エンドユーザ主導 Web アプリケーション開発技法の提案と評価

Evaluation of End-User-Initiative Web Application Development Technique

橋本 竜太† 中所 武司†
Ryuta Hashimoto Takeshi Chusho

1. はじめに

我々は，変化の激しい時代には，エンドユーザ(業務の専門家)主導のアプリケーション開発とその保守が重要になるという観点から，特に，小さな部門や個人の業務を対象とする中小規模のWebアプリケーションに関して，低コストで短期間に開発するとともに，頻繁な機能変更を伴う保守にも対応するために，その分野の業務の専門家主導で開発・保守ができるような技法を研究してきた．

特にエンドユーザにはユーザインタフェースやワークフローに注目したアプローチがわかりやすいという観点からUI駆動型やモデル駆動型のフレームワークやビジュアルモデリングツールの研究と試作を行ってきた[1]．その一方で，データベース処理に関しては，エンドユーザによる本格的な設計は難しいので，簡単なテーブル定義ツールによる支援にとどまっていた．

しかしながら，Web アプリケーション開発ではデータベースの設定やフォームより送られたデータの処理などに多くのファイルの作成やコードの記述を伴う．このことがエンドユーザ主導開発を阻んでいると思われる．

これらの課題を解決するために，本研究ではエンドユーザにとってプログラミングの不要な Web アプリケーション構築ツールを研究試作し，小規模な Web アプリケーションへの適用実験を通して評価を行った．

なお，本研究で開発した Web アプリケーション構築ツールを EI-WAG(End-user Initiative Web Application Generator)と名付ける．

2. エンドユーザ主導開発の実現方法

EI-WAG は GUI(Graphical User Interface)ベースのツールとなっている．エンドユーザは GUI を操作して，構築する Web アプリケーションの設定を行うだけでよい．この実装技術としては，テキスト処理能力に優れシンプルな文法で可読性の高いコードが記述できる[2][3]という観点から Ruby を採用した．また設定が最小限で済み，サーバや ORM などが備わったフルスタックな Web アプリケーションフレームワークである Ruby on Rails[4](以下 Rails と略す)を用いた．Rails の機能を使うには CUI(Command User Interface)操作が必要である．しかし CUI 操作はエンドユーザにはわかりにくい操作である．Rails をエンドユーザ主導開発に繋げるため，エンドユーザにとって分かりやすい GUI アプリケーションを用意することにした．

実際の開発手順を図 1 に示す．①エンドユーザは EI-WAG の GUI を操作して，開発しようとする Web アプリケーションに関する設定を行う．②EI-WAG はエンドユーザが設定した内容を基に，Rails に新規プロジェクトを構築させるためのコマンドを発行する．③命令を受けた Rails はエンドユーザにより指定されたファイルパスに必要なファイルやフォルダを自動生成する．④EI-WAG は必要に応じて Rails により生成されたファイルの編集や新たなファイルの自動生成を行うことで Web アプリケーションの構築を行う．このようにグラフィカルな GUI アプリケーションと Ruby on Rails とを連携させたエンドユーザ主導の Web アプリケーション開発技法を提案する．

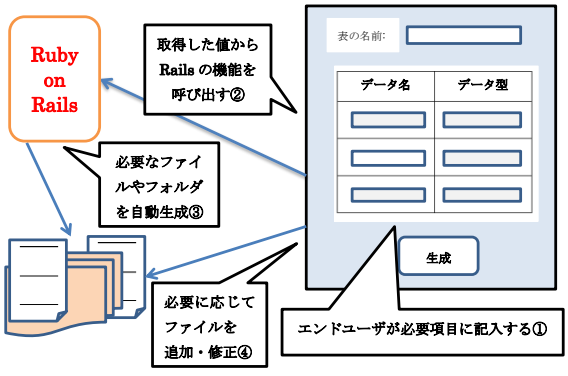

図 1: EI-WAG を用いた Web アプリケーション開発概要

†明治大学大学院理工学研究科基礎理工学専攻情報科学系ソフトウェア工学研究室

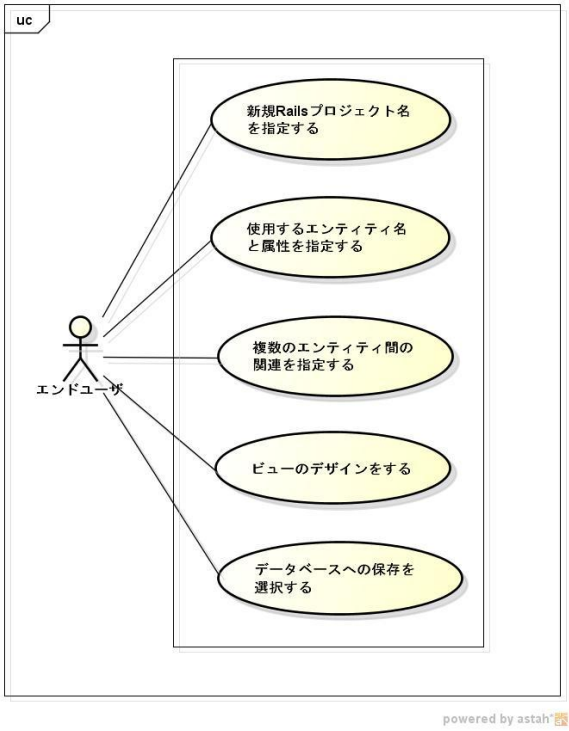

図 2: EI-WAG のユースケース図

3. EI-WAG システム概要

3.1. 機能概要

本ツールの機能の全体を図 2 に示す.

①エンドユーザは EI-WAG を用いて，新規の Rail を用いた Web アプリケーションを構築するファイルパスやプロジェクト名等を記述する.

②Rails の機能で，あるエンティティの CRUD(Create, Read, Update, Delete)操作を備えた Web アプリケーションの骨組みを生成する Scaffolding[4]を実行するために必要なエンティティ名や必要な属性数やそれぞれの属性名等を記入する. Scaffolding 機能を使えば CRUD 操作を備えた Web アプリケーションが簡単に生成される．そのため複雑な設定操作等が必要なく, エンドユーザ主導開発に有用であると考え本研究で採用した.

③生成された Web アプリケーションへの追加の機能を選択する．CRUD 機能を必要最低限の機能として，これに上乗せする形で検索やバリデーションといった機能を選択できるようにした.

④複数のエンティティ間の関連を決める. 複数のリソースを管理できるような Web アプリケーションをエンドユーザが構築できることを目指してエンティティ間の関連を定義できるようにした.

⑤ビューのデザインを行う.

⑥データベースへの保存の設定を行う.

本稿では Scaffolding の実行、検索・バリデーション機能の付加について，図書(属性に title，author，price)を管理する Web アプリケーションを例題に用いて説明する.

3.2. Scaffolding の実行

Scaffolding はあるエンティティの CRUD(Create, Read, Update, Delete)機能を備えた Web アプリケーションの骨組みを半自動生成する Rails の機能である[4]. 図 3 に Scaffolding 実行のための GUI 画面を示す．エンドユーザはリソース(エンティティ)名と属性数を設定し，各フィールド名(属性名)とそのデータ型を設定する.

図 3 ではリソース名を book として，必要な属性数を 3 つとした．また 3 つの属性はそれぞれ title(String 型), author(String 型), price(Integer 型)として Scaffolding 実行を行った.

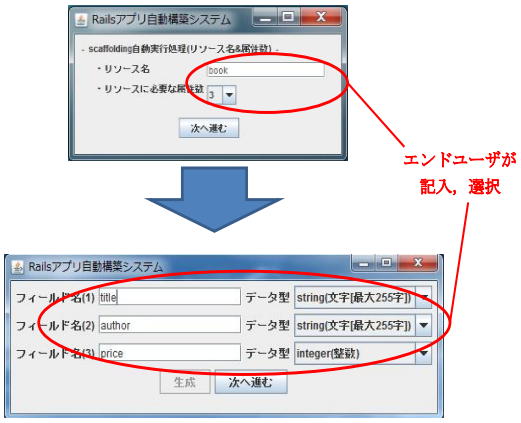

図 3: Scaffolding 実行に必要な事項の記入・選択

3.3. 検索機能の付加

Scaffolding 実行によって生成された Web アプリケーションに検索機能を付加する設定を図 4，図 5 に示す.

エンドユーザは図 4 のように追加機能の設定で検索機能を選び, 機能の追加ボタンを押す. そして図 5 のように, 検索が必要な属性を選択することで検索機能の設定を行う.

図 5 では title と price にチェックを付け，この 2 項目についてのみ検索できるように設定した.

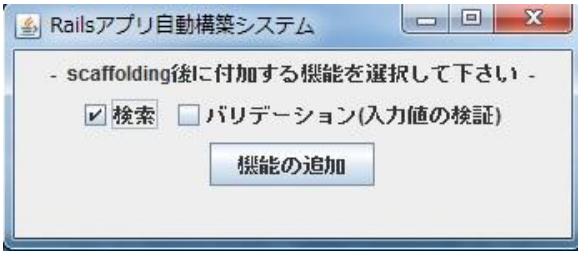

図 4: 検索機能の付加設定(1)

図 5: 検索機能の付加設定(2)

3.4. バリデーション機能の付加

バリデーション機能を付加する詳細設定の画面を図 6, 図 7 に示す．なおバリデーション機能追加の選択は図 4 において「バリデーション機能」のチェックボックスをチェックしておけば良い．

図 6 がバリデーションの必要な属性の選択画面であり，図 7 は選択された属性ごとの詳細なバリデーション設定画面である．

バリデーションではデータ型ごとに選択できる項目を分けた．入力数の指定，値が入力されていない場合の許可/不許可，一意の値(既に登録された値)の許可/不許可については汎用的にバリデーションとして利用されると考え詳細設定項目に採用した．

図 6 では title と price のみバリデーションを選択した．図 7 では title では空白を許可しない，文字数の指定をする(範囲指定で最小 3 文字以上最大 100 文字以下)，既に同じ値が登録されていたら登録しないにチェックを付けた．price では整数値以外許可しない，最初値・最大値の設定をする(最小 1 以上最大 10000 未満)にチェックを付けた．

以上のようにしてバリデーションの機能の付加を行った．

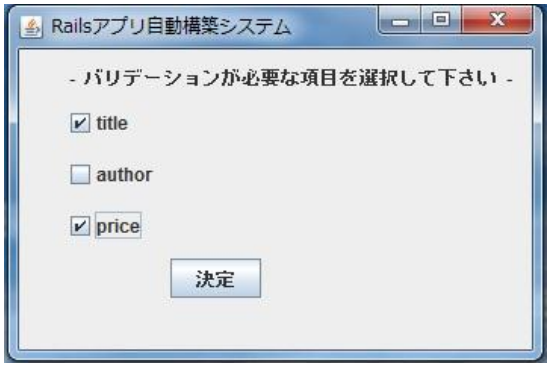

図 6: バリデーション機能の付加設定(1)

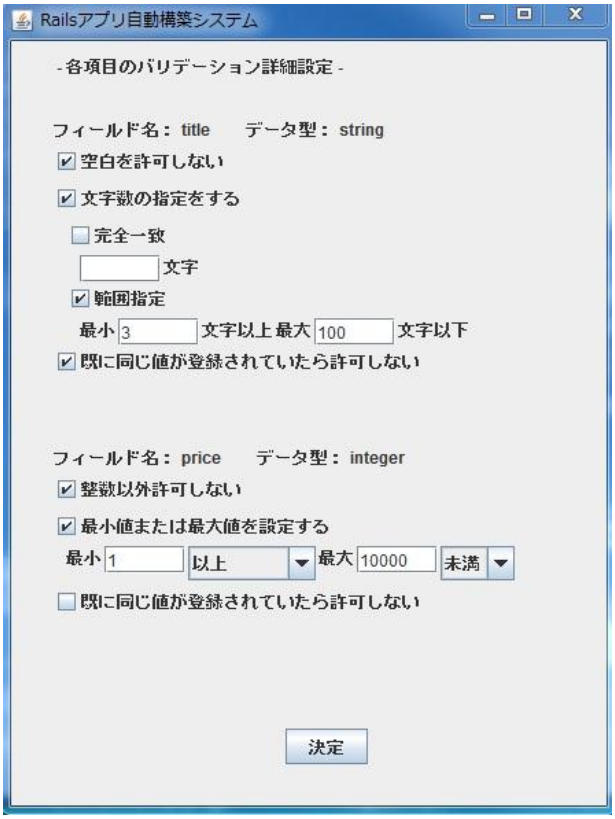

図 7: バリデーション機能の付加設定(2)

4. 構築実験

4.1. 検索機能の実験

3 章では，属性に title，author，price を持ち，CRUD 機能と検索とバリデーション機能を備えた図書を管理する Web アプリケーションを構築した．この Web アプリケーションを，Web サーバを立ち上げて Web ブラウザを通して操作した時に，3 章で付加した検索とバリデーション機能が正しく動作することを確認するため実験を行った．

検索機能の動作確認を行った結果を図 8，図 9，図 10 に示す．図 8 は構築した Web アプリケーションのトップページ画面であり，丸で囲んだものは検索へのリンクである．あらかじめ数冊の図書情報を登録してから検索機能の実験を行った．このリンクの先が図 9 の検索画面である．検索に必要な内容をフォームに記述してから検索ボタンを押すことで図 10 の検索結果画面へとリダイレクトされる．

図 9 では title に b，price に 1000 として図 8 での登録済みの図書の検索を行ったところ図 10 のように 2 件の検索結果が得られた．現状検索には OR 検索を用いているため price が 1000 で title に b が含まれる 2 件が検索された．このことより検索機能は正しく機能していることを確認できた．

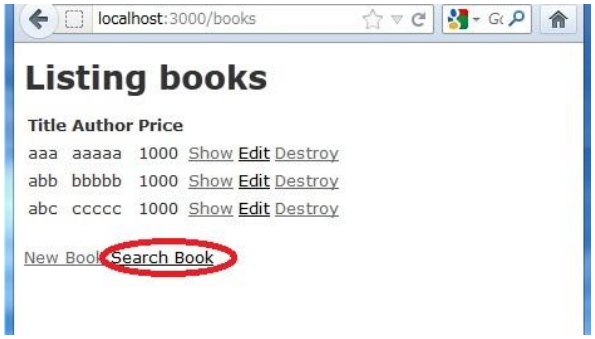

図 8: 生成されたトップページにおける検索へのリンク

図 9: 検索のビュー画面

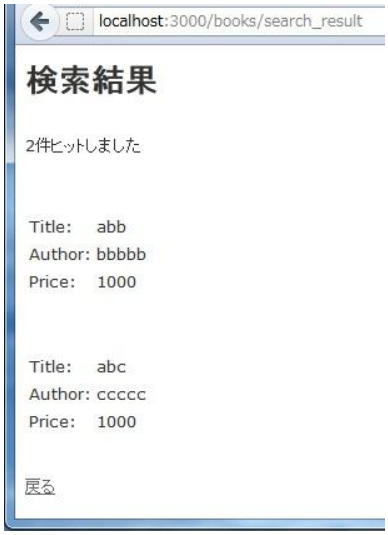

図 10: 検索結果のビュー画面

4.2. 検証機能の実験

4.1 と同様にバリデーション機能付加後の動作確認を行ったのが図 11 である．新規データ登録の際にバリデーションにより登録が拒否された様子を示している．バリデーションにより登録が拒否された内容の詳細は，図 11 のようにフォームより前部のテーブル内に詳細表示される．Title に a, Author に a, Price に a と入力して下の Create Book ボタンを押したところ「Title が短か過ぎる(最小でも 3 文字)」ということと「Price が整数でない」という警告がテーブル内に出力されていることがわかる．これは 3.4 でバリデーションの設定を行った際に，title は 3 文字以上 100 文字以下で price は整数値以外許可しないとしたためである．警告の内容は図 7 で Title と String にバリデーションを設定したものと一致することがわかる．

このことより，バリデーション機能は GUI を通して設定した内容通りに機能していると確認できた．

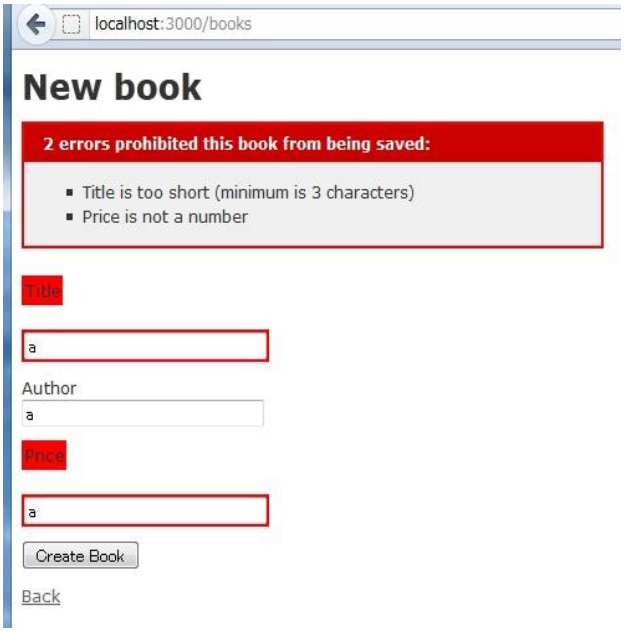

図 11: バリデーションにより登録が拒否された詳細

5. 終わりに

EI-WAG を用いることでエンドユーザはプログラミングレスで 1 つのエンティティの CRUD 操作に検索とバリデーション機能の付いた Web アプリケーションを構築できることを確認した．

今後さらに機能を増やし，複数エンティティ間の関連を定義して，連携した処理を行えるようにしていきたい．

6. 参考文献

[1] 中所武司，“業務の知識を有するエンドユーザ主導のアプリケーション開発技法 ～フレームワーク・ドメインモデル・サービス連携～”，電子情報通信学会 技術研究報告 Vol.107,No.331，知能ソフトウェア工学研究会 KBSE2007-30,19-24(Nov. 2007).

[2] 橋本竜太，Java と Ruby の比較，明治大学理学部情報科学科 2011 年度卒業論文.

[3] Ruby, http://www.ruby-lang.org/ja/.

[4] Ruby on Rails, http://rubyonrails.org/.