

INITIATION AU .NET



Delivering Transformation. Together.

sopra  steria

ME, MYSELF & I : NICOLAS FLEURY



- Architecte .NET
- MIAGE Promo 2007
- 11 ans de .NET
 - Dev
 - Lead Dev
 - CP
 - Archi
- 4 ans chez Sopra Steria
 - Membre de la cellule AET



AGENDA

- Introduction au .NET
- Pause
- Focus sur Windows Foundation Communication
- Focus sur ASP.NET Core Web API



INITIATION AU .NET

Introduction au Framework Microsoft.NET



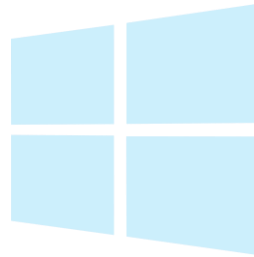
UNIVERSITE D'ORLEANS



Delivering Transformation. Together.

QU'EST-CE QUE LE .NET

- Ca vous fait penser à quoi?
- Historiquement
 - C'est la réponse de Microsoft à Java et à J2E
 - Multi langage, mono plateforme
 - Framework fermé et propriétaire
- **Mais ça, c'était avant...**

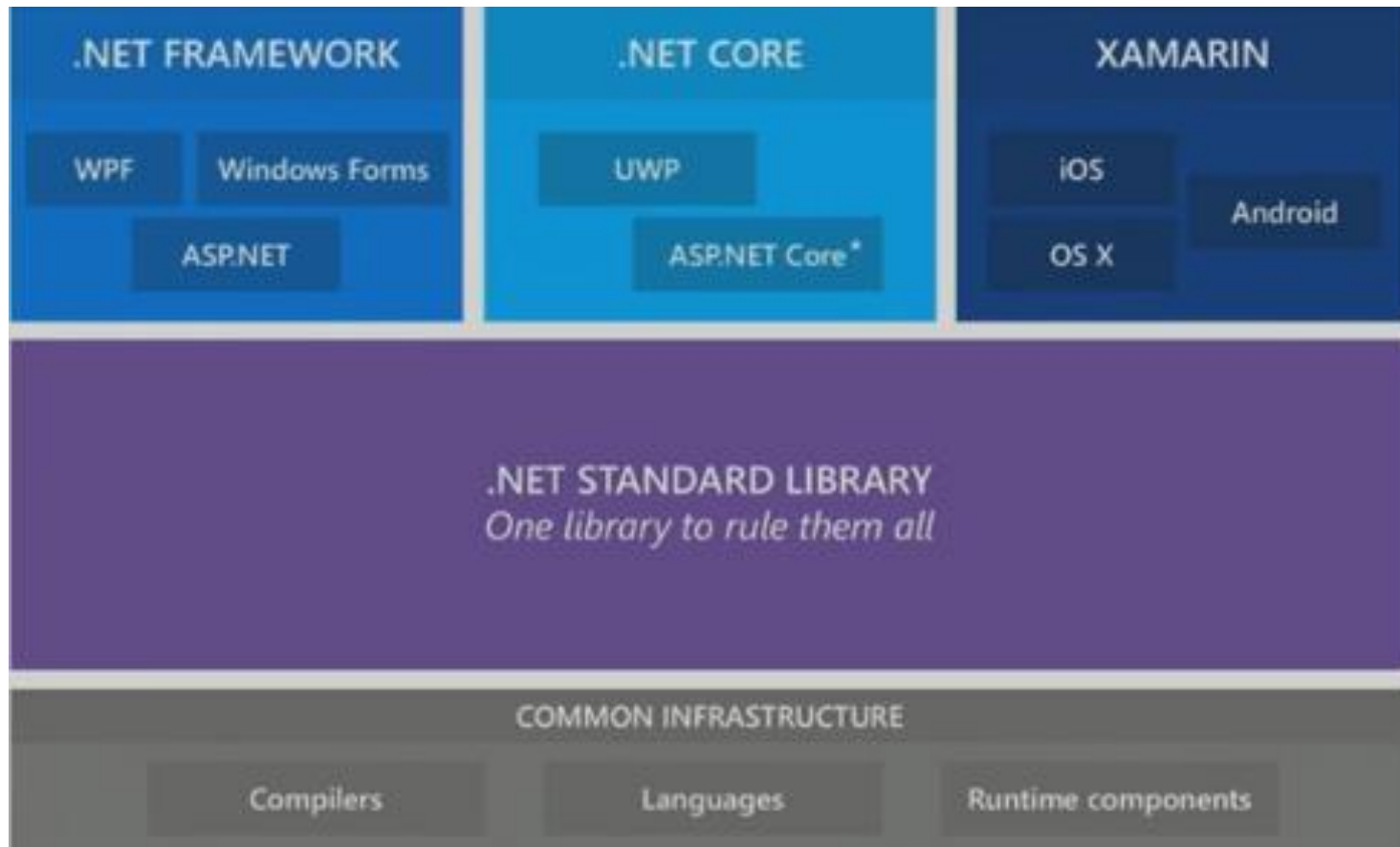


.NET



QU'EST-CE QUE LE .NET

- Aujourd'hui : plusieurs implémentations



QU'EST-CE QUE LE .NET

- .NET Standard

- Spécification officielle des API .NET
 - APIs destinées à être disponibles sur toutes les implémentations de .NET
 - Objectif : Etablir une meilleure uniformité dans l'écosystème .NET
- Framework socle pour les implémentations communes:
 - Si votre code cible une version de .NET Standard, il peut s'exécuter sur n'importe quelle implémentation de .NET qui prend en charge cette version de .NET Standard.



IMPLÉMENTATIONS DE .NET

- .NET Framework

- Implémentation d'origine du .NET
- Existe depuis 2002



- Implémente .NET Standard depuis la version 4.5
- Contient des API spécifiques à Windows



IMPLÉMENTATIONS DE .NET

- .NET

- M

- O

-



[Github.com/aspnet](https://github.com/aspnet)

[Github.com/dotnet](https://github.com/dotnet)



IMPLÉMENTATIONS DE .NET

- .NET Core



Totalement modulaire



Cloud-ready



Open Source
avec contributions



Productivité pour
les développeurs



Indépendant de l'éditeur



Cross-Platform

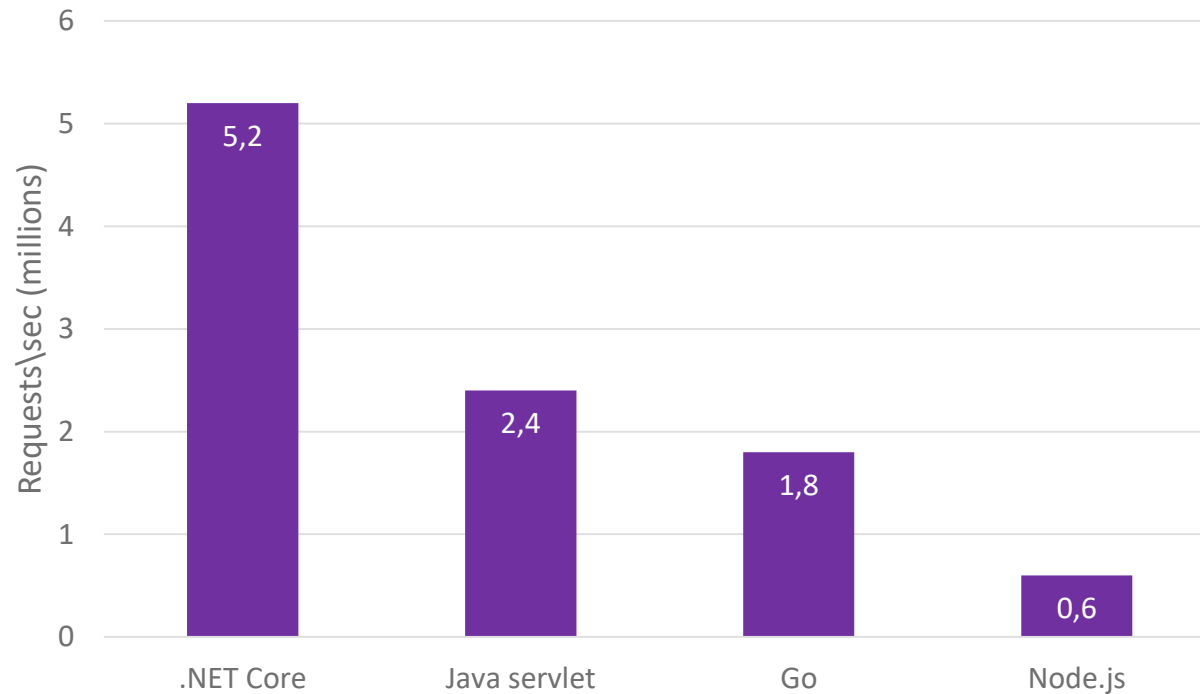


Performant



IMPLÉMENTATIONS DE .NET

- .NET Core: >8x faster than Node.js



- <https://github.com/aspnet/benchmarks>



IMPLÉMENTATIONS DE .NET

- Mono
 - utilisée quand un runtime réduit est requis
 - alimente les applications Xamarin sur Android, Mac, iOS, tvOS et watchOS
 - axé sur une faible consommation des ressources
 - prend en charge toutes les versions de .NET Standard publiées.
 - Historiquement : Implémentation Unix du Framework .NET
- Universal Windows Platform
 - Applications Windows tactiles modernes pour l'IoT.
 - Unification des développements pour plusieurs types d'appareil:
 - PC, Tablettes, Windows Phone, Xbox.
 - Peut être écrit en Javascript.



QUE PEUT-ON FAIRE AVEC ?

- Web :
 - ASP.NET WebForms
 - ASP.NET MVC
- Client lourd
 - Windows Presentation Foundation (WPF)
 - WinForm
- Services Web:
 - Windows Communication Foundation (WCF)
 - ASP.NET WebAPI
- Divers:
 - Windows Workflow Foundation
 - Applications Windows Store
 - Services Windows
 - Applications console
 - Applications mobiles



QUELQUES PRINCIPES EN VRAC

- Runtimes en fonction de l'implémentation
 - CLR (Common Language Runtime) pour .NET Framework
 - CoreCLR (Core Common Language Runtime) pour .NET Core
 - .NET Native pour la plateforme Windows universelle
 - Le runtime Mono pour Xamarin.iOS, Xamarin.Android, Xamarin.Mac et le framework de bureau Mono
- Applications compilées
- Assembly
- Code managé (Garbage Collector)
 - Nettoyage automatique de la mémoire

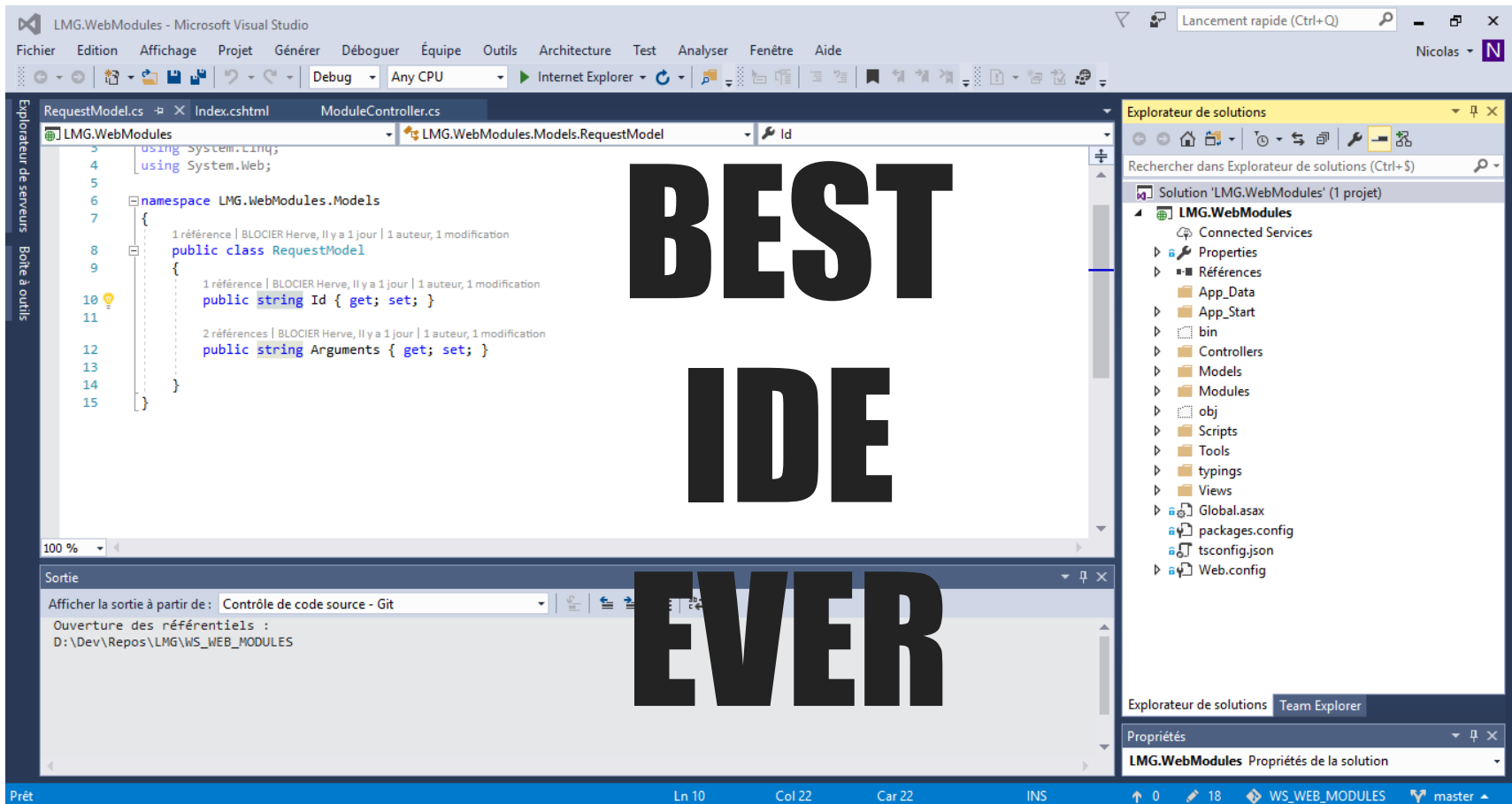


QUELQUES PRINCIPES EN VRAC

- Web : Compilation au runtime (JIT)
- Paramétrage dans des fichiers config XML
 - web.config et app.config
 - Prise en compte des modifications à chaud
- Beaucoup d'outils pour faciliter les développements
 - **Msbuild** : moteur de génération utilisé pour générer les projets
 - **NuGet**: gestionnaire de package de Microsoft pour .NET
 - Usines de développements compatibles:
 - Team Foundation Server
 - Team City
 - Jenkins



VISUAL STUDIO



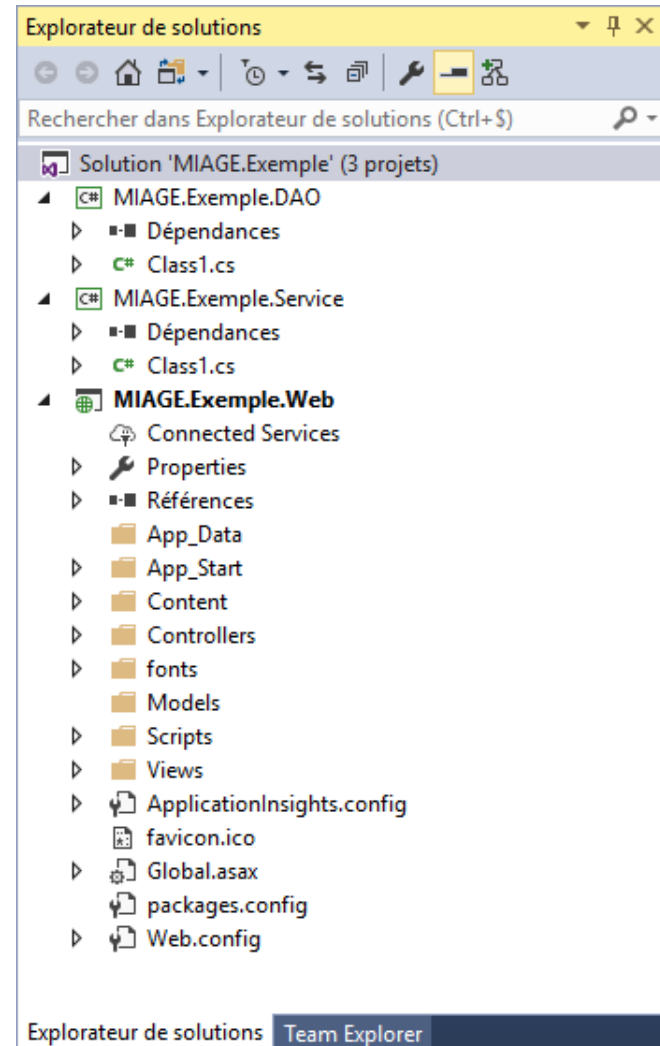
VISUAL STUDIO

- IDE Officiel de développement depuis 2003
- Nouvelle version tous les 2 ans
 - VS 2017 aujourd'hui
- 3 éditions:
 - Community (Gratuite)
 - Professionnelle (Payante)
 - Entreprise (Payante et super chère)
- Développement en .NET mais pas que...



ORGANISATION D'UN PROJET EN .NET

- Pas de Workspace comme en Java...
- Solution (.sln) : Regroupement de projets
- 1 projet = 1 assembly ou 1 application
- Les projets se référencent entre eux pour accéder à leur code
 - Attention au références cycliques
- Les projets portent les fichiers de code source



LES LANGAGES

```
public class MonExemple
{
    private int var1;
    private int var2;

    public int Prop1 { get; set; }
    public List<string> MyList { get; set; }

    public MonExemple()
    {
        MyList = new List<string>();
        var1 = 1;
        var2 = 2;
    }

    public void ReadList()
    {
        foreach(string item in MyList)
        {
            Console.WriteLine(item);
        }
    }

    public int Calc(int var3)
    {
        int result = var1 + var2 + var3;
        return result;
    }
}
```



```
Public Class MonExemple

    Private var1 As Integer
    Private var2 As Integer

    Public Property Prop1 As Integer
    Public Property MyList As New List(Of String)

    Public Sub MonExemple()
        var1 = 1
        var2 = 3
    End Sub

    Public Sub ReadList()
        For Each item As String In MyList
            Console.WriteLine(item)
        Next
    End Sub

    Public Function Calc(ByVal var3 As Integer) As Integer
        Dim result As Integer = var1 + var2 + var3
        Return result
    End Function

End Class
```

VB.net



LES LANGAGES

- F# : Langage de programmation fonctionnelle
- Les langages sont interopérables
 - Le compilateur transforme les instructions C# et VB en code intermédiaire nommé **Common Intermediate Language** (CIL)
 - C'est ce code qui est compilé par la CLR
- Possibilité de mélanger les langages dans une même solution
 - Absolument pas recommandé!!



FOCUS SUR C# : PRÉSENTATION

- Langage de Programmation Orienté Objet fortement typé
- Dérivé de C et C++ et ressemblant au Java
- Code source dans des fichiers **.cs**
- Vous savez programmer en Java, vous saurez programmer en C#
- **ATTENTION AUX NORMES DE NOMMAGE**

```
public class MonExemple
{
    private int var1;
    private int var2;

    public int Prop1 { get; set; }
    public List<string> MyList { get; set; }

    public MonExemple()
    {
        MyList = new List<string>();
        var1 = 1;
        var2 = 2;
    }

    public void ReadList()
    {
        foreach(string item in MyList)
        {
            Console.WriteLine(item);
        }
    }

    public int Calc(int var3)
    {
        int result = var1 + var2 + var3;
        return result;
    }
}
```



FOCUS SUR C# : LES BASES

• Les types de base

		<code>bool</code>	<code>System.Boolean</code>
<code>byte</code>	<code>System.Byte</code>	<code>char</code>	<code>System.Char</code>
<code>short</code>	<code>System.Int16</code>	<code>string</code>	<code>System.String</code>
<code>int</code>	<code>System.Int32</code>	<code>object</code>	<code>System.Object</code>
<code>long</code>	<code>System.Int64</code>	<code>DateTime</code>	<code>System.DateTime</code>
<code>float</code>	<code>System.Single</code>	<code>TimeSpan</code>	<code>System.TimeSpan</code>
<code>double</code>	<code>System.Double</code>		

• Les opérations de contrôle

```
if (itest == 10)
    Console.WriteLine("Bonjour");
```

```
if (condition1)
{ ... }
else if (condition2)
{ ... }
else
{ ... }
```

```
switch (var)
{
    case 0:
        //...
        break;
    case 1 :
    case 2 :
        //...
        break;
    default :
        //...
        break;
}
```

• Les opérateurs

- affectation : `a = b`
- arithmétiques : `+` `-` `*` `%`
 - `++` `--`
- logiques : `!` (non), `&&` (et), `||` (ou),
 - `&` (et), `|` (ou)
 - `+` pour concaténer des chaînes
- relationnels : `==` `!=` `<` `<=` `>` `>=`
- ternaire : si alors sinon simplifié : `(a ? b : c)`

• Les boucles

```
for (int i = 0; i <= 10; i++)
{
    Console.WriteLine(i);
}
```

```
string[] tab = { "bleu", "rouge", "vert" };
foreach (string item in tab)
    Console.WriteLine(item + "<br>");
```

```
int i = 0;
while (i <= 15)
{
    Console.WriteLine("Valeur de i = " + i);
    i++;
}
```



```

public class Vehicule
{
    private string couleur;
    private DateTime dateMiseEnCirculation;
    public string Couleur
    {
        get { return couleur; }
        set { couleur = value; }
    }
    public DateTime DateMiseEnCirculation
    {
        get { return dateMiseEnCirculation; }
        set { dateMiseEnCirculation = value; }
    }
    public void Rouler()
    {
        // TODO : coder de la méthode Rouler
    }
    public double CalculerConsommation()
    {
        // TODO : Coder la méthode CalculerConsommation
    }
}

```

• Namespace

- Une classe appartient à un Namespace

```

namespace MIAGE.Exemple.Web.Models
{
    public class PlayerModel
    {
        ...
    }
}

```

classe précédé de son namespace

• Héritage et interfaces

```

public class Heir : Elder
{
    public override string WhoAmI()
    {
        return base.WhoAmI();
    }
}

```

• Polymorphisme

```

public class Elder
{
    public virtual string WhoAmI()
    {
        return "je suis un Ancetre";
    }
}

```

méthode d'une classe mère (La méthode mère doit être **virtual**)

```

public class Heir : Elder{
    public override string WhoAmI() : base()
    {
    }
}

```



FOCUS SUR C# : LES COLLECTIONS & GÉNÉRICITÉ

- ArrayList

- Peut contenir n'importe quoi
- Principales méthodes et propriétés :
 - Add()
 - Clear()
 - Count
 - Contains()
 - ToArray()
 - **foreach**

- List<T>

```
List<Person> list = new List<Person>();
foreach (Person p in list)
{
    // Pas de transtypage de object en Personne
    // Le type personne est connu à se stade
    Console.WriteLine(p.Nom);
}
```

- Hashtable

- Dictionnaire indexé
- Principales méthodes et propriétés :
 - Add()
 - Clear()
 - Count
 - Contains()
 - **Keys**
 - **Values**

- Dictionary<T1, T2>

```
Dictionary<string, Person> dic = new Dictionary<string, Person>();
foreach (Person p in dic.Values)
{
    Console.WriteLine(p.Name);
}
```



FOCUS SUR C# : QUELQUES SPÉCIFICITÉS

• Propriétés

```
private string couleur;  
public string Couleur  
{  
    get { return couleur; }  
    set { couleur = value; }  
}
```

```
public string Couleur { get; set; }
```

• Exceptions

- Pas de déclaration des exceptions levée en entête de méthode

• Lambda

```
List<Animal> liste = new List<Animal>();  
liste.add(new Cheval("Jumper"));  
liste.add(new Perroquet("Coco"));  
liste.add(new Cheval("Maurice"));  
  
string nomRecherche = "Maurice";  
Animal animalRecherche;  
animalRecherche = liste.Find(a => a.Nom ==  
nomRecherche);
```

• Normes de nommage

- PascalCase: Classes, méthodes, propriétés, enum
- camelCase: variable, attributs, etc...

• Visibilité

- private
- protected
- internal
- public

• Var

- Déclaration dynamique de variable
- Variables de méthode uniquement
- Reste fortement typé

• Linq

```
List<Person> persons = ...;  
  
var q1 = from p in persons  
         where p.Sex == Sex.Female && p.Age > 30  
         select p;  
  
int result = q1.Count();
```



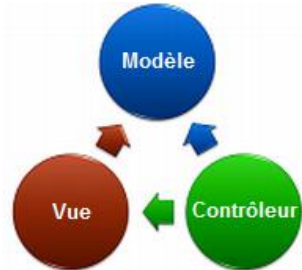
ASP.NET

- Applications Web portés par .NET
- Hébergé par IIS (Windows)
- 3 déclinaisons
 - ASP.NET WebForms
 - Fichiers aspx
 - Faire du web comme ans un client lourd (notions d'évènements)
 - ASP.NET MVC
 - ASP.NET WebAPI 2
 - A suivre!! ;)
- ASP.NET Core
 - Version open source d'ASP.NET
 - Réécriture total du modèle de programmation
 - Peut être hébergé sur n'importe quelle plateforme

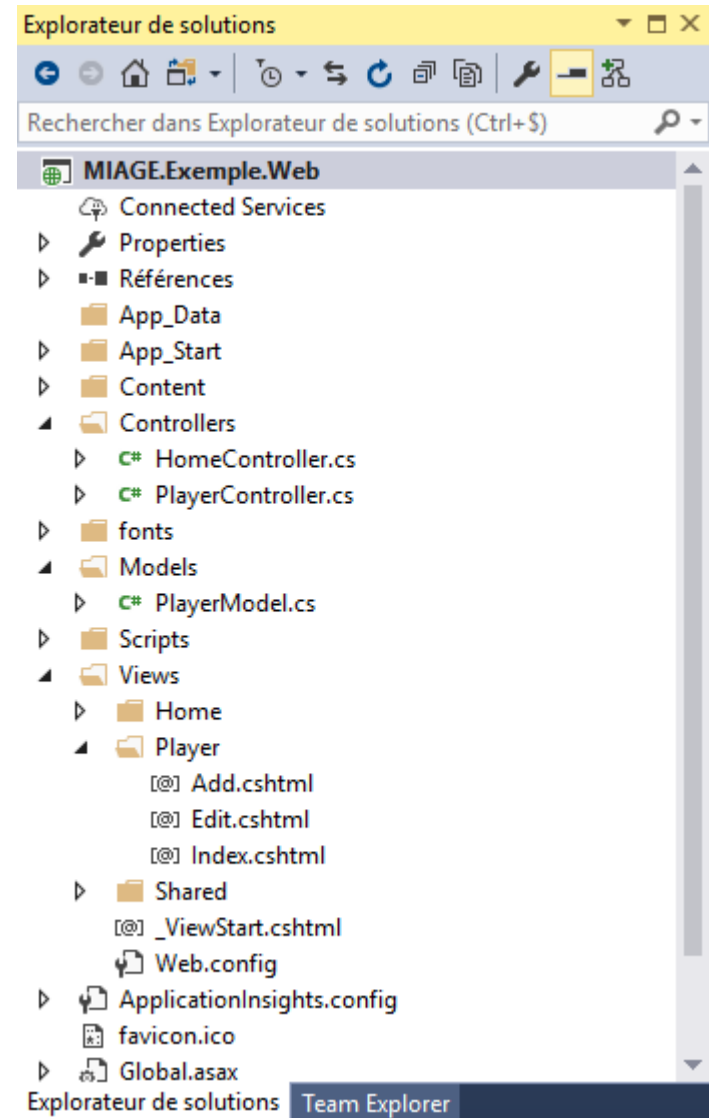


ASP.NET MVC

- Pattern MVC



- Appel des routes et non des pages
- Route définie par:
 - Un contrôleur
 - Une action
 - Un paramètre
 - Un verbe



ASP.NET MVC : MODÈLES

- Permet de transmettre la donnée
- Objet POCO

```
namespace MIAGE.Exemple.Web.Models
{
    public class PlayerModel
    {
        [Required]
        public int Id { get; set; }

        [Required]
        [MaxLength(20)]
        public string Name { get; set; }

        [Required]
        [EmailAddress]
        public string Email { get; set; }
    }
}
```



ASP.NET MVC : CONTROLEURS

- Contient les actions
- Par défaut, l'action Index est appelée
- Des attribut permettent d'orienter sur la bonne méthode en fonction du verbe Http

```
public class PlayerController : Controller
{
    public ActionResult Add()
    {
        PlayerModel model = new PlayerModel();
        return View(model);
    }

    [HttpPost]
    public ActionResult Add(PlayerModel model)
    {
        if (!ModelState.IsValid)
            return View(model);
        players.Add(model);

        return RedirectToAction("Index");
    }
}
```



ASP.NET MVC : VUES

- Permet d'afficher et de mettre en forme les données
- Fichier .cshtml
- Moteur de rendu Razor

```
@model MIAGE.Exemple.Web.Models.PlayerModel

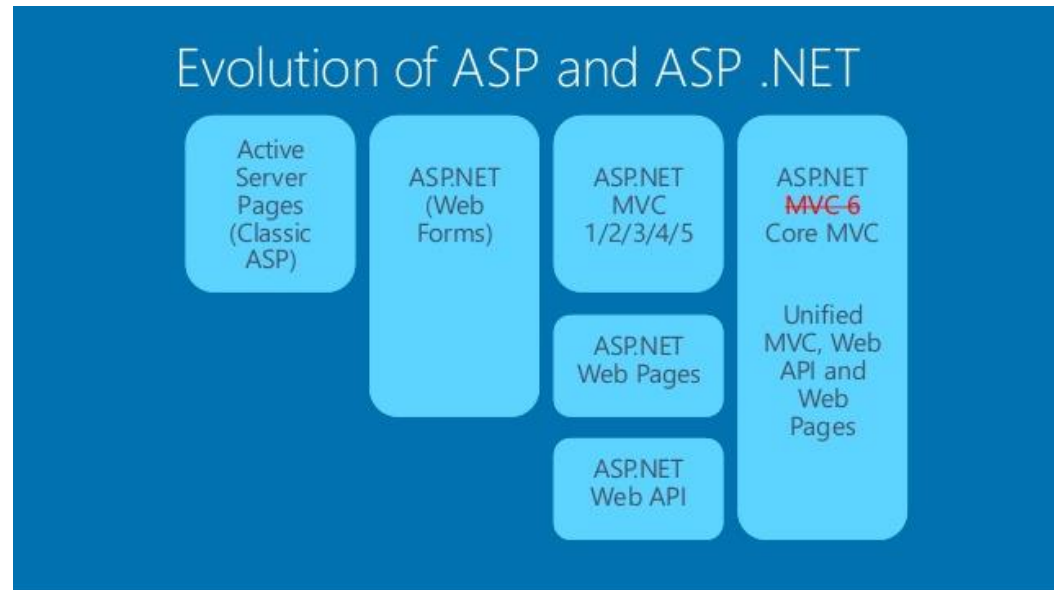
@{
    ViewBag.Title = "Player";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Player</h2>
@Html.ActionLink("<< Retour", "Index")
@using (Html.BeginForm())
{
    <div>
        @Html.LabelFor(x => x.Id)
        @Html.TextBoxFor(x => x.Id)
        @Html.ValidationMessageFor(x => x.Id)
    </div>
    <div>
        @Html.LabelFor(x => x.Name)
        @Html.TextBoxFor(x => x.Name)
        @Html.ValidationMessageFor(x => x.Name)
    </div>
    <div>
        @Html.LabelFor(x => x.Email)
        @Html.TextBoxFor(x => x.Email)
        @Html.ValidationMessageFor(x => x.Email)
    </div>
    <div>
        <button>Valider</button>
    </div>
}
```



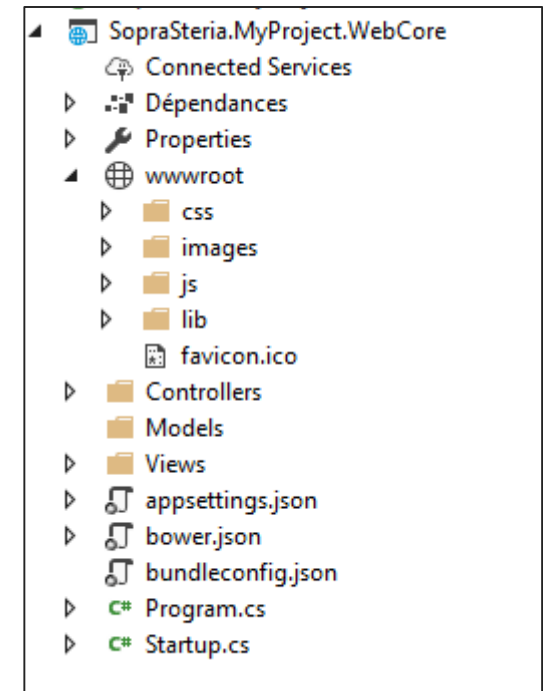
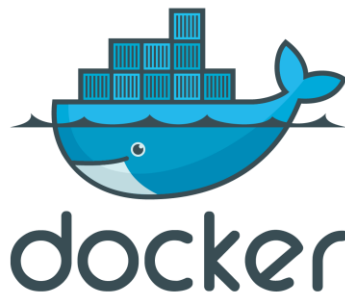
ASP.NET CORE

- Ca change quoi?
 - Refonte totale du modèle de programmation.
 - Fonctionne uniquement en .NET Core mais peut référencer des projet en .NET Framework
 - Evolution naturelle de l'ancien modèle



ASP.NET CORE

- Ce qui change concrètement
 - Nouvelle arborescence projet
 - wwwroot contient toutes les ressources statiques
 - Devient une application console qui peut être exécuté par n'importe quel serveur web
 - Configuration applicative définie en json
 - La syntaxe des vues
 - La prise en charge de Docker

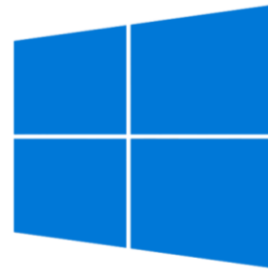


ET PLEINS D'AUTRES CHOSE ENCORE...

- Window Presentation Foundation:
 - Développement des interface en XAML dérivé de XML
 - Utilisation de BLEND pour designer ses interfaces



- Universal Windows Application:
 - Applications pour Windows 10
 - Portables sur WP10, Xbox ONE

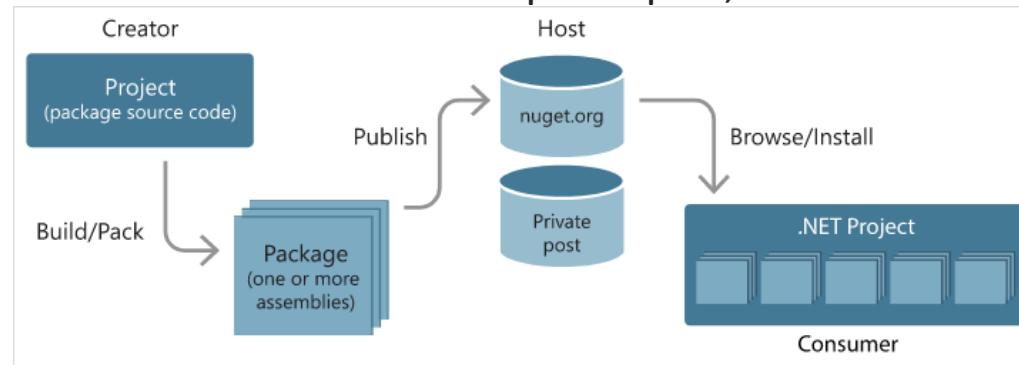


- Xamarin
 - Applications natives mobiles pour WP, android et IOS



NUGET : WHAT IS IT?

- Gestionnaire de packages
- Repository public :
 - 91 185 packages alimentés par la communauté et par Microsoft
 - Si vous avez besoin d'une fonctionnalité spécifiques, il existe sûrement sur NuGet



- Repository privé:
 - Possibilité d'héberger son propre repo pour ses besoins

Important : Si vous utilisez un package dans votre projet, pensez à vérifier la licence d'utilisation.



L'INFORMATIQUE DANS LES NUAGES

- C'est quoi le cloud ?
 - Un ensemble de service hébergé ailleurs que dans votre infrastructure
 - Facturation à la minute

- Qui sont les principaux acteurs



Google Cloud Platform

- Quelques termes :
 - On-premise
 - Iaas
 - Paas
 - Saas



ECOSYSÈMES MICROSOFT



RÉFÉRENCES

- MSDN : Microsoft Developer Network
 - <https://msdn.microsoft.com>
- DOCS
 - <https://docs.microsoft.com/fr-fr/>
- Microsoft Virtual Academy
 - <https://mva.microsoft.com>
- Stackoverflow
 - <http://stackoverflow.com/>



INITIATION AU .NET

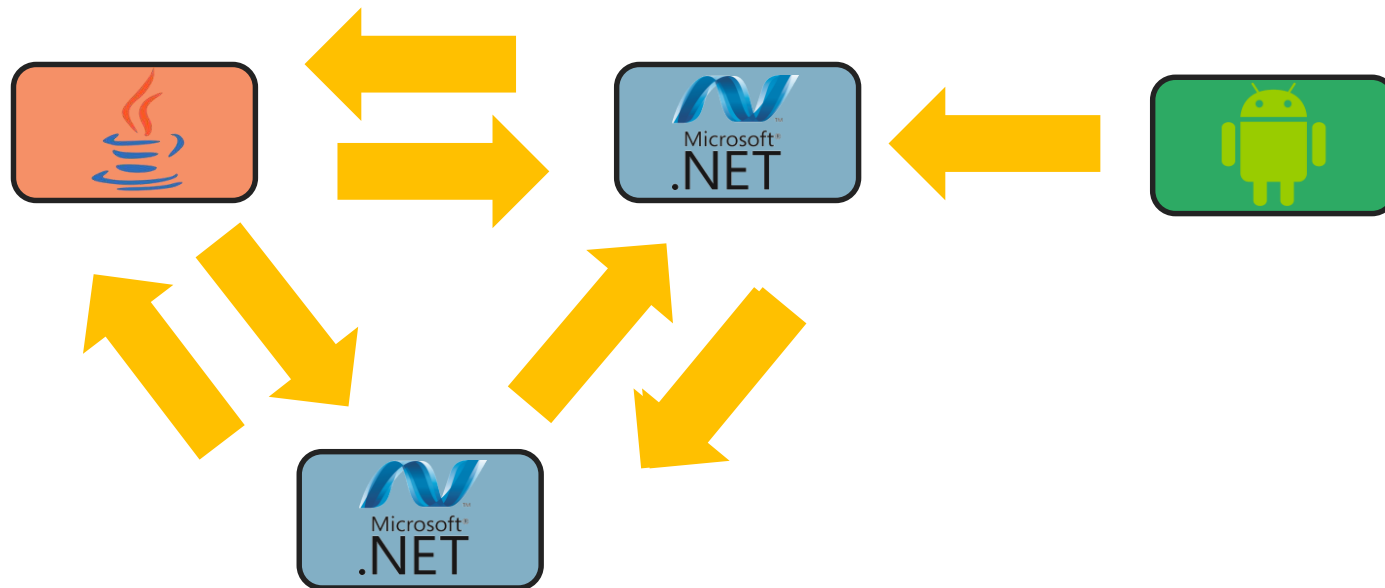
Focus sur Windows Communication Foundation



Delivering Transformation. Together.

A QUOI SERT UN WEB SERVICE?

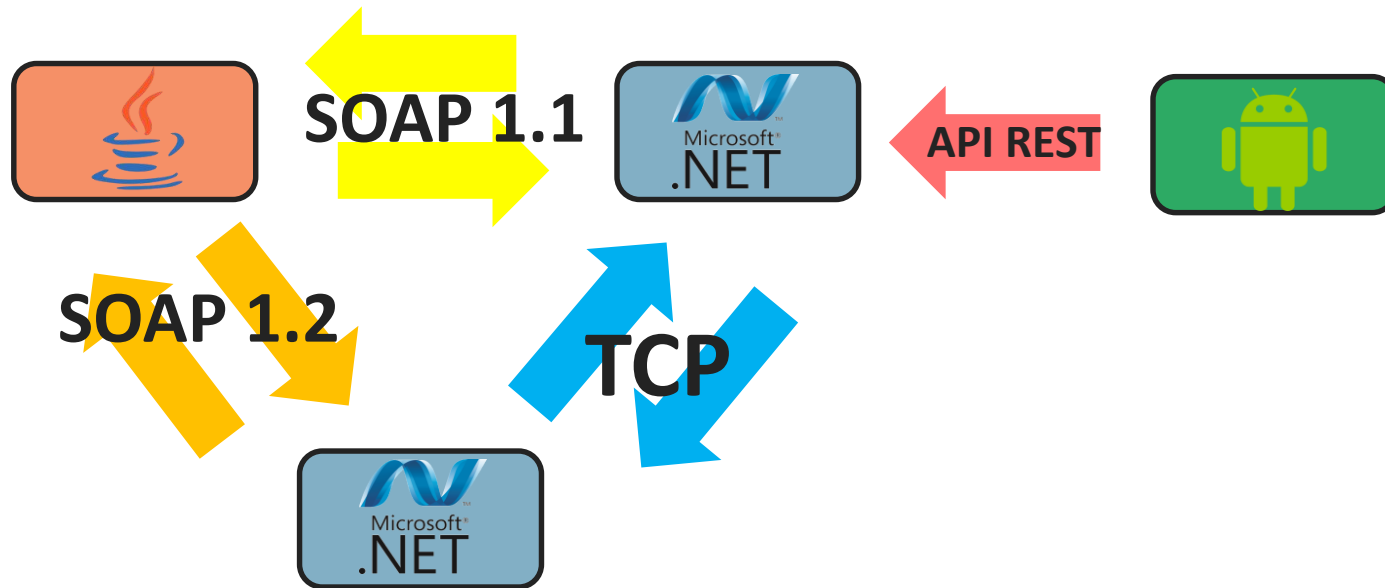
- Permettre à des applications de technologies homogènes ou hétérogènes de pouvoir communiquer entre elles.



- Interopérabilité par le protocole SOAP ou par des API respectant l'architecture REST

ET WCF DANS TOUT ÇA?

- Couche orienté service du Framework .NET
- Approche orienté service avec interopérabilité
- Possibilité de développer des service en abstraction du protocole utilisé



- Open source depuis le mai 2015 : <https://github.com/dotnet/wcf>

BASE D'UN SERVICE

- Un service (on parle d'*Endpoint*) est défini par :
 - Une adresse
 - Où se trouve le service (URI)
 - Un contrat
 - Quelles méthodes sont exposées (symbolisé par le WSDL)
 - Un binding
 - La méthode de communication utilisée

BINDINGS

Binding	Sécurité	Protocole	Encodage
basicHttpBinding	None, Transport, Message, Mixed	HTTP	Text/XML, MTOM
wsHttpBinding	Message, Transport, Mixed	HTTP	Text/XML, MTOM
netTcpBinding	Transport, Message, Mixed	TCP	Binary
netNamedPipeBinding	Transport, None	Named Pipe	Binary
netMsmqBinding	Message, Transport, None	TCP	Binary



CRÉATION D'UN SERVICE

- Définition du contrat
 - C'est une interface au sens objet
 - L'interface doit être marquée par l'attribut `[ServiceContract]`
 - Chaque méthode exposée à l'extérieur doit être marquée par l'attribut `[OperationContract]`
- Exemple :

```
[ServiceContract]
public interface IService1
{
    [OperationContract]
    string GetData(int value);

    [OperationContract]
    CompositeType GetDataUsingDataContract(CompositeType composite);
}
```

CRÉATION D'UN SERVICE

- Implémentation des méthode
 - Implémentation dans un fichier svc
- Exemple :

```
public class Service1 : IService1
{
    public string GetData(int value)
    {
        return string.Format("You entered: {0}", value);
    }

    public CompositeType GetDataUsingDataContract(CompositeType composite)
    {
        if (composite == null)
        {
            ...
        }
    }
}
```

CRÉATION D'UN SERVICE

- Modification des entités transférées
 - Pour qu'un objet soit transmis il faut qu'il possède les attributs de sérialisation
 - [DataContract] sur la classe
 - [DataMember] sur les propriétés à transmettre
- Exemple

```
[DataContract]
public class CompositeType
{
    bool boolValue = true;
    string stringValue = "Hello ";

    [DataMember]
    public bool BoolValue{...}

    [DataMember]
    public string StringValue{...}
}
```

CRÉATION D'UN SERVICE

- Cas du polymorphisme
 - Lorsque l'objet transféré peut être dérivé, la classe de base doit définir les types dérivés possibles avec l'attribut `[KnownType]`
 - Exemple

```
[DataContract]
[KnownType(typeof(EntiteChampInt))]
[KnownType(typeof(EntiteChampString))]
[KnownType(typeof(EntiteChampDate))]
public abstract class EntiteChamps
{
    [DataMember]
    public int Id { get; set; }
    [DataMember]
    public int IdConfig { get; set; }
}
```

```
[DataContract]
public class EntiteChampString : EntiteChamps
{
    [DataMember]
    public string Valeur { get; set; }
}
```

CONSOMMATION D'UN SERVICE

- Sur le programme client, ajout d'une référence vers le service (Add Service Reference)
- Appeler le service depuis une fonction :

```
ServiceReference1.Service1Client client = new ServiceReference1.Service1Client();  
string retour = client.GetData(12);
```

- Configurer le programme

CONSOMMATION D'UN SERVICE

- Configuration (app.config ou web.config)

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="BasicHttpBinding_IService1" closeTimeout="00:01:00"
        openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
        allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
        maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
        messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
        useDefaultWebProxy="true">
        <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
          maxBytesPerRead="4096" maxNameTableCharCount="16384" />
        <security mode="None">
          <transport clientCredentialType="None" proxyCredentialType="None"
            realm="" />
          <message clientCredentialType="UserName" algorithmSuite="Default" />
        </security>
      </binding>
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint address="http://localhost:1081/Service1.svc" binding="basicHttpBinding"
      bindingConfiguration="BasicHttpBinding_IService1" contract="ServiceReference1.IService1"
      name="BasicHttpBinding_IService1" />
  </client>
</system.serviceModel>
```



```

20
21
22 while (again) {
23     iN = -1;
24     again = false;
25     getline(cin, sInput);
26     system("cls");
27     stringstream(sInput) >> dblTemp;
28     iLength = sInput.length();
29     if (iLength < 4) {
30         again = true;
31         continue;
32     } else if (sInput[iLength - 3] != '.') {
33         again = true;
34         continue;
35     }
36     if (iN < iLength) {
37         cout[iN]

```

DEMO

INITIATION AU .NET

Focus sur ASP.NET Core Web API



ASP.NET CORE WEB API: WHAT IS IT?

- Surcouche ASP.NET équivalent à ASP.NET MVC
- Permet de créer des services respectant l'architecture REST
- Petit rappel:

Requêtes REST : 3 caractéristiques

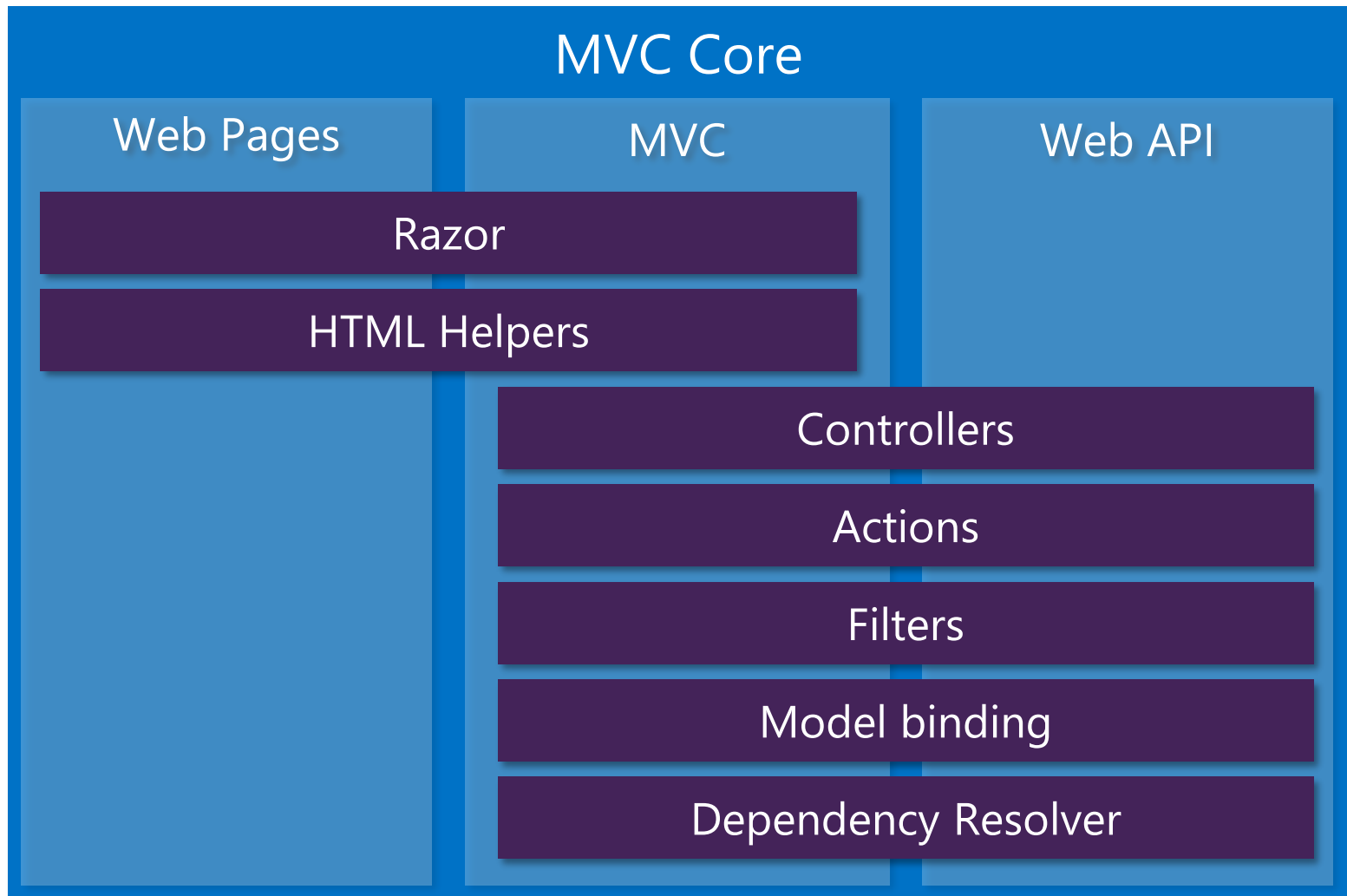
- Ressources
 - ▣ Identifiée par une URI
`http://univ-orleans.fr/cursus/master/miage/sir/2`
- Méthodes (verbes) permettant de manipuler les ressources (identifiants)
 - ▣ Méthodes limitées strictement à HTTP : GET, POST, PUT, DELETE
- Représentation : Vue sur l'état de la ressource
 - ▣ Format d'échanges entre le client et le serveur (XML, JSON, text/plain,...)

CRÉATION FACILITÉ POUR DES SERVICES WEB RESTFULL

- Modèle de programmation HTTP moderne
 - Permet de manipuler des requêtes et réponses HTTP à l'aide d'un modèle objet HTTP fortement typé
- Négociation de contenu
 - Client serveur travaillant ensemble pour déterminer le bon format de retour
 - XML et JSON par défaut
- Prise en charge complète des routes
 - la cartographie des actions possède un support complet pour les conventions
 - Plus besoins d'attributs aux classes et aux méthodes
 - Configuration faite à travers le code
- Validation des modèles
 - Extraction des données de diverses parties d'une requête HTTP
 - Conversion de celles-ci en objets .NET



ASP.NET CORE



UN EXEMPLE DE CONTRÔLEUR

- Le nom de la méthode n'est pas important
- Renvoi :
 - Un objet
 - Une collection d'objet
 - Vide
 - Un résultat de type IActionResult

```
namespace SopraSteria.MyProject.WebApiCore.Controllers
{
    [Route("api/[controller]")]
    public class StudentController : Controller
    {
        // GET api/values
        [HttpGet]
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        // GET api/values/5
        [HttpGet("{id}")]
        public string Get(int id)
        {
            return "value";
        }

        // POST api/values
        [HttpPost]
        public void Post([FromBody]string value)
        {
        }

        // PUT api/values/5
        [HttpPut("{id}")]
        public void Put(int id, [FromBody]string value)
        {
        }

        // DELETE api/values/5
        [HttpDelete("{id}")]
        public void Delete(int id)
        {
        }
    }
}
```



SYSTÈME DE ROUTAGE POUR WEB API

- L'utilisation de HTTP permet donc à Web API d'identifier automatiquement la méthode du contrôleur qui sera exécutée en fonction du verbe HTTP qui est effectué (GET, POST, etc.).
- Routage porté par le contrôleur :

```
[Route("api/[controller]")]
```

- Par convention, en ASP.NET, on préfixe les contrôleur par api/ dans les urls
- [controller] se substitue par le nom du contrôleur dans l'url
- L'action désignée en fonction du verbe HTTP dépend de l'attribut



SYSTÈME DE ROUTAGE POUR WEB API

Méthode HTTP	Chemin URI	Attribut	Description
Get	api/values	HttpGet	Retourne toutes les données comme un IEnumerable.
Get	api/values/2	HttpGet("{id}")	Retourne l'élément ayant pour Id 2.
Post	api/values	HttpPost()	Enregistrement d'un nouvel élément.
Put	api/values/2	HttpPut()	Modification de l'élément ayant pour Id 2.
Delete	api/values/2	HttpDelete("{id}")	Suppression de l'élément ayant pour Id 2.



UN EXEMPLE PLUS POUSSÉ

```
[Produces("application/json")]
[Route("api/Customer")]
public class CustomerController : Controller
{
    private List<CustomerModel> Customers = new List<CustomerModel>()
    {
        new CustomerModel { Id = 1, FirstName = "Hinault", LastName = "Romaric", EMail = "hr@gmail.com"},
        new CustomerModel { Id = 2, FirstName = "Thomas", LastName = "Perrin", EMail = "thomas@outlook.com"},
        new CustomerModel { Id = 3, FirstName = "Allan", LastName = "Croft", EMail = "allan.croft@crt.com"},
        new CustomerModel { Id = 3, FirstName = "Sahra", LastName = "Parker", EMail = "sahra@yahoo.com"},
    };

    // GET: api/Customer
    [HttpGet]
    public IActionResult Get()
    {
        return Ok(Customers);
    }

    // GET: api/Customer/5
    [HttpGet("{id}", Name = "Get")]
    public IActionResult Get(int id)
    {
        CustomerModel model = Customers.SingleOrDefault(x => x.Id == id);

        if (model == null)
            return BadRequest("Is inexistant");

        return Ok(model);
    }
}
```

RETOUR DE L'EXEMPLE

• XML

```
<ArrayOfCustomer xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://schemas.datacontract.org/2004/07/FirstWebAPI.Models">
  <Customer>
    <EMail>hr@gmail.com</EMail>
    <FirstName>Hinault</FirstName>
    <Id>1</Id>
    <LastName>Romaric</LastName>
  </Customer>
  <Customer>
    <EMail>thomas@outlook.com</EMail>
    <FirstName>Thomas</FirstName>
    <Id>2</Id>
    <LastName>Perrin</LastName>
  </Customer>
  <Customer>
    <EMail>allan.croft@crt.com</EMail>
    <FirstName>Allan</FirstName>
    <Id>3</Id>
    <LastName>Croft</LastName>
  </Customer>
  <Customer>
    <EMail>sahra@yahoo.com</EMail>
    <FirstName>Sahra</FirstName>
    <Id>3</Id>
    <LastName>Parker</LastName>
  </Customer>
</ArrayOfCustomer>
```

• JSON

```
[{
  "Id" : 1,
  "FirstName" : "Hinault",
  "LastName" : "Romaric",
  "EMail" : "hr@gmail.com"
}, {
  "Id" : 2,
  "FirstName" : "Thomas",
  "LastName" : "Perrin",
  "EMail" : "thomas@outlook.com"
}, {
  "Id" : 3,
  "FirstName" : "Allan",
  "LastName" : "Croft",
  "EMail" : "allan.croft@crt.com"
}, {
  "Id" : 3,
  "FirstName" : "Sahra",
  "LastName" : "Parker",
  "EMail" : "sahra@yahoo.com"
}
]
```



EXEMPLE D'APPEL

```
<script type="text/javascript">
    $(document).ready(function () {
        // Envoi de la requête Ajax
        $.getJSON("api/Customer/",
            function (data) {
                // En cas de succès, 'data' contient la liste des clients.
                $.each(data, function (key, val) {

                    var str = val.Id + ' ' + val.FirstName + ' ' + val.LastName;

                    $('<li/>', { html: str }).appendTo($('#customers'));
                });
            });
    });
</script>
```



```

20
21
22 while (again) {
23     iN = -1;
24     again = false;
25     getline(cin, sInput);
26     system("cls");
27     stringstream(sInput) >> dblTemp;
28     iLength = sInput.length();
29     if (iLength < 4) {
30         again = true;
31         continue;
32     } else if (sInput[iLength - 3] != '.') {
33         again = true;
34         continue;
35     }
36     if (iN < iLength) {
37         cout[iN]

```

DEMO

QUESTIONS POTENTIELLES

- Comment s'appelle la spécification officielle des API .NET ?
- Quel est la dernière version du Framework .NET ?
- Comment s'appelle la version open-source du .NET ?
- Quel Framework permet de faire des applications mobiles en .NET ?
- Qu'est-ce que Nuget ?
- Citer les 3 langages géré dans le Framework .NET ?
- Quelle implémentation du Framework permet de faire des webservice SOAP ?
- Quelle sont les 4 mots clés importants en WCF ?
- Qu'est ce qui permet de relier un verbe http à une méthode de contrôleur en ASP.NET WebAPI ?
- Quelle est la forme la plus commune d'une réponse à une requête ASP.NET WebAPI ?

