Shiyue Wang
Qing Wang
Yu Chen
Dong Chai
Maxime Sarilar

# TME 5 – Composants C#

*Ce document présente la solution du groupe 1 pour le TME 5 de l'UE Composants.*

La solution Visual Studio se compose de douze projets. Il y a quatre IHM indépendantes :

- L'IHM Serveur Catalogue, qui doit être démarrée en premier
- Les IHM Serveurs Echo et Mathematique, qui seront démarrées après le catalogue pour pouvoir s'enregistrer
- L'IHM Client, qui va pouvoir interroger le catalogue serveur sur les services enregistrés

Chaque service proposé (« `echo` », « `mathematique` » et « `catalogue` ») sont placés dans des `dll` qui seront utilisées par les IHM respectives.

Chaque IHM dispose d'un menu composé de trois entrées :

- « Fichier »
- « ? » pour afficher une aide éventuelle (pas implémentée)
- « Fenêtre », pour accéder à la fenêtre de configuration. Cette fenêtre de paramètre permet de régler (si l'on ne souhaite pas utiliser la configuration par défaut) :
  - Le port du serveur (si l'IHM contrôle un serveur)
  - L'adresse du serveur catalogue (si l'IHM n'est pas celle qui contrôle le serveur catalogue)
  - Le port du serveur catalogue (idem)

Une fois tous les serveurs lancés, le client va cliquer sur le bouton « Interroger Catalogue », ce qui va appeler la méthode getInfos sur le catalogue. La liste des serveurs disponibles se remplie alors, et il suffit d'en cocher un puis de cliquer sur « Connexion » pour pouvoir s'y connecter.

Au moment où le client clique sur « Connexion », une nouvelle fenêtre s'ouvre. Cette fenêtre va se construire dynamiquement, selon les informations reçues par le getInfos du catalogue. Il y a un RadioButton par opération disponible, et pour chaque opération il y aura une TextBox par paramètre requis. Le type de paramètre attendu s'écrit automatiquement dans la TextBox pour aiguiller l'utilisateur. Si l'utilisateur rentre un paramètre incorrect (par exemple s'il essaye d'additionner une série de lettre au lieu de deux chiffres), il recevra une erreur « BAD_PARAMETERS_CODE ». Si le serveur s'est arrêté alors que le client s'était déjà connecté, il recevra un message l'en informant quand il tentera d'envoyer une requête.

Une `dll` d'interfaçage propose un moyen d'implémenter un serveur et un client (Client et Server) par le biais d'interfaces (IClient et IServer). L'IHM Client ne se servira que de IClient, tout comme l'IHM Catalogue ne se servira que de IServer. En revanche, les IHM Echo et Mathematique se comporteront à la fois comme des serveurs (vis-à-vis de l'IHM Client) et à la fois comme des clients (vis-à-vis de l'IHM Catalogue). Elles utilisent donc à la fois IClient et IServer.

Ces interfaces IClient et IServer vont quant à elles manipuler la couche Socket, qui se trouve dans une `dll` séparée. Cette couche Socket est très indépendante du reste et permet simplement d'utiliser la couche TCP en tant que client ou en tant que serveur.

La couche d'interfaçage va manipuler la `dll` correspondant au protocole `ELASTIC`. C'est là que les méthodes d'encodage et de décodage seront invoquées.

L'information circule de la manière suivante :

- « du haut vers le bas » (c'est-à-dire de l'IHM vers la couche socket en passant par la couche d'interfaçage), par simple appel de méthode (chaque couche « connaît » la couche qui est juste en dessous)
- « du bas vers le haut » (c'est-à-dire de la couche socket vers l'IHM), par levée d'évènement (chaque couche s'abonne aux évènements lancés par la couche juste en dessous)

Un service désirant s'enregistrer auprès du catalogue doit envoyer toutes les informations spécifiées dans le protocole, ainsi que la liste de ses opérations (en utilisant l'emplacement « `title` »). La chaîne dénommée `title` se structure de la façon suivante :

```
operation1(typeDuParamètre1,typeDuParamètre2,etc…)typeDeLaValeurDeRetour_operation2(etc…
```

Cette chaîne est construite en utilisant la réflexion, toutes les méthodes de la `dll` du service se devant d'utiliser l'annotation définie dans la couche d'interfaçage (cette annotation spécifie la visibilité d'une méthode (`GLOBALE`, `LOCALE`, `PRIVATE`), la dénomination de l'opération correspondant à la méthode (opération `echo` pour la méthode `DoEcho` par exemple) ainsi que le nom d'une méthode d'interface, qui doit être définie au même endroit, et qui prend en paramètre une `List<string>` et qui retourne également une `List<string>`.

Concernant ce dernier point, l'idée est de pouvoir également appeler les méthodes de manière réflexive. Pour toutes les méthodes exposées (c'est-à-dire dont l'annotation a sa valeur de visibilité à `GLOBALE`), on va chercher celle qui porte la même dénomination (dans son annotation) que le nom de l'opération demandée. Ensuite, on va appeler de manière réflexive la méthode dont le nom est dans l'annotation de l'opération. Cela permet de changer les `dll` de service sans avoir à modifier le code de l'IHM, car tout est fait de manière réflexive (l'enregistrement auprès du catalogue et l'exécution des opérations).

Le serveur catalogue envois toutes les cinq secondes un message à tous les services enregistrés sur lui (ce message est en réalité un code définit de manière statique dans `LibraryDataStructure.DataUtils` de manière à pouvoir être reconnu partout). Tout serveur souhaitant être maintenu au sein du catalogue devra alors répondre par un code de type acknowledgement pour confirmer au catalogue qu'il est toujours présent. On aurait pu choisir un intervalle de temps plus court (par exemple, 100ms au lieu de 5s) mais nous voulions afficher toutes les opérations effectuées par les applications dans le log (la TextBox noire) et ces envois de message inondaient ce log.

Un détail à noter, nous avons dû utiliser un « hack » pour faire fonctionner l'application. En effet, pour s'enregistrer auprès du catalogue, un service doit fournir son adresse IP. La méthode que nous avons utilisé (`DataUtils.GetIPaddresses(string)`) nous renvoyait une adresse qui ne permettait pas de se connecter (du moins pas en local). Le « hack » évoqué plus haut consiste simplement à toujours utiliser `127.0.0.1` lorsque l'on souhaite se connecter, même si cela n'apparaît jamais sur l'IHM.