TP 8 - Formulaires HTML

1 Introduction

Les formulaires HTML permettent à l'utilisateur d'une page de saisir des informations, puis de lancer une action qui va générer une requête sur le serveur. Vous allez étudiez dans cette partie à la fois la syntaxe HTML des formulaires mais aussi la façon dont le serveur va recevoir les informations.

1.1 Principe

1. Dans un squelette html minimal mais valide, saisir le code suivant et l'enregistrer en tant que **form_1.html** :

```
<form>
  Identification
  <label>Nom:</label>
  <input type="text" name="lastname" placeholder="Durand"
      required="required">
  <label>Prénom:</label>
  <input type="text" name="firstname" placeholder="Paul">
  <input type="submit" value="Envoyer">
</form>
```

2. Afficher la page, tapez quelques caractères dans les zones de texte puis valider. Noter l'URL qui apparaît dans la barre d'adresse du navigateur :

   _____

3. En déduire, lors de la requête sur le serveur :
   (a) le caractère utilisé pour séparer l'URL de la page des valeurs transmises : _____
   (b) le caractère utilisé pour séparer les différentes paires champs-valeur transmises : _____
   (c) le caractère utilisé pour séparer le nom du champ de la valeur saisie : _____

4. A quoi sert l'attribut **required** (essayer avec et sans) :

   _____

5. A quoi sert l'attribut **placeholder** (essayer avec et sans) :

   _____

6. Ajouter avant le champ du bouton "submit" un champ "password" :

```
<label>Mot de passe:</label>
<input type="password" name="pwd">
```

   Afficher la page et saisir des valeurs. Dans le champ "password", les caractères tapés s'affichent-ils ? : _____

7. Valider le formulaire avec le bouton d'envoi et observez l'URL. Ce mode de saisie d'un mot de passe est-il pertinent ? Pourquoi ? :

   _____

1.2 Plusieurs formulaires dans la même page

8. Ajouter dans la même page un **autre** formulaire avec les champs suivants :

```
<form>Sexe:<br>
  <input type="radio" name="sexe" value="M">Male<br>
  <input type="radio" name="sexe" value="F">Femelle<br>
  <input type="radio" name="sexe" value="S" checked="checked">
      Oui<br>
  <input type="submit" value="Envoyer">
</form>
```

9. A quoi sert l'attribut **checked** :

   _____

10. En considérant que les attributs **name** et **value** ont pour valeur **AAA** et **BBB**, donner la forme sous laquelle ce type de champ est transmis dans l'URL : `form_1.html` _____

On a ici une page avec deux formulaires. Lors de l'envoi au serveur, il est nécessaire que celui-ci puisse identifier le formulaire source, d'autant que deux formulaires peuvent avoir des champs avec des noms identiques.

11. Attribuer à chaque formulaire (balise **form**) un attribut **name** avec des valeurs différentes. Faire l'essai et vérifier l'URL de la requête générée. La valeur de cet attribut est-elle transmise ? : _____

12. Pour identifier le formulaire source, on ajoute un champ invisible pré-rempli dans la page et qui permettra au serveur d'identifier à coup sûr le formulaire

source. Ajouter dans chaque formulaire de la page le code suivant, en mettant comme valeur "f1" et "f2" pour les deux formulaires :

```
<input type="hidden" name="id" value="f1">
```

Vérifier ensuite la requête générée (URL visible).

1.3 Autres types de champ

Ajouter le code suivant dans la page :

```
<form>
  J'aime manger:<br>
  <input type="checkbox" name="manger" value="fri">des frites<br>
  <input type="checkbox" name="manger" value="pom">des pommes<br>
  <input type="checkbox" name="manger" value="ch">des chiens<br>
  <input type="submit" value="Envoyer">
</form>
```

13. Quelle est la différence avec l'entrée de type "radio" ?

    _____

14. Sous quelle forme le choix de l'utilisateur est-il transmis ?

    _____

15. Chercher sur `http://www.w3schools.com/tags/tag_input.asp` le nombre de types de champs possibles : _____

2 Personnalisation de la requête

La balise **form** permet d'orienter la requête du navigateur lors de l'envoi du formulaire sur une autre page, via l'attribut **action**. Par exemple :

```
<form action="index.php">
...
</form>
```

On peut aussi ajouter un attribut **method** qui permet de modifier la façon dont les valeurs sont transmises au serveur (protocole HTTP).

La valeur par défaut est **GET**, qui correspond au fonctionnement vu ci-dessus. L'autre valeur possible est **POST**, qui va transmettre au serveur la requête **sans afficher les champs et leur valeur** dans l'URL. Dans ce cas, le navigateur génère directement une requête http contenant les données, sans limitation de taille.

Dans cette section, vous allez utiliser **netcat** [1], un utilitaire réseau qui permet de simuler des échanges HTTP, à la fois en tant que serveur en affichant les requêtes reçues, ou en tant que client en générant des requêtes. Pour cette partie, vous utiliserez la machine virtuelle Debian 8 que vous avez utilisée dans le module M1105 (Introduction aux OS).

2.1 Deux instances de netcat qui dialoguent

16. Ouvrir deux terminaux dans **C:\temp**, et placer leurs fenêtres côte à côte.
17. Dans le terminal de gauche, créer un serveur qui va écouter des requêtes (l'option **-l** est pour *listen*) sur le port 1234 :

    ```
    nc -p 1234 -l
    ```

18. Dans le terminal de droite, se connecter sur le serveur local, sur le port 1234 :

    ```
    nc localhost 1234
    ```

19. Vérifier que la communication fonctionne : en entrant des caractères dans l'un des terminaux, ils apparaissent dans l'autre fenêtre, après avoir validé. On quitte via CTRL-C.

2.2 Dialogue client-serveur

On va configurer une instance de netcat en serveur de façon à observer la façon dont un navigateur envoie sa requête http.

20. Ouvrir un terminal et une fenêtre Iceweasel (similaire à Firefox) et les placer côte à côte pour pouvoir voir les deux en même temps.
21. Créer le serveur dans un terminal grâce à la commande `nc -k -l -p 1234`
22. En vous aidant de `man netcat` ou de `netcat -h`, expliquer à quoi sert l'option **-k** :

    _____

    _____

1. L'éxécutable s'appelle **nc.exe**

23. Dans Firefox, entrer l'adresse suivante :
`http://localhost:1234/monfichier.html`
Vous pouvez observer dans la console la requête envoyée par le navigateur, composée du mot clé GET suivi du fichier demandé et de la version HTTP, ainsi que de différentes chaînes de type "clé : valeur".

24. Quelle est la signification de la clé **`User-Agent`** ?

_____

2.3 Observation des requêtes de formulaire

Vous allez maintenant observer ce qui est transmis au serveur lors de la validation d'un formulaire. Toujours avec le pseudo-serveur netcat à l'écoute dans une console, vous allez étudier les différences entre GET et POST.

25. Reprendre un des formulaires crées et éditer la balise **`form`** :

```
<form method="get" action="http://localhost:1234/a.html">
```

26. Charger la page et cliquer sur le bouton "envoyer". La console doit afficher la requête du navigateur.

27. Remplacer la valeur de l'attribut **`method`** par **`post`**. Recharger la page dans le navigateur (F5), et valider le formulaire. Le serveur reçoit-il la requête ? Qu'est-ce qui change par rapport à la méthode **`get`** ? :

_____

_____

3 Mise en forme de formulaires

En l'état, les formulaires sont peu esthétiques. On peut améliorer les choses avec des balises sémantiques spécifiques.

28. Reprendre le code de la question 1), le **dupliquer** et le transformer de la façon suivante :

```
<form>
<fieldset>
  <legend>Identification:</legend>
  <label>Nom:</label><input type="text" name="lastname">
  <label>Prenom:</label><input type="text" name="firstname">
  <br>
  <input type="submit" value="Envoyer">
</fieldset>
</form>
```

29. Qu'apporte la balise **`legend`** :

_____

30. Qu'apporte la balise **`fieldset`** :

_____