont une priorité ("precedence") et une associativité. Voici une liste non exhaustive avec priorités décroissantes :

++, --	inc et déc : `++$x;$y--;`
unaires	~ (not binaire), - (moins arithmétique), ! (not logique) ;
arithmét. et concat.	(*,/,%) puis (+, -, .) (point de concaténation) ;
décalages	(<<,>>)
comparaison	(<,<=,>=,>) puis (==, !=, ===, !==) (comparaison typée). ATTENTION : fonctionne avec les chaînes mais attention aux chaînes numériques : `"1"==" 01.0"` mais `"1"!==" 01.0"`
bit à bit	(&, ^, |)
logiques	(&&, ||)
$i==$j ? 1 : 2	expression conditionnelle

Expressions et Opérateurs II

affectation (=, .=, +=, -=, *=, /=, %=)

logiques (and, xor, or) moins prioritaire que les affectations, utilisé après les appels risqués :
$x=danger() or die("erreur dans la fon danger !";)

Exemple (comparaison.php)

```
<html><head><title>comparaison</title></head><body>
<?php
if (0=="" && null==false && $tab==0 && "2"=="02"){
  echo "BIZARRE";
}
?>
</body></html>
```

affiche : BIZARRE

Expressions et Opérateurs III

Exemple (la fonction strpos())

La fonction `strpos($chaine, $facteur)` retourne false si le facteur est absent, la position comprise entre 0 et `strlen($chaine)-1` si le facteur est trouvé. Le test `if(strpos('abc','ab')){...` est incorrect car `false==0` ! Il faut donc absolument faire le test suivant :

```
if (strpos('abc','ab') !== false) {....
```

Certaines particularités historiques sont encore présentes :

- $$s valeur de la variable dont le nom est dans s ; on ne peut pas tripler le $!
- eval("...") évaluation d'une chaîne : TRES dangereux !
- commande syst. `` `ls` ``

Expressions et Opérateurs IV

L'affect. de chaîne longue (heredoc) est possible :

```
$s=<<<FIN
bla bla..
etc
FIN; // en début de ligne et avec un ";"
```

Structures de contrôle I

Les coupures de lignes sont traitées comme les espaces. Les structures de contrôle classique à la C existent en PHP :

- alternative `if (expr) {instons1} else {instons2}`
  - choix `if (e1) {ins1} elseif (e2) {ins2} else {ins3}`
  - choix `switch ($i) {case e0: instons0; break; case e1: instons1; break; default: instons; }`
- répétitives
  - `while (e) {instons}`
  - `do { instons } while (e)`
  - `for (exp-init; exp-cond; exp-itér) { instons }`
  - ruptures possibles : `break;`, `continue;`
- parcours de tableau
  - `foreach($tab as $val) { instons utilisant $val}`

Structures de contrôle II

- `foreach($tab as $cle => $val) { instons utilisant $cle et/ou $val }`

inclusion `include 'monfic.php';` inclusion de fichier

inclusion unique `include_once 'monfic.php';` inclusion au plus une fois !

require, require_once même effet que include sauf qu'en cas d'inexistence du fichier requis, une erreur fatale est envoyée (seulement warning pour include)

Fonctions I

- Un nombre impressionnant de fonctions prédéfinies !
- Passage des paramètres scalaires ou tableaux **par valeur** (par référence si précédé de & dans la déclaration)
- Les arguments scalaires peuvent avoir une valeur par défaut (optionnels) et être en nombre variable
- Les arguments ayant une valeur par défaut doivent être à droite de ceux n'en ayant pas
- Les valeurs de retour peuvent être scalaire ou tableaux
- On peut supprimer l'affichage des erreurs produites par l'appel à une fonction f par : `@f($i)`

Fonctions sur les tableaux I

int count($tab) taille du tableau

int array_push($pile,$elem) empile à la fin et ret la taille

mixed array_pop($pile) dépile le sommet

int array_unshift($fifo,$prem) ajoute au début

mixed array_shift($fifo) retire le premier

array array_values($asso) resp. `array_keys`

void shuffle($tab) mélange les valeurs

Fonctions sur les tris de tableau I

void sort($tab) tri croissant (décroissant avec rsort) selon les valeurs

void ksort($asso) tri croissant selon les clés (krsort)

void asort($asso) tri croissant selon les valeurs (arsort)

void usort($tab, moncmp) tri croissant selon la fon de cmp définie par l'utilisateur

uksort, uasort tri utilisateur pour les asso

Fonctions sur les chaînes I

De très nombreuse fonctions dont voici les principales. $s est supposée être une chaîne.

int strlen($s) taille

int strcmp($s1,$s2) comparaison -1, 0, 1

string substr($s, 2, 10) sous-chaîne à partir de 2, de taille 10

string strstr($s,$facteur) ret tout s à partir de la 1ère occurrence de facteur

array split(' ;',$s,10) ret un tableau des 10 (maxi) premiers champs séparés par des ;

string join(' ;',$tab) ret la chaîne constituée des valeurs séparées par ;

string trim($s) retire les blancs de début et de fin de chaîne. Blancs : `\n,\r,\t,\v,\0,espace`

string chop($s) supprime les blancs de fin de chaîne

Fonctions sur les chaînes II

string ltrim ($s) supprime les blancs de début

string strtolower($s) met en minuscules (resp. `strtoupper`)

string nl2br($s) remplace les `\n` par des `<br>`

Fonctions sur les expressions régulières I

correspondance `int preg_match('/^[^.]*\.(.*)$/',$s,$tabr)` met dans tabr la partie à droite d'un point dans s à partir de l'indice 1. Retourne TRUE si trouvé au moins une.

remplacement

`string preg_replace($patterns, $replacements, $stri`
retourne une chaîne obtenue par remplacement dans la chaîne `string` des facteurs correspondant aux `patterns` par les `replacements`.

```
php > echo preg_replace("/tu/", "il","tu manges");
il manges
```

Entrées/Sorties I

De nombreuses fonctions de gestion du système de fichier à la C

echo string, string ... ; affiche plusieurs arguments **chaînes**

print(string) ; affiche un arg. (**classé** dans les fons chaînes)

$desc=fopen("fic.txt","r+") ; ouverture en lecture et écriture. Modes (r, w (création ou raz), w+ , a (append), a+).

$d=fopen("c : \\data\\info.txt","r"); Windows

$d=fopen("ftp ://user :password@example.com/", **"w") ;** ouverture de session ftp

fclose($desc) fermeture

bool feof($desc) test la fin

$ligne=fgets($desc, 4096) lecture de la ligne (maxi 4096 octets)

$ligne=readline("un entier SVP") ; ne fonctionne pas sur le Web !

$car=fgetc($desc) ; lecture d'un car ;

fputs($desc,"chaine") ; écriture d'une chaîne ;

Fonctions utilisateur I

La référence en avant est possible.

définition
```
function f($arg0, $arg1="toto"){instons; return
        array(1, 4);}
```

pointeur de fonction
```
$pf='f';list($i,$j)=$pf(1,2);
```

Diverses fonctions prédéfinies I

echo	exp1, exp2, ... ; affiche les expressions ; ATTENTION, echo n'est pas une fonction mais une structure de contrôle !
print(exp1)	affiche une expression ;
var_dump($var)	affiche la variable scalaire ou tableau ou objet récursivement (débogage) ;
void exit()	termine le script
$l=system("ls") ;	exécute une commande
void die ("message ")	affiche le message puis termine
void eval("instons PHP")	il faut échapper les caractères spéciaux : `eval('$' . f($i) . '= "toto";')` ; Ne pas oublier le " ;"

Tableaux super-globaux I

Des tableaux super-globaux sont prédéfinis et permettent d'accéder à diverses informations.

$GLOBALS Contient une référence sur chaque variable qui est en fait disponible dans l'environnement d'exécution global. Les clés de ce tableau sont les noms des variables globales. Ainsi, `$GLOBALS['x']` permet de connaître ou d'affecter la variable globale `$x`.

$_SERVER Les variables fournies par le serveur web, ou bien directement liées à l'environnement d'exécution du script courant, notamment `$_SERVER['PHP_SELF']` qui est le chemin du script en cours d'exécution par rapport à la racine web (sans les paramètres GET) ;

$_GET Les variables fournies au script via la chaîne de requête URL.

Tableaux super-globaux II

$_POST Les variables fournies par le protocole HTTP en méthode POST.

$_COOKIE Les variables fournies par le protocole HTTP, dans les cookies.

$_FILES Les variables fournies par le protocole HTTP, suite à un téléchargement de fichier.

$_ENV Les variables fournies par l'environnement

$_REQUEST Les variables fournies au script par n'importe quel mécanisme d'entrée dans un certain ordre et qui ne doivent recevoir qu'une confiance limitée. Note : lorsque vous exécutez un script en ligne de commande, cette variable ne va pas inclure les variables argv et argc. Elles seront présentes dans la variable $_SERVER. php.ini : `variables_order = "EGPCS"`

Plan

Gestion de formulaire I

Les champs de **formulaire HTML** sont accessibles via les tableaux superglobaux $_GET ou $_POST selon la **méthode** utilisée par le formulaire. Par exemple :

```
<form name='F' method='get'>
<input type='text' name='nom'>
<select multiple name="biere[]">
 <option value="blonde">je préfère la bière blonde</option>
 <option value="brune">je préfère la bière brune</option>
</select>
<input type='submit' name='bouton' value='ok'/>
</form>
<?php if (!empty($_GET['biere']))
        var_dump($_GET['biere']);
?>
```

Après saisie et validation, l'url suivante est requise :

Gestion de formulaire II

`essai.php?nom=toto&biere[]=blonde&biere[]=brune&bouton=ok`

L'affichage suivant aura lieu :

```
Array ( [0] => blonde [1] => brune )
```

Remarques

- si l'attribut `action` n'est pas défini, l'url courante est à nouveau requise (avec les nouveaux paramètres !) ;
- les clés du tableau `$_GET` sont les valeurs des attributs HTML `name` des champs de saisie (input) ;

Gestion de formulaire III

- si la méthode est `get`, les paramètres passés par le formulaire lors de la soumission sont visibles dans la "query string" : la chaîne de requête est la partie d'URL commençant par un point d'interrogation et finissant à la fin de l'URL. Le séparateur des paramètres est le `&` et chaque paramètre est de la forme : `name=value`. L'URL est visible et modifiable dans la barre d'adresse du navigateur ce qui permet le débogage mais aussi l'accès visuel (password) !
- si la méthode est `post`, les paramètres sont passés dans la requête HTTP sans être visibles dans la barre d'adresse (sécurité).
- lorsqu'on souhaite que PHP recueille une liste de valeurs associée à un nom (i.e. `biere`), il faut suffixer ce nom par `[]` **dans le HTML**, sinon c'est la dernière valeur uniquement qui sera prise en compte ! Ceci est valable pour les listes à sélection multiple (select) mais aussi pour les cases à cocher. Dans le code PHP, on ne met pas les crochets (`$_GET['biere']`) !

Gestion de formulaire IV

Exercice (Multiplication)

Ecrire :

- une page HTML “Multiplication” contenant un formulaire ayant deux champs de saisie numériques x et y et un bouton `Multiplier !` ;
- un script php appelé par ce formulaire et qui affiche le résultat de la multiplication $x * y$;

Gestion de formulaire V

Solution (multiplication.html)

```
<html><head><title>Multiplication</title></head><body>
<h1>Multiplication</h1>
<form action="multiplication.php" method="get">
X<input type="number" name="x" size="10"><br>
Y<input type="number" name="y" size="10"><br>
<input type="submit" value="Multiplier !" name="mult">
</form>
</body></html>
```

Gestion de formulaire VI

Solution (multiplication.php)

```
<html><head><title>Multiplication</title></head><body>
<h1>Multiplication</h1>
<?php
if (isset($_GET['x']) && isset($_GET['y'])){
  echo "Résultat {$_GET['x']} * {$_GET['y']} = ",
  ↪  $_GET['x']*$_GET['y'], " !";
}
?>
</body></html>
```

Exercice (Unique fichier)

Réécrire le même exercice en un seul fichier !

Gestion de formulaire VII

Solution (mult.php)

```
<html><head><title>Multiplication</title></head><body>
<h1>Multiplication</h1>
<?php
if (isset($_GET['mult']) && isset($_GET['x']) &&
↪  isset($_GET['y'])){
  echo "Résultat {$_GET['x']} * {$_GET['y']} =
  ↪  ".$_GET['x']*$_GET['y']." !<br/>";
  echo "<hr/> Nouvelle Multiplication :<br/>";
} ?>
<form action="" method="get">
<label for="x">X</label><input type="number" name="x" id="x"
↪  size="10"><br>
<label for="y">Y</label><input type="number" name="y" id="y"
↪  size="10"><br>
<input type="submit" value="Multiplier !" name="mult">
</form></body></html>
```

Plan

Plan

Session PHP I

- HTTP n'étant pas un protocole orienté connexion, chaque nouvelle requête semble être la première
- Depuis PHP4, Les **sessions** PHP permettent de conserver de l'information **du côté serveur** sur la suite de requêtes émises par le même client (navigateur)
- Les variables enregistrées dans le tableau super-global `$_SESSION` peuvent être de type **quelconque**
- Pour les variables de type tableau ou objet, elles seront sérialisées et désérialisées automatiquement dans le fichier de session côté serveur (copie profonde)
- Attention cependant à définir la classe d'un objet enregistré en session **avant** l'appel à `session_start()`

La communication des identifiants de sessions est réalisée :

Session PHP II

- soit par cookie ce qui est le plus simple ;
- soit par URL de manière quasi-transparente pour le programmeur.

Par défaut, la durée de vie du cookie de session (0) est égale à la durée de vie du navigateur.

bool session_start() démarre une session ; doit être réalisé en tout **début de script** avant tout en-tête !

string session_id() retourne l'identifiant de session qui est une suite de chiffres hexadécimaux.

$_SESSION['z']="contenu" ; ajoute une variable de session ;

echo $_SESSION['z'] ; affiche le contenu d'une variable de session ;

bool session_is_registered("mavar") teste si $mavar a été sauvé dans la session ;

bool session_unregister("x") supprime la variable de session x ;

Session PHP III

$_SESSION=array() ; réinitialise la session !

bool session_destroy() supprime toutes les données de la session ; le cookie de session sera supprimé dès la prochaine page.

bool session_register("x", "y", ...) démarre une session si pas encore fait et y enregistre les variables $x et $y ;

Session PHP IV

Session sans Coockie

Afin de gérer les sessions de manière transparente, le script PHP doit tenir compte du fait que le navigateur client peut accepter ou refuser les cookies. Pour ce faire, il suffit d'indiquer dans chaque référence interne du site (href d'ancre, action de formulaire, ...), la constante `SID` comme paramètre de l'URL afin de transmettre l'identifiant de session. En effet, la constante `SID` a comme valeur :

- '' chaîne vide lorsque le navigateur accepte les cookies ;
- 'PHPSESSID=0c92dbd...51ff2' lorsque le navigateur ne les accepte pas ;

Ainsi dans un formulaire, l'attribut action devra avoir la forme suivante :

```
action="<?php echo "{$_SERVER['PHP_SELF']}".(strlen(SID)?'?'.S
```

L'exemple suivant illustre les sessions avec un historique de multiplications.

Multiplication avec mémorisation des résultats dans une session I

```php
<?php session_start(); ?>
<!doctype html><html lang="fr"><head><meta charset="utf-8" />
<title>Multiplication et session</title></head><body>

<h1>Multiplication et session</h1>
<?php
if(isset($_SESSION['historique'])){ // var_dump($_SESSION);
?><h2>Historique des multiplications</h2>
<?php
  foreach($_SESSION['historique'] as $tab){ // [['x'=>2, 'y'=>3,
  ↪  'r'=>6], ...]
    echo "{$tab['x']} * {$tab['y']} = {$tab['r']}<br/>\n";
  }
  echo "<hr/>\n";                    // pour délimiter l'historique
} else {                            // début de session
```

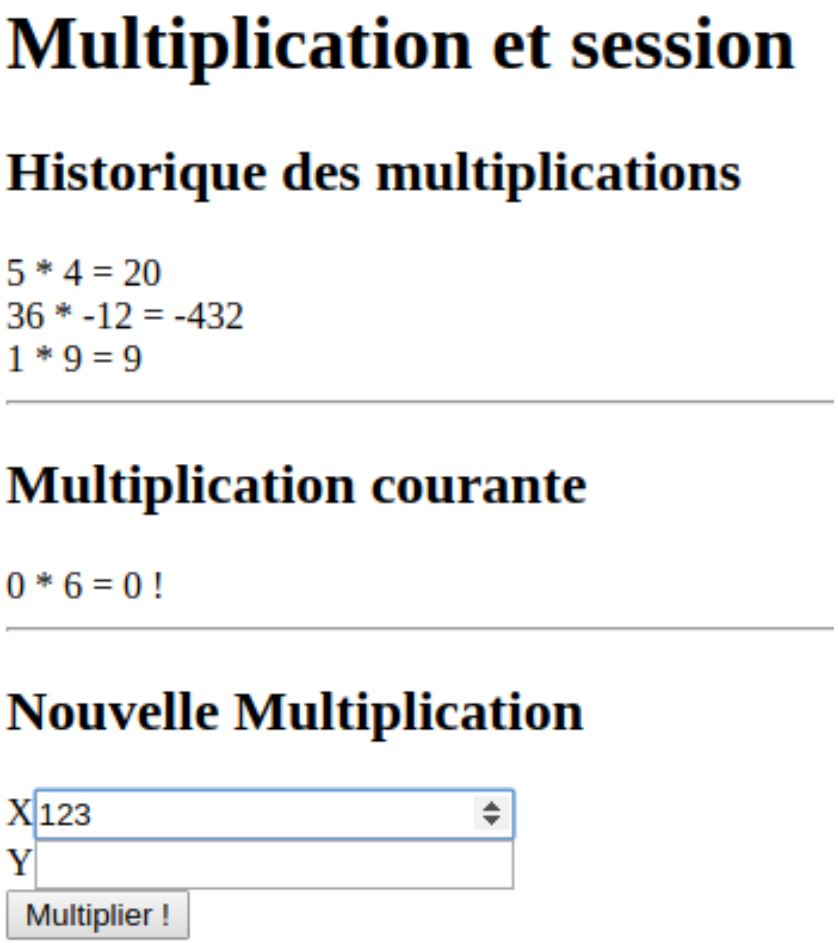
Multiplication avec mémorisation des résultats dans une session II

```
  $_SESSION['historique']=array(); // init tableau de tableau
}

if (isset($_GET['mult'])){                    // nouvelle mult
?><h2>Multiplication courante</h2><?php
  echo " {$_GET['x']} * {$_GET['y']} = ".$_GET['x']*$_GET['y'].
    " !<br/>";
  $tab=array('x'=>$_GET['x'], 'y'=>$_GET['y'],
  ↪  'r'=>$_GET['x']*$_GET['y']);

  $_SESSION['historique'][]=$tab;                 // ajout dans
  ↪  l'historique
  echo "<hr/>";
}
?>
```

Multiplication avec mémorisation des résultats dans une session III

```
<h2>Nouvelle Multiplication</h2>
<form action="<?php echo
↪  "{$_SERVER['PHP_SELF']}".(strlen(SID)?'?'.SID:''); ?>"
      method="get">
X<input type="number" name="x" size="10"><br>
Y<input type="number" name="y" size="10"><br>
<input type="submit" value="Multiplier !" name="mult">
</form>
</body></html>
```

Multiplication avec mémorisation des résultats dans une session IV



Multiplication avec mémorisation des résultats dans une session V

Si les cookies ne sont pas acceptés dans `php.ini`, voici le contenu de la barre d'adresse :

`.../multsession.php?PHPSESSID=1d5192e149789a9a52b10f948bbf6843`

Que mettre comme variable de session ? I

- Si l'on utilise une BD pour stocker les informations, il est souhaitable de ne mettre en session qu'un identifiant afin de récupérer à chaque fois les informations courantes
- Si l'on stocke un objet, il faut comprendre que tous les objets référencés par cet objet sont représentés dans la variable de session (faire un `var_dump`)
- Aussi, les mises-à-jour dans la BD d'objets référencés peuvent devenir invisibles depuis l'objet de session !
- Si l'on n'utilise pas de BD, alors toute l'information étant dans l'objet de session, il conviendra de le mettre à jour.

Session et Objets I

- Si l'on souhaite mettre un objet en variable de session, il faut que la définition de la classe soit chargée avant le début de session (`session_start()`)
- Ceci peut se faire par un `include` classique ou bien par l'auto-chargement (`autoload`) qui permet de charger le fichier définissant la classe à la demande

```
// classLoader.php
function __autoload($class_name) {
    include 'MesClasses/' . $class_name . '.php';
}

// testClass.php dans le rép MesClasses
<?php
class testClass {
  private $prop1;
```

Session et Objets II

```
  function __construct($propValue) {
    $this->prop1 = $propValue;
  }

  function showProp() {
    return $this->prop1;
  }
}
?>

// page1.php
<?php
require_once('classLoader.php');
session_start();
$_SESSION['testObj'] = new testClass('foo');
echo '<a href="page2.php">Go to page 2</a>';
```

Session et Objets III

```php
?>

// page2.php
<?php
require_once('classLoader.php');
session_start();
echo $_SESSION['testObj']->showProp(); // displays foo
?>
```

Plan

Cookies I

- Un cookie est stocké du côté client par le navigateur et contient un couple `nom=valeur` ainsi qu'un domaine, un chemin et une date d'expiration
- Quand le client envoie une requête à une URI d'un domaine et d'un chemin dont il possède des cookies, tous ces cookie sont envoyés au serveur
- Un cookie a une durée de vie (expire) par défaut égale à la durée de vie de la session
- Pour accepter/afficher les cookies sur `Firefox`, utiliser le menu Outils, Option, Vie Privée, Accepter/Afficher les cookies.

Fonctions PHP :

Cookies II

int setcookie('x','Hello !',int expire, string path, `string domain, int secure`) doit être exécutée avant toute chose et permet de positionner la variable $x avec la valeur 'Hello !' ; attention le cookie ne sera renvoyé au serveur qu'à la prochaine requête ;

$_COOKIE['x'] accès à une variable de cookie ;

setcookie('x','',time()-1000) efface la variable x du cookie ;

setcookie('y',$value,time()+3600) ; expire dans une heure ;

setcookie('x','345') ; expire en fin de session c'est-à-dire lorsque le navigateur sera fermé (expire=0) ;

Cookies de session PHP I

- Les variables de session sont conservées côté serveur
- Du côté client, si les cookies sont acceptées, une variable de cookie, généralement nommée `PHPSESSID` est présente
- Elle contient un identifiant qui est envoyé au serveur à chaque requête ce qui permet de l'associer à la session correspondante
- La variable de configuration `session.cookie_lifetime` spécifie la durée de vie du cookie en secondes
- Par défaut, la valeur de 0 signifie : "Jusqu'à ce que le navigateur soit éteint"
- Cependant, une autre variable de configuration `session.gc_maxlifetime` qui vaut généralement 1440 soit 24 minutes, indique le temps maximum dont jouissent les données de session sur le serveur

Cookies de session PHP II

- Par conséquent, au bout de 24 minutes d'INACTION, les données de session seront supprimées même si le cookie de session perdure côté navigateur

Plan

Eléments de formulaires I

Dans HTML5, de nouveaux éléments de saisie sémantiques sont apparus associés à des contrôles de validité automatiques. Ils sont plus ou moins bien implémentés selon l'âge du navigateur.

form élément racine d'un formulaire contenant les attributs suivants :

- **action** est l'URI où sera soumis le formulaire (généralement un script php ou un programme cgi). Cette URI peut être absolue ou relative au répertoire courant. La norme recquiert cet attribut qui doit être une URI, or une URI ne peut être vide mais, la norme HTML4.0 admet l'exception d'attribut vide pour désigner le document courant !

Eléments de formulaires II

method soit get, soit post ; la première (get) insère les champs saisis par l'utilisateur à la fin de l'url après un point d'interrogation sous la forme `http://localhost/index.php?nom1=val1&nom2=val2` ; les noms sont les valeurs des attributs `name` des champs tandis que les valeurs sont les contenus saisis par l'utilisateur. La seconde méthode post "cache" les contenus saisis

target avec la valeur `_blank`, une nouvelle fenêtre sera ouverte, avec `_self` (par défaut) on recouvrira la fenêtre actuelle.

onsubmit code javaScript à exécuter avant la soumission ; Si le code JavaScript retourne faux, la soumission **est annulée** !

Eléments de formulaires III

name ou id : nom ou id du formulaire pouvant être utile pour le référencement des champs de saisie dans le code javaScript de vérification

label : texte préfixant le champ de saisie et lié à lui par l'attribut `for`. Indispensable pour l'accessibilité

input : élément **vide** définissant un champ de saisie ayant :

- un attribut **name**, le nom du champ evoyé comme nom de paramètre lors de la soumission
- un attribut **type** permettant d'indiquer la forme du champ de saisie parmi :

  text : champ de type texte libre sur une ligne ; autres attributs : `size` la taille du champ de saisie à l'écran, `maxlength` le nombre maximum de caractères à saisir, `value` valeur par défaut

Eléments de formulaires IV

password	champ de mot de passe caché
button	bouton permettant de déclencher un script javascript ; autres attributs : `value` valeur affichée à l'écran, `onclick` code javaScript à déclencher
checkbox	case à cocher ; autres attributs : `value` valeur affectée au nom dans le cas où cette case est cochée, `checked="checked"` si on veut que la case soit cochée par défaut

Eléments de formulaires V

radio case à cocher **exclusive** ; tous les boutons radio mutuellement exclusifs doivent avoir le même attribut `name` ; autres attributs : `value` valeur affectée au nom dans le cas où cette case est cochée, `checked="checked"` si on veut que la case soit cochée par défaut

hidden champ caché n'apparaissant pas dans le formulaire ; historiquement, les champs cachés permettaient de faire transiter l'information passée de proche en proche lors de la navigation ; autres attributs : `value` ; Actuellement, le mécanisme de session est plus pratique à utiliser

Eléments de formulaires VI

submit bouton de soumission permettant d'exécuter l'`action` du formulaire ; plusieurs boutons de soumission peuvent coexister dans un même formulaire

image bouton de soumission utilisant une image comme visuel ; autres attributs : `src`, `alt`,`width`, `height`

reset réinitialisation des champs du formulaire

file pour envoyer au serveur un fichier localisé chez le client ; le formulaire doit posséder un attribut `enctype` ayant la valeur `multipart/form-data` et sa méthode doit être `post` ; Il est également possible de limiter la taille du fichier à envoyer en ajoutant un champ caché nommé

Eléments de formulaires VII

`max_file_size` et de valeur la taille maximale acceptée en octets

date HTML5 permet de saisir une date grâce à un calendrier

time HTML5 permet de saisir un horaire

datetime-local HTML5 permet de saisir un horaire et un jour

time, week, month, datetime HTML5 les dates sont retournées selon la norme RFC 3339, par exemple : `1985-04-12T23:20:50.52Z`

number HTML5 permet de saisir un nombre flottant dans un intervalle

```
<input type="number" name="qte"
↪  min="10" max="20" step="2"
↪  value="12">
```

Eléments de formulaires VIII

range HTML5 permet de saisir un nombre flottant grâce à un curseur ;

```
<input type="range" name="son"
↪  min="10" max="11" step="0.2"
↪  value="11">
```

color HTML5 permet de saisir une couleur au format hexadécimal #1234bb (RVB) grâce à un nuancier de couleur

```
<input type="color" name="coul"
↪  value="#1234bb">
```

email HTML5 permet de saisir une adresse email valide

url HTML5 permet de saisir une url valide

tel HTML5 permet de saisir un numéro de téléphone valide

Eléments de formulaires IX

search HTML5 champ de texte assez classique dont la sémantique est d'indiquer un champ destiné à la recherche d'information

L'attribut pseudo-booléen `disabled` permet de désactiver un champ de saisie (`input`) qui ne permet alors plus la saisie. Il ne sera pas soumis

textarea élément non vide contenant une zone de texte multi-ligne ; attributs : `name`, `rows` nb de lignes, `cols` nb de colonnes

select élément non vide contenant une liste déroulante d'options ; attributs : `name`, `size` nb de lignes visibles, `multiple` pseudo-booléen indiquant que plusieurs options sont possibles ;

Eléments de formulaires X

option élément non vide contenu dans `select` ; attributs : `value` valeur qui sera associée au `name` du `select` conteneur, `selected` de valeur `selected` indique si cette option est présélectionnée ; le contenu de l'élément option sera le texte affiché dans la liste déroulante

datalist élément contenant une liste de choix possibles mais non limitatifs. Il permet de suggérer certaines valeurs habituelles pour un champ texte (avec auto-complétion) en permettant une saisie libre. Son attribut `id` devra être référencé en tant que valeur de l'attribut `list` du champ de texte.

Eléments de formulaires XI

```
<datalist id="lprenoms">
  <option value="Pierre">Pierre Durand</option>
  <option value="Michel">Michel Meynard</option>
  <option value="Paul">
</datalist>
<label for="pre">Prénom : </label><input
↪  type="text" id="pre" list="lprenoms"
↪  name="prenom">
```

button élément non vide permettant de transformer en bouton son contenu (span, text, ...). Son attribut `type` est très important :

- `submit` (défaut) : permet de valider le formulaire contenant
- `button` : permet d'exécuter du code javascript
- `reset` : pour effacer les champs déjà saisis

Eléments de formulaires XII

Un formulaire de login

```
<form action="login.php" method="post" >
<label for="log">Nom : </label><input type="text" id="log"
↪  size="20" name="login" /><br />
<label for="pwd">Mot de passe : </label><input type="password"
↪  id="pwd" size="20" name="passwd" /><br />
<input type="submit" value="Valider">
</form>
```

Plan

La validation des données et l'aide à la saisie I

- Avant HTML5, la validation des données côté client devait être exécutée par du code Javascript
- Désormais, plus simple d'utiliser des éléments de formulaires sémantiques (`email`, `url`, `date`, `tel`, `color`, `number`, `range`) qui, par nature, sont vérifiés par le navigateur avant l'envoi du formulaire
- De plus, d'autres éléments de langages permettent d'assister l'utilisateur lors de la saisie
- **la validation côté client permet de ne pas submerger le serveur avec des données erronées ou incomplètes MAIS qu'en aucun cas, elle ne peut être suffisante**
- il faut absolument vérifier la correction des données soumises côté serveur avant de les traiter (SÉCURITÉ)

Expressions régulières I

- L'attribut `pattern` permet de définir une expression régulière qui doit correspondre au texte saisi dans un champ de type `text` lorsque la soumission est demandée
- Dans le cas où la saisie n'est pas correcte, un message "Veuillez respecter le format requis" sera affiché par le navigateur

Exemple

Exemple de validation

```
<input type="text" name="nomvar"
↪  pattern="[A-Za-z_][A-Za-z_0-9]*">
```

Renseigner un élément de saisie I

En raison des règles d'accessibilité, chaque élément de saisie doit être associé à un élément de `label` ayant un attribut `for` lont la valeur est l'`id` du champ de saisie :
Afin d'indiquer ce que doit contenir un champ, on peut utiliser des attributs tels que :

placeholder affiche un texte indicatif grisé dans le champ avant la saisie.

title affiche une info-bulle (*tooltip*) à côté du champ.

Aides à la saisie

```
<label for="netu">Numéro d'étudiant sur 8 chiffres</label>
<input type="text" name="numetu" id="netu" pattern="[0-9]{8}"
↪  placeholder="Numéro
 d'étudiant" title="par exemple : 20131234">
```

Champ obligatoire et ordre des éléments I

- l'attribut pseudo-booléen `required` d'imposer la saisie de cet élément
- A la validation, les éléments requis non saisis empêcheront la soumission et seront encadrés en rouge
- Afin qu'un élément ait le focus dès l'ouverture de la page, il suffit de lui ajouter (et seulement à lui) l'attribut pseudo-booléen `autofocus`
- Pour les autres champs, on utiliser l'attribut `tabindex` avec une valeur entière (entre 1 et n) pour indiquer l'ordre logique de saisie
- Evidemment, l'élément ayant l'`autofocus` devra avoir un `tabindex="1"`
- Pour les vieux navigateurs, afin d'assurer une même interface visuelle et la validation aux différents utilisateurs de vieux navigateurs, on peut utiliser une librairie javascript telle que `H5F` ou `Webforms2` qui se chargent de vérifier la compatibilité du navigateur courant et d'émuler les fonctionalités manquantes avec du code javascript.

Plan

Inclusion d'autres objets I

- L'élément `audio` (HTML5) permet d'inclure un lecteur audio pour écouter un son au format mp3, wav, ou ogg. Ses attributs :

src	url du fichier son
controls	pour afficher les boutons de pause, play, ...
autoplay	pour lancer le son dès le chargement
loop	pour répéter le son
preload	pour charger le son avant la page

- L'élément `video` (HTML5) permet d'inclure un lecteur video pour voir un fil au format mp4, ogg ou wbm. Ses attributs :

src	url du fichier
controls	pour afficher les boutons de pause, play, ...
autoplay	pour lancer le film dès le chargement
loop	pour répéter
preload	pour charger le film avant la page

Inclusion d'autres objets II

muted muet

poster url de l'image de remplacement pendant le téléchargement du film ou avant sa lecture

heighth, width taille de l'écran

- Des balises `sources` peuvent être utilisées afin d'indiquer différents fichiers supports en différents formats (audio ou vidéo). Cela assure une plus grande compatibilité car certains navigateurs ne supportent pas certains formats.

Exemple

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Votre navigateur ne supporte pas la balise video !
</video>
```

Inclusion d'autres objets III

- Object
  L'élément `Object` permet d'inclure un objet typé dans une page HTML. Ses attributs sont :

type	indique le type de l'objet tel que : "text/html", "application/x-shockwave-flash", "video/x-msvideo", ...
data	url du fichier tel que : `../commun.html`, `ma_video.avi`, `animflash.swf`, ...
width	et `height` indique la taille de l'inclusion

  Cet élément n'est pas vide et peut ou doit contenir des sous-éléments **paramètres** utilisés par le navigateur pour un type d'objet particulier.

Exemple

```
<param name="autostart" value="true" />
<param name="loop" value="true" />
Texte alternatif
```

Plan

HTML5 c'est beaucoup plus !

Attention, HTML5 fournit bien d'autres fonctionalités qui ne sont pas décrites dans ce cours introductif. Nous citerons par exemple :

- Canvas et son API pour dessiner et traiter des images
- drag and drop
- XMLHttpRequest level 2 ou AJAX 2
- géolocalisation
- applications offline (cache)
- WebSocket qui offre une connexion et la prog. événementielle
- Microdata pour décrire et marquer des parties de pages afin que des programme tiers (robots) puissent extraire automatiquement de l'information

Plan

Le modèle objet depuis PHP5 I

- Depuis PHP 5, modèle objet réécrit
- proche de celui du C++
- Une classe peut contenir ses propres constantes, variables (appelées "propriétés" ou "attributs"), et fonctions (appelées "méthodes")
- Une classe est introduite par le mot-clé `class` suivi de son nom
- Le constructeur se nomme `__construct()` et le destructeur `__destruct()`
- Par défaut, l'appel au constructeur de la classe parente n'est pas fait ! Pour le faire : (`parent::__construct()`)
- Les méthode magiques `__toString()` et `__clone()` sont également très utiles

Le modèle objet depuis PHP5 II

- L'affectation et le passage de paramètre objet est effectué **par référence** : il n'y pas de duplication de l'objet mais la variable affectée ou le paramètre formel contient l'adresse de l'objet

Le modèle objet depuis PHP5 III

Exemple (Une classe Point)

```php
<?php
class Point{
  public static $nbPoints=0;
  public $x,$y;
  public function __construct($px=0,$py=0){ // constructeur
    $this->x=$px;$this->y=$py;
    self::$nbPoints+=1;
  }
  public function getxy(){
    return array($this->x, $this->y);
  }
}
$p1=new Point(); // (0,0)
$p2=new Point(3,4);
list($a,$b)=$p2->getxy();
```

Propriétés, constructeur, méthodes, ... I

Propriétés

- Les attributs d'instance ou propriétés doivent être déclarés à l'aide de `public`, `protected` ou `private`
- L'ancien déclarateur `var` est obsolète et est remplacé par `public`
- Une déclaration initialisante est permise en utilisant une valeur constante
- Dans les méthodes d'instance, ces propriétés sont accessibles via la notation "fléchée" `$this->x`
- Une propriété statique (attribut de classe) peut être définie grâce au mot-clé `static`. L'accès à cette propriété à l'intérieur d'une classe se fait par `self::$nbPoints`.

Propriétés, constructeur, méthodes, ... II

Constructeur et destructeur

- Les constructeurs d'une classe se nomment `__construct(...)` et **le** destructeur `__destruct()`
- Par défaut, l'appel au constructeur de la classe parente n'est pas fait ; pour le faire : (`parent::__construct()`)
- En l'absence de constructeur explicite, toute classe admet un constructeur par défaut qui ne fait rien.

Méthodes

Les méthodes des classes peuvent être définies en tant que publiques, privées ou protégées. Les méthodes sans déclaration seront automatiquement définies comme étant publiques.

Propriétés, constructeur, méthodes, ... III

Héritage (extends), redéfinition, surcharge

- Une classe peut hériter des méthodes et des attributs d'**une** autre classe en utilisant le mot-clé extends dans la déclaration
- Il n'y a pas d'héritage multiple.
- Les méthodes et attributs hérités peuvent être redéfinis en les redéclarant avec le même nom que dans la classe parente (sauf si la classe parente a défini la méthode comme `final`) (Java)
- Il est possible d'accéder à une méthode ou un membre parent avec l'opérateur `parent::`
- Lors de la redéfinition de méthodes, la signature doit rester la même sinon PHP génèrera une erreur de niveau `E_STRICT` (**pas de surcharge**)
- Ceci ne s'applique pas au constructeur, qui accepte la surcharge (avec des paramètres différents)

Interfaces et classes abstraites I

Interfaces

- Une interface permet de spécifier quelles méthodes une classe doit implémenter
- Les interfaces sont définies en utilisant le mot-clé `interface`, de la même façon qu'une classe standard mais sans aucun contenu de méthode
- Toutes les méthodes déclarées dans une interface doivent être publiques.
- Pour implémenter une interface, l'opérateur `implements` est utilisé. Toutes les méthodes de l'interface doivent être implémentées dans une classe ; si ce n'est pas le cas, une erreur fatale sera émise
- Les classes peuvent implémenter plus d'une interface en séparant chaque interface par une virgule (extends one, implemments many)

Interfaces et classes abstraites II

Classe abstraite

- On ne peut créer une instance d'une classe définie comme `abstract`
- Toutes les classes contenant au moins une méthode abstraite doivent également être abstraites
- Pour définir une méthode abstraite, il faut simplement déclarer la signature de la méthode (précédée de `abstract` et ne fournir aucune implémentation
- Lors de l'héritage d'une classe abstraite, toutes les méthodes marquées comme abstraites dans la déclaration de la classe parente doivent être définies par l'enfant
- de plus, ces méthodes doivent être définies avec la même visibilité, ou une visibilité moins restreinte
- Par exemple, si la méthode abstraite est définie comme protégée, l'implémentation de la fonction doit être définie comme protégée ou publique, mais non privée.

Méthodes magiques et clônage I

Les méthodes :
`__construct, __destruct, __toString, __clone, __get, __set, __`
sont magiques en PHP. Vous ne pouvez pas utiliser ces noms de méthode dans vos classes, sauf si vous voulez implémenter le comportement associé à ces méthodes magiques.

toString : détermine comment l'objet doit réagir lorsqu'il est traité comme une chaîne de caractères (`echo` ou `print`) ;

clone : une fois le clonage effectué, si une méthode `__clone()` est définie, celle-ci sera appelée sur le nouvel objet ;

get, set : `void __set(string $name, mixed $value)` sera appelé lorsque l'on essaie d'affecter une valeur à un attribut inaccessible. Cela permet d'ajouter dynamiquement de nouveaux attributs (prototype).

Méthodes magiques et clônage II

call, callStatic appelé lorsque l'on essaie d'appeler une méthode inaccessible. Cela permet d'ajouter dynamiquement de nouvelles méthodes (prototype).

Clônage

Lorsqu'un objet est cloné, PHP effectue une copie **superficielle** de toutes les propriétés de l'objet. Toutes les propriétés qui sont des références à d'autres variables demeureront des références. L'opérateur `clone` est utilisé comme suit :

```
$copie = clone $objet;
```

Méthodes magiques et clônage III

Comparaison d'objets

Lors de l'utilisation de l'opérateur de comparaison ==, deux objets sont égaux s'ils ont les mêmes attributs et valeurs, et qu'ils sont des instances de la même classe. Lors de l'utilisation de l'opérateur d'identité ===, les objets sont identiques uniquement s'ils font référence à la même instance de la même classe.

Méthodes magiques et clônage IV

L'interface iterator

Cette interface permet d'utiliser la structure de contrôle `foreach` afin de parcourir tous les élément dans une itération. Voici la liste des méthodes à implémenter :

```
Iterator extends Traversable {
/* Methods */
abstract public mixed current ( void )
abstract public scalar key ( void )
abstract public void next ( void )
abstract public void rewind ( void )
abstract public boolean valid ( void )
}
```

Espaces de nom I

Comme dans les autres langages à objet, les espaces de nom sont utilisés :

- afin d'éviter les collisions entre des noms utilisés dans votre code mais également dans des fichiers inclus ;
- afin d'éviter de manipuler des noms très longs pour éviter les collisions.

Les noms d'espace ne sont pas sensibles à la casse et doivent commencer par une lettre. Ils concernent : classes, interfaces, fonctions et constantes. Ils peuvent être organisés en hiérarchie en utilisant le séparateur anti-slash \

Espaces de nom II

Exemple (Déclaration d'un espace de noms)

```php
<?php
namespace MonProjet\MaBD; // DOIT ABSOLUMENT DEMARRER le fichier
const TAILLE = 100;
class Connexion { /* ... */ }
function connecte() { /* ... */  }
?>
```

On peut utiliser un bloc pour encadrer les définitions comprises dans l'espace de nom. Un espace de nom global préexiste, il n'a pas de nom.

Espaces de nom III

Exemple (Utilisation des espaces de noms (edn))

On peut accéder aux noms d'un espace de 3 façons à rapprocher de la façon dont on référence un fichier dans un système Unix :

- nom sans séparateur (`toto`) : résolu en `ednCourant\toto`
- nom avec des séparateurs mais pas au début (`titi\toto`) est résolu en `ednCourant\titi\toto`
- nom commençant par un antislash (`\MonProjet\MaBd\TAILLE`) est absolu

Attention, dans un edn, l'accès aux fonctions PHP globales est réalisé en préfixant ces noms globaux par antislash.

```php
<?php
namespace MonProjet\MaBD; // DOIT DEMARRER le fichier
function connecte($n) { if (\strlen($n)<=10) ...  } // strlen
?>
```

Importation et aliassage de noms I

Exemple (Importation et alias avec l'opérateur use)

Après avoir défini des noms dans des espaces de nom, on peut les utiliser en les important (use) et éventuellement en les aliassant :

```php
<?php use MonProjet\Bd\Connexion as MaCo;

// les 2 lignes suivantes sont équivalentes :
use MonProjet\Bd\Connexion as Connexion;
use MonProjet\Bd\Connexion;

// importation d'une classe globale
use ArrayObject;
```

Plan

Bases de données avec PDO I

- Les fonctions PHP d'accès aux Systèmes de Gestion de Bases de Données sont nombreuses
- Il existait des APIs spécialisées natives pour des sgbds tels que MySQLi ou Oracle OCI8 ou PostgreSQL
- depuis PHP 5.1, l'extension PHP Data Objects (PDO) est installée par défaut et **doit être utilisée** car elle est indépendantes du sgbd cible
- Cette interface d'abstraction à l'accès de données signifie que vous utilisez les mêmes fonctions pour exécuter des requêtes ou récupérer les données quelque soit la base de données utilisée (portabilité) (MySQL, Oracle, ...)
- De plus, les requêtes préparées permettent d'évicter les attaques par injection de code SQL

Connexions et gestionnaire de connexion I

Les connexions sont établies en créant des instances de la classe de base (PDO) quel que soit le pilote (driver) de SGBD utilisé. Le constructeur nécessite un paramètre pour spécifier la source de la base de données (Data Source Name) et optionnellement, le nom d'utilisateur et le mot de passe (s'il y en a un). Le DSN est une chaîne de caractères composée :

- d'un préfixe indiquant le gestionnaire, par exemple `mysql:` ou `pgsql:` ou ... ;
- d'une suite de paramètres séparés par des **point-virgules** de la forme `param1=val1;param2=val2;...`. Ces paramètres peuvent être : `host, port, dbname, user, password, charset`

Remarquons que le couple (nom d'utilisateur, mot de passe) peut être fourni dans le DSN (prioritaire) ou comme paramètres du constructeur.

Connexions et gestionnaire de connexion II

```php
<?php // Connexion à MySQL
$user="mmeynard";$pass="XXX";
try{
  $dbh = new PDO('mysql:host=mysql.etu.umontpellier.fr;
  ↪  dbname=p00000010402; charset=UTF8', $user, $pass,
  ↪  array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,));
} catch(PDOException $e){
  echo $e->getMessage();
  die("Connexion impossible !");
}
?>
```

S'il y a des erreurs de connexion, un objet PDOException est lancé. Vous pouvez attraper cette exception si vous voulez gérer cette erreur, ou laisser le gestionnaire global d'exception la traiter via la fonction : `set_exception_handler()`

Connexions et gestionnaire de connexion III

```php
<?php // Connexion à PostgreSQL avec gestion des erreurs
try {
  $dbh = new PDO("pgsql:dbname=$dbname; host=$host;
  ↪  username=$username; password=$password");
  foreach($dbh->query('SELECT * from etudiant') as $row) {
    var_dump($row);
  }
  $dbh = null;     // déconnexion
} catch (PDOException $e) {
  print "Erreur !: " . $e->getMessage() . "<br/>";
  die();
}
?>
```

Pour fermer la connexion, il suffit de déréférencer l'objet PDO en affectant la variable à nul : `$dbh=null;`

Connexion persistante I

- Les connexions **persistantes** ne sont pas fermées à la fin du script, mais sont mises en cache
- puis réutilisées lorsqu'un autre script demande une connexion en utilisant les mêmes paramètres.
- Le cache des connexions persistantes permet d'éviter d'établir une nouvelle connexion à chaque fois qu'un script doit accéder à une base de données
- améliore la vitesse de l'application web

```php
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass,
 array(PDO::ATTR_PERSISTENT => true));
?>
```

Requêtes uniques I

Pour des requêtes individuelles, on peut utiliser la méthode `query` pour une consultation ou la méthode `exec` pour une modification. Il faut noter que l'objet `PDOStatement` retourné par `query` est `Traversable`, c'est-à-dire qu'on peut itérer directement dessus avec `foreach` (`fetch` automatique).

Requêtes uniques II

Exemple (Requête unique)

```php
<?php
try {
    $sql =  'SELECT nom, ue FROM utilisateur u, etudiant e WHERE
    u.id=e.id ORDER BY ue, nom';
    foreach($dbh->query($sql) as $row) {
        print($row['nom'] . "\t" . $row['ue'] . "<br />");
    }
    $nb=$dbh->exec("DELETE FROM utilisateur WHERE id NOT IN
    ↪  (SELECT id FROM etudiant)");
    print("Nb de suppressions : " . $nb);
?>
```

Requêtes préparées (PDOStatement) I

Une requête préparée est une sorte de modèle compilé pour la(es) requête(s) SQL que vous voulez exécuter. Les requêtes préparées offrent deux fonctionnalités essentielles :

- La requête ne doit être analysée (ou préparée) qu'une seule fois, mais peut être exécutée plusieurs fois avec des paramètres identiques ou différents. L'optimisation de son plan d'exécution permet à la requête préparée d'utiliser moins de ressources et de s'exécuter plus rapidement.
- Les paramètres pour préparer les requêtes (`:nomparam`) n'ont pas besoin d'être entre guillemets ; le driver gère l'association avec une valeur grâce à l'association (liaison) **bind**. Si votre application utilise exclusivement les requêtes préparées, vous pouvez être sûr qu'aucune injection SQL n'est possible.

Requêtes préparées (PDOStatement) II

PDO émule les requêtes préparées pour les drivers qui ne les supportent pas. Ceci assure de pouvoir utiliser la même technique pour accéder aux données, sans se soucier des capacités de la base de données. **Attention, les paramètres ne peuvent pas être utilisés pour remplacer des noms SQL** (mot-clé, identificateurs de fonction ou de colonne ! Par exemple, `order by :col` est interdit. Ils ne peuvent que remplacer des valeurs (chaines, entier, ...).

Exemple d'insertions répétées en utilisant les requêtes préparées I

Cet exemple effectue une requête INSERT en y substituant un nom et une valeur pour les **marqueurs nommés**.

```php
<?php
$stmt = $dbh->prepare("INSERT INTO REPERTOIRE (name, value)
↪  VALUES (:name, :value)");
$stmt->bindParam(':name', $name);   // association paramètre
↪  marqueur nommé
$stmt->bindParam(':value', $value); // avec variable PHP

// insertion de deux lignes
$name = 'Dupont';
$value = 0612345678;
$stmt->execute();

$name = 'Durand';
```

Exemple d'insertions répétées en utilisant les requêtes préparées II

```
$value = 0468901234;
$stmt->execute();
?>
```

Utilisation de marqueur anonyme <?> I

```php
<?php
<$stmt = $dbh->prepare("INSERT INTO REPERTOIRE (name, value)
↪  VALUES (?,?)");
$stmt->bindParam(1, $name);    // association paramètre marqueur
↪  anonyme
$stmt->bindParam(2, $value);   // avec variable PHP

// insertion
$name = 'Martin';
$value = 0612435678;
$stmt->execute();
?>
```

Utilisation de marqueur anonyme <?> II

Exemple (Consultation paramétrée avec marqueur anonyme)

```php
<?php
$stmt = $dbh->prepare("SELECT * FROM REPERTOIRE where name = ? or
↪  value = ? ");
if ($stmt->execute(array($_POST['nom'],$_POST['numero']))){ //
↪  liaison implicite
  while ($row = $stmt->fetch()) {
    print($row['name']." : ".$row['value']);
  }
}
?>
```

La récupération (fetch) des lignes résultant d'une consultation est réalisée séquentiellement selon la valeur du paramètre optionel de la méthode fetch. Par défaut, la valeur `PDO::FETCH_BOTH` permet de retourner la ligne

Utilisation de marqueur anonyme <?> III

suivante en tant que tableau indexé par le nom et le numéro de la colonne (0 à n-1). Mais on peut également :

- récupérer un objet dont les attributs correspondront aux colonnes ;
- se déplacer selon une orientation (vers le bas ou le haut) ;
- indiquer un déplacement différent de 1 dans la séquence de récupération.

Transaction I

- Les transactions offrent 4 fonctionnalités majeures : Atomicité, Consistance, Isolation et Durabilité (ACID)
- Le travail d'une transaction (suite de requêtes) peut être annulé (abort) à votre demande ou bien validé (commit)
- Malheureusement, toutes les bases de données ne supportent pas les transactions, donc, PDO doit s'exécuter en mode "autocommit" lorsque vous ouvrez pour la première fois la connexion
- Le mode "autocommit" signifie que toutes les requêtes que vous exécutez ont leurs transactions implicites, si la base de données le supporte ou aucune transaction si la base de données ne les supporte pas
- Si vous avez besoin d'une transaction, vous devez utiliser la méthode `PDO::beginTransaction()` pour l'initialiser

Transaction II

- Si le driver utilisé ne supporte pas les transactions, une exception PDO sera lancée
- Une fois que vous êtes dans une transaction, vous devez utiliser la fonction `PDO::commit()` ou la fonction `PDO::rollBack()` pour la terminer
- **Attention**, PDO ne vérifie la possibilité d'utiliser des transactions qu'au niveau du pilote
- Si certaines conditions à l'exécution empêchent les transactions de fonctionner, `PDO::beginTransaction()` retournera tout de même TRUE sans erreur si le serveur accepte de démarrer une transaction
- Ce sera le cas en utilisant le pilote MySQL sur des tables au format MyISAM car ce dernier ne supporte pas les transactions (utiliser plutôt InnoDB)

Transaction III

- Lorsque le script se termine ou lorsque la connexion est sur le point de se fermer, si vous avez une transaction en cours, PDO l'annulera automatiquement. Ceci est une mesure de sécurité afin de garantir la consistance de vos données dans le cas où le script se termine d'une façon inattendue. Si vous ne validez pas explicitement la transaction, alors, on présume que quelque chose s'est mal passé et l'annulation de la transaction intervient afin de garantir la sécurité de vos données.

```php
<?php
try {
  $dbh = new PDO('mysql:host=localhost;dbname=test', $user,
  ↪  $pass,
      array(PDO::ATTR_PERSISTENT => true));
  echo "Connecté\n";
} catch (Exception $e) {
  die("Impossible de se connecter: " . $e->getMessage());
}
```

Transaction IV

```php
try {
  $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  // pour lancer une exception sur chaque erreur
  $dbh->beginTransaction();
  $dbh->exec("insert into utilisateur (id, nom, prenom) values
  ↪  (12, 'Dupont', 'Jean')");
  $dbh->exec("insert into etudiant (id, ue) values (12, 'l3'");
  $dbh->commit();
} catch (Exception $e) {
  $dbh->rollBack();
  echo "Impossible d'ajouter l'étudiant : " . $e->getMessage();
}
?>
```

Synopsis des classes PDO et PDOStatement I

```
 PDO {
public __construct ( string $dsn [, string $username [, string
↪  $password [, array $driver_options ]]] )
public bool beginTransaction ( void )
public bool commit ( void )
public bool rollBack ( void )
public int exec ( string $statement )
public static array getAvailableDrivers ( void )
public bool inTransaction ( void )
public PDOStatement prepare ( string $statement [, array
↪  $driver_options = array() ] )
public PDOStatement query ( string $statement )
...
}
 PDOStatement implements Traversable {
/* Propriétés */
```

Synopsis des classes PDO et PDOStatement II

```
readonly string $queryString;
/* Méthodes */
public bool bindColumn ( mixed $column , mixed &$param [, int
↪  $type [, int $maxlen [, mixed $driverdata ]]] )
public bool bindParam ( mixed $parameter , mixed &$variable [,
↪  int $data_type = PDO::PARAM_STR [, int $length [, mixed
↪  $driver_options ]]] )
public bool bindValue ( mixed $parameter , mixed $value [, int
↪  $data_type = PDO::PARAM_STR ] )
public bool closeCursor ( void )
public int columnCount ( void )
public string errorCode ( void )
public array errorInfo ( void )
public bool execute ([ array $input_parameters ] )
public mixed fetch ([ int $fetch_style [, int $cursor_orientation
↪  = PDO::FETCH_ORI_NEXT [, int $cursor_offset = 0 ]]] )
public array fetchAll ([ int $fetch_style [, mixed
↪  $fetch_argument [, array $ctor_args = array() ]]] )
```

Synopsis des classes PDO et PDOStatement III

```
public string fetchColumn ([ int $column_number = 0 ] )
public mixed fetchObject ([ string $class_name = "stdClass" [,
↪  array $ctor_args ]] )
public int rowCount ( void )
public bool setFetchMode ( int $mode )
...
}
```

Un exemple de TP I

```
<html><head>  <title>Test de PDO et du driver MySQL</title>
    <meta http-equiv="Content-Type" content="text/html;
    ↪  charset=utf-8">
  </head>
  <body>
    <h1>Test de PDO</h1>
<?php
try {
  $dbh = new PDO('mysql:host=venus;dbname=mmeynard', "mmeynard",
  ↪  "XXXX");
  foreach($dbh->query('SELECT * from test') as $row) {
    print_r($row);
  }
  $dbh = null;    // fermeture connexion
} catch (PDOException $e) {
  print "Erreur !: " . $e->getMessage() . "<br/>";
```

Un exemple de TP II

```
    die();
}
?>
    </body>
</html>
```

MySQL et phpMyAdmin I

- phpMyAdmin est un outil d'administration d'une BD MySQL écrit en PHP
- Cet outil est accessible via le web et permet donc d'administrer à distance une BD
- On peut : créer, supprimer, modifier (alter) des tables, interroger, ajouter, supprimer, modifier des lignes, importer des fichiers textes contenant les données ...
- Le site de référence où l'on peut tester en direct est : `http://www.phpmyadmin.net/home_page/index.php`
- MySQL supporte plusieurs moteurs de stockage, qui gère différents types de tables. Les moteurs de tables MySQL peuvent être transactionnels ou non-transactionnels

MySQL et phpMyAdmin II

- Historiquement et par défaut, le moteur de stockage est `MyISAM` : il ne permet ni contraintes d'intégrité référentielles (clés étrangères), ni transactions mais est très rapide
- Le moteur de stockage `InnoDB` est une alternative à privilégier puisqu'elle offre ces fonctionalités indispensables (transactions, références) en plus d'autres.

Plan

Développement et débogage I

Le développement d'un projet PHP est réalisé, en général, avec un serveur local à la machine de développement (`localhost`). Puis en production, le projet est déplacé sur une machine serveur qui n'aura pas la même configuration, notamment pour l'affichage des erreurs. Pour déboguer, voici quelques conseils

- l'utilisation d'un bon IDE est indispensable (Zend, VSCode, Sublime)
- l'exécution en ligne de commande du fichier php (`php index.php`) permet de voir sur la sortie d'erreur standard (le terminal) les erreurs de l'interprète PHP que l'on ne voit pas sur le navigateur **notamment les erreurs de syntaxe**. Cependant, d'autres erreurs (session, cookies, ...) apparaîtront et tout ce qui concerne les paramètres (`GET, POST`) ne sera pas testable !
- Visualiser le code HTML sur le navigateur (F12) à l'aide des outils de débogage est **indispensable**

Développement et débogage II

- La console Javascript ou console d'erreurs permet également de visualiser les problèmes locaux aux navigateur : javascript et feuille de style.
- Les données complexes (objets, tableaux) peuvent être affichés récursivement de manière indentée grâce à `var_dump($tab)` ou `print_r($tab)`. Cependant, il faut privilégier `var_dump` car son comportement est modifié par l'utilisation de l'extension PHP indispensable **xdebug**.

Contexte de débogage I

Les fonction `array debug_backtrace();` et `debug_print_backtrace()` permettent de récupérer ou d'afficher le contexte de débogage sous la forme d'un tableau de tableau. Chaque case correspond à un appel de fonction emboité depuis la courante à l'indice 0 (celle qui appele `debug_backtrace`) jusqu'à la plus lointaine (*main*). Chaque case contient un tableau associatif :

- **function** nom de la fonction courante. Voir aussi __FUNCTION__ ;
- **line** numéro de la ligne courante. Voir aussi __LINE__ ;
- **file** nom du fichier courant. Voir aussi __FILE__ ;
- **class** nom de la classe si on est dans une méthode ;
- **object** objet courant si on est dans une méthode ;
- **type** type d'appel : méthode d'instance "->", méthode statique "::" ;

Contexte de débogage II

args liste des arguments ou liste des fichiers inclus.

Certaines constantes magiques peuvent être affichées à des fins de débogage :

__FUNCTION__ nom de la fonction courante ;

__FILE__ nom du fichier exécuté ;

__LINE__ numéro de ligne courante ;

Gestion des erreurs I

L'affichage des erreurs de l'interprète PHP est géré par la directive de configuration `display_errors` de `php.ini`. Si cette directive est `on`, on doit fixer un niveau de rapport exigeant à l'interprète afin de vérifier tous les avertissements et erreurs possibles : `error_reporting(E_ALL);`
Si cette directive `display_errors` est à `Off`, ce qui est souhaitable en mode production, les erreurs PHP sont loggées dans un fichier `php_error.log` qu'il faut ouvrir à chaque fois que l'on suspecte une erreur ! Si l'on est hébergé et que l'on ne peut modifier le `php.ini`, on peut modifier la directive `display_errors` :

- soit en débutant chaque script par :

```php
<?php
ini_set('display_errors', 1); error_reporting(E_ALL);
```

- soit en ajoutant un fichier `.htaccess` dans le répertoire (à condition que la directive Apache `AllowOverride` le permette) :

Gestion des erreurs II

```
php_flag display_startup_errors on
php_flag display_errors on
php_flag html_errors on
```

Plan

Débogage avec l'Extension PHP xdebug I

Un débogage professionnel nécessite un IDE permettant le débogage et une extension PHP Xdebug :

- Une extension PHP est une bibliothèque de fonctionalités permettant d'enrichir le langage de base (core)
- Certaines extensions sont destinées à réaliser des graphiques (gd), d'autres à dialoguer avec des SGBD (mysql), etc
- La liste des extensions chargées ainsi que des infos sur la configuration de ces extensions est affiché par `phpinfo()`
- Xdebug permet de déboguer du php en permettant le pas à pas, les points d'arrêts, etc
- Pour cela, Xdebug utilise un pprotocole de débogage `dbgp` qui utilise un **port différent de celui du serveur Web** (par défaut 9000)
- Il contient d'autres fonctionalités (surcharge de `var_dump()` en limitant à 3 la profondeur d'affichage des objets, ...)

Débogage avec l'Extension PHP xdebug II

- Afin de permettre un débogage graphique, il existe des extensions des principaux IDE (VSCode, Sublime, ...) pour dialoguer avec Xdebug

Installation et utilisation de Xdebug sous VSCode I

- télécharger l'extension Xdebug dans le répertoire prévu à cet effet (dans les XAMP, cette phase est généralement inutile) ;
- l'activer dans le `php.ini` du serveur Apache en décommentant la ligne de la section `[xdebug]` correspondant à `zend_extension` ;
- sur MacOS X, modifier 2 fichiers php.ini : `/Applications/MAMP/conf/php7.4.2/php.ini` et `Applications/MAMP/bin/php/php7.4.2/conf/php.ini`
- paramétrer l'extension en modifiant des directives de configuration de sa section ; (ou en utilisant `ini_set(clé,valeur)`)

Installation et utilisation de Xdebug sous VSCode II

```
xdebug.remote_enable=1
xdebug.remote_host=127.0.0.1     ; ip de VSCode
xdebug.remote_connect_back=1   ; Not safe for production
↪  servers
xdebug.remote_port=9000        ; port par défaut de l'IDE
↪  serveur
xdebug.remote_handler=dbgp
xdebug.remote_mode=req
xdebug.remote_autostart=0      ; à éviter sinon toute page
↪  locale
xdebug.idekey=debug      ; à transmettre au serveur dbgp
↪  (VSCode)
```

- télécharger éventuellement l'extension de votre IDE correspondante : par exemple PHP Debug 1.13 de Felix Becker pour VSCode ;
- Sélectionner l'icône de Run/Debug dans la barre d'activité

Installation et utilisation de Xdebug sous VSCode III

- Ouvrir la configuration de lancement en cliquant sur la roue dentée du haut gauche
- éditer le fichier `launch.json` afin de spécifier la config. :

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Listen for XDebug",
      "type": "php",
      "request": "launch",
      "port": 9000
    },
    {
      "name": "Launch currently open script",
      "type": "php",
      "request": "launch",
```

Installation et utilisation de Xdebug sous VSCode IV

```
            "program": "${file}",
            "cwd": "${fileDirname}",
            "port": 9000
        }
    ]
}
```

- Les 2 entrées permettent :

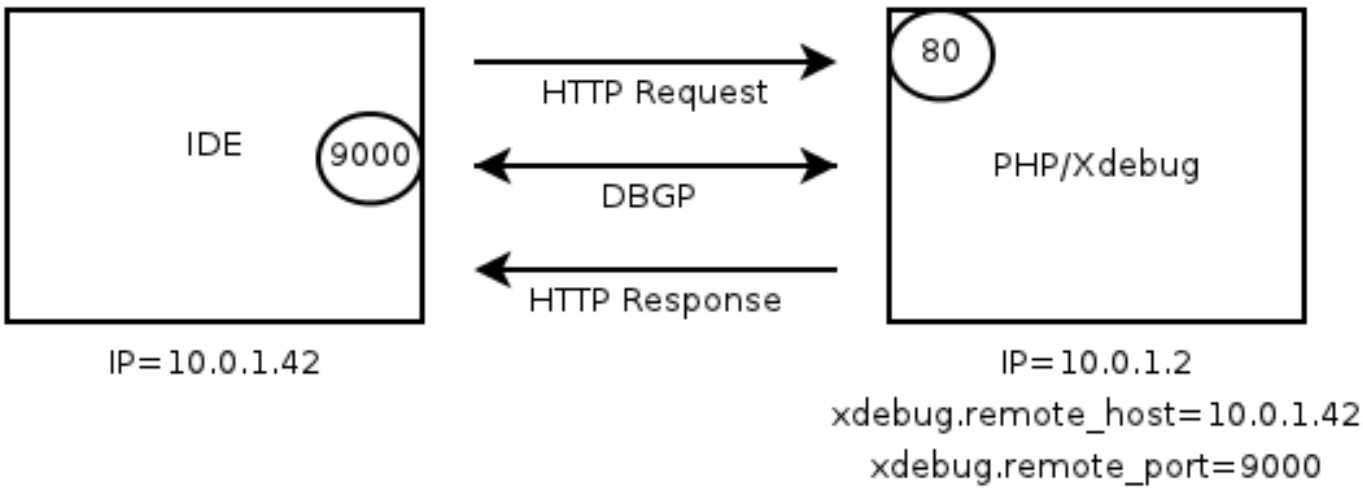
  Launch currently open script permet de déboguer un script en visualisant ses variables, ses superglobales avec des points d'arrêts : mode classique de débogage d'un script

Installation et utilisation de Xdebug sous VSCode V

Listen for XDebug permet à VSCode de créer un serveur dbgp qui écoute sur le port 9000 local une requête provenant de Xdebug agissant comme client dbgp suite à une requête http sur le serveur http. Cela permet de déboguer depuis une interaction humaine dans le navigateur (get ou post)

Installation et utilisation de Xdebug sous VSCode VI

FIGURE – fonctionnement du DeBuG Protocol



La transmission de la clé `idekey` depuis le navigateur vers le serveur http et son extensio XDebug peut être réalisé de plusieurs façons :

Installation et utilisation de Xdebug sous VSCode VII

- via un paramètre GET dans la "query string" : `http://localhost:8888/Preinscriptions/index.php?XDEBUG_SESSION_START=debug`
- via une variable d'environnement : `export XDEBUG_CONFIG="idekey=debug"` pour les scripts lancés depuis un terminal
- via une extension de navigateur comme "Xdebug Helper" pour Firefox

Déboguer sans droit administrateur I

Côté Serveur HTTP

- sans les droits administrateurs, impossible de modifier les fichiers de configuration dont le `php.ini` !
- Il faut donc lancer un serveur `http` local à la ligne de commande (`php -S localhost:8080`) **dans votre répertoire php** `~/public_html/Archiweb/`
- De plus, pour pouvoir déboguer, il faut fournir un certain nombre d'options :

  -dzend_extension=xdebug.so pour utiliser l'extension `Xdebug`
  -dxdebug.remote_enable=1 pour déboguer à distance
  -dxdebug.idekey=debug pour indiquer à l'extension la clé partagée (n'importe quel mot peut être utilisé)
- Au final, la commande peut être sauvée dans le `.bashrc` afin d'éviter les oublis :

Déboguer sans droit administrateur II

```
alias phps="php -S localhost:8080 -dxdebug.remote_enable=1
↪  -dxdebug.idekey=debug"
```

- l'option `-dxdebug.remote_autostart=1` peut aussi être utilisé pour toujours tenter d'ouvrir une session de débogage dans l'IDE sans utiliser de clé partagée
- les options `remote_host` et `remote_port` sont les bonnes par défaut

Côté IDE (serveur `dbgp`) à la FDS

- A la FDS, utiliser VSCodium et installer les deux extensions PHP situées dans le "PHP Extension Pack" de F. Becker (intellisense et debug)
- Ouvrir un dossier (projet) `~/public_html/Archiweb/` contenant des sources php
- choisir la vue PLAY/DEBUG

Déboguer sans droit administrateur III

- cliquer sur la roue dentée pour visualiser le `launch.json`
- ouvrir un fichier php, i.e. `bataille.php`, puis poser un point d'arrêt (disque rouge) en cliquant dans la gouttière verticale à gauche de la ligne désirée
- lancer une session de débogage en choisissant le mode `Listen for XDebug` (la ligne d'état change de couleur et passe à l'orange : on est dans une session)
- ouvrir son navigateur préféré sur l'url `http://localhost/bataille.php?XDEBUG_SESSION_START=debug`
- visualiser dans l'IDE la position courante du débogueur dans le code matérialisé par un triange
- ensuite, exécution pas à pas, examen des variables y compris les superglobales ...
- Ne pas oublier de fermer la session à la fin en cliquant sur l'icône carrée (stop)

Plan

Caractères spéciaux et validation des données I

Sources de séances de débogages longues et pénibles, il est important de comprendre les transformations des caractères spéciaux. Auparavant, la configuration `magic_quotes_gpc` ajoutait automatiquement des **échappement** par un anti-slash de certains caractères. Cette configuration est obsolète. L'échappement doit être réalisé en fonction de chaque formulaire.

- Lors de l'affichage d'une chaîne contenant des caractères html spéciaux (`&"<'>`), il faut les transformer en entité HTML (`&quot;`). Pour afficher proprement une chaîne contenant ces caractères, il faut au préalable la traiter avec `htmlspecialchars($champ, ENT_QUOTES))`.
- **Attention**, si le champ posté doit être remis en tant que "value" dans un `<input type='text'` de formulaire, il faut donc transformer ces caractères en entités HTML.

Caractères spéciaux et validation des données II

- Par conséquent, pour un champ interactif, on écrira :
  `echo '<input name="c" value="'.htmlspecialchars( $_POST['c'],ENT_QUOTES).'">';`
- les fonctions `html_entities(\$s)` et `html_entity_decode($s)` réalisent le même encodage/décodage pour toutes les entités HTML existantes (e.g. `&Bigcup;` pour le symbole Union)
- Lors de l'envoi d'une requête à un SGBD, il faut parfois échapper les caractères spéciaux tels que : `',\,",NULL`. La fonction `addslashes($chaine)` effectue ce travail. La fonction : `addcslashes (string $str, string $charlist) : string` permet de d'échapper les caractères de str qui sont dans une liste spécifiée dans charlist.
- De même en JavaScript, la fonction `addslashes()` permetra d'échapper du code.

Caractères spéciaux et validation des données III

- Remarquons que dans une session, les caractères spéciaux ne sont pas échappés.

Plan

Administration PHP I

Pour visualiser la configuration PHP, il suffit d'appeler la fonction `phpinfo()` pour voir :

- la version de php ;
- la localisation du fichier de configuration de php : `php.ini` ;
- les librairies incluses et leur configuration (mysql, gd, ...) ;
- de nombreuses autres choses importantes.

La configuration de l'interprète PHP est réalisé dans le fichier `php.ini`. Ce fichier `php.ini` contient des directives (sous forme d'affectation de variables) fondamentales :

register_globals=On si **on** alors les variables d'Environnement, Get, Post, Cookie, Serveur sont globales ; si **off** alors les variables sont accessibles via `$_POST[]`, ... (Sécurité +) ;

Administration PHP II

variables_order="EGPCS" ordre de résolution des conflits de noms de variables Env, GET, POST, COOKIE, Serveur ;

magic_quotes_gpc=On permet d'anti-slasher les caractères anti-slash, guillemet et apostrophe dans les variables GPC.

Plan

Authentification I

L'authentification par un couple (login, password) peut être réalisé de bien des façons différentes sur le web. Dans la plupart des cas, on peut réaliser l'authentification par un formulaire HTML classique puis interrogation en PHP d'une base de données des utilisateurs contenant les couples (login, password **crypté**). Une fois vérifié la concordance, il suffira de conserver dans le tableau de session les informations concernant l'utilisateur loggé (id, nom, type, ...) : `$_SESSION['uid']`, ...
Cependant, il faut également connaître les autres types d'authentification ! La procédure d'**authentification HTTP** est associée à un **nom de domaine** (realm ou AuthName) et à un répertoire. Elle peut être déclenchée :

- soit par Apache, indépendamment de PHP ;
- soit en utilisant PHP.

Authentification II

L'authentification **HTTP via Apache** est valable pour toute une **arborescence**. Un fichier `.htaccess` spécifie les règles d'authentification valables pour ce répertoire et tous ses **descendants** :

```
AuthType Basic
AuthUserFile "/auto/.../AuthApache/.htpasswd"
AuthName "Le Domaine Privé"
<Limit GET POST>
 require valid-user
</Limit>
```

Le fichier `.htpasswd` contient la liste des utilisateurs et leurs mots de passe cryptés. Il est obtenu grâce à l'utilitaire `htpasswd` fourni par Apache. Par exemple : `htpasswd -c .htpasswd un` ; crée un fichier avec l'utilisateur `un`.
Lors de tout accès à un fichier descendant de `AuthApache`, Apache va envoyer un en-tête au navigateur client qui va afficher une fenêtre popup

Authentification III

d'authentification. Par la suite, l'utilisateur reste authentifié pour "Le Domaine Privé" jusqu'à la fin du navigateur ou si une nouvelle authentification PHP est lancée.

Authentification HTTP via PHP I

PHP doit être un module d'Apache. On utilise alors la fonction `header` pour demander une authentification ("WWW-authenticate") au client, générant ainsi l'apparition d'une fenêtre de demande d'utilisateur et de mot de passe. Une fois que les champs ont été remplis, l'URL sera de nouveau appelée, avec les variables `$_SERVER['PHP_AUTH_USER']`, `$_SERVER['PHP_AUTH_PW']` et `$_SERVER['PHP_AUTH_TYPE']` contenant respectivement le nom d'utilisateur, le mot de passe et le type d'authentification. Actuellement, seule l'authentification de type "Basic" est supportée. Si l'authentification est réalisée via Apache (`.htaccess`), la variable `$_SERVER['REMOTE_USER']` est égale à `$_SERVER['PHP_AUTH_USER']`.

Authentification HTTP via PHP II

Exemple (Exemple d'authentification HTTP par PHP)

```php
<?php
if (!isset($_SERVER['PHP_AUTH_USER']) || !verifierAuth())
  header("WWW-Authenticate: Basic realm=\"Le Domaine Privé\"");
  header("HTTP/1.0 401 Unauthorized");
  echo "Texte à envoyer si le client annule\n";
  exit;
  } else {
  echo "Bonjour ",$_SERVER['PHP_AUTH_USER'];
  // suite de la page privée
  }
?>
```

La fonction booléenne `verifierAuth()` utilisera tout moyen nécessaire à la vérification du nom et du mot de passe (Base de données, ...).
Remarquons que les variables `$_SERVER['PHP_AUTH_...']` sont

Authentification HTTP via PHP III

utilisables même si l'authentification n'a pas été effectuée par le module PHP.

Les variables d'authentification PHP sont valables pour tous les fichiers descendants du répertoire. Par contre, tout fichier html ou php ne testant pas l'authentification est accessible, contrairement à l'authentification par .htaccess (Apache) qui sécurise tout le répertoire.

Désauthentification PHP : Le navigateur écrase le cache d'authentification client d'un domaine quand il reçoit une nouvelle demande d'authentification. Cela permet de déconnecter un utilisateur, pour le forcer à entrer un nouveau nom et son mot de passe. Si l'utilisateur annule l'authentification, il est alors désauthentifié ! Penser à recharger la page.

Certains programmeurs changent dynamiquement le nom de domaine pour donner un délai d'expiration, ou alors, fournissent un bouton de réauthentification.

Téléchargement I

- du site web vers le client : download ou téléchargement ;
- du client vers le site web : upload ou chargement ou téléversement ;

En HTML, pour permettre le download, on réalise un lien référençant le fichier à télécharger, par exemple : `<ahref="Public/toto.pdf">Cliquerici</a>`. Si le type du fichier est affichable par le navigateur, il sera affiché (après son téléchargement), sinon il sera proposé à l'utilisateur soit de sauver le fichier, soit de lancer l'application associée. Remarquons que tout lien peut être enregistré en utilisant le bouton droit de la souris sur le lien.

Téléversement I

On veut réaliser un formulaire de chargement envoyant une requête POST. L'action effectuée (script) après soumission doit vérifier que le fichier chargé est du bon type et de la bonne taille puis l'afficher depuis la zone temporaire (tmp/) ou il est chargé. Le fichier temporaire sera **automatiquement effacé** de la zone à la fin du script, s'il n'a pas été déplacé ou renommé :

```
<FORM ENCTYPE="multipart/form-data" ACTION="traitement.php"
↪  METHOD="POST">
  <INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" value="1000">
  Envoyez ce fichier : <INPUT NAME="fichier" TYPE="file"
  ↪  SIZE=15>
  <INPUT TYPE="submit" VALUE="Envoyer le fichier">
</FORM>
```

Dans le script `traitement.php` on a accès à différentes variables (différent selon les versions de PHP) :

Téléversement II

- `$_FILES['fichier']['name']` le nom du fichier original chez le client ;
- `$_FILES['fichier']['type']` le type mime du fichier "image/gif, text/plain, ..." ;
- `$_FILES['fichier']['size']` la taile du fichier chargé ;
- `$_FILES['fichier']['tmp_name']` le nom du fichier temporaire sur le serveur ;
- `$_FILES['fichier']['error']` le code d'erreur (PHP 4.2.0).

Remarquons que tous les navigateurs ne vérifient pas tous le `MAX_FILE_SIZE`.

SPL et Phar I

- La *Standard PHP Library (SPL)* fournit par défaut (sans include, sans modifier le php.ini) des interfaces et des classes permettant de gérer des collections de données. Par exemple, suit une liste de quelques classes utiles de la SPL : SplDoublyLinkedList, SplStack, SplQueue, SplHeap, SplPriorityQueue, SplObjectStorage. Mais aussi des fonctions comme `spl_autoload()`.
- `phar` est un format de fichier archive (PHp ARchive) permettant de sauvegarder une application (arborescence) dans un unique fichier d'extension `phar`. C'est l'équivalent des fichiers `jar` de Java.
  L'interprète PHP en ligne de commandes est capable d'exécuter un fichier `phar` directement : `$ php monappli.phar`
  On peut également définir une bibliothèque dans un fichier phar et n'inclure que certaines parties grâce à une syntaxe appropriée :

SPL et Phar II

```php
<?php
include 'malib.phar'; // inclusion de tous les fichiers
include 'phar://malib2.phar/Commun/inc-Html.php'; // Commun
↪  est un répertoire de malib2
```

Le format interne des fichiers de l'archive peut utiliser différents formats de compression (zip, phar ou tar).

Sérialisation des tableaux et objets I

La sérialisation permet de transformer toute variable complexe (objet, tableau) de PHP en une chaîne de caractères qui pourra être désérialisée ultérieurement. La chaîne sérialisée peut être sauvée dans un fichier ou transmise sur un réseau.

```
string serialize  ( mixed $value  )
mixed unserialize  ( string $str  )
```

Dans une session PHP, tableau superglobal `$_SESSION`, les variables sont sérialisées automatiquement dans le fichier du serveur qui les conserve entre 2 requêtes.
La sérialisation réalise une copie **profonde** d'un objet contrairement à l'opérateur clone qui effectue une copie superficielle.

```
$copieprof=unserialize(serialize($object))
```

Sérialisation des tableaux et objets II

Attention, les attributs de type ressource (fichier, connexion) ne sont pas sérialisables.

Plan

Apports de ce cours I

- introduction aux technologies du web non exhaustive
- séparation de la forme (styles CSS) et du contenu HTML
- utilisation d'un framework css responsive : bootstrap
- utilisation du protocole de débogage XDebug
- répartition de la logique de l'application entre script côté client (js) et côté serveur (php)
- double validation des données côté client et serveur
- architecture 3-tiers avec SGBD

Perspectives I

- architecture MVC
- architecture logicielle à composants
- Design Pattern (ORM, ...)
- le routage, une notion primordiale
- la sécurité des sites Web (HTTPS, authentification, . . .)
- templates PHP (twig, ...)
- structuration d'un projet complexe nécessitant de multiples pages : utilisation de frameworks (PHP : Laravel, Symfony, ... JS : Angular 2, React, Vue ...)

Bibliographie I


[1] *Programmation HTML et Javascript*, de Chaléat, Charnay, Eyrolles (2000), "simple et rapide pour les bases HTML, CSS, JavaScript, 450 pages"


[2] *Webmaster in a nutshell*, de S. Spainhour, R. Eckstein, O'Reilly (2003), "Manuel de référence rapide pour HTML, CSS, JavaScript, PHP, XML, 550 pages, en anglais"


[3] *Spécification HTML*, `http://www.w3.org/TR/REC-html40/`, W3C (1999), "La spécification complète en anglais"


[4] *Un site de documentation sur les technologies du web*, `http://www.w3schools.com/`, "Très complet et en anglais (XHTML, CSS, JavaScript) "


[5] *Le manuel PHP en français*, `https://www.php.net/manual/fr/`, "La référence "

Bibliographie II

 [6]*Site WAMP*, `http://www.wampserver.com/`, “Le site Web de WAMP (Apache, MySQL, PHP pour Windows)”