Troisième partie : développement de la page formCR_VISITE

La maquette fournie initialement (`formRAPPORT_VISITE.html`) a été retravaillée. Le résultat (`formCR_VISITE.php`) se trouve dans le dossier de ressources de cette partie du projet.

1ère demande : afficher le formulaire de création d'un nouveau compte-rendu de visite

1 Adaptation du code

Vous devez dans un premier temps mettre à jour le lien dans `entete.inc.php` et modifier `index.php` : la première demande consiste en l'affichage de la nouvelle page.

La fonction `formSelectDepuisRecordset()` retourne le code HTML d'une liste déroulante fabriquée à partir d'un recordset de type `PDOStatement`. Il serait encore plus intéressant de disposer d'une fonction qui effectuerait le même traitement mais à partir d'un tableau à 2 dimensions (un tableau de tableaux associatifs).
Ce qui autoriserait une totale souplesse dans le choix de la source de données (pensons à l'architecture MVC …) : cette fonction pourrait ainsi aussi bien fabriquer le code de la liste déroulante des motifs de visite à partir d'une liste des motifs fournie, que celui de la liste des praticiens à partir de la liste des praticiens de la base de données. Autre avantage ici : 2 listes déroulantes des praticiens sont utilisées. Elles devraient être fabriquées à partir du même jeu d'enregistrements. Or par défaut, après un `fetch()` on ne peut pas revenir en arrière (pour pouvoir le faire, il faut préparer préalablement la requête avec *`PDO::CURSOR_SCROLL`*). En utilisant des tableaux, on évite donc ce problème.

Créez une fonction nommée `formSelectDepuisTab2D()` qui effectue le même traitement que `formSelectDepuisRecordset()`, et accepte les mêmes paramètres, ormis le jeu d'enregistrements qui devra être remplacé par un tableau de tableaux associatifs et un paramètre indiquant si le contrôle est `disabled` ou pas.

Créez une fonction nommée `formInputCheckBox()` qui retourne le code HTML d'une zone de saisie de type case à cocher, et qui prend en paramètre :

- un booléen indiquant si la case sera affichée avant son label ou après,
- le label de la case,
- le nom de la case,
- son id,
- un booléen indiquant si la case est cochée ou pas.
- son index de tabulation,
- un booléen indiquant si la case est en lecture seule.

Créez une fonction nommée `formInputNumber()` qui retourne le code HTML d'une zone de saisie de type zone numérique, et qui prend en paramètre :

- le label de la zone de saisie,
- le nom de la zone,
- son id,
- sa valeur minimale,
- sa valeur maximale,
- un pas de variation,
- son index de tabulation,
- un booléen indiquant si la zone est en lecture seule.
- un booléen indiquant si la saisie est obligatoire.
- un texte indicatif (attribut `placeholder`).

Modifiez la fonction nommée `formInputSubmit()` de votre bibliothèque : il faut ajouter un paramètre booléen qui indique si le bouton est en lecture seule (`readonly`). Sa valeur par défaut sera FALSE, et il faudra tenir compte de sa valeur dans le code HTML à générer.

Créez une fonction nommée `formInputReset()` qui retourne le code HTML d'un bouton de type `reset`, et qui prend les mêmes paramètres que la fonction `formInputSubmit()`.

Créez un fonction `getListePraticiensTab()` qui retourne les données sous forme de tableau.

Testez votre nouvelle page web. Les problèmes dans la liste des produits présentés et dans celle des échantillons distribués seront corrigés plus tard.

2 Les traitements côté client

Nous utiliserons la bilbiothèque jQuery. Installez et référencez dans votre page la version de production compressée légère (*compressed, production jQuery x.x.x slim build*), comme cela a été expliqué en cours au chapitre "Introduction à jQuery".

Il faut pouvoir aussi référencer dans l'entête de la page web un fichier JavaScript spécifique. Modifiez le fichier `entete.inc.php` pour qu'il teste si une variable nommée `$jsFichier` existe et si c'est le cas qu'il ajoute dans l'entête une balise `<script>` qui référence le fichier dont le nom est stocké dans cette variable.

Le code côté client (spécifique à la page web) sera développé dans un fichier nommé `crVisite.js`. Stockez ce nom dans la variable `$jsFichier` avant d'effectuer l'include de l'entête.

Ajoutez ensuite un fichier JavaScript nommé `crVisite.js`. à votre projet.

2.1 Le motif de la visite

Lorsque le motif de la visite "Autre" est sélectionné et uniquement dans ce cas, la zone de saisie à droite de la liste déroulante doit être activée et doit recevoir automatiquement le focus (l'explication du motif "Autre" sera par la suite enregistrée dans la colonne `VISITE_MOTIF_EXPLICATION` de la table `VISITE`).

Si l'utilisateur a saisi une explication, puis choisit un autre motif (donc différent de "Autre"), l'explication doit être automatiquement effacée, et la zone de saisie doit être désactivée (voir la propriété `disabled`).

C'est l'événement `onchange` de la liste déroulante qui nous intéresse ici, donc `change()` dans jQuery …

Dans `crVisite.js` créez le gestionnaire de l'événement `ready` : dans ce gestionnaire d'événement vous créerez le gestionnaire de l'événement `change` de la liste déroulante. Il appellera une fonction nommée `rendreDisponibleExplicationMotif()` qui devra recevoir en paramètre le formulaire de compte-rendu. Celui-ci devra lui être passé sans utiliser JQuery.

Implémentez cette fonction dans le même fichier. Elle ne nécessite pas d'utiliser jQuery.

Testez votre code.

2.2 Le praticien remplaçant

La liste déroulante permettant de sélectionner un médecin remplaçant doit être activée ou désactivée en fonction du changement d'état de la case à cocher "REMPLAÇANT".

Procédez comme précédemment :
- créez le gestionnaire de l'événement `change()` de la case à cocher. Il appellera une fonction nommée `rendreDisponibleListeRemplacants()` qui devra recevoir en paramètre le formulaire de compte-rendu. Celui-ci devra lui être passé sans utiliser JQuery.

- Implémentez cette fonction. Elle ne nécessite pas d'utiliser jQuery.

Testez votre code.

2.3 Les produits présentés

La liste des médicaments pour les produits présentés (et les échantillons distribués) sera créée via la fonction `formSelectDepuisTab2D()`. Une requête SQL devra retourner les données sous la forme d'un tableau de tableaux dont les éléments pourront être désignés par une clé ou par leur indice (voir `PDO::BOTH` dans la documentation de PDO) …

Dans la maquette de la page, on constate que des listes numérotées (`<ol>`) sont utilisées pour afficher les produits présentés et les échantillons distribués. Les contrôles de saisie n'ont donc pas de label.

Modifiez les fonctions `formSelectDepuisTab2D()` et `formInputNumber()` afin que si une valeur nulle est passée en tant que label, le code HTML du label ne soit pas généré.

De la même manière, dans ces deux listes les contrôles de saisie n'ont pas d'id (il sont regroupés et ont donc un `name` particulier).

Modifiez les fonctions `formSelectDepuisTab2D()` et `formInputNumber()` afin que si la valeur nulle est passée en tant qu'id, le code HTML de l'id ne soit pas généré.

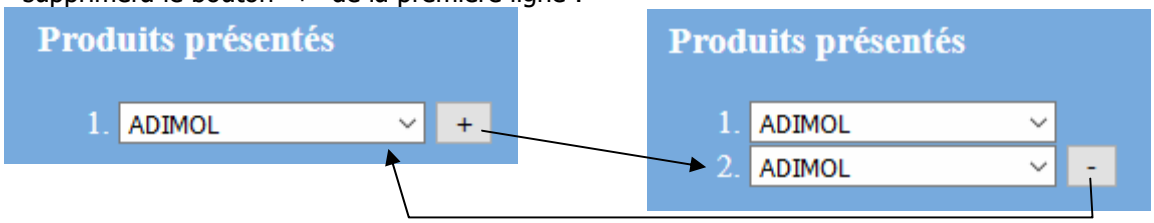
Créez une fonction nommée `formButton()` qui retourne le code HTML d'un bouton (Cf. la balise `<button>`), et qui prend en paramètre :

- le nom du bouton,	- sa valeur,	- son index de tabulation,
- son id,	- son texte	- un booléen indiquant si le bouton est en lecture seule.

Si une valeur nulle est passée en tant qu'id, le code HTML de l'id ne doit pas être généré.

Dans la ligne de saisie du premier produit présenté, le bouton "+" n'a pour l'instant aucun gestionnaire d'événement. Dans le fichier `crVisite.js` ajoutez-en un pour l'événement `click` du bouton (méthode `click()` de jQuery). Il appellera une fonction nommée `ajouterLigneProduitPresente()` qui devra recevoir en paramètre la première et pour l'instant la seule ligne de produit présenté (sélectionnée via jQuery).

Cette fonction :
- ajoutera à la liste des produits présentés une deuxième ligne dotée d'un bouton "-" permettant de la supprimer,
- supprimera le bouton "+" de la première ligne :



Récupérez et intégrez à votre projet le fichier `Bibliotheque01.js` fourni par votre chef de projet, qui contient la fonction `jqFormBouton()`.

Créez ensuite dans le fichier `crVisite.js` la fonction `ajouterLigneProduitPresente()` à l'aide de jQuery (et de sa documentation). Elle doit :
- supprimer le bouton "+" de la dernière ligne (`remove()`) ;
- copier la dernière ligne dans une variable mémoire (`clone()`) qui sera la base de la nouvelle ligne ;
- renommer la liste déroulante de la nouvelle ligne (`attr()`). Rappel : on utilise des groupes de contrôles de saisie ;
- ajuster la valeur de l'attribut `tabindex` de la liste déroulante (`attr()`) : elle doit être égale à sa valeur actuelle (qu'il faut lire) incrémenté de 1 ;
- créer un bouton "-", dont l'id est `btnSupprimerPP`. L'attribut `tabindex` aura pour valeur celle de la liste déroulante incrémentée de 1 ;

- ajouter au bouton le gestionnaire de l'événement click qui appellera une fonction nommée `supprimerLigneProduitPresente()` à laquelle il passera la liste des produits présentés.
- ajouter ce bouton à la nouvelle ligne (`append()`) ;
- et enfin ajouter la nouvelle ligne à la liste des produits présentés.

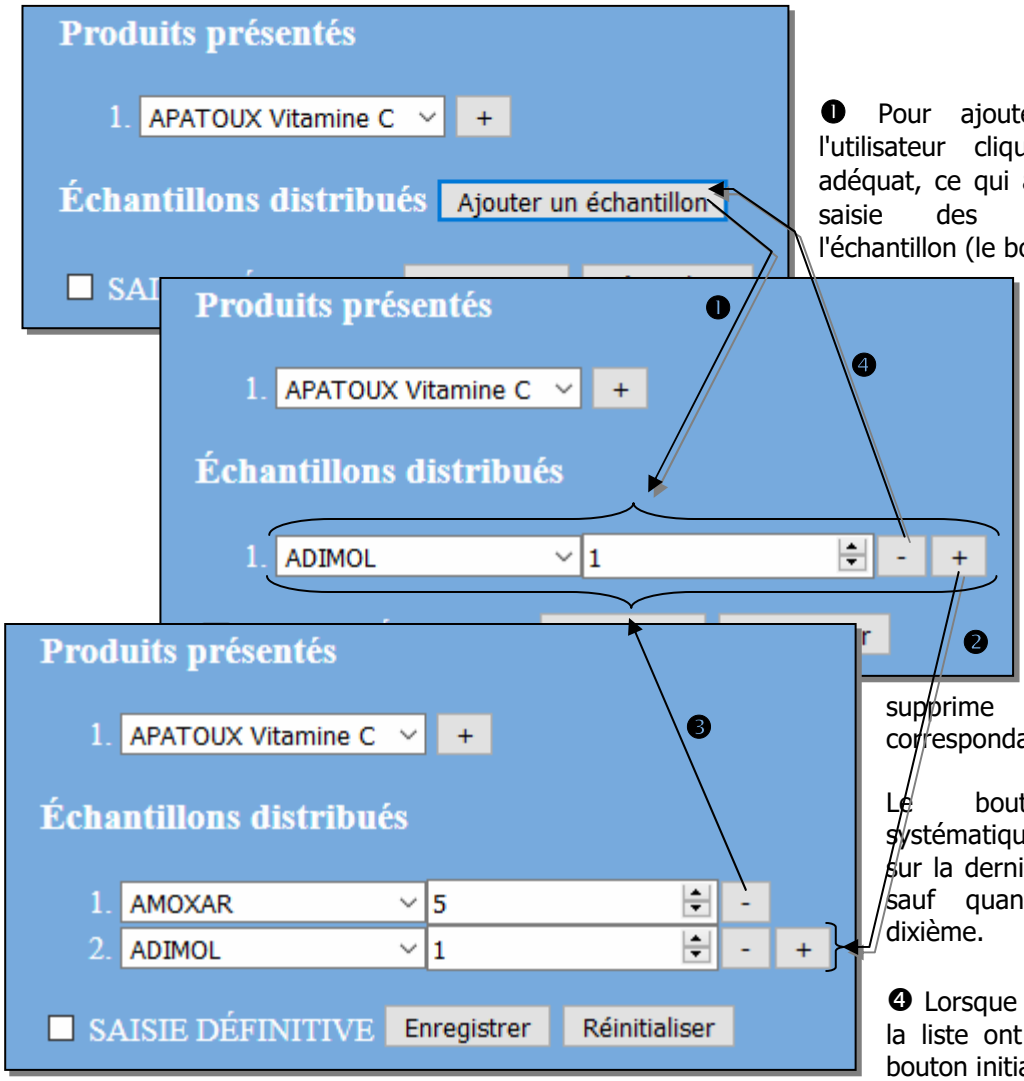
Il reste à implémenter la fonction `supprimerLigneProduitPresente()` qui doit :
- créer un bouton "+", dont l'`id` est `btnAjouterPP`, et le `tabindex` 100.
- Lui ajouter le gestionnaire de l'événement click qui appelle la fonction `ajouterLigneProduitPresente()` en lui passant la liste des produits présentés.
- Supprimer la dernière ligne des produits présentés (`remove()`).
- Ajouter le bouton "+" à la ligne restante.

Testez ensuite votre application.

2.4 Les échantillons distribués

Il faut respecter les cardinalités du MCD : aucun ou au maximum 10 échantillons peuvent être distribués lors d'une visite.

Le mode de fonctionnement suivant a donc été retenu :



❶ Pour ajouter un échantillon, l'utilisateur clique sur le bouton adéquat, ce qui affiche une ligne de saisie des informations sur l'échantillon (le bouton disparaît).

❷ Un click sur le bouton "+" ajoute une nouvelle ligne de saisie.

❸ Un clic sur le bouton "-" supprime la ligne correspondante.

Le bouton "+" doit systématiquement apparaître sur la dernière ligne de la liste, sauf quand celle-ci est la dixième.

❹ Lorsque toutes les lignes de la liste ont été supprimées, le bouton initial réapparait.

On a besoin de fonctions destinées à :
- créer un bouton (on dispose déjà de la fonction `jqCreerBouton()`).
- Créer un contrôle de saisie de type `number`.
- Ajouter un bouton "+" à une ligne de saisie d'un échantillon.
- Créer une ligne de saisie d'un échantillon (avec ou sans bouton "+").
- Créer la liste des échantillons en y incorporant une première ligne.
- Ajouter une ligne de saisie à la liste des échantillons.
- Supprimer une ligne de la liste.

Vous allez donc les développer …

2.4.1 La fonction de création d'un contrôle de saisie de type number

Pour chaque échantillon, la quantité distribuée doit être choisie via un contrôle de type `number`. La valeur par défaut est 1.

La fonction s'appelle `jqFormInputNumber()`. A part le label, elle doit prendre les mêmes paramètres et avoir le même comportement que la fonction PHP `formInputNumber()` que vous avez déjà réalisée. Elle doit donc créer un contrôle de saisie de type `number` sans label, car dans le DOM il s'agit de 2 nœuds différents. Elle retournera un objet de la classe JQuery.

Pour sa réalisation, vous pouvez vous baser sur la fonction `jqFormButton()` fournie, et sur la fonction PHP que vous avez écrite.

2.4.2 La fonction d'ajout d'un bouton "+" à une ligne de saisie d'un échantillon

Elle s'appelle `ajouterBoutonPlusLigne()` et doit prendre en paramètre :
- une ligne (un objet JQuery) ;
- le `tabindex` du bouton.

Elle doit créer un bouton "+" dont l'id est `btnAjouterED` et doit se contenter de l'ajouter aux nœuds enfants de la ligne passée en paramètre. Elle ne retournera rien.

2.4.3 La fonction de création d'une ligne de saisie d'un échantillon

Elle s'appelle `creerLigneEchantillonDistribue()` et doit prendre en paramètre :

- la liste de produits. Il s'agit d'une copie de la liste déroulante utilisée pour sélectionner un produit présenté.
- Le nom (`name`) de la liste des produits.
- Son `tabindex`.
- Le nom (`name`) du contrôle de saisie de la quantité.
- Un booléen indiquant si la ligne comporte le bouton "+".

La ligne est un nœud (un objet JQuery créé avec le code `$('<li></li>')`), parent de 3 nœuds enfants :
- une liste déroulante des produits (celle qui est reçue en pramètre),
- un contrôle de saisie du nombre de produit (à créer),
- un bouton "-", (à créer, et auquel on ajoutera bientôt son gestionnaire pour l'événement click).

Le booléen passé à la fonction détermine s'il faut ajouter le 4ème nœud enfant (le bouton "+" qui lui aussi recevra bientôt un gestionnaire pour l'événement click).

Le tabindex des contrôles de saisie suivant la liste des produits peut être incrémenté de 1.

La fonction doit retourner la ligne créée (un objet JQuery).

2.4.4 La création de la liste des échantillons distribués avec sa première ligne de saisie

Ajoutez le gestionnaire de l'événement `click` du bouton "Ajouter un échantillon" (methode `click()` de jQuery) dans votre fichier `crVisite.js`. Il appellera une fonction nommée `creerListeEchantillonsDistribuesAvecPremiereLigne()` qui ne recevra aucun paramètre.

L'étape suivante consiste à créer cette fonction dans laquelle vous commencerez par créer une copie de la liste déroulante des produits présentés.

Créez ensuite (avec JQuery) la liste numérotée (vide) dont l'`id` sera `listeEchantillonsDistribues`. Cet objet sera inséré dans le DOM après le titre "Échantillons distribués" (`insertAfter()`).

Supprimez du DOM le bouton "Ajouter un échantillon".

En utilisant les fonctions que vous venez de créer dans les étapes précédentes, ajoutez une nouvelle ligne de saisie à la liste.

Attention ! C'est le moment de bien réfléchir au nom de chaque contrôle de saisie de la ligne : pour être efficace, votre futur programme PHP de traitement du formulaire devra trouver sous forme de tableaux les valeurs qui auront été saisies dans les groupes de contrôles de saisie que vous aurez créés. Basez vous sur les noms que vous avez utilisés pour la liste des produits présentés. Faites validez les noms que vous aurez choisi par votre chef de projet.

Ensuite ne perdez pas de vue que l'utilisateur de l'application doit pouvoir travailler sans souris. Aussi, prévoyez que pour cette première ligne le `tabindex` de la liste des produits sera celui du dernier contrôle de saisie de la liste des produits présentés, incrémenté de 10 (voir `children()`, `eq()` et d'autres méthodes dans la partie *Traversing* de la documentation de JQuery).

Pour finir, toujours dans un soucis d'ergonomie, placez le focus sur la liste déroulante des produits.

Testez votre application.

2.4.5 L'ajout d'une ligne de saisie à la liste des échantillons distribués

Dans la fonction `ajouterBoutonPlusLigne()`, ajoutez au bouton le gestionnaire de l'événement click qui appelle la fonction nommée `ajouterLigneEchantillonDistribue()` en lui passant la liste des échantillons distribués.

Créez ensuite la fonction `ajouterLigneEchantillonDistribue()`. Elle sera appelée pour ajouter une ligne à la liste, à partir de la deuxième. Elle fera appel à la fonction `creerLigneEchantillonDistribue()`. Cette fonction a besoin de 5 paramètres :

- Une copie de la liste déroulante de la première ligne des produits produits présentés.
- Le `name` de la liste des produits (on a besoin pour cela de déterminer le nombre de lignes actuel de la liste d'échantillons distribuées).
- Le `tabindex` de la liste déroulante des produits dans la nouvelle ligne : il sera égal à celui de la liste de produits de la ligne précédente incrémenté de 10.
- Le `name` du contrôle de saisie de la quantité.
- Le booléen qui indique si le bouton "+" sera ajouté à la nouvelle ligne.

Vous pouvez supprimer le bouton "+" de la dernière ligne actuelle de la liste, juste avant d'appeler la fonction de création de la nouvelle ligne et d'ajouter cette nouvelle ligne à la liste des échantillons distribués.

Le focus devra finalement être placé sur le premier contrôle de saisie de la dernière ligne de la liste.

Testez votre application, et vérifiez bien que la liste peut comporter aucune ou 10 lignes au maximum.

2.4.6 La suppression d'une ligne de la liste des échantillons distribués

La fonction nommée `supprimerLigneEchantillonDistribue()` que vous allez développer attend en paramètre l'indice de la ligne à supprimer. Les indices des lignes de la ligne sont numérotés à partir de 0 (voir `index()`)

Une fois que la ligne a été supprimée, il faut vérifier si la dernière ligne de la liste comporte bien le bouton "+" (c'est le dernier enfant de la dernière ligne de la liste et il porte l'id `btnAjouterED`). En effet c'est peut-être cette ligne qui vient d'être supprimée. Si la vérification est négative, il faut ajouter le bouton "+" à la dernière ligne en appelant la fonction créée à cet effet (avec les bons paramètres).

La création d'une ligne se base sur le nombre de lignes de la liste pour déterminer l'indice des noms des contrôle de saisie de la nouvelle ligne. Après la suppression d'une ligne il y a des risques d'incohérence dans les indices. Exemple :

1. la liste contient 5 lignes. Les `name` ont un indice de 0 à 4.
2. La première ligne est supprimée. Les `name` ont maintenant un indice de 1 à 4.
3. Une ligne est ajoutée. Les `name` ont maintenant les indices 1, 2, 3, 4, 4.

On doit donc vérifier si la liste comporte encore des lignes. Si c'est le cas, afin d'éviter les incohérences dans les indices utilisés dans les noms des contrôles saisie de la liste il faut renuméroter ces indices. la méthode `each()` permet d'exécuter une fonction sur plusieurs éléments du DOM sans utiliser les instructions classiques d'itération telles que `for()`.

Utilisez-la pour renuméroter les indices des noms. Vous aurez besoin des méthodes JavaScript `indexOf()`, `substr()` et/ou `substring()` pour déterminer la position de l'indice dans le nom et reconstituer un nom avec un nouvel indice.

Il faut ensuite placer le focus sur la liste déroulante des produits de la dernière ligne.

Si en revanche la liste n'a plus aucune ligne après la suppression, il faut la supprimer du DOM, recréer le bouton "Ajouter un échantillon" et lui donner le focus.

A ce stade, bon tests !

2ème demande : enregistrer les données saisies

L'enregistrement ne doit pas s'effectuer directement : après la vérification des données saisies côté client, un message devra demander à l'utilisateur de confirmer l'enregistrement des données (cf. documentation sur la fonction JavaScript `confirm()`).

Côté serveur, les vérifications devraient être refaites au cas où le JavaScript aurait été désactivé côté client. Mais nous confierons la programmation de cette fonctionnalité ultérieurement à un autre membre de votre service …

Afin de respecter le cahier des charges (voir le point de vue du délégué régional), le mode de fonctionnement suivant a été retenu : un compte-rendu peut-être modifié tant que la case "Saisie définitive" n'a pas été cochée. Lorsqu'elle sera cochée, le compte-rendu sera considéré comme définitif. Ses date et heure de saisie seront alors renseignées dans la base de données et le visiteur ne pourra plus le modifier.

Implémentez l'enregistrement des données dans votre application.

Une fois les données enregistrées, le compte-rendu devra être réaffiché. S'il est définitif, les contrôles de saisie ne devront pas être actifs.

Dans tous les cas, un message devra indiquer si l'enregistrement a bien été effectué.

Testez votre application.

1 Mise à jour 2017 suite à la sortie de Firefox 57

Le contrôle de saisie `input` de type `date` est maintenant opérationnel dans Firefox 57 (c'était déjà le cas avec Chrome, Edge, Opera et Safari).

Ajoutez à votre bibliothèque une fonction nommée `formInputDate()` qui génèrera le code HTML d'un contrôle de saisie de type `date`. A vous de voir quel(s) paramètre(s) elle acceptera. Adaptez ensuite votre formulaire afin que la date de saisie du compte-rendu puisse être choisie via ce type de contrôle de saisie.