CRUD Modal Bootstrap dans une application web ASP.NET MVC 5

Objectifs du cours : gérer une liste d'objet via des fenêtres de dialogue (popup) Bootstrap.
Bootstrap est un framework HTML, CSS et JavaScript utile pour produire un design responsive, ce qui signifie que les pages HTML s'adapteront pour un affichage idéal sur mobile.

1. Une simple fenêtre modale Bootstrap

Pour construire une simple fenêtre popup, il faut un bouton ou lien d'appel :

```
<button type="button" data-toggle="modal" data-target="#myModal">Open Modal</button>
```

Pour un lien on écrira :

```
<a href="#" data-toggle="modal" data-target="#coucou">Open Modal</a>
```

Le bouton (ou lien) doit avoir 2 attributs spécifiques :

- le premier attribut `data-toggle="modal"` signifie que le bouton (ou lien) sert à ouvrir un popup
- le second `data-target="#myModal"` attribut précise l'id du popup à afficher

Il faut également un bloc div qui va contenir les informations affichées dans le popup.
Le popup Boostrap est un ensemble de blocs div imbriqués, chacun de ces blocs div doivent avoir un attribut class css spécifique au framework Bootstrap :

```
<div class="modal" id="myModal">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                Ceci est mon header
            </div>
            <div class="modal-body">
                Ceci est mon body
            </div>
            <div class="modal-footer">
                Ceci est mon footer
                <button data-dismiss="modal">Fermer</button>
            </div>
        </div>
    </div>
</div>
```

Le premier div est le conteneur du popup, c'est sur base de son id qu'il est ouvert pour le bouton (ou le lien).
Il doit obligatoirement avoir la class « modal » pour marquer le div comme popup invisible

```
<div class="modal" id="myModal"> . . . </div>
```

Le second div doit avoir la class « modal-dialog » pour définir la largeur et la marge

```
    <div class="modal-dialog"> . . . </div>
```

Le 3ème div est le contenu de la forme popup et doit avoir la class « modal-content »

```
        <div class="modal-content"> . . . </div>
```

Le div « modal-content » contiendra classiquement 3 parties :

- le header : div avec la class « modal-header »
- le body : div avec la class « modal-body »
- le footer : div avec la class « modal-footer »

```
        <div class="modal-content">
            <div class="modal-header"> . . . </div>
            <div class="modal-body"> . . . </div>
            <div class="modal-footer"> . . . </div>
        </div>
```

Dans le header on ajoutera classiquement un bouton « X » pour fermer le popup :

```
<div class="modal-header">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title">Modal Header</h4>
</div>
```

Le bouton doit avoir l'attribut « data-dismiss="modal" » pour causer la fermeture. La class « close » ajoute du style au bouton (&times ; est la croix en version html).

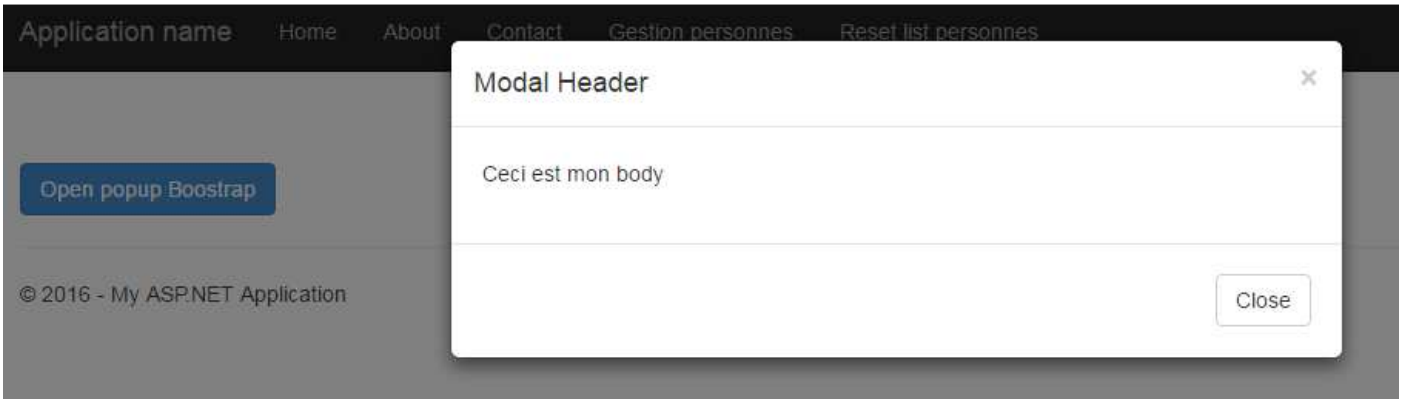
Dans le footer on ajoutera classiquement un bouton de fermeture (et de sauvegarde par exemple) :

```
<div class="modal-footer">
 <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
</div>
```

Ici aussi le bouton de fermeture doit avoir l'attribut « data-dismiss="modal" ».

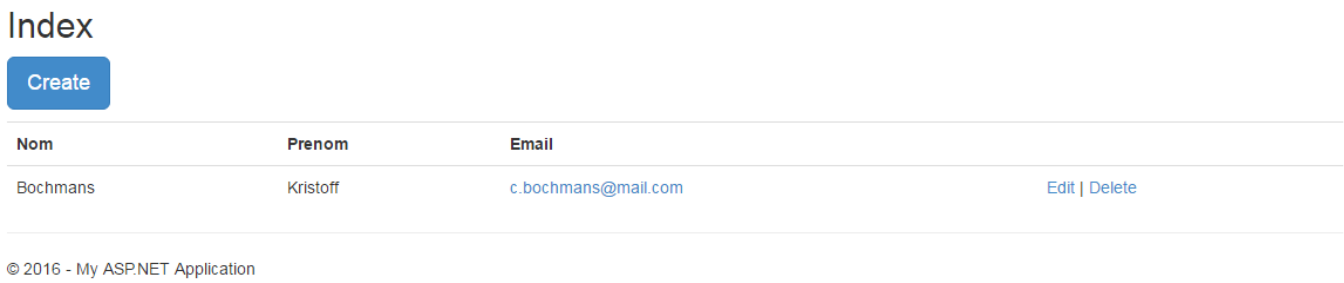
Exemple complet

```
<div>
    <a href="#" class="btn btn-primary" data-toggle="modal" data-target="#myPopupBootstrap">
        Open popup Boostrap
    </a>
</div>
<div class="modal" id="myPopupBootstrap">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Modal Header</h4>
            </div>
            <div class="modal-body">
                Ceci est mon body
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>
```



Pour aller plus loin : http://v4-alpha.getbootstrap.com/components/modal/

2. Page liste de personnes



Pour créer cette page, on va :

- créer model Personne (Nom, Prenom, Email)
- créer le controller Personne vide
- simuler une DB avec une liste de personne static
- ajouter l'action index qui renvoie la liste de personne
- créer la vue Index sur base d'une liste de Personne

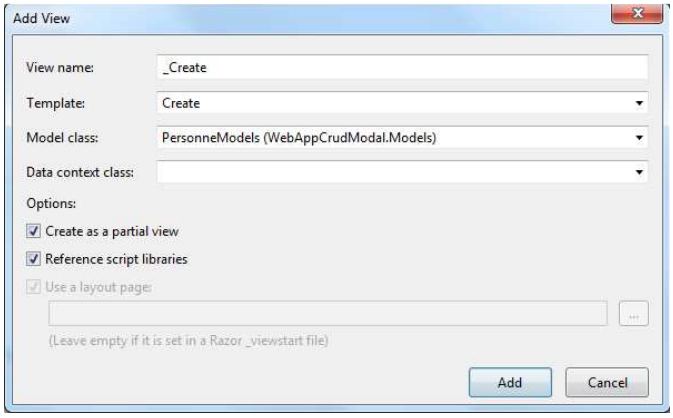
3. Création d'une personne avec une fenêtre modale

Dans le controller implémenter l'action Create() qui renvoie une vue partielle :

```
[HttpGet]
public ActionResult Create()
{
    return PartialView("_Create");
}
```

Générer une partial view « _Create » :

- clic droit sur l'action Create
- add view
- choisir le template Create
- sélectionner la class du Model
- cocher l'option « Create as a partial view » :



Notez que le nom de la vue est « _Create ». Le caractère « _ » n'est pas obligatoire, c'est classiquement une convention de nommage pour distinguer une vue normale d'une vue partielle.

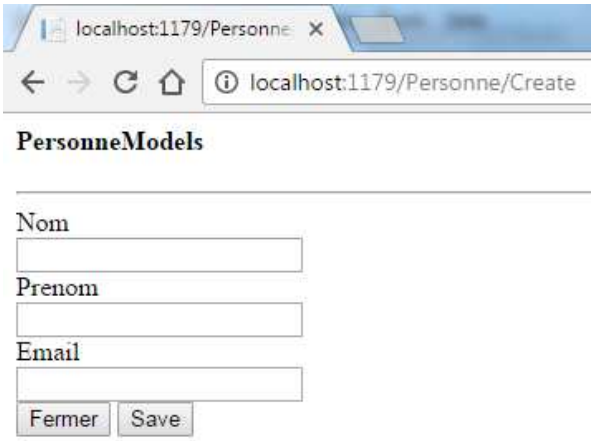
Notez également que la seule différence entre une vue normale et une vue partielle est le titre de la page :

```
@{
    ViewBag.Title = "Create normal";
}
<h2>Create normal</h2>
```

Important

Le rendu de la vue partielle est défini par le return de l'action du controller : `return PartialView("_Create");`

De cette manière, la vue est affichée sans master page :



Dans la page Index ajoutons le div modal et le bouton d'ouverture du popup :

```
<p><a href="#" class="btn btn-primary btn-lg" id="btnCreate">Create</a></p>

<div class="modal" id="myModal">
    <div class="modal-dialog">
        <div class="modal-content" id="myModalcontent">
            <div class="modal-header">
                Mon header
            </div>
            <div class="modal-body">
                Ceci est mon body
            </div>
            <div class="modal-footer">
                Mon footer
                <button data-dismiss="modal">Fermer</button>
            </div>
        </div>
    </div>
</div>
```

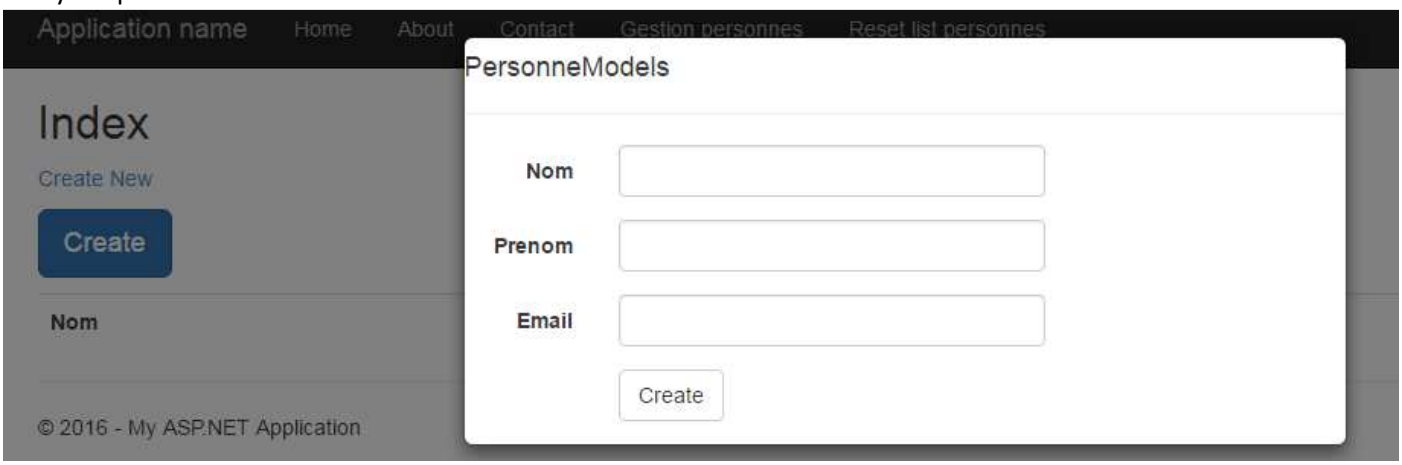
Sur l'évènement click du bouton, lancer une commande ajax qui va appeler l'action Create en GET et charger le résultat dans la forme modale :

```
@section Scripts {
<script>
    $(document).ready(function () {
        $("#btncreate").click(function () {
            $.ajax({
                type: "GET",
                url: '@Url.Action("Create","Personne")',
                success: function (data) {
                    $("#myModalContent").html(data);
                    $('#myModal').modal('show');
                }
            });
        });
    });
</script>
}
```

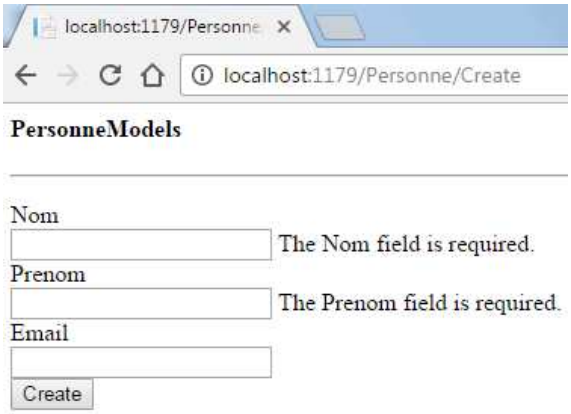
L'appel ajax renvoie des données « data » qui contient le code html de la vue partielle. Ce code html est chargé dans la forme modale via l'instruction `$("#myModalContent").html(data);`.

Il reste ensuite à afficher la forme modale `$('#myModal').modal('show');`.

Essayons pour vérifier :



Si on a rien modifié dans la view _Create, en cliquant sur Create, on perd l'aspect modal :



La solution est de poster via jQuery le formulaire de la view _Create.
On va travailler sur le click du bouton, poster en ajax le formulaire, sur le retour il faudrait :

- si le formulaire est valide recharger la page Index
- si le formulaire n'est pas valide, recharger le div modal avec le retour du serveur pour afficher les erreurs

Modifions la vue _Create pour y ajouter les boutons spécifiques à une forme modale :

```
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <button data-dismiss="modal" class="btn btn-default">Fermer</button>
                <input type="button" value="Save" class="btn btn-default" id="btnCreateModal" />
            </div>
        </div>
```

Le bouton « Fermer » ferme la fenêtre modale grâce à l'attribut `data-dismiss="modal"`.
Le bouton « Save » ne poste pas le formulaire (ce n'est pas un input type="submit").
La gestion du click sur le bouton « Save » doit être écrite dans la vue Index car la vue _Create ne connaît pas la forme modale.
Modifions la vue _Create pour ajouter un id au formulaire afin que nous puissions le poster depuis la vue Index :

```
@using (Html.BeginForm("Create", "Personne", FormMethod.Post, new { id = "frmCreateModal" }))
{
    . . .
}
```

Ou plus simplement encore, remplaçons le « using . . . BeginForm » par une simple balise form :

```
<form id="frmCreateModal">
    . . .
</form>
```

Dans la vue Index, on traitera le click du bouton « Save » de la vue _Create comme ceci :

```
$(document).on('click', "#btnCreateModal", function () {
    $.ajax({
        type: "POST",
        url: '@Url.Action("Create","Personne")',
        data: $("#frmCreateModal").serialize(),
        success:
            function (data) {
                if (data.result == "ok") {
                    $("#myModal").modal("hide");
                    location.reload();
                }
                else {
                    $("#myModalcontent").html(data);
                    $("#myModal").modal("show");
                }
            }
    })
});
```

Notez que nous n'avons pas écrit `$("#btnCreate").click(function () { . . .`
mais bien `$(document).on('click', "#btnCreateModal", function () { . . .`
La raison est la suivante : au chargement de la page Index, le bouton « Save » de la vue _Create n'existe pas puisque nous n'avons pas encore chargé cette vue partielle dans la forme modale.
L'appel « `on('click',` » permet d'écrire le code d'un évènement sur un élément qui a été chargé dynamiquement.
Remarque : merci à HAJAJ Yassin (si je ne me trompe pas) pour cette solution trouvée pendant le cours.

L'appel ajax poste le formulaire `frmCreateModal` vers l'action Create du controller Personne.
On y mentionne :

- qu'il s'agit d'une requête post : `type: "POST"`
- l'url : `'@Url.Action("Create","Personne")'`
- les données contenues dans le formulaire : `data: $("#frmCreateModal").serialize()`.

En cas de succès de l'appel ajax, on teste le retour :

- si « ok », on ferme le popup modal et on rafraîchit la page Index
- sinon, on charge le contenu du popup avec le retour de l'appel (la vue partielle _Create) et on affiche le popup modal

```
success:
    function (data) {
        if (data.result == "ok") {
            $("#myModal").modal("hide");
            location.reload();
        }
        else {
            $("#myModalcontent").html(data);
            $("#myModal").modal("show");
        }
    }
```

Coté serveur, il faut implémenter l'action post Create :

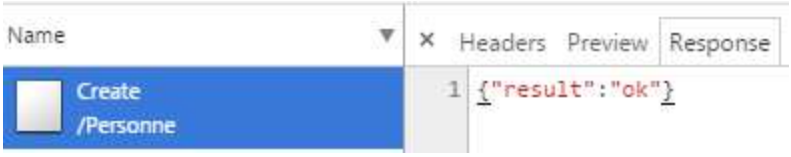
```
[HttpPost]
public ActionResult Create(PersonneModels model)
{
    if (ModelState.IsValid)
    {
        if (_listPers.Count == 0)
        {
            model.Id = 101;
        }
        else
        {
            model.Id = 1 + _listPers.Max(x => x.Id);
        }
        _listPers.Add(model);
        return Json(new { result = "ok" });
    }
    return PartialView("_Create", model);
}
```

Logiquement, cette action reçoit en paramètre le modèle.

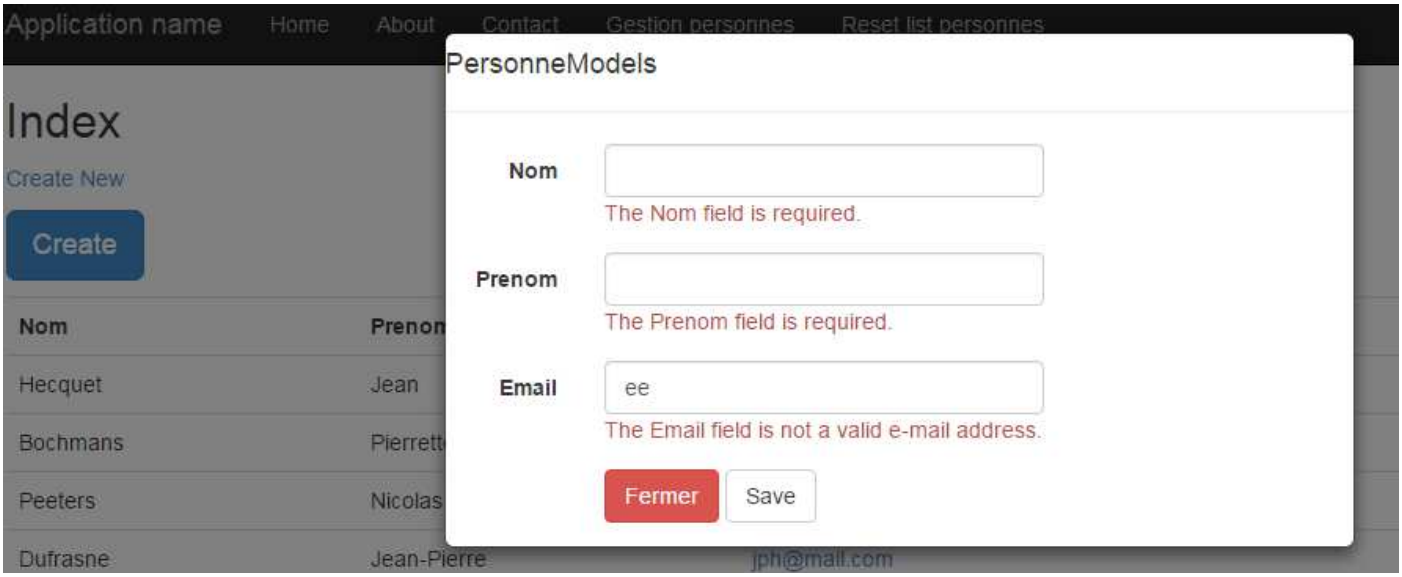
Si le modèle est valide :

- on ajoute la personne à la liste des personnes
- on renvoie un JSON avec une seule entrée result avec la valeur « ok » :

```
return Json(new { result = "ok" });
```



Si le modèle n'est pas valide, on renvoie la vue partielle _Create avec le modèle passé en paramètre de l'action pour afficher dans le popup les erreurs :



On obtient donc le résultat escompté, un popup modal qui nous permet d'encoder une nouvelle personne avec la gestion des erreurs d'encodage.

4. Update

Pour l'update, c'est similaire au create, il suffit de créer les actions get, post et la vue partielle associée (voir projet dans votre Campus Virtuel).

5. Delete

On créera un div modal spécifique dans la vue Index :

```
<div class="modal" id="modalDelete">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                Delete personne
            </div>
            <div class="modal-body" id="modalDeleteContentBody" ></div>
            <div class="modal-footer">
                <button class="btn btn-default"  data-dismiss="modal">Fermer</button>
                <button class="btn btn-danger" id="btnDeletePersonne">Supprimer</button>
            </div>
        </div>
    </div>
</div>
```

Et un input hidden pour stocker l'id de la personne à supprimer :

```
<div><input type="hidden" id="IdToDelete" /></div>
```

Dans la liste des personnes, on adaptera le lien « Delete » :

```
<a href="#" onclick="ShowModalDelete(@item.Id)">Delete</a>
```

Sur click du lien « Delete » on appelle la fonction ShowModalDelete avec l'id de la personne à supprimer en paramètre.

```
        function ShowModalDelete(id) {
            $.ajax({
                type: 'GET',
                dataType: 'json',
                url: '@Url.Action("GetPersonne","Personne")',
                data: { id: id },
                success:
                    function (response) {
                        if (response.result == "ok") {
                            $("#modalDeleteContentBody").html(
                                  "Supprimer la personne " +
                                  response.personne.Nom + " " +
                                  response.personne.Prenom + " ?");
                            $("#IdToDelete").val(id);
                            $("#modalDelete").modal("show");
                        }
                    }
            })
        }
```

Cette fonction va effectuer les opérations suivantes :

- un appel ajax pour récupérer un objet JSON avec l'objet Personne de la personne à supprimer

```
            $.ajax({
                type: 'GET',
                dataType: 'json',
                url: '@Url.Action("GetPersonne","Personne")',
                data: { id: id },
```

- il faut bien entendu ajouter l'action correspondante dans le controller

```
        [HttpGet]
        public JsonResult GetPersonne(int id)
        {
            PersonneModels pers = _listPers.Find(x => x.Id == id);
            return Json(new { result = "ok", personne = pers },JsonRequestBehavior.AllowGet);
        }
```

- modifier le contenu du popup delete pour y afficher le message approprié

```
$("#modalDeleteContentBody").html(
      "Supprimer la personne " +
      response.personne.Nom + " " +
      response.personne.Prenom + " ?");
```

- stocker l'id de la personne à supprimer dans l'input hidden : `$("#IdToDelete").val(id);`
- afficher le popup delete : `$("#modalDelete").modal("show");`

Il faut encore écrire la fonction qui est exécutée en cliquant sur le bouton « Supprimer » du popup delete :

```
$("#btnDeletePersonne").click(function () {
    $.ajax({
        type: "POST",
        url: '@Url.Action("Delete","Personne")',
        data: { id: $("#IdToDelete").val() },
        success:
            function (data) {
                if (data == "ok") {
                    $("#modalDelete").modal("hide");
                    location.reload();
                }
                else {
                    $("#modalDeleteContentBody").html('Error !!!!');
                    $("#modalDelete").modal("show");
                }
            }
    })
});
```

Cette fonction effectue un appel ajax en post vers l'action Delete du controller Personne avec l'id de la personne à supprimer comme paramètre (cet id est récupéré via le champ hidden `IdToDelete`).
Ici aussi, on écrira l'action correspondante dans le controller :

```
[HttpPost]
public JsonResult Delete(int id)
{
    _listPers.RemoveAll(x => x.Id == id);
    return Json("ok");
}
```

Notez qu'on pourrait gérer le delete modal comme le create ou l'update. Il faudrait créer une vue partielle _Delete avec affichage des infos de la personne, un input hidden avec l'id de la personne à supprimer.