Objectifs

Comprendre les mécanismes de base d'AJAX. Savoir écrire des applications AJAX simples à la main.

1 Javascript

Afin de se faire la main en JavaScript, Creez une page `html` nommée `ajax1.html` avec un formulaire simple contenant un bouton "non submit" et un champs de saisie de type `Text`. Ecrire une fonction JavaScript `appel()` qui affichera un message via la fonction `alert` lors de son appel. Testez l'appel de cette méthode selon les évènements

— `onclick` sur le bouton
— `onchange` sur l'objet text
— `onblur` sur l'objet text
— `onkeypress` sur l'objet text

Voir notamment `http://www.w3schools.com/jsref/dom_obj_event.asp` pour l'ensemble des évènements possibles.

2 AJAX

1. **Lire et afficher en AJAX un fichier issu du serveur**.
   Recopiez la page précédente dans `ajax2.hml` et modifiez là pour que un click sur le bouton permette par un appel AJAX de charger un fichier texte contenant une simple phrase et afficher cette phrase dans le champs `Text` via son attribut `Value`. Il vous faut donc :
   (a) un fichier `data.txt` avec une simple phrase "Hello World" dedans
   (b) un fichier `ajax2.html` avec un formulaire contenant 2 objets `Input` l'un de type `Button` et l'autre de type `text`. Attention ! le bouton ne doit pas être de type `submit` !
   (c) Dans `ajax2.html` un script Javascript contenant l'instanciation du serveur `XMLHttpRequest`
   (d) ainsi que les 2 methodes Ajax classiques `appel()` appelée `onClick` sur le bouton qui effectue la requete GET sur `data.txt`, et `retour()` la méthode "callback" qui traite la réponse, et donc affecte au champs `Text` le message issu du fichier

2. **Affichage du résultat dans une JSP**.
   Afin de bien visualiser le fait que la page n'est pas complètement rechargée, nous allons y placer un compteur de rechargement. Recopiez la page précédente en `ajax3.jsp` qui contient dans sa partie haute un compteur indiquant le nombre de chargements de cette page.
   ```
   <%! int cpt=0; %>
   <% cpt++; %>
   ```
   Vérifiez qu'un click sur Reload de la page incrémente ce compteur, tandis qu'un click sur le bouton ne l'incrémente pas.

3. Nous faisons maintenant de même avec le fichier `data.txt`. Remplacez ce fichier par `data.jsp` qui contient lui aussi son propre compteur global.
   Cette fois le click sur le bouton incrémente uniquement le compteur dans le textfield, tandis que Reload incrémente uniquement le compteur de la page.

4. **Lire une donnée dans une base**.
   Sur le même principe créez la page `ajax4.jsp` qui affiche `onClick` le nombre de lignes de la table `produits(pno, libelle,prix)` dans une phrase indiquant "La table produit contient actuellement x articles". Cette fois il faut évidemment créer une servlet spéciale pour répondre à l'appel AJAX. On récupère ici simplement les infos en mode texte. Vous testerez cette page en inserant des données dans la table via `psql`.

5. **Rechargements automatiques réguliers**.
   Modifiez votre page `ajax3.jsp` pour qu'un timer lance à intervalles réguliers l'appel Ajax `setinterval(appel,500); \\` devez maintenant voir la page web se modifier automatiquement à chaque modification (insert, update) de la base de données.

6. **Récupération des infos en format XML**
   Sur le meme principe, creez une page `ajax4.html` qui récupère et affiche les informations de l'article le plus cher de la base, affichées dans 3 champs `text` distincts rangés dans un formulaire. La servlet générant la réponse devra renvoyer cette fois une structure xml. On utilisera alors dans le callback l'attribut `responseXML`, ainsi que l'analyse de l'arbre DOM de cette réponse pour récupérer les infos.

7. **Utilisation simple de l'arbre DOM**.
   idem mais cette fois, au lieu de mettre les informations récupérées dans des objets `text`, on souhaite les placer directement en rouge au sein d'une phase. On utilisera pour cela une balise `span` pour identiter le mot à modifier.

8. **XML + DOM**
   idem mais cette fois, on charge dans un `TextArea` les 5 articles les plus chers de la base issus d'une servlet renvoyant une structure xml. Il faut donc récupérer des données XML et analyser l'arbre DOM correspondant pour récupérer chaque information.