INRAE

Cati Sicpa

Xamarin

Consommer un webservice Soap

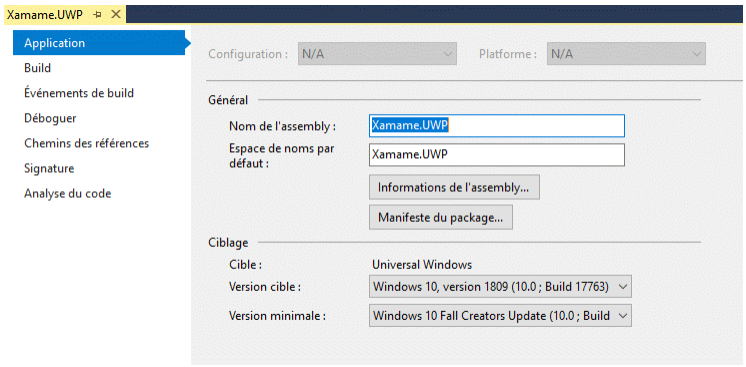
Table des matières

Donner les autorisations

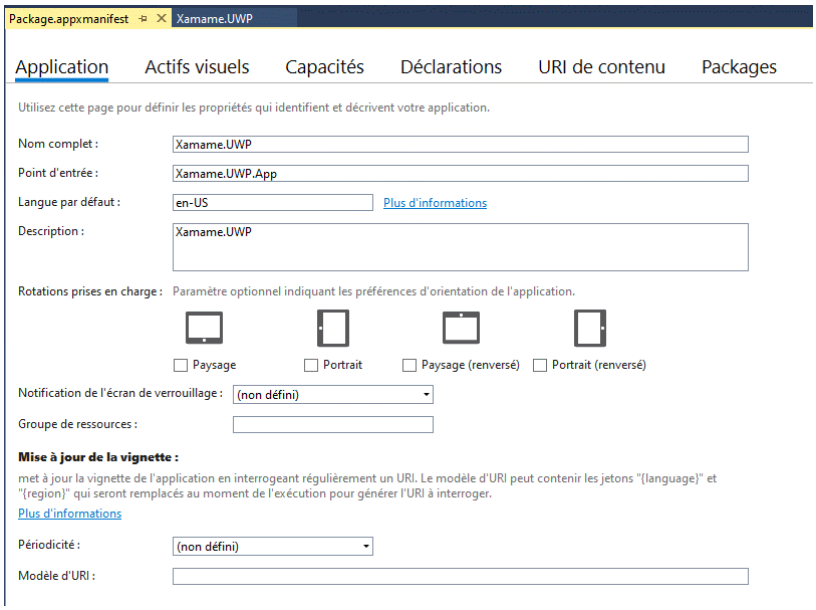
Tout d'abord, il est nécessaire de donner les autorisations aux projets-cibles concernées.

Projet Windows UWP

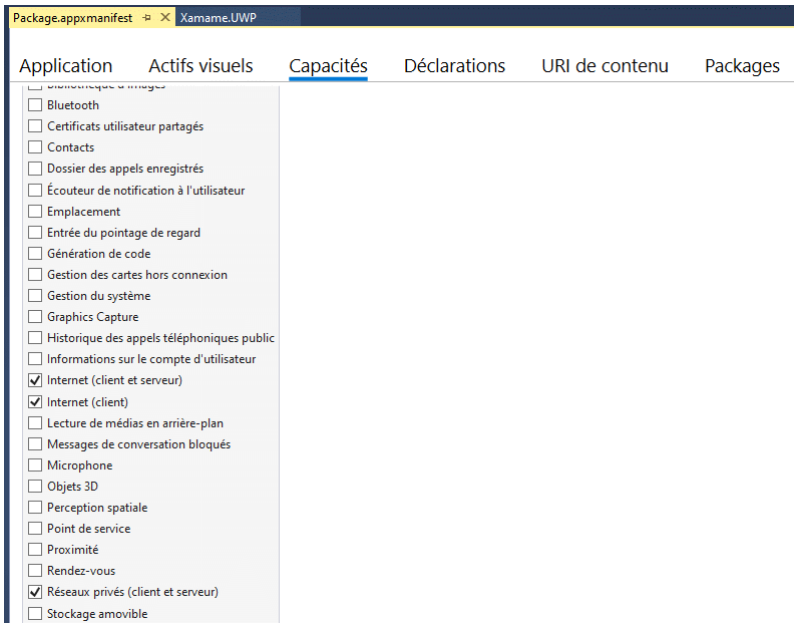
- Sur le projet UWP, faire un clic-droit « Propriété »



- Cliquer sur « Manifeste du package… »



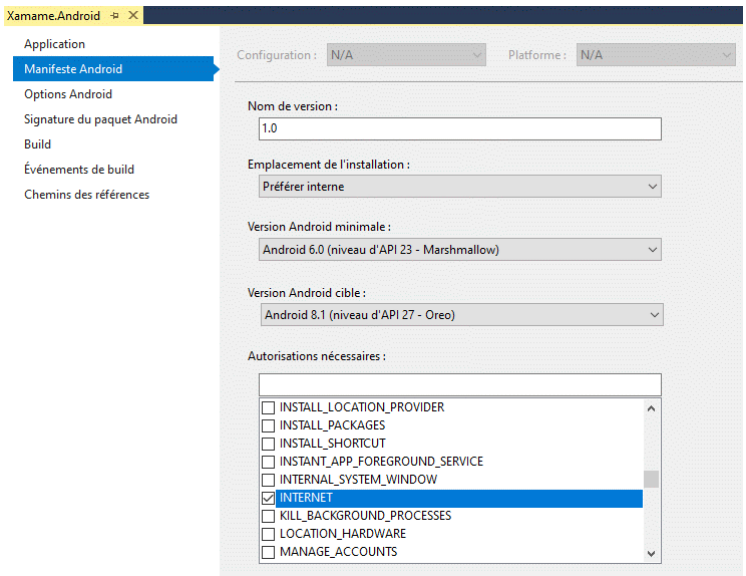
- Cliquer sur l'onglet « Capacités »



- Cocher les cases :
  - Internet (client et serveur)
  - Internet (client)
  - Réseaux privés (client et serveur)

Projet Android

- Sur le projet Android, faire un clic-droit « Propriété »
- Cliquer sur l'onglet « Manifeste Android »
- Cocher les autorisations
  - ACCESS_NETWORK_STATE
  - INTERNET



Créer le code d'appel d'une fonction d'un webservice Soap

Générer le code client d'un web service Soap

Visual Studio permet de générer le code client d'un service Soap en utilisant Microsoft WCF. Mais, le code généré correspond à une consommation asynchrone du webservice. Or cette génération de code n'est pas supportée par MONO (pour info : Les applications Xamarin.Android s'exécutent dans l'environnement d'exécution MONO).
Cf. la discussion : https://github.com/dotnet/wcf/issues/1808
Il faut donc générer le code en générant uniquement les méthodes synchrones. Pour cela, j'ai utilisé l'utilitaire svcutil.exe. Cet utilitaire est disponible sur *C:\Program Files (x86)\Microsoft SDKs\Windows\Vx* en fonction de votre version de Windows.

- Exemple de ligne de commande à lancer :

```
svcutil.exe /syncOnly http://adresse.du.webservice?wsdl
```

- Si le webservice est sur une url sécurisée :
  1. Lancer l'adresse du WS depuis un navigateur
  2. Sauvegarder le résultat dans un fichier nomWebservice.wsdl
  3. Lancer l'adresse du xsd correspondant
  4. Sauvegarder le résultat dans un fichier nomWebservice.xsd
  5. Modifier le fichier wsdl, au niveau de la balise `<xsd:import`, afin de le faire pointer vers le fichier xsd enregistré
  6. Lancer la commande ci-dessous

```
svcutil.exe /syncOnly nomWebservice.wsdl nomWebservice.xsd
```

Dans les 2 cas, cela va générer 2 fichiers :

- NomWebService.cs (qui contient toutes les méthodes correspondantes aux fonctions proposées par le ws)
- Output.config (qui contient l'adresse du endPoint et la configuration du binding)

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="GeedocServiceImplPortBinding">
                    <security mode="Transport" />
                </binding>
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="https://sicpa-interop.inra.fr:443/GeedocWs/GeedocService"
                binding="basicHttpBinding" bindingConfiguration="GeedocServiceImplPortBinding"
                contract="GeedocServiceImpl" name="GeedocServiceImplPort" />
        </client>
    </system.serviceModel>
</configuration>
```

Utilisation de des fichiers générés dans votre code

- Créer un répertoire *ws.nomduwebservice* (namespace) dans votre projet principal
- Copier-coller ces 2 fichiers dans ce répertoire
- Au début du fichier *NomWebService.cs*, écrire le nom du namespace

```
namespace Xamame.ws.GeedocService
{

    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
    [System.ServiceModel.ServiceContractAttribute(Namespace = "http://ws/", ConfigurationName = "GeedocServiceImpl")]
    29 références
    public interface GeedocServiceImpl
    {
```

- Créer une classe statique *WebServicesHelper.cs* dans le répertoire ws
- Dans cette classe statique créer la méthode statique qui permettra de renvoyer l'objet client du webservice

```
public static class WebServicesHelper
{
    public static BasicHttpBinding binding = new BasicHttpBinding(BasicHttpSecurityMode.Transport);

    0 références
    public static GeedocServiceImplClient GeedocServiceImplClient
    {
        get
        {
            //l'adresse du endPoint se trouve dans le fichier output.config
            var endPoint = new EndpointAddress("https://sicpa-interop.inra.fr:443/GeedocWs/GeedocService");
            return new GeedocServiceImplClient(binding, endPoint);
        }
    }
```

Appel d'une fonction contenu dans le webservice

Il suffit d'écrire l'appel de la méthode à partir de l'objet client.

Exemple :

```
String version = WebServicesHelper.GeedocServiceImplClient.getVersion();
```