MIAGE – INITIATION AU .NET

Delivering Transformation. Together.

sopra steria

ME, MYSELF & I : NICOLAS FLEURY

- Architecte .NET
- MIAGE Promo 2007
- 10 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 Web API 2

MIAGE – INITIATION AU .NET

Introduction au Framework Microsoft.NET

Delivering Transformation. Together.

sopra steria

QU'EST-CE QUE LE .NET

- Plateforme permettant de développer et d'exécuter des applications orientés Windows
- Réponse de Microsoft à Java et à J2EE
- Historiquement : Multi langage, mono plateforme
  - C#, VB.NET, J#, C++, Eiffel, Python
  - Java est mono langage, multi plate forme
- Code exécuté par le CLR (Common Language Runtime)
- Environnement d'exécution managé

QU'EST CE QU'ON PEUT FAIRE AVEC DU .NET

- Applications lourdes (WinForm, WPF)
- Applications Web (ASP.NET WebForms, ASP.NET MVC)
- WebServices (WCF, ASP.NET WebAPI)
- Services Windows
- Applications Console
- Applications Cloud-Ready (Azure)
- Applications mobiles natives Windows Phone, Android, IOS (Xamarin)
- Add-In Office, Sharepoint, Visual Studio

.NET À TRAVERS L'HISTOIRE

- Version 1.1
  - WinForms et applications console
  - ASP.NET et WebServices
  - .NET Remoting
  - ADO.NET
- Version 2.0
  - ASP.NET 2.0
  - Généricité et Type nullable
- Version 3 (surcouche de 2.0)
  - Windows Presentation Foundation (WPF)
  - Windows Communication Foundation (WCF)
  - Windows Workflow Foundation (WWF)
- Version 3.5
  - Entity Framework
  - LINQ
  - Expressions lambda
- Version 4.0
  - Refonte Entity Framework
  - Refonte Workflow Foundation
  - ASP.NET MVC 2
  - ASP.NET MVC 3
- Version 4.5
  - Amélioration de l'asynchronisme
  - ASP.NET MVC 4
  - ASP.NET Web API
  - Applications Windows Store
- Version 4.6
  - Amélioration ASP.NET Webforms
  - ASP.NET MVC 5
  - ASP.NET Web API 2
  - RyuJIT

.NET AUJOURD'HUI : 2 FRAMEWORK

- Framework .NET « classique »
  - Version : 4.6.2
  - Améliorations des version 4.6 et 4.6.1
- Framework.NET CORE:
  - Version modulaire Open-Source du Framework .NET depuis Novembre 2014
  - Multi-plateformes

  - Sources sur github

Microsoft ASP.NET

Github.com/aspnet

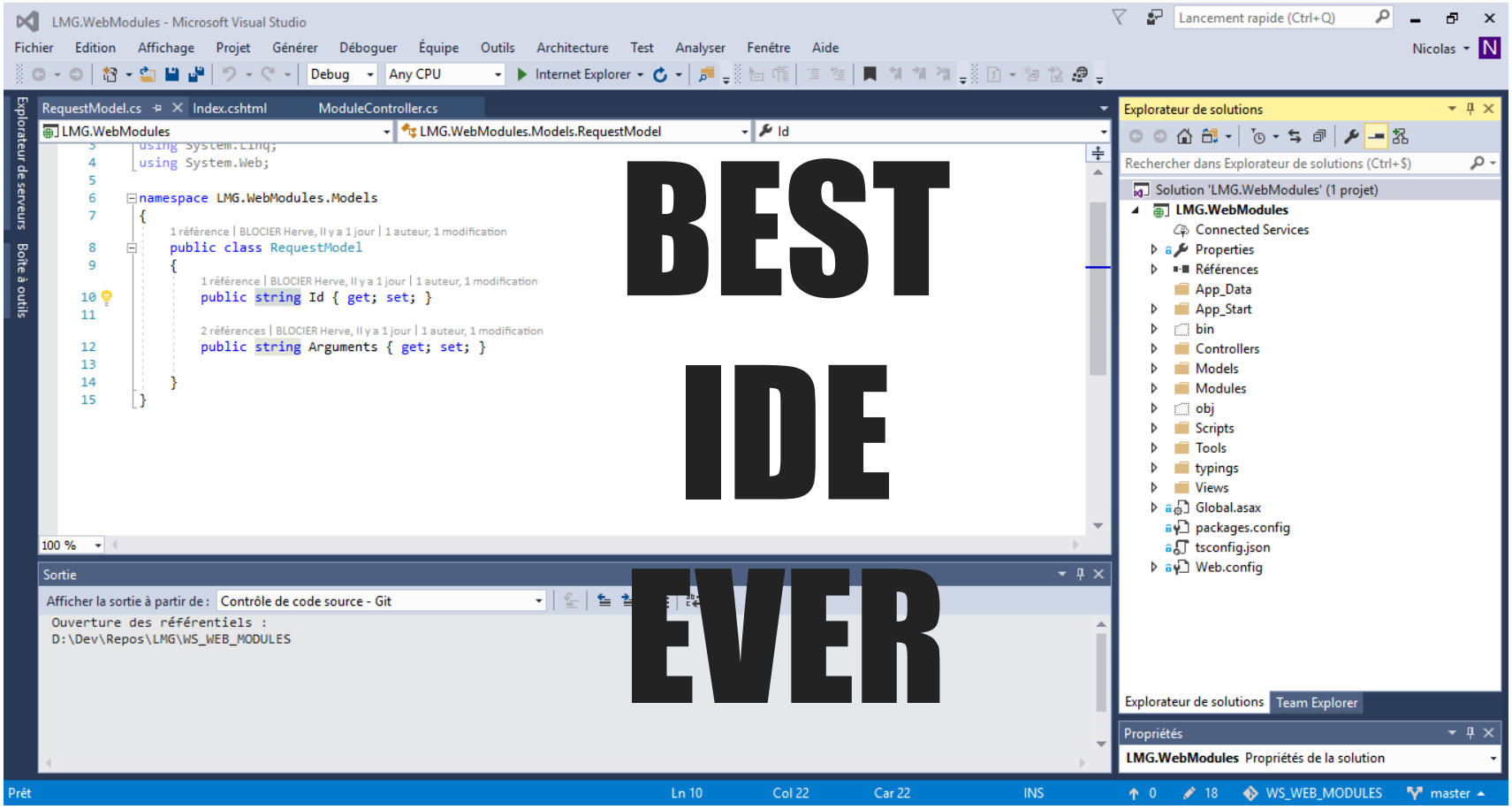
.NET foundation

Github.com/dotnet

QUELQUES PRINCIPES EN VRAC

- Applications compilées
- Assembly
- Code managé (Garbage Collector)
  - Nettoyage automatique de la mémoire
- 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
- Visual Studio

VISUAL STUDIO

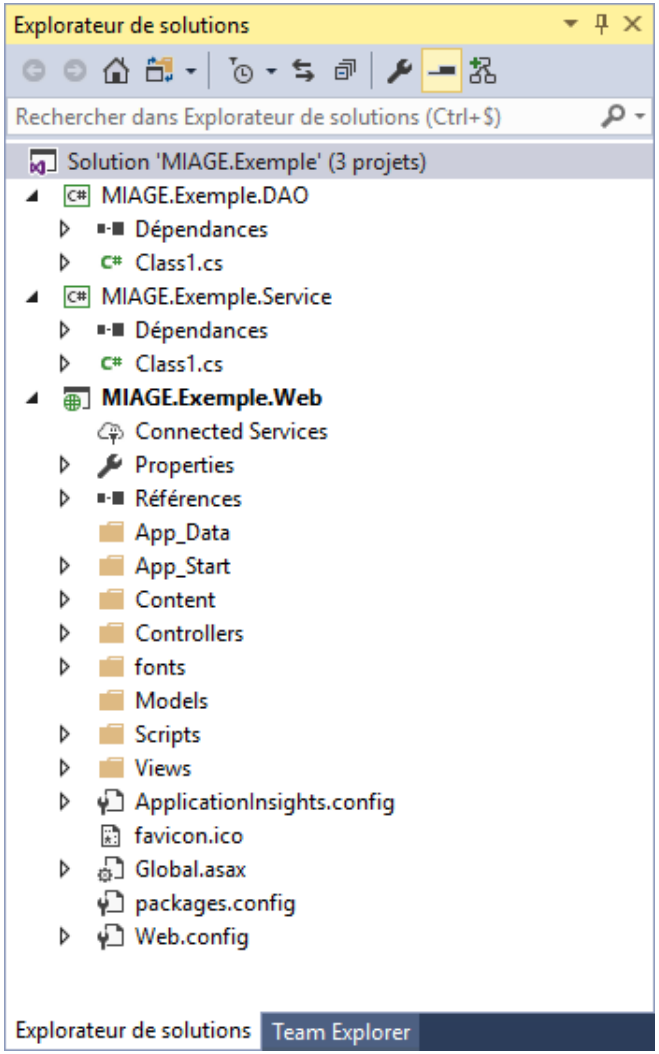


VISUAL STUDIO

- IDE Officiel de développement depuis 2003
- Nouvelle version tous les 2 ans
  - VS 2015 aujourd'hui
  - VS 2017 incoming!!!
- 3 éditions:
  - Community (Gratuite)
  - Profesionelle (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

```csharp
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;
    }
}
```


```vb
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
```


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 programmez 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

byte	System.Byte
short	System.Int16
int	System.Int32
long	System.Int64
float	System.Single
double	System.Double

bool	System.Boolean
char	System.Char
string	System.String
object	System.Object
DateTime	System.DateTime
TimeSpan	System.TimeSpan

- 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 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 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
  - …
  - …
  - …
  - Nom complet d'une classe : Nom de la classe précédé de son namspace

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

- Héritage et interfaces

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

}
```

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

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

```
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**
- Dictionnary<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**

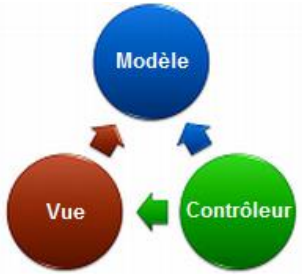
```
 List<Person> persons = …;

var q1 = from p in persons
         where p.Sex == Sex.Female && p.Age > 30
         select p;

int result = q1.Count();
```


ASP.NET

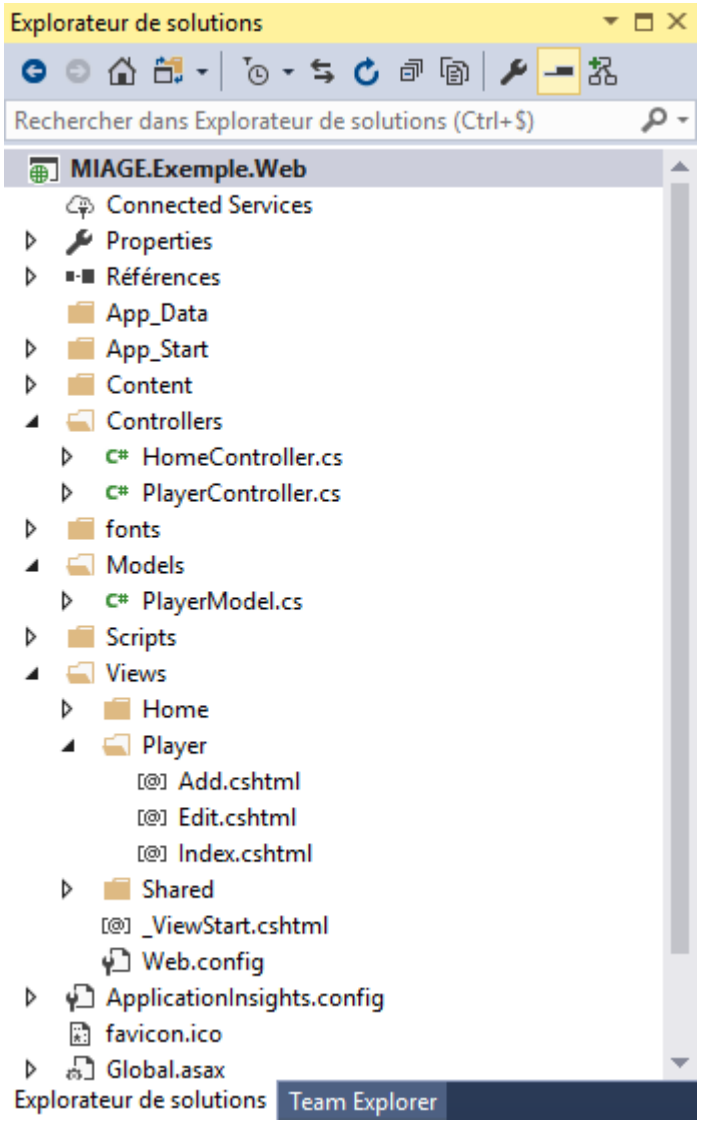
- Applications Web portés par .NET
- Hébergé par IIS (Windows)
- 3 déclinaisons
  - ASP.NET WebForms
    - Fichiers aspx
    - Faire du web comme ans un client lourd (notions d'évènements)
  - ASP.NET MVC
  - ASP.NET WebAPI 2
    - A suivre!! ;)
- ASP.NET Core
  - Version open source d'ASP.NET
  - Réécriture 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>
}
```

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, andoid et IOS

ECOSYSÈMES MICROSOFT

RÉFÉRENCES

- MSDN : Microsoft Developer Network
  - https://msdn.microsoft.com
- Microsoft Virtual Academy
  - https://mva.microsoft.com
- Stackoverflow
  - http://stackoverflow.com/

MIAGE – INITIATION AU .NET

Focus sur Windows Communication Foundation

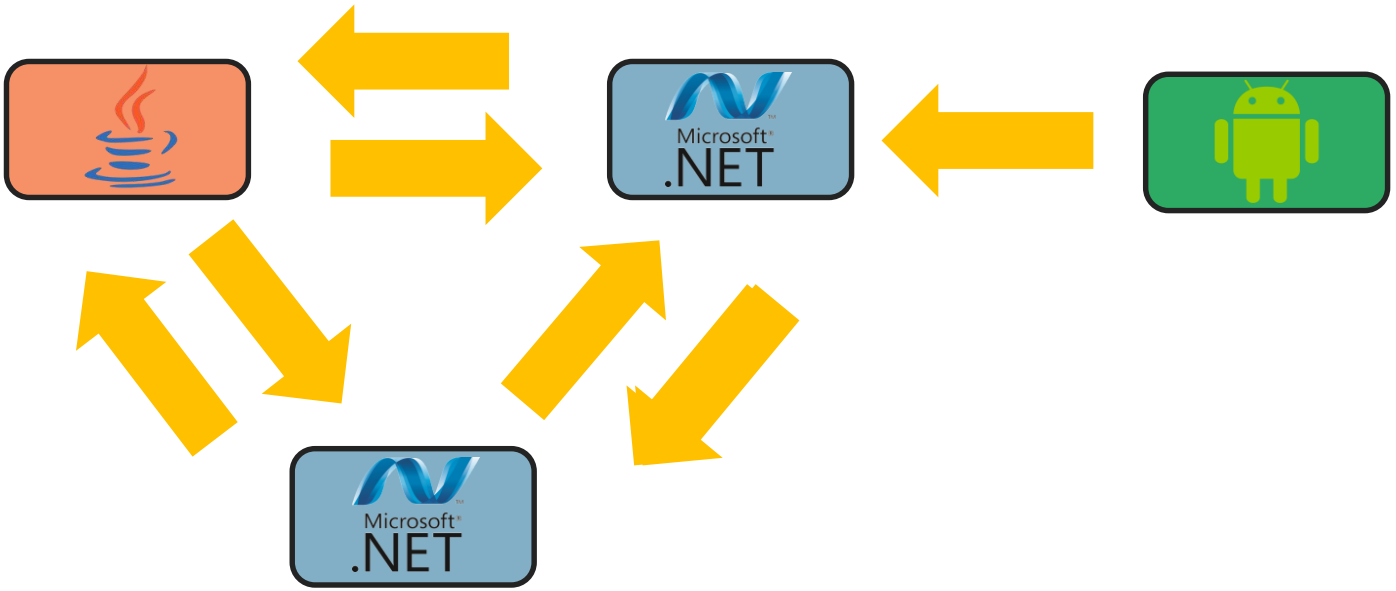
sopra steria

*Réussir la transformation. Ensemble.

Delivering Transformation. Together.

A QUOI SERT UN WEB SERVICE?

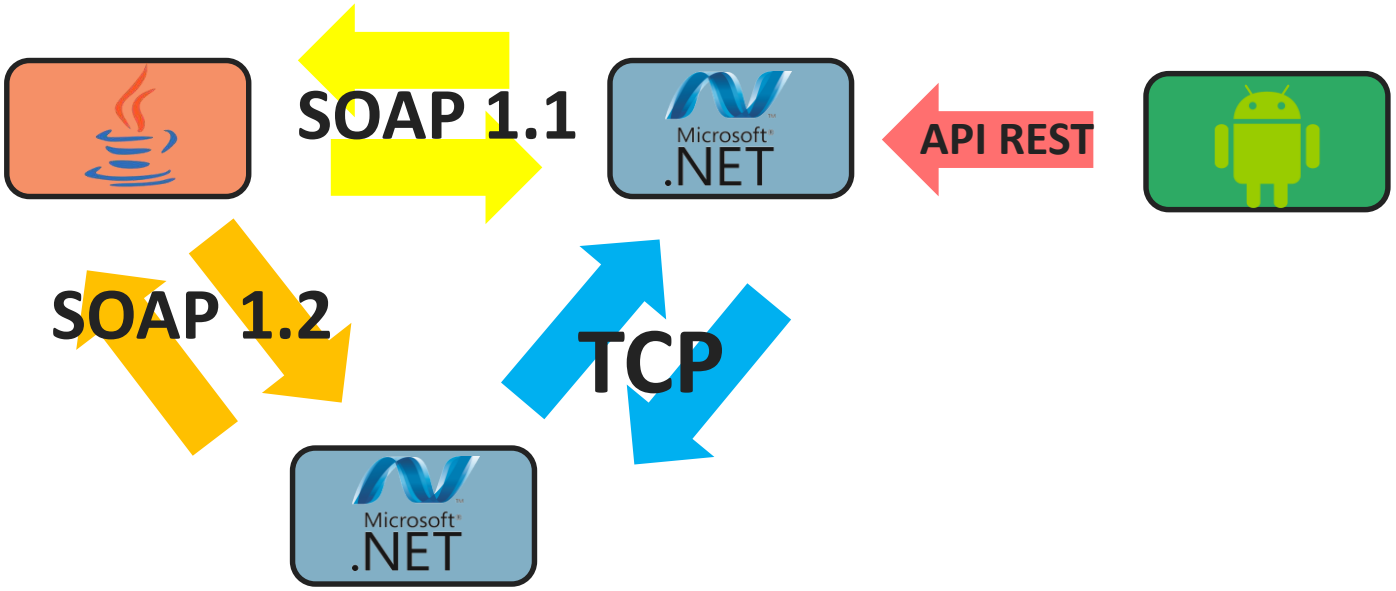
- Permettre à des applications de technologies homogènes ou hétérogènes de pouvoir communiquer entre elles.



- 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 définit 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>
```


while (again) {
iN = -1;
again = false;
getline(cin, sInput);
system("cls");
stringstream(sInput) >> dblTemp;
iLength = sInput.length();
if (iLength < 4) {
again = true;
continue;
} else if (sInput[iLength - 3] != '.') {
again = true;
continue;

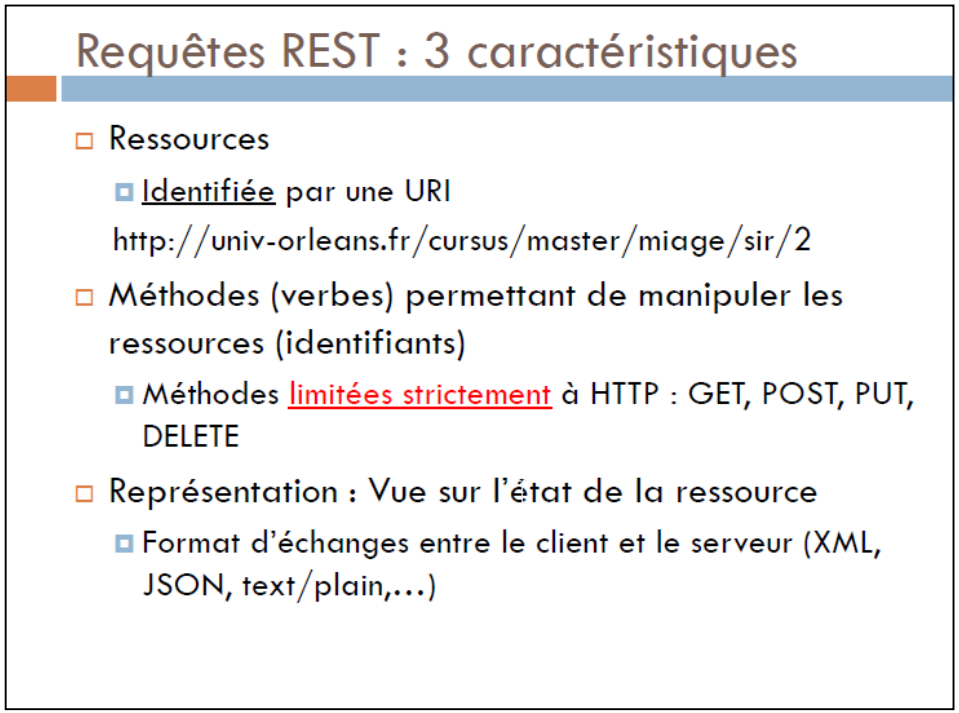
DEMO

MIAGE – INITIATION AU .NET

Focus sur ASP.NET Web API 2

ASP.NET WEB API 2 : WHAT IS IT?

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



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

UN EXEMPLE DE CONTROLLER

```
public class ValuesController : ApiController
{
    // GET api/values
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }
    // GET api/values/5
    public string Get(int id)
    {
        return "value";
    }
    // POST api/values
    public void Post([FromBody]string value)
    {
    }
    // PUT api/values/5
    public void Put(int id, [FromBody]string value)
    {
    }
    // DELETE api/values/5
    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.).

- Table de routage dans~/App_Start/WebApiConfig.cs

```
config.Routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional }
);
```

  - La route est ajoutée dans MapHttpRoute()
  - "api" est un segment de chemin littéral
  - éviter des collisions avec le routage d'ASP.NET MVC
  - {controller} et {id} sont des variables référencées,
  - Permettent de définir respectivement le contrôleur et la valeur qui sera utilisée comme identifiant

SYSTÈME DE ROUTAGE POUR WEB API

Méthode HTTP	Chemin URI	Action	Description
Get	api/values	Get()	Retourne toutes les données comme un IEnumerable.
Get	api/values/2	Get()	Retourne l'élément ayant pour Id 2.
Post	api/values	Post()	Enregistrement d'un nouvel élément.
Put	api/values/2	Put()	Modification de l'élément ayant pour Id 2.
Delete	api/values/2	Delete()	Surpression de l'élément ayant pour Id 2.

UN EXEMPLE PLUS POUSSÉ

```
public class CustomerController : ApiController
{
    Customer[] Customers = new Customer[]
    {
        new Customer { Id = 1, FirstName = "Hinault", LastName = "Romaric", EMail = "hr@gmail.com"},
        new Customer { Id = 2, FirstName = "Thomas", LastName = "Perrin", EMail = "thomas@outlook.com"}
        new Customer { Id = 3, FirstName = "Allan", LastName = "Croft", EMail = "allan.croft@crt.com"},
        new Customer { Id = 3, FirstName = "Sahra", LastName = "Parker", EMail = "sahra@yahoo.com"},
    };
    public IEnumerable<Customer> GetAllCustomers()
    {
        return Customers;
    }
    public Customer GetCustomerById(int id)
    {
        var Customer = Customers.FirstOrDefault((c) => c.Id == id);
        if (Customer == null)
        {
            throw new HttpResponseException(new HttpResponseMessage(HttpStatusCode.NotFound));
        }
        return Customer;
    }
}
```


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"
}
]
```


EXPEMPLE 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>
```



while (again) {
iN = -1;
again = false;
getline(cin, sInput);
system("cls");
stringstream(sInput) >> dblTemp;
iLength = sInput.length();
if (iLength < 4) {
again = true;
continue;
} else if (sInput[iLength - 3] != '.') {
again = true;
continue;

DEMO