Ruby on Rails による Web アプリ開発の授業実践

高橋圭一[†1]

概要：2016 年度に本学科のプログラミングの導入言語を Java から Ruby に切り替えた．そのため，3 年次の Web アプリケーション開発の講義および演習で使用するフレームワークを J2EE から Ruby on Rails に変更した．本稿では，授業計画時の検討内容と実践結果を報告する．

キーワード：Ruby，プログラミング演習，開発フレームワーク

The class practice of web application development with Ruby on Rails

KEIICHI TAKAHASHI[†1]

Abstract: In 2016 we changed the introductory programming language in our department from Java to Ruby. Due to the change, we changed the web application development framework from J2EE to Ruby on Rails, which has been used in lectures and exercises of the 3rd year. In this paper we report the plan and the result of practices of the lecture and the exercise.

Keywords: Ruby, Programming Exercise, Development Frameworks

1. 背景

本学科は 2004 年に新設した定員 70 名の学科であり，ACM /IEEE-CS が策定したコンピューティング・カリキュラム（CC-2001）をもとに設計したカリキュラムにより，ネットワークやソフトウェアの企画・設計・開発・保守などに関わる技術者の輩出を目指してきた．2016 年度に，時代の変化に合わせて CG や Web などのメディア制作やデータサイエンスのコースを加えてカリキュラムを改定した．

カリキュラム改定前には，導入から応用まで一貫してプログラミング言語として Java を採用してきたが，カリキュラム改定後には，プログラミングを必ずしも課さないコースもあるため，1 年前期のみプログラミング科目を必修とし，学習の容易さからそのプログラミング言語として Ruby を選択した．なお，所属コースによっては選択科目も含まれるが，プログラミング言語としては，1 年後期に Processing（≒Java），2 年次に C#と C++を学習する科目があり，データサイエンスコースの科目では R や Python を学習する機会がある．2 年後期にはデータベース科目で SQL を学習する．

本稿で対象とする Web アプリケーション開発の科目は 3 年次前期の科目であり，講義と演習を 1 コマずつ連続した時間割で実施する．カリキュラム改定前には，統合開発環境 Eclipse および Java の Web アプリケーションフレームワークである J2EE（Java 2 Platform Enterprise Edition）を採用してきたが，Ruby への変更に合わせて，開発環境は Cloud9 IDE[1]に，Web アプリケーションフレームワークには Ruby on Rails（以降 Rails と略す）に変更した．本稿では，この変更に伴う授業計画時の検討内容と， 2018 年度前期に実施した講義および演習の結果について報告する．

2. 授業計画

2.1 Rails の特徴

Rails は Ruby で書かれたオープンソースの Web アプリケーションフレームワークである。公開されてから 15 年近く経過しており，GitHub やクックパッドなど国内外に多くの利用者を抱えるサービスでも採用されている．Rails チュートリアル [2]などの学習用のオンライン・ドキュメントも充実しており，また，Ruby の開発者が日本人であるため，日本語で読めるリソースがインターネット上に豊富にあり国内の初学者が学びやすいことも特徴の 1 つである．

Rails は基本哲学として DRY（Don’t Repeat Yourself；同じことを繰り返さない）と CoC（Convention over Configuration；設定より規約）を掲げており，Rails が採用している MVC（Model View Controller）アーキテクチャを理解し，rails コマンドなどの各種ツールを使いこなすことで品質の高いソフトウェアを効率的に開発できる． 図 1 に Rails の基本的な構成を示す．Rails アプリケーションが Web ブラウザから HTTP リクエストを受信すると，ルーティング（routes.rb）に書かれたコントローラ（Ruby クラス）を呼び出し，対応するビュー（ERB ファイルa）やモデル（Ruby クラス）を呼び出す．これらのファイルは rails コ

†1 近畿大学産業理工学部情報学科
Kindai University

a Ruby を埋め込める HTML ファイル

マンドによりテンプレートが自動生成され，クラス定義など定型的なコードの記述を省略できる．これらファイルを Rails が定めた場所に配置することにより，設定ファイルに定義を記述することなく呼び出すことができる．こうした様々な規約が Rails の DRY や CoC といった基本哲学の表れである．

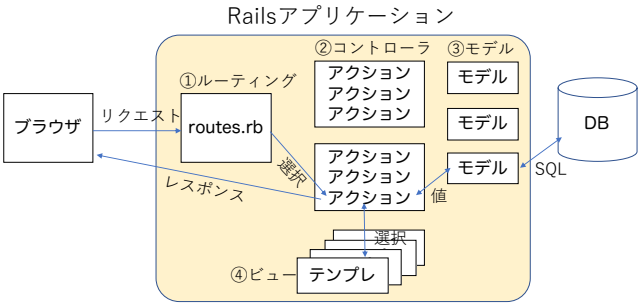

図 1　Rails の MVC アーキテクチャ

2.2 Rails を選択する上での懸念事項

カリキュラム改定前の J2EE を使用した Web アプリケーション開発の講義・演習と比較すると Rails の採用では以下のような懸念があった．

- 構成要素が多い
- 憶えることが多い
- 自作していないコードのデバッグが難しい

図 2 に本講義で扱う範囲の J2EE と Rails の主な構成要素を挙げる．Rails の方が開発フレームワークで規定している構成要素が多く（着色部），これらの構成要素の仕組みを正しく理解し，定められた使い方に従ってプログラミングする必要がある．憶えて使いこなす構成要素が多くなるため，Web アプリケーション開発の初学者にとってはハードルが高くなることが懸念された．さらに，Rails には自動生成されたコードが含まれるため，デバッグ時に誤り箇所を探すのが困難なことも予見された．

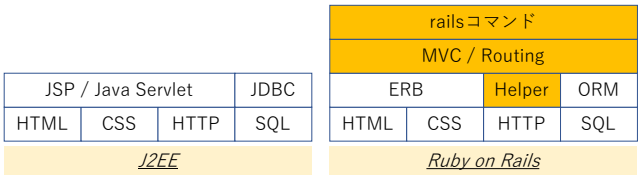

図 2　J2EE と Rails の主な構成要素

2.3 学習内容

Rails を学習する第一の方法は Michael Hartl 著の Rails チュートリアルである[2]．Web アプリケーションを作りながら必要な知識を段階的に学ぶことができる．インターネット上で無償で公開されており日本語にも翻訳されている．解説動画を制作して公開している企業と提携して学生向けに学習コンテンツを提供している大学もある．ただ，本チュートリアルは基本的に独習用となっており，講義と演習で利用するには工夫が必要である．

第二の方法は，カルフォルニア大学バークレー校の ESaaS（Engineering Software as a Service）と呼ばれる講義である．講師らが執筆した教科書や授業スライドが入手可能であり，講義動画も公開されており，いくつかの大学も採用している教育プログラムである[3]．我々は本プログラムの採用に向けて数ヶ月間に渡り準備を進めたが，最終的に本プログラムで扱っている内容が広範囲であり，所定の時間では教授が困難と考え断念した．

第三の方法として，Rails 開発について書かれた書籍を探し，候補の中から黒田氏らによる「基礎 Ruby on Rails」を選定した[4]．本書では HTTP なども解説があり，Rails の操作に留まらず初学者が Web アプリケーションを開発する上で必要な知識を学ぶ配慮があり，Rails を講義している他大学でも採用実績があったことなどが採用の理由である．

図 3 に講義科目のシラバスの一部を示す．演習科目もほぼ同様の構成である．受講生らは Ruby を 1 年次に学習しただけのため，基本文法の復習が必要であり，さらに一般的な Ruby による応用プログラミングの学習も必要と考え，講義のうち第 8 回までは，モジュールプログラミング，ファイル，データベース，Rails を用いない Web アプリケーションの開発方法を学ぶこととした．第 9 回から[4]を参考にして Rails について講義・演習を行う．

第1回 導入講義、Rubyの基礎（復習）
第2回 クラスモジュール
第3回 ファイルによる永続化
第4回 データベース基礎（復習）
第5回 データベースによる永続化
第6回 Webアプリケーション基礎
第7回 蔵書管理アプリのWeb化
第8回 Webアプリケーション開発
第9回 Ruby on Rails：ルーティングとMVC
第10回 Ruby on Rails：scaffoldを使わないアプリ開発
第11回 Ruby on Rails：画像共有アプリ
第12回 Ruby on Rails：バリデーションとテスト
第13回 Ruby on Rails：総合演習
第14回 Ruby on Rails：総合演習
第15回 定期試験
第16回 解説

図 3　講義科目のシラバス

2.4 環境構築

本学科の演習室には約 150 台のパソコン（Windows10）があり，ネットワークブート方式のシンクライアント・システムとして動作している．本方式では，アプリケーションなどはディスクイメージとして固定されており，アプリケーションの設定情報などのプロファイル情報は，ユーザ毎にネットワーク上のファイルサーバで管理される．

Linux や macOS と比べると Windows での Rails 開発環境の構築は難しいと言われており，ユーザ情報がファイルサーバー上で分散管理されるシンクライアント・システムでは開発環境の構築は困難であることが予想された．以下，試行した方式について述べる．

ローカルドライブにインストール：1 年次のプログラミング演習用に Ruby がインストール済みであったため，そのパッケージ管理ツールである gem で Rails をインストールしたところ動作したが，再起動するとインストールした情報が消えて利用できなかった．

ネットワークドライブにインストール：シンクライアント環境では，ユーザのホームディレクトリはファイルサーバー上にある．そこに Rails をインストールして動作することを確認したが，再起動するとファイル自体は保存されるものの，必要な設定ファイルなどが消えるため利用できなかった．

仮想化ソフトウェアを利用してインストール：仮想化ソフトウェアの 1 つであるオラクル社の VitualBox をインストールし，その上で Linux 環境の構築を試したが，VirtualBox のインストールは実行できたものの，再起動するとパソコンが起動せず解決できなかった．また，Windows10 上に Linux 環境を構築できる Windows Subsystem for Linux も試したが，同様の状況でセットアップすることができず断念した．

クラウドベースの統合開発環境：Web ブラウザのみで利用できる開発環境でプログラムの編集・実行はもちろんターミナルが利用できるため，ローカルパソコンと同様の感覚で Linux コマンドを実行することができる．このクラウドベースの統合開発環境の 1 つである Cloud9 社による Cloud9 IDE[1]は基本的に無料で利用できるが，アカウント登録時にクレジットカード情報を入力する必要があったため採用が困難であった．2016 年に同社が開始した Cloud9 for Education というサービスを利用することで回避できるようになったため，本稿で対象とする講義・演習では本サービスを利用することとした．

3. 授業実施

2 章に示した授業計画をもとに 2018 年度前期に実施した Rails の Web アプリ開発の講義・演習の実施結果について述べる．

3.1 科目と受講生

2018 年度の本科目の受講生は 32 名であった．本学科（定員 70 名）の 4 年生の約半数が IT 系企業の就職を希望してきたが，カリキュラム改定後の受講生のうち，こうした IT 企業を志望する学生が履修したと思われる．受講生には本科目が必修ではないコース生も含まれていたが，多くが選択科目にも関わらずプログラミングやデータベースの科目を受講済みであった．なお，カリキュラム改定前の 2017 年度の受講生は 56 名であった．カリキュラム改定前では本科目は必修科目であったことがこの差の要因である．演習科目の講師は筆者 1 名とティーチング・アシスタントが 1 名である．

3.2 定量的な比較

J2EE と Rails の演習の難易度の変化を比較するため，演習科目の課題の未提出者率と提出された課題の平均点の推移を調べた（図 4）．課題点は 10 点満点であるため，10 点未満ということは，解答が不正解もしくは途中まで解答して提出したことを表している．各回の演習内容や難易度が異なるため単純な比較は難しいが，Rails を開始した第 9・10 回は比較的提出者数が多い．第 11 回は REST アーキテクチャや画像アップロードを，第 12 回ではテストを扱い，第 9・10 回と比べてデバッグが難しく躓く者が多かったため未提出が増えたようである．第 13・14 回はアプリ自作であり，難度が高いことと講義終盤であり十分に課題を提出した者が未提出となるケースがあったと思われる．

比較のため前年の 2017 年度の推移を図 5 に示す．科目の必修・選択が異なり受講者や演習問題が異なるが，筆者が講義・演習を担当する点は共通する．両年度の課題の未提出者率と平均点はほぼ同じであった（表 1）．両年度の未提出者率および課題平均点について，有意水準 5%で t 検定および比率差を検定したところ，有意差は認められなかった． つまり，課題提出状況からは J2EE と Rails による演習の難易度には明らかな差は認められなかったと考えられる．

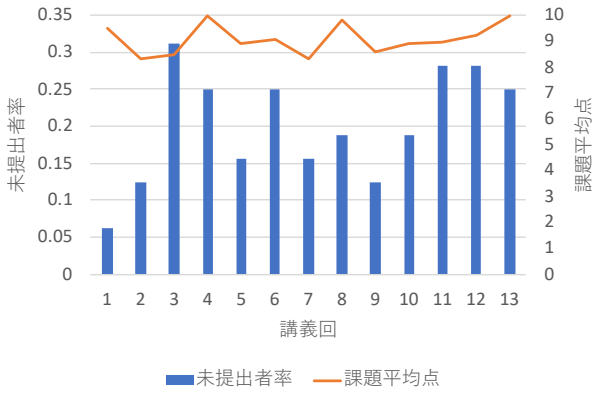

図 4　課題の未提出者率と平均点の推移（2018 年度）

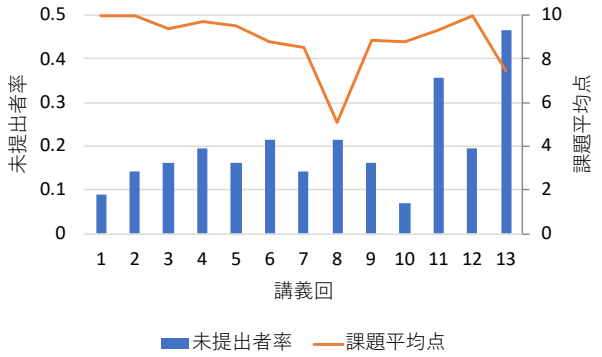

図 5　課題の未提出者率と平均点の推移（2017 年度）

表 1　課題の未提出者率と平均点

	2018 年度	2017 年度
未提出者率	20.2%	19.8%
課題平均点	9.0	8.9

3.3 講師による主観評価

講師が演習時間中の指導時に気づいた点について Rails

の機能別に述べる．

ビュー：ビューを記述する ERB ファイルは HTML と Ruby のスキルがあれば記述できる．これは J2EE の JSP（Java Server Pages）と同等であり，指導上，大きな差は見られなかった．

モデル：J2EE を用いた演習では Java で SQL を記述する必要があり，文字型カラムでのシングルクォーテーション指定忘れ，文字型・数値型間での型変換の必要があるなど，演習中に躓く受講生が多かった．この躓きは教育的に意図的であり，学習済みのプログラム言語と SQL を組み合わせてある程度のサイズのコードを構築する訓練になると我々は考えてきた．一方，Rails のモデルクラスは Active Record クラスを継承しており，この Active Record クラスがデータベースとのアクセスを仲介してくれるため，開発者は SQL を記述せずに一般的な Ruby クラスのインスタンスを操作するだけで済む．Active Record クラスのメソッドを憶えて使い慣れる必要はあるが，SQL の文法誤りのように実行時にしかわからないようなミスの相談はほとんど無かった．

コントローラとルーティング：J2EE による演習では，採用していた統合開発環境 Eclipse のおかげで，特別な設定をしなくても作成した Java Servlet や JSP にアクセスできた．一方，Rails では作成したビューにアクセスするにはコントローラの作成とルーティングの設定が必要になる．その設定忘れや記述ミスをしたときに表示されるエラーメッセージと原因の対応づけに慣れるまで受講生からの相談があった．

テスト：テストの記述方法や実行方法の基本的な手順は解説し演習したが理解が難しかったようである．本講義で参考にした書籍[4]でも，次版である第 4 版では「理解が難しく躓く読者が多い」ということで，テストに関する記述の分量が抑えられたようである．

ヘルパー・メソッド：Rails には HTML タグに対応したヘルパー・メソッドが用意されている．例えば，`<a href=”/home”>ホーム</a>` のような A タグを生成するヘルパー・メソッドは `link_to “ホーム”, “/home”` と記述する．HTML のみで記述する場合と比べて記述量を減らすことができ，さらに HTML の文法エラーを発見しやすくするメリットがある．Rails の開発者はヘルパー・メソッドを優先的に使用することが一般的のようであるが，本講義では最初は可能な限りそれまでに学習した HTML で記述し，段階的にヘルパー・メソッドに置き換えるように指導した．form タグに対応したヘルパー・メソッドではルーティングの設定誤りの可能性もあり相談が頻繁にあったが，演習が進むにつれて慣れたようであった．

演習環境：Rails の機能ではないが，今回採用した Cloud9 IDE については全く問題が見られなかった．クラウドベースで Web ブラウザのみで利用できるため，学生らの多くは自宅で課題に取り組んだようである．ただ，Cloud9 社は Amazon に買収されて AWS Cloud9 というサービスが立ち上がっているため，Cloud9 IDE がいつまで使えるか不安がある．また，Cloud9 IDE 上で立ち上げた Web サービスは外部に公開できず，共同作業する機能もあるが設定が必要なため，受講生が躓いたときに遠隔から支援することが難しい．

3.4 学生による授業評価

本大学では，半期の講義の終盤に所定の用紙を用いて授業評価アンケートを実施している．その中に「この教員の授業を 10 点法で評価してください」という設問がある．解答欄には，10 が非常によいとし，7 がほぼ満足，1 が非常に悪いとある．2018 年度は講義科目が 7.9，演習科目が 8.1 であった．2017 年度はそれぞれ 7.5 と 7.2 であった．2017 年度は受講生全員にとって本科目が必修であったが，2018 年度の受講者には選択科目である学生も多くいたことから，自ら進んで受講した学生が評価を上げた結果になったのではないかと考えられる．

4. 考察

Rails が公開されて 15 年近く経過しており，カリキュラム改定以前にも変更する機会はあった．カリキュラム改定前の卒業生から Apache Struts や Spring Framework を学びたかったという声もあった．これら開発フレームワークの採用を躊躇していた理由は 2.2 の通りである．3 年次の本科目では，それまでに学習してきたプログラミングや SQL やネットワークの知識を組み合わせて，プログラムが論理的に文法的に正しくとも環境要因で動作しないことを体験させることを狙っていた．

J2EE では最初は JSP Model1 を教えていた．HTTP リクエストを受け取り，その内容に応じて処理を振り分ける素朴な Web アプリケーションの形式である．基本的な処理の流れは単純であるがタイプミスによって様々な実行時エラーが発生し，SQL の記述ミスや DB ライブラリ追加忘れや DB サーバー未起動によって動作しないなど様々な失敗体験ができた．

一方，Rails は各ファイルを手動で作成することもできるが，これまで述べた通り，rails コマンドによってテンプレートが自動生成されるため，開発者がソフトウェアの基本的な構造に触ることがない．したがって，タイプミスに起因する文法エラーはもちろん実行時エラーもある程度は避けられ，開発者が本来考え記述すべきロジックの実装に集中できる．

2.2 で危惧したように Rails では憶えることが確かに多いが，Rails の規約に慣れてしまえば，フレームワークによる Web アプリケーションの基本構造の抽象化によって記述量が減ることで J2EE と比べて手軽に Web アプリを作ることができたようであった．Rails の情報はインターネット上に豊富にあり，エラー発生時に自分自身で問題を解決することができたようである．J2EE では 15 回かけて応用プログ

ラムがやっと作れるようになったが，受講生によってはやり甲斐を感じつつも苦痛に感じていたようであった. 一方，Rails では 6 回と半分の講義・演習で Web アプリが作れるようになり，手軽に作れるという感覚からか，受講生によっては学習したことをもとに課外活動のための Web サービスを授業時間外に開発した事例も耳にした.

5. まとめ

本稿では，本学科の 3 年次を対象とした Ruby on Rails による Web アプリ開発の授業の準備と実施結果について，前年度までの J2EE との比較を中心に報告した．課題提出状況による定量的評価では，カリキュラム改定前後で講義や演習の難易度に明らかな変化は無かったことが示された．本稿で取り上げた講義・演習科目は来年度も引き続き開講するため，演習中にエラーメッセージなど収集する仕組みを取り入れるなどして，躓き要因の解析や予防するための支援システムの構築に繋げていきたい.

参考文献

1) “Cloud9”, https://c9.io, （参照 2019- 2-10）.
2) “Ruby on Rails Tutorial”, https://www.railstutorial.org/book, （参照 2019-2-10）.
3) Armando Fox, David Patterson. Engineering Software as a Service: An Agile Approach Using Cloud Computing. Strawberry Canyon LLC, 2013.
4) 黒田努, 佐藤和人. 基礎 Ruby on Rails 第 3 版, インプレス, 2015.
5) “JSP Model1”, https://en.wikipedia.org/wiki/JSP_model_1_architecture, （参照 2019- 2-10）.