ASP.NET MVC

Mme. Fatima-Ezzahra AIT BENNACER
f.aitbennacer@emsi.ma

2025 - 2026

Plan du cours

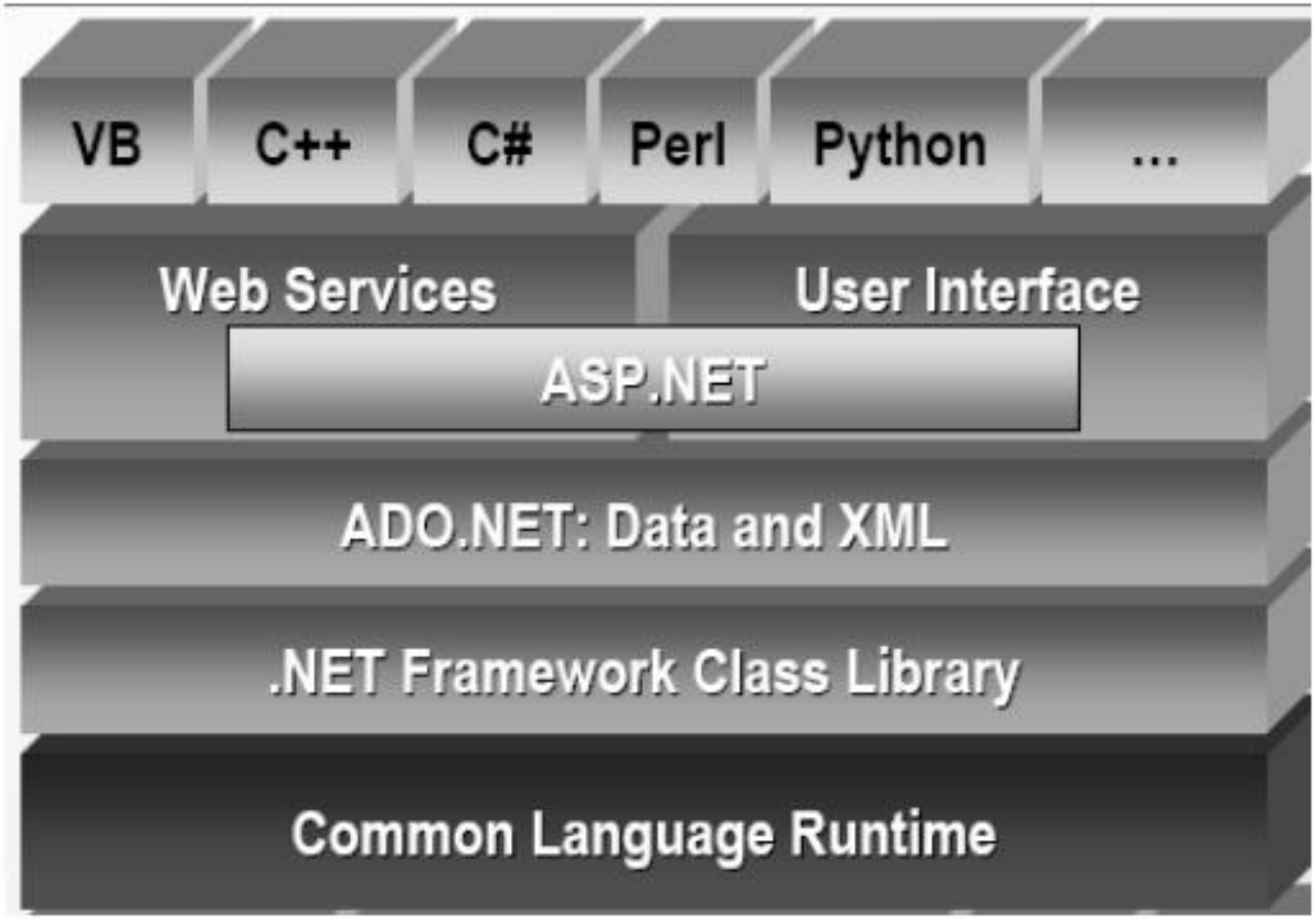
Introduction à .Net

C'est quoi .NET ?

- Un standard proposé par Microsoft en 2002.
- Une plateforme de développement, gratuite et open source qui supporte un grand nombre de langages de programmation.
- Un cadre de travail formé d'outils, de modules actifs au runtime et de classes formant une API très étendue.
- Un environnement d'exécution sécurisé.

Architecture .Net

Architecture .NET



Architecture .NET

- **Le CLR (Common language Runtime)** : il représente la machine virtuelle de la plate-forme (un peu comme la JVM pour Java). Il est responsable de l'exécution des applications et gère tous les aspects de sécurité

- **Un ensemble de librairie de classes** : située au-dessus de la CLR, cette rubrique offre une couche pour la gestion des données. On y retrouve les **Windows Forms (WinForms)**, ensemble de classes permettant la conception d'IHM pour Windows (un peu comme AWT ou Swing).

Architecture .NET

- **AdoNet (Data and XML)** : une nouvelle génération de composants d'accès aux bases de données.
- **ASP.NET**, qui fournit un ensemble de classes pour la conception de sites dynamiques, la création d'IHM pour le web, les WebForms et la conception de services web.
- **Au sommet de la pile**, nous avons les langages supportés par .NET tels que **C#, VB.NET, C++, F#**, qui sont des produits de Microsoft, et d'autres tels que **Cobol, Delphi**, etc.

Les bases .Net

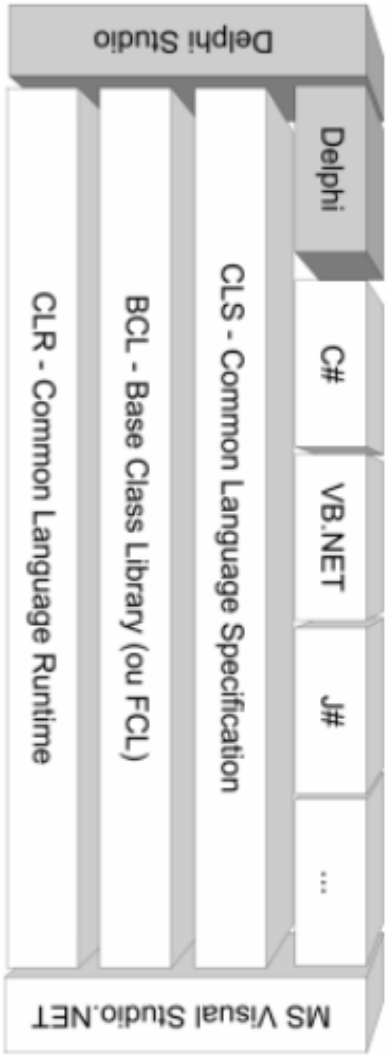
Les bases .NET

- **Le CLR**

  Ce runtime est le moteur de .NET, c'est lui qui est en charge de l'exécution des logiciels écrits pour .NET compilés en CIL (Common Intermediate Language) avec un système appelé JIT (Just in Time).

  Le CLR gère:

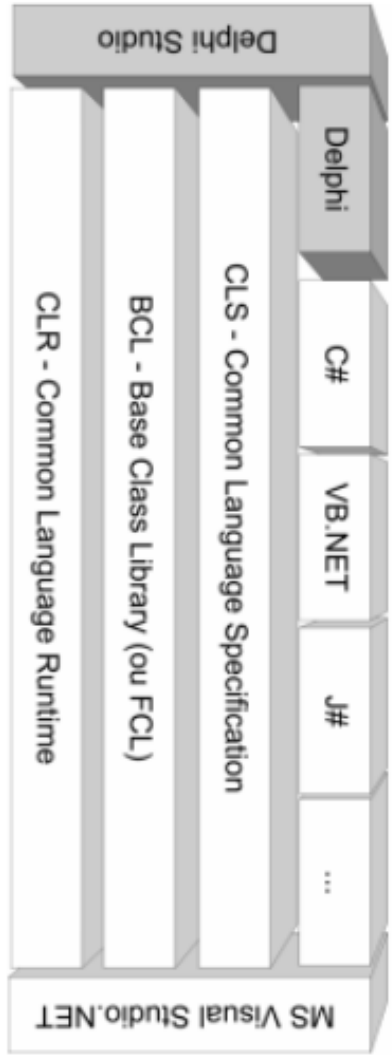
  - La compilation en code natif et l'exécution de code IL (Intermediate Language).
  - La gestion de la sécurité.
  - La gestion de la mémoire.
  - La gestion des processus.
  - La gestion des threads (tâches).



Les bases .NET

- **La BCL (Base Class Library) ou FCL (Framework Class Library**

  - Se plaçant directement au dessus du CLR
  - Offre de nombreuses classes, interfaces et types qui forment le socle des développements sous .NET
  - Permet d'unifier les développement puisqu'elle fournit l'ensemble des outils de base aux applications

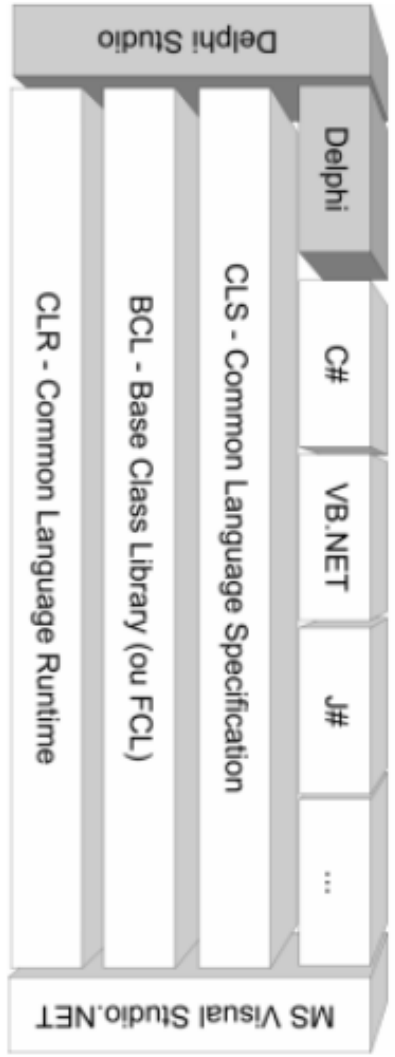


Les bases .NET

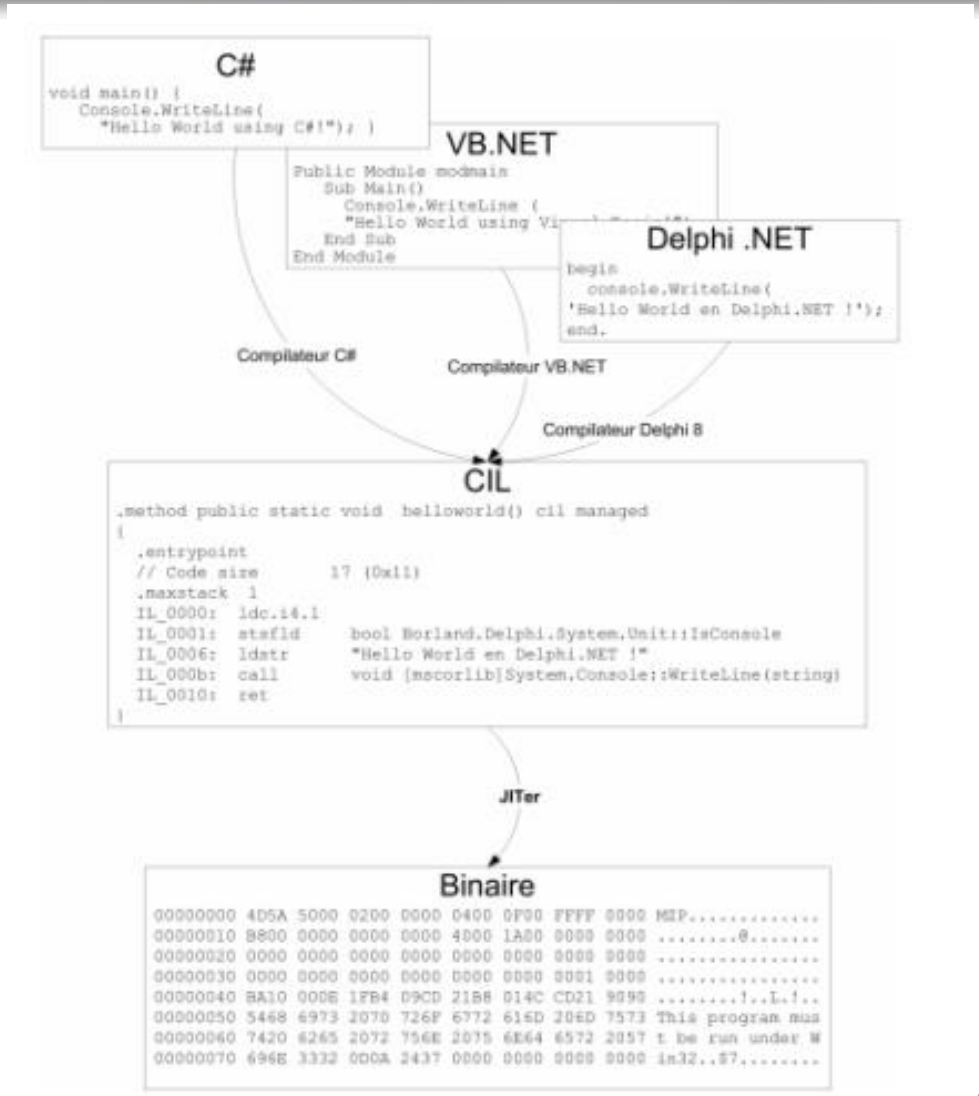
- **Les espaces de nom**

  - Se plaçant directement au dessus du CLR Une sorte de « boîte » dans laquelle des classes ayant (plus ou moins) rapport entre elles sont regroupées. Par exemple: System.Data.SqlClient, System.Web

  - Une Assembly (assemblée en français) est un fichier DLL dans lequel les classes sont stockées.



Du code source au binaire exécuté



Asp .Net

C'est quoi ASP.NET

- Plateforme de développement d'applications web sous Windows
- Utilise, par défaut, le serveur Web de référence de Microsoft: IIS (pour Internet Information Services)
- Repose sur le .NET Framework
- Plateforme Web unifiée fournissant les services nécessaires à la création des applications
- Ne dépend ni du langage de programmation ni du navigateur

Modèle MVC

MVC

- Introduit par Trygve Reenskaug en 1978
- Permettant de bien organiser le code source
- Conçu, initialement, pour les applications client-lourd et ensuite généralisé aux applications client-léger (Web)
- Une approche consistant à séparer l'affichage des informations, les actions de l'utilisateur et l'accès aux données
- Chacun de ces composants est construit pour manipuler un aspect particulier de développement de l'application
- Indispensable pour des applications dynamiques et de taille importante

MVC: Les trois composants

1. Modèle: La partie qui concerne les données et l'état de notre application. Pour le cas d'une base de données, on peut utiliser un ORM (Object-Relational Mapping) comme Entity Framework pour ASP.NET MVC

2. Vue: La partie qui concerne l'affichage : l'interface avec laquelle l'utilisateur interagit (HTML + CSS...)

3. Contrôleur: c'est l'intermédiaire entre le modèle et la vue. Il reçoit la requête de l'utilisateur. Il demande les données au modèle, les analyse et renvoie le résultat à afficher à la vue

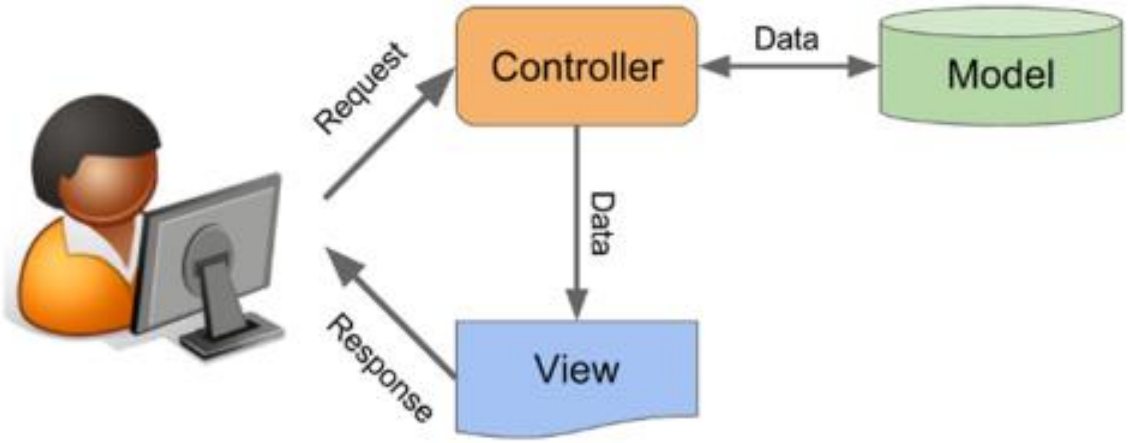
ASP.NET MVC

- Créé en 2007 par Scott Guthrie et intégré dans ASP.NET depuis 2009
- Framework de développement d'applications web selon le design pattern: MVC

- Par rapport à ASP.NET:
  - ASP.NET MVC permet de structurer d'avantage l'application, en créant des composants avec des rôles bien identifiés
  - Indispensable pour des applications dynamiques et de taille importante

ASP.NET MVC: Déroulement

1. Le client envoie une requête depuis la Vue
2. Le Contrôleur intercepte et analyse la requête du client
3. Le Contrôleur détermine quelle partie du Modèle est concernée afin d'effectuer les traitements nécessaires
4. Le Modèle s'occupe de l'interaction avec les données, applique les règles métier et renvoie les données au Contrôleur
5. Le Contrôleur sélectionne la Vue correspondante et lui injecte les données
6. La Vue affiche les données au client



ASP.NET MVC: Déroulement

- MVC est un patron de conception particulièrement adapté pour réaliser une application web

- MVC est l'acronyme de Modèle-Vue-Contrôleur :
  1. Le Modèle contient les données de l'application
  2. La Vue contient le code pour afficher les pages de l'application
  3. Le Contrôleur gère les interactions de l'utilisateur en faisant le lien avec le Modèle et la Vue

- MVC permet une séparation claire des intentions et optimise la création d'une application web ou d'un site, sa maintenance et ses tests automatisés.

ASP.NET vs ASP.NET Core

ASP.NET

- Pour Windows.
- Utilise le runtime .NET Framework.
- Bonnes performances.
- Supporte C#, F# et VB.
- Disponible sous Visual Studio.

ASP.NET Core

- Pour Windows, Mac et Linux.
- Utilise le runtime .NET Core.
- Plus performant que ASP.NET.
- Supporte C# et F#.
- Disponible sous VS, VSC.

Exemple d'applications web, IDE pour ASP.NET MVC

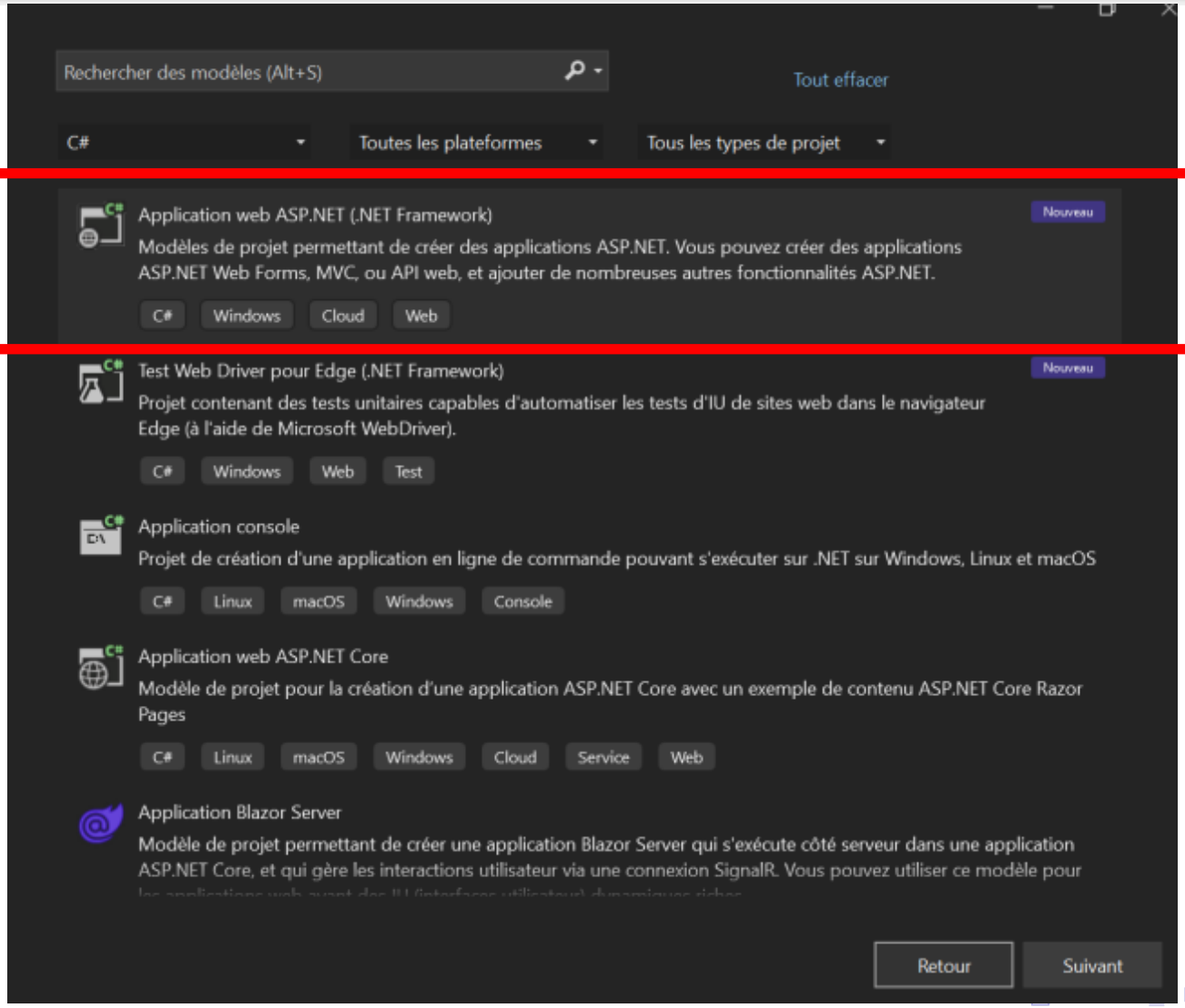
IDE pour ASP.NET MVC

- IDE pour ASP.NET MVC:
  - Web Matrix (pour les petites applications)
  - MonoDevelop (pour Windows et Mac)
  - Visual Studio

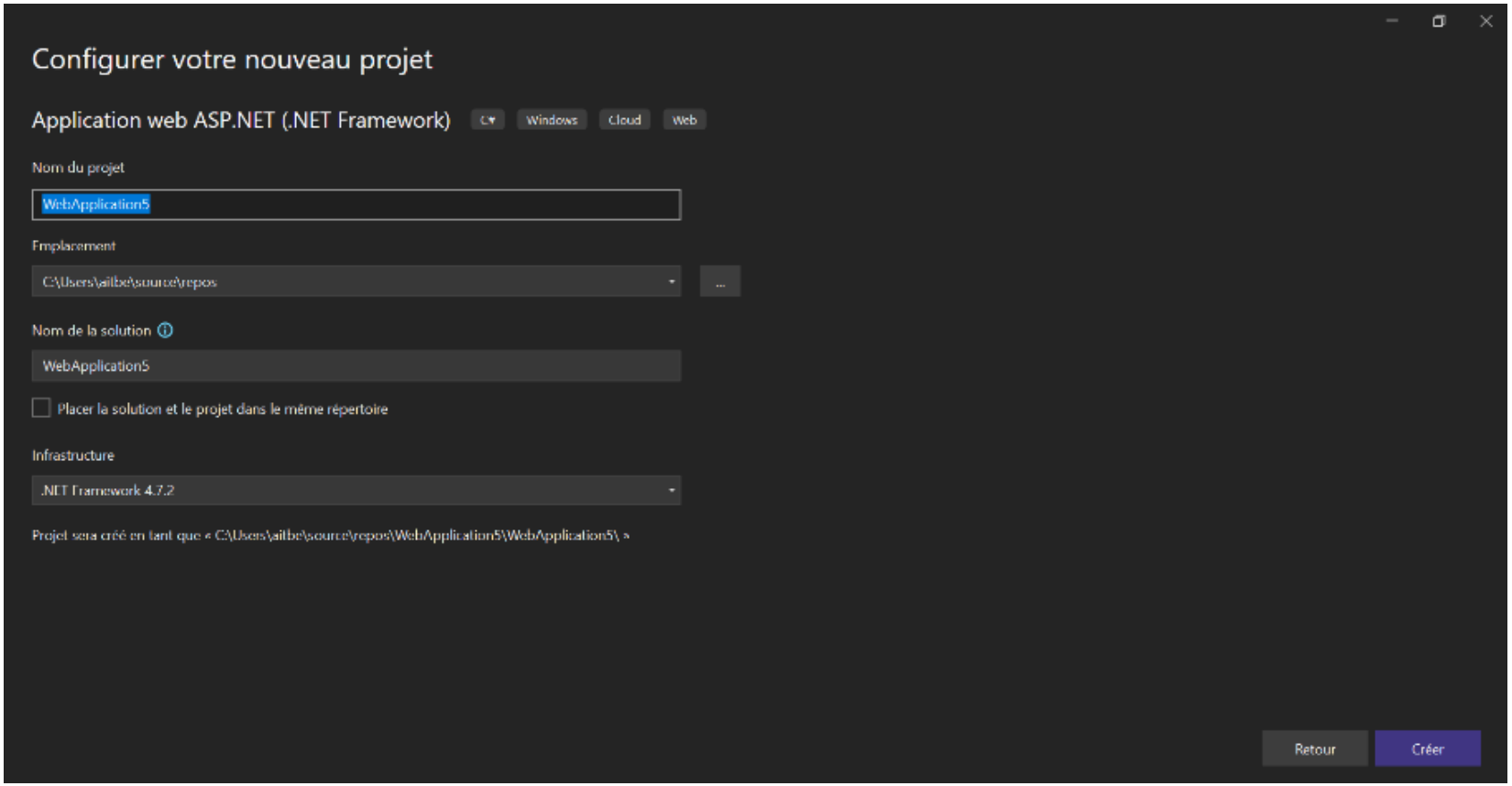
- Installation du .Net Framework: https://dotnet.microsoft.com/en-us/download/dotnet-framework/

  Choisissez developer pack offline installer

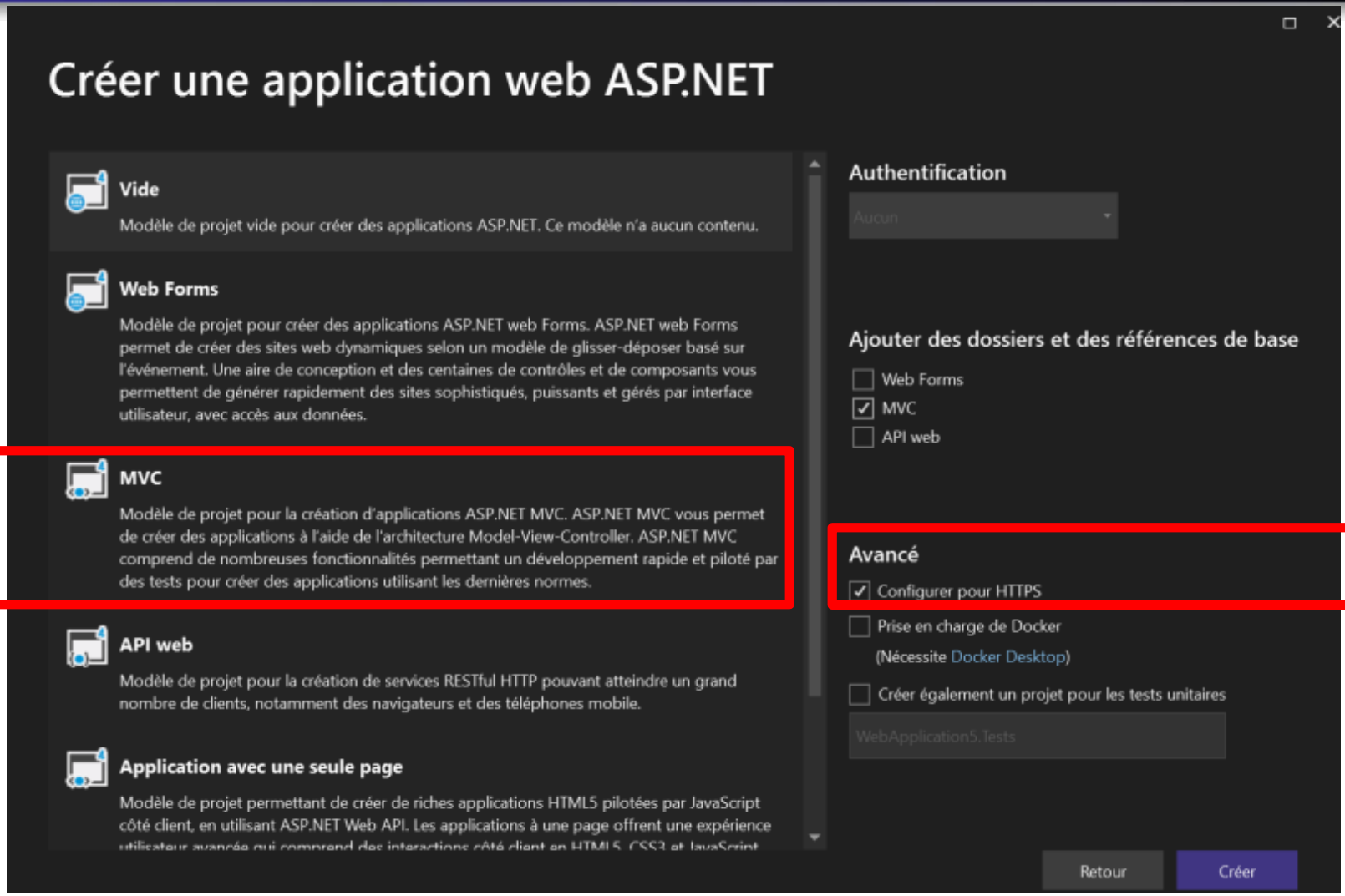
Création d'une page web avec ASP.NET



Création d'une page web avec ASP.NET



Création d'une page web avec ASP.NET



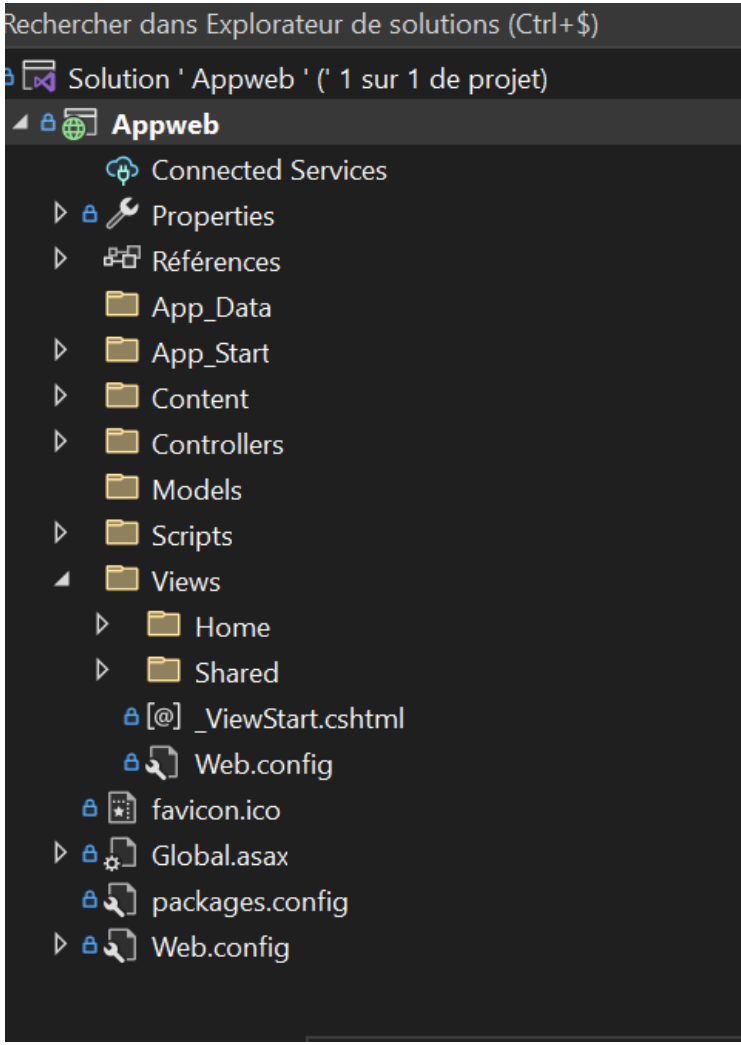
Création d'une page web avec ASP.NET

- **Models:** où on va placer les classes qui vont interagir avec les données.
- **Views:** où on va placer les pages (HTML/CSS, etc) qui serviront de réponse à la requête utilisateur.
- **Controllers:** où on va mettre les classes qui contiendront les actions à exécuter des réception d'une requête.

packages.config: fichier de configuration des packages NuGet.

App Start: où on va placer les fichiers de configuration. Il contient RouteConfig.cs qui servira de mapping route/controleur/action.

Global.asax: point d'entrée vers l'application. Il fait appel à RouteConfig présent dans App Start.

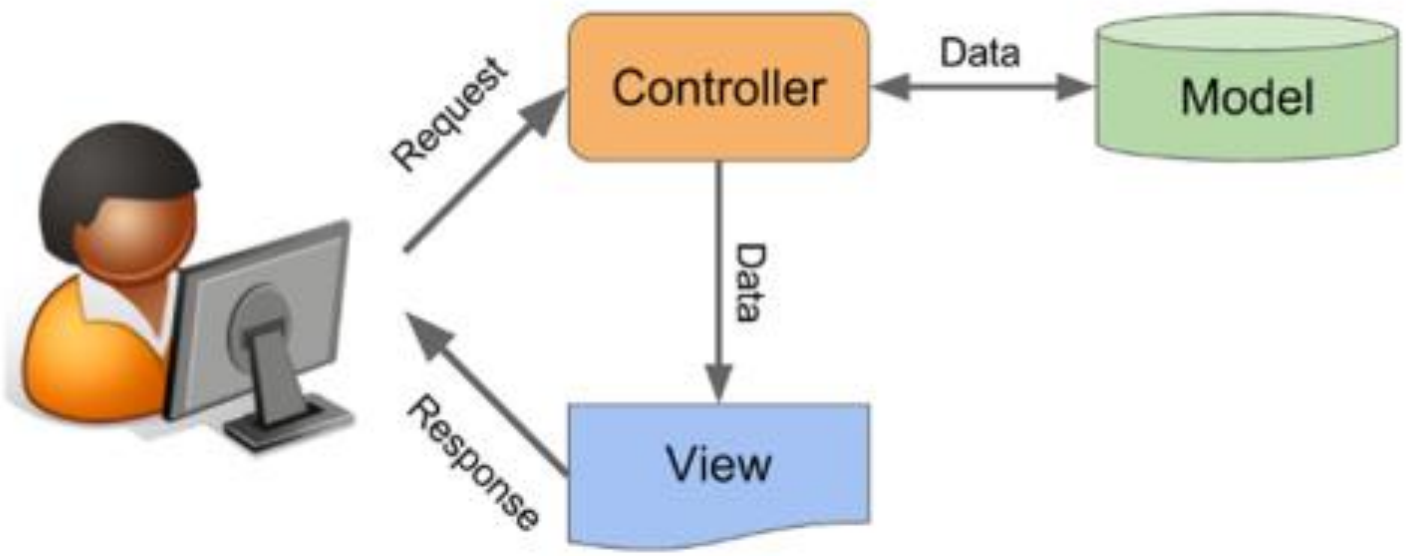


Création d'une page web avec ASP.NET

Controllers/ => Logique métier, actions (C#)
Models/ => Données, objets, classes (C#)
Views/ => Pages HTML dynamiques (.cshtml)
Shared/ => Layout général (_Layout.cshtml)
Home/ => Pages spécifiques au contrôleur Home
App_Start/ => Fichiers de config (RouteConfig.cs)
Global.asax => Point de départ de l'app, appelle RouteConfig
packages.config => Packages NuGet installés
Web.config => Config de l'application (base de données, etc.)

TP N°3:
Création d'une page web avec ASP.NET: Contrôleur

Rappel



Création d'une page web avec ASP.NET: MVC

Création du Contrôleur

- Faire un clic droit sur **Controllers** et aller dans **Ajouter > Controleur**.
- Sélectionner **MVC 5 Controller - Empty** et cliquer sur **Ajouter**.
- Saisir un nom pour le controleur (par exemple : **EtudiantController**) et cliquer sur **Ajouter**.
- Une classe EtudiantController qui hérite de Controller (du namespace Web.Mvc) est générée.

Création du Contrôleur

```
public class EtudiantsController : Controller
{
    public IActionResult Index()
    {
        // Logique métier
        return View();
    }
}
```

- **Controller**: Classe C# héritant de Controller, qui pilote la logique
- **IActionResult:** Type de retour standard d'une action dans MVC
- **Index()** : Méthode appelée par défaut (si aucune action précisée dans l'URL)
- return **View()**: Charge une vue (.cshtml) liée à cette action

Création du Modèle

```
public class Etudiant
    {
        public int Id { get; set; }
        public string Nom { get; set; }
        /*
        */
    }
```

- Un modèle est une classe qui représente une entité métier.
- Utilisée pour transporter des données entre le contrôleur et la vue.

Création d'une Vue

- Faire un clic droit sur la méthode **Index** et cliquer sur **Ajouter** une vue.
- Sélectionner **Vue MVC 5** et cliquer sur **Ajouter.**
- Sélectionner : Empty (sans modèle) pour le modèle et cliquer sur Ajouter.

Création d'une Vue

```
@model List<ApplicationWeb.Models.Etudiant>
<h2>Liste des étudiants</h2>
<table border="1" cellpadding="5">
    <thead>
<tr><th>Nom</th><th>Prénom</th><th>Âge</th><th>Détail</th></tr>
    </thead>
    <tbody>
        @foreach (var e in Model)
        { <tr>
                <td>@e.Nom</td>
                <td>@e.Prenom</td>
                <td>@e.Age</td>
          </tr>
        }
    </tbody>
</table>
```

Élément Razor	Rôle
@model	Indique le type de données reçu
@Model	Accède aux données dans la vue
@foreach (...)	Boucle C# dans la vue
@e.Propriété	Affiche les données du modèle

Création d'une Vue

Pour Transfert des données: Vue :

- **ViewBag, ViewData:** permet de stocker des données dans une action de contrôleur qui sera utilisée dans la vue correspondante.

Création d'une Vue

Pour transmettre une valeur simple :

```
public ActionResult Index(int? id) {
    ViewData["id"] = id;
    return View();
   }
```

Modifier Index.cshtml

```
<html> <head>
<title> Hello World </title> </head>
<body>
<p> <b> Hello @ViewData["id"] </b></p>
</body> </html>
```

Le routage en ASP.NET

- Le **routage** est un mécanisme qui détermine comment une **URL entrante** est **mappée à un contrôleur et une action**.

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

Élément	Signification
{controller}/{action}/{id?}	Structure des URL MVC
{id?}	Paramètre facultatif pour passer un ID

Le routage en ASP.NET

À noter: sur ASP.net -- RouteConfig

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional } );
```

Le routage en ASP.NET

Les routes par défaut

Une URL est composée de:

- **controller:** mot-clé obligatoire permettant de préciser le nom du contrôleur à exécuter.
- **action:** mot-clé obligatoire permettant de préciser le nom de la méthode à exécuter.
- **id:** correspond aux paramètres de la méthode.

TP N° 4