DI コンテナを利用した エンタープライズアプリケーションにおける プロトコル依存性の分離

齋藤 伸也‡

株式会社オージス総研‡

Separating Protocol Dependency in Enterprise Applications

Shinya Saito‡

Osaka Gas Information System Research Institute‡

1. はじめに

近年、エンタープライズアプリケーションの構築では、DI コンテナを利用したフレームワークが活用されている。DI コンテナはアプリケーションの階層を分離した設計を可能にする。

また、現在のエンタープライズアプリケーションは他のシステムと連携することが求められており、連携のために ESB(Enterprise Service Bus)やアダプタと呼ばれるミドルウェアを利用することが多い。本稿では DI コンテナと ESB を使いプロトコル依存性を分離したアーキテクチャについて述べる。

2. プロトコル依存性の分離と問題

2.1. プロトコル依存性

ここで言うプロトコル依存性とは、「OSI の 7 階層モデルで表現されるプロトコル及び特定システムの API への依存」を指す。

プロトコルに依存したアプリケーションには以下のような問題点がある。

① アプリケーションの仕様変更時

仕様変更に影響を大きく受け、テストを行うには依存先のシステムが必要になる。

② 外部連携先の変更時

外部連携先が変更された場合、外部連携部分を新たに作りなおす必要がある。

2.2. ESB の活用

上記 2 つの問題を解決するため一般的にはアダプタを利用し、プロトコルに依存した処理をアプリケーションから分離するアーキテクチャを構成することが多い。

しかし 2.1 節の問題は解決されるが、あらたにアプリケーションが特定のアダプタに依存する処理が発生する。

例えばアダプタを利用した連携部分のクラス図は図 1 のようになる。

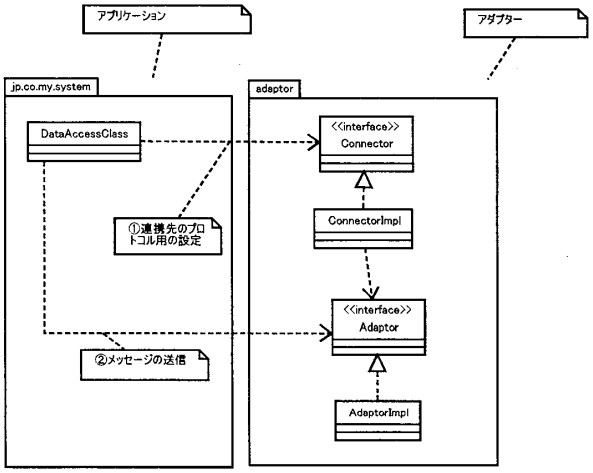

図 1:アプリケーションとアダプタの依存関係

図 1 ではアプリケーションがアダプタと連携するクラスを「DataAccessClass」と定義する。「DataAccessClass」の処理の流れは次のようになる。

① 連携先のプロトコル用の設定を「Connector」に行う。

② 「Adaptor」の API を使って外部連携先にメッセージを送信する。

図 1 からアプリケーションがアダプタに依存しているのがわかる。この依存を回避するために ESB を利用することが多い。ESB は連携・統合のために多くの機能を備えている。しかし、複雑なミドルウェアであるため開発者が ESB を利用するには新しい環境を用意し、複雑な機能を学習しなければならない。

本稿ではこれらの問題を解決するためにモジュール S2Mule[1]を開発した。

3. S2Mule

3.1. 概要

S2Mule は代表的な DI コンテナである Seasar2[2]をベースに開発した。連携する ESB として Mule[3]を利用する。Mule は米 MuleSource 社によって OSS(Open Source Software)として開発・提供されている軽量な ESB である。

S2Mule には次のような特徴がある。

- プロトコルに依存した設定は dicon ファイル(Seasar2 の設定ファイル)で行う。このため Mule 依存の XML 文法を学習することなく利用することができる。
- DI、AOP を利用して ESB 依存の処理を分離できる。
- アプリケーションの実装をほとんど変更することなく、ESB の機能が利用できる。

3.2. 機能

S2Mule は Java アプリケーションから Mule を簡単に利用できるようにしたモジュールである。S2Mule は以下の機能を提供する。

- POJO から ESB を利用してメッセージを送信する機能。
- 外部から受信したメッセージを POJO で受信する機能。

3.3. アーキテクチャ

S2Mule を利用したときのアプリケーションと ESB の依存関係は図 2 に示す。

図 1 で示したアダプタに依存した処理①はソースコードから dicon ファイルに分離されている。dicon ファイルの設定を書き換えることでソースコードの変更なしに、外部連携先を切り替えることができる。②のメッセージ送信を行う部分の処理は同期/非同期によって処理が異なる。

- 同期通信

Seasar2 の AOP を使って、アプリケーションと S2MuleSender の間にインターセプタが入る。インターセプタを利用することで、アプリケーションから ESB への通信を抽象化している。アプリケーションは S2Mule を意識することなく、メッセージ交換を行うことができる。

- 非同期通信

S2Mule の API を利用してメッセージを送信する。アプリケーションから S2MuleSender への依存性を低くするため、Seaser2 の DI を利用している。

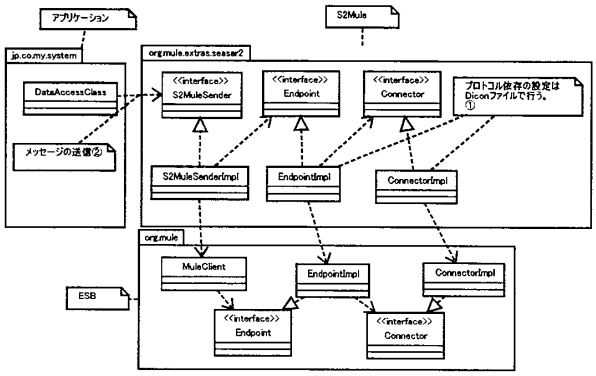

図 2: S2Mule を利用したときのアプリケーションと ESB の依存関係

4. 考察

S2Mule は DI コンテナの機能を使うことで ESB 依存の処理を分離し、複雑な ESB の機能の利用が容易にできると考える。また S2Mule はプロトコルに依存した処理をアプリケーションから分離するアーキテクチャを実現するのに有効だと考えられる。

5. まとめ

本稿では DI コンテナを利用し、プロトコルに依存した処理を ESB に分離するアーキテクチャを実現するモジュール S2Mule を紹介した。

しかし S2Mule は以下のような課題が未解決である。

- メッセージ交換(同期/非同期)の処理は分離できていない。
- S2Mule 自体が特定の ESB(Mule)に依存にしてしまっている。

今後はこれらの課題を検討しながら、モジュールの改善を行っていく予定である。

6. 参考文献

[1]S2Mule
http://www.mulesource.org/display/SEASAR2/Home
[2]Seasar2
http://www.seasar.org/
[3]Mule Open Source ESB
http://mule.mulesource.org/