D314 – Projet simple de Web-service

Sommaire

[Environnement utilisé 1](#_Toc434963231)

[Projet Web-service « OdeWebService» 1](#_Toc434963232)

[Création du projet 1](#_Toc434963233)

[WSDL du Web-service « OdeWebService » 1](#_Toc434963234)

[types 2](#_Toc434963235)

[message 2](#_Toc434963236)

[portType 2](#_Toc434963237)

[binding 2](#_Toc434963238)

[service 2](#_Toc434963239)

[Classe Dimension 2](#_Toc434963240)

[Méthodes exposées 2](#_Toc434963241)

[Methode GetCombinaisons 2](#_Toc434963242)

[Méthodes Metropolis 2](#_Toc434963243)

[Méthodes MaterialisationPartielle 2](#_Toc434963244)

[Consume a Web Service 3](#_Toc434963245)

# Environnement utilisé

* **OS** : Microsoft Windows 7 Pro 64 bits FR
* **IDE** : Oracle Netbeans IDE 8.0.2 for Java EE
* **JDK** : Oracle JDK 1.8.0 u 60
* **Serveur applicatif** : Oracle GlassFish (Fourni avec NetBeans)

# Projet Web-service « OdeWebService»

## Création du projet

Catégorie de projet : Java Web

* Projet : Web application
* Project name : OdeWebService
* Pas de répertoire dédié aux librairies
* Serveur : Glassfish Server 4.1
* Java EE version : Java EE 7 Web
* Pas de framework

## WSDL du Web-service « OdeWebService »

Dans l’élément racine <*definitions*> du WSDL, on retrouve les 5 parties :

### types

Définition des types de données utilisées

### message

Description des noms et les types des données à échanger

### portType

Définition d'un ensemble d'opérations proposées par le Web-service

### binding

Définition du format des messages et du protocole utilisé par chaque type de port.

### service

Collection d’endpoints

### Classe Dimension

string dimensionName; // Nom "officiel" de la dimension - potentiellement plusieurs si "dimensionOrder" >= 2

long dimensionCount; // Nombre de lignes de la dimension

int dimensionMemory; // Taille d'une ligne, en octets

int dimensionOrder; // Indique la dimension du cuboide. Ex : 1 - 1D / 2 : 2D...

## Méthodes exposées

### Methode GetCombinaisons

Fonction récursive

/\* WEB-SERVICE CALL : OdeWebService.GetCombinaisons \*/

GetCombinaisons(listDim1D, 0, "", 0, listCuboides, index\_cuboides, prefix\_index);

|  |  |  |
| --- | --- | --- |
| ***Paramètre*** | ***Type*** | ***Commentaires*** |
| listDim1D <IN> | List<Dimension>() | Rempli |
| 0 | int profCourante | Constante |
| "" | String prefix | Constante |
| 0 | int rang | Constante |
| listCuboides <IN><OUT> | List<Dimension>() | liste des cuboïdes > VIDE |
| index\_cuboides <IN><OUT> | List<String>() | liste des index des cuboïdes > VIDE |
| prefix\_index <IN><OUT> | String | Constante "" |

### Méthodes Metropolis

/\* WEB-SERVICE CALL : OdeWebService.Metropolis \*/

Metropolis(listCuboides, seuil\_poids, nb\_boucle, solution);

|  |  |  |
| --- | --- | --- |
| ***Paramètre*** | ***Type*** | ***Commentaires*** |
| listCuboides <IN> | List<Dimension>() | Rempli |
| seuil\_poids <IN> | double seuil\_poids | Paramètre |
| nb\_boucle <IN> | int nb\_boucle | Paramètre |
| solution <OUT> | new int[listCuboides.Count] | Liste d’integers |

### Méthodes MaterialisationPartielle

/\* WEB-SERVICE CALL : OdeWebService.MaterialisationPartielle \*/

MaterialisationPartielle(listCuboides, seuil\_poids, solution);

|  |  |  |
| --- | --- | --- |
| ***Paramètre*** | ***Type*** | ***Commentaires*** |
| listCuboides <IN> | List<Dimension>() | Rempli |
| seuil\_poids <IN> | double seuil\_poids | Paramètre |
| solution <OUT> | new int[listCuboides.Count] | Liste d’integers |

<https://support.microsoft.com/en-us/kb/308359>

### Consume a Web Service

1. Open Visual Studio .NET.
2. Under Project types click Visual C# Projects, then click Console Application under Templates.
3. Add a reference for the MathService Web service to the new console application.  
     
   This step creates a proxy class on the client computer. After the proxy class exists, you can create objects based on the class. Each method call that is made with the object then goes out to the uniform resource identifier (URI) of the Web service (usually as a SOAP request).
   1. On the Project menu, click Add Web Reference.
   2. In the Add Web Reference dialog box, type the URL for the Web service in the Address text box and press ENTER. If you set the local computer to host the Web service, the URL is http://localhost/MathService/MathService.asmx.
   3. Click Add Reference. Alternatively, you can type the URL to the discovery file (MathService.vsdisco) or click Web References on Local Web Server in the left pane to select the MathService service from the list.
   4. Expand the Web References section of Solution Explorer and note the namespace that was used.
4. Create an instance of the proxy object that was created. Place the following code in the function called Main:

localhost.Service1 myMathService = new localhost.Service1();

1. Invoke a method on the proxy object that you created in the previous step, as follows:

Console.Write("2 + 4 = {0}", myMathService.Add(2,4));

1. Click Build on the Build menu to build the console application.
2. Click Start on the Debug menu to test the application.
3. Close and save the project