Formulaires HTML

HTTPS://DEVELOPER.MOZILLA.ORG/FR/DOCS/WEB/GUIDE/HTML/FORMULAIRES/COMMENT_STRUCTURER_UN_FORMULAIRE_HTML

L'élément <form>

```
Form action="/ma-page-de-traitement" method="post"> </form>
```

- Cet élément définit un formulaire.
- C'est un élément conteneur au même titre que les éléments ou <p>, mais il accepte aussi quelques attributs spécifiques afin de contrôler la manière dont il se comporte.
- Tous ses attributs sont optionnels mais définir au moins les attributs action et method est considéré comme une bonne pratique.
  - L'attribut action définit la localisation (une URL) où doivent être envoyées les données collectées par le formulaire.
  - L'attribut method définit la méthode HTTP utilisée pour envoyer les données (cela peut être « get » ou « post »).

Ajoutez des items à l'aide des éléments <label>, <input> et <textarea>

```
<form action="/ma-page-de-traitement" method="post">
<div> <label for="nom">Nom :</label> <input type="text" id="nom" /> </div>
<div> <label for="courriel">Courriel :</label> <input type="email" id="courriel" /> </div>
<div> <label for="message">Message :</label> <textarea id="message"></textarea> </div>
</form>
```

Les éléments sont ici pour structurer notre code et rendre la décoration plus facile (vois ci-dessous).
Veuillez noter l'utilisation de l'attribut for sur tous les éléments <label>.

Concernant l'élément <input>, l'attribut le plus important est l'attribut type. Ce dernier est extrêmement important puisqu'il définit le comportement de l'élément <input>

Un élément <button>

Pour permettre à l'utilisateur d'envoyer les données qu'il fournit dans le formulaire il faut ajouter un élément <button>

```
<form action="/ma-page-de-traitement" method="post">
<div> <label for="nom">Nom :</label> <input type="text" id="nom" /> </div>
<div> <label for="courriel">Courriel :</label> <input type="email" id="courriel" />
</div>
<div> <label for="message">Message :</label> <textarea id="message"></textarea> </div>

<div class="button"> <button type="submit">Envoyer votre message</button> </div>

</form>
```

Un bouton peut être de trois types : submit, reset ou button.

- un bouton « submit » envoie les données du formulaire vers la page définie par l'attribut action de l'élément <form>.
- un bouton « reset » réinitialise tous les blocs du formulaire à leurs valeurs par défaut immédiatement.

Exemple d’utilisation pour tester les WS

```
<form method="get" action="resultat.php">
   <input name="recherche" />
   <input name="options" />
   <input type="submit" value="Envoyer">
</form>
```

http://www.neocorp.com/resultat.php?recherche=valeur1&options=valeur2

A partir de là vous devriez facilement répondre à la question à a première question du 2.2 du TD 3.

CGI-bin

HTTP://WWW.FIRENODE.NET/TECHNO/WGUI.SH?ID=SHELL-CGI

CGI-bin

Les CGI permettent l'interfaçage entre un client web et un serveur web

Si le client est un programme et non un formulaire et que le serveur web exécute un CGI-bin nous sommes

- dans le cadre d’un paradigme de programmation distribué Over Web
- Web Services de type REST

Les CGI peuvent être fait avec n'importe quel langage : PHP, perl... et le shell, bien entendu.

Cela permet de traiter des URLs avec passage de paramètres (GET) ou des message de requête basé sur une URL avec passage de paramètres dans le corps du message (POST)

Les formulaires permettent de générer interactivement depuis une page Web ces deux types de requêtes

Quelques éléments

Les CGI ne sont rien d'autre que des programmes placés dans un répertoire spécial (par défaut sous Apache le répertoire cgi-bin)

leur fonction est de rediriger automatiquement tout affichage non pas sur la sortie standard du serveur mais sur le navigateur distant

Requête HTTP pour une CGI-BIN

lorsqu'on envoie une requête HTTP à un CGI, il y a deux méthodes : La méthode GET et la méthode POST.

Ces deux méthodes sont diffusées de deux manières différentes et donc doivent être récupérées de deux manières différents

CGI-bin sur un GET

La méthode GET

La méthode GET est celle qui est utilisée par défaut lorsqu'on utilise un lien.

C'est aussi la méthode par défaut utilisée lorsqu'on exécute le script en local, en ligne de commande (comme n'importe script UNIX).

Cette méthode utilise une variable QUERY_STRING qui stocke la requête dans une chaîne de caractère ayant un format spécial.

Pour utiliser cette méthode dans un shell, il suffira donc tout simplement d'utiliser la variable ${QUERY_STRING} et de décoder son contenu.

CGI-Bin sur un POST

La méthode POST

La méthode POST envoie directement les données au serveur sur l'entrée standard, sans passer par une variable.

cependant, le codage de la chaîne est identique à celui de la méthode GET, ce qui signifie que le décodage sera identique dans les deux cas.

Au niveau de votre Serveur :

- Pour récupérer la chaîne de la méthode POST, il suffira simplement de faire une lecture de l'entrée standard grâce à la commande read et de la rediriger dans une variable que nous nommerons QUERY_STRING pour des raisons de commodité et de compatibilité avec la méthode GET.

Format de la variable QUERYSTRING

Les CGI-bin positionnent la variable d'environnement $QUERY_STRING en lisant toute la chaine de caractère d'une URL après le " ?"

Par exemple,

- http://exemple.com?a=123&b=456&c=ok
- Correspond à la variable QUERY_STRING=a=123&b=456&c=ok.
- Nous pouvons par exemple afficher les variables transmises avec la commande shell suivante :

- Ainsi
- b=$(echo "$QUERY_STRING" | sed -n 's/^.*b=\([^&]*\).*$/\1/p' | sed "s/%20/ /g")

- Affectera la valeur 456 à la variable shell b

- >echo $b
- >456

Exemple

<form method="POST" action="http://*{votre domaine ou adresse ip}*/cgi-bin/TestForm.sh">

Ce formulaire génére une requête sur l'URL http://*{votre domaine ou adresse ip}:{port}*/cgi-bin/TestForm.sh

Les données saisies dans le formulaire sont alors stockées dans le corps de la requête HTTP.

Le serveur Web à l'adresse *{votre domaine ou adresse ip}:{port}* reçoit ces paramètre sur son entrée standard (stdin)

Pour être compatible avec la réception de GET pour accèder à un cgi-bin, le serveur doit alors lecture de l'entrée standard grâce à la commande read et de la rediriger dans une variable que nous nommerons QUERY_STRING

Question du TP 3 à rendre (simplifions le code Python !!)

Question : En reprenant le serveur Web du TD précédent, que devez-vous rajouter pour implémenter un service Web REST qui implémente une méthode « incr », qui récupère un nombre et retourne son incrément ?

1. Récupérer le code du serveur Web TCP/IP
2. En version préliminaire on peut implémenter une invocation au service REST avec une méthode GET …
3. L'URL http://<serveur>:<port>/<path cgi-bin>/incr.sh?val=<value> est gérée par le client ainis
4. Connexion au serveur <IPserveur>:<port>
5. Envoie la requete HTTP : GET /<path cgi-bin>/incrs.sh?val=<value>,
6. Le serveur prévoit l'exécution de incr.sh présent dans le répertoire $HTTPROOT/<path cgi-bin>/ (attention au droits d'exécution sur incr.sh)
7. $QUERY_STRING est ici égal à « val=<value> » (c'est mon serveur Web qui s'en charge)
8. Le code shell de incr.sh renvoie <value + 1> dans le format attendu par le client (JSON ou whatever)

Question du TP 3 à rendre (simplifions le code Python !!)

Le test peut se faire à partir d'un formulaire

Le client python doit simplement envoyer une requête HTTP GET bien formater contenant >/<path cgi-bin>/incr.sh?val=<value>

Question : Implémentez ce service web et testez le avec un formulaire ou un client python. Quel format de donnée avez-vous choisi ?

Question : Que devrions-nous modifier pour implémenter ce service en mode WS-SOAP ?

Question du TP 3 à rendre

Question : Que devrions-nous modifier pour implémenter ce service en mode WS-SOAP ?

Juste requête POST avec body SOAP (Cf. cours)

Juste réponse avec body SOAP

Correction Question 2.2

Invocation du Web service décrit sur

www.dataaccess.com/webserviceserver/numberconversion.wso?op=NumberToWords en REST/GET

Se fait avec :

www.dataaccess.com/webserviceserver/Numberconversion.wso/UbiNum=3