

# **TP N°1**

**Ma première application web utilisant**

**ASP .NET MVC core**

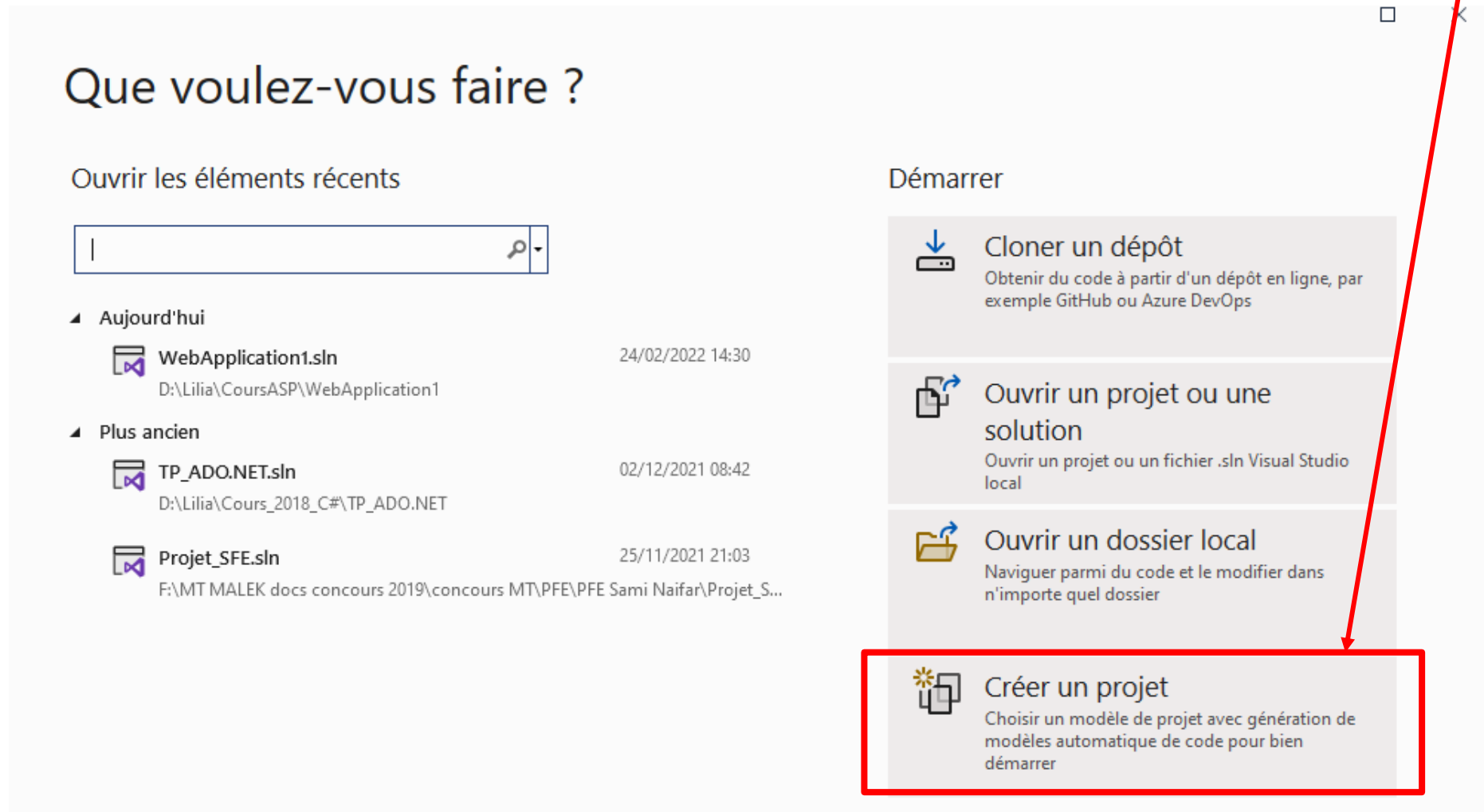
**GESTION DES EMPLOYÉS**

**Enseignant : Malek Zribi**



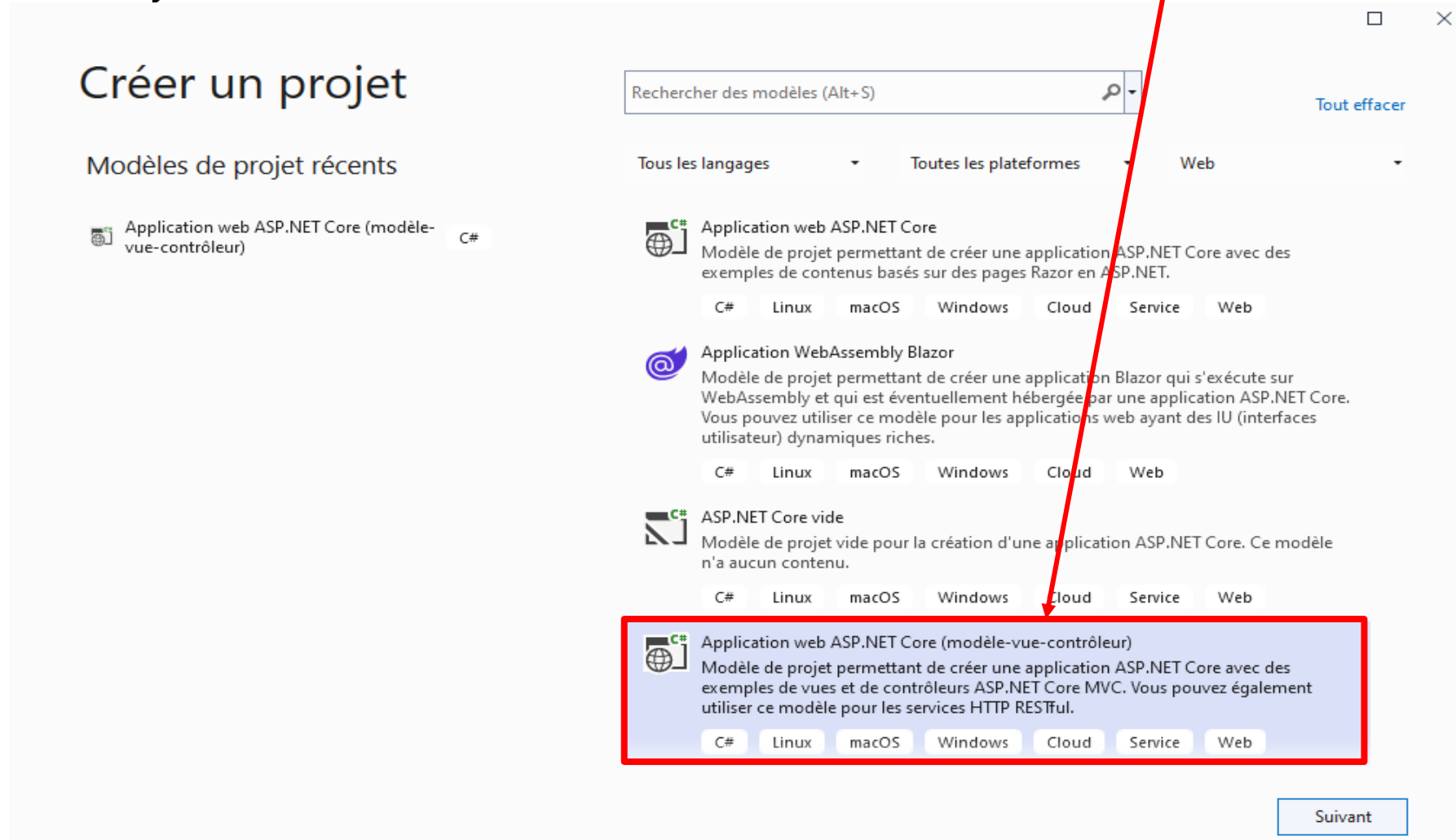
# CRÉER UN PROJET ASP .NET CORE

- Ouvrez Visual Studio 2022 et cliquez sur **Créer un nouveau projet** , comme indiqué ci-dessous.



# CRÉER UNE APPLICATION ASP .NET CORE

- Choisissez une **Application web ASP .Net Core (Modèle-Vue-Contrôleur)**.



# CRÉER UNE APPLICATION ASP .NET CORE

- Indiquez le nom, l'emplacement et le nom de solution appropriés pour l'application ASP.NET Core, puis cliquez sur **suivant** puis **créer**:

The screenshot displays the ASP.NET Core project creation wizard in two panels. The left panel, titled 'Configurer votre nouveau projet', contains fields for 'Nom du projet' (WebApplication1), 'Emplacement' (C:\Users\USER\source\repos), and 'Nom de la solution' (WebApplication1). The right panel, titled 'Informations supplémentaires', shows settings for 'Framework cible' (.NET 7.0), 'Type d'authentification' (Aucun), and checkboxes for 'Configurer pour HTTPS' (checked), 'Activer Docker', and 'Activer la compilation de runtime Razor'. At the bottom of each panel are 'Retour' and 'Suivant' (left) or 'Retour' and 'Créer' (right) buttons. Red arrows originate from the text 'suivant' and 'créer' in the instruction above, pointing to the 'Suivant' and 'Créer' buttons respectively. A red circle is visible in the bottom right corner of the image.

**Configurer votre nouveau projet**

Application web ASP.NET Core (modèle-vue-contrôleur) C# Linux macOS Windows Cloud Service Web

Nom du projet

WebApplication1

Emplacement

C:\Users\USER\source\repos

Nom de la solution ⓘ

WebApplication1

☐ Placer la solution et le projet dans le même répertoire

Retour Suivant

**Informations supplémentaires**

Application web ASP.NET Core (modèle-vue-contrôleur) C# Linux macOS Windows Cloud Service Web

Framework cible ⓘ

.NET 7.0 (En cours)

Type d'authentification ⓘ

Aucun

☒ Configurer pour HTTPS ⓘ

☐ Activer Docker ⓘ

Or Docker ⓘ

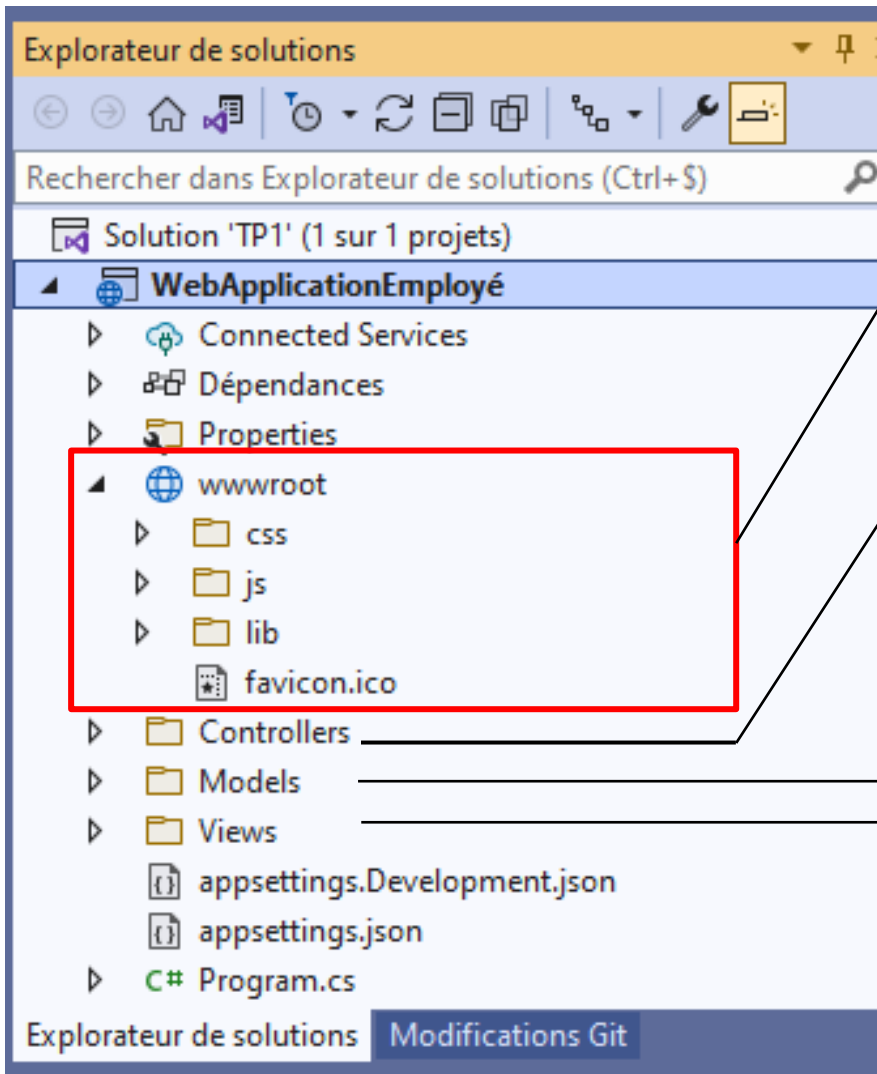
Linux

☐ Activer la compilation de runtime Razor ⓘ

Retour Créer

# CRÉER UNE APPLICATION ASP .NET CORE

- La structure de notre application est la suivante :



Va contenir des dossiers séparés pour les différents types de fichiers statiques tels que JavaScript, CSS, Images, scripts de bibliothèque, etc.

Le contrôleur est chargé de la synchronisation du modèle et de la vue. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer.

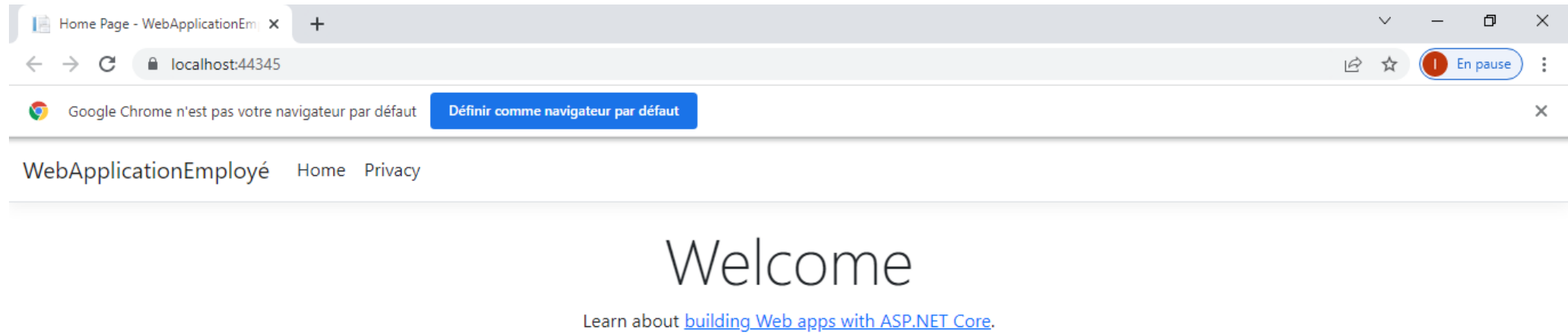
Le modèle contient les données manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité.

La vue fait l'interface avec l'utilisateur. Sa première tâche est d'afficher les données qu'elle a récupérées auprès du modèle. Sa seconde tâche est de recevoir tous les actions de l'utilisateur (clic de souris, sélection d'une entrées, boutons, ...). Ses différents événements sont envoyés au contrôleur.



# EXÉCUTION DE VOTRE APPLICATION

- Pour exécuter cette application Web, cliquez sur **IIS Express** ou appuyez sur **Ctrl + F5**. Cela ouvrira le navigateur et affichera le résultat suivant :



# AJOUT DE LA CLASSE EMPLOYÉ

- Soit le code de la classe Employee suivante :

```
public class Employee
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Departement { get; set; }
    public int Salary { get; set; }
}
```

- On vous demande de l'ajouter dans le dossier **Model** de votre application,



# CRÉER L'INTERFACE IRepository

- Sous le dossier **Model** créer un nouveau dossier Repositories.
- Dans ce nouveau dossier créer l'interface **IRepository** suivante :

```
public interface IRepository<T>
{
    IList<T> GetAll();
    T FindByID(int id);
    void Add(T entity);
    void Update(int id, T entity);
    void Delete(int id);
    List<T> Search(string term);
}
```





# CRÉER LA CLASSE **EmployeeRepository**

- Créer aussi dans le dossier Repositories une classe qui implémente l'interface nommée **EmployeeRepository** et donner le code des méthodes en utilisant LINQ.

```
public class EmployeeRepository : IRepository<Employee> {  
    List<Employee> lemployees;  
    public EmployeeRepository() {  
        lemployees = new List<Employee>()  
        {  
            new Employee {Id=1,Name="Sofien ben ali", Departement= "comptabilité",Salary=1000},  
            new Employee {Id=2,Name="Mourad triki", Departement= "RH",Salary=1500},  
            new Employee {Id=3,Name="ali ben mohamed", Departement= "informatique",Salary=1700},  
            new Employee {Id=4,Name="tarak aribi", Departement= "informatique",Salary=1100}  
        };  
    }  
}
```



# CRÉER LA CLASSE `EmployeeRepository`

```
public void Add(Employee e) {  
    employees.Add(e);  
}  
public Employee FindByID(int id) {  
    var emp = employees.FirstOrDefault(a => a.Id == id);  
  
    return emp;  
}  
public void Delete(int id) {  
    var emp = FindByID(id);  
    employees.Remove(emp);  
}  
public IList<Employee> GetAll() {  
    return employees;  
}  
public void Update(int id, Employee newemployee) {  
    var emp = FindByID(id);  
    emp.Name = newemployee.Name;  
    emp.Département = newemployee.Département;  
    emp.Salaire = newemployee.Salaire;  
}
```



# CRÉER UN CONTRÔLEUR

- Dans le dossier Controllers, Créer un nouveau contrôleur nommé **EmployeeController** (avec read/write actions) permettant de gérer les opérations sur les employés.
- Donner le code des méthodes d'actions de EmployeeController

```
0 références
6 public class EmployeeController : Controller
7 {
8     // GET: EmployeeController
3 références
9     public ActionResult Index()
10     {
11         return View();
12     }
13
14     // GET: EmployeeController/Details/5
0 références
15     public ActionResult Details(int id)
16     {
17         return View();
18     }
19
20     // GET: EmployeeController/Create
0 références
21     public ActionResult Create()
22     {
23         return View();
24     }
25 }
```



# INJECTION DES DÉPENDANCES

- Inscription des services dans la classe Program.cs : est l'endroit où il faut enregistrer les dépendances de classes pour pouvoir utiliser des instances d'une classe dans d'autres classes.
- Ajouter ces **instructions** dans le code de la classe Program.cs de votre projet :

```
using WebApplicationEmployé.Models;  
using WebApplicationEmployé.Models.Repositories;
```

```
var builder = WebApplication.CreateBuilder(args);
```

```
// Add services to the container.  
builder.Services.AddControllersWithViews();
```

```
builder.Services.AddSingleton<IRepository<Employee>, EmployeeRepository>();
```



# INJECTION DES DÉPENDANCES

- EmployeeController dépend de IRepository pour la récupération des données des employés.
- Au lieu de créer une nouvelle instance de IRepository , nous injectons IRepository dans EmployeeController à l' aide du constructeur.
- C'est ce qu'on appelle l'**injection de constructeur** , car nous utilisons le constructeur pour injecter la dépendance.
- Ajoutons alors les instructions suivantes dans le code de notre contrôleur :

```
readonly IRepository<Employee> employeeRepository;
```

```
//injection de dépendance  
public EmployeeController(IRepository<Employee> empRepository)  
{  
    employeeRepository = empRepository;  
}
```



# CRÉER UN CONTRÔLEUR

- Compléter le code des méthodes d'actions de EmployeeController en se utilisant les instructions suivantes :

```
var employees = employeeRepository.GetAll();  
    return View(employees);
```

```
var employee = employeeRepository.FindByID(id);  
    return View(employee);
```

```
employeeRepository.Add(e);
```

```
employeeRepository.Update(id, newemployee);
```

```
employeeRepository.Delete(id);
```

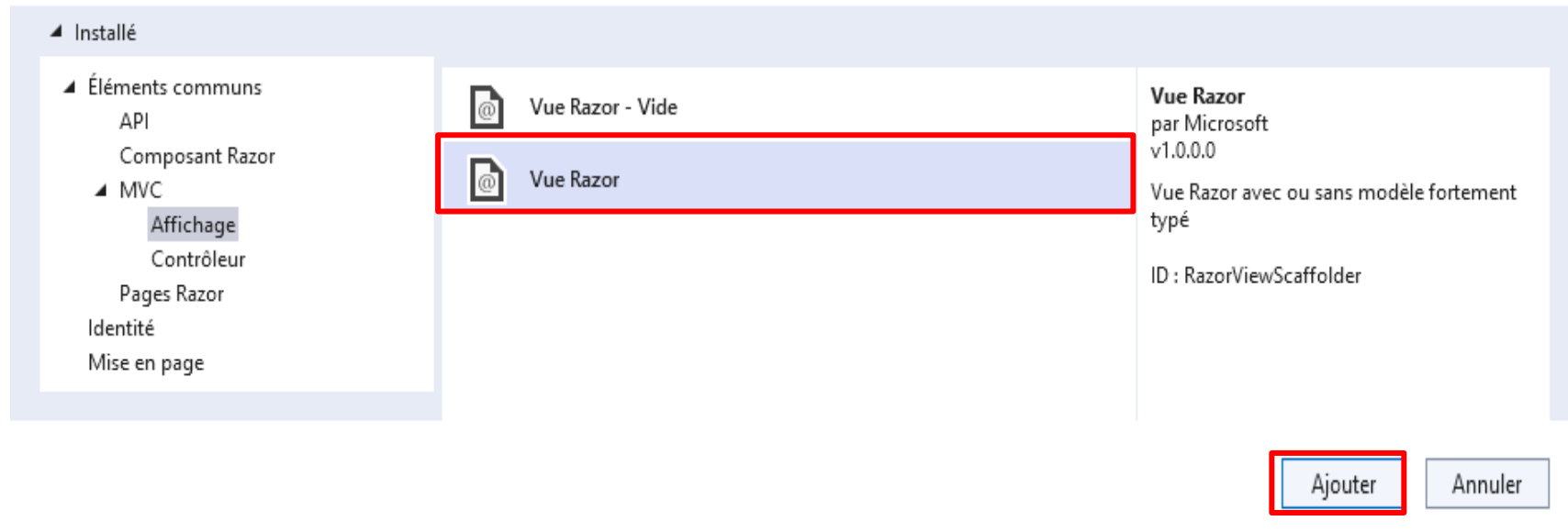


# CRÉER LES VUES

- Passons maintenant à la création des vues de notre application, et commençons par la vue de la méthode Index :

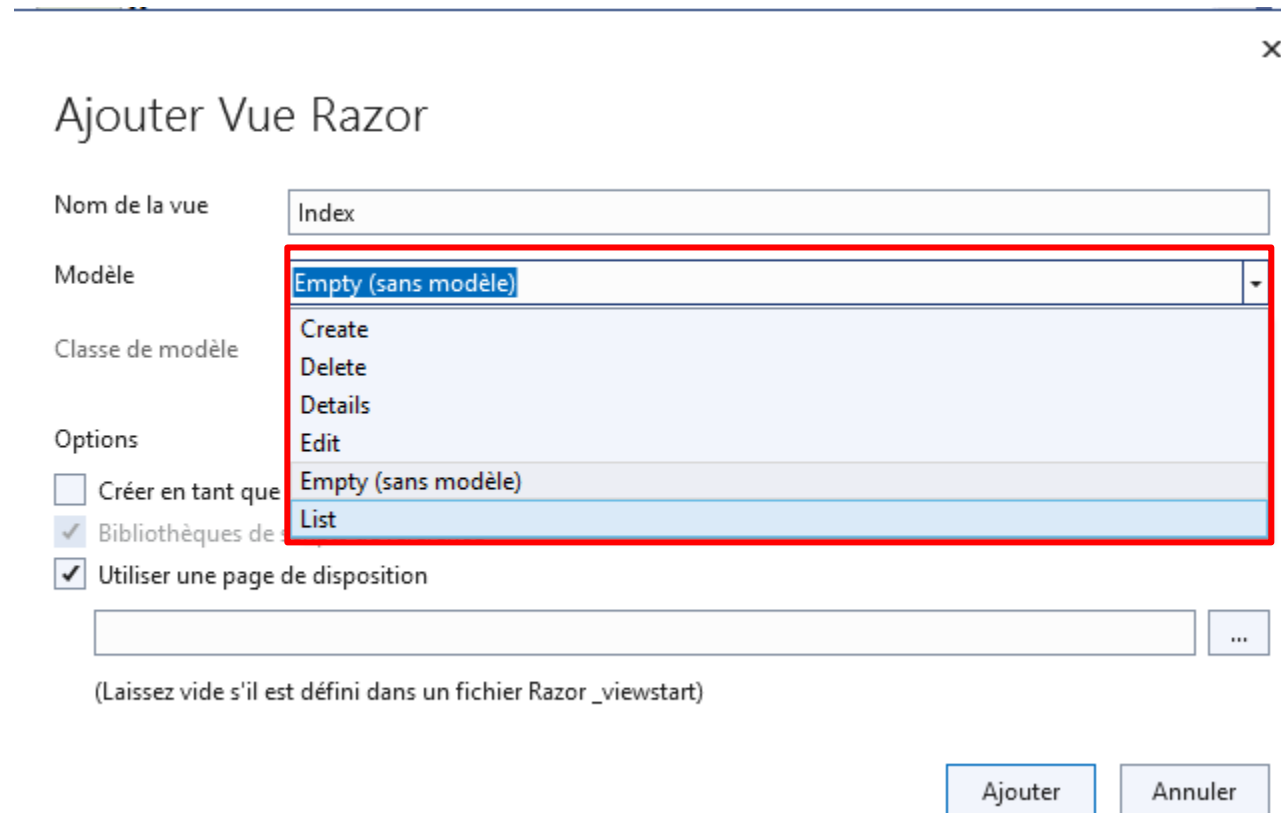
```
// GET: EmployeeController
3 références
public ActionResult Index()
{
    Ajouter une vue...
```

Ajouter un nouvel élément généré automatiquement



# CRÉER LES VUES

- Sélectionnez le modèle de scaffolding.
- La liste déroulante des modèles affiche les modèles par défaut disponibles pour les vues Créer, Supprimer, Détails, Modifier, Liste ou Vide.
- Sélectionnez le modèle "Liste" car nous voulons afficher la liste des employés dans la vue.



Ajouter Vue Razor

Nom de la vue: Index

Modèle: Empty (sans modèle)

Classe de modèle: Create, Delete, Details, Edit, Empty (sans modèle), List

Options:

- ☐ Créer en tant que
- ☒ Bibliothèques de
- ☒ Utiliser une page de disposition

(Laissez vide s'il est défini dans un fichier Razor \_viewstart)

Ajouter Annuler





# CRÉER LES VUES

- Sélectionnez Employee dans le dropdown.
- La liste déroulante des classes de modèle affiche automatiquement le nom de toutes les classes du dossier de modèle.

Ajouter Vue Razor

Nom de la vue

Index

Modèle

List

Classe de modèle

Employee (WebApplicationEmployé.Models)

EmployeeRepository (WebApplicationEmployé.Models.Repositories)

ErrorViewModel (WebApplicationEmployé.Models)

Program

Views\_Shared\_Error (AspNetCore)

Options

☐ Créer en tant que

☒ Bibliothèques de

☒ Utiliser une page de disposition

...

(Laissez vide s'il est défini dans un fichier Razor \_viewstart)

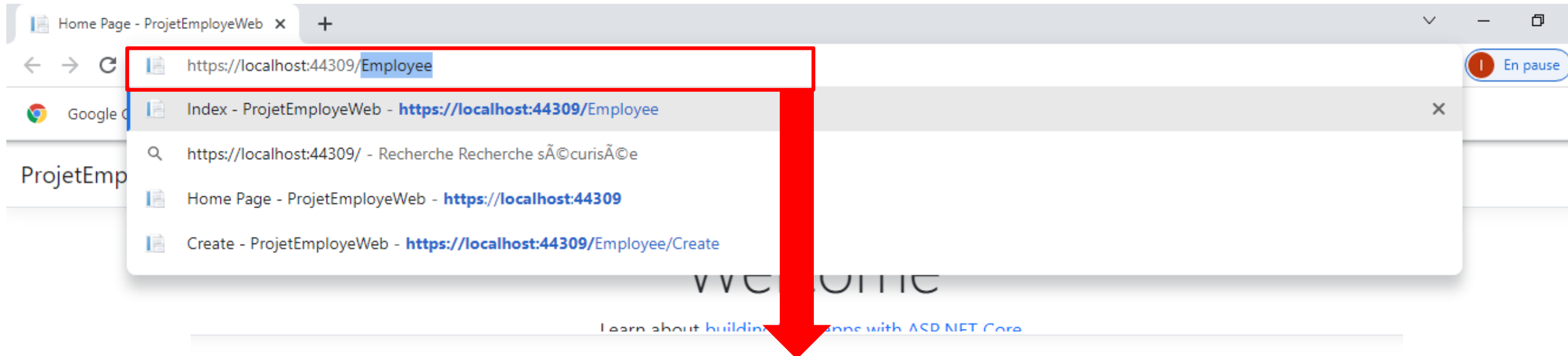
Ajouter

Annuler



# CRÉER LES VUES

- Pour visualiser la liste des employés, ajouter /Employee dans la barre d'adresse :



The screenshot shows a web browser with the address bar containing 'https://localhost:44309/Employee'. A dropdown menu is visible, showing suggestions for the current page and other pages. A red arrow points from the address bar to the 'Index' page below.

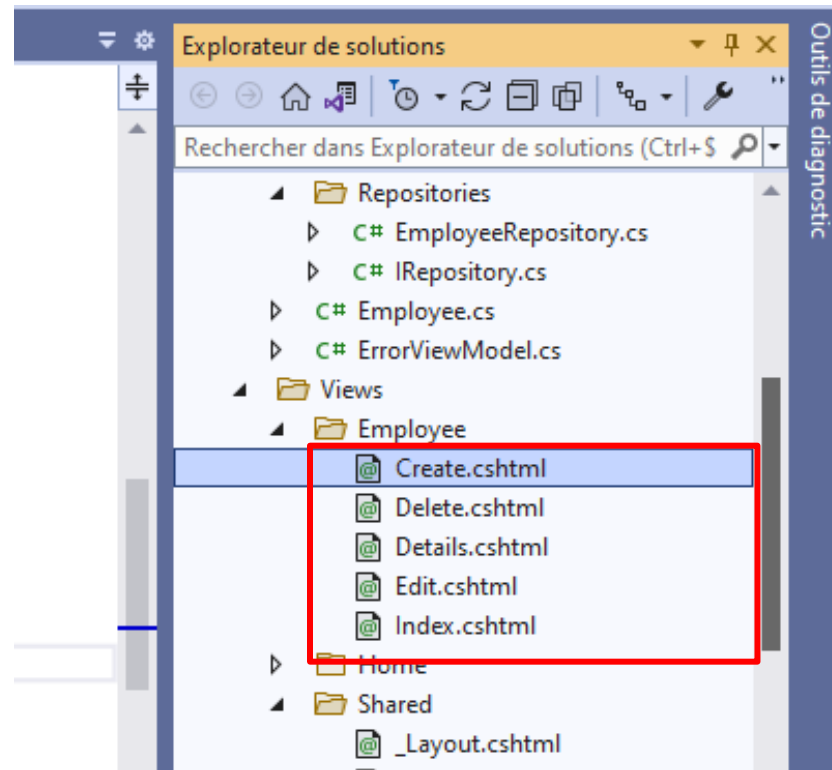
## Index

[Create New](#)

Id	Name	Departement	Salary	
1	Sofien ben ali	comptabilité	1000	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
2	Mourad triki	RH	1500	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
3	ali ben mohamed	informatique	1700	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
4	tarak aribi	informatique	1100	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

# CRÉER LES VUES

- De la même manière, créer les vues correspondantes aux actions : Create, Edit, Delete et Details :



# CRÉER LES VUES

- Pour pouvoir récupérer les informations d'un employé pour modification ou suppression ou encore affichage, modifier les instructions suivantes dans :

```
<td>  
</td>  
</tr>  
</tbody>  
</table>
```

```
@Html.ActionLink("Edit", "Edit", new { id=item.Id }) |  
@Html.ActionLink("Details", "Details", new { id=item.Id }) |  
@Html.ActionLink("Delete", "Delete", new { id=item.Id })
```

## Edit Employee

Id

Name

Departement

Salary

[Back to List](#)



# CRÉER LES VUES

- Pour pouvoir récupérer les informations d'un employé pour modification ou suppression ou encore affichage, ajouter l'Id de l'employé comme paramètre du lien `Html.ActionLink` dans le fichier « `Index.cshtml` »:

```
<td>  
</td>  
</tr>  
</tbody>  
</table>
```

```
@Html.ActionLink("Edit", "Edit", new { id=item.Id}) |  
@Html.ActionLink("Details", "Details", new { id=item.Id }) |  
@Html.ActionLink("Delete", "Delete", new { id=item.Id })
```

## Edit Employee

Id

Name

Departement

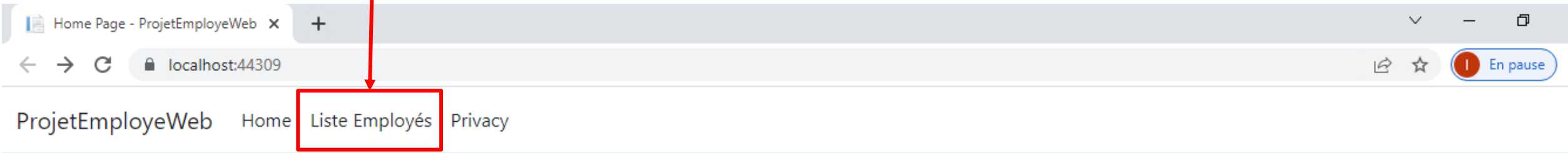
Salary

[Back to List](#)



# AJOUTER UN LIEN DANS LE MENU

- On va maintenant ajouter un lien dans la barre de navigation de notre application pour pouvoir accéder directement à la liste des employés :



- Pour cela, on va modifier le fichier « \_Layout.cshtml » du dossier « Shared » et ajouter les lignes suivantes :

```
<li class="nav-item">  
    <a class="nav-link text-dark" asp-area="" asp-controller="Employee" asp-action="Index">Liste Employés</a>  
</li>
```



# AJOUTER LES ANNOTATIONS

- Pour contrôler la saisie des champs de la classe Employee, on va ajouter les annotations suivantes :

```
using System.ComponentModel.DataAnnotations;
namespace ProjetEmployeWeb.Models
{
    33 références
    public class Employee
    {
        20 références
        public int Id { get; set; }

        [Required, StringLength(10, ErrorMessage = "Taille max 10 characters")]
        public string Name { get; set; }

        [Required]
        public string Departement { get; set; }

        [Range(200, 5000)]
        public int Salary { get; set; }
    }
}
```



# AJOUTER UNE ZONE DE RECHERCHE

- Pour pouvoir filtrer les employés selon leur Id, nous allons ajouter une zone de recherche en suivant les étapes suivantes :

- Dans l'interface "IRepository" ajouter la déclaration suivante :

```
List<T> Search(string term);
```

- Dans la classe "EmployeeRepository" ajouter la méthode suivante :

```
public List<Employee> Search(string term) {  
    if (!string.IsNullOrEmpty(term))  
        return employees.Where(a => a.Name.Contains(term)).ToList();  
    else  
        return employees;  
}
```

- Dans le contrôleur, ajouter la méthode suivante :

```
public ActionResult Search(string term){  
    var result = employeeRepository.Search(term);  
    return View("Index", result);  
}
```



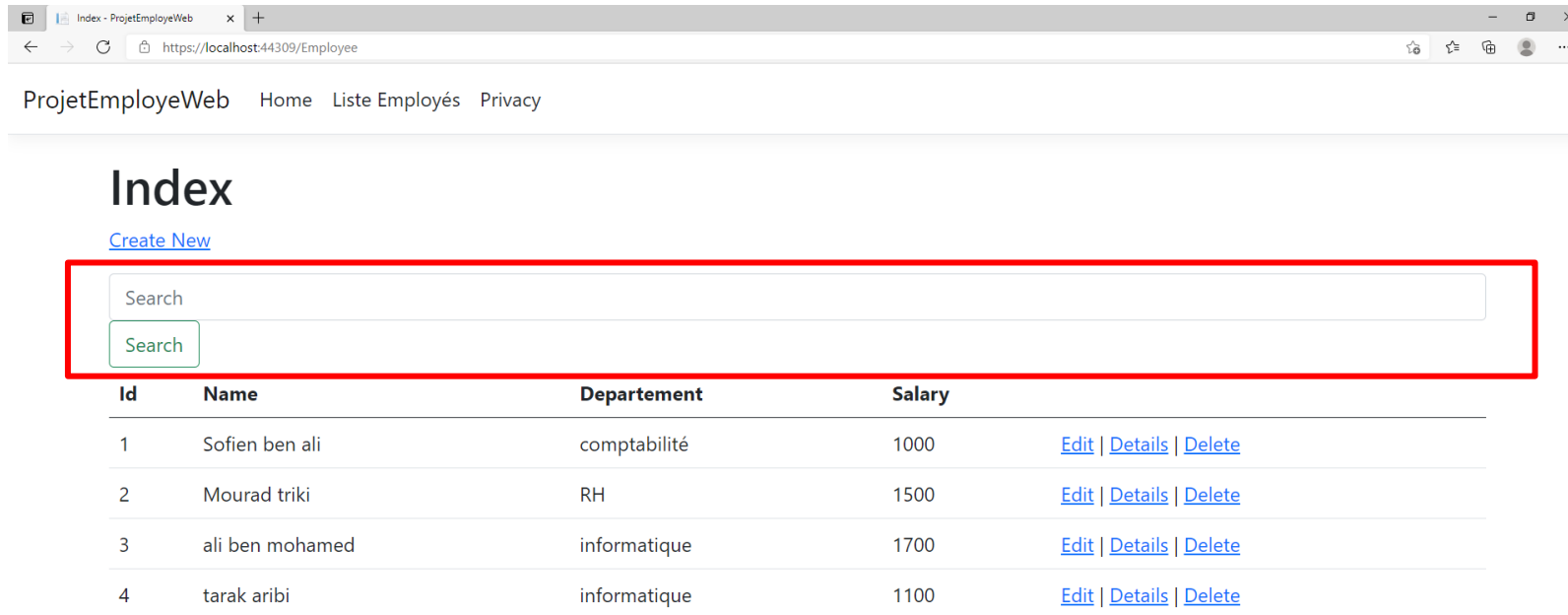


# AJOUTER UNE ZONE DE RECHERCHE

- Dans la page "Index.cshtml" créer une zone de recherche à l'aide de l'instruction suivante :

```
<form class="form-inline my-2 my-lg-0" asp-action="Search" asp-controller="Employee" >  
  <input class="form-control mr-sm-2" name="term" type="search" placeholder="Search"  
  aria-label="Search">  
  <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>  
</form>
```

- On obtient alors :



The screenshot shows a web browser window with the URL `https://localhost:44309/Employee`. The page title is "Index". Below the title is a link "Create New". A search bar is highlighted with a red box, containing a text input field with the placeholder "Search" and a "Search" button. Below the search bar is a table with 4 columns: "Id", "Name", "Departement", and "Salary". The table contains 4 rows of employee data. Each row has links for "Edit", "Details", and "Delete".

Id	Name	Departement	Salary	
1	Sofien ben ali	comptabilité	1000	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
2	Mourad triki	RH	1500	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
3	ali ben mohamed	informatique	1700	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
4	tarak aribi	informatique	1100	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>



# UTILISATION DE VIEWDATA POUR

- Dans notre application, nous avons besoin de récupérer les informations suivantes :
  - ✓ Le nombre d'employés du département « RH »;
  - ✓ Le salaire Max et
  - ✓ La moyenne des salaires
- Pour ce faire, on va suivre les étapes suivantes :

1. Dans l'interface "IRepository" nous allons ajouter les déclarations suivantes :

```
double SalaryAverage();  
double MaxSalary();  
int HrEmployeesCount();
```

2. Dans la classe "EmployeeRepository" nous allons ajouter les déclarations suivantes :

```
public double SalaryAverage(){  
    return lemployees.Average(x => x.Salary);  
}  
  
public double MaxSalary(){  
    return lemployees.Max(x => x.Salary);  
}  
  
public int HrEmployeesCount() {  
    return lemployees.Where(x => x.Département == "RH").Count();  
}
```



# UTILISATION DE VIEWDATA

3. Nous allons modifier l'ActionResult Index du contrôleur "EmployeeController" en ajoutant les instructions suivantes :

```
// GET: Employee
```

```
public ActionResult Index()
{
    var employees = employeeRepository.GetAll();
    ViewData["EmployeesCount"] = employees.Count();
    ViewData["SalaryAverage"] = employeeRepository.SalaryAverage();
    ViewData["MaxSalary"] = employeeRepository.MaxSalary();
    ViewData["HREmployeesCount"] = employeeRepository.HrEmployeesCount();
    return View(employees);
}
```



# UTILISATION DE VIEWDATA

4. Enfin, nous allons à la fin du fichier « Index.cshtml » en ajoutant les instructions suivantes :

```
<div>
    Nombre d'employés : @ViewData["EmployeesCount"]
</div>
<div>
    Salaire Moyen : @ViewData["SalaryAverage"]
</div>
<div>
    Salaire Max : @ViewData["MaxSalary"]
</div>
<div>
    Nombre d'employés du département HR : @ViewData["HREmployeesCount"]
</div>
<div>
    <a asp-action="Index">Afficher tous les employés</a>
</div>
```



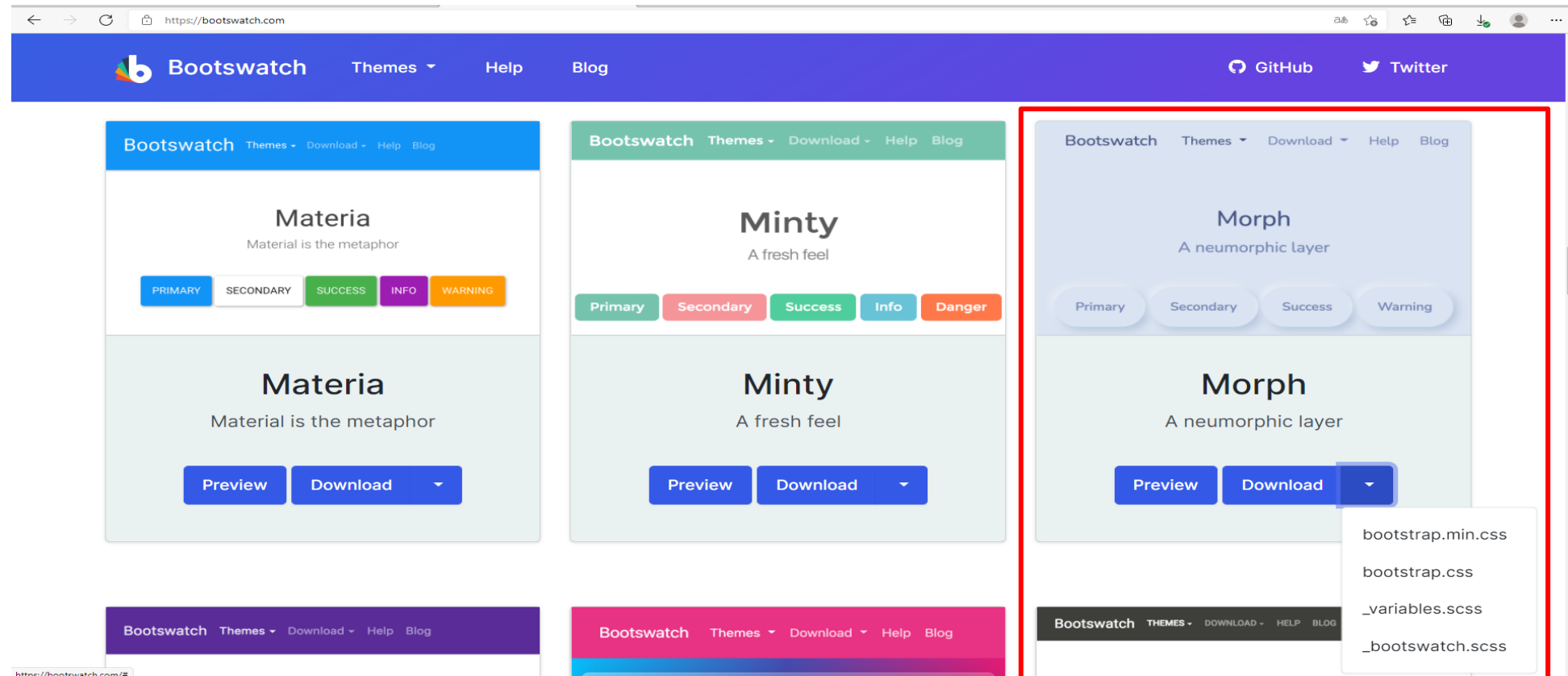
# MODIFIER LE THÈME DE L'APPLICATION

Dans le but de modifier le thème d'affichage des pages de notre application, nous allons suivre les étapes suivantes :

1. Ouvrir le lien suivant :

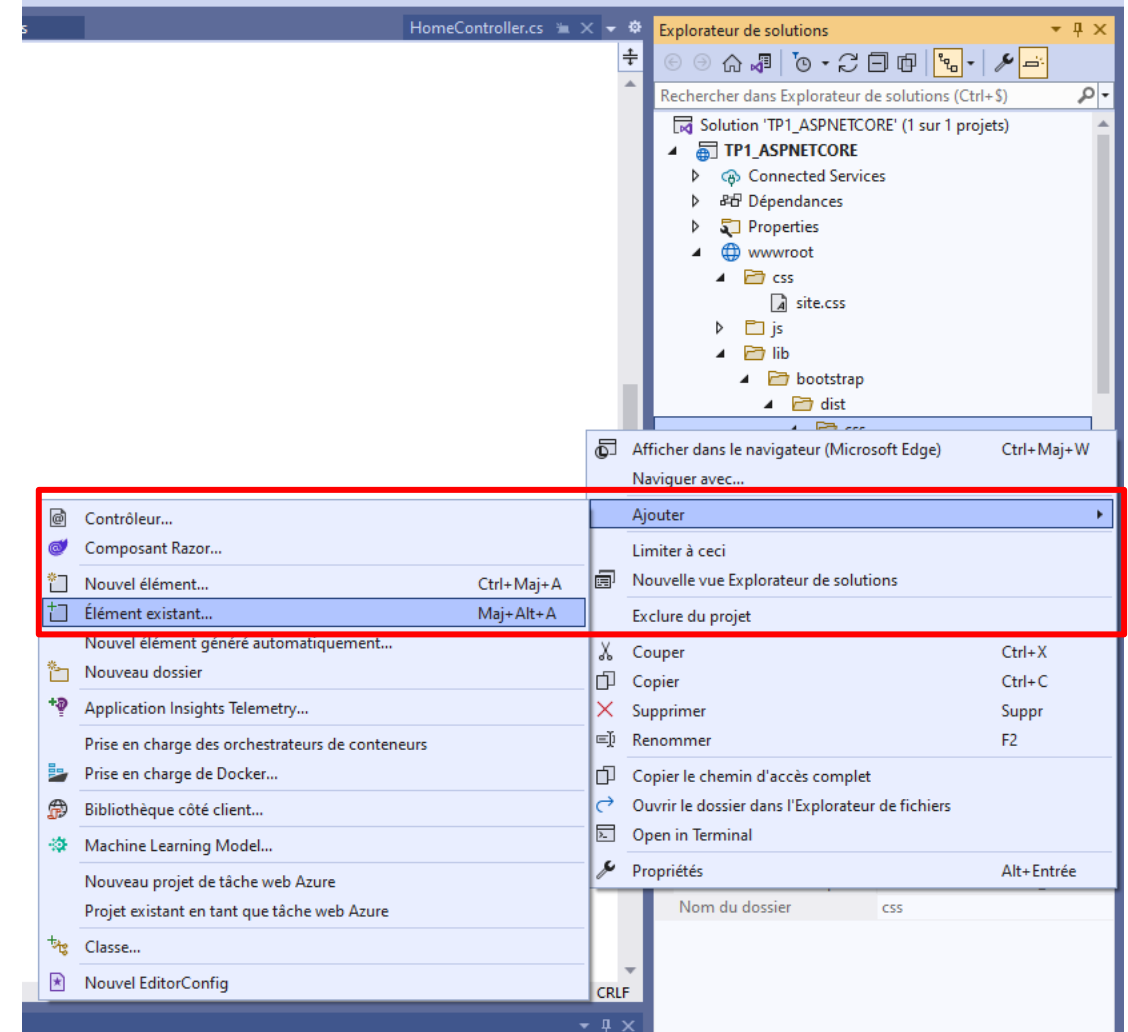
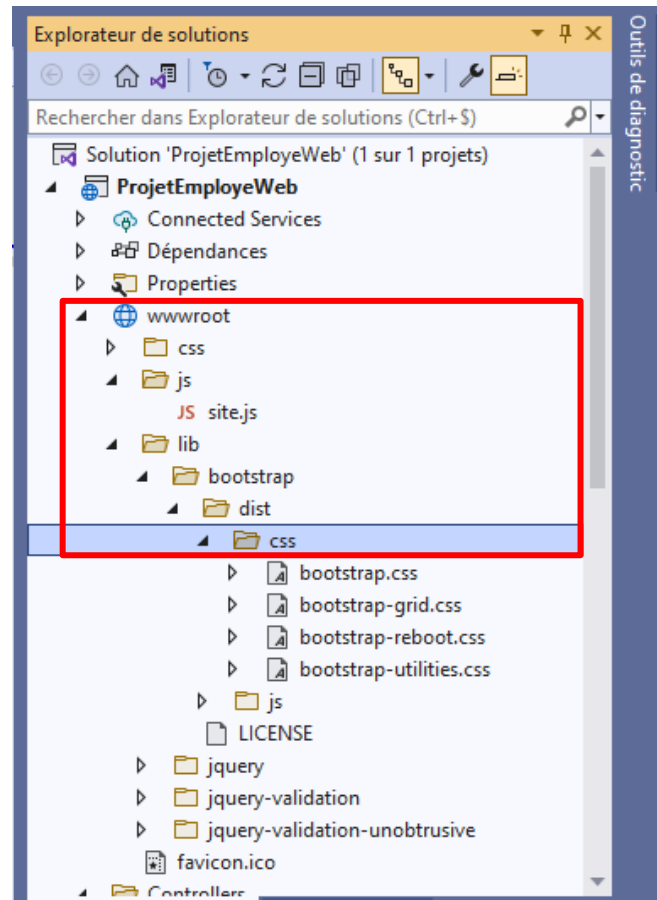
<http://bootswatch.com>

2. Choisissez le thème désiré puis cliquez sur le bouton download « bootstrap.css » en modifier le nom, dans notre exemple on va le nommer « bootstrap\_Morph.css »



# MODIFIER LE THÈME DE L'APPLICATION

3. Revenant à notre application, et ajouter dans le sous répertoire « css » de l'arborescence ci-dessus, le fichier téléchargé en choisissant l'option « Ajouter → Élément existant »:



# MODIFIER LE THÈME DE L'APPLICATION

4. Maintenant, modifier dans le fichier « \_Layout.cshtml », la ligne suivante :

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

En remplaçant le nom du fichier de « bootstrap.css » en « bootstrap\_Morph.css » dans notre exemple.

On obtient alors :

ProjetEmployeeWeb Home Liste Employés Privacy

## Index

[Create New](#)

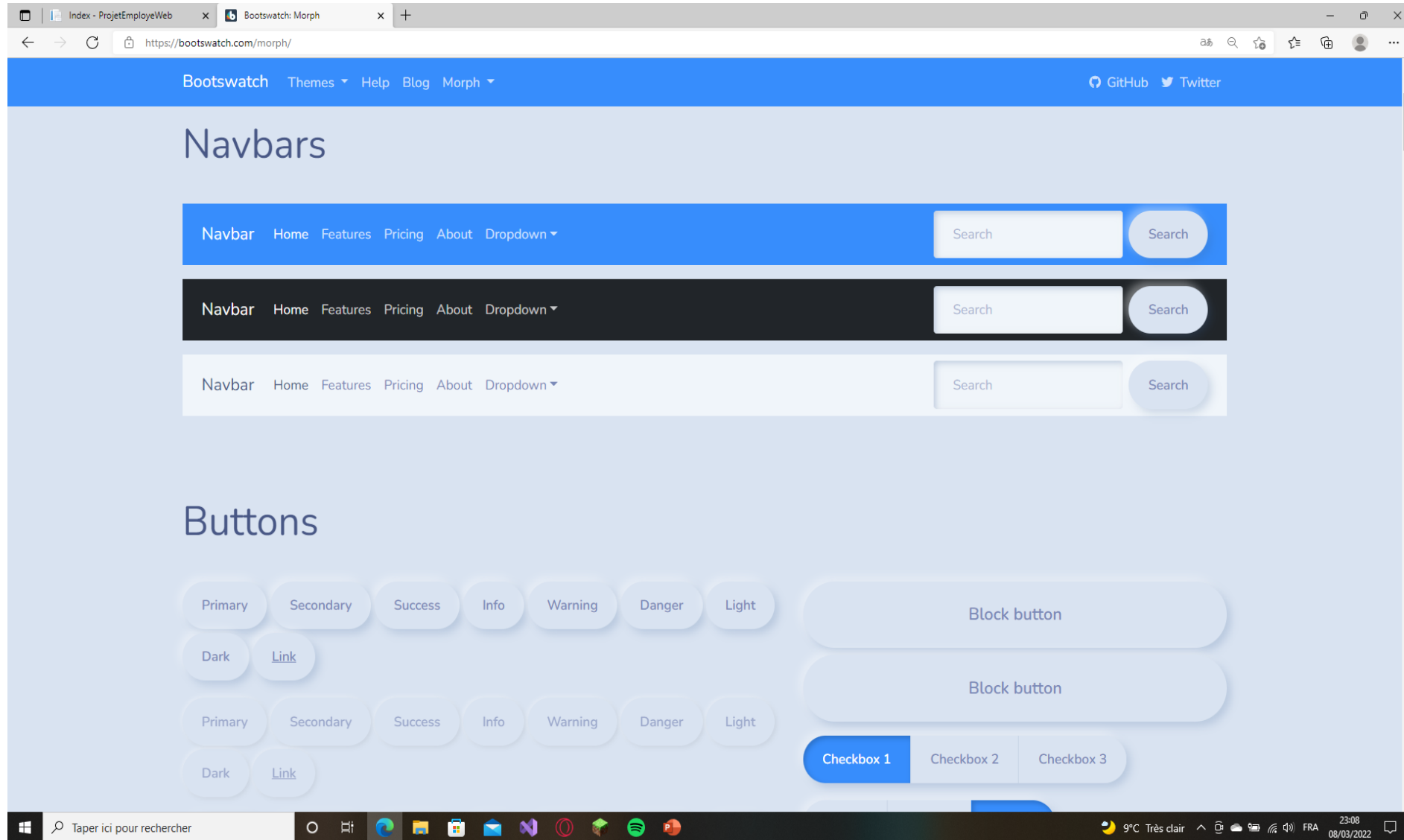
Search

Id	Name	Departement	Salary	
1	Sofien ben ali	comptabilité	1000	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
2	Mourad triki	RH	1500	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
3	ali ben mohamed	informatique	1700	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
4	tarak aribi	informatique	1100	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

Nombre d'employés : 4  
Salaire Moyen : 1325  
Salaire Max : 1700  
Nombre d'employés du département RH : 1  
[Afficher tous les employés](#)

# MODIFIER LA FORME DU NAVBAR

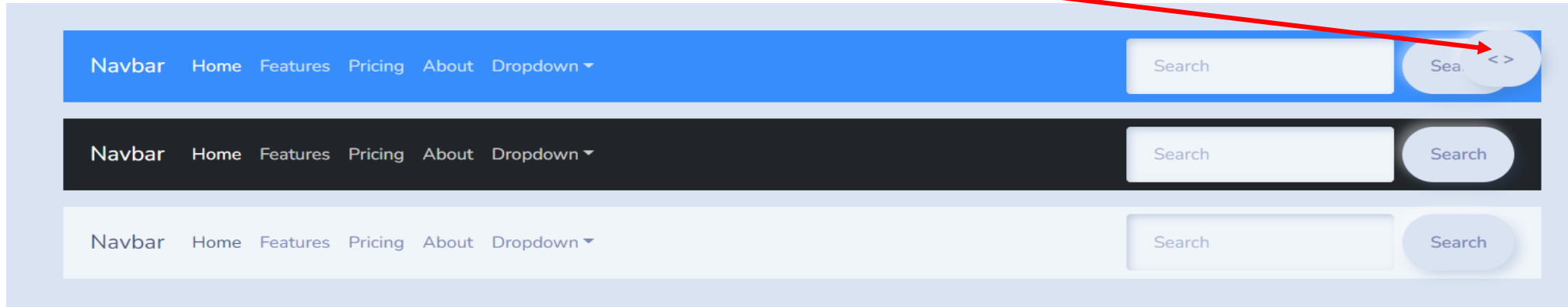
Revenons à la page [bootswatch.com](https://bootswatch.com) et cliquez sur le bouton « Preview » du thème téléchargé :





# MODIFIER LA FORME DU NAVBAR

En glissant la souris sur un des Navbars afficher ce bouton sera affiché :



En cliquant sur ce bouton une fenêtre de code source s'affiche :

- Sélectionnez ce code ;
- Dans le fichier « \_Layout.cshtml » placer ce code à la place du code de votre navbar

```
Source Code
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarColor01"
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarColor01">
      <ul class="navbar-nav me-auto">
        <li class="nav-item">
          <a class="nav-link active" href="#">Home
            <span class="visually-hidden">(current)</span>
          </a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pricing</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">About</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#">Action</a>
          <div class="dropdown-menu">
            <a class="dropdown-item" href="#">Action</a>
            <a class="dropdown-item" href="#">Another action</a>
          </div>
        </li>
      </ul>
    </div>
  </div>
</nav>
```