



550, rue Sherbrooke Ouest, bureau 100
Montréal (Québec) H3A 1B9
Tél. : (514) 840-1234; Téléc. : (514) 840-1244
888, rue St-Jean, bureau 555
Québec (Québec) G1R 5H6
Tél. : (418) 648-8080; téléc. : (418) 648-8141
<http://www.crim.ca>

Analyse des techniques et des standards pour l'interopérabilité entre plateformes

Maguelonne Héritier

Agente de recherche

Mustapha Es-salihe

Conseiller
Développement et technologies Internet

14 mars 2006

TABLE DES MATIÈRES

LISTE DES FIGURES	3
1. INTRODUCTION	4
2. DEFINITIONS ET PROBLEMATIQUES	4
3. TECHNIQUES D'INTEROPERABILITE.....	4
3.1 Échange des données entre applications	5
3.1.1 Sérialisation.....	5
3.1.2 Scénarios d'interopérabilité	5
3.2 Technique d'interopérabilité point à point	6
3.2.1 Services Web : solution à couplage faible.....	6
3.2.2 Appel de procédures distantes : solution à couplage fort.....	10
3.3 Les EAI (Enterprise Application Integration)	11
3.3.1 Présentation des EAI	11
3.3.2 Architectures des EAI.....	12
3.4 Techniques d'interopérabilité au niveau de la couche des données.....	13
3.4.1 Partage d'une base de données commune.....	13
3.4.2 Interopérabilité asynchrone	14
3.5 Patron de conception.....	17
3.5.1 Le patron façade	18
3.5.2 Le patron interface de service	18
BIBLIOGRAPHIE	19

LISTE DES FIGURES

- Figure 1 Architecture des services Web.....* **Erreur ! Signet non défini.**
- Figure 2 Architecture d'accès à une base de données selon Java et .NET..* **Erreur ! Signet non défini.**
- Figure 3 Partage de la couche d'accès à la base de données entre deux plateformes*
..... **Erreur ! Signet non défini.**

1. INTRODUCTION

Offrir davantage d'ouverture et de compatibilité entre systèmes informatiques est depuis longtemps un défi majeur pour les organisations. Au-delà de l'aspect technique, l'interopérabilité répond à un enjeu de modernisation des organisations et leurs systèmes d'informations. L'avènement des technologies Internet n'a fait qu'étendre ce vieux concept à des applications distribuées au-delà d'un réseau d'entreprises. Le secteur économique des nouvelles technologies d'information et de communication (NTIC) est bien évidemment à l'écoute de ce genre de problématique. En effet, fournir des services informatiques intégrables directement, et de manière transparente, au sein de systèmes d'informations existants intéresse nécessairement des acteurs, existants ou futurs, de ce marché. D'ailleurs, les grands éditeurs tels que BEA, IBM ou encore Microsoft communiquent fortement sur la problématique de l'interopérabilité et travaillent même de concert sur le sujet dans différents organismes de normalisation tels qu'OASIS [1].

Le présent document a pour but d'étudier les différentes techniques permettant l'interopérabilité entre plateformes et les standards d'industrie qui facilitent cette interopérabilité.

2. DÉFINITIONS ET PROBLÉMATIQUES

L'interopérabilité correspond à la capacité, plus ou moins grande, qu'ont les applications, quelque soit leur origine, appartenance, diversité, à coopérer entre elles. Elle consiste à utiliser conjointement des fonctionnalités d'applications basées sur des technologies différentes (J2EE, .NET, PHP, C++, etc.). Une des motivations peut provenir de la volonté de consommer, depuis ses applications, des services métiers gérés par des partenaires externes (vente d'assurances, suivi d'envois de colis, etc.).

Le véritable objectif est alors de permettre cette interopérabilité le plus simplement possible, en abstrayant à la fois aux utilisateurs finaux et aux développeurs la complexité et la diversité des environnements.

Tout ensemble d'échanges entre applications doit également être géré de manière sécurisée afin de garantir l'intégrité, l'authenticité et la confidentialité de l'interaction. Des meilleures pratiques de sécurité doivent être adoptées. Il s'agit de gérer l'identité des utilisateurs pour pouvoir identifier le demandeur d'un service et savoir si ce dernier a le droit d'utiliser le service en question. Il s'agit aussi de protéger les données et veiller à leur intégrité et leur disponibilité.

L'administration technique du système d'information ne doit cependant pas s'en trouver complexifiée outre mesure. Il est important que les équipes responsables de cette administration puissent facilement prendre le contrôle et gérer ces solutions.

3. TECHNIQUES D'INTEROPÉRABILITÉ

Il y a plusieurs niveaux d'interopérabilité :

- l'intégration au niveau interface utilisateur : l'interface utilisateur d'une application sert ici de point d'entrée/sortie;

- l'interopérabilité entre interface utilisateur et la couche logique d'affaires;
- l'intégration au niveau de la couche logique d'affaires;
- l'interopérabilité au niveau de la couche de la logique d'affaires et la couche de données.

Il existe plusieurs approches pour l'interopérabilité dont les principales sont :

- les services Web. Des applications fournissent des services dans des formats standards;
- les ponts. Ils font des liens ponctuels entre des plateformes. Par exemple, des classes Java peuvent être vues comme des classes .NET et vice-versa;
- les systèmes de messagerie. Ils offrent un système asynchrone de communication;
- les partages de bases de données. La connexion à la base de données est partagée;
- les EAI (Enterprise Application Integration). Cela consiste à utiliser un négociateur central, qui gère les interactions entre les applications.

Ces différentes approches sont plus ou moins adaptées selon le niveau d'interopérabilité que l'on envisage. Les sections qui suivent vont présenter des solutions d'interopérabilité qui vont les mettre en jeux.

3.1 Échange des données entre applications

3.1.1 Sérialisation

La sérialisation de données permet d'échanger les données d'une application à l'autre. Les techniques de sérialisation binaire ne peuvent être utilisées pour l'interopérabilité entre .Net et Java car elles ne sont pas compatibles entre les deux plateformes. Cependant, la sérialisation XML transforme les données sous forme de fichiers XML qui peuvent être lus aussi bien dans les deux plateformes. Ainsi, pour assurer la compatibilité des types, il suffit de définir les types communs dans un schéma XML (un fichier XSD) utilisé comme référence aux fichiers XML de sérialisation. Les schémas XSD peuvent être compréhensifs dans les deux plateformes. Il faut s'assurer que les types utilisés sont compatibles dans les deux plateformes.

3.1.2 Scénarios d'interopérabilité

3.1.2.1 Lier deux nouvelles applications provenant des deux plateformes différentes

1. Utiliser un schéma XSD pour définir les types communs. Générer ensuite le code de ces types communs pour chacune des plateformes.
2. Placer des schémas XSD dans un répertoire central pour l'équipe de développement afin d'assurer la consistance dans la génération des types à travers l'application.
3. Éviter d'utiliser des éléments que le XSD ne peut définir.

4. Effectuer des tests pour chaque nouveau type utilisé avant d'écrire l'application.

3.1.2.2 Lier une nouvelle application à une autre déjà existante entre deux plateformes différentes

1. Générer un XSD à partir de l'application existante.
2. Utiliser ce XSD pour générer ensuite le code des types partagés pour la nouvelle application.
3. Suivre les indications du premier scénario.

3.1.2.3 Lier deux à plusieurs applications existantes sur des plateformes différentes

Scénario 1 (le plus commun).

1. Sélectionner les types de données d'une des applications comme format commun à toutes les autres applications.
2. Implémenter une couche d'adaptation à toutes les autres applications pour convertir leurs types de données au format commun.

Scénario 2

1. Créer un schéma XSD qui représente globalement les données échangées par l'ensemble des applications.
2. Générer le code des types de données pour chacune des plateformes.
3. Implémenter des adaptateurs pour convertir les types de chaque application au format des types générés à partir du schéma XSD.

3.2 Technique d'interopérabilité point à point

3.2.1 Services Web : solution à couplage faible

Les services Web forment une technologie émergente permettant à des applications de dialoguer à distance via Internet et ceci, indépendamment des plateformes sur lesquelles elles reposent. Les services Web visent à réduire les problèmes d'interopérabilité en permettant à des applications de communiquer directement entre elles et d'échanger des données indépendamment de leur localisation, des systèmes d'exploitation ou langages utilisés.

3.2.1.1 Fonctionnement des services Web

Un service Web est une application modulaire, qui est accessible par un réseau (Internet, Intranet ou Extranet) par le biais d'une interface au format standard XML.

Comme illustré sur la figure ci-dessous, les services Web offerts à un partenaire ou à un client sur Internet s'appuient donc sur un ensemble de standards à base de XML à savoir :

- SOAP (Simple Object Access Protocol). C'est un protocole gérant l'échange d'informations. Il utilise XML pour formater ses messages et HTTP pour les véhiculer. Dans sa version 1.2, il s'étend au support d'autres protocoles que HTTP (TCP, UDP, etc.) tout en étant capable d'intégrer des fichiers attachés (par le biais de SOAP Attachment et DIME);
- WSDL (Web Service Description Language) donne la description au format XML des services Web en précisant les méthodes pouvant être invoquées, leur signature et le point d'accès (URL, port, etc.). C'est, en quelque sorte, l'équivalent du langage IDL pour la programmation distribuée CORBA;
- UDDI (Universal Description and Discovery Integration). C'est un annuaire indexant les services Web accessibles à un ensemble de partenaires selon le niveau d'échanges (intra entreprise, inter entreprise ou public). L'objectif est de permettre à terme aux services Web de s'invoquer dynamiquement.

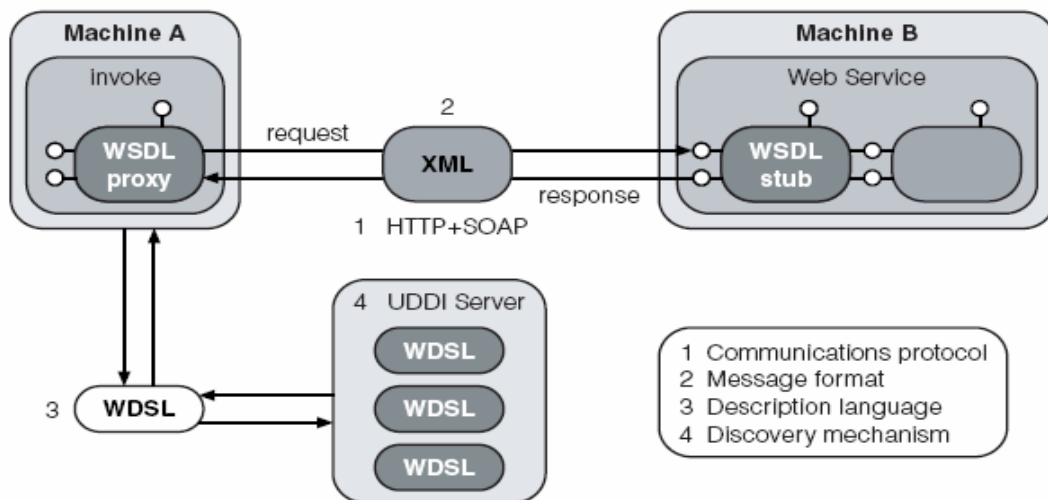


Figure 1 Architecture des services Web

Un service Web doit tout d'abord être décrit à l'aide du langage de description WSDL. Ensuite, il doit être enregistré dans le répertoire UDDI. Finalement, il pourra être découvert et invoqué par le biais d'un protocole de transport appelé SOAP. En d'autres termes, les services Web permettent à des applications externes de localiser, d'exécuter d'autres applications résidant sur un réseau local ou public et, par la même occasion, retourner l'information via de simples messages basés sur XML. Un scénario typique d'échange revient à ce qu'une application cliente puisse connaître quelles fonctionnalités un service Web fournit et comment appeler cette fonctionnalité, en demandant le fichier WSDL. Ensuite le client envoie une requête à l'URL donné en utilisant le protocole SOAP par HTTP. Le service reçoit la requête, l'exécute, et retourne une réponse. La requête et la réponse sont formatées en XML en utilisant le protocole SOAP.

3.2.1.2 Standards des services Web

L'architecture des services Web s'est imposée grâce à sa simplicité, à sa lisibilité et à ses fondations normalisées.

3.2.1.2.1 Les organismes de standardisations

Web Services Interoperability Organization (WS-I) [2]. Ce regroupement a été lancé par Microsoft et IBM début 2002 en vue de faciliter la mise en œuvre de services Web compatibles entre eux, quels que soient les plateformes et langages sous-jacents utilisés. Pour ce faire, il publie une série de guides de bonnes pratiques et fournit des outils de test de compatibilité. Pour l'heure, le WS-I compte près de 150 membres, dont Sun.

Organization for the Advancement of Structured Information Standards (OASIS)[1]. OASIS exploite les technologies définies au sein du W3C, le XML notamment, en vue de répondre aux problématiques d'échange de données et d'orchestration de processus inter-organisation. Dans ce contexte, il s'intéresse à la fois à la standardisation des démarches commerciales (avec ebXML) et légales (avec LegalXML), tout en intégrant certaines problématiques sous-jacentes (gestion des droits numériques, etc.).

World Wide Web Consortium (W3C) [3]. Organisme officiellement chargé de normaliser les standards liés au Web, le W3C a notamment publié les spécifications du langage de présentation HTML et de formats d'images (PNG, etc.). Il couvre également XML et ses dérivés en matière de publication de contenus et d'applications (services Web, etc.). Il compte notamment parmi ses membres France Telecom, IBM, Microsoft et Sun.

Internet Engineering Task Force (IETF) [4]. Créé en 1986, l'IETF est un organisme indépendant qui a pour but de travailler sur les couches réseau reposant sur le protocole IP (Internet Protocol) : routage, transport, sécurité, etc. Dépendant de l'ISOC (Internet Society), l'IETF réalise des notes de recommandations techniques touchant à de très nombreux domaines.

3.2.1.2.2 Les projets de standardisation

La normalisation actuelle autour des services Web est cependant un vaste chantier qui va bien au-delà de la simple invocation d'une méthode d'un objet distant. Différents travaux ont ainsi démarré pour tenter de définir une véritable infrastructure distribuée, capable de satisfaire l'ensemble des besoins d'une application distribuée, aussi bien en termes de normalisation des échanges qu'en termes de services transverses.

À l'heure actuelle, seule la couche de transport est réellement normalisée et ne souffre d'aucune contestation. Elle s'appuie sur le protocole SOAP pour l'échange des messages et sur le langage WSDL pour la définition du contrat de l'interface. Dans les autres couches, il se développe de nombreux projets plus ou moins concurrentiels.

Dans la couche de découverte :

- **WS- Inspection** : issus des projets ADS (IBM) et DISCO, WS-Inspection se charge d'agréger les pointeurs vers les services Web qu'une entreprise souhaite exposer. À la différence d'UDDI, il ne permet pas de rechercher des services dont on ne connaîtrait pas les coordonnées;

- XRI : ce mécanisme (Extensible Resource Identifier) qui repose sur le standard URI (Uniform Resource Identifier) vise à proposer une solution équivalente à celle des noms de domaines (URL) pour retrouver un service Web. Projet soutenu par Microsoft, Sun et IBM.

Échange :

- WS-Addressing : WS-Addressing, projet dessiné par IBM, Microsoft et BEA, permettrait de véhiculer des messages SOAP de manière bi-directionnelle, que ce soit en mode synchrone ou asynchrone, le tout indépendamment de la couche de transport.

Gestion de l'intégrité des messages :

- WS-Reliability : cette spécification standardise le dispositif mis en œuvre entre deux services Web en vue de garantir la réception des messages transmis (par le biais d'accusés de réception notamment). Projet initié par Sonic Software et Sun;
- WS-Acknowledgment : avancée par BEA System, l'objectif de cette proposition semble comparable à celle de WS-Reliability;
- WS-ReliableMessaging : très proche de WS-Acknowledgment (y compris en termes de vocabulaire), WS-ReliableMessaging a été défini conjointement par IBM, Microsoft, BEA et TIBCO.

La gestion de la sécurité et des transactions est actuellement le frein le plus important à la mise en place d'architectures distribuées à base de services Web. Plusieurs chantiers sont ouverts mais aucun n'est réellement accepté.

Gestion de l'intégrité des transactions :

- WS-Transaction : reposant sur WS-Coordination (IBM), WS-Transaction vise à garantir l'intégrité de transactions exécutées via les services Web. Il devrait être prochainement soumis par Microsoft à un organisme de standardisation. Projet initié par Microsoft;
- BTP : Initié par BEA, Business Transaction Protocol gère l'intégrité des transactions longues et complexes. Il est comparable à WS-Transaction;
- WSCL : Services Conversation Language (WSCL) qui a été soumis au W3C par HP a pour but de préciser les règles d'invocation d'un service Web et a le statut de Note.

Sécurité :

- SAML : protocole couvrant les problématiques d'authentifications et d'habilitations des transactions SOAP. Projet soutenu par IBM et Microsoft;
- XACML : Access Control Markup Language définit un schéma pour décrire les politiques de droits d'accès à des objets XML. Projet initié par Entrust et soutenu par IBM;
- WS-Security : soumis par Microsoft, IBM et Verisign, WS-Security couvre l'authentification, la gestion de l'intégrité et le cryptage des échanges. Il précise en outre la manière de décrire les droits et les conditions d'utilisation associés à un service Web. Il se décline en nombreuses sous-spécifications (WS-Trust, WS-Secure, WS-Policy, WS-Privacy, WS-Federation et WS-Authorization);
- XrML : proposé par l'éditeur américain ContentGard, eXtensible rights Markup Language aborde les conditions d'utilisation (ou droits numériques) associées au contenu d'un service Web.

3.2.1.3 Service Web et interopérabilité

Ainsi, les services Web répondent à trois principales problématiques de l'interopérabilité :

- protocole de transport standard : HTTP et HTTPS sont les plus communes implémentations, mais les services Web ont la flexibilité d'utiliser n'importe quel protocole de transport;
- définition de type : les services Web peuvent fournir des données fortement typées. Si un service Web utilise un type, un autre service Web peut comprendre et utiliser ce type sans se soucier de la plateforme ou du langage de l'environnement qui le supporte;
- multiple niveaux de support : habilité d'implémenter des services Web dans n'importe quel langage, n'importe quelle plateforme, et utiliser n'importe quel outil commercial. Ainsi le client n'a pas besoin de connaître la plateforme sur laquelle un service particulier s'exécute.

Cependant, les services Web ne sont pas toujours la solution idéale pour tous les scénarios d'interopérabilités. En effet SOAP est basé sur du texte, les appels aux services Web peuvent être lents pour des applications qui demandent des communications rapides et fréquentes. Aussi, ils ne sont pas toujours indiqués avec les modèles conventionnels orientés objet. Les objets activés par le client comme ceux que l'opérateur « new » construit, et les requêtes d'accès aux méthodes statiques ne sont généralement pas supportés. De même, ils ne sont pas recommandés pour une application qui a besoin d'accéder à une grande variété de classes et d'objets. Les services Web ne supportent pas les procédures de rappel de la même manière qu'une architecture locale orientée objet. Aussi, l'enveloppe technique initiale des services Web laisse de nombreuses lacunes, ne traitant pas encore les aspects suivants de cryptage, d'authentification, de signature, de transactions, et d'orchestration (paragraphe 3.2.1.2.2).

3.2.2 Appel de procédures distantes : solution à couplage fort

L'appel de procédures et l'invocation d'objets à distance est un mécanisme qui permet d'exécuter une fonction située dans un autre exécutable pouvant être sur une machine distante. Il permet aussi à des objets distribués de communiquer par appel de méthodes. Son but est de masquer à l'appelant qu'une procédure appelée s'exécute sur une machine distante comme si elle était locale.

Comme technique d'appel de procédures distantes dans l'environnement Microsoft .NET, Net Remoting est un mécanisme de communication et de transfert de données pour des applications distribuées. Bien qu'il soit conçu au départ pour communiquer entre des applications .Net. Net Remoting dispose d'un Framework extensible que l'on peut personnaliser pour effectuer des connections avec des applications autre que .NET. Par exemple, pour utiliser Net Remoting avec des applications Java, il faut implémenter un pont (runtime bridge). Un pont est une solution d'interopérabilité ponctuelle qui permet principalement l'accès bas niveau à des objets Java en .NET. Les classes Java sont vues comme des classes .NET et vice-versa. Cela peut se faire avec Ja.Net ou JNBridgePro de JNBridge.Inc. L'avantage de cette approche est qu'elle est facile d'implémentation et permet parfois une meilleure performance. L'inconvénient majeur est qu'elle offre un lien fortement couplé et qu'elle est très dépendante de la plateforme .NET.

Dans le monde Java, RMI (Remote Method Invocation) est une technique permettant de manipuler des objets distants (c'est-à-dire un objet instancié sur une autre machine virtuelle, éventuellement sur une autre machine du réseau) de manière transparente pour l'utilisateur, c'est-à-dire de la même façon que si l'objet était sur la machine virtuelle (JVM) de la machine locale. Ainsi un serveur

permet à un client d'invoquer des méthodes à distance sur un objet qu'il instancie. Deux machines virtuelles sont donc nécessaires (une sur le serveur et une sur le client) et l'ensemble des communications se fait en Java. Les connexions et les transferts de données dans RMI sont effectués par Java sur TCP/IP grâce à un protocole propriétaire (**JRMP**, *Java Remote Method Protocol*) sur le port 1099. À partir de Java 2, version 1.3, les communications entre client et serveur s'effectuent grâce au protocole RMI-IIOP (*Internet Inter-Orb Protocol*), un protocole normalisé par l'OMG (*Object Management Group*) et utilisé aussi dans l'architecture CORBA.

3.3 Les EAI (Enterprise Application Integration)

3.3.1 Présentation des EAI

Par « plateforme-EAI » nous désignons un logiciel qui prend en charge le dialogue inter-applications. Ces plateformes jouent, en quelque sorte, le rôle d'îlotier du système d'information. Ce nœud central apporte une notion de découplage très intéressante : l'utilisation d'un format intermédiaire de communication évite d'intégrer « n » fois une application avec « n » autres applications (technique point à point), mais simplement une fois sur le nœud central. Ce nœud central assure ensuite la communication avec les autres applications. Les applications d'une entreprise étant nombreuses et échangeant beaucoup de données, les techniques point à point, dans ces circonstances, sont parfois trop lourdes et mal adaptées. D'où le besoin de déployer dans l'entreprise un logiciel qui va permettre d'industrialiser et de rationaliser ces échanges d'informations. Une plateforme d'EAI remplit ce rôle en assumant quatre types de fonctions :

- le routage;
- la transformation;
- les connecteurs (aux applications);
- le transport.

Autrement dit, en fonction d'événements préalablement définis, un logiciel d'EAI récupère les données d'une application, puis les « route » vers leur destination (une autre application), non sans les avoir préalablement converties dans un format adéquat. À cette fin, il met en œuvre :

- un serveur d'intégration qui comprend un moteur de règles et un gestionnaire de messages (pour le routage et la transformation);
- des connecteurs (pour dialoguer avec les applications);
- un MOM (middleware oriented messages) pour transporter les messages. Quasiment tous les fournisseurs d'EAI disposent de leur propre MOM même si la plupart du temps les plateformes d'EAI sont déployées sur des MOM déjà installés dans l'entreprise.

La gestion des processus métiers BPM (Business Process Management) prolonge assez naturellement les quatre fonctions de l'EAI. Le BPM ajoute une couche d'abstraction où l'information manipulée n'est plus une donnée ou un flux technique d'une application vers une autre mais un processus métier. À travers le logiciel de BPM, un architecte métier définit un processus (par exemple le circuit de validation des informations concernant un nouveau client) qui sera traduit en flux techniques, lesquels sont transmis au serveur d'intégration à des fins de paramétrage. Dans la réalité, ce n'est pas vraiment une surprise, tout n'est pas aussi transparent et automatique. Passer les infor-

mations du BPM au serveur d'intégration demande des interventions humaines pour, au minimum, finaliser le paramétrage technique.

Avec l'ouverture croissante du système d'information aux partenaires et clients, les éditeurs d'EAI ont naturellement étendu le champ d'action de leurs plateformes à l'intégration B2B (Business to Business). À cette fin, beaucoup de ces éditeurs ont rendu leur plateforme capable d'interpréter les sémantiques définies par des consortiums tels que RosettaNet. Dans le domaine de l'intégration B2B, deux grandes écoles cohabitent :

- ceux qui, à l'instar de WebMethods, estiment que les flux B2B exploitant les technologies Web (de HTTP à XML) ne sont pas de la même nature que les flux internes. Et demandent donc des serveurs d'intégration distincts;
- ceux pour qui, à l'instar de Seebeyond, l'intégration B2B prend la forme d'un connecteur enrichi de fonctionnalités relatives aux problématiques B2B et qui s'adosse au serveur d'intégration également exploité pour les flux internes.

3.3.2 Architectures des EAI

L'EAI, comme nous l'avons défini précédemment, s'apparente plus à un concept qu'à une solution. Et ce concept ne renvoie pas à un modèle d'architecture mais au moins à deux modèles : « Hub and spoke » et « Network Centric ».

3.3.2.1 L'architecture « Hub and spoke »

C'est le modèle centralisé de l'EAI. Ici, tout passe par un « hub » central. Aucun flux n'est possible sans l'entremise de ce « hub ». Quand une application envoie un message, ce dernier est expédié à destination du « hub ». Le référentiel (la base où sont stockées les règles de routage et de transformation) est donc lui aussi centralisé. L'avantage d'une telle architecture saute aux yeux : l'administration est grandement facilitée. En revanche, la gestion de la charge s'avère complexe dans ce type d'environnement : la seule solution consiste en effet à multiplier les « hubs » sur les différents segments du réseau, sachant qu'il faudra veiller à synchroniser les règles stockées sur ces différents nœuds. L'administration devient alors moins aisée. Seebeyond, Sun (iPlanet) et Tibco présentent des produits de type « Hub and spoke ».

3.3.2.2 L'architecture « Network Centric »

Il s'agit cette fois de la version décentralisée de l'implémentation de l'EAI. Des référentiels de règles et des gestionnaires de messages sont distribués sur l'ensemble des nœuds (point de connexion à une application). Quand une application émet un message, ce dernier est traité par le référentiel du nœud correspondant afin que les applications abonnées à ce type de messages le reçoivent. Avec ce type d'architecture, la charge est donc naturellement répartie sur l'ensemble des nœuds. WebMethods, Sybase-NEON, BEA, Vitra et Mercator présentent des produits de type « Network Centric ».

Aucune de ces architectures n'est fondamentalement meilleure que l'autre. Une étude au cas par cas s'avère nécessaire pour les départager. D'autant que cette dichotomie technique a des conséquences tarifaires. Une plateforme de type « Network Centric » est facturée en premier lieu en fonction du nombre de nœuds; résultat, le client doit d'emblée payer pour le périmètre des applications qu'il souhaite couvrir. En revanche, avec le modèle « Hub and spoke », un client a la possibilité de n'utiliser qu'un seul « hub » dans un premier temps pour couvrir un large éventail d'applica-

tions. Autrement dit, l'architecture « Hub and spoke » donne plus de souplesse que sa concurrente pour acheter des « hubs » au fur et à mesure de la montée en charge.

3.4 Techniques d'interopérabilité au niveau de la couche des données

Ce chapitre explique comment implémenter une connectivité entre les Plateformes .Net et Java au niveau de la couche de données tel qu'une base de données. .Net Remoting et les services Web peuvent fournir des systèmes de liens soit fortement couplés ou faiblement couplés sur Intranet ou Internet.

3.4.1 Partage d'une base de données commune

.NET utilise ADO.NET comme moyen de connexion à une base de données et Java utilise JDBC. Ces deux technologies ont l'avantage d'offrir une abstraction de l'accès à la base de donnée du reste de l'application. En effet, pour une facilité de la programmation et d'une bonne maintenance du code, il est recommandé d'implémenter une couche qui abstrait le code d'accès à la base de données de la couche logique d'affaires. Cette couche d'accès aux données abstrait la sémantique de la base de données ainsi que sa technologie d'accès (ADO.NET, JDBC) et fournit ainsi une simple interface programmable pour effectuer les opérations sur les données stockées.

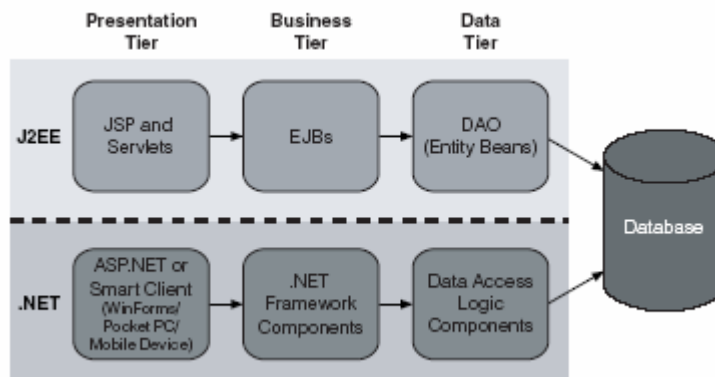


Figure 2 Architecture d'accès à une base de données selon Java et .NET

Cette approche commune ouvre la porte à des scénarios d'interopérabilité où la couche logique d'affaires d'une plateforme peut appeler la couche d'accès aux données d'une autre.

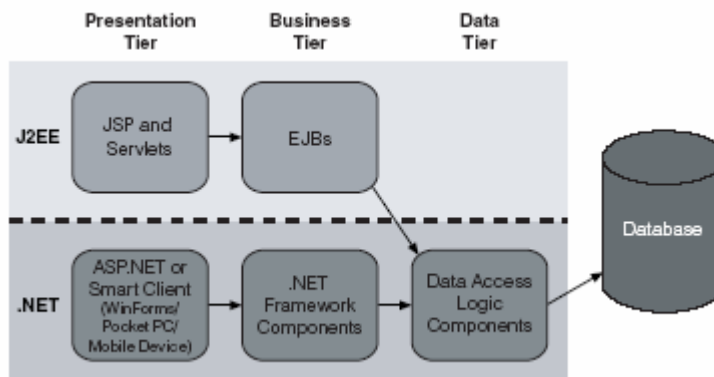


Figure 3 Partage de la couche d'accès à la base de données entre deux plateformes

Le défi dans cette méthode est de créer un schéma de base de données approprié pour toutes les applications.

3.4.2 Interopérabilité asynchrone

Dans une communication asynchrone, il n'y a pas de garantie que l'application demandée soit disponible. Les techniques d'interopérabilité point à point nécessitent que les communications soient synchrones pour être efficaces.

On distingue deux types d'opérations asynchrones :

- **les opérations non bloquantes.** C'est le client qui gère l'asynchronisme. L'application cliente continue à fonctionner en créant un processus séparé qui prend en charge l'opération asynchrone;
- **les opérations à sens uniques.** C'est une vraie opération asynchrone dans le sens où le client envoie une requête qui est gérée par un composant serveur séparé. Le client peut ensuite vérifier la progression de sa requête en accédant aux informations de statuts avant de recevoir le résultat de sa transaction.

3.4.2.1 Les opérations non bloquantes

Bien que les services Web sont de nature synchrone, il est possible de les utiliser pour des opérations de connexions asynchrones non bloquantes. L'appel aux fonctionnalités d'un service Web se fait par l'intermédiaire d'une classe proxy. Celle-ci peut implémenter des méthodes d'appels asynchrones aux services Web en plus des méthodes d'appels synchrones.

Ce type d'appels asynchrones doit se faire dans le but d'améliorer les performances de l'application. Cette technique est recommandée pour les cas où le client et les services Web ont une connexion permanente, l'appel au service Web n'est pas trop long et le client peut effectuer d'autres processus durant l'appel au service Web. Dans les autres cas, il est alors recommandé d'utiliser une opération à sens unique.

3.4.2.2 Opérations à sens unique : vraie communication asynchrone

Les opérations à sens unique requièrent l'utilisation de file d'attente. Les files d'attente de messages fournissent des liaisons asynchrones normalisées. Elles ont pour but que l'expéditeur et le récepteur du message ne soient pas contraints de s'attendre l'un l'autre. Des messages placés dans la file d'attente sont stockés jusqu'à ce que le destinataire les recherche. L'expéditeur n'a pas à attendre que le récepteur commence à traiter son message, il poste son information et peut passer à autre chose. Les messages peuvent avoir des tailles différentes, des types différents, des utilités différentes. Un processus crée une file d'attente dans laquelle plusieurs processus de n'importe qu'elles plateformes peuvent interagir, en lisant ou écrivant des messages. Elles permettent ainsi une communication entre différents systèmes informatiques, connectant plusieurs applications, ou plusieurs systèmes d'exploitation. Plusieurs produits commerciaux de file d'attente sont disponibles. Seulement la norme JMS de Java et les produits Microsoft Message Queuing (MSMQ) et IBM WebSphere MQ (MQSeries) vont être le support de l'étude qui suit. Websphere MQ est basé sur la technologie Java et est particulièrement utilisé par les entreprises. MSMQ est le produit de Microsoft le plus adapté à la plateforme .NET.

L'interopérabilité par messagerie permet un couplage faible des opérations, particulièrement pour les liens multiples entre applications. La messagerie supporte les transactions, la sécurité, la tolérance aux interruptions de service, et l'enregistrement de la délivrance des messages. Mais, la messagerie n'offre aucune forme d'opérations synchronisées et peut échouer lors de l'assignation des ports et des opérations de couvre-feu.

3.4.2.2.1 Présentation de JMS

JMS est un service de messagerie Java qui fait partie des spécifications J2EE. Il fournit une couche d'abstraction pour plusieurs produits de file d'attente, et tous les vendeurs J2EE implémentent un support JMS. JMS définit plusieurs entités :

- un fournisseur JMS : outil qui implémente l'API JMS pour échanger les messages : ce sont les courtiers de messages;
- un client JMS : composant écrit en Java qui utilise JMS pour émettre et/ou recevoir des messages;
- un message : données échangées entre les composants.

Les messages sont asynchrones mais JMS définit deux modes pour consommer un message :

- mode synchrone. Dans ce cas, l'application est arrêtée jusqu'à l'arrivée du message;
- mode asynchrone.

Les courtiers de messages ou MOM (Middleware Oriented Message) permettent d'assurer l'échange de messages entre deux composants nommés clients. Ces échanges peuvent se faire dans un contexte interne (pour l'EAI) ou un contexte externe (pour le B2B). Les deux clients n'échangent pas directement des messages : un client envoie un message et le client destinataire doit demander la réception du message. Le transfert du message et sa persistance sont assurés par le courtier. Les échanges de message sont asynchrones et fiables (les messages ne sont délivrés qu'une et une seule fois).

Les MOM représentent le seul moyen d'effectuer un échange de messages asynchrones. Ils peuvent aussi être très pratiques pour l'échange synchrone de messages plutôt que d'utiliser d'autres mécanismes plus compliqués à mettre en œuvre (sockets, RMI, CORBA ...).

Les courtiers de messages peuvent fonctionner selon deux modes :

- le mode point à point (queue);
- le mode publication/abonnement (publish/subscribe).

Le mode point à point (point to point) repose sur le concept de files d'attente (queues). Le message est stocké dans une file d'attente puis il est lu dans cette file ou dans une autre. Le transfert du message d'une file à l'autre est réalisé par le courtier de messages. Chaque message est envoyé dans une seule file d'attente. Il y reste jusqu'à ce qu'il soit consommé par un client et un seul. Le client peut le consommer ultérieurement : la persistance est assurée par le courtier de messages.

Le mode publication/abonnement repose sur le concept de sujets (Topics). Plusieurs clients peuvent envoyer des messages dans ce « topic ». Le courtier de message assure l'acheminement de ce message à chaque client qui se sera préalablement abonné à ce sujet. Le message possède donc potentiellement plusieurs destinataires. L'émetteur du message ne connaît pas les destinataires qui se sont abonnés.

3.4.2.2.2 Websphere MQ

Une des grandes différences entre MSMQ et Websphere MQ est que ce dernier tourne sur plusieurs systèmes d'exploitation. (Linux, UNIX, AIX, HP-UX, Sun Solaris, Windows) Il supporte aussi les messages de plus de 35 différentes plateformes. Les applications utilisent la file d'attente via une interface programmable connue sous le nom de MQI (Message Queue Interface). C'est une API multi plateforme. Websphere MQ fournit une implémentation des fonctionnalités MQI pour .NET et Java avec un support pour JMS.

3.4.2.2.2.1 Connexion d' une application Java à Websphere MQ

Il y a trois moyens pour se connecter à WebSphere à partir d'une application Java :

- les classes Websphere MQ pour Java;
- les classes WebSphere MQ pour JMS par messagerie point à point;
- les classes WebSphere MQ pour JMS par publication / abonnement.

Les classes WebSphere MQ pour Java permettent de se connecter au serveur WebSphere MQ soit directement soit par l'intermédiaire d'un client Websphere MQ. Ces classes permettent à des applications Java, des applets, des servlets de communiquer avec Websphere MQ. Envoyer et recevoir des messages sont alors des tâches faciles. Cependant ces classes ne font pas partie des spécifications J2EE. Seulement Websphere implémente ces classes. Aussi, il faut veiller à envoyer des messages dans un format de données qui soit compréhensible par toutes les plateformes.

Les classes WebSphere MQ pour JMS implémentent les interfaces JMS de Sun, permettant aux programmes Java d'accéder à Websphere MQ. Les classes pour JMS supportent aussi bien les techniques point à point que la technique publication / abonnement de JMS.

3.4.2.2.2 Connexion d'une application .NET à Websphere

Il y a deux moyens pour connecter des applications .NET à Websphere :

- utiliser les classes WebSphere MQ pour Microsoft .NET;
- utiliser JMS avec un pont.

Les classes WebSphere MQ pour Microsoft .NET fournissent un accès à Websphere MQ à peu près de la même façon que pour Java. Mais la version pour .NET ne supporte pas les connexions groupées ou l'envoi de messages à plusieurs files d'attente. Le transfert des données avec des clients Java et avec des clients .NET se fait dans un format brut sans sérialisation ou paquetage. Ainsi, il est possible de recevoir et d'envoyer des messages à la file d'attente en utilisant les mêmes objets et les mêmes méthodes pour les deux plateformes.

Il est aussi possible d'utiliser un pont comme Ja.NET ou JNBridgePro pour accéder aux fonctionnalités JMS de Websphere MQ à partir des applications .NET. L'accès aux fonctionnalités JMS d'un client .NET peut ainsi se faire de la même façon que pour un client Java. Pour ce faire, il suffit de créer des proxys .NET pour toutes les classes Java nécessaires à l'envoi de messages par JMS. Les proxys gèrent la communication entre les clients .NET et les clients WebSphere MQ.

3.4.2.2.3 Utilisation de MSMQ

La connexion asynchrone entre deux plateformes différentes peut se faire via MSMQ en utilisant des messages basés sur un schéma XML commun (paragraphe 3.1).

Il n'y a pas de problème pour connecter .NET à MSMQ. La plateforme .Net fournit un espace de nommage et un ensemble de classe qui supporte MSMQ. Mais il n'est pas facile de connecter MSMQ à une application Java car Microsoft ne fournit pas un client MSMQ pour Java. Il faut utiliser d'autres stratégies. Il y a trois moyens pour répondre à ce problème :

- utiliser un pont Java-COM (Component Object Model);
- utiliser un JMS fournisseur pour MSMQ;
- créer une interface avec service Web.

3.5 Patron de conception

Les patrons de conception façade et interface de service sont très utiles et recommandés dans le contexte de l'interopérabilité.

3.5.1 Le patron façade

L'utilisation du patron façade pour encapsuler la couche logique d'affaires est une pratique populaire et recommandée. Ce patron s'utilise aussi bien dans les applications de l'environnement .NET que les applications J2EE pour abstraire plusieurs composants de la couche logique d'affaires. Le patron de conception façade fournit une seule interface cliente pour accéder à la couche logique d'affaires. Une simple application peut avoir une seule façade qui encapsule toutes les fonctionnalités. Des applications plus importantes peuvent avoir plusieurs façades qui exposent différentes parties des fonctionnalités d'affaires.

3.5.2 Le patron interface de service

C'est un élément logiciel qui gère le « mapping » et les transformations nécessaires pour la communication avec un service. Une interface service Web peut fournir des fonctionnalités d'affaires comme un service Web. Elle fournit un point d'entrée qui abstrait l'implémentation de la couche logique d'affaires. Les interfaces de services peuvent aussi agir comme porte d'accès aux composants affaires, tout en prenant en charge des tâches complexes comme l'authentification, la validation ou la gestion d'un cache.

L'application peut aussi avoir besoin d'exposer cette fonctionnalité affaires à différents canaux de communications, qui accèdent seulement à une seule interface de service ou façade. Cette interface de service rassemble les composants d'affaires en une seule interface. Il est possible de réutiliser plusieurs interfaces de service dans les scénarios d'interopérabilité.

BIBLIOGRAPHIE

- [1] OASIS : Organization for the Advancement of Structured Information Standards, voir <http://www.oasis-open.org/>
- [2] WS-I : Web Services Interoperability Organization, voir <http://www.ws-i.org/>
- [3] W3C : World Wide Web Consortium, voir <http://www.w3.org/>
- [4] IETF : Internet Engineering Task Force, voir <http://www.ietf.org/>
- [5] Microsoft, Application Interoperability : Microsoft .NET and J2EE, Pattern & Practices.