

Découverte du Framework jQuery Mobile en autonomie pour le contexte GSB (PARTIE 4)

Description du thème

Propriétés	Description
Intitulé long	Découverte en autonomie du Framework jQuery Mobile avec le contexte GSB dans sa partie gestion des rapports de visite
Formation concernée	BTS SIO option SLAM
Matière	SLAM4 ou phase préparatoire de PPE3 et PPE4
Présentation	Accompagnement dans la découverte de jQuery Mobile ; des sites de références sont proposés afin de développer pas à pas une application à partir du contexte GSB
Notions	<ul style="list-style-type: none">• D4.1 - Conception et réalisation d'une solution applicative• D4.2 - Maintenance d'une solution applicative Savoir-faire <ul style="list-style-type: none">• Programmer un composant logiciel• Exploiter une bibliothèque de composants• Adapter un composant logiciel• Valider et documenter un composant logiciel• Programmer au sein d'un framework
Prérequis	Les principes du développement web, PHP, SQL
Outils	SGBD MySql, un environnement de développement
Mots-clés	GSB, jQuery Mobile, Ajax
Durée	10 heures
Auteur(es)	Patrice Grand, Gildas Spéno relectrice attentive et éclairée
Version	v 1.0
Date de publication	Mars 2016

Quatrième et dernière partie de cette découverte approfondie de jQuery Mobile qui aborde la création dynamique, dans le code jQuery, de *widgets* ainsi que quelques points de sécurité.

Création d'un rapport de visite

Dernière étape ; la création d'un nouveau rapport de visite. C'est certes un peu long mais cela ne fait pas appel à des notions nouvelles.

On désire, à partir du menu « mes visites », créer un nouveau rapport.

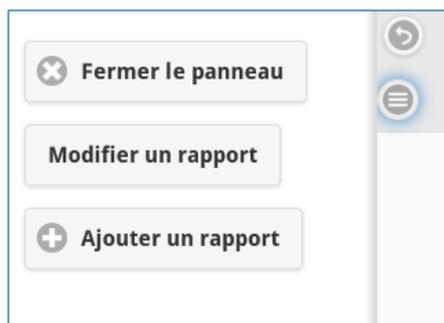


Fig 38

Cette création de rapport contient des champs de recherche et de saisies :

La partie haute du formulaire contient la recherche du médecin et la saisie des motifs, bilan et date ; nous avons privilégié le mode tablette ici encore :



Gestion des visites et des médecins

Ajouter un rapport

Rechercher le médecin

Q Nom...

Nom médecin

Motif

Bilan

Date

Fig 39

La partie inférieure permet d'ajouter les médicaments offerts, ainsi que leurs quantités :

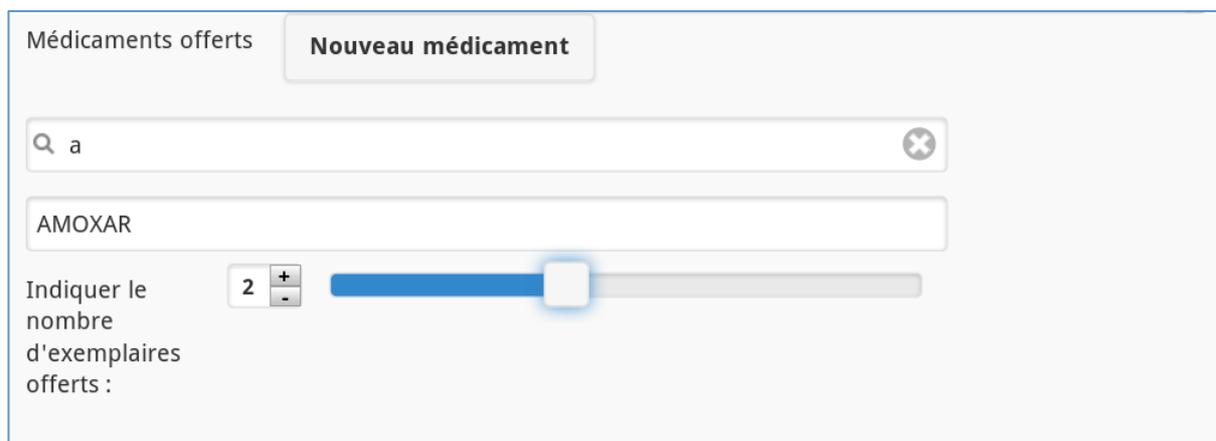


The screenshot shows a web form with the following elements:

- A 'Date' input field.
- A 'Médicaments offerts' section containing a 'Nouveau médicament' button.
- A search dropdown menu with the letter 'a' entered. The dropdown list contains the following items:
 - ADIMOL
 - AMOPIL
 - AMOXAR
 - AMOXI Gé

Fig 40

Lorsque l'on clique sur nouveau médicament, un champ de recherche du médicament s'ouvre ; on peut, après avoir choisi le médicament, indiquer la quantité offerte :



The screenshot shows the following elements:

- A search bar with 'a' entered and 'AMOXAR' selected.
- A label 'Indiquer le nombre d'exemplaires offerts :'. Below it is a numeric input field showing '2' and a slider control.

Fig 41

Le visiteur peut sélectionner plusieurs médicaments et leurs quantités.

Le formulaire se termine par l'enregistrement du rapport de visite.

La partie médecin

a) Côté HTML

Il faut ajouter une nouvelle page *ajouterrapport.php* dans laquelle figurent le champ de recherche et son couple *label/input* associé, ainsi que les 3 autres couples *label/input* pour le bilan et le motif de type *textarea*, et un dernier couple pour la date, *label* et *input* de type *date*.

Travail à faire

37. Ajouter cette nouvelle page avec les éléments HTML évoqués ci-dessus.

b) Côté jQuery

Pour mettre en œuvre le champ de recherche du médecin, le code est le même que dans la page *médecins* dans sa première version sans champ caché.

c) Côté PHP

Vous utilisez la même page PHP que pour la page *médecins*.

Travail à faire

38. Traiter la partie jQuery et PHP. Tester.

La partie médicaments

C'est un peu plus délicat car le visiteur peut ajouter à volonté de nouvelles zones de recherche de médicament et la quantité ; ainsi, tout doit être écrit en jQuery de manière dynamique.

a) Côté HTML

Peu de chose :

```
<label for="lblmedicament">Médicaments offerts</label>
<a href="#" data-role="button" id="btnajoutmedicament" data-inline="true" > Nouveau médicament</a>
<div id="lesListesMedicaments" class="ui-field-contain">
</div> <!-- /fin ui-field-contain les listes médicaments-->
```

Fig 42

La *div lesListesMedicaments* va recevoir tout le code jQuery dynamique.

Travail à faire

39. Compléter la page *ajouterrapport.php* avec les éléments HTML évoqués ci-dessus.

b) Côté jQuery

Il faut d'abord intervenir sur le "click" du bouton « nouveau médicament »

```
$("#pageajouterrapport #btnajoutmedicament").bind("click",function(){...
```

Observons d'abord ce que doit rendre le code jQuery lorsque l'on ajoute un nouveau médicament ; FireBug nous permet de visualiser le code de la source :

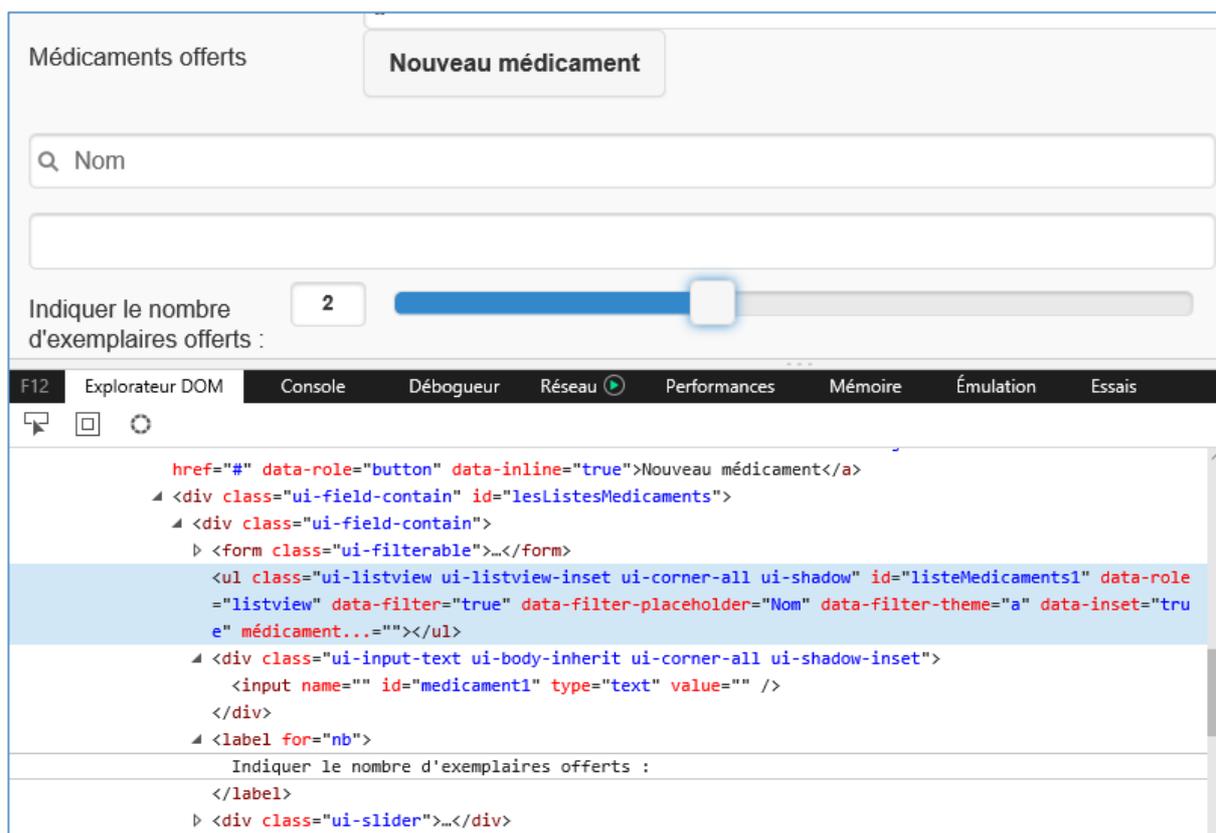


Fig 43

La zone surlignée en bleu montre la zone de liste (*listview*) qu'il faudra générer en jQuery ; notez l'*id* pour cette première liste demandée par le visiteur. La zone de texte est également identifiée de manière dynamique, *medicament1*. À la dernière ligne apparaît le *widget (slider)* qui permet de définir la quantité de ce médicament offert (pas encore déplié ici).

Ainsi, il faudra :

- générer une liste (*listview*), *data-filter=true* avec un *id* dynamique ;
- ajouter la zone de texte ;
- ajouter le *widget slider*.

Pour créer ces éléments, **qui n'existaient pas dans la page HTML d'origine**, il faudra appeler un événement de création à la place de la méthode *refresh* déjà utilisée :

```
$("#pageajouterrapport #lesListesMedicaments").trigger( "create");
```

Par ailleurs, il faudra déclarer une variable à portée globale afin de mémoriser le nombre de médicaments offerts : *window.nbMedicaments* et l'incrémenter à chaque demande d'ajout de médicament.

Pour le nouveau widget, il faudra aller sur le site officiel de jQuery ou consulter le PDF en annexe.

On vous fournit le début du code :

```
$("#pageajouterrapport #btnajoutmedicament" ).bind("click",function(){
    window.nbMedicaments ++ ;
    var i = window.nbMedicaments;
    var html ="<div class=ui-field-contain>";
    html += "<ul id=listeMedicaments" + i + " data-role=listview ";
    html += " data-filter-theme=a data-filter-placeholder=Nom médicament... ";
```

Travail à faire

40. Réaliser ce qui est détaillé ci-dessus. Tester avec plusieurs médicaments.

La deuxième partie jQuery concerne le chargement des médicaments dans la *listview*, qui a lieu sur l'événement *filterablebeforefilter* que vous avez déjà rencontré :

```
$("#pageajouterrapport #lesListesMedicaments").on( "filterablebeforefilter","ul", function (e, data ){...
```

On vous fournit le code commenté de cette méthode un peu délicate :

```
$("#pageajouterrapport #lesListesMedicaments").on( "filterablebeforefilter","ul", function (e, data ){
    var idul = e.currentTarget.id; // on récupère l'id de l'ul grâce à la classe event (e)
    window.ul = idul; // on stocke cet id pour être utilisé dans la fonction de retour
    var nommedicament = data.input.val();// on récupère la saisie
    if(nommedicament && nommedicament.length >=1){
        $.post("ajax/traiterrechermedicaments.php",{
            "nommedicament" : nommedicament
        }, foncRetourRechercheMedicaments,"json" );
    }
});
```

Fig 44

La page PHP associée *traiterrecherchemedicaments.php* reprend les mêmes objectifs que la recherche de médecins. Cette page appelle une fonction *pdo* :

```
$lesMedicaments = $pdo->getLesMedicaments($nomMedicament);
```

La fonction de retour *foncRetourRechercheMedicaments* n'offre pas de difficultés particulières :

- parcours des médicaments ;
- pour chaque médicament, ajout d'une balise *li* + *HREF* qui contient le nom du médicament ;
l'*id* de la balise *li* est l'*id* du médicament ;
- après la boucle, on met le code HTML dans la bonne *ul* !! celle dont l'*id* a été stocké en global ;
- ne pas oublier de rafraîchir la *listview*.

Travail à faire

41. Réaliser ce qui est détaillé ci-dessus. Tester avec plusieurs médicaments.

Nous avons presque terminé. Il nous reste à sélectionner le médicament et le copier dans la zone de texte prévue à cet effet. Cette action est réalisée lorsque l'on clique sur un élément de la *listview* des médicaments, figures 40 et 41 :

```
$("#pageajouterrapport #lesListesMedicaments").on("click", "li", function(e,data) {...
```

La difficulté est de mettre le médicament courant dans la zone de texte qui lui est adjacente, enfin presque !! Observez le code HTML (transformé par jQuery) et sa structure du DOM :

```
<ul class="ui-listview ui-listview-inset ui-corner-all ui-shadow" id="listeMedicaments1" data-rc
data-filter-placeholder="Nom" data-filter-theme="a" data-inset="true" médicament...="">
  <li class="ui-first-child" id="ADIMOL9">...</li>
  <li id="AMOPIL7">...</li>
  <li id="AMOX45">...</li>
  <li id="AMOXIG12">...</li>
  <li class="ui-last-child" id="APATOUX22">...</li>
</ul>
<div class="ui-input-text ui-body-inherit ui-corner-all ui-shadow-inset">
  <input name="AMOX45" id="medicament1" type="text" value="" />
```

Fig 45

L'événement "click" est relatif à chaque balise *li* ; par rapport à chaque balise *li* (*this* dans le code de la méthode *on*), l'*input text* est le premier *fils* (la dernière ligne du code) de l'élément le plus proche de son *père*. On vous rappelle les syntaxes suivantes :

```
$(selecteur).next() retourne le premier voisin de même niveau
```

`$(selecteur).children()` retourne son premier fils dans le DOM

`$(selecteur).parent()` retourne le père

Par exemple :

`$(selecteur).children().parent()` retourne le sélecteur lui-même.

La méthode *on*, dont la signature est fournie plus haut, doit donc :

- récupérer l'*id* et le champ *text* de la balise *li* courante ;
- récupérer le sélecteur de l'*input text* associé ;
- mettre la valeur de l'*id* dans la propriété *name* de l'*input text* et dans sa *value* le *champ text* de la balise *li* courante (cf fig 45) ;
- vider la *listview*.

Travail à faire

42. Réaliser ce qui est détaillé ci-dessus.

Enregistrement du rapport

C'est la dernière étape ; sur le "click" du bouton *Enregistrer* figurant en bas de la page de création d'un rapport, il faudra mener une requête Ajax après avoir récupéré les données saisies.

`$("#pageajouterrapport #btnenregistrerrapport").bind("click",function){...`

Il n'y a pas de difficultés particulières ; seulement peut-être la découverte de la gestion des tableaux d'objets en JavaScript puisque les médicaments offerts seront *passés* sous forme de *tableau d'objets*. Plusieurs sites traitent de cette question ; plusieurs syntaxes sont proposées, néanmoins vous pouvez utiliser :

- déclaration et construction d'un tableau
`var lesMédicaments = [] ;`
- comment on déclare et valorise un objet
`var unMédicament = { id : idMédicament , qte : laQte} ;`
- comment on ajoute un objet dans un tableau
`lesMédicaments.push(unMédicament) ;`

Travail à faire

43. Écrire la méthode *bind*, la page PHP appelée par Ajax et le code *pdo* pour les méthodes *insert*.

Remarque :

Vous pouvez utiliser un outil de débogage côté serveur en utilisant la fonction *error_log* de PHP.

Ainsi l'appel, `error_log($req,3,"log.php")`

permet d'enregistrer dans le fichier *log.php* le contenu de la variable *\$req*.

Pour terminer, un peu de sécurité

Nous devons sécuriser un minimum l'application afin que quiconque ne puisse accéder aux informations sur le serveur, s'il n'est pas authentifié comme visiteur. Chaque page PHP, appelée par une requête Ajax, doit ainsi vérifier que l'appel provient de quelqu'un qui est connecté.

Rappelez-vous qu'à la connexion, le visiteur est enregistré en session (figure 18) ; voici le code pour un fichier PHP :

```
<?php
session_start();
if(isset($_SESSION['visiteur'])){
    require_once '../data/pdogsbrapports.php';
    $pdo = PdoGsbRapports::getPdo();
    $idRapport = $_REQUEST['idRapport'];
    $leRapport = $pdo->getLeRapport($idRapport);
    echo json_encode($leRapport);
}
else {
    echo 0;
}
?>
```

Fig 46

Travail à faire

44. Procéder de la sorte pour tous les fichiers PHP.

Ces modifications n'empêcheront pas un utilisateur de lancer l'application, mais aucune donnée ne lui sera retournée.

Si nous voulons, en plus, empêcher l'accès aux pages, nous pouvons demander à jQuery, à chaque demande de page, de lancer une requête Ajax pour vérifier si le visiteur est connecté.

1) Côté jQuery

```
/*-----Toutes les pages----- */
$(document).on("pageinit", function(e) {
    var page = window.location.hash.substr(1); // récupère la partie après #
    if(page!=""){
        $.get("ajax/traiterdemandepage.php",
            foncRetourConnecte );
    }
});
function foncRetourConnecte(data){
    if(data!=1)
        $.mobile.changePage("#");
}
```

Fig 47

2) Côté PHP

Le fichier *traiterdemandepage.php* :

```
<?php
    session_start();
    echo isset($_SESSION['visiteur']);
?>
```

Fig 48

Bien évidemment, ce dernier traitement côté jQuery peut être contourné puisque l'utilisateur dispose du code...

D'autres contrôles peuvent être envisagés ; s'assurer que le visiteur qui modifie un rapport en est bien le propriétaire (simple) ou éviter qu'un visiteur ne se connecte plus d'une seule fois en même temps (plus délicat).

Ceci termine la présentation de jQuery Mobile ; le corrigé se trouve dans le répertoire **CorrigebsbMobileFinal**.