Laboratoire: semaine 8

# Objectif(s)

* Champs spéciaux
  + TextAreas
  + Gestion de fichiers
* Authentification
  + MS Identity
  + Modification de la page Register

# Créer la branche pour les laboratoires

## Création de la branche de la séance

1. Dans Gitkraken, clonez votre repository de laboratoire
2. Créez la branche de la séance **S08\_Authentification\_ChampsSpeciaux**
3. *Comment-commit-push*

# Ajouter des icônes aux boutons

## Installer la librairie FOnt Awesome

1. Allez sur le site de [Font Awesome](https://fontawesome.com)
2. Cliquez sur **Start**
3. Téléchargez la librairie
4. « Dézipez » le fichier dans le dossier **root/** et standardiser le nom du dossier. ATTENTION aux sous-dossiers inutiles
5. Ajoutez la référence relative dans l’en-tête du **\_Layout**

<link href="~/fontawesome/css/all.css" rel="stylesheet">

## Ajouter les icônes aux boutons

1. Sur le site de Font Awesome, cliquez sur **Icons** cherchez un icône pour chacun des boutons du site à l’aide de mots clés :
   * Back to list : left arrow
   * New/Create : plus
   * Edit/Update : Edit
   * Delete : Delete
   * Details
   * Etc..
2. Cliquez sur **Free** afin de limiter la recherche aux icônes gratuits
3. Cliquez sur l’icône choisi, puis cliquez sur le HTML exemple : <i class="fas fa-plus-circle"></i>
4. Dans les views Home Index, Create et Edit, Upsert, Delete, etc, collez le HTML avant le texte du bouton et ajoutez un espace &nbsp;

**Exemple :**

<div class="col-3 text-right">

           <a **asp-controller**="Home" **asp-action**="Create" class="btn btn-info">

               <i class="fas fa-plus"></i> &nbsp; Create New Travel</a>

# implanter un text Area

## Propriétés visées par TextArea

* Travel : Description
* TravelRecommendation : Description

## Utiliser la librairie Summernote

1. Allez sur le site [Summernote](https://summernote.org/), dans la section **Include js/css**
2. Copiez les link et script
3. Ajoutez les références requises dans le **\_Layout**

NOTE : les références pour Bootstrap et JQuery sont déjà dans le \_layout proposé par Microsoft

**Dans l’en-tête :**

<head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>@ViewData["Title"] - ChampsSpeciaux</title>

    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

    <link rel="stylesheet" href="~/fontawesome-free-5.15.4-web/css/all.css" />

    <link href="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote.min.css" rel="stylesheet">

    <link rel="stylesheet" href="~/css/site.css" />

</head>

**Dans le pied de page**

 </footer>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>

    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote.min.js"></script>

    <script **src**="~/js/site.js" **asp-append-version**="true"></script>

    @await RenderSectionAsync("Scripts", required: false)

</body>

## Mettre en place le textArea

1. Dans les Views visées, convertissez le *Input* **Description** en TextArea
2. Ajoutez un Id OU une classe **summernote**.

NOTE : si vous avez plus d’un champ dans la View sur lesquels appliquer un Textarea Summernote, utilisez la 2e méthode, soit la classe. L’exemple du site utilise le Id #summernote. Nous utiliserons la classe .summernote

<div class="col-8">

<textarea **asp-for**="Description" class="form-control summernote"></textarea>

<span **asp-validation-for**="Description" class="text-danger"></span>

</div>

1. Ajoutez un script Javascript JQuery en bas de la View qui active la librairie sur le champ après le chargement de la page.

Ce que fait le script : lorsque tous les éléments du DOM de la page sont chargés et disponibles, exécuter la méthode **summernote** sur tous les éléments ayant la classe **summernote** (sélector de classe identifié par le POINT.

@section Scripts{

    <**partial** **name**="\_ValidationScriptsPartial" />

    <script>

        $(document).ready(function () {

            $('.summernote').summernote();

        });

    </script>

}

1. Testez le champ, exécutez l’application

# implanter la gestion des images

## Propriétés visées par TextArea

* Travel : Photo (AJOUTER dans Model)
* Country : Flag (AJOUTER dans Model)
* Guide : Photo (AJOUTER dans Model)

## Ajouter un classe pour les constantes du projet

1. Créez un Projet (Bibliothèque/*Librairy*) appelé **Utility** et une classe appelée **AppConstants**
2. Définissez une consante pour chaque élément : Pour Country, Travel et Guide
   * Chemin d’accès pour les images
   * Chemin d’accès pour les images dans les VIews

**Exemple :**

public static class AppConstants

   {

       public static string ImageGuidePath = @"images\Guides\";

public static string ImageGuidePathView = @"/images/Guides/";

   }

## Ajouter la référence à la classe pour les constantes dans les Views

1. Ajoutez la référence dans la \_View\_Import

@using ChampsSpeciaux

@using ChampsSpeciaux.Models

@using ChampsSpeciaux.Utility;

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

## Ajouter le nécessaire pour la gestion des constantes du site

1. Dans le Home Controller, ajoutez la gestion des constantes du site via le Framework Buiding dependency
   * Avec l’intellisence choisissez : Microsoft.AspNetCore.Hosting

public class HomeController : Controller

  {

      private readonly ILogger<HomeController> \_logger;

      private readonly IUnitOfWork \_unitOfWork;

      private readonly IWebHostEnvironment \_webHostEnvironment;

      public HomeController(ILogger<HomeController> logger, IUnitOfWork unitOfWork

, IWebHostEnvironment webHostEnvironment)

      {

          \_logger = logger;

          \_unitOfWork = unitOfWork;

          \_webHostEnvironment = webHostEnvironment;

      }

## Ajouter la gestion des images dans Views **Edit/Create/Upsert**

1. Dans le Controller, l’action Edit/Create/Upsert (GET), modifiez le code si nécessaire
2. Dans la View Edit/Create/Upsert, modifiez le type du Input pour l’image pour **file**, attribut **name** pour  **files**

## Le controller Action **Create/EDIT/UPSERT GET**

1. Testez la création

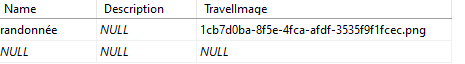
NOTE : Le Get n’enregistre pas dans la BD.

## Le controller Action **Create/EDIT/UPSERT POST**

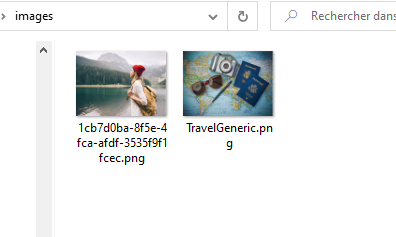
1. Dans le Controller, l’action Edit/Create/Upsert (POST), ajoutez le code pour la gestion des images

## Tester la view

1. Testez la View
2. Dans SQL Server, vérifiez l’inscription du fichier, le code, nom donné à l’image sera évidemment différent



1. Dans le dossier **Root/images,** vérifiez que l’image avec le nouveau nom a été transférée



## Ajouter la gestion des images dans View

1. Dans le Controller, l’action Edit (GET), rien à changer pour l’instant
2. Dans la View Edit, modifiez le type du Input pour TravelImage pour **file**, attribut **name** pour  **files**

<input **type**="file" **name**="files" id="uploadBox" **asp-for**="TravelImage" class="form-control" />

<span **asp-validation-for**="TravelImage" class="text-danger"></span>

1. Modifiez l’attribut **enctype** du form

<form class="form-horizontal" method="post" enctype="multipart/form-data" **asp-action**="Edit">

1. Ajoutez le code qui affiche l’image générique si TravelImage est vide, sinon l’image enregistrée

@if (Model.TravelImage != "")

       {

         <img src="@AppContants.ImagePathView@Model.TravelImage" id="TravelImage" width="100%" style="border-radius:5px; border:1px solid #bbb9b9" />

       }

       else

       {

         <img src="@AppContants.ImagePathVIew@AppContants.ImageGeneric" id="TravelImage" width="100%" style="border-radius:5px; border:1px solid #bbb9b9" />

       }

## Le controller Action **EDIT GET**

1. Testez le Edit de l’enregistrement de Travel créé précédemment.

## Le controller Action **Edit POST**

1. Dans le Controller, l’action Edit (POST), ajoutez le code afin d’enregistrer l’image modifiée ou de conserver celle originale

if (ModelState.IsValid)

     {

       string webRootPath = \_webHostEnvironment.WebRootPath; //Chemin des images de voyage: Travel

       var files = HttpContext.Request.Form.Files; //nouvelle image récupérée

       if (files.Count > 0)

       {

         string fileName = Guid.NewGuid().ToString();// Nom fichier généré, unique

         var uploads = Path.Combine(webRootPath, AppContants.ImagePath);// chemin pour les image

         var extenstion = Path.GetExtension(files[0].FileName); // extraire l'extention du fichier

if (travel.TravelImage != null)

         {

           //L'image est modifiée: l'ancienne doit être supprimée

           var imagePath = Path.Combine(webRootPath, travel.TravelImage.TrimStart('\\'));

           if (System.IO.File.Exists(imagePath))

           {

             System.IO.File.Delete(imagePath);

           }

         }

         // Create un canal pour transférer le fichier

         using (var filesStreams = new FileStream(Path.Combine(uploads, fileName + extenstion), FileMode.Create))

         {

           files[0].CopyTo(filesStreams);

         }

         // Composer le nom du fichier avec son extension qui sera enregister dans la BD

         // avec le path relatif à partir du Root

         // sans le path relatif (le path devra être ajouté dans la View)

         travel.TravelImage = fileName + extenstion;

       }

       else

       {

         // L'image n'est pas modifiée: on récupère l'information

         // afin qu'elle ne soit pas perdue après l'update

          Travel objFromDb = await \_unitOfWork.Travel.GetAsync(travel.Id);

           travel.TravelImage = objFromDb.TravelImage;

       }

       \_unitOfWork.Travel.Update(travel);

           \_unitOfWork.Save();

           return RedirectToAction(nameof(Index));

    }

     else

     {

       return View(travel);

     }

   }

# implanter les notifications élaborées

## Ajouter les constantes dans le projet **Utility**

1. Dans la classe **AppConstants**
2. Définissez une consante pour chaque élément : pour éviter les chaînes « magiques » harcodées
   * Success
   * Echec

public static class AppConstants

   {

….

       public const string Success = "Success";

        public const string Error = "Error";

   }

## Ajouter la référence ;a la librairie Toastr

1. Dans l’en-tête du layout, ajoutez la référence à la librairie Toastr

<head>

  <meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - ChampsSpeciaux</title> <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<link href="~/fontawesome/css/all.css" rel="stylesheet"> <link href="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote.min.css"

rel="stylesheet">

<link rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/css/toastr.min.css"

crossorigin="anonymous" />

<link rel="stylesheet" href="~/css/site.css" />

</head>

1. Dans le pied de page du layout, ajoutez la référence à la librairie Toastr

 </footer>

  <script src="~/lib/jquery/dist/jquery.min.js"></script>

  <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

  <script **src**="~/js/site.js" **asp-append-version**="true"></script>

  <script src="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote.min.js"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/js/toastr.min.js"></script>

  @await RenderSectionAsync("Scripts", required: false)

</body>

## Ajouter une Partial View pour les notifications

1. Créez une Partial View appelée **\_Notifications**
2. Ajoutez dans l’en-tête de la view une référence aux constantes de *Utility*
3. Ajoutez une condition qui prend l’information envoyée via le TempData et affiche le message Toastr correspondant

@using ChampsSpeciaux.Utility

@if (TempData[AppContants.Success] != null)

{

  <script src="~/lib/jquery/dist/jquery.min.js"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/js/toastr.min.js"></script>

  <script type="text/javascript">

  toastr.success('@TempData[AppContants.Success]');

  </script>

}

@if (TempData[AppContants.Error] != null)

{

  <script src="~/lib/jquery/dist/jquery.min.js"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/js/toastr.min.js"></script>

  <script type="text/javascript">

  toastr.error('@TempData[AppContants.Error]');

  </script>

}

## Ajouter le transfert de l’information via TempData

1. Ajoutez le code permettant d’appeler les Notifications via TempData dans les controllers

**Exemple Create**

// Ajouter à la BD

       await \_unitOfWork.Travel.AddAsync(travel);

           \_unitOfWork.Save();

       TempData[AppContants.Success] = "The travel was created.";

       return RedirectToAction(nameof(Index));

       }

     TempData[AppContants.Error] = "An error occurred while creating the Travel.";

     return this.View(travel);

**Exemple Edit**

\_unitOfWork.Travel.Update(travel);

          \_unitOfWork.Save();

      TempData[AppContants.Success] = "The travel was updated.";

      return RedirectToAction(nameof(Index));

   }

    else

    {

      TempData[AppContants.Error] = "An error occurred while updating the Travel.";

      return View(travel);

# Implanter l’Authentification MS Identity

## Installer les packages dans les projets

* Microsoft.AspNetCore.Identity.EntityFrameworkCore
  + Projet Web/MVC
  + Projet DataAccess
* Microsoft.AspNetCore.Identity.UI
  + Projet Web/MVC
* Microsoft.Extensions.Identity.Stores
  + Projet Models

## Configurer le MS Identity

1. Ajoutez l’héritage au DbContext à **IdentityDbContext**
2. Créez une classe ApplicationUser héritant de *IdentityUser*
   * Ajouter *Nickname*
3. Modifiez la page *Register* afin que les champs suivants soient affichés :
   * Nickname
   * FirstName
   * LastName

Injectez dans **Startup** Identity

services.AddDefaultIdentity<ApplicationUser>()

.AddEntityFrameworkStores<ApplicationDbContext>();

…

//Contrôle ajouté au pipeline http

app.UseAuthentication();

app.UseAuthorization();

# Ajouter identification ThirdParty

Au choix:

Google ou Facebook