O-007

データを社外に預けずに利用可能な SaaS アプリケーション構築方式「データ保護モデル」のサーバサイド実行機能

Server-side execution feature of Data-Detached Application Model

乾　敦行†　　小島　剛†

Atsuyuki Inui　　Go Kojima

1. はじめに

SaaS は自社内にシステムを持つ必要がなく，低コストですぐに利用できるメリットがある．しかし，社外にデータを預けるのが不安であるという顧客が多い．ある調査 1) によると，SaaS に対する不安として，社外にデータを預けるためセキュリティが問題であると答えた人が最も多かった．そこで我々は，データを社外に預けずに利用可能な SaaS アプリケーション構築方式「データ保護モデル」の研究を推進している．データ保護モデルアプリケーションは従来の Web アプリケーションとアーキテクチャが異なり，既存のフレームワークを使った開発ができない．そのため，ゼロから開発する必要があり，開発コストが高くなるという課題があった．これを解決するため，我々はデータ保護モデル向けフレームワークの開発を行った．

2．データ保護モデルの概要

データ保護モデルは，アプリケーションと機密データを分離することにより，機密データを組織外に出さない Web アプリケーションを実現する点が特徴である（図 1）．

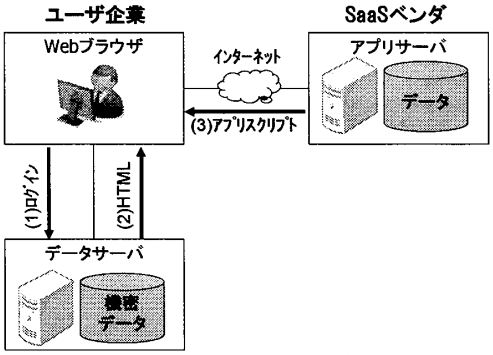

図1．データ保護モデル

SaaS ベンダはアプリスクリプト（JavaScript☆プログラム）を配布するアプリプロバイダを用意し，ユーザ企業は機密データを保持するデータプロバイダを用意する．プログラムが動くまでの過程を述べる．(1)ユーザはデータプロバイダにアクセスし，(2)HTML を得る．(3)この HTML にはアプリプロバイダからアプリスクリプトをロードする命令が記述されているため，Web ブラウザは自動的にアプリスクリプトを取得する．以後プログラムはこのアプリスクリプトに従って Web ブラウザ上で動作する．ただし後述するアプリプロバイダでの実行機能により，アプリプロバイダでプログラムを実行することも可能である．

データアクセスは Web ブラウザとデータプロバイダ間，Web ブラウザとアプリプロバイダ間，アプリプロバイダ内の 3 種類がある．それぞれ XMLHttpRequest，JSONP，JDBC を用いる．XMLHttpRequest は Web ブラウザに組み込まれた通信用 API であり，JSONP はクロスドメインで通信するための方法，JDBC は Java でデータベースに接続する API である．

3．データ保護モデルフレームワークの概要

生産性が高い Web アプリケーションフレームワークである Ruby on Rails に注目し，その MVC アーキテクチャ，コード自動生成，Convention over Configuration（設定より規約）の考え方を参考にフレームワークを開発した．本フレームワークはコンパイラ，実行時ライブラリ，コード自動生成機能からなる．アプリケーション開発者はコード自動生成機能によってモデル(M)，ビュー(V)，コントローラ(C) の雛形を作成し，コードを追加した後コンパイルを行う．コントローラはアクションの定義からなる．アクションとは，ユーザがリンクやボタンをクリックしたときに呼び出される関数のことである．

コンパイラは MVC のコードに対して，後述する変換を含む種々の変換を行った後，実行時ライブラリと結合し，アプリスクリプトを生成する．

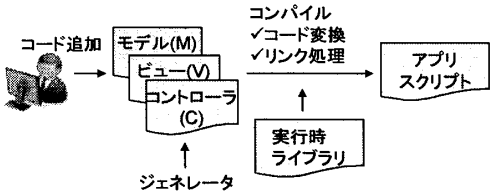

図2．アプリ開発の流れ

4．アプリプロバイダでのプログラム実行機能

4.1　要件と課題

データ保護モデルのアプリケーションは Web ブラウザ上で実行させることから，すべて JavaScript で記述するため，ユーザにソースコードが見えてしまい，SaaS ベンダのノウハウやアルゴリズムが隠せないという問題がある．本フレームワークはアプリプロバイダでプログラムを実行する機能を提供することで，ユーザにソースコードを

† 日立製作所システム開発研究所

☆米国およびその他の国における米国 Sun Microsystems,Inc.の商標または登録商標

見せないようにし，この問題を解決した．我々はこの機能を実現するにあたり，2 つの要件を定めた．

(1) アプリケーション開発時にはクライアント側で動くかサーバ側で動くかを意識せずにすむように，アプリプロバイダで実行することを後から設定変更できること

(2) 単一のプログラミング言語でアプリケーションを記述できるように，アプリプロバイダで実行するプログラムも JavaScript で記述できること

(1)の要件を満たすため，アプリ開発者はアプリサーバで実行するアクションも通常のアクションと同じように記述できるようにした．コンパイラが設定ファイルに従って，アプリプロバイダで動作させるアクションについてはアプリプロバイダへリクエストを送信するコードに変換する．変換方法は次節で述べる．アプリ開発者は設定ファイルにアプリプロバイダで実行したいアクションを記述する．

(2)の要件を満たすため，サーバ側では jrunscript で JavaScript のコードを実行することにした．jrunscript は jdk に付属している Java による JavaScript の実行処理系である．JavaScript プログラムから Java の API を呼び出すことができるため，JDBC によってデータベースへのアクセスが可能である．

4.2 変換方法

アプリケーション開発者が記述したコントローラはアプリプロバイダがアクセスできるディレクトリに配置する（図 3①）．その際，アプリプロバイダで動くプログラムはアプリプロバイダ内のデータにのみアクセス可能なため，アプリプロバイダのデータベースへ接続するための設定を追加する．アプリプロバイダは Web ブラウザからリクエストを受けると，このコントローラを jrunscript で実行する Web API を提供する．Web ブラウザに配信するアプリスクリプトの内，隠したいアクションはこの Web API を呼び出すコードに変換する（図 3②）．

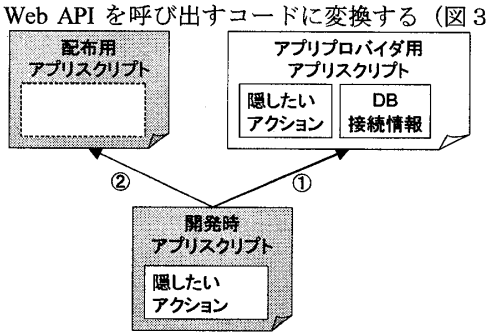

図 3 変換方法

変換例を示す．変換前のコントローラは図 4 のようになっている．3〜5 行目が show アクションの定義である．コンパイラはこのアクションの中身（4〜5 行目）を図 5 のように変換する．

params はこのアクションに渡されたパラメータを示す変数である．よって，サーバのプログラムにもこの内容を渡す必要があり，URL のパラメータに埋め込んで渡す処理を行う．$C.xhr は本フレームワークが提供する JSONP に対応した通信用の関数である．8〜10 行目がレスポンスを受け取るためのコールバック関数である．html という引数がレスポンスで，描画すべき HTML を表す文字列である．Web ブラウザは，本フレームワークが提供する render という関数でこの内容を描画する．

```
1: $R.create_controller("wikis").
2: define({
3:   show:function(){
4:     ...
5:     render({text:text});
6:   },
7:   ...
8: });
```

図 4 変換前のコントローラ

```
 1: var queries = [];
 2: for (var p in params)
 3:   queries.push(p+":'"+params[p]+"'");
 4: var query =
 5:   "params={"+queries.join(",")+"}";
 6: $C.xhr('http://test/wikis/show?'+query,
 7:   {dataType:"jsonp"})
 8: (function(html) {
 9:   this.render({text:html});
10: });
```

図 5 変換後のアクション

4.3. 動作検証

本フレームワークを用いてサンプルアプリとしてプロジェクト管理ツールを開発した．このアプリはプロジェクトの問題（チケット），Wiki，変更履歴，進捗状況を管理する機能を持つ．図 6 はこのうちのチケット管理画面を表している．サーバサイドで実行するアクションの場合，ユーザに配布されるアプリスクリプトのコードは図 5 のようにアプリプロバイダに実行させる処理のみが記述されており，ユーザに処理本体のアルゴリズムを隠蔽することができる．

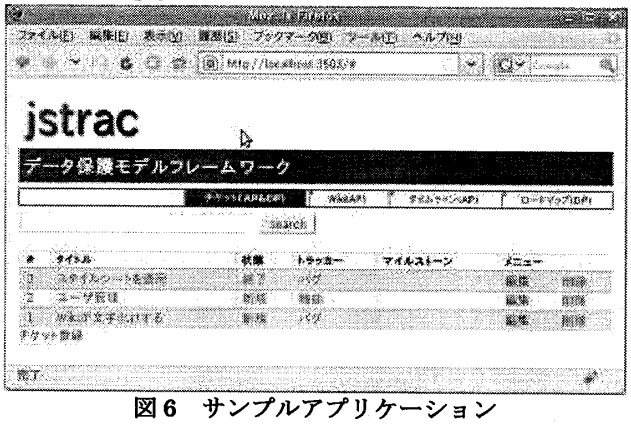

図 6 サンプルアプリケーション

5. まとめ

データ保護モデルにおいて，アプリプロバイダで実行する機能を持つフレームワークを設計・実装し，動作検証を行った．本フレームワークによって，単一のプログラミング言語で実行場所がクライアントであるかサーバであるか意識せずにアプリケーションを開発できることを確認できた．

参考文献

[1] ユーザー・アンケートで見えた，根強い「セキュリティ面の不安」(2007)．http://itpro.nikkeibp.co.jp/article/COLUMN/20070606/273780/?ST=cloud&P=2.