Dotnet France
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

Introduction à WCF

Version 1.2

Jean DALAT

James RAVAILLE
http://blogs.dotnet-france.com/jamesr

Sommaire

1 Les Application distribuées

1.1 Définition

Une application distribuée est une application dont tous les éléments qui la composent (classes, persistance, logique métier) sont distants géographiquement et communiquent entre eux via des réseaux locaux d'entreprise ou par Internet en utilisant le protocole IP.

Les composants des applications distribuées sont réutilisables et servent souvent à plus d'une application simultanément. Cependant, l'utilisateur final n'a pas conscience de la nature distribuée de l'applicatif car il utilise une interface – Web ou Windows Forms –, dont le but souvent est de fédérer tous les éléments distants pour atteindre l'objectif de l'application tout en masquant tous les mécanismes d'accès.

1.2 Technologies pour applications distribuées

Le principe même des applications distribuées n'est pas nouveau. Beaucoup de technologies et de protocoles, souvent incompatibles entre eux ont été mis en œuvre pour permettre aux composants d'une application distribuée, de communiquer entre eux. Parmi ces technologies, on peut citer :

- **COM** (Component Object Model) : technologie de communication inter application propre à l'environnement Windows. Cette technologie ne permet pas de construire à proprement parler des applications distribuées mais sert d'interface pour faire communiquer des applications sur une même machine. (les différents logiciels de la suite bureautique Microsoft Office par exemple).
- **DCOM** (Distributed Component Object Model) : version Distribuée de COM.
- **.Net Remoting** : technologie de communication inter application incluse dans le Framework .Net 2.0, basée sur un modèle client/serveur permettant la communication et la transmission d'objets entre les applications.
- **RMI** : équivalent Java du .Net Remoting
- **Services Web** : technologie permettant de faire communiquer des composants entre eux, indépendamment de la plateforme sur laquelle ils sont hébergés, en utilisant un protocole de communication et un format de fichier unique et standardisé basé sur du XML. Contrairement aux technologies évoquées précédemment, grâce à la nature standardisée des échanges, il devient tout à fait possible de faire communiquer des modules hébergés sous Windows avec d'autres hébergés sous Linux ou Unix. C'est cette interopérabilité associée à une relative facilité de mise en œuvre, qui fait que les services web sont aujourd'hui une des technologies les plus utilisées, pour construire des applications distribuées.

1.3 Nécessité d'unification des technologies : Windows Communication Foundation (WCF)

A la vue de la diversité des technologies d'applications distribuées et surtout de leur nature propriétaire, qui fait que non seulement la plupart sont incompatibles entre elles mais qu'en plus, chacune étant différente, il est souvent nécessaire pour un développeur d'en connaitre plusieurs, Microsoft a développé Windows Communication Foundation.

2 Les architectures orientées services (SOA)

2.1 Définition

Selon l'OASIS (Organisation for Avancement of Structured Information Standards), l'organisation pour la promotion des standards de l'information structurée, **« l'architecture orientée service (SOA : Services Oriented Architecture) est un paradigme d'organisation des ressources distribuées, potentiellement contrôlées par des domaines différents. ».**

2.2 La naissance du SOA

Pour comprendre l'avènement et en quoi consiste le SOA, il est nécessaire de bien identifier que le but de la programmation structurée, est d'écrire du code qui soit robuste et réutilisable.

Au tout début, il n'existait que les langages purement procéduraux, la seule façon d'écrire du code réutilisable était d'écrire des fonctions et des procédures dans un fichier séparé du corps du programme, et de faire appel à ce fichier chaque fois que nécessaire. Ensuite, est apparue la Programmation Orientée Objet (POO). Elle était innovante dans le sens ou le concept même d' « objet » permet l'encapsulation et donc de masquer la complexité des opérations. Les objets peuvent s'envoyer des messages, grâce aux appels de méthodes exposées par l'objet avec lequel ils souhaitent communiquer sans pour autant savoir comment ledit objet implémente le traitement qu'on lui demande d'exécuter.

Malgré le fait que des technologies comme DCOM, RMI ou .Net Remoting permettent de transporter les objets et donc de dépasser les frontières de la machine grâce au réseau, on s'est souvent heurté à des problèmes de compatibilité entre plateformes, d'où le besoin d'une standardisation et la mise en commun des protocoles (SOAP, XML, …). De là est née la notion d'architecture orientée services (SOA).

2.3 Les 4 grands principes du SOA

Le principe même de SOA repose sur les 4 principes suivants :

- **La définition du service est explicite :**

Un service est une classe exposée à travers les réseaux. Le but de ce service est de fournir une prestation bien définie (exemple : fournir la date et l'heure dans un pays donné). Les détails d'un service (la manière dont il doit être appelé, les paramètres à spécifier…) sont contenus dans un document standardisé, qui fait office de contrat entre le client et le serveur.

- **Les services sont autonomes :**

Un service se doit d'être totalement autonome. On doit pouvoir le remplacer ou le déplacer sans que cela affecte d'autres services. Un service implémente ses propres composants et ses propres méthodes d'accès aux données, il ne doit dépendre d'aucun élément externe.

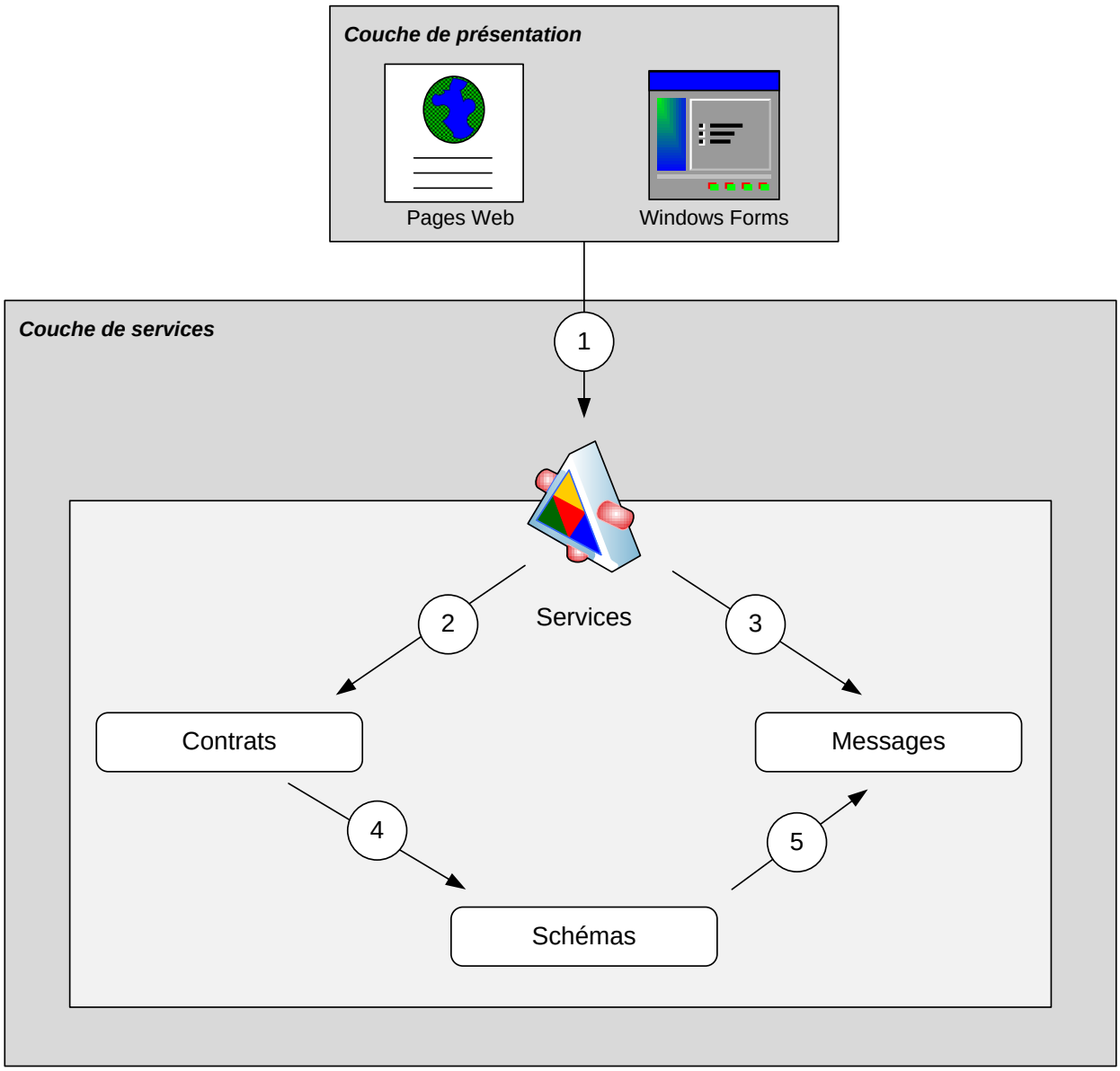
- **Les clients et les services ne partagent que des contrats :**

On a vu dans le premier principe du SOA, que les services exposent des contrats pour exposer aux clients leurs fonctionnalités et comment les utiliser. Cette interface est la seule chose que le serveur partage avec le client. Comme en programmation orientée objet, le client n'a pas a connaitre comment procède le service pour arriver à ses fins. En aucun cas, le service et le client ne doivent partager du code.

- **La compatibilité est basée sur les règles :**

Nous verrons plus tard dans les chapitres suivants qu'il est possible et même recommandé de sécuriser l'accès à un service, en utilisant un nom d'utilisateur et un mot de passe par exemple.

Voici un schéma permettant de présenter les composants fondamentaux d'une architecture SOA :



① Les applications consomment des services distants, pouvant réaliser des tâches métiers et/ou techniques, en s'échangeant des messages (③).

② Chaque service possède un contrat qui fournit des spécifications techniques sur les opérations qu'il propose (signature, données à fournir entrée, données retournée…).

④ et ⑤ Chaque contrat possède un schéma, qui décrit des messages échangées entre les services et les applications qui les consomment.

3 Conclusion

Dans ce chapitre d'introduction à WCF, nous avons vu le concept d'applications distribuées et d'architectures orientée services. Après avoir lu ce chapitre, vous devriez être en mesure de :

- Savoir expliquer ce qu'est une application distribuée.
- Décrire briévement le paradigme SOA, et les régles sur lesquelles il se base.