Laboratoire: Champs Spéciaux ThirdParty

# Objectif(s)

* Ajouter des icônes aux boutons ou pages
* Gérer les champs spéciaux :
  + TextArea
  + Images

# Créer La branche pour la séance

## Création de la branche de la séance

1. Dans Gitkraken, clonez votre repository de laboratoire
2. Créez la branche de la séance **S07\_ThirdParty\_1** à partir de la branche **Main**
3. Téléchargez le contenu de la branche de l'enseignant, les **S07\_fichiers\_complementaires\_1**
4. Pour la 1ère partie du laboratoire, vous devez partir avec la solution de l’enseignant
5. *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 trois boutons du site à l’aide de mots clés :
   * Back to list : left arrow
   * New/Create : plus
   * Edit/Update : Edit
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, 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

## 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 **Create** et **Edit,** 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 () {

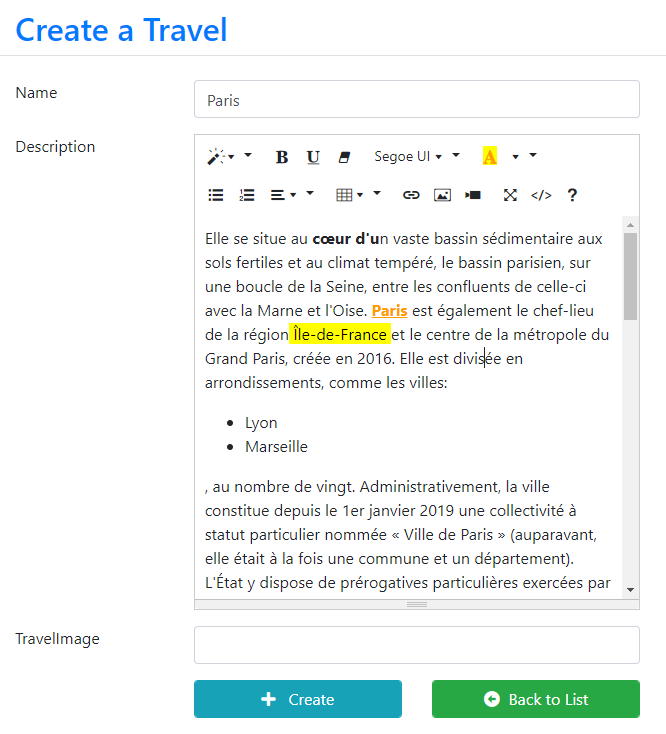
            $('.summernote').summernote();

        });

    </script>

}

1. Testez le champ, exécutez l’application, allez à la view Create Travel
2. Remplissez le champ Name : Paris
3. Copiez-coller le contenu du texte disponible dans le fichier **Paragraphes de texte Champ Description.docx**
4. Sélectionnez des portions de texte et modifier la mise en forme.



1. Cliquez sur Create
2. Refaites les étapes pour la view Edit
3. Sur la View Index, cliquez sur Edit vis-à-vis Paris et cliquez sur Edit
4. Vous devriez voir vos modifications de mise en forme

# implanter la gestion des images

NOTE : par défaut, une image générique (située dans Root/images/, est affichée dans la balise img de la View Create. Dans la View Edit, celle-ci sera affichée si une image n’est pas associée

## Ajouter un classe pour les constantes du projet

1. Créez un dossier et une classe **Utility/AppConstants**
2. Définissez une consante pour chaque élément :
   * Chemin d’accès pour les images
   * Chemin d’accès pour les images dans les VIews
   * Nom de l’image générique pour les voyages

public static class AppConstants

   {

       public static string ImagePath = @"images\";

public static string ImagePathView = @"/images/";

    public static string ImageGeneric = "TravelGeneric.png";

   }

## 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 ApplicationDbContext \_db;

      private readonly IWebHostEnvironment \_webHostEnvironment;

      public HomeController(ILogger<HomeController> logger, ApplicationDbContext applicationDbContext

, IWebHostEnvironment webHostEnvironment)

      {

          \_logger = logger;

          \_db = applicationDbContext;

          \_webHostEnvironment = webHostEnvironment;

      }

## Ajouter la gestion des images dans View Create

1. Dans le Controller, l’action Create (GET), rien à changer pour l’instant
2. Dans la View Create, 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. Dans la zone Script (en bas) de la *View*, ajoutez le script suivant :

function validateInput() {

    if (document.getElementById("uploadBox").value == "") {

      swal("Error", "Please select an image", "error");

      return false;

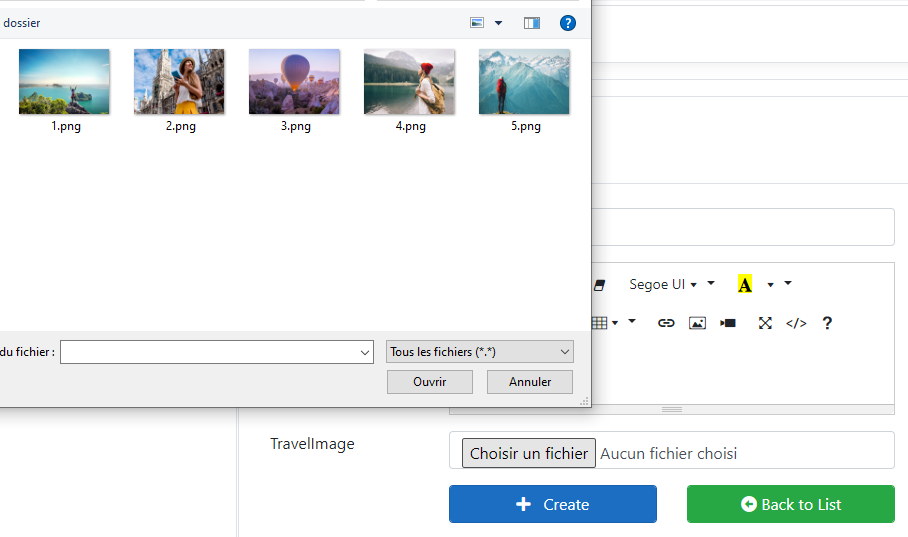
    }

    return true;

  };

## Le controller Action **Create GET**

1. Testez la création de Travel à l’aide des fichiers png fournis dans les fichiers complémentaires



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

## Le controller Action **Create POST**

1. Dans le Controller, l’action Create (POST)

//POST CREATE

   [HttpPost]

   public async Task<IActionResult> Create(Travel travel)

   {

       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, AppConstants.ImagePath);// chemin pour les image

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

         // Create un cannal 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;

       }

       // Ajouter à la BD

       await \_db.Travel.AddAsync(travel);

           await  \_db.SaveChangesAsync();

           return RedirectToAction(nameof(Index));

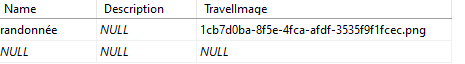
       }

     return this.View(travel);

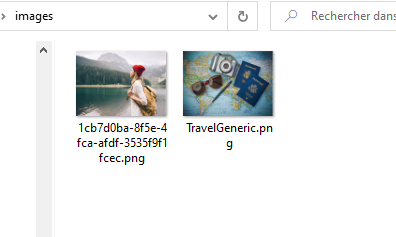
   }

## Tester la view Create

1. Testez la View Create, Travel  **Randonnée**, sélectionnez l’image **4.png**
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 EDIT

1. Dans le Controller, l’action Edit (GET), rien à changer pour l’instant (NOTE : Update n’a pas de Async)
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="@AppConstants.ImagePathView@Model.TravelImage" id="TravelImage" width="100%" style="border-radius:5px; border:1px solid #bbb9b9" />

       }

       else

       {

         <img src="@AppConstants.ImagePathView@AppConstants.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. NOTE : Mettre en Async pour le SaveChangesAsync

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, AppConstants.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 \_db.Travel.GetAsync(travel.Id);

           travel.TravelImage = objFromDb.TravelImage;

       }

       \_db.Travel.Update(travel);

           Await \_db.SaveChangesAsync();

           return RedirectToAction(nameof(Index));

    }

     else

     {

       return View(travel);

     }

   }

# implanter les notifications élaborées

## 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 les constantes de notifications

1. Dans Utility, AppConstants, ajoutez les constantes suivantes :

// Pour les notifications Toastr

      public const string Success = "Success";

      public const string Error = "Error";

## 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[AppConstants.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[AppConstants.Success]');

  </script>

}

@if (TempData[AppConstants.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[AppConstants.Error]');

  </script>

}

## Ajouter le transfert de l’information via TempData

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

**Create**

// Ajouter à la BD

               await \_db.Travel.AddAsync(travel);

              await \_db.SaveChangesAsync();

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

              return RedirectToAction(nameof(Index));

           }

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

           return this.View(travel);

**Edit**

           \_db.Travel.Update(travel);

           await \_db.SaveChangesAsync();

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

            return RedirectToAction(nameof(Index));

        }

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

        return View(travel);