## Références

- W3Schools on JavaScript : http://www.w3schools.com/js/
- JQuery : http://jquery.com/

## Exercice 42 – AJAX et JSON avec JQuery

Cet exercice propose de construire le côté client d'une application web, en utilisant notamment la technologie AJAX (avec JQuery) pour appeler un service web de type REST.

L'application web est un gestionnaire de marque-pages (bookmarks) en ligne, un peu comme http://www.stumbleupon.com/. Un utilisateur dispose d'un espace privé où il peut déposer des marque-pages avec un titre, une description, et des mots-clés (tags) attachés. Les marque-pages sont stockés sur un site, qui expose une API REST permettant de les accéder et de les manipuler. Le format d'échange est JSON.

Dans la suite, on vous propose de construire un client HTML+CSS+JavaScript capable de s'interfacer avec le service web distant. On ne s'intéressera qu'aux aspects fonctionnels : toutes les questions liées à la sécurité, la confidentialité, l'authentification, l'optimisation des accès, etc. sont en-dehors du sujet.

On vous donne les fichiers `Exercice-42.html` et `Exercice-42.css` (à utiliser sans les modifier) et le squelette `Exercice-42.js` à remplir selon les indications ci-après.

### Groupes

Dans la suite, un certain nombre d'URLs seront paramétrés par les logins de votre équipe. Vous instancierez la variable `<Groupe>` selon le cas par l'une des chaînes suivantes :

- `eblohoua-mehric-milionlu`
- `bachee-fuchsg-pelletgu`
- `ditishej-ducn-guda-mehennsa`
- `bellasa-kumbhara-kuntzd`
- `brunga-ceccatoj-laodicir-mougnem`

### Objets manipulés

Le service web manipule deux types d'objets : des tags (tags) et des marque-pages (bookmarks). Il ne communique qu'au format JSON.

- Un tag possède un identifiant unique attribué par le système (`id`) et un nom (`name`). En JSON, un tag est représenté par un objet `{ 'id': <id>, 'name': <name> }`. Si l'identifiant n'est pas connu (p.ex. lors de la création d'un nouveau tag, et avant que le système lui attribue son numéro unique), on peut spécifier juste `{ 'name': <name> }`.
- Une liste de tags est représentée par un tableau : `[ tag1, tag2,..., tagn ]`.
- Un marque-page possède un identifiant unique attribué par le système (`id`), un titre (`title`), une URL (`link`), une description optionnelle (`description`) et une liste de tags attachés (`tags`), cette liste pouvant être vide. Sa représentation en JSON est : `{ 'id': <id>, 'title': <title>, 'description': <description>, 'link': <link>, 'tags': <liste-de-tags> }`.
- Enfin, une liste de marque-pages est un tableau : `[ bookmark1, bookmark2,...]`.

### Service web à utiliser

Chaque équipe dispose d'un service web propre, pour travailler en autonomie. Les URLs du service

sont de la forme https://dsi-dev.grenoble-inp.fr/BTM/<Groupe>/<Path>[?Param]. Pour faciliter les tests, le même service est également accessible dans une version non sécurisée (http://dsi-dev.grenoble-inp.fr/BTM/<Groupe>/<Path>[?Param]), mais c'est bien le protocole HTTPS qu'il faudra utiliser dans le TP.

Le paramètre Param permet de spécifier le nom d'une méthode du protocole HTTP sous la forme `x-http-method=<nom-de-methode>`. Cette construction permet de contourner la limitation des navigateurs, qui proposent seulement des requêtes GET et POST : en spécifiant par exemple https://dsi-dev.grenoble-inp.fr/BTM/<Groupe>/<Path>?x-http-method=put dans la barre d'adresse d'un navigateur, la requête GET résultante sera interprétée par le service web comme s'il s'agissait d'une requête PUT.

La composante <Path> des URLs peut prendre plusieurs valeurs, présentées dans le tableau suivant. Les cellules grisées correspondent à des appels interdits (erreur 405). Pour les autres cas, le code de retour attendu est indiqué (CR).

| **<Path>** | **GET** | **POST** | **PUT** | **DELETE** |
|---|---|---|---|---|
| /bookmarks | donne la liste des marque-pages<br>CR 200 | crée un nouveau marque-page, dont la définition aura été passée dans le paramètre json<br>CR 200 | | |
| /bookmarks/<bid> | donne la description du marque-page dont l'ID est <bid><br>CR 200 | | modifie le marque-page avec la nouvelle définition passée en paramètre json<br>CR 204 | efface le marque-page dont l'ID est <bid><br>CR 204 |
| /tags | donne la liste des tags<br>CR 200 | crée un nouveau tag à partir de la définition passée par le paramètre json<br>CR 200 | | |
| /tags/<tid> | donne la description du tag dont l'ID est <tid><br>CR 200 | | modifie le tag avec la nouvelle définition passée en paramètre json<br>CR 204 | efface le tag dont l'ID est <tid><br>CR 204 |

Quelques exemples :

- l'URL https://dsi-dev.grenoble-inp.fr/BMT/<Groupe>/tags?x-http-method=get permet de récupérer la liste de tous les tags
- l'URL https://dsi-dev.grenoble-inp.fr/BMT/<Groupe>/tags/12?x-http-method=get donne la description du tag dont l'identifiant unique est « 12 »
- l'URL https://dsi-dev.grenoble-inp.fr/BMT/<Groupe>/tags/12?x-http-method=put &json={'id':12,'name':"toto"} permet de modifier le nom du tag « 12 » à « toto »
- l'URL https://dsi-dev.grenoble-inp.fr/BMT/<Groupe>/tags/12?x-http-method=delete efface le tag « 12 »

En outre, pour faciliter les tests, deux autres URLs sont fournies :

- https://dsi-dev.grenoble-inp.fr/BMT/<Groupe>/clean?x-http-method=post permet d'effacer tous les tags et tous les marque-pages
- https://dsi-dev.grenoble-inp.fr/BMT/<Groupe>/reinit?x-http-method=post permet de réinitialiser les tags et les marque-pages (quelques objets sont fournis par défaut)

## Espace de travail

Une autre limitation des navigateurs, appelée Same origin policy, empêche d'interroger par JavaScript un site différent de celui dont provient le JavaScript lui-même. Autrement dit, si vous utilisez un fichier JavaScript local (dont l'URL commence donc par file:///...), votre navigateur vous empêchera d'appeler le web-service à l'adresse http(s)://dsi-dev.grenoble-inp.fr/BMT/...

C'est pourquoi on vous fournit un espace de travail spécifique sur ce même serveur dsi-dev.grenoble-inp.fr, où vous pourrez déposer vos fichiers et tester leur fonctionnement dans un navigateur.

L'URL de votre espace de travail est https://dsi-dev.grenoble-inp.fr/dav/<Groupe> ; p.ex. si vous y déposez votre fichier Exercice-42.html, vous pourrez l'accéder à l'adresse https://dsi-dev.grenoble-inp.fr/dav/<Groupe>/Exercice-42.html.

Pour déposer les fichiers, vous pouvez vous connecter[1] avec le protocole WebDAV sécurisé (WebDAVS) :

| | |
|---|---|
| **serveur** | dsi-dev.grenoble-inp.fr |
| **port** | 443 |
| **chemin** | /dav/<Groupe>/ |
| **URL entière** | https://dsi-dev.grenoble-inp.fr/dav/<Groupe>/ |
| **login** | votre login habituel |
| **mot de passe** | votre mot de passe Agalan (=Ensimag) |

## Bibliothèques JavaScript fournies

On vous fournit les bibliothèques JQuery et Json2. Sur cette dernière, il suffit de savoir que l'appel `JSON.stringify(obj,null,2)` permet de transformer un objet JavaScript `obj` en JSON (avec indentation de 2 caractères), et que `JSON.parse(json)` fournit la transformation inverse.

1 cf. par exemple http://wiki.alwaysdata.com/wiki/Se_connecter_avec_WebDAV

## 1. Identification de l'équipe

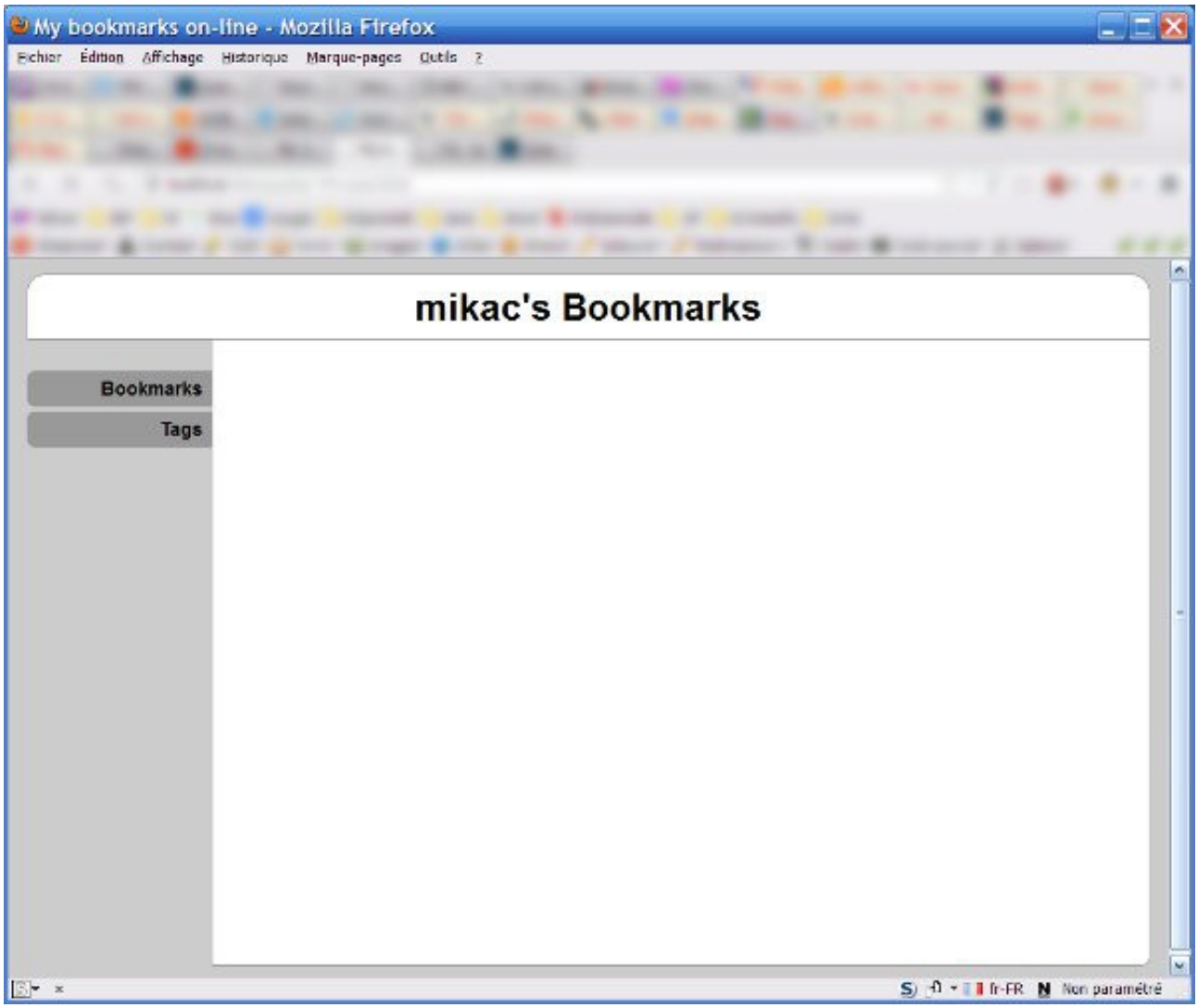
Pour préparer la suite, instanciez la variable `wsBase` dans le fichier `Exercice-42.js` avec le code de votre équipe.

Implémentez la fonction `setIdentity()` qui est appelée au chargement du document : elle doit extraire le nom de votre équipe de la variable `wsBase` et l'afficher dans l'élément `<span class="identity">` qui est à l'intérieur de `<h1>`. De cette manière, l'équipe apparaît dans le titre de la page.

*Indications* : la méthode split() des objets String

## 2. Adapter la hauteur de la page

Implémentez la fonction `setContentHeight()` pour agrandir verticalement l'élément `<div id="contents">` de manière à ce qu'il prenne toute la place restante dans la fenêtre du navigateur :




*Indications* : méthodes JQuery offset() et height()

## 3. Gestion du type d'objet

Implémenter la fonction `selectObjectType(type)`. Cette fonction sera appelée pour changer le type d'objets affichés : soit lors d'un clic sur un item du menu, soit directement lors du chargement de la page.

Le paramètre `type` peut prendre deux valeurs : `bookmarks` ou `tags`. Le type couramment affiché (s'il existe) est celui qui correspond à la ligne du menu (`<ul id="menu">`) qui porte la classe `selected`.

La fonction `selectObjectType(type)` doit donc fonctionner comme ceci :

- si le type demandé est le type couramment affiché, rien ne se passe
- sinon,
  - il faut enlever la classe `selected` de l'item du menu qui la porte, pour la transmettre à l'item qui correspond au type demandé
  - si le type demandé est `bookmarks`, il faut appeler la fonction `listBookmarks()` et enlever la classe `selected` du `<div class="tag">` qui se trouve dans le `<div id="add">`
  - sinon si le type demandé est `tags`, il faut appeler la fonction `listTags()` et ajouter la classe `selected` au `<div class="tag">`

Si la fonction marche correctement, on voit qu'au chargement du document, l'item Bookmarks est sélectionné, et que les clics sur les items du menu transfèrent la sélection sur l'item choisi. En plus, si les Tags sont sélectionnés, un petit formulaire New tag apparaît.

## 4. Affichage des bookmarks

Implémenter la fonction `listBookmarks()` :

- vider `<div id="items">` de son éventuel contenu
- charger la liste des marque-pages en JSON depuis le web-service (URL .../bookmarks)
- parser la chaîne JSON pour récupérer un objet JavaScript
- parcourir la liste des marque-pages, et pour chacun
  - créer une copie de `<div class="model bookmark">`
  - insérer les propriétés du marque-page dans cette copie : `title` dans `<h2>`, `link` dans `<a>`, description dans `<div class="description">` et les `tags` dans `<ul class="tags">` (un tag par `<li>`)
  - ajouter l'attribut `num` au `<div>` représentant le marque-page, avec pour valeur l'`id` du bookmark
  - enlever la classe `model` et la remplacer par la classe `item`
  - ajouter cet item dans `<div id="items">`

Ainsi, p.ex. l'objet `{'id':12, 'title':'Toto', 'link':'file:///toto.txt', 'tags':[{'id':3, 'name':'one'},{'id':4, 'name':'two'}]}` donnera

```
<div id="items">
  ...
  <div class="item bookmark" num="12">
    <h2>Toto</h2>
    <a href="file:///toto.txt">file:///toto.txt</a>
    <div class="description"></div>
    <ul class="tags">
      <li>one</li>
      <li>two</li>
    </ul>
  </div>
  ...
</div>
```

Si la fonction est correctement implémentée, vous voyez au chargement du document la liste des marque-pages :


*Indications* : méthodes JQuery clone() et jQuery.get()

## 5. Affichage des tags

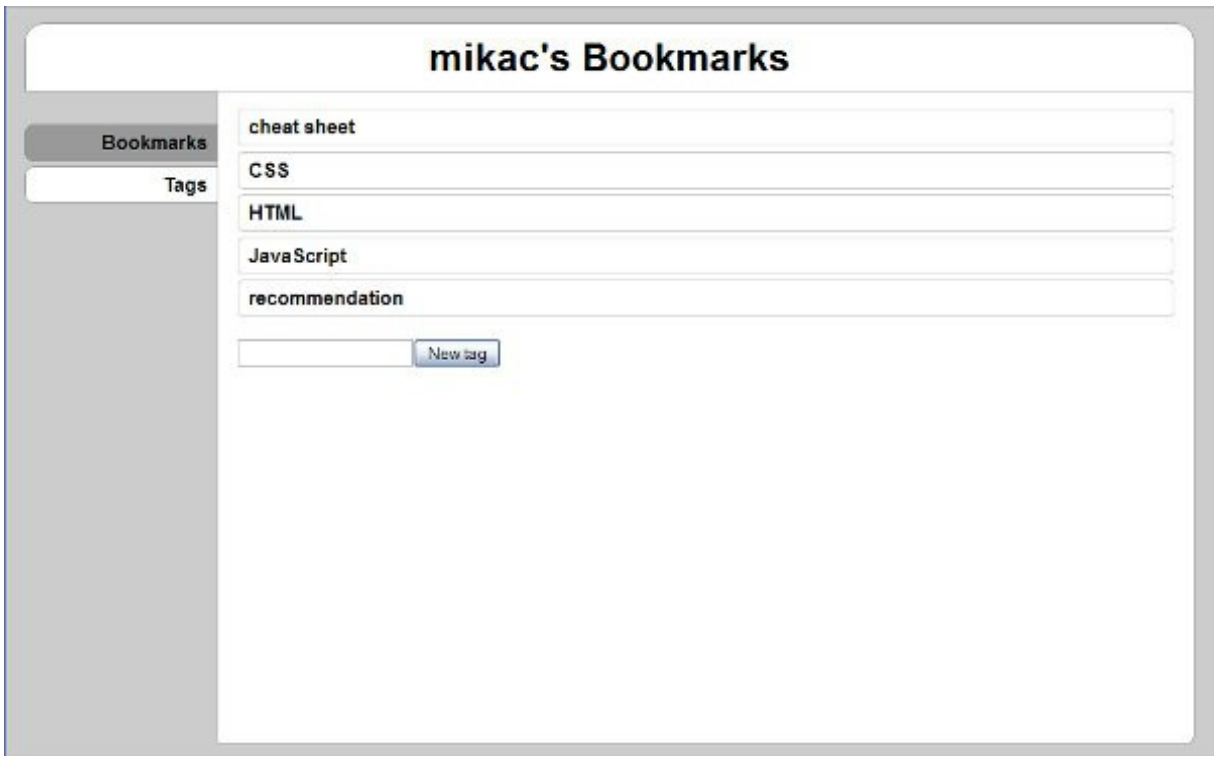
Implémenter la fonction `listTags()` :

- vider `<div id="items">` de son éventuel contenu
- charger la liste des tags en JSON depuis le web-service
- parser la chaîne JSON pour récupérer un objet JavaScript
- parcourir la liste des tags, et pour chacun
  - créer une copie de `<div class="model tag">`
  - insérer le nom (`name`) du tag dans l'élément `<h2>`
  - ajouter l'attribut `num` au `<div>` représentant le tag, avec pour valeur l'`id` du tag
  - enlever la classe `model` et la remplacer par la classe `item`
  - ajouter cet item dans `<div id="items">`

Ainsi, p.ex. l'objet `{'id':3, 'name':'one'}` donnera

```
<div id="items">
  ...
  <div class="item tag" num="3">
    <h2>one</h2>
  </div>
  ... </div>
```

Si la fonction est bien implémentée, on peut voir la liste des tags :



## 6. Ajout de tags

Implémenter la fonction `addTag()` qui est appelée lorsqu'on active le bouton New tag :

- vérifier que qu'un nom de tag est bien spécifié dans le champ `<input type="text" name="name">` : si ce n'est pas le cas, afficher un message d'erreur et terminer
- sinon, appeler la création de nouveau tag dans le web-service, en lui passant le nom du tag à créer sous forme d'objet JSON (dans le paramètre `json`)
- peu importe si l'appel réussit (code-retour 200) ou non[2] (code 4xx ou 5xx), appeler `listTags()` à la réception de la réponse

Si la fonction est correcte, elle permet donc de rallonger la liste des tags avec de nouveaux items.

## 7. Édition de tags

Implémenter la fonction `clickTag()` qui est appelée à chaque clic sur un item représentant un tag :

- si le tag cliqué était déjà sélectionné (ie s'il portait la classe `selected`), ne rien faire
- sinon
  - si un autre item était sélectionné, le désélectionner (enlever la classe `selected` et défaire toute autre modification, cf. ci-dessous)
  - pour l'item sélectionné
    - ajouter la classe `selected`
    - cacher son nom (`<h2>`)
    - lui ajouter un champ de saisie contenant son nom actuel, un bouton permettant de valider la modification et un bouton permettant de supprimer le tag. P.ex. voici le tag HTML avant et après sélection :


    - assurer que lors d'un clic sur le bouton de modification, la fonction `modifyTag` sera appelée
    - idem pour le bouton de suppression et la fonction `removeTag`

## 8. Modification de tag

Implémenter la fonction `modifyTag()` :

- déterminer la tag à modifier
- appeler le service de modification, en lui passant les bons paramètres
- à la réception de la réponse, appeler la fonction `listTags()`

## 9. Suppression de tag

Implémenter la fonction `removeTag()` :

- déterminer la tag à supprimer

2 La création d'un tag va échouer s'il existe déjà un tag portant le même nom.

- appeler le service de suppression, en lui passant les bons paramètres
- à la réception de la réponse, appeler la fonction `listTags()`

## 10. Suppression de bookmark

En vous inspirant des points 7 et 9, implémentez de manière analogue la suppression d'un bookmark.

## 11. Ajout de bookmark

Implémentez une manière d'ajouter des bookmarks.

## 12. Modification de bookmark

Implémentez une manière de modifier des bookmarks existants.