economie
SPF Economie, P.M.E., Classes moyennes et Energie

SPF Economie, P.M.E., Classes moyennes et Energie

Client/WCF BCE Public Search Web service

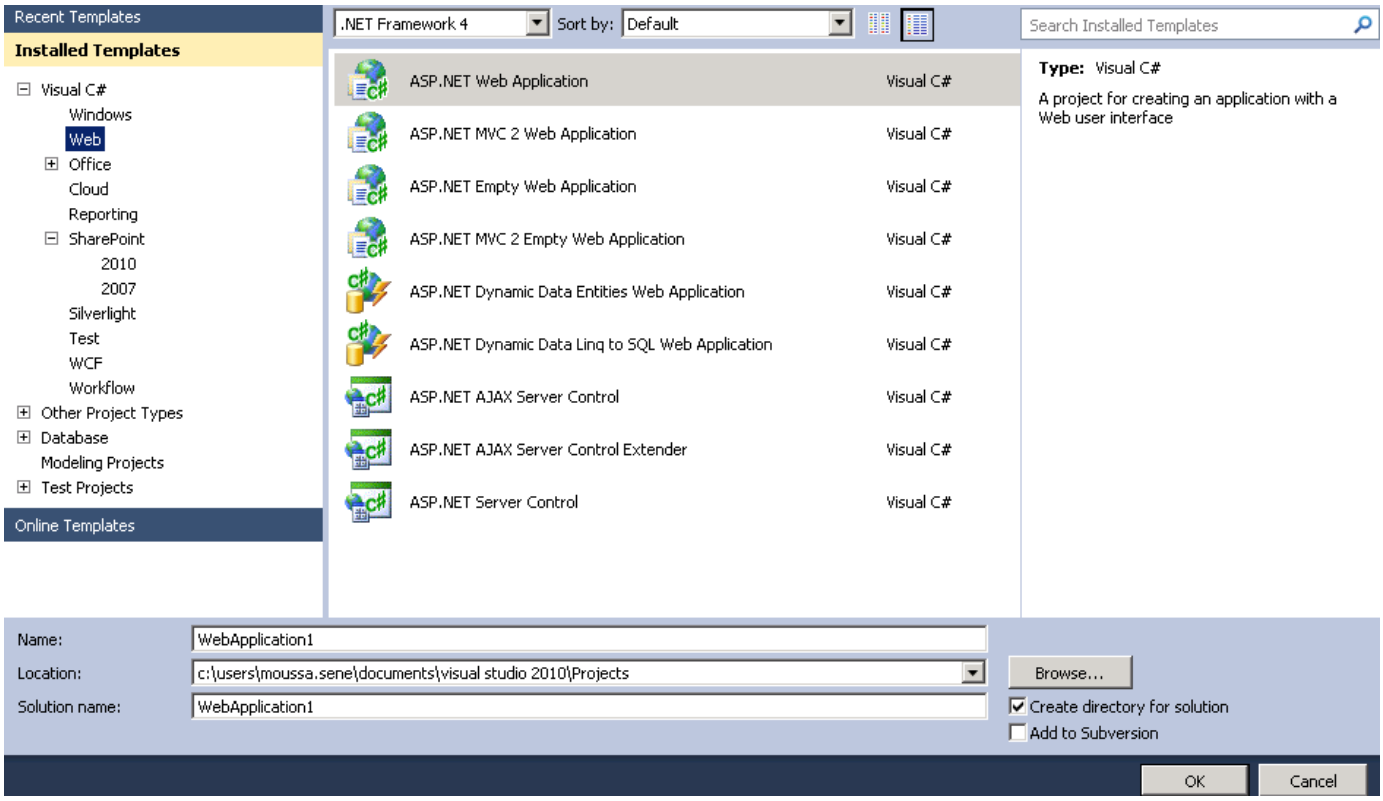
Documentation Technique

2015

A travers ce document nous expliquerons les étapes à suivre pour se connecter au BCE Public Search Web service en utilisant un client WCF (Windows Communication Fondation).

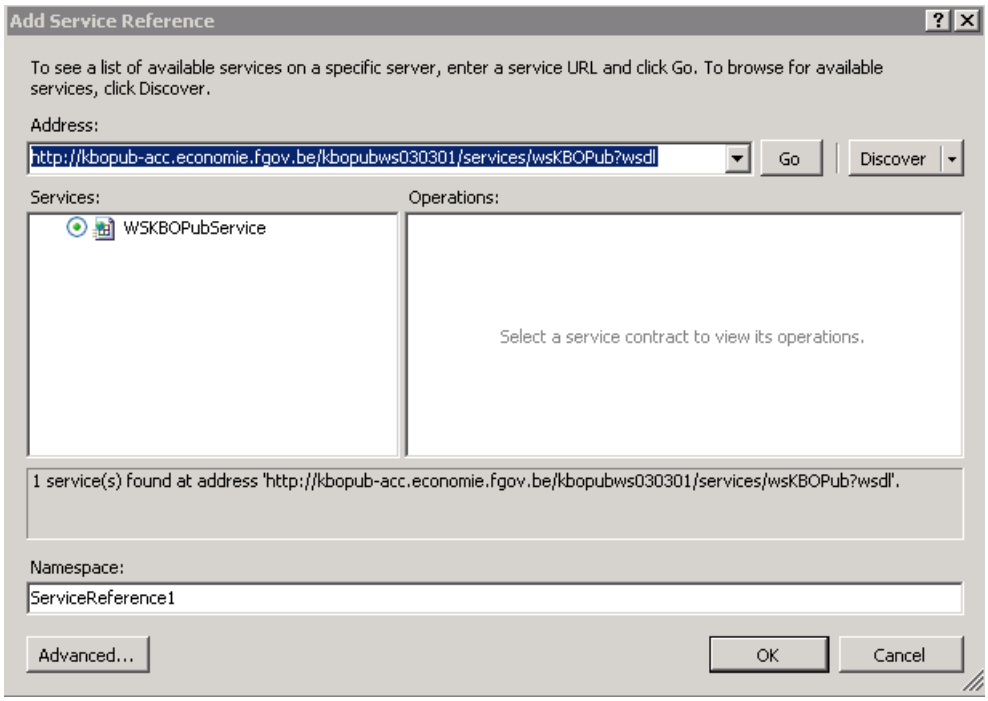
Le web service utilise l'UserName Token du WS-Security pour authentification.

Création du projet client/WCF dans Visual studio 2010 :



Dans la fenêtre solution du nouveau projet on fait clique-droit sur **References** puis **Add Service Reference.**

On saisit l'url du WSDL dans « Address : » puis on clique sur **GO** et ensuite sur **OK.**



Password Digest et Digest Nonce ne sont pas par défaut configurable dans WFC. Ainsi il faut créer un Custom ClientCredentials.

Création du Custom ClientCredentials :

Pour mettre en place ce Custom ClientCredentials on implémentera des custom classes qui hériterons des trois suivantes:

- ClientCredentials

```
public class CustomCredentials : ClientCredentials
{
    public CustomCredentials()
    { }

    protected CustomCredentials(CustomCredentials cc)
        : base(cc)
    { }

    public override System.IdentityModel.Selectors.SecurityTokenManager CreateSecurityTokenManag
    {
        return new CustomSecurityTokenManager(this);
    }

    protected override ClientCredentials CloneCore()
    {
        return new CustomCredentials(this);
    }
}
```

- ClientCredentialsSecurityTokenManager

```
public class CustomSecurityTokenManager : ClientCredentialsSecurityTokenManager
{
    public CustomSecurityTokenManager(CustomCredentials cred)
        : base(cred)
    { }

    public override System.IdentityModel.Selectors.SecurityTokenSerializer
        CreateSecurityTokenSerializer(System.IdentityModel.Selectors.SecurityTokenVersion version)
    {
        return new CustomTokenSerializer(System.ServiceModel.Security.SecurityVersion.WSSecurity11
    }
}
```

- WSSecurityTokenizer

```
public class CustomTokenSerializer : WSSecurityTokenSerializer
{
    public CustomTokenSerializer(SecurityVersion sv)
        : base(sv)
    { }

    protected override void WriteTokenCore(System.Xml.XmlWriter writer, System.IdentityModel.Tokens.SecurityToken toker

    protected string GetSHA1String(string phrase)...


    public string CreatePasswordDigest(byte[] nonce, string createdTime, string password)...
}
```

Utilisation du Custom ClientCredentials :

La méthode suivante nous permet de créer un objet client-proxy (WSKBOPubClient)

```
public static WSKBOPubClient CreateRealTimeOnlineProxy(string url, string username, string password)
{
    if (string.IsNullOrEmpty(url))
        url = "https://kbopub-acc.economie.fgov.be/kbopubws030301/services/wsKBOPub";

    CustomBinding binding = new CustomBinding();
    var security = TransportSecurityBindingElement.CreateUserNameOverTransportBindingElement();
    security.IncludeTimestamp = false;
    security.DefaultAlgorithmSuite = SecurityAlgorithmSuite.Basic256;
    security.MessageSecurityVersion = MessageSecurityVersion.WSSecurity10WSTrustFebruary2005WSSecureConversationFebruary2005WSSecuri
    var encoding = new TextMessageEncodingBindingElement();
    encoding.MessageVersion = MessageVersion.Soap11;
    var transport = new HttpsTransportBindingElement();
    transport.MaxReceivedMessageSize = 20000000; // 20 megs

    binding.Elements.Add(security);
    binding.Elements.Add(encoding);
    binding.Elements.Add(transport);

    WSKBOPubClient client = new WSKBOPubClient(binding, new EndpointAddress(url));

    client.ChannelFactory.Endpoint.Behaviors.Remove<System.ServiceModel.Description.ClientCredentials>();
    client.ChannelFactory.Endpoint.Behaviors.Add(new CustomCredentials());

    client.ClientCredentials.UserName.UserName = username;
    client.ClientCredentials.UserName.Password = password;

    return client;
}
```

Ainsi l'objet créé est utilisé dans l'évenement Button1_Click :

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        EnterpriseName.Text = "";
        EnterpriseActivity.Text = "";
        //
        WSKBOPubClient client = CreateRealTimeOnlineProxy("", UserName.Text, Password.Text);

        ReadEnterpriseReplyType reply = new ReadEnterpriseReplyType();
        ReadEnterpriseRequestType request = new ReadEnterpriseRequestType();
        RequestContextType requestContext = new RequestContextType();
        ReplyContextType replyContext = new ReplyContextType();

        requestContext.Id = "ABCD";
        requestContext.Language = new RequestContextTypeLanguage[] { RequestContextTypeLanguage.nl };

        request.EnterpriseNumber = Convert.ToInt64(EnterpriseNumber.Text);

        client.ReadEnterprise(requestContext, request, out reply);

        if(reply.Enterprise.Denomination[0].Description[0].Value != null)
            EnterpriseName.Text = reply.Enterprise.Denomination[0].Description[0].Value.ToString();

        //if (reply.Enterprise.Activity[0] != null && reply.Enterprise.Activity[0].Description[0] != null)
        //    EnterpriseActivity.Text = reply.Enterprise.Activity[0].Description[0].Value.ToString();

        if (reply.Enterprise.Function[0].NaturalPersonFounder != null)
        {
            EnterprisePersonFounder.Text = reply.Enterprise.Function[0].NaturalPersonFounder.EnterpriseNumber.ToString();
        }
        else
        {
            EnterprisePersonFounder.Text = "Tag NaturalPersonFounder is empty !";
        }

        ErrorMessage.Text = "";
    }
    catch (Exception exception)
    {
        ErrorMessage.Text = exception.Message;
    }
}
```