



Module 5

# PHP : un langage pour le Web


# Contenu du module

- Interactions avec le protocole HTTP
- Transmission d'informations
  - Récupération des informations en PHP
- Construction de formulaires HTML.
  - Modèles d'interaction HTML / PHP
  - Récupération des données transmises via PHP en fonction des différents types de contrôles
  - Redirection de flux
- Gestion des sessions des utilisateurs
  - Les principes du suivi de session HTTP
  - Les cookies et leur manipulation
  - Les sessions PHP

# Interactions avec le protocole HTTP

- Le protocole HTTP est, par définition, LE protocole d'Internet
  - Permet des échanges clients/serveurs (Serveurs Web/Navigateurs)
  - Basé sur du texte
  - Utilise les URLs et les méthodes HTTP
- URL : Uniform Resource Locator
  - Adresse permettent d'identifier de manière unique une ressource sur Internet
    - Ex. : <http://www.monsite.com/achats/livres/0001>
- Méthode HTTP
  - Permet de préciser le type d'interaction avec la ressource
  - En navigation Web classique on utilise essentiellement GET et POST
    - Il existe également PUT, DELETE, TRACE, ...

# Transmission d'informations

- Selon la méthode HTTP utilisée, la transmission d'information se fait de différente manière
- Méthode GET
  - Sémantiquement, une demande d'information
    - Saisie d'une URL dans la barre d'adresse du navigateur, clique sur un lien
  - Par de corps dans la requête
    - On utilise l'URL avec une « Query String » pour transmettre des paramètres
    - Ex. : <http://www.monsite.fr/index.php?page=admin&action=edit>

Paramètres : **page** et **action**  
Et leurs valeurs associées...
- Méthode POST
  - Sémantiquement, un envoi de données
    - Via un formulaire
  - Les paramètres sont transmis dans le corps de la requête
    - Leurs noms sont les noms des champs de formulaire

# Récupération des informations en PHP

- Pour récupérer les informations transmises, PHP met à disposition 3 tableaux associatifs prédéfinis contenant les paramètres transmis par une requête
  - `$_GET`
    - Pour les paramètres transmis en GET via l'URL
  - `$_POST`
    - Pour les paramètres transmis en POST via un formulaire
  - `$_REQUEST`
    - Pour les paramètres transmis en GET et/ou en POST
      - Revient à ne pas distinguer la méthode HTTP utilisée
  - Les clés de ces tableaux sont les noms des paramètres !
- Exemple :
  - Pour l'URL : <http://www.monsite.fr/index.php?page=admin&action=edit>
  - On récupère les paramètres via `$_GET['page']` et `$_GET['action']` dans le script `index.php`
    - Attention à bien contrôler l'existence de ces paramètres et la cohérence de leurs valeurs !

# Construction de formulaires HTML

- Outil indispensable pour les sites web dynamiques
  - Permet à l'utilisateur d'interagir avec le site ou l'application
- Défini entre les balises `<form>` et `</form>`
- Syntaxe :

```
<form
  [ action="url_de_traitement" ]
  [ method="GET"|"POST" ]
  [ id="identifiant_formulaire" ]>

...
</form>
```

- Entre les balises `<form>``</form>` on y placera les différents champs de formulaire avec les balises HTML appropriées

# Les champs de formulaire – 1

```
<div>
  <label for="nom">Nom :</label>
  <input type="text" id="nom" name="nom" value="" size="20" />
</div>
<div>
  <label for="nom">Mot de passe :</label>
  <input type="password" id="mot_de_passe" name="mot_de_passe" />
</div>
```

Nom :

Mot de passe :

```
<div>
  <fieldset>
    <legend>Sexe : </legend>
    <input type="radio" id="sexeM" name="sexe" value="M" />
    <label for="sexeM">Masculin</label>
    <input type="radio" id="sexeF" name="sexe" value="F" />
    <label for="sexeM">Féminin</label>
    <input type="radio" id="sexeI" name="sexe" value="?" checked="checked" />
    <label for="sexeM">Ne sait pas</label>
  </fieldset>
</div>
```

Sexe : ☐ Masculin ☐ Féminin ☒ Ne sait pas

```
<div>
  <label for="sexeM">Photo : </label>
  <input type="file" id="photo" name="photo" size="50" />
</div>
```

Photo :  Choisir un fichier

```
<div>
  <label for="langue">Langue : </label>
  <select id="langue" name="langue">
    <option value="E">Espagnol</option>
    <option value="F" selected="selected">Francais</option>
    <option value="I">Italien</option>
  </select>
</div>
```

Langue :  ▾

# Les champs de formulaire – 2

```
<div>
  <fieldset>
    <legend>Couleurs préférées : </legend>
    <input type="checkbox" id="bleu" name="couleurs[bleu]" />
    <label for="bleu">Bleu</label>
    <input type="checkbox" id="blanc" name="couleurs[blanc]" />
    <label for="blanc">Blanc</label>
    <input type="checkbox" id="rouge" name="couleurs[rouge]" />
    <label for="rouge">Rouge</label>
    <input type="checkbox" id="pas" name="couleurs[pas]" checked="checked" />
    <label for="rouge">Rouge</label>
  </fieldset>
</div>
```

Couleurs préférées :

☐ Bleu ☐ Blanc ☐ Rouge ☒ Rouge

```
<div>
  <label for="fruits">Fruits préférés : </label>
  <select id="fruits" name="fruits[]" multiple="multiple" size="8">
    <option value="A">Abricots</option>
    <option value="C">Cerises</option>
    <option value="F">Fraises</option>
    <option value="P">Pêches</option>
    <option value="?" selected="selected">Ne sait pas</option>
  </select>
</div>
```

Fruits préférés :

Abricots  
Cerises  
Fraises  
Pêches  
Ne sait pas

```
<div>
  <label for="commentaire">Commentaire : </label>
  <textarea id="commentaire" name="commentaire" rows="4" cols="50"></textarea>
</div>
```

Commentaire :

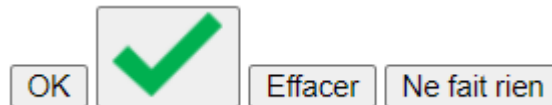


# Les champs de formulaire – 3

```
<div>
  <input type="hidden" id="invisible" name="invisible" value="123" />
</div>
```

Champ caché pour transmettre une valeur sans action utilisateur

```
<div>
  <button type="submit" id="soumettre" name="soumettre">OK</button>
  <button type="reset" id="effacer" name="effacer">Effacer</button>
  <button type="button" id="action" name="action">Ne fait rien</button>
  <button type="button" id="valider" name="valider">
    
  </button>
</div>
```



- `<button type="button" ...>`
  - Type de bouton à câbler sur des événements JavaScript

# Modèles d'interaction HTML / PHP

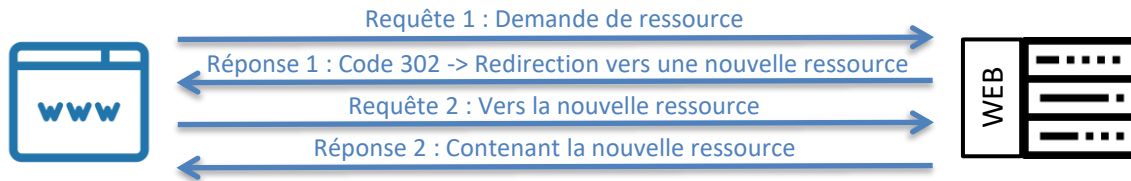
- 3 Techniques :
  - 1 fichier HTML + 1 script PHP :
    - Formulaire dans un document HTML (.htm ou .html)
    - Indiquer le nom du script PHP qui doit traiter le formulaire dans l'attribut action de la balise <form>
      - Le formulaire ne contient aucun élément dynamique.
  - 2 scripts PHP :
    - Formulaire dans un script PHP
      - Pour construire une partie du formulaire dynamiquement
    - Faire traiter le formulaire par un autre script PHP
      - Mentionné dans l'attribut action de la balise <form>
  - 1 script PHP :
    - Formulaire dans un script PHP
      - Pour construire une partie du formulaire dynamiquement
      - Le faire traiter par le même script PHP
        - » Mentionné dans l'attribut action de la balise <form>

# Récupération des données transmises

- Toutes les zones de formulaire sont automatiquement enregistrées dans le script PHP qui traite le formulaire :
  - \$\_POST pour les formulaires POST
  - \$\_GET pour les formulaires GET
- La clé du tableau est égale au nom du champ du formulaire
  - Attribut name de la balise <input>, <select> ou <textarea>
  - La valeur = la valeur saisie dans la zone
- Il est impératif de contrôler :
  - La présence des clés dans le tableau associatif concerné
  - La cohérence des valeurs

# Redirection de flux

- En fin de sortie de traitement de formulaire, ou après d'autres traitements, il est parfois nécessaire de rediriger l'utilisateur vers une page spécifique
  - Une redirection implique 2 requêtes et deux réponses



- En PHP, les redirections se font en envoyant l'entête de réponse Location au navigateur
  - La fonction `header()` PHP permet d'envoyer des entêtes de réponse
  - ATTENTION :
    - L'envoi d'entête de réponse se fait avant l'envoi de données dans le corps de la réponse
      - Le corps de la réponse contient habituellement le code HTML
    - Il est donc impératif que l'appel à la fonction `header()` se fasse avant tout appel à `echo` !

# Gestion des sessions utilisateurs

- HTTP est un protocole "sans état"
  - Pas d'état conversationnel
    - Rien ne permet d'identifier que l'utilisateur qui arrive sur une page 2 était précédemment sur la page 1
- Un site interactif a souvent besoin d'identifier un utilisateur d'une page à l'autre et de conserver des informations
  - Panier d'achat, authentification, ...
- Une session est la période de temps correspondant à la navigation continue d'un utilisateur sur un site
- La gestion des sessions consiste donc à être capable :
  - d'identifier l'instant où un nouvel utilisateur accède à une page du site
  - de conserver des informations relatives à cet utilisateur jusqu'à ce qu'il quitte le site
- A noter que l'utilisateur n'est pas forcément authentifié !

# Les principes du suivi de session HTTP

- Une session utilisateur est associée à un emplacement mémoire sur le serveur Web
  - Chaque session est associée à un identifiant unique généré par le serveur
  - Les données peuvent aussi être stockées sur disque pour soulager le serveur
- Pour maintenir le suivi de session, le serveur renvoie l'identifiant de session généré dans la réponse au navigateur
  - Via un cookie = Une paire clé/valeur
  - Dans le cas de la gestion des sessions PHP :
    - la clé s'appelle PHPSESSID
    - La valeur est l'identifiant généré
  - Le cookie est ensuite renvoyé par le navigateur à chaque nouvelle requête
  - Le cookie est perdu dès lors que le processus du navigateur se termine
    - Toutes les fenêtres sont fermées
- Ce mécanisme permet d'associer un utilisateur unique, associé à un navigateur recevant un identifiant de session, à un emplacement de stockage d'informations sur le serveur

# Les sessions PHP

- PHP propose un ensemble de fonctions pour gérer les sessions
- Principes :
  - Un identifiant unique est automatiquement attribué à chaque session
  - L'identifiant unique est transmis d'une page à l'autre par cookie de session (stocké dans la mémoire du navigateur)
  - On stocke les données à conserver dans le tableau de données de session `$_SESSION` accessible dès lors qu'une session est active

# Fonctions de gestion de sessions

- `session_start()`
  - Ouvre une nouvelle session ou réactive la session courante
  - Après l'appel, on ne sait jamais si une session a été créée ou bien récupérée !
  - Indispensable pour utiliser `$_SESSION`
  - Envoie un cookie pour suivi, via les entêtes de réponse
    - Il ne faut pas avoir envoyé de données avant l'appel à `session_start()` !
    - Sinon :
      - `Warning: Cannot modify header information - headers already sent by ...`
- `session_id()`
  - Retourne (ou éventuellement modifie) l'identifiant de la session
- `session_name()`
  - Retourne (ou éventuellement modifie) le nom de la variable utilisée pour stocker l'identifiant de la session
- `session_destroy()`
  - Supprime la session
- `session_status()`
  - Retourne le statut actuel d'une session



# En pratique...

- Créer une session
  - Appeler `session_start()` et stocker une information dans le tableau `$_SESSION`
- Vérifier l'existence d'une session
  - Appeler `session_start()` et vérifier que la donnée attendue (celle stockée au moment de la création de la session) est bien présente
- Supprimer une session
  - En pratique, un appel à `session_destroy()` ne suffit pas
  - Il faut :
    - Vider le tableau `$_SESSION`
    - Appeler `session_destroy()`

# Travaux pratiques

