



Windows Communication Foundation (WCF)

Cours basé sur la technologie WCF du Framework .NET dans le cadre de la création d'un service

Sommaire

- Présentation générale de WCF et des services
- Notion de base WCF
- Etude des formats et des protocoles
- Procédure de création d'un service
- Consomme son service
- Tester son service

Présentation générale de WCF

- WCF permet la mise en place des applications distribuées
 - « Une application distribuée est une application dont tous les éléments qui la composent (classes, persistance, métier, etc.) sont distants et indépendants »
 - Permet la réutilisation de certains composants dans plusieurs applications
 - Invisible pour l'utilisateur final, l'IHM fédère tous les éléments distants
- WCF est basé sur une architecture orientée services (SOA : Service Oriented Architecture)
- WCF peut aussi être utilisé pour créer une architecture orientée ressources (ROA : Resource Oriented Architecture)

Principes de l'architecture SOA

- Le principe du SOA repose sur 3 axes :
 - La définition du service : Le but d'un service est de fournir un besoin bien défini. Les détails d'un service sont contenus dans un document qui sert de « contrat » entre le client et le serveur (la manière dont il doit être appelé, les paramètres, etc.)
 - Les services sont autonomes : Un service implémente ses propres composants et des propres méthodes. On doit pouvoir le déplacer ou le remplacer sans que cela affecte d'autres services
 - Les clients et les services ne partagent que des contrats : Le contrat est la seule chose que le serveur partage avec le client. Le client ne connaît pas comment procède le service. Le service et le client ne doivent pas partager de code.

Principe de l'architecture ROA

- Le principe du ROA repose sur les mêmes axes que le SOA à quelques exceptions près :
 - Des URL « riches » : L'url d'un service orienté architecture se veut riche c'est-à-dire détaillé. En principe, ces URL permettent de gérer les opérations du CRUD (exemple d'url : /books/6/comments/87 pour obtenir le commentaire 87 appartenant au livre ayant l'id 6)
 - RestFul : un service orienté ressource utilise forcément le protocole REST.
 - Orienté nouvelle technologie : Internet est massivement orienté ressource car plus simple et plus flexible

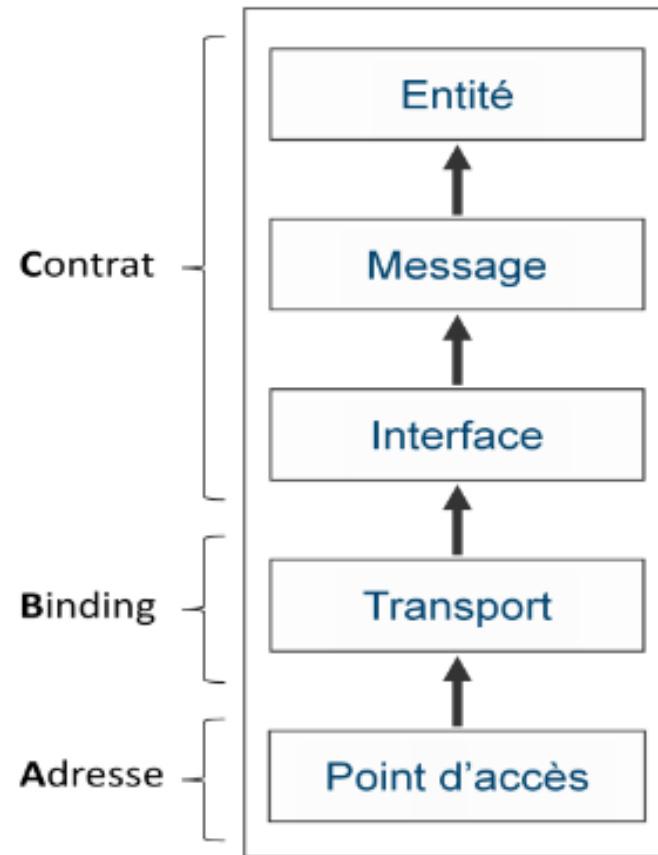
Avantage WCF

- WCF est un modèle de distribution de services avantageux pour les raisons suivantes :
 - Pas ou peu de couplage
 - Souplesse
 - Disponibilité
 - Interopérabilité
 - Sécurité
- A ce jour, les architectures orientées ressources sont plébiscitées par la communauté et par les entreprises (on parle alors de services web RESTful)

Notion de base WCF - ABC

- On peut résumer la structure WCF aux éléments A B et C pour :
 - Une Adresse : adresse à laquelle le client doit se connecter pour utiliser le service
 - Un Binding : protocole à utiliser par le client pour communiquer avec le service
 - Un Contrat : informations échangées entre le serveur et le client afin que ce dernier sache comment utiliser le service

Notion de base WCF - ABC



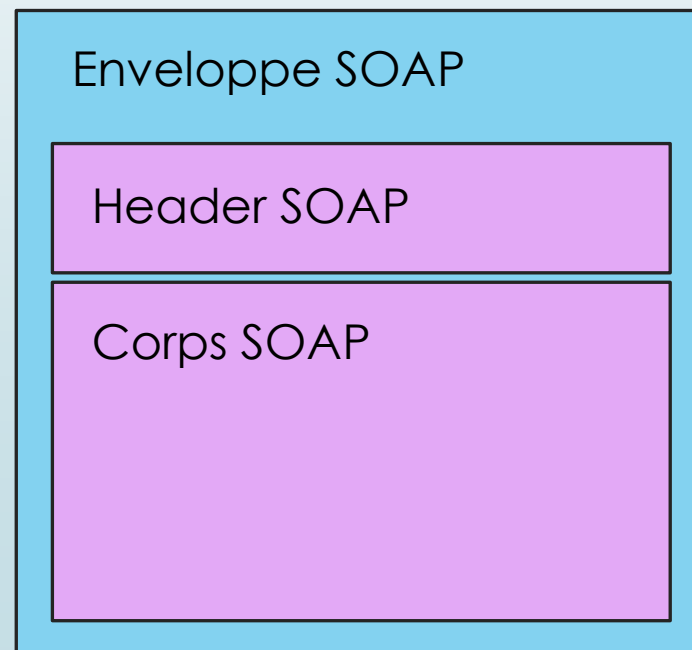
- On peut schématiser le principe ABC sous cette forme
- Un service peut avoir plusieurs points d'accès (endpoint)

Notion de base WCF – espace de nom

- La mise en œuvre de WCF passe par l'utilisation de deux espaces de nom du Framework .NET :
 - `System.ServiceModel`
 - `System.Runtime.Serialization`

Le protocole SOAP (Binding)

- SOAP = Simple Object Access Protocol
- C'est un protocole de transmission de message orienté SOA. Le transfert est le plus souvent réalisé à l'aide du protocole HTTP
- Il est bâti sur du XML
- Il est constitué de plusieurs couches mais respecte l'architecture suivante :



Le protocole SOAP

Demande

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">http://tempuri.c
  </s:Header>
  <s:Body>
    <GetNombreLettreMot xmlns="http://tempuri.org/">
      <mot>enveloppe</mot>
    </GetNombreLettreMot>
  </s:Body>
</s:Envelope>
```

Réponse

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <GetNombreLettreMotResponse xmlns="http://tempuri.org/">
      <GetNombreLettreMotResult>Le mot enveloppe contient 9 lettre(s)</GetNombreLettreMotResult>
    </GetNombreLettreMotResponse>
  </s:Body>
</s:Envelope>
```

- Une demande est formulée
- Une réponse est envoyée
- Architecture Enveloppe – Header – Body

Avantages et inconvénients de SOAP

► Avantages de SOAP :

- Indépendant du langage et de la plateforme, on retrouve du SOAP en C# / C++ / Python / JAVA / etc.
- Pas de problème de pare-feu et / ou de proxy
- Communication facilitée

► Inconvénients de SOAP :

- Le format XML étant assez verbeux, les échanges peuvent devenir assez conséquents
- Couplage fort entre les clients et le serveur

Le protocole REST (Binding)

- REST = Representational State Transfer
- C'est un protocole de transmission de message orienté ROA. Le transfert est le plus souvent réalisé à l'aide du protocole HTTP
- Il est basé sur les 4 verbes HTTP : GET, POST, PUT et DELETE pour réaliser les 4 actions du CRUD (Create, Retrieve, Update et Delete)
- Le format est le plus souvent du JSON mais peut aussi être du XML

Avantages de REST

- Avantages de REST :
 - Indépendant du langage et de la plateforme, on retrouve du REST en C# / C++ / Python / JAVA / etc.
 - Pas de problème de pare-feu et / ou de proxy
 - Plus simple et plus lisible que SOAP. Implémentation plus aisée
 - Performance accrue
 - Utilisation de nombreux formats (JSON, XML, HTML, etc.)
 - Proche de la conception et de la philosophie web (URI, GET, POST, PUT, etc.)
 - Sécurisation sous forme de token
 - Etc.

Le format WSDL

- WSDL pour Web Services Description Language permet de décrire de manière standard un service web (version 2.0)
- Prononcé « Whiz-Deul »
- Le WSDL décrit l'interface publique d'accès à un service web (notamment dans le cadre des architectures de type SOA)
- Le WSDL est écrit en XML
- Il se génère automatiquement en appelant l'argument ?wsdl (notion vue plus tard lors de l'étude des tests applicables à un service web)

Le format WSDL - exemple

- Le WSDL est un document XML qui comment par la balise <definitions>
 - <binding> définit le protocole à utiliser pour invoquer le service web
 - <service> décrit l'ensemble des points finaux (endpoints) du réseau
 - <port> spécifie l'emplacement du service

```
- <wsdl:definitions name="ServicePetitRobert" targetNamespace="http://tempuri.org/">
  + <wsdl:types></wsdl:types>
  + <wsdl:message name="IServicePetitRobert_GetNombreLettreMot_InputMessage"></wsdl:message>
  + <wsdl:message name="IServicePetitRobert_GetNombreLettreMot_OutputMessage"></wsdl:message>
  + <wsdl:message name="IServicePetitRobert_GetInformationMot_InputMessage"></wsdl:message>
  + <wsdl:message name="IServicePetitRobert_GetInformationMot_OutputMessage"></wsdl:message>
  + <wsdl:portType name="IServicePetitRobert"></wsdl:portType>
  - <wsdl:binding name="BasicHttpBinding_IServicePetitRobert" type="tns:IServicePetitRobert">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    + <wsdl:operation name="GetNombreLettreMot"></wsdl:operation>
    + <wsdl:operation name="GetInformationMot"></wsdl:operation>
    </wsdl:binding>
  - <wsdl:service name="ServicePetitRobert">
    + <wsdl:port name="BasicHttpBinding_IServicePetitRobert" binding="tns:BasicHttpBinding_IServicePetitRobert"></wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```


Créer son service - procédure

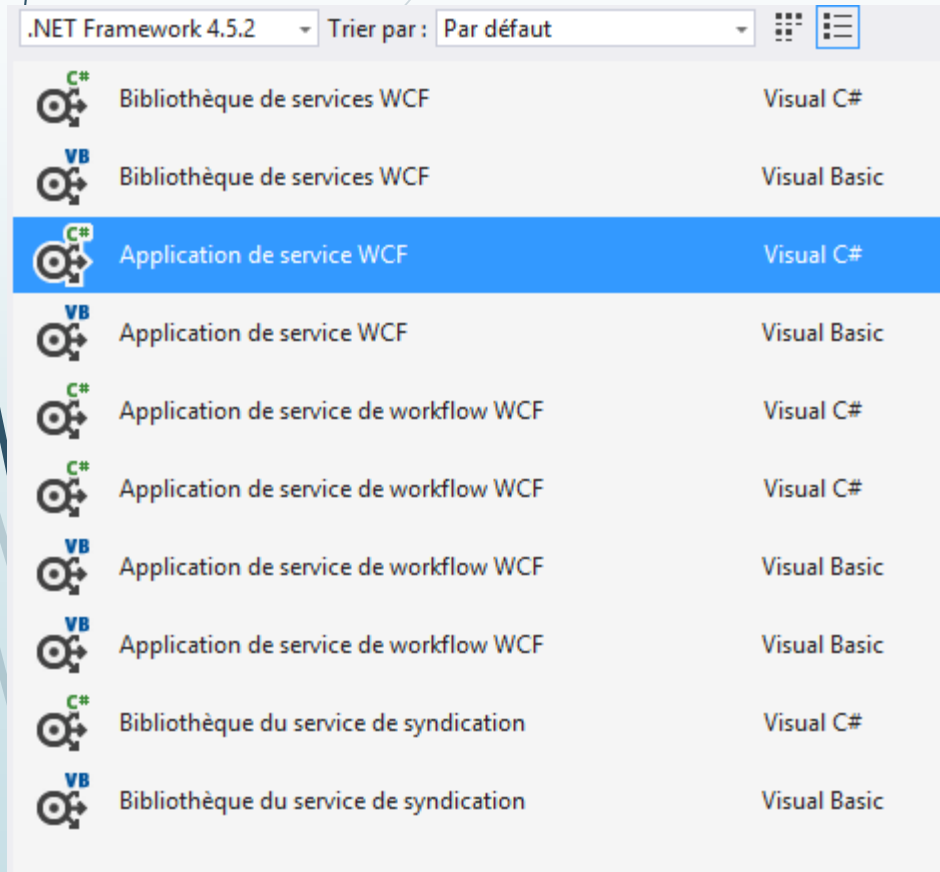
- Créer un service WCF se résume aux 3 étapes suivantes :
 - Définir le contrat, définir les signatures des méthodes et définir le format des données à échanger
 - Implémentation du contrat et implémentation du service
 - Configuration et accessibilité du service
- Une fois le service WCF créé, on peut le consommer à partir d'une application console, web, etc.

TD : créer un service WCF

► Objectifs :

- Créer un web service WCF permettant de remplacer un dictionnaire. Il contiendra des méthodes permettant de retourner le nombre de lettre d'un mot ou encore de retourner la définition d'un mot
- Il doit être accessible en SOAP et en REST
- Le format REST sera du JSON (en lieu et place du XML)
- Tester son service en utilisant la console fournie à cet effet
- Créer quelques tests unitaires pour tester le service

Créer un projet WCF



- Visual Studio permet de générer l'ossature d'un projet WCF c'est-à-dire :
 - Un contrat type
 - Un service type
 - Un fichier de configuration

Créer le contrat

```
[ServiceContract]
1 référence
public interface IServicePetitRobert
{
    // on retourne un simple "message" sans utiliser de datacontract
    [OperationContract]
    1 référence
    string GetNombreLettreMot(string mot);

    // on retourne un datacontract défini
    [OperationContract]
    1 référence
    MotDuDictionnaire GetInformationMot(string mot);
}
```

- Le contrat est une interface décorée par l'annotation **[ServiceContract]**
- Le contrat contient deux opérations (ou actions) définies par l'annotation **[OperationContract]**
- Une des opérations retourne un objet défini par un **DataContract** qu'il faudra créer

Créer le DataContract

```
[DataContract]
5 références
public class MotDuDictionnaire
{
    1 référence
    public MotDuDictionnaire(string mot)
    {
        Mot = mot;
        NbLettre = mot.Count();
        PremiereLettre = mot.Substring(0, 1).ToString();
    }

    [DataMember]
    1 référence
    public string Mot { get; set; }

    [DataMember]
    1 référence
    public string Definition { get; set; }

    [DataMember]
    1 référence
    public int NbLettre { get; set; }

    [DataMember]
    1 référence
    public string PremiereLettre { get; set; }
}
```

- La classe **MotDuDictionnaire** est décorée par l'annotation **DataContract**.
- Sans ce mot clé, l'objet ne peut pas être utilisé dans une opération définie dans le contrat
- Les propriétés de cette classe sont décorées par l'annotation **DataMember**
- Les propriétés non décorées par **DataMember** ne seront pas exposées (propriétés privées servant pour du calcul par exemple)

Créer le service

- Le service est auto-généré par Visual Studio lors de la création d'un nouveau projet WCF (il génère aussi un fichier de configuration adéquat ainsi qu'un contrat)
- Extension du service au format .svc
- Il implémentera le contrat défini au préalable
- Il utilisera la classe de type DataContract définie au préalable
- Attention au **BALISAGE** (clic droit sur le service, afficher le balisage)

```
<%@ ServiceHost Language="C#" Debug="true" Service="Cours3.ServicePetitRobert" CodeBehind="ServicePetitRobert.svc.cs" %>
```

Nom du service

Nom du fichier cs du code
behind

Créer le service

```
public class ServicePetitRobert : IServicePetitRobert
{
    1 référence
    public string GetNombreLettreMot(string mot)
    {
        return string.Format("Le mot {0} contient {1} lettre(s)", mot, mot.Count());
    }

    1 référence
    public MotDuDictionnaire GetInformationMot(string mot)
    {
        if (String.IsNullOrEmpty(mot))
        {
            throw new ArgumentNullException("mot");
        }

        MotDuDictionnaire motDuDictionnaire = new MotDuDictionnaire(mot);
        motDuDictionnaire.Definition = "Definition à aller chercher en base de ce mot";

        return motDuDictionnaire;
    }
}
```

- Implémentation de l'interface
- Gestion des erreurs
- Défini le « corps » de chaque opération du contrat
- Ne tient pas compte du protocole ou du format

Configurer son service en SOAP

- Par défaut, Visual Studio configure les services en SOAP
- La configuration se réalise dans le fichier web.config
- On reprends la notion d'ABC détaillée au préalable pour configurer un point d'accès à notre service en SOAP

Configuration HTTP:

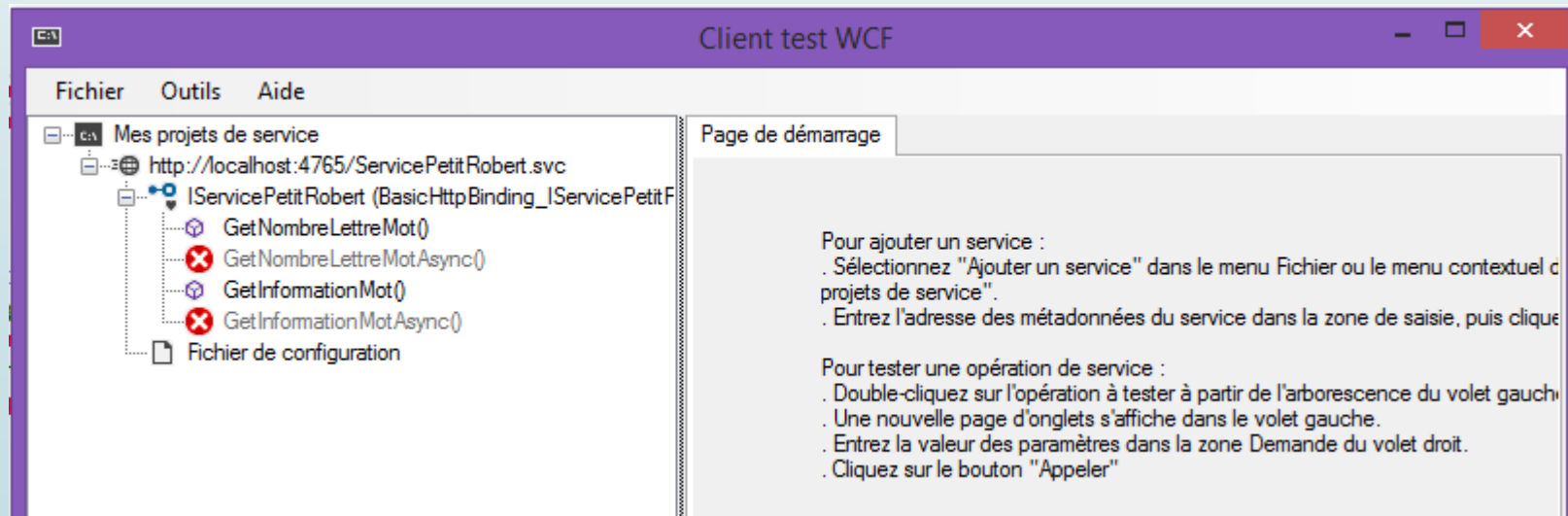
```
<protocolMapping>  
  <add binding="basicHttpsBinding" scheme="https" />  
</protocolMapping>
```

Configuration SOAP :

```
<services>  
  <service behaviorConfiguration="mybehavior" name="Cours3.ServicePetitRobert">  
    <endpoint address="soap" contract="Cours3.IServicePetitRobert" binding="basicHttpBinding"/>  
  </service>  
</services>
```

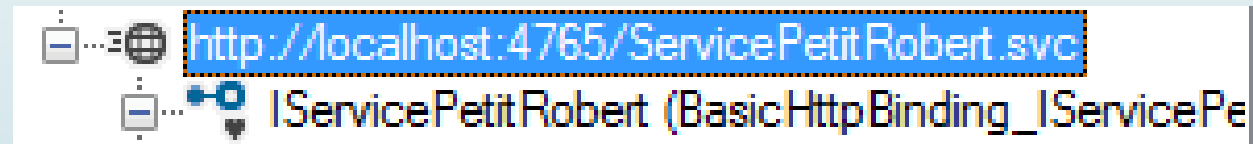

Tester son service en SOAP

- Visual Studio offre la possibilité de tester les services dans une petite console « **client test WCF** »
- Se lance comme un débbugger
- Se charge des opérations pour parser la réponse au format XML
- Permet d'inspecter les messages SOAP avec l'enveloppe, le header et le body



Accéder au WSDL

- Le WSDL est accessible en utilisant l'adresse du service et en rajoutant ?wsdl (voir diapositive sur cette notion)
- L'adresse du serveur est affichée dans l'outil vu précédemment
- Mode DEBUG = hébergé sur localhost



WSDL = `http://localhost:4765/ServicePetitRobert.svc?wsdl`

Tester son service en SOAP

GetNombreLettreMot

Demande		
Nom	Valeur	Type
mot	Chapeau	System.String

Réponse ☐ Démarrer un nouveau proxy

Nom	Valeur	Type
(return)	"Le mot Chapeau contient 7 lettre	System.String

Mis en forme XML

- Paramètre d'entrée : type string
- On envoie un message (une demande) à l'opération **GetNombreLettreMot**
- On récupère le résultat (type string dans ce cas là)

Tester son service en SOAP

GetInformationMot

Demande		
Nom	Valeur	Type
mot	Chapeau	System.String

Réponse ☐ Démarrer un nouveau proxy Appeler

Nom	Valeur	Type
└ (return)		Cours3.MotDuDictionnaire
Definition	"Definition à aller chercher en bas	System.String
Mot	"Chapeau"	System.String
NbLettre	7	System.Int32
PremiereLettre	"C"	System.String

Mis en forme XML

- Paramètre d'entrée : type string
- On envoie un message (une demande) à l'opération **GetInformationMot**
- La réponse est un objet « composite » décoré avec un **DataContract**. On peut accéder à toutes ses propriétés déclarées par un **DataMember**

Configurer son service en REST

- La configuration se réalise dans le fichier web?config au même titre qu'une configuration SOAP
- Il faut configurer un point d'accès de type REST en plus du point d'accès de type SOAP

Configuration SOAP + REST (deux endpoints) :

```
<services>
  <service behaviorConfiguration="mybehavior" name="Cours3.ServicePetitRobert">
    <endpoint address="rest" binding="webHttpBinding" behaviorConfiguration="web" contract="Cours3.IServicePetitRobert"/>
    <endpoint address="soap" contract="Cours3.IServicePetitRobert" binding="basicHttpBinding"/>
  </service>
</services>
```

```
<endpointBehaviors>
  <behavior name="web">
    <webHttp />
  </behavior>
</endpointBehaviors>
```

Déclaration d'un behavior pour le REST :

Définir les ressources

- Pour utiliser un service en REST, il faut ajouter des annotations dans les opérations du contrat pour définir la ressource

```
[ServiceContract]
1 référence
public interface IServicePetitRobert
{
    // on retourne un simple "message" sans utiliser de datacontract
    [OperationContract]
    [WebInvoke(Method = "GET",
        ResponseFormat = WebMessageFormat.Json,
        RequestFormat = WebMessageFormat.Json,
        UriTemplate = "nombreLettre?mot={mot}")]
    1 référence
    string GetNombreLettreMot(string mot);

    // on retourne un datacontract défini
    [OperationContract]
    [WebInvoke(Method = "GET",
        ResponseFormat = WebMessageFormat.Xml,
        RequestFormat = WebMessageFormat.Json,
        UriTemplate = "information?mot={mot}")]
    1 référence
    MotDuDictionnaire GetInformationMot(string mot);
}
```

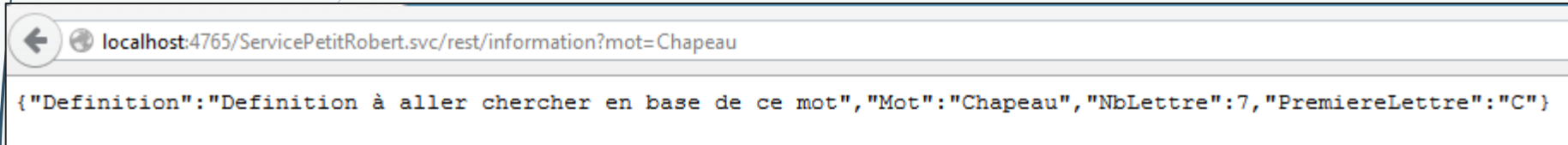
Définir les ressource

- Une ressource se définit donc à l'aide de l'annotation **WebInvoke**
 - Une méthode (GET / POST / PUT / DELETE)
 - Le format de la demande (XML ou JSON)
 - Le format de la réponse (XML ou JSON)
 - L'URI c'est-à-dire le chemin pour accéder à cette ressource
- Exemple :
 - **/rest/nombreLettre?mot=enveloppe** permet d'appeler l'opération GetNombreLettreMot avec le paramètre « enveloppe »

Tester son service en REST

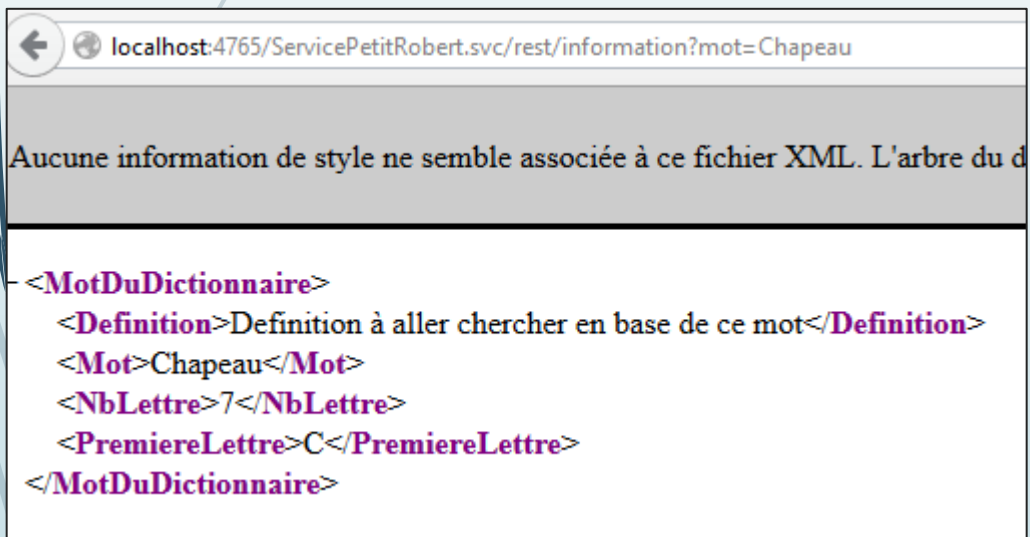
- Le résultat du REST doit évidemment être le même que celui du POST
- En fonction de format de la réponse choisi, le service renverra soit une structure XML (comme le SOAP mais en plus simple), soit une structure JSON
- Les tests se font dans n'importe quel navigateur en tapant l'url de la ressource que l'on souhaite obtenir

Tester son service en REST



localhost:4765/ServicePetitRobert.svc/rest/information?mot=Chapeau

```
{"Definition": "Definition à aller chercher en base de ce mot", "Mot": "Chapeau", "NbLettre": 7, "PremiereLettre": "C"}
```



localhost:4765/ServicePetitRobert.svc/rest/information?mot=Chapeau

Aucune information de style ne semble associée à ce fichier XML. L'arbre du d

```
<MotDuDictionnaire>  
  <Definition>Definition à aller chercher en base de ce mot</Definition>  
  <Mot>Chapeau</Mot>  
  <NbLettre>7</NbLettre>  
  <PremiereLettre>C</PremiereLettre>  
</MotDuDictionnaire>
```

- Le résultat est bien le même que ce soit en XML ou en JSON, seul la structure change
- Le résultat est le même qu'en SOAP
- **Attention**, tout n'est pas testable via un navigateur (par exemple les requêtes en POST)
- Pour automatiser et rendre les tests plus pertinents, il faut passer par **des tests unitaires**

Tests unitaires

- Possibilité de tester le service à l'aide de tests unitaires.
- Vérifier le type de données en retour (XML / JSON).
- Valider la cohérence du schéma.
- Etc.