

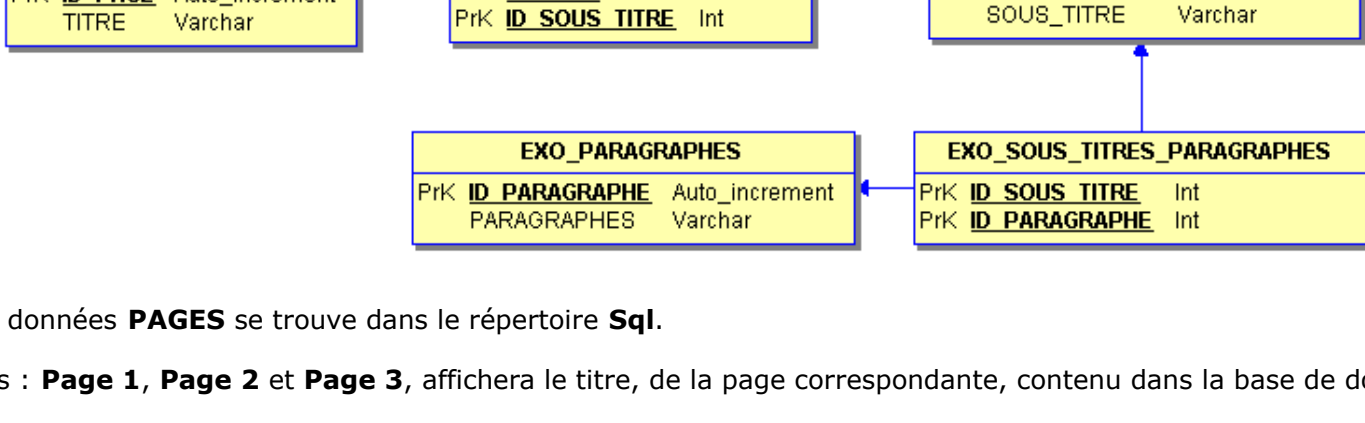
## Exercices

### Exercice 1 ★

Le but des exercices de 1 à 6 est de reprendre les exercices de d'AJAX-JSON de 1 à 7 mais avec une base de données au lieu d'un fichier.

Voici, ci-dessous le **MLD** (Modèle Logique de Données) de la base de données **PAGES**.

L'attribut **ORDRE** de la table **EXO\_SOUS\_TITRES\_PARAGRAPHERES** correspond à l'ordre d'affichage des paragraphes sous un sous-titre.





Le dépôt de la base de données **PAGES** se trouve dans le répertoire **Sql**.

Le clic sur un des items : **Page 1**, **Page 2** et **Page 3**, affichera le titre, de la page correspondante, contenu dans la base de données.

Pour ce développement, ajouter dans un modèle **MPages** avec une méthode **SelectTitre()** qui récupérera le titre de la page correspondant à l'item cliqué.

Dans le contrôleur, modifier la fonction **page()** afin qu'elle utilise la méthode **SelectTitre()** de la classe **MPages**.

 Visualisation

 Téléchargement (.zip)

### Exercice 2 ★★

Le clic sur un des items : **Page 1**, **Page 2** et **Page 3**, affichera le titre et les sous-titres, de la page correspondante, contenus dans la base de données.

Pour ce développement, ajouter, dans le modèle **MPages**, une méthode **SelectSousTitresAll()** qui récupérera les sous-titres de la page correspondant à l'item cliqué.

Dans le contrôleur, modifier la fonction **page()** afin qu'elle utilise la méthode **SelectSousTitresAll()** de la classe **MPages**.

 Visualisation

 Téléchargement (.zip)

### Exercice 3 ★★

Le clic sur un des items : **Page 1**, **Page 2** et **Page 3**, affichera le titre, les sous-titres et les paragraphes, de la page correspondante, contenus dans la base de données.

Pour ce développement, ajouter, dans le modèle **MPages**, une méthode **SelectParagraphesAll()** qui récupérera les paragraphes des sous-titres de la page correspondant à l'item cliqué.

Dans le contrôleur, modifier la fonction **page()** afin qu'elle utilise la méthode **SelectParagraphesAll()** de la classe **MPages**.

 Visualisation

 Téléchargement (.zip)

### Exercice 4 ★★


Comme dans les exercices de d'AJAX-JSON, le clic sur le titre remplacera le contenu de l'élément **<h1>** par un élément **<input>** avec comme contenu le texte du titre cliqué.


Après modification de ce texte nous supprimons l'élément **<input>** et le remplaçons par le texte modifié, tout en mettant à jour la table **EXO\_PAGES** de la base de données.

Pour ce développement, ajouter dans le modèle **MPages** une méthode **UpdateTitre()**.

La méthode **UpdateTitre()** modifiera le titre de la table **EXO\_PAGES** avec le texte récupéré de l'élément **<input>**.

Dans le contrôleur, modifier la fonction **change()** afin qu'elle utilise la méthode **UpdateTitre()** de la classe **MPages**.

 Visualisation

 Téléchargement (.zip)

### Exercice 5 ★★


Comme dans les exercices de d'AJAX-JSON, le clic sur un sous-titre remplacera le contenu de l'élément **<h2>** par un élément **<input>** avec comme contenu le texte du sous-titre cliqué.


Après modification de ce texte nous supprimons l'élément **<input>** et le remplaçons par le texte modifié, tout en mettant à jour la table **EXO\_SOUS\_TITRES** de la base de données.

Pour ce développement, ajouter dans le modèle **MPages** une méthode **UpdateSousTitre()**.

La méthode **UpdateSousTitre()** modifiera le titre de la table **EXO\_SOUS\_TITRES** avec le texte récupéré de l'élément **<input>**.

Dans le contrôleur, modifier la fonction **change()** afin qu'elle utilise la méthode **UpdateSousTitre()** de la classe **MPages**.

 Visualisation

 Téléchargement (.zip)

### Exercice 6 ★★


Comme dans les exercices de d'AJAX-JSON, le clic sur un sous-paragraphe remplacera le contenu de l'élément **<p>** par un élément **<textarea>** avec comme contenu le texte du paragraphe cliqué.


Après modification de ce texte nous supprimons l'élément **<textarea>** et le remplaçons par le texte modifié, tout en mettant à jour la table **EXO\_PARAGRAPHERES** de la base de données.

Pour ce développement, ajouter dans le modèle **MPages** une méthode **UpdateParagraphe()**.

La méthode **UpdateParagraphe()** modifiera le titre de la table **EXO\_PARAGRAPHERES** avec le texte récupéré de l'élément **<textarea>**.

Dans le contrôleur, modifier la fonction **change()** afin qu'elle utilise la méthode **UpdateParagraphe()** de la classe **MPages**.

 Visualisation

 Téléchargement (.zip)

### Exercice 7 ★★★

Dans cet exercice, nous allons ajouter un paragraphe dans la table **EXO\_PARAGRAPHERES** et l'afficher à la fin de la page.

Pour ce développement, ajouter à la fin de la page un élément paragraphe **<p>** sans identifiant et contenant un blanc (**&nbsp;**).


Ajouter dans le modèle **MPages** une méthode **InsertParagraphe()**.


La méthode **InsertParagraphe()** insérera le nouveau paragraphe dans la table **EXO\_PARAGRAPHERES** avec le texte récupéré de l'élément **<textarea>**.

Penser à insérer l'identifiant du paragraphe dans la table **EXO\_SOUS\_TITRES\_PARAGRAPHERES** avec l'identifiant du sous\_titre et l'ordre du paragraphe dans ce sous-titre.

Dans le contrôleur, ajouter la fonction **insert()** afin qu'elle utilise la méthode **InsertParagraphe()** de la classe **MPages**.

Dans le fichier **pages.js**, penser à ajouter l'identifiant sur le paragraphe inséré et à ajouter un nouveau paragraphe vide, afin de pouvoir à nouveau ajouter un paragraphe.

 Visualisation

 Téléchargement (.zip)

### Exercice 8 ★★★

Dans cet exercice, nous allons supprimer un paragraphe dans la table **EXO\_PARAGRAPHERES** et supprimer l'élément paragraphe **<p>** correspondant dans la page.

La suppression d'un paragraphe s'effectuera quand nous supprimerons le texte à l'intérieur d'un input.

Ajouter dans le modèle **MPages** une méthode **DeleteParagraphe()**.


La méthode **DeleteParagraphe()** supprimera un paragraphe dans la table **EXO\_PARAGRAPHERES**.


Penser à supprimer le tuple contenant l'identifiant du paragraphe dans la table **EXO\_SOUS\_TITRES\_PARAGRAPHERES**.

Dans le contrôleur, ajouter la fonction **delete()** afin qu'elle utilise la méthode **DeleteParagraphe()** de la classe **MPages**.

Dans le fichier **pages.js**, penser à tester si le texte est vide et dans ce cas déclencher le contrôleur avec le paramètre de contrôle **delete** et comme paramètre l'identifiant du paramètre, puis supprimer l'élément paragraphe **<p>**.

Penser à garder un paragraphe vide permettant ainsi d'insérer un nouveau paragraphe.

 Visualisation

 Téléchargement (.zip)

### Exercice 9 ★★★

Dans les cinq exercices suivants, nous allons développer la réorganisation (modification de l'ordre) des paragraphes dans une page par le déplacement d'un paragraphe en utilisant la technique du glisser-déposer.

Dans ce premier exercice, nous développerons le déplacement d'un paragraphe à l'intérieur de son sous-titre et seulement du haut vers le bas (voir la vidéo).

Dans le fichier Javascript **pages.js**, écrire les écouteurs **dragstart** et **dragover** avec les fonctions associées.

La fonction **dragstart()** associée à l'écouteur **dragstart** mettra un pourtour en pointillé autour de l'élément devant être déplacé et activera un écouteur **dragenter**.

La fonction **dragenter()** associée à l'écouteur **dragenter** permettra de visualiser l'élément actif, c'est à dire l'élément dans lequel l'élément déplacé se trouve. Celle-ci modifiera la couleur de fond de l'élément actif et mettra autour de ce dernier un pourtour en pointillé.

De plus, la fonction **dragenter()** activera pour l'élément actif deux écouteurs **dragleave** et **drop** et activera pour le paragraphe suivant un écouteur **dragenter**.

La fonction **dragover()** associée à l'écouteur **dragover** stoppera seulement l'événement.

La fonction **dragleave()** associée à l'écouteur **dragleave** remettra la couleur de fond par défaut et supprimera le pourtour pointillé de l'élément rendu inactif et supprimera sur cet élément les écouteurs **drop**, **dragenter** et **dragleave**.

La fonction **drop()** associée à l'écouteur **drop** :


- supprimera les différents écouteurs copiera l'élément déplacé dans une variable, récupérera l'élément suivant et insérera la copie de l'élément déplacé avant l'élément suivant, puis supprimera l'élément déplacé ;
- récupérera les identifiants des éléments **<p>** du paragraphe déplacé et du paragraphe de dépôt et en extraira les valeurs des clefs primaires (**ID\_PARAGRAPHE**) de ces paragraphes ;
- activera la fonction AJAX-JSON **actionParam()** permettant de transmettre ces valeurs afin de modifier dans la base de données l'ordre des paragraphes concernés.


A la suite de la fonction AJAX-JSON **actionParam()**, le paragraphe transféré devra avoir de nouveau les écouteurs **dragstart** et **dragover**.

Ajouter dans le contrôleur un paramètre de contrôle **drop** et la fonction associée **drop()**, cette dernière appellera la méthode **MoveParagraphe()** de la classe **MPages**.

La méthode **MoveParagraphe()** de la classe **MPages** modifiera l'ordre des paragraphes dû au déplacement d'un des paragraphes.

Penser que l'ordre des paragraphes compris entre le paragraphe déplacé et le paragraphe de dépôt doit être diminué de 1.

 Visualisation

 Téléchargement (.zip)


### Exercice 10 ★★★


Dans cet exercice, nous développerons le déplacement d'un paragraphe à l'intérieur de son sous-titre et seulement du bas vers le haut (voir la vidéo).

Le développement est quasiment identique à celui de l'exercice précédent.

Attention, la fonction **dragenter()** devra activer le paragraphe précédent avec un écouteur **dragenter** au lieu du paragraphe suivant.

Penser que l'ordre des paragraphes compris entre le paragraphe déplacé et le paragraphe de dépôt doit être augmenté de 1.

 Visualisation

 Téléchargement (.zip)

### Exercice 11 ★★★

Pour cet exercice, le déplacement d'un paragraphe se fera du haut vers le bas, comme dans l'exercice 8, mais cette fois-ci en pouvant aller dans un autre sous-titre (voir la vidéo).

Le développement est quasiment identique à celui de l'exercice 8.

Ajouter dans la méthode **showPage()** de la classe **VPages** et dans l'affichage du paragraphe un attribut **data-id\_sous\_titre** qui contiendra l'identifiant du sous-titre auquel appartient le paragraphe.

Dans la fonction **dragenter()**, empêcher que l'élément **<h2>** soit actif, et ajouter dans la variable **param** les paramètres donnant l'identifiant des sous\_titres auxquels appartiennent le paragraphe déplacé et le paragraphe de dépôt.

Modifier en conséquence la méthode **MoveParagraphe()** qui permettra de gérer le déplacement du paragraphe soit dans son sous-titre, soit dans un sous-titre différent.

 Visualisation

 Téléchargement (.zip)

### Exercice 12 ★★★


Pour cet exercice, le déplacement d'un paragraphe se fera du bas vers le haut, comme dans l'exercice 9, mais cette fois-ci en pouvant aller dans un autre sous-titre (voir la vidéo).


Le développement est quasiment identique à celui de l'exercice 9.

Comme dans l'exercice précédent, ajouter dans la méthode **showPage()** de la classe **VPages** et dans l'affichage du paragraphe un attribut **data-id\_sous\_titre** qui contiendra l'identifiant du sous-titre auquel appartient le paragraphe.

Dans la fonction **dragenter()**, empêcher que l'élément **<h2>** soit actif, ainsi que l'élément **<h1>**, et ajouter dans la variable **param** les paramètres donnant l'identifiant des sous\_titres auxquels appartiennent le paragraphe déplacé et le paragraphe de dépôt.

Modifier en conséquence la méthode **MoveParagraphe()** qui permettra de gérer le déplacement du paragraphe soit dans son sous-titre, soit dans un sous-titre différent.

 Visualisation


 Téléchargement (.zip)

### Exercice 13 ★★★

Dans cet exercice, le déplacement d'un paragraphe se fera dans les deux sens, tout en pouvant aller dans un autre sous-titre (voir la vidéo).

Fusionner les deux exercices précédents.

 Visualisation

 Téléchargement (.zip)

### Exercice 14 ★★★

Fusionner, l'exercice 8 et l'exercices 13.

Séparer le code Javascript en deux parties **pages.js** qui gèrera le contenu des pages et **drag.js** qui régira les glisser-déposer.

Dans la fonction Javascript **initPages()**, penser à rajouter les événements pour le glisser-déposer.

Dans la fonction Javascript **textTextarea()**, rendre le paragraphe **non draggable**.

Dans la fonction Javascript **changeText()** récupérer le sous-titre du dépôt afin de l'ajouter comme attribut **data-id\_sous\_titre**, et penser à rendre le paragraphe déplacé **draggable** et à rajouter les événements permettant le glisser-déposer.

Dans la fonction Javascript **drop()**, penser à ajouter l'événement click sur le paragraphe déplacé.

Dans la méthode **DeleteParagraphe()** de la classe **MPages**, penser à modifier l'ordre des paragraphes qui suivent le paragraphe supprimé.

 Visualisation

 Téléchargement (.zip)

