Laboratoire: GitHub et révision MVC

# Objectif(s)

**GitHub/GitKraken**

* Repository
* Branches

**Révision 2W5**

* 1 seul projet MVC (1 layer)
* CRUD à partir de listes *ViewBag*
* FCT Create Zombie
* FCT Create Zombie Category
* Validation côté-client et côté-serveur
* Annotations models
* Erreur *span*

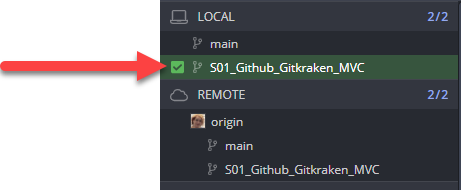
# Repository GitHub pour les laboratoires

## Création du Repository et de la branche

1. Dans GitHub, créez un nouveau repository privé **A21\_3W6\_Labos\_NOM\_PRENOM**
   * Private
   * Add a ReadMe file
   * Add .gitignore (visual Studio)
2. Ajoutez votre enseignante comme collaborateur: valerieturgeon
3. Créez la branche de la séance: **S01\_Github\_Gitkraken\_MVC**

## Clonage du repository et activation de la branche de la séance

1. Dans Gitkraken, clonez le repository
   * Soit via URL
   * Soit via GitHub.com
2. Ouvrez le repository (OK)
3. Assurez-vous d'être dans la branche de la séance, dans LOCAL



NOTE: si la branche de la séance n'apparaît que la dans section **REMOTE**: cliquez 2 fois sur celle-ci, puis sur sa copie dans LOCAL (voir image précédente)

# Solution/projet MVC ZombieParty

## Création du projet et de la solution ZombieParty

1. Dans Visual Studio/Visual Studio Code, créez un nouveau projet **Application Web Asp Net Core application MVC**
   * Nom: **ZombieParty**
   * Emplacement**: Dossier où vous avez cloné le repo.**
   * NE PAS mettre la solution et le projet dans le même répertoire

Étape 2:

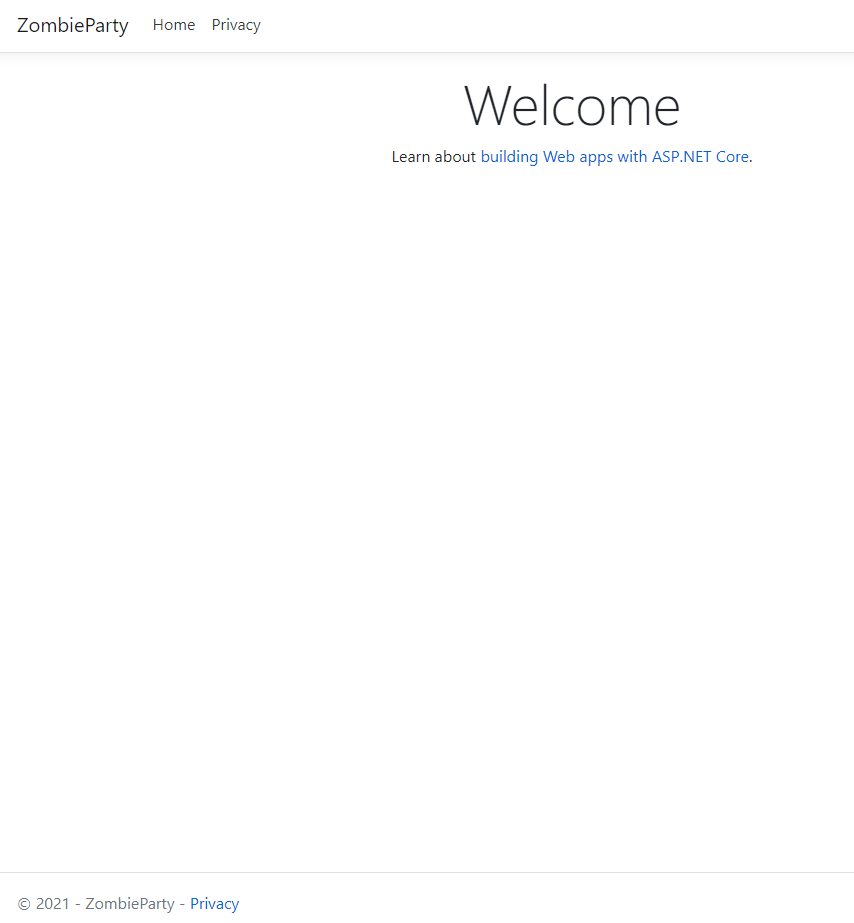
* NET 5.0
* Cochez Activer la compilation de runtime Razor

1. Attendez le chargement…soyez patient 😉
2. Vérifiez que vous êtes bien dans la branche de la séance (en bas à droite)



NOTE: Si vous voyez MAIN, vous n'êtes pas dans la bonne branche. Changez de branche et recréez le projet. (on corrigera le Main plus tard avec les Merges).

1. Exécutez le projet: la présentation de la page Index par défaut devrait apparaître



1. Arrêtez l'exécution

## 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 la création des 52 fichiers. Cliquez sur **Stage all changes**
4. Commentez en respectant les standards proposés (vous serez noté là-dessus):
   * **Summary**: INIT Solution Projet Web
   * **Description**: Création initiale du projet MVC (1 seul projet)
5. Cliquez sur **Commit**

NOTE: les changements sont enregistrés dans le repo. LOCAL SEULEMENT

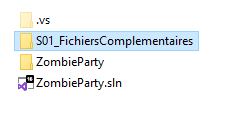
1. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 

NOTE: Vous pourriez faire plusieurs **commits,** et faire un **Push** avec tous les commits. Conseil: ne pas attendre trop longtemps avant de Push.

## Modification de la présentation du projetBootswatch et cie

### Les images de présentation

1. Téléchargez le dossier contenant les fichiers supplémentaires de la séance dans le Repo GitHub, dans la branche de départ de la séance [S01\_Github\_Gitkraken\_MVC](https://github.com/valerieturgeon/A21_3W6_Laboratoires/tree/S01_Github_Gitkraken_MVC)
2. Extrayez (dézipper) le contenu du dossier de la solution



Dans votre dossier de Solution/Projet **ZombieParty**:

1. Créez un dossier **images** dans **wwwroot**
2. Ouvrez le dossier *S01\_FichiersComplementaires/images*
3. Copiez le dossier (**imagesLayout**) et son contenu dans votre dossier **images**, il contient les images générales :
   * **cimetierePale.png**
   * **Tile.png**
   * **Degrade.png**
   * **Fond.png**
4. Remplacez le fichier **wwwroot/favicon.ico** par celui contenu dans *S01\_FichiersComplementaires/images*

### Modification du layout et et application d'un style bootswatch

**Style**

1. Dans votre dossier de Solution/Projet **ZombieParty**:
2. Remplacez le fichier fichier **wwwroot/css/site.css** par celui contenu dans *S01\_FichiersComplementaires/site.css*
3. Testez

**Bootstrap/bootswatch**

1. Allez sur le site : [Bootswatch.com](https://bootswatch.com/)
2. Choisissez un thème et télécharger le CSS (Celui de la solution est **United**, couleurs Halloween)
3. Dans le dossier du projet, remplacez le fichier **wwwroot/lib/bootstrap/dist/css/bootstrap.css** par celui téléchargé
4. Testez: NON concluant pour les couleurs
5. Sur le site de Bootswatch, allez voir Preview du modèle choisi.

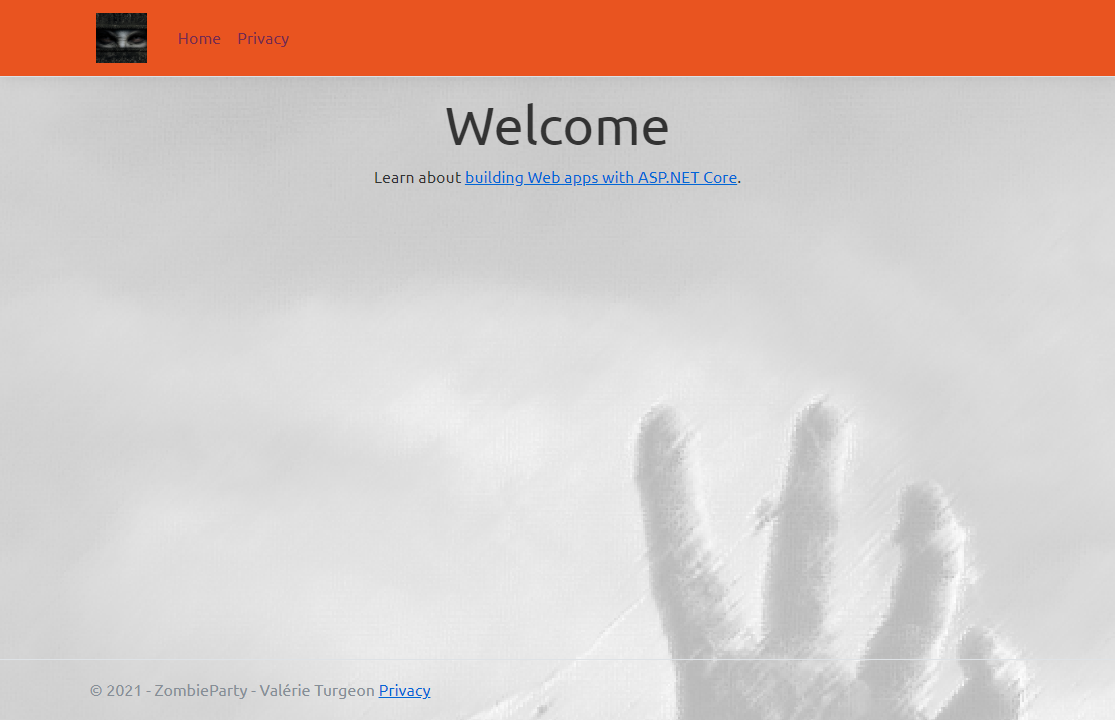
**Modifier \_Layout,** rappel de BootStrap

1. Dans **Views/Shared/\_Layout,** remplacez les deux classes de la **nav bar:**
   * navbar-light
   * bg-white

**Par:**

* + navbar-dark
  + bg-primary

1. Testez (il sera peut-être nécessaire de vider la cache du navigateur)
2. Remplacez le nom de l’application (ZombieParty)dans le coin gauche de la **nav bar** par l'image **images/imagesLayout/tile.png** ( balise img, width:80%)
3. Testez
4. Ajoutez votre nom dans le **Footer,** après le nom de l'application



## 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**: DGN Design du Projet Web
   * **Description**: Modification du CSS, application de la librairie Bootswatch et ajout des images du design.
5. Cliquez sur **Commit**

NOTE: les changements sont enregistrés dans le repo. LOCAL SEULEMENT

1. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 

Sur GitHub, dans votre Repository A21\_3W6\_Labos\_NOM\_PRENOM:

1. Allez sur la branche de la séance, ouvrez le dossier de la solution **ZombieParty**
2. Cliquez sur **History**
3. Visualisez la liste des modifications

# Fonctionnalité Zombie

## RunTime Compilation

Si vous avez oubliez de cocher l'option: Activer la compilation de runtime Razor à la création du projet:

1. Installez le package

Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation

1. Dans **startup**, ajoutez le code en vert:

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews().AddRazorRuntimeCompilation();

}

## Affichage de la liste des Zombies

### Création du *Model* Zombie

1. Dans **Models**, créez la classe publique Zombie
2. Ajoutez les propriétés suivantes à la classe:
   * Name (string)
   * Type (string)
   * Point (int)

### Création du *controller* Zombie

1. Ajoutez le controller **Zombie** (Controller MVC – vide)
2. Dans l’action **Index**, ajoutez une nouvelle liste de Zombies dans le **ViewBag**

public IActionResult Index()

{

this.ViewBag.MaListe = new List<Zombie>()

{

new Zombie(){Name= "LeChuck", Point=5, Type="Fiction",},

new Zombie(){Name= "Lenore", Point=4, Type="Fiction",},

new Zombie(){Name= "Draugr", Point=2, Type="Légendaire",},

new Zombie(){Name= "Ragamuffin", Point=5, Type="Légendaire",},

new Zombie(){Name= "Taxidermy", Point=1, Type="Légendaire",},

new Zombie(){Name= "chien de l'enfer", Point=7, Type="Fiction",},

new Zombie(){Name= "Avogadro", Point=9, Type="Fiction",}

};

return View();

}

1. Ajoutez un Point d’arrêt (BreakPoint) sur **Index**
2. Démarrez l’application
3. Dans le navigateur, **LocalHost :nnnnn/Zombie**
4. Dans VS, allez au pas principal suivant :**F10,** consultez le contenu du ViewBag
5. Faites F10, verifier le contenu du **ViewBag**
6. Continuez, arrêtez l’application (la View Index n’existe pas)

### Création de la view Zombie/Index

1. Créez la view **Index** partir du Controller (bouton droit sur Index()), View Razor vide
2. Remplacez le contenu par défaut de la View par le contenu du fichier *S01\_FichiersComplementaires/* *View Zombie Index.txt*
3. Testez avec le URL **/Zombie**
4. Ajoutez le code nécessaire pour afficher les informations de chaque élément de la liste (NAME et POINT)
5. Testez avec le URL **/Zombie**

Le contenu de la liste devrait apparaître avec une présentation moche

1. Ajoutez des classes bootstrap et des div afin de présenter les Zombies dans des **Card** Bootstrap

<div class="container p-3">

<div class="row">

@foreach (var item in this.ViewBag.MaListe)

{

<div href="#" class="card border m-3 text-center bg-warning" style="width: 20rem; background-image: url('/images/imagesLayout/Fond.png'); ">

<div class="card-body text-warning">

<h3 class="card-title"> @item.Name </h3>

<h5> @item.Point </h5>

</div>

</div>

}

</div>

</div>

1. Tester avec le URL **/Zombie**

### Ajouter un menu deroulant pour atteindre la view Index de Zombie

1. Sur le site de bootstrap, copiez le contenu de la **Navbar,** seulement la partie du <li> de la classe **nav-item dropdown**

<https://getbootstrap.com/docs/5.0>

1. Ajoutez (collez) le contenu dans \_**Layout,** après **Privacy**
2. Changez le nom du menu pour **Zombies** et la 1ère Action pour **Manage** Zombies, qui affiche la view Index de Zombie.
3. Tester le menu : visible mais non fonctionnel
4. Sur le site de Bootstrap, section Introduction, copiez le script js (Bundle) qui permet de faire fonctionner le menu
5. Collez le script, après le dernier script à la fin du Layout

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-gtEjrD/SeCtmISkJkNUaaKMoLD0//ElJ19smozuHV6z3Iehds+3Ulb9Bn9Plx0x4" crossorigin="anonymous"></script>

NOTE: nous reviendrons sur js dans les prochaines séances.

1. Tester le menu

## 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 Zombies Index
   * **Description**: Ajout de la classe Zombie, le Controller Zombie et la View Index affichant la liste des zombies. Ajouter Menu de gestion avec bootstrap.
5. Cliquez sur **Commit**

NOTE: les changements sont enregistrés dans le repo. LOCAL SEULEMENT

1. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 

# Fonctionnalité ZombieType

## Fonctionnalités Créer et Afficher ZombieType

### Création du Model ZombieType

1. Créez le modèle ZombieType
   * Id (int)
   * TypeName (string)

### Création du Controller ZombieType

1. Créez le Controller pour ZombieType (MVC vide)
2. Dans l’action **Index**, ajoutez une nouvelle liste de ZombieTypes dans le **ViewBag**

public IActionResult Index()

{

this.ViewBag.MaListe = new List<ZombieType>()

{

new ZombieType(){TypeName= "Virus", Id=1},

new ZombieType(){TypeName= "Contact", Id=2}

};

return View();

}

### Création de la View ZombieType/Index

1. Créez la view **Index** partir du Controller (bouton droit sur Index()), View Razor vide
2. Remplacez le contenu par défaut de la View par le contenu du fichier *S01\_FichiersComplementaires/* *View ZombieType Index.txt*
3. Testez dans le navigateur, **LocalHost :nnnnn/ZombieType**
4. Ajoutez le code nécessaire pour afficher les informations de chaque élément de la liste
5. Testez dans le navigateur, **LocalHost :nnnnn/ZombieType**

### Ajouter dans le menu ZombieType

1. Modifiez le \_Layout
2. Changez la 2e Action pour **Manage** Types, qui affiche la view **Index** de **ZombieType**.
3. Testez le menu

## Fonctionalité Create ZombieType

### Ajouter l’action CREATEau Controller

1. Dans le ZombieTypeController, ajoutez l’action Create qui retourne la View du même nom

//GET CREATE

public IActionResult Create()

{

return View();

}

1. À partir de la View ZombieType/**Index,** modifiez le bouton **Create New Type** afin qu’il affiche la View ZombieType/**Create**

### Ajouter la View ZombieType CREATE

1. Créez la view **Create** partir du Controller (bouton droit sur Index()), View Razor vide
2. Remplacez le contenu par défaut de la View par le contenu du fichier *S01\_FichiersComplementaires/* *View ZombieType Create.txt*
3. Testez
4. Ajoutez la référence, l’injection, au model **ZombieType** en haut de la View Create
5. Ajoutez les TagHelpers pour le Form, les labels et les inputs
6. Ajoutez le tagHelper pour le bouton Back To List qui nous ramène à la view **Index**
7. Testez

# Validation règles du model

### Ajouter la validation du model Côté Serveur

1. Ajoutez des annotations au model **ZombieType** pour l’affichage et la validation

|  |  |
| --- | --- |
| Élément | Règle |
| TypeName | S’affiche Type Name |
| TypeName | Obligatoire  Message d’erreur : "Type Name have to be fill." |

1. Ajoutez tagHelpers et les balises nécessaires à la validation du Model dans la View ZombieType/**Create**
2. Ajoutez l’Action **Create** POST dans le Controller
3. Ajoutez un point d’arrêt (BreakPoint) dans le Controller **ZombieType**
4. Testez des données Valides et des données invalides
5. En bas de la View **ZombieType/Create**, ajouter l’appel à la Partial View de validation côté client

@section Scripts{

@{ <partial name="\_ValidationScriptsPartial" /> }

}

1. Testez des données Valides et des données invalides

## 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 ZombieType Index, Create et Validation
   * **Description**: Ajout de la validation côté client et côté serveur ZombieType, View Create.
5. Cliquez sur **Commit**

NOTE: les changements sont enregistrés dans le repo. LOCAL SEULEMENT

1. "Poussez" **Push** les modifications sur le repo en ligne **Remote**: cliquez sur 