SYM : Labo 2 - Protocoles applicatifs

Auteurs : Julien Béguin, Robin Cuénoud & Gaëtan Daubresse
Date : 15.11.2020
Classe : B

Introduction

Ce deuxième laboratoire consiste à implémenter 5 manières différentes de communiquer avec un serveur applicatif en se basant sur le protocole HTTP. Ces 5 façon sont les suivantes: transmission asynchrone, transmission différée, transmission d'objet (JSON et XML), transmission GraphQL et transmission compressée.

Réponses aux questions

Traitement des erreurs

> Les classes et interfaces SymComManager et CommunicationEventListener, utilisées au point 3.1, restent très (et certainement trop) simples pour être utilisables dans une vraie application : que se passe-t-il si le serveur n'est pas joignable dans l'immédiat ou s'il retourne un code HTTP d'erreur ?
> Veuillez proposer une nouvelle version, mieux adaptée, de ces deux classes / interfaces pour vous aider à illustrer votre réponse.

Si le serveur n'est pas joignable l'instance de HttpURLConnection va générer une exception. Elle sera traitée par le bloque catch et la StackTrace sera affichée dans la console. Depuis l'application mobile l'utilisateur ne se rendra pas compte de cette injoignabilité et ne recevra simplement pas de réponse du serveur.

Nous pourrions améliorer le comportement en vérifiant au préalable si le téléphone est connecté à internet et en remontant le message d'erreur à l'utilisateur.

Si le serveur renvoie un code d'erreur il y'a également un exception qui est levée.

Une amélioration que nous pourrions imaginer consisterai à remonter le code d'erreur à l'utilisateur ou d'effectuer un traitement différents suivant le code d'erreur retourné.

Authentification

> Si une authentification par le serveur est requise, peut-on utiliser un protocole asynchrone ? Quelles seraient les restrictions ? Peut-on utiliser une transmission différée ?

Une authentification permet de valider l'identité d'un utilisateur. Cette étape est nécessaire sur certains site/application pour accéder à des **ressources réservées** à un certain rôle ou utilisateur.

De ce fait, il n'est pas possible de "laisser passer" un utilisateur en attendant la réponse du serveur lors de l'authentification car cela causerai un problème de sécurité. Les ressources réservées serait alors vulnérable.

Ainsi, il est absolument nécessaire d'attendre la réponse du serveur avant de valider l'authentification.

> Peut-on utiliser un protocole asynchrone ? Quelles seraient les restrictions ?

En théorie oui, c'est possible. Mais l'expérience utilisateur sera mauvaise car il devra attendre la réponse du serveur pour accéder aux ressources réservées.

Donc il vaut mieux éviter un protocole asynchrone pour l'authentification.

> Peut-on utiliser une transmission différée ?

En plus des problèmes d'expérience utilisateur cité plus haut, une transmission différée nécessiterait de stocker les informations de connexion en local avant de pouvoir les envoyer sur le serveur. Cela peut poser des problèmes supplémentaires en terme de sécurité.

Il vaut donc mieux d'éviter cette solution également (même si possible en théorie).

Threads concurrents

> Lors de l'utilisation de protocoles asynchrones, c'est généralement deux threads différents qui se répartissent les différentes étapes (préparation, envoi, réception et traitement des données) de la communication. Quels problèmes cela peut-il poser ?

Si les deux threads partagent des mêmes ressources, il pourrait y avoir des problème de concurrences. Il faudrait donc protéger ces ressources partagées à l'aide de verrous. Il est également possible que l'ordre d'exécution des threads puisse varier en fonction des décisions prisent par l'ordonnanceur. Il est donc important qu'une synchronisation soit fait entre les deux threads.

Écriture différée

> Lorsque l'on implémente l'écriture différée, il arrive que l'on ait soudainement plusieurs transmissions en attente qui deviennent possibles simultanément. Comment implémenter proprement cette situation ? Voici deux possibilités :
>
> - Effectuer une connexion par transmission différée
> - Multiplexer toutes les connexions vers un même serveur en une seule connexion de transport. Dans ce dernier cas, comment implémenter le protocole applicatif, quels avantages peut-on espérer de ce multiplexage, et surtout, comment doit-on planifier les réponses du serveur
>   lorsque ces dernières s'avèrent nécessaires ? Comparer les deux techniques (et éventuellement d'autres que vous pourriez imaginer) et discuter des avantages et inconvénients respectifs.

L'utilisation d'une connexion par transmission différée sera plus propice à l'envoi de fichier volumineux. Elle est également à privilégier lorsque la connexion n'est pas stable.

L'avantage du multiplexage est que cela permet de traiter un nombre supérieur de requête comparé à la transmission différée. Cela facilite également le partage d'une même ressource entres différentes activités.

Transmission d’objets

> a. Quel inconvénient y a-t-il à utiliser une infrastructure de type REST/JSON n'offrant aucun service de validation (DTD, XML-schéma, WSDL) par rapport à une infrastructure comme SOAP offrant ces possibilités ? Est-ce qu’il y a en revanche des avantages que vous pouvez citer ?

L'utilisation de JSON est légèrement plus léger que le XML. Nous nous somme également rendu compte que le sérialisation et désérialisation se faisait plus simplement qu'en XML. Notamment grâce à l'utilisation de librairie comme Gson(). Il existe également un librairie appelée Jackson pour sérialiser du XML mais nous avons eu plus de mal à la prendre en main et avons donc d'effectuer la sérialisation/déserialisation "à la main".

> b. L’utilisation d’un mécanisme comme Protocol Buffers5 est-elle compatible avec une architecture basée sur HTTP ? Veuillez discuter des éventuelles avantages ou limitations par rapport à un protocole basé sur JSON ou XML ?

Son avantage premier est sa légèreté comparé au JSON et XML. Il est également possible d'effectuer des vérifications sur sa structure de donnée. Par contre, il est très difficile voir impossible de le lire sans un désérialiseur.

> c. Par rapport à l’API GraphQL mise à disposition pour ce laboratoire. Avez-vous constaté des points qui pourraient être améliorés pour une utilisation mobile ? Veuillez en discuter, vous pouvez élargir votre réflexion à une problématique plus large que la manipulation effectuée.

Nous pouvons observer que certaines grandes requêtes peuvent consommer beaucoup de donnée. Une solution pourrait être de garder en cache les réponses afin d'économiser les ressources. Il faudrait néanmoins vérifier que la base de donnée n'a pas changé entre temps. Nous pourrions également faire de la compression.

Transmission compressée

> Quel gain de compression (en volume et en temps) peut-on constater en moyenne sur des fichiers texte (xml et json sont aussi du texte) en utilisant de la compression du point 3.4 ? Vous comparerez plusieurs tailles et types de contenu.

Voici les résultats d'un test de transmission effectué avec et sans compression sur un texte en latin de N caractères :

Nombre de caractères sans compression	**Nombre de caractères avec compression**	**Gain de taille**	**Temps sans compression**	**Temps avec compression**	**Gain de temps**
10	24	**-58%**	0.66s	0.7s	**-6%**
100	78	**28%**	0.83s	0.77s	**8%**
1000	413	**142%**	0.95s	0.9s	**6%**
10000	2854	**250%**	2.18s	1.13s	**93%**

On peut constater que plus le nombre de caractères est grand, plus le gain en taille et en temps est important. Néanmoins, il faut que le contenu fasse plusieurs dizaine de Ko pour remarquer un gain de temps significatif.

A l'opposer, lorsque le nombre de caractère est très faible (< 50 caractères), le gain est négatif. Cela signifie que la taille du texte compressé est plus grande que la taille du texte original. Dans ce cas la, il est préférable de ne pas compresser le contenu.

A noter que nous avons également effectué ce test avec du contenu de type xml et json et les résultats sont similaires.

Problèmes connus

- Sur l'activité GraphQL, il est nécessaire de cliquer sur les ListViews pour actualiser leurs contenus.
- Sur l'activité différée, il est nécessaire de recharger l'activité pour envoyer un deuxième message.

Conclusion

Ce laboratoire nous a permis de découvrir différents mode de transmission entre une application mobile et un serveur. Nous avons du appréhender des problématiques différentes que celle rencontrées lors de développement d'application pour ordinateur.