Ruby on Railsを用いたWebアプリケーション開発支援の研究

2006MI197 山田 博貴
指導教員　蜂巣 吉成

1　はじめに

近年 Web アプリケーションが普及しており，その開発方法として Ruby on Rails が注目されている．Ruby on Rails はラピッドプロトタイピングに基づく開発であり，scaffold という生成系を用いて Web アプリケーションの雛型を生成し，機能を追加して開発する．

Web アプリケーションは種類毎に共通の機能がある．また，Web アプリケーション開発企業などが種類の異なる Web アプリケーションを作成する場合でも，デザインやレイアウトなどを共通に利用する場合がある．scaffold は CRUD(Create, Read, Update, Destroy) のみ実行できる雛型しか生成せず，Web アプリケーションを開発する毎に，これらの共通に利用できる機能やデザインを追加するのは開発効率が悪い．

本研究では，scaffold を拡張して共通の機能やデザインが追加された雛型を生成する方法を提案する．ユーザは作成したい Web アプリケーションの種類と追加したい機能，デザインを指定することで，それらが反映された雛型が生成される．生成される雛型は，共通に利用できる機能やデザインが埋め込まれているので，それらを追加するコードを記述する必要がなく，従来の開発方法の機能を追加する手間が省ける．

2　Ruby on Rails の問題点

Web アプリケーション開発企業では，ブログ，SNS などの Web アプリケーションを複数の企業から請け負い，開発している．ブログのトラックバック機能など，各 Web アプリケーションには共通の機能が多くあるが，相手企業ごとに機能の一部をカスタマイズする必要がある．一つの企業向けに複数の Web アプリケーションを開発する場合，企業のロゴやヘッダなど，共通のデザインやレイアウトを採用することも多い．scaffold は CRUD のみ実行できるコードしか生成しないので，Web アプリケーションで共通に利用できる機能や，デザインを反映させる類似したコードを Web アプリケーションを開発する毎に記述するのは開発効率が悪い．

これらの解決策として scaffold を修正して CRUD 以外の機能が追加された雛型を生成する方法 [1] や，動的なモデルをベースにしたビューを生成する方法 [2] などがある．しかし，Web アプリケーション毎に scaffold で生成するためのコード修正が必要であるので，十分な解決策とはいえない．

3　拡張可能な scaffold の提案

2 節の問題を解決するために CRUD のみでなく，作成したい Web アプリケーションに共通する機能やデザインが追加された雛型を生成する方法を提案する．既存の Web アプリケーションを分析し，共通機能の整理を行った．また，実際に Web アプリケーションを作成し，scaffold を構成するファイルの分析，整理を行った．本研究では，これらの共通機能を部品として組み合わせて再利用できる枠組みを提供する．

3.1　Web アプリケーションの分析

Ruby on Rails の公式ページ [3] で紹介されている 71 個の Web アプリケーションを種類や機能，データの観点で分析した．Web アプリケーションは 8 種類で，機能は 9 種類，データは 7 種類に分類した．この関連の一部を図 1 に示す．左側はデータ，中央は Web アプリケーション，右側は機能である．また，実線は Web アプリケーションから機能，データへの利用関係を表す．

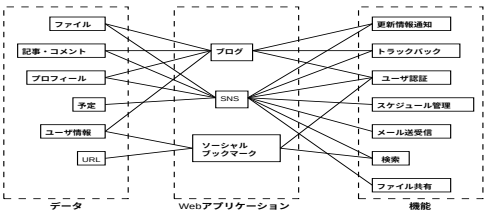

図 1　アプリケーションと機能，データの関連図

3.2　scaffold を構成するファイルの分析

scaffold が生成するコードに共通の機能やデザインなどを埋め込むために，ファイルの構成を分析した．ブログ，掲示板アプリケーションを作成したところ，scaffold が生成するコードの 17 箇所を変更可能な箇所として抽出した．表 1 に scaffold を構成するファイルの一部と変更箇所を示す．

表 1 scaffold を構成するファイル

scaffold 構成ファイル	ファイルの説明	変更箇所数
scaffold_generator.rb	生成するファイルの定義	2
controller.rb	利用するメソッドの定義	3
model.rb	データベース接続に関する定義	1
layout.html.erb	すべてのページのレイアウト定義	2
view-show.html.erb	データ一覧ページの表示	1

ユーザ認証機能を埋め込む場合の例を挙げる．ユーザ認証機能はユーザとパスワードのデータ，ログイン画面が必要になる．scaffold_generator.rb の manifest メソッドに新規コントローラファイル (session_controller.rb) 生成，scaffold_view メソッドに新規ビューファイル (view_login.html.erb) 生成の定義を記述する．model_generator.rb の manifest メソッドには新規モデルファイル (user.rb) 生成の定義を記述する．実際に認証を行うコードは session_controller.rb，user.rb に記述する．また，ログイン画面は view_login.html.erb を修正し，すべてのページでログアウトできるように layout.html.erb を修正する．

3.3 拡張可能な scaffold

各機能やデザインごとにコードを埋め込む箇所が決まっているので，機能部品等と埋め込み箇所の対応関係を管理し，Web アプリケーション毎に必要な部品などを選択できる scaffold を提案する．scaffold に与える選択肢としては，Web アプリケーションの基本機能を提供するアプリケーション定義，オプションとして追加できる機能部品，画面構成を定義するページテンプレートの 3 つであり，これらをアプリケーション設定ファイルに記述し，拡張 scaffold によって雛型を生成する (図 2)．ページテンプレートはユーザ毎に異なるので，ユーザ自身が利用するファイルを追加する必要がある．

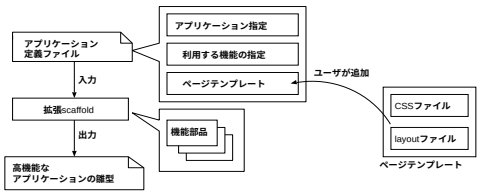

図 2　自動生成ツールの概略図

3.3.1 機能部品

機能部品には機能毎に scaffold を構成するファイルの変更箇所に追加すべきコードを対応付け，記述する．例として認証機能の一部を挙げる．scaffold_generator.rb の manifest メソッドを指定し，新規コントローラ生成のコードを埋め込む処理が記述されている．

```
Point:  scaffold_generator.rb, manifest
Code:  ${ m.template( 'session_controller.rb',
File.join('app/controllers',
controller_class_path,
"#{controller_file_name}_controller.rb" )) $}
```

3.3.2 ページテンプレート

ページテンプレートは CSS ファイルと layout ファイルで構成され，配備先のサイト毎に用意する．layout.html.erb の<head>内で CSS ファイルを指定し，<body>内に会社のロゴやヘッダなどの共通に利用するデザインを記述する．

3.3.3 アプリケーション設定ファイル

アプリケーション設定ファイルには構築したい Web アプリケーションの種類，追加したい機能，用いるデザインを記述する．認証機能のあるブログアプリケーションを作成する際の記述例を以下に示す．

```
Application:  Blog
Option:  login-auth, before-index
Style:  nanzan-style.css, nanzan-style.html.erb
```

アプリケーション定義は機能部品の集合であり，複数の機能部品を適用することで各 Web アプリケーションを構成する．Application: Blog ではブログの機能部品集合があらかじめ定義されたファイルを読み込む．Option: login-auth で読み込んだブログ用雛型の該当箇所に認証機能追加コードを埋め込む．また，認証を行うタイミングは複数の可能性があるので，オプションとして指定できる．上記で示す例ではトップページが表示される前にログイン画面を表示すると指定している．ページテンプレートは，あらかじめ CSS ファイルと layout ファイルが用意してある場合は，指定することで反映される．指定していない場合は従来の scaffold による画面構成として生成される．

4 考察

提案する拡張 scaffold を機械的に適用して，認証機能のあるブログアプリケーションを作成し，従来の scaffold を利用した開発方法とのコード記述量を比較した．従来の開発方法に比べて 96 行分が自動で生成され，ユーザによる記述の手間が省ける．また，作成したブログ，掲示板で認証機能，レイアウトが修正することなく再利用できることを確認した．

5 おわりに

本研究では Web アプリケーション開発現場における scaffold の問題点に着目し，Web アプリケーションで共通に利用できる機能やデザインが追加された雛型を生成する scaffold の拡張の提案を行った．今後の課題は機能部品の追加，実装，評価などが挙げられる．

参考文献

[1] CodeZine, “Ruby on Rails の scaffold をカスタマイズして使いやすくする，” http://codezine.jp/article/detail/2912.

[2] MINIELEMENTS, “ActiveScaffold, ” http://activescaffold.com.

[3] 37signals, “Real applications live in the wild, ” http://rubyonrails.org/applications.