



INSTITUT SUPÉRIEUR D'INFORMATIQUE  
DE MODÉLISATION ET DE LEURS APPLICATIONS

Campus des Cézeaux  
24 Av. des Landais  
BP 10125  
63173 AUBIERE CEDEX

RAPPORT D'INTÉGRATION D'APPLICATION  
FILIERE GÉNIE LOGICIEL ET SYSTÈMES INFORMATIQUES

---

VALORISATION D'UNE BASE DE DONNÉES AVEC  
GOOGLE APP ENGINE ET GOOGLE WEB TOOLKIT

---

*Étudiants* : Antoine COLMARD et Nicolas PRUGNE

Antoine COLMARD et Nicolas PRUGNE : *Rapport d'intégration d'application*, Valorisation  
d'une base de données avec Google App Engine et Google Web Toolkit, Décembre  
2014

## TABLE DES FIGURES

---

FIGURE 1	Classes métiers persistantes . . . . .	3
FIGURE 2	Interface BiblioService . . . . .	4
FIGURE 3	Architecture RPC . . . . .	5
FIGURE 4	Demande de la liste des auteurs via le mécanisme des RPC . . .	6

# TABLE DES MATIÈRES

---

1	INTRODUCTION	1
2	SERVEUR GWT	2
2.1	Gestion de la base de données . . . . .	2
2.2	Support des RPC avec GWT . . . . .	3
2.3	Workflow du mécanisme des RPC avec GWT . . . . .	5
3	CLIENT GWT	8
3.1	L'architecture MVP . . . . .	8
3.2	Mise en oeuvre du modèle MVP dans le TP . . . . .	8
4	RÉSULTATS ET TESTS	9
5	CONCLUSION	10

# INTRODUCTION

---

## SERVEUR GWT

---

Pour rappel, le logiciel développé au cours de ce TP concerne la mise en œuvre d'un service de gestion de livres centralisé. Cette application est donc divisée en deux parties. D'un côté, se trouve une application cliente, celle que les utilisateurs finaux utiliseront et de l'autre se trouve une application serveur. Cette dernière doit offrir le support de tous les mécanismes de gestion des livres de manière centrale.

Ces deux outils ont vocation à s'exécuter sur des machines différentes mais doivent tout de même pouvoir dialoguer à travers le réseau. Le client doit pouvoir appeler des méthodes implémentées par le serveur, et ce dernier doit pouvoir lui retourner des résultats. Il s'agit là de mettre en place un système permettant au client de faire des RPC (Remote Procedure Call) sur le serveur.

Lorsque ce dernier recevra un de ces appels distant, il devra alors traiter la requête et la retourner au client. Le traitement implique bien entendu l'interaction avec une base de données implantées côté serveur.

C'est pourquoi, dans cette partie, il sera tout d'abord question de découvrir l'architecture de la base de données utilisées par le serveur. Puis, viendra ensuite une partie sur le support des RPC que propose GWT. Enfin, un exemple de mise en œuvre de ces RPC viendra compléter les explications de la partie précédente.

### 2.1 GESTION DE LA BASE DE DONNÉES

Par rapport au TP précédant, l'architecture de la base n'a pas du tout évolué, c'est pourquoi cette partie ne sera constituée que de rappels faisant référence à des explications données antérieurement.

Ainsi, la base s'appuie toujours sur le service NoSQL de Google appelé Datastore. L'interaction avec ce dernier ne s'effectue par directement avec l'API bas niveau que

propose Google, mais avec la bibliothèque Objectify. Cette bibliothèque Java permet en quelque sorte de faire de l'ORM avec le Datastore.

De ce fait, deux classes ont été modélisées (figure 1). Celles-ci sont rendues persistantes grâce à quelques annotations placées dans le code de leurs déclarations. Ainsi, une instance de type Livre ou de type Auteur peut être stockée dans le Datastore en quelques lignes grâce à l'API d'Objectify.

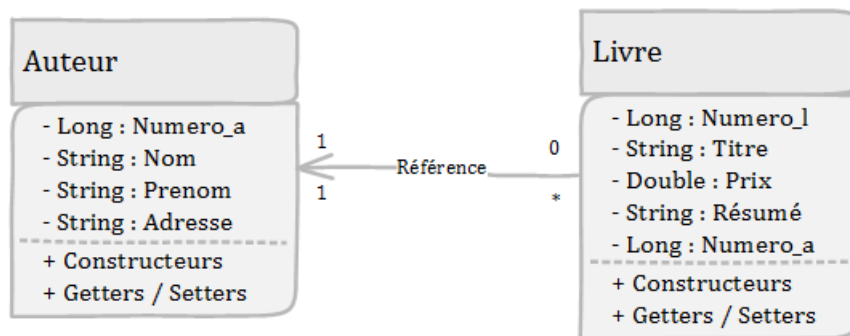


FIGURE 1: Classes métiers persistantes

Comme le NoSQL ne fournit pas de mécanisme permettant de lier deux types d'entités, à la manière des clefs étrangère en SQL, le lien entre les livres et les auteurs doit être géré manuellement (via du code) par les développeurs.

Une fois que le problème d'interaction avec la base est résolu, il faut prévoir toute la logique applicative que le serveur va devoir fournir aux clients. Cette logique repose sur un ensemble de méthodes exposées aux clients, et que ces derniers peuvent appeler de manière distantes. La partie suivante explique ce procédé dans le cadre d'une application GWT.

## 2.2 SUPPORT DES RPC AVEC GWT

Une application GWT est généralement constituée de deux choses, une partie cliente et une partie serveur. La partie cliente va être exécutée via le navigateur web de l'utilisateur, tandis que la partie serveur, sera exécutée sur un serveur Java EE destiné à servir les clients.

Ainsi, pour communiquer, le client va devoir effectuer des requêtes HTTP sur le réseau pour accéder aux services offerts par le serveur. Le traitement de ces requêtes par le serveur s'effectue via une servlet, mais une servlet un peu particulière.

En effet, GWT fournit une classe spécifique à son support des RPC baptisée `RemoteServiceServlet`. Lorsqu'un développeur souhaite implémenter la partie qui va traiter les RPC de son application, il doit commencer par créer une classe qui va hériter de ce composant.

Ensuite, pour que le client et le serveur s'entendent sur l'ensemble des méthodes exposées, il faut définir une première interface que la servlet devra implémenter. Ici, elle est baptisée, `BiblioService` (figure 2).

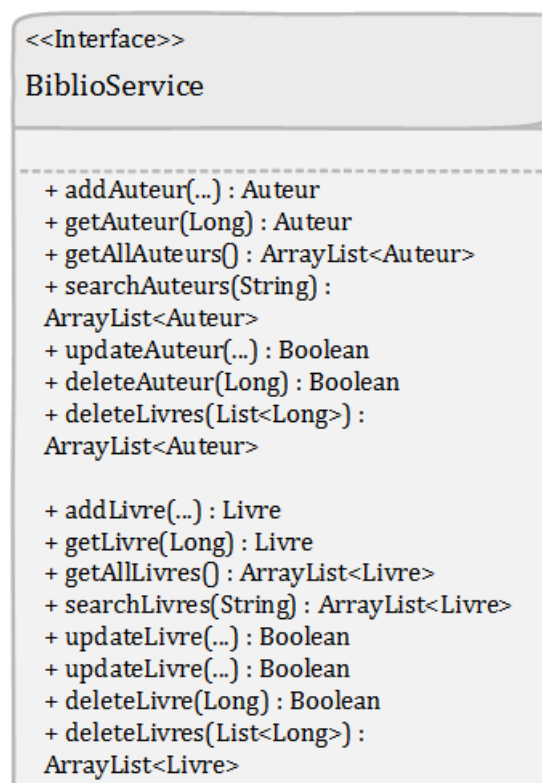


FIGURE 2: Interface `BiblioService`

Mais ce n'est pas tout, l'une des particularité de GWT est que le support des RPC est asynchrone, c'est-à-dire qu'un client qui fait appel à l'une des méthodes exposées par le serveur ne sera jamais bloqué par celle-ci. Par conséquent, il est nécessaire de définir une seconde interface dite asynchrone, qui reprend les prototypes des méthodes de `BiblioService` en les modifiant légèrement (figure 3).



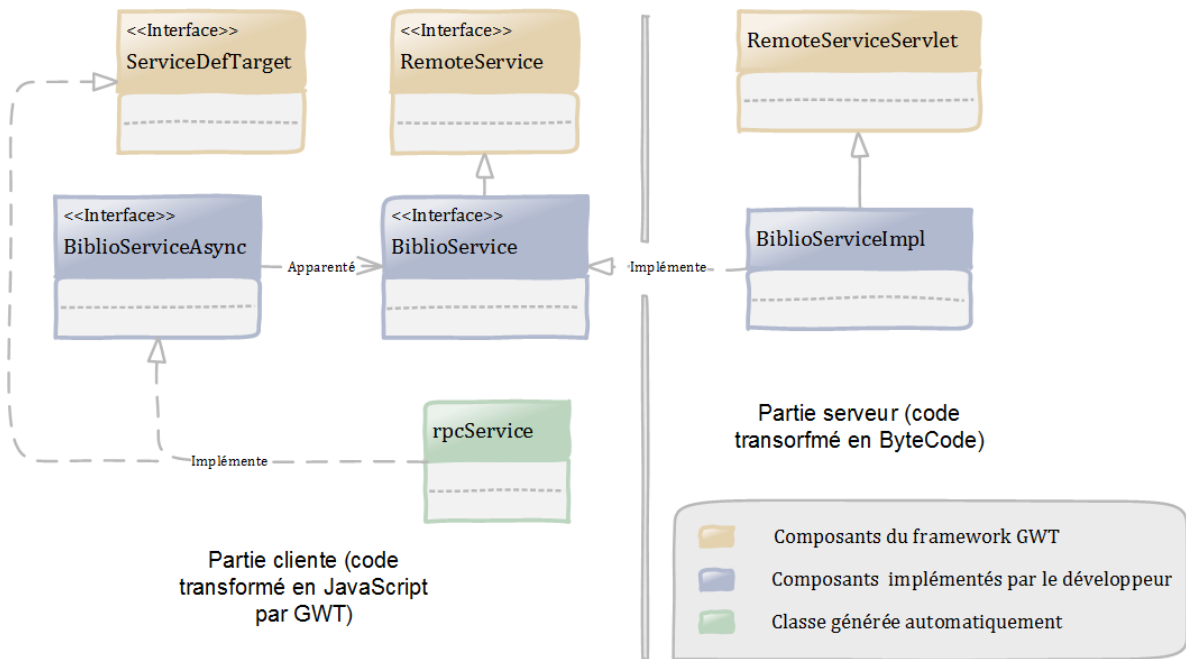


FIGURE 3: Architecture RPC

Dans l'implémentation réelle de l'application, la frontière de communication entre le client et le serveur est définie par l'interface `BiblioService`. A partir de cette dernière, l'interface asynchrone qu'utilise le client peut elle aussi être définie. Elle porte le nom de `BiblioServiceAsync`. Côté serveur, l'implémentation de la remote servlet est réalisée par la classe `BiblioServiceImpl` qui au passage implémente aussi l'interface `BiblioService`.

Seules ces trois classes sont nécessaires à la mise en place de RPC avec le framework GWT. A partir de ce moment, il ne reste plus qu'à lancer le serveur Java EE en chargeant la servlet codée précédemment, et côté client, il suffit d'instancier une classe qui fera office de proxy avec le serveur, baptisée ici `rpcService`. Cette classe n'a pas besoin d'être implémentée, elle est directement créée à la volée par le framework en se basant sur les méthodes de l'interface `BiblioService`.

## 2.3 WORKFLOW DU MÉCANISME DES RPC AVEC GWT

La partie précédente a permis de révéler l'architecture pour le support des RPC dans sa globalité, mais elle n'explique en rien le déroulement d'un de ces appels.

Tout d'abord, il faut préciser un point important du système. Le client et le serveur vont bien entendu devoir s'échanger des données afin de réaliser leurs tâches respectives. Dans le TP précédant, cet échange été réalisé grâce au protocole SOAP. Ici, il n'en n'est rien. GWT permet aux deux applications, cliente et serveur, de s'échanger directement des objets Java. La seule contrainte imposée étant que ces objets soient sérialisable.

Dans le cadre de cette application, les principaux objets que vont s'échanger les deux parties du système vont être des objets de type Livre ou de type Auteur. Voilà pourquoi ces deux classes doivent être déclarées sérialisable, à l'aide d'une annotation placée dans le code de leurs déclarations respectives. Dès lors, même un échange d'une liste de livres ou d'auteurs ne pose pas de problèmes car celui-ci s'effectuera grâce à une instance d'ArrayList elle même sérialisable.

Une fois ce problème résolu, le flux de données qui circule entre le client et le serveur est très simple à comprendre. Tout d'abord, le client va envoyer une requête au serveur par l'intermédiaire du composant rpcService vu précédemment. Ce dernier, va transmettre la requête HTTP au serveur et rendre la main au client. De cette manière, l'interface du client ne sera jamais gelée en attendant la réponse du serveur. Pendant ce temps là, la classe BiblioServiceImpl va se charger d'effectuer la requête, comme par exemple, charger la liste des auteurs présents dans la base de données. Une fois le calcul de la requête terminée, il va retourner un ArrayList d'auteurs que le framework va se charger de sérialiser et de renvoyer au client (figure 4).

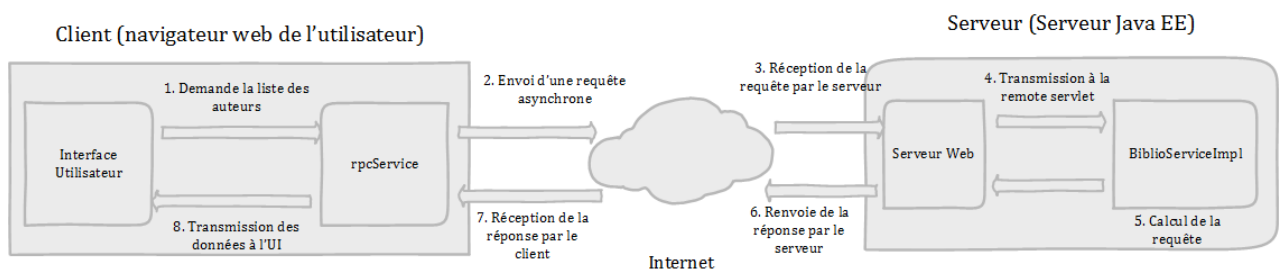


FIGURE 4: Demande de la liste des auteurs via le mécanisme des RPC

Comme le composant rpcService implémente l'interface asynchrone de BiblioService, il est capable, lorsqu'une réponse en provenance du serveur arrive, de transmettre les

informations contenues dans cette réponse au client. L'interface du client n'a plus qu'à être rafraichie, et les données en provenance du serveur se retrouvent affichées dans le navigateur de l'utilisateur. Ce mécanisme repose en réalité sur l'échange de requêtes AJAX entre le client et le serveur, mais la complexité de ce type de fonctionnement est masqué par l'API de GWT.

## CLIENT GWT

---

### 3.1 L'ARCHITECTURE MVP

### 3.2 MISE EN OEUVRE DU MODÈLE MVP DANS LE TP

## RÉSULTATS ET TESTS

---

## CONCLUSION

---