マルチテナント対応 アプリケーション開発を効率化する「SystemDirector Enterprise」

小林 茂憲・小泉 健

要　旨

複数のサービス利用企業（テナント）に対し、アプリケーションをサービスとして提供する際には、リソースを効率的に利用し、サービス提供コストを抑える必要があります。そのため複数のテナントを1つのアプリケーションで対応する、アプリケーションのマルチテナント化が求められています。本稿では、SDE（SystemDirector Enterprise）が提供するマルチテナント開発支援機能を利用した、マルチテナント・アプリケーション開発手法について説明します。

キーワード

●SaaS ●クラウド ●マルチテナント ●SystemDirector Enterprise

1. はじめに

Web、仮想化技術などIT技術の急速な発展、及びビジネスユーザのオンデマンド型でのアプリケーション利用のニーズの高まりから、インターネット経由でソフトウェアをサービスとして提供するSoftware as a Service（SaaS）が急速に発展してきています。

複数のサービス利用者（テナント）に対し、アプリケーションをSaaSの形態で提供する際には、リソースを効率的に利用し、サービス提供コストを抑える必要があります。そのため複数のテナントを1つのアプリケーションで対応できる仕組みが必要となります。このように、単一のシステムを複数のテナントで共有する方式をマルチテナント方式と呼びます。

本稿では、SDE（SystemDirector Enterprise）が提供する、マルチテナント・アプリケーション・フレームワークのアーキテクチャや開発手法について説明します。

2. マルチテナント方式

マルチテナントの方式には、**図1** の通り、シングルインスタンス方式とマルチインスタンス方式があります。インスタンスとは、アプリケーションが実行されているイメージであり、個々のテナント向け環境を収容する実体のことです。

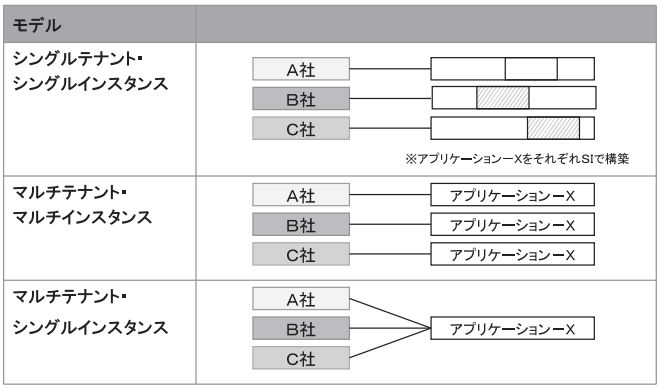

図1　テナントとインスタンスの関係

マルチテナントを実現するためには、アプリケーション及びデータを分離する必要があります。

アプリケーションの分離方式にはOSレベル、ミドルウェアレベル、アプリケーションレベルで分離する方式があります（**図2**）。SDEではシングルプロセスパターンの分離を実現するフレームワーク及び開発支援ツールを提供します。

データの分離方式は、DataBase Management System(DBMS)のインスタンスレベルで分離する方式、スキーマと呼ばれるDBMSの論理分割単位で分離する方式、テーブルのカラムを利用して分離する方式があります（**図3**）。SDEでは個別DB

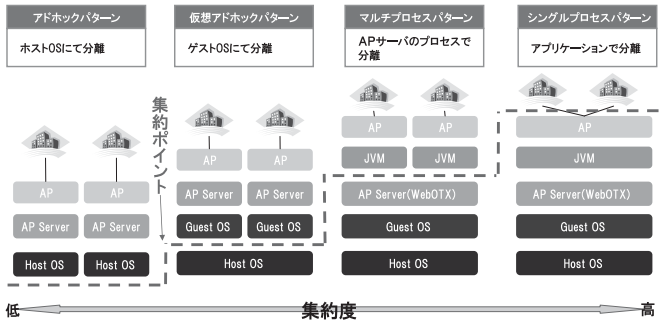

図2　アプリケーションの分離パターン

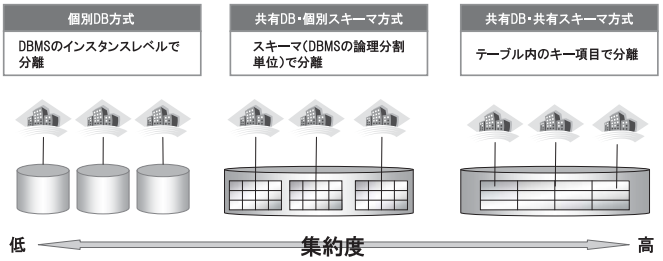

図3　データの分離パターン

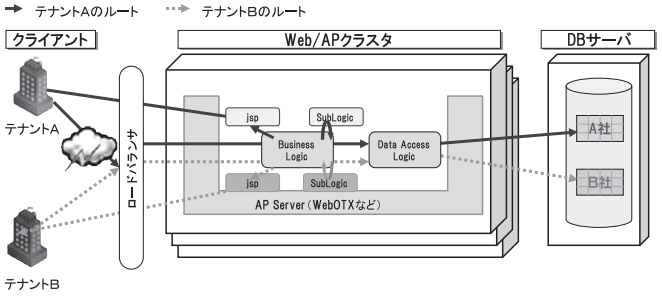

図4　マルチテナント・アプリケーション例

方式、共有DB・個別スキーマ方式の分離をサポートします。

図4 は、アプリケーションとデータをマルチテナント化したシステムの例です。アプリケーションの分離は、シングルプロセス方式、データの分離は、共有DB・個別スキーマ方式を採用した例となっています。

アプリケーションやデータをどのパターンで分離するかはアプリケーションのサービスレベルとリソース活用効率（テナント集約度）のトレードオフとなりますが、サービス提供側の立場で見ると、最終的には少ないリソースでより多くのテナントをサポートして、投資回収率を向上させることが重要となります。以降では少ないリソースでサービス提供が可能なシングルプロセスパターンを、SDEで実現する手法について解説します。

3. マルチテナント開発時の課題とSDEでの提供機能

シングルプロセス方式でのマルチテナント化を実現する場合には、あらかじめシングルプロセス方式でマルチテナント化を実現できるように設計をする必要があります。そのため以下を検討する必要があります。

(1) マルチテナント・アプリケーション開発プロセス

テナント共通処理とテナント個別処理を分離し、管理するための開発プロセス

(2) カスタマイズ性を持ったアーキテクチャ

マルチテナント・アプリケーションを開発するためのアーキテクチャ

SDEでは、マルチテナント・アプリケーション開発を支援する以下の機能を提供します。

- マルチテナント開発ガイド
- マルチテナント開発支援ジェネレータ
- マルチテナント・フレームワーク

詳細を **図5** に示します。

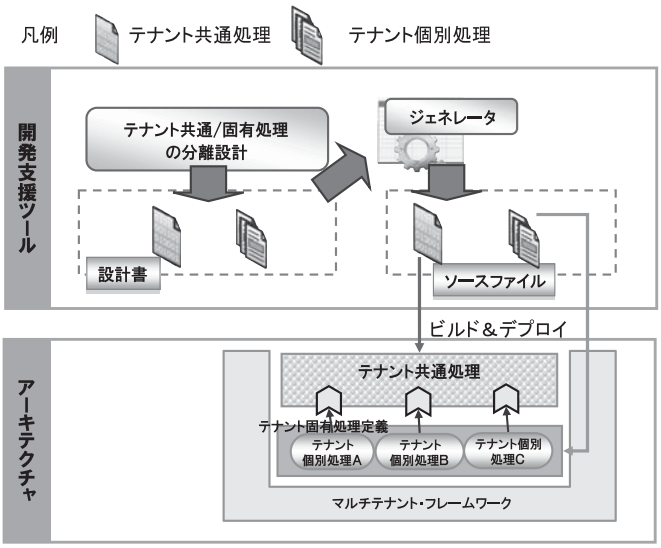

図5　SDEのマルチテナント・アプリケーション開発支援機能

SDEでは、テナント共通処理とテナント個別処理とを分離して設計できるよう、設計書のテンプレートを提供しています。またその設計書に記載した情報から、マルチテナント・フレームワークに従ったソースコードを自動生成する機能（ソースコード・ジェネレータ）を提供しています。マルチテナント・アプリケーション開発プロセスに従ったこれらの開発支援ツール、フレームワークにより、マルチテナント・アプリケーションの効率的な開発を支援します。

3.1 マルチテナント・アプリケーション開発プロセス

マルチテナント・アプリケーションを開発するに当たって、テナント共通処理とテナント個別処理を分離して開発する場面では、「プロダクトライン開発」の考え方を取り入れることで、効率的に開発を進めることが可能です。

プロダクトライン開発では、アプリケーションの変更できる特性を「可変性」と呼び、「可変性」のうち変更可能な箇所を「可変点」、変更のための選択肢を「変異体」と呼びます。

これをマルチテナント・アプリケーションの観点で考えると、「可変性」を「テナント個別処理」と置き換えることができます。具体的には、テナント個別処理に入れ替え可能なポイントを「可変点」、テナント個別処理の実態を「変異体」とみなすことができます（**図6**）。

上記の「可変性」を管理し、テナント共通処理と、テナント個別処理を開発プロセスにてドキュメント化し、開発プロセス全体を通して管理することが重要となります。

3.2 マルチテナント・フレームワーク

「プロダクトライン開発」では、可変性を標準的に固定するアーキテクチャを採用することでアプリケーションの開発・保守効率を向上することを推奨しています。マルチテナント・アプリケーションも同様に、テナント共通処理に、テナント個別処理を取り込むことができる標準的なアーキテクチャを適用することで、後述する「標準アプリケーションの開発」と「カスタマイズ・アプリケーションの開発」を分離することができ、開発、保守効率を向上させることが可能となります。

SDEでは、DI（Dependency Injection：依存性注入）コンテナの「依存性注入」という特徴を利用し、Spring Frameworkを用いたマルチテナント・フレームワークを提供します。依存性注入とは、コンポーネント間の依存関係の解決を、コンポーネント自身でなく、第三者となるDIコンテナが解決する手法です。

Spring Frameworkを用いたマルチテナント・フレームワークのイメージを**図7**に示します。

マルチテナント・フレームワークにDIコンテナを利用することで、下記の特徴を持った、マルチテナント・フレームワークを実現しています。

- モジュール（クラス）のコンポーネント化
- コンポーネント間の依存性の排除
- カスタマイズ情報の外部化

SDEでは前述したDIコンテナの機能を利用して、テナントごとにトランザクションを切り替える仕組み（シナリオコントローラ機能）を提供しています。

シナリオコントローラでは、テナントごとに「テナント実行シナリオ」を定義することができ、全テナント共通で利用するシナリオでなく、テナント個別に定義が必要なトランザクションについては、ビジネスロジックコンポーネント（SDEでの名称はSubLogic）を差し替えることで、テナント個別処理を実装することが可能となります（**図8**）。

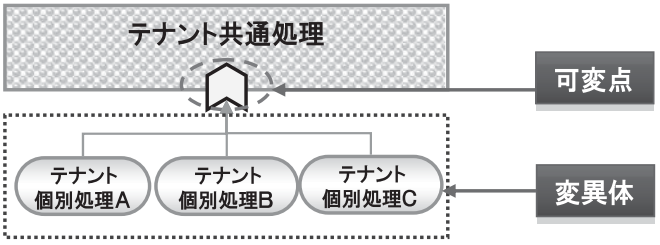

図6　プロダクトライン開発の「可変性」とテナント個別処理の関係

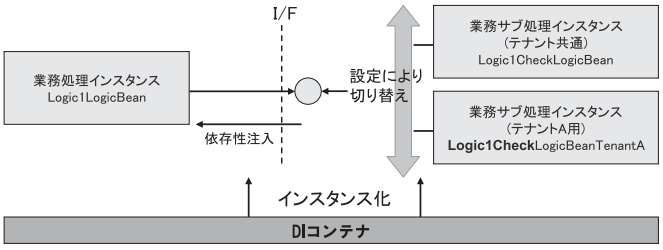

図7　Spring Frameworkによるカスタマイズ

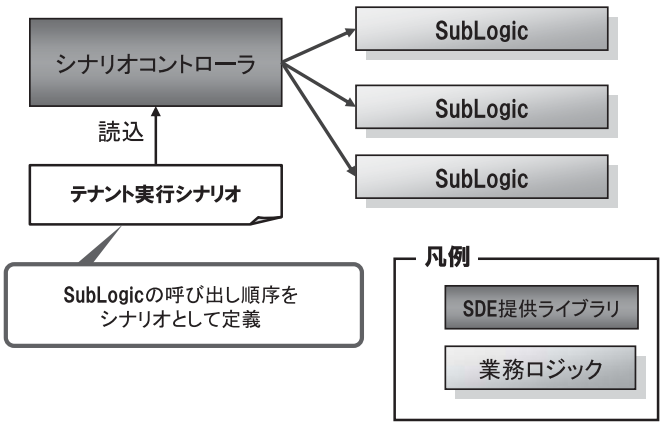

図8　シナリオコントローラ

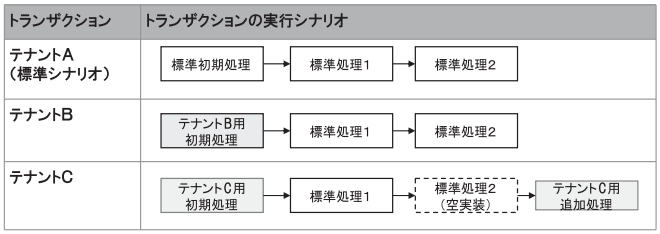

図9　テナントごとのトランザクション切り替え

図9 の例では、テナント個別処理がない、テナントAのトランザクションの場合には、標準的なトランザクションを実行します。テナント個別のカスタマイズが定義された、テナントB、テナントCの場合には、テナント共通部をテナント個別処理に差し替えてトランザクションが実行されます。

4. SDEを利用したマルチテナントアプリケーション開発

マルチテナント・アプリケーション開発は「標準アプリケーションの開発」と「カスタマイズ・アプリケーションの開発」の2つに分けて考えることができます。

(1) 標準アプリケーションの開発

サービスとして提供する標準的なアプリケーションを開発するプロセスであり、SaaS事業者が担当します。

(2) カスタマイズ・アプリケーションの開発

標準アプリケーションの開発成果物を元に、テナント個別のカスタマイズ処理を開発するプロセスであり、SaaS事業者及びSSP（SaaS Solution Provider）が行います。

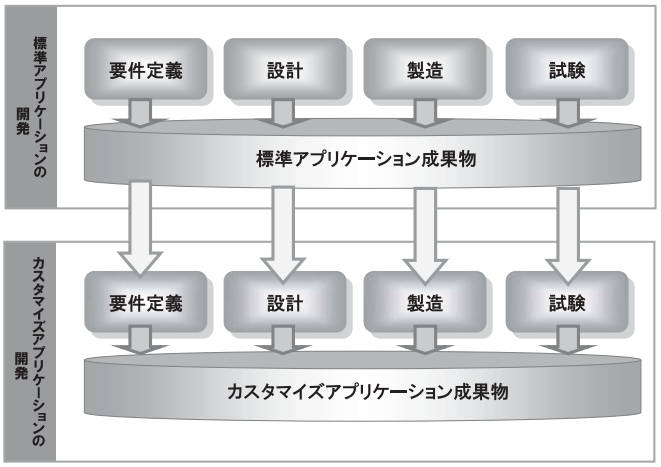

図10　マルチテナント・アプリケーションの開発プロセス

このように開発プロセスを分けることで、堅牢な標準アプリケーションの構築と、特定テナント向けカスタマイズの短期開発という2つの関心事を分離することができます（**図10**）。

4.1 標準アプリケーションの開発

開発方法論において定義された開発プロセスにしたがって、マルチテナントを想定した設計・実装を行い、標準アプリケーションを開発します。通常の開発との違いはマルチテナント化の設計、つまり「可変性」の管理になります。アプリケーションの機能要件、非機能要件を元に標準アプリケーションの可変点（可変性の対象）を定義します。

業務アプリケーション開発を習得した開発者がマルチテナント・アプリケーションを容易に開発できるよう、SDEが従来から提供している業務アプリケーション向け開発方法論の開発プロセスをそのままに、マルチテナント・アプリケーション開発ガイドを新たに提供しています。

4.2 カスタマイズ・アプリケーションの開発

標準アプリケーションをもとに、特定テナント向けにカスタマイズを行う開発です。

SDEでは、テナント個別処理の開発は、テナント共通処理

とは別の設計書を用いて行います。処理仕様のカスタマイズについては、以下の観点からカスタマイズを行います。

- SubLogic の呼び出しの追加・削除
- SubLogicの呼び出し順序の変更
- SubLogic の実装の変更

5. むすび

本稿では、SDEでのマルチテナント・アプリケーション開発手法について紹介しました。現在、SaaSビジネスは急激なスピードで拡大しており、NECにおいてもSaaSアプリケーションの共通基盤の整備に注力するなど、対応を急いでいます。今後もSDEでは最新技術を取り入れ、SaaSアプリケーション開発を支援する機能の強化、及びSaaS基盤サービス「RIACUBE/SP」との連携の強化を行っていきます。

執筆者プロフィール

小林 茂憲
ソフトウェア生産革新部
マネージャー

小泉 健
ソフトウェア生産革新部
主任

●システム構築統合開発基盤　SystemDirector Enterprise

関連URL

http://www.nec.co.jp/cced/SystemDirector/SystemDirectorEnterprise/index.html