Laboratoire: Views ViewModels et Partial Views

# Objectif(s)

* Convevoir des views complexes
  + Views
  + ViewModels
  + PartialViews
  + linq

# Créer la branche de la fonctionnalité

Clonage du repository

1. Dans Gitkraken, clonez le repository
   * Soit via URL
   * Soit via GitHub.com
2. Ouvrez le repository (OK)

Création d’une branche pour la fonctionnalité

1. Créez une branche appelée **FCT \_VuesPartielles**
2. Assurez-vous d'être dans la branche que vous avez créée dans LOCAL

# Initialisation

1. Faire un update-database afin d’avoir la dernière version à jour dans la base de données

## Lazy Loading

1. Activez le LazyLoading en installant le package nuget **(Installez la même version que les autres packages d’Entity Framework, ça devrait être 6.0.21**)



1. En le configurant dans **Program.cs**

builder.Services.AddDbContext<ZombiePartyDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")).  
UseLazyLoadingProxies());

1. En mettant **chaque** propriété de navigation virtual, dans le modèle de données, par exemple :

public virtual ZombieType ZombieType { get; set; }

1. Testez l’application afin de vous assurez qu’elle fonctionne comme avant avec le *LazyLoading*

## Commentaires et validation (Commit) des changements dans le code

1. Dans Gitkraken, assurez-vous d'être dans le bon dossier/repo et sur la bonne branche
2. Cliquez sur **View changes**
3. Validez les modification: cliquez sur **Stage all changes**
4. Commentez en respectant les standards proposés (vous serez noté là-dessus):
   * **Summary**: REFACTOR LazyLoading
   * **Description**: Configuration du LazyLoading
5. Cliquez sur **Commit**
6. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 

# Ajustements à la classe Zombie

## Modifications aux propriétés

1. Ajouter une propriété nommée Force à la classe Zombie qui est un entier.
2. Ajustez les éléments dans la classe ModelBuilderDataGenerator pour cette nouvelle propriété en remplaçant le contenu de la région Zombie par celui-ci.

builder.Entity<Zombie>().HasData(

new Zombie { Id = 1, Point = 5, Force =20, ShortDesc = "Pirate des Caraibes", Name = "LeChuck", ZombieTypeId = 1 },

new Zombie { Id = 2, Point = 4, Force = 10, ShortDesc = "Cute Little Dead Girl", Name = "Lenore", ZombieTypeId = 2 },

new Zombie { Id = 3, Point = 8, Force = 12, ShortDesc = "En costume de soirée, avec un chapeau haut de forme blanc et des lunettes soleil", Name = "Baron Samedi", ZombieTypeId = 2 },

new Zombie { Id = 4, Point = 2, Force = 5, ShortDesc = "Vivant dans sa tombe grandant son trésor", Name = "Draugr", ZombieTypeId = 3 },

new Zombie { Id = 5, Point = 5, Force = 20, ShortDesc = "Ancien vampire transformé en poupée de chiffon, ami de Lenore", Name = "Ragamuffin ", ZombieTypeId = 2 },

new Zombie { Id = 6, Point = 6, Force = 5, ShortDesc = "Tête de sac avec yeux en boutons, amoureux de Lenore", Name = "Mr Gosh", ZombieTypeId = 1 },

new Zombie { Id = 7, Point = 1, Force = 12, ShortDesc = "Tête de cerf empaillé, voisin de Lenore", Name = "Taxidermy ", ZombieTypeId = 3 },

new Zombie { Id = 8, Point = 3, Force = 10, ShortDesc = "Chat mort de Lenore", Name = "Kitty", ZombieTypeId = 1 },

new Zombie { Id = 9, Point = 2, Force = 20, ShortDesc = "Voleur très rapide", Name = "Singe zombie", ZombieTypeId = 3 },

new Zombie { Id = 10, Point = 7, Force = 23, ShortDesc = "chien très rapide, pouvant être enflammé", Name = "chien de l'enfer", ZombieTypeId = 3 },

new Zombie { Id = 11, Point = 9, Force = 15, ShortDesc = "Attaque avec des éclairs suite à un orage", Name = "Avogadro", ZombieTypeId = 3 },

new Zombie { Id = 12, Point = 6, Force = 10, ShortDesc = "", Name = "Lady Rose", ZombieTypeId = 3 },

new Zombie { Id = 13, Point = 2, Force = 4, ShortDesc = "Jeune étudiante qui cherche son professeur pour se venger", Name = "Matbeth", ZombieTypeId = 2 },

new Zombie { Id = 14, Point = 5, Force = 5, ShortDesc = "Le clown malheureux qui court après les enfants", Name = "The Clown", ZombieTypeId = 3 },

new Zombie { Id = 15, Point = 4, Force = 10, ShortDesc = "Le comptable mécontent qui cherche les failles", Name = "Clicker", ZombieTypeId = 2 },

new Zombie { Id = 16, Point = 10, Force = 12, ShortDesc = "L'étudiant happé par Teams. Trop d'heures de vidéos ont transformé cet étudiant en zombie", Name = "TeamsZombie", ZombieTypeId = 3 },

new Zombie { Id = 17, Point = 4, Force = 20, ShortDesc = "La fêtarde du samedi soir qui cherche les fêtes et les bars en vogue", Name = "Mathilde", ZombieTypeId = 3 }

);

1. Ajoutez une migration **AjoutZombieForce\_Seed**, mettez à jour votre base de données
2. Ajustez les vues qui utilisent **Zombie** ou **ZombieVM** comme modèle pour **afficher la propriété force**
3. Testez que les vues fonctionnent toujours tel que prévu et affichent la **force**

## Commentaires et validation (Commit) des changements dans le code

1. Dans Gitkraken, assurez-vous d'être dans le bon dossier/repo et sur la bonne branche
2. Cliquez sur **View changes**
3. Validez les modification: cliquez sur **Stage all changes**
4. Commentez en respectant les standards proposés (vous serez noté là-dessus):
   * **Summary**: FCT Ajout Zombie Force et seed
   * **Description**: Ajout de la propriété Force dans Zombie, modification du Seed et des vues Create, Update et Details pour ajouter le champ Force
5. Cliquez sur **Commit**
6. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 

# ZombieCard

Nous voulons créer une vue partielle présentant l’information principale d’un zombie sous forme de Card Bootstrap qui pourra être utilisée dans d’autres vues. Cette vue partielle présente une image générique de zombie qui sera, utltérieurement, remplacée par l’image spécifique du zombie. Cette image se trouve dans **Root/images/zombies**

## Créer la vue partielle ZombieCard

1. Créez une nouvelle vue partielle appelée **\_ZombieCard** à partir du code en annexe. Les vues partielles vont habituellement sous le dossier **/Shared/** et commencent par un caractère \_
2. Remplacez le contenu par le code ci-dessous

<div class="col-lg-4 col-md-6 pb-4">

  <div class="card border-primary rounded shadow-sm" style="background: url('/images/imagesLayout/fond.png');">

    <div class="card-body px-3 pt-3 pb-1 row">

      <p class="card-title text-center text-warning h5"><b>NAME</b></p>

    </div>

    <img class="card-img-top img-fluid d-block mx-auto mb-3" src="~/images/zombies/GenericZombie.png" alt="Zombie photo"  style="width:85%">

    <div class="card-body p-1 px-3 row">

      <div class="col-6" style="padding-left:25px;">

        <span class="badge p-2 border w-100 btn-dark">ZOMBIE\_TYPE NAME</span>

      </div>

      <div class="col-6" style="padding-left:3px;padding-right:25px;">

        <span class="badge p-2 border w-100 btn-primary">FORCE</span>

      </div>

      <div style="padding-left:25px;">

        <p class="text-white">Points: <span><b class="">POINT</b></span></p>

      </div>

    </div>

      <a class="p-2 border w-100 btn btn-warning px-3">

        Show Detail Card

      </a>

    <hr/>

  </div>

</div>

1. Ajoutez le modèle **Zombie** à la vue (assurez-vous qu’il est bien au début de la vue)
2. Modifiez le texte en majuscule pour qu’il utilise les informations du modèle.

## Utiliser la vue ZombieCard

Nous allons utiliser la vue où il y a l’affichage des informations sur les zombies :

* + dans Zombie/Index (Note : Vous allez perdre les boutons Edit et Delete, mais ne vous inquiétez pas pour ça, on va avoir une solution dans un autre laboratoire)
  + sous ZombieType/Details

<div class="row">

@\* POUR CHAQUE ZOMBIE:\*@

@foreach (var item in Model.ZombiesList)

{

<**partial** **name**="\_ZombieCard" **model**="item" />

}

</div>

### Zombie/Index

La vue Zombie/Index utilise IEnumerable **Zombie**,

1. Ouvrez la vue **Zombie/Index**
2. Remplacez le code dans la boucle par la vue partielle
3. Testez la vue **Zombie/Index**

### ZombieType/Details

La vue ZombieType/Details utilise **ZombieTypeVM**,

1. Ouvrez la vue **ZombieType/Details**
2. Remplacez le code dans la boucle par la vue partielle
3. Testez la vue **ZombieType/Details**

Pourquoi est-ce pertinent d’avoir des vues partielles?

## Commentaires et validation (Commit) des changements dans le code

1. Dans Gitkraken, assurez-vous d'être dans le bon dossier/repo et sur la bonne branche
2. Cliquez sur **View changes**
3. Validez les modification: cliquez sur **Stage all changes**
4. Commentez en respectant les standards proposés (vous serez noté là-dessus):
   * **Summary**: REFACTOR Ajout vue partielle \_ZombieCard
   * **Description**: Ajout de la vue partielle \_ZombieCard utilisée dans Zombie/Index et ZombieType/Details
5. Cliquez sur **Commit**
6. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 

# UPSERT Weapon

Un **Upsert,** de la contraction **Up**date et In**sert**, permet de faire l’ajout et la modification sur une seule interface. Cela diminue le nombre de formulaire à maintenir. Le titre doit s’accorder avec l’action désirée et l’action du formulaire doit être modifiée pour **Upsert**

Le formulaire doit proposer le bouton Create sans passer de Id ou Edit en passant le Id, selon le cas et Back to List (revenu à la vue Index). Ces boutons se répètent dans tous les formulaires Upsert\* : une vue partielle est tout indiquée.

## Création de la vue partielle \_upsertButtons

1. Créez la vue partielle **\_UpsertButtons** contenant le code ci-dessous :

@model int

<div class="col">

@if (Model == 0)

{

<input type="submit" class="btn btn-success w-100" value="Create" />

}

else

{

<input type="submit" class="btn btn-info w-100" value="Edit" />

}

</div>

<div class="col">

<**a** **asp-action**="Index" class="btn btn-secondary w-100">Back to list</**a**>

</div>

1. Notez le modèle est un int qui reçoit le **Id** du **Weapon**
2. Insérez la vue partielle dans la vue **Create** à la place du bouton Create actuel et enlever le « Back to List » du bas de la page qui est maintenant superflue. Dans le cas du create, le model est simplement 0.

## Mise à jour de la vue create : Upsert

1. Pour le **Weapon**, nous avons les actions et Vues Index et Create. Nous allons mettre à jour l’action Create afin qu’elle accepte également la mise à jour (update).
2. Modifiez le nom de la vue **Weapon/Create** pour **Upsert**
3. Ouvrez la vue **Weapon/Upsert**
4. Ajoutez la boîte d’entrée pour le Id masqué (hidden). Le formulaire pourra ainsi transmettre le Id (WeaponId) mais il n’a pas besoin d’être visible à l’utilisateur.

<**input** **asp-for**="WeaponId" class="form-control" hidden/>

## Mise à jour du contrôleur

Les actions **Upsert** doivent permettre d’ajouter ou de modifier des armes (*Weapons***)**

1. Ouvrez le controlleur **Weapon**
2. Le nom des actions GET et POST de **Create** doit être remplacé par **Upsert**

### GET Upsert

Dans le cas du create, aucun Id n’est passé en paramètre (le Id est null), par contre dans le cas du update un Id doit pouvoir être passé.

1. Modifiez le code du GET **Upsert** afin que lorsque l’id est null ou 0, le formulaire s’affiche vide permettant de créer un nouveau **Weapon.** S et lorsqu’il est égale à un autre chiffre, que le formulaire retourne les informations du **Weapon** concerné.

Si vous avez des difficultés, découvrez la réponse en mettant le texte ci-dessous en noir.

//GET

public IActionResult Upsert(int? id)

{

if (id == null || id == 0)

// create

return View(new Weapon());

else

//update

return View(\_baseDonnees.Weapons.Find(id));

}

1. Ajoutez un point d’arrêt sur la première ligne de la méthode
2. Testez le create (sans id) et le Update (avec un id) via les URLs **Weapon/Upsert et Weapon/Upsert/Id**

### POST Upsert

1. Modifiez le code du POST **Upsert** afin que lorsque l’id est égal à 0, un nouveau **Weapon** est ajouté à la BD**.** S et lorsqu’il est égale à un autre chiffre, que les informations du **Weapon** concerné soient mises à jour.

Si vous avez des difficultés, découvrez la réponse en mettant le texte ci-dessous en noir.

[HttpPost]

public IActionResult Upsert(Weapon weapon)

{

if (ModelState.IsValid)

{

// Create

if (weapon.WeaponId == 0)

{

// Ajouter à la BD

\_baseDonnees.Weapons.Add(weapon);

TempData["Success"] = $"{weapon.Name} weapon added";

}

else

{

// Update

\_baseDonnees.Weapons.Update(weapon);

TempData["success"] = $"{weapon.Name} weapon updated";

}

\_baseDonnees.SaveChanges();

return this.RedirectToAction("Index");

}

return this.View(weapon);

}

1. Ajoutez l’annotation [ValidateAntiForgeryToken].

## Modifier la vue Weapon Upsert

1. Modifiez la section du titre afin qu’il affiche Create ou Edit selon le cas
2. Si vous avez des difficultés, découvrez la réponse en mettant le texte ci-dessous en noir.
3. Modifiez également l’action du form pour exécuter un Upsert et assurez-vous de bien afficher la vue partielle \_UpsertButtons dans les 2 cas.

@model ZombieParty.Models.Weapon

@if (Model.WeaponId != 0)

{

ViewData["Title"] = "Edit";

}

else

{

ViewData["Title"] = "Create";

}

<h1> @ViewData["Title"] </h1>

<h4>Weapon</h4>

## Modifier Weapon Index

1. Ouvrez la vue **Weapon/Index**
2. Modifiez l’action du bouton « Create New »
3. Enlevez le commentaire sur le bouton **Edit** et complétez le code

@Html.ActionLink("Edit", "Upsert", new { id=item.WeaponId }) |

## Commentaires et validation (Commit) des changements dans le code

1. Dans Gitkraken, assurez-vous d'être dans le bon dossier/repo et sur la bonne branche
2. Cliquez sur **View changes**
3. Validez les modification: cliquez sur **Stage all changes**
4. Commentez en respectant les standards proposés (vous serez noté là-dessus):
   * **Summary**: FCT Usert Weapon Ajout vue partielle \_UpsertButtons
   * **Description**: Ajout de la vue partielle \_Upsert utilisée dans Weapon/Upsert et de la vue Weapon/upsert
5. Cliquez sur **Commit**
6. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 

# Compléter HuntingLog

## AJout de HuntingLog/Index

1. Créez le contrôleur **HuntingLogController**
2. Complétez l’action **Index**
3. Créez la vue **Index,** affichant le *Title* et les boutons **Create, Edit, Details et Delete** en vous inspirant des vues créées précédement
4. Testez la vue **Index** via l’URL
5. Ajoutez dans **\_Layout** un menu **Hunters** et une commande dans le menu appelée **Manage Hunting Log** qui mène à la vue **Index**
6. Testez

## AJout de HuntingLog/Upsert

1. Ajoutez l’action GET **Upsert** au contrôleur **HuntingLogController**
2. Créez la vue **Upsert,** contenant tous les champs du modèle en vous inspirant des vues créées précédement
3. Assurez-vous que Description soit un TextArea
4. Testez la vue GET **Upsert** via l’URL
5. Ajoutez l’action POST **Upsert** aucontrôleur **HuntingLogController**

NOTE : Nous ajouterons les zombies liés à l’entrée du journal de chasse (huntingLog) ultérieurement

## commentaires et validation (Commit) des changements dans le code

1. Dans Gitkraken, assurez-vous d'être dans le bon dossier/repo et sur la bonne branche
2. Cliquez sur **View changes**
3. Validez les modification: cliquez sur **Stage all changes**
4. Commentez en respectant les standards proposés (vous serez noté là-dessus):
   * **Summary**: FCT Ajout HuntingLog
   * **Description**: Ajout du Controlleur HuntingLog et des vues Upsert et Index
5. Cliquez sur **Commit**
6. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 
7. Une fois que tout est terminé, n’oubliez pas de faire un push.
8. Merge la branche **FCT\_VuesPartielles** dans la branche **Main** et faites un **dernier push.**

**Fin du laboratoire!**