**Mini Projet ASP.NET MVC : Gestion de Contacts**

**Étape 1 : Créer un Projet ASP.NET MVC**

1. **Ouvrir Visual Studio**.
2. **Créer un nouveau projet** :
   * Type : **ASP.NET Web Application (.NET Framework)**
   * Modèle : **MVC**
   * Nom du projet : GestionContacts
3. **Lancer le projet** pour s’assurer que tout fonctionne.

### ****Étape 2 : Créer le Modèle (Model)****

Dans le dossier **Models**, ajoute un fichier Contact.cs :

using System.ComponentModel.DataAnnotations;

namespace GestionContacts.Models

{

public class Contact

{

public int Id { get; set; }

[Required]

public string Nom { get; set; }

[Required, EmailAddress]

public string Email { get; set; }

[Required, Phone]

public string Telephone { get; set; }

}

}

**Explication :**

* **Id** : Identifiant unique.
* **Nom** : Nom du contact (**obligatoire**).
* **Email** : Email (**validation automatique**).
* **Téléphone** : Numéro de téléphone (**validation automatique**).

### ****Étape 3 : Créer le Contrôleur (Controller)****

Dans **Controllers**, ajoute un contrôleur **ContactsController** :

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using GestionContacts.Models;

namespace GestionContacts.Controllers

{

public class ContactsController : Controller

{

private static List<Contact> contacts = new List<Contact>(); // Liste temporaire

public ActionResult Index()

{

return View(contacts); // Affiche tous les contacts

}

public ActionResult Create()

{

return View();

}

[HttpPost]

public ActionResult Create(Contact contact)

{

if (ModelState.IsValid)

{

contact.Id = contacts.Count + 1; // Génération d'un ID unique

contacts.Add(contact);

return RedirectToAction("Index");

}

return View(contact);

}

public ActionResult Delete(int id)

{

var contact = contacts.FirstOrDefault(c => c.Id == id);

if (contact != null)

contacts.Remove(contact);

return RedirectToAction("Index");

}

}

}

**Explication :**

* Index() : Affiche la liste des contacts.
* Create() (GET) : Affiche le formulaire d'ajout.
* Create() (POST) : Ajoute un contact.
* Delete(id) : Supprime un contact.

### ****Étape 4 : Créer les Vues (Views)****

#### 📝 **Vue** Index.cshtml (Liste des contacts)

📁 **Views/Contacts/Index.cshtml**

@model List<GestionContacts.Models.Contact>

@{

ViewBag.Title = "Liste des Contacts";

}

<h2>Contacts 📞</h2>

<a href="@Url.Action("Create")" class="btn btn-primary">Ajouter un Contact</a>

<table class="table">

<tr>

<th>Nom</th>

<th>Email</th>

<th>Téléphone</th>

<th>Actions</th>

</tr>

@foreach (var contact in Model)

{

<tr>

<td>@contact.Nom</td>

<td>@contact.Email</td>

<td>@contact.Telephone</td>

<td>

<a href="@Url.Action("Delete", new { id = contact.Id })" class="btn btn-danger">Supprimer</a>

</td>

</tr>

}

</table>

#### **Vue** Create.cshtml (Formulaire d’ajout)

📁 **Views/Contacts/Create.cshtml**

@model GestionContacts.Models.Contact

@{

ViewBag.Title = "Ajouter un Contact";

}

<h2>Ajouter un Contact</h2>

@using (Html.BeginForm())

{

<div class="form-group">

@Html.LabelFor(m => m.Nom)

@Html.TextBoxFor(m => m.Nom, new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.Nom)

</div>

<div class="form-group">

@Html.LabelFor(m => m.Email)

@Html.TextBoxFor(m => m.Email, new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.Email)

</div>

<div class="form-group">

@Html.LabelFor(m => m.Telephone)

@Html.TextBoxFor(m => m.Telephone, new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.Telephone)

</div>

<button type="submit" class="btn btn-success">Ajouter</button>

}

**Étape 5 : Configuration et Test**

1. **Lancer le projet** (F5 ou Ctrl+F5).
2. Aller sur **/Contacts** pour voir la liste des contacts.
3. Ajouter un contact via **"Ajouter un Contact"**.
4. Supprimer un contact via **"Supprimer"**.

### ****Résumé****

| **📌 Étape** | **📜 Description** |
| --- | --- |
| **Modèle** | Création de Contact.cs |
| **Contrôleur** | Création de ContactsController.cs |
| **Vues** | Index.cshtml (Liste) & Create.cshtml (Formulaire) |
| **Test** | Ajouter et supprimer des contacts |

Ce projet utilise **une liste en mémoire** pour stocker les contacts.  
Pour une **persistance réelle**, on peut utiliser **Entity Framework et une base de données SQL**.

**enregistrer la liste List<Contact> contacts dans un fichier ?**

Solution recommandée : Utilisation de JSON

Ajoute **Newtonsoft.Json** (si non installé) :

1. **Ouvre le terminal NuGet** (Outils > Gestionnaire de Packages NuGet > Console du Gestionnaire de Package).
2. **Tape la commande suivante**

Install-Package Newtonsoft.Json

1. **Modifier** ContactsController **pour gérer la sauvegarde**

Ajoute ces **méthodes** dans ContactsController.cs :

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web.Mvc;

using GestionContacts.Models;

using Newtonsoft.Json;

namespace GestionContacts.Controllers

{

public class ContactsController : Controller

{

private static string filePath = System.Web.Hosting.HostingEnvironment.MapPath("~/App\_Data/contacts.json");

private static List<Contact> contacts = ChargerContacts();

// 🔹 Charger les contacts depuis le fichier JSON

private static List<Contact> ChargerContacts()

{

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

{

string json = System.IO.File.ReadAllText(filePath);

return JsonConvert.DeserializeObject<List<Contact>>(json) ?? new List<Contact>();

}

return new List<Contact>();

}

// 🔹 Sauvegarder les contacts dans un fichier JSON

private void SauvegarderContacts()

{

string json = JsonConvert.SerializeObject(contacts, Formatting.Indented);

System.IO.File.WriteAllText(filePath, json);

}

public ActionResult Index()

{

return View(contacts);

}

public ActionResult Create()

{

return View();

}

[HttpPost]

public ActionResult Create(Contact contact)

{

if (ModelState.IsValid)

{

contact.Id = contacts.Count > 0 ? contacts.Max(c => c.Id) + 1 : 1;

contacts.Add(contact);

SauvegarderContacts(); // 🔹 Sauvegarde après ajout

return RedirectToAction("Index");

}

return View(contact);

}

public ActionResult Delete(int id)

{

var contact = contacts.FirstOrDefault(c => c.Id == id);

if (contact != null)

{

contacts.Remove(contact);

SauvegarderContacts(); // 🔹 Sauvegarde après suppression

}

return RedirectToAction("Index");

}

}

}

**Étape 6 : Ajouter package Newtonsoft.Json dans visual studio**

### ****6.1 : Ouvrir le terminal****

Dans **VS Code**, ouvre le terminal :

* **Windows** : Ctrl + ' (touche backtick)
* **Mac/Linux** : Ctrl + Shift + P, puis tape **"Terminal: Create New Terminal"**

### ****6.2 : Installer**** Newtonsoft.Json ****avec**** dotnet add package

Dans le terminal, exécute cette commande :

*dotnet add package Newtonsoft.Json*

### ****6. 3 : Vérifier l'installation****

1. Ouvre le fichier \*.csproj de ton projet.
2. Tu devrais voir cette ligne ajoutée :

<PackageReference Include="Newtonsoft.Json" Version="13.0.3" />

### ****6.4 : Utiliser**** Newtonsoft.Json

Dans tes fichiers C#, ajoute :

using Newtonsoft.Json;

Puis utilise **JSON pour sérialiser/désérialiser** tes objets :

string json = JsonConvert.SerializeObject(monObjet, Formatting.Indented);

MonObjet obj = JsonConvert.DeserializeObject<MonObjet>(json);