ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

2014-09-22

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

第 1 章ガイダンス

連絡事項

資料等の入手先

- GitHub の下記リポジトリにまとめておきます
  - ychubachi/enpit
- 資料は随時 update するので，適宜，最新版をダウンロードしてください

Twitter のハッシュタグ

- Twitter のハッシュタグは `#enpit_aiit` を使ってください
- まとめサイトなど作ってくれると嬉しいです
  - 昨年の例 -> enPiT BizApp AIIT ビジネスアプリケーション演習 1 日目 - Togetter まとめ

学習目標と目的

目標

- ビジネスアプリケーションを構築するための基礎力
- 分散型 PBL を実施する上で必要となる知識やツールの使い方
- これら活用するための自己組織的なチームワーク

目的

- 分散ソフトウェア開発のための道具を学ぶ
  - 開発環境（Ruby），VCS とリモートリポジトリ（GitHub）
  - テスト自動化，継続的インテグレーション，PaaS

前提知識と到達目標

前提とする知識

- 情報系の学部レベルで基礎的な知識を持っていること

最低到達目標

- 授業で取り上げる各種ツールの基本的な使い方を身につける

上位到達目標

- 授業で取り上げる各種ツールの高度な使い方に習熟する.

授業の形態

対面授業

- 担当教員による講義・演習

個人演習

- 個人によるソフトウエア開発

グループ演習

- グループによるソフトウエア開発

講義・演習・課題

講義

- ツールの説明
- ツールの使い方

演習

- 個人でツールを使えるようになる
- グループでツールを使えるようになる

成績評価

課題

- 個人でソフトウエアを作る
- グループでソフトウエアを作る

評価の方法

- 課題提出と実技試験

評価の観点

- 分散 PBL で役に立つ知識が習得できたかどうか

ソフトウエア開発のための方法・言語・道具

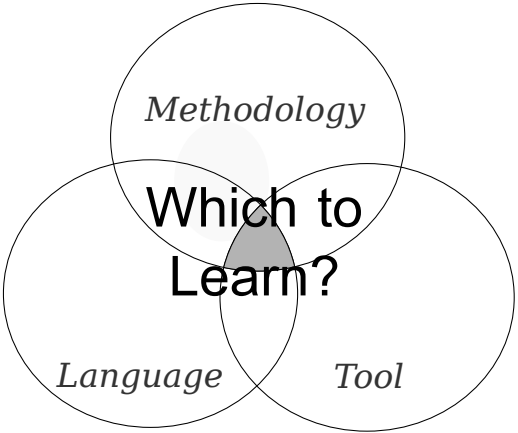

Figure: The Framework-Language-Tool framework.

授業で取り上げる範囲

取り上げること

- 方法を支えるための道具
- 良い道具には設計概念として方法論が組み込まれている
- 道具はプログラミング言語を問わない

取り扱わないこと

- 方法論そのものについてはアジャイル開発特論で学ぶ
- 言語の備えるエコシステムについては必要な範囲で学ぶ

Scrum するための道具

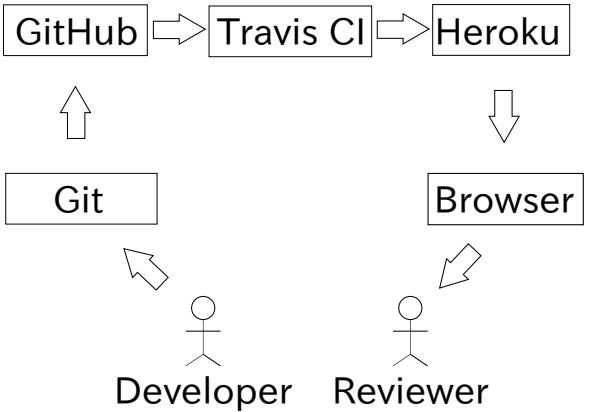

Figure: The modern tools for Scrum developments.

モダンな開発環境の全体像

仮想化技術（Virtualization）

- Windows や Mac で Linux 上での Web アプリケーション開発を学ぶことができる
- Heroku や Travis CI 等のクラウドでの実行や検査環境として用いられている

ソーシャルコーディング（Social Coding）

- Linux のソースコードの VCS として用いられている Git を学ぶ
- Git は GitHub と連携することで OSS 型のチーム開発ができる

enPiT 仮想化環境

インストール済みの言語と道具

- エディタ（Emacs/Vim）
- Ruby の実行環境
- GitHub，Heroku，Travis CI と連携するための各種コマンド（github-connect.sh，hub，heroku，travis）
- PostgreSQL のクライアント・サーバーと DB
- 各種設定ファイル（.bash_profile，.gemrc，.gitconfig）
- その他

仮想化環境の構築用リポジトリ（参考）

- ychubachi/vagrant_enpit

クラウドのアカウント作成

GitHub

- [Join GitHub · GitHub]

Heroku

- [Heroku - Sign up]

Travis CI

- [Travis CI]
  - Travis CI は，GitHub のアカウントでログインできる

enPiT 仮想化環境のアップデート

作業内容

- enPiT 仮想化環境（vagrant の box）を更新しておく

コマンド

```
cd ~/enpit
vagrant destroy
vagrant box update
```

Port Forward の設定

説明

- Guest OS で実行するサーバに，Host OS から Web ブラウザでアクセスできるようにしておく
- 任意のエディタで Vagrantfile を変更

変更前

```
# config.vm.network "forwarded_port", guest: 80, host: 8080
```

変更後

```
config.vm.network "forwarded_port", guest: 3000, host: 3000
config.vm.network "forwarded_port", guest: 4567, host: 4567
```

enPiT 仮想化環境にログイン

作業内容

- 前の操作に引き続き，仮想化環境に SSH 接続する

コマンド

```
vagrant up
vagrant ssh
```

github-connect スクリプト

URL

- [github-connect.sh]

git conifg を代行

- GitHub にログインし，名前と email を読み込んで git に設定

SSH の鍵生成と登録

- SSH 鍵を作成し，公開鍵を GitHub に登録してくれる

github-connect.sh の実行

作業内容

- スクリプトを起動し，設定を行う
- GitHub のログイン名とパスワードを聞かれるので，入力する
- rsa key pair のパスフレーズは入力しなくて構わない

コマンド

```
github-connect.sh
```

Git と GitHub の設定確認

Git の設定確認

```
git config --list
```

GitHub の設定確認

- ブラウザで GitHub の SSH Key ページを開く

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

第 2 章ローカルリポジトリの操作

Git のローカルリポジトリの作成

ローカルリポジトリ

- ソースコードや各種のファイルを保存し，開発に利用する
- 「my_enpit 」というディレクトリを作成し，初期化する

コマンド

```
mkdir ~/my_enpit
cd ~/my_enpit
git init
```

Git の設定ディレクトリ

隠しフォルダ「.git 」

- Git ソースコードの履歴情報や，各種の設定を Git が保存するディレクトリ
- このフォルダは通常，Git を経由しないで変更することはない

確認方法

```
ls -a
find .git
```

Hub コマンド

enPiT 環境の Hub コマンド

- github/hub

Git への GitHub 操作機能追加

- 通常の Git の機能に加えて，GitHub 用のコマンドが利用できる
- エイリアス設定しており，コマンド名は「git」のまま

確認方法

```
git version
alias git
```

Hub コマンドによるリモートリポジトリの作成

作業内容

- コマンドライン操作で，GitHub にリポジトリを作成する
- Hub コマンドの機能である `git create` を利用
- 初回既動時にはパスワードか聞かれる

コマンド

```
git create
```

リポジトリの確認方法

確認方法

- Web ブラウザで GitHub を開き，「`my_enpit` 」ができていることを確認

コマンドラインで確認

```
git remote -vv
```

Git の操作方法

マニュアル等

- Git - Documentation

commit ログの書き方

- Writing good commit messages ・erlang/otp Wiki

ステータスの確認

リポジトリの状態を確認する

- `git status` は，頻繁に利用するコマンド
- リポジトリの状態を確認することができる
- この表示の読み方を理解することが重要

コマンド

```
git status
```

ファイルの追加とステータスの確認

作業内容

- テキストエディタで README.md を作成
- ステータスの変化を見る

コマンド

```
emacs README.md
git status
```

Add/Commit の方法

ステージングエリアを利用する場合

- git add README.mb
- git commit -m 'First commit'

ステージングエリアを省略する場合

- git commit -a -m 'First commit'
  - トラックされていないファイルは commit しないので注意

Log の閲覧

コミットログ

- ソースコードに加えた変更の履歴を，commit を単位として閲覧できる

コマンド

```
git log
```

Push の方法

push とは？

- ローカルで作成した commit を，リモートのリポジトリにアップロードすること
- origin とは，リモートのリポジトリの内部的な名前
- upstream とは，ブランチ（後述）が紐づいているリポジトリのこと
- 最初にそのブランチを push するときは，`--setupstream` オプションを指定

コマンド

```
git push --set-upstream origin master
```

コミットのログを詳細に書く方法

エディタを使ったログの記述

- コミットのログや，Pull Request の記述を，より詳しく書くことができる
- `commit` や `pull_request` から `-m` オプションを外すと，エディタが立ち上がる
  - エディタは emacs を起動するようになっている
  - `C-x C-s` で保存，`C-x C-c` で終了

コマンド

```
git commit
git pull_request
```

Init/Status/Add の練習

演習課題

1. 解説した手順に従い，my_enpit リポジトリを作成
2. git status コマンドを実行
3. README.md ファイルを作成しなさい
4. git status コマンドを実行し，変化を見なさい
5. commit しなさい．ログを必ず書くこと
6. git status コマンドを実行し，変化を見なさい

Commit/Log/Push の練習

演習課題

1. README.md を修正して commit しなさい
2. 新しいファイルを作成して commit しなさい
3. 作業が完了したら，push しなさい（`--set-upstream` が必要）
4. コミットが push されていることを Web ブラウザで確認しなさい
5. 作成したファイルを削除して commit して push しなさい
6. エディタを使って，詳細なログを書きなさい
7. その他，自由に commit の作業を試しなさい

ここまでの課題の提出

提出物

- 下記のものを提出してください
  - GitHub と Heroku アカウント
  - 作成した my_enpit リポジトリの URL

提出先

- [enPiT 演習アカウント (2014)]

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

第 3 章ブランチとリモートリポジトリ

GitHub Flow (1)

1. 思い立ったらブランチ作成
   - 新しい機能追加や，アイディアを試す
2. ブランチにコミットを追加
   - 変更点をコミットとして作成
   - コミットのログは，他人が読んでわかるように書く
3. Pull Request を開く
   - コミットについて，意見交換ができる
   - 作業途中で Pull Request を出しても構わない

GitHub Flow (2)

1. 議論とレビュー
   - レビューをしたり，質疑応答をしたりする
2. マージしてディプロイ
   - `master` ブランチにマージする（自動でディプロイ）
   - マージの前にテストしたいときは，ローカルで試す

参考文献

- Understanding the GitHub Flow ・GitHub Guides

branch の作成

ブランチとは？

- リポジトリには master ブランチがある
- 新しい作業を行う場合，必ず branch を切る

コマンド

```
git branch new_branch
git branch -vv
```

branch の checkout

branch を切り替える

- checkout してブランチを切り替える
- ブランチを commit することができる
- 切り替える前に，ブランチでの作業は commit しておく（stash も可）

コマンド

```
git checkout new_branch
< 編集作業 >
git commit -a -m 'Create a new branch'
```

他の branch を merge する

merge とは

- ブランチで作業した内容（commit）を，他のブランチに統合すること
- new_branch での作業を master に統合する場合，最初に master を checkout する

コマンド操作

```
git checkout master
git merge new_branch
```

Conflict（競合）とその解消

Conflict とは

- branch で行う作業がかち合った場合，発生する
- merge する際，conflict が生じた場合，エラーになる

解消方法

- エディタ等で編集を行い，解消する

参考文献

- Resolving a merge conflict from the command line - GitHub Help

Branch の Push

リモートへの Push

- Branch を GitHub に Push することができる
- master ブランチを Push した際と同様，upstream を指定する
- Push できたかどうかを Web ブラウザで確認する

コマンド

```
git push --set-upstream origin new_branch
```

Pull Request の作成

Pull Roquest とは？

- push した branch での作業の統合（merge）を依頼する
- hub コマンドの `pull-request` で発行できる

コマンド

```
git pull-request -m 'Update a new branch'
```

Pull Request の merge

Pull Request をレビューする

- Web ブラウザで Pull Request を確認する

ブラウザで merge

- 問題なければ merge ボタンを押す

コマンドラインで merge する場合

```
git merge pull_request_URL
```

Branch の Pull

Branch を Pull するとは

- リモートで行われた変更を適用すること
- 内部的には fetch でダウンロードしてから merge する

コマンド

```
git checkout master
git pull
```

branch の操作（ローカル）

演習課題

1. `my_enpit` リポジトリでブランチを作成しなさい（`new_branch` ）
2. `checkout` で `new_branch` に移動する
3. ファイルを編集し commit する
4. `master` ブランチに移動してファイルの内容が「編集されていないこと」を確認しなさい
5. `merge` して，変更を適用しなさい

競合の発生と解消

演習課題

1. `new_branch` でファイルを編集して，commit する
2. `master` に移動し，ファイルの同じ箇所を編集して，commit する
3. `master` に `new_branch` を merge して，コンフリクトを発生させる
4. エディタで競合箇所を修正して commit する

リモートの branch の操作

演習課題

1. 新しいブランチを作成して，remote に push する
2. Pull Request を送る
3. ブラウザで，Pull Request をマージする
4. `master` ブランチに移動して，`pull` することで，更新する

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

第 4 章 GitHub を使った協同作業

リモートのリポジトリを Clone

Clone とは

- GitHub で公開されているリポジトリはだれでも複製（clone）できる
- ソースコードはローカルにコピーされ，閲覧やコンパイルなどができるようになる
- アクセス権限がない場合は，push できない

コマンド

```
git clone octocat/Spoon-Knife
```

Pull Request をチェックアウト

Pull Request のチェックアウト

- 誰かが作成した Pull Request の内容を，ブランチとしてローカルにコピーする
- 試しに動作させたり，コードをチェックするときなどに利用

コマンド

```
git checkout https://github.com/octocat/Spoon-Knife/pull/3166
```

オリジナルのリポジトリを Fork する

Fork とは

- Clone したリポジトリを，自分のアカウントが所持するリポジトリとして GitHub 上で複製する
- `remote` の値は，オリジナルのリポジトリが `origin`，自分のリポジトリは自分の GitHub ユーザ名になる

コマンド

```
git fork
git remote -vv
```

ブランチを作成し自分のリポジトリに push

オリジナルの改変等

- 新しい機能追加等を行う場合，ブランチを作成する
- ブランチは，自分のリポジトリに push する

コマンド

```
git branch my_branch
git checkout my_branch
＜編集＞
git commit -a -m 'Update'
git push -u ychubachi my_branch
```

Fork した元に Pull Request を送る

コードのレビューやマージを依頼する

- 新しい機能ができたら，オリジナルに Pull Request を送り，レビューやマージをしてもらう

コマンド

```
git pull_request -m 'Pull Request'
```

Issue/Wiki

Issue

- 課題管理（ITS: Issue Tracking System）
- コミットのメッセージで close できる
  - Closing issues via commit messages ・GitHub Help

Wiki

- GitHub のリポジトリに Wiki を作る
  - About GitHub Wikis ・GitHub Help

GitHub

GitHub Pages

- 特殊なブランチを作成すると，Web ページが構築できる
  - GitHub Pages

Git brame

- だれがどの作業をしたかわかる（誰がバグを仕込んだのかも）
  - Using git blame to trace changes in a file ・GitHub Help

our_enpit にファイルを追加する

演習課題

1. ychubich/our_enpit を clone して fork する
2. 新しいブランチを作成し，新規にファイルを追加する
   - 内容は任意（自己紹介など）
   - Markdown で書いてください（拡張子は.md）
3. コミットを作成し，pull request を送信する
4. 教員がマージ作業を行います

既存のファイルを変更する

演習課題

1. README.md を改変して，pull request を送信する
2. GitHub の Pull Request 一覧を確認する
3. おそらくコンフリクトが発生するので，GitHub の指示に従い競合を解消する

隣の人との協同作業

演習課題

1. 新しくリポジトリを作成する（名称は任意）
2. 互いに，隣の席の人にリポジトリ名を教え，fork してもらい Pull Request を送ってもらう
3. マージしてあげる
4. 2〜3 を繰り返し，協同作業を行ってみよう

Issue/Wiki の利用

演習課題

- GitHub の Issue の機能を使ってみなさい
- commit のログで Issue をクローズさせてみなさい
- Wiki を作ってください

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

第 5 章 Sinatra アプリの開発

Sinatra を使った簡単な Web アプリケーション

Sinatra とは？

- Web アプリケーションを作成する DSL
- Rails に比べ軽量で，学習曲線が緩やか

参考文献

- Sinatra

Sinatra アプリ用リポジトリを作成する

内容

- Sinatra アプリを作成するため，新しいリポジトリを作る

コマンド

```
mkdir ~/sinatra_enpit
cd ~/shinatra_enpit
git init
git create
```

Sinatra アプリを作成する

コマンド

```
emacs hello.rb
git add hello.rb
git commit -m 'Create hello.rb'
```

コード: hello.rb

```
require 'sinatra'

get '/' do
  "Hello World!"
end
```

Sinatra アプリを起動する

起動の方法

- hello.rb を ruby で動かせば，サーバが立ち上がる
- vagrant の port forward を利用するためのオプションを追加する
  - ruby - Unable to access Sinatra app on host machine with Vagrant forwarded ports - Stack Overflow

コマンド

```
ruby hello.rb -o 0.0.0.0
```

Sinatra アプリの動作確認

動作確認の方法

- Host OS の Web ブラウザで，`http://localhost:4567` にアクセスする．

コマンドラインで Heroku にログインする

内容

- enPiT 環境には heroku コマンドをインストールしてある
- heroku コマンドを用いて，Heroku にログインできる
- 以後の作業は Heroku コマンドを利用する

コマンド

```
heroku login
```

heroku に SSH の公開鍵を設定する

内容

- Heroku も git のリモートリポジトリである
- ここに公開鍵でアクセスできるようにする

コマンド

```
heroku keys:add
```

確認

```
heroku keys
```

Heroku で動作できる Sinatra アプリ

内容

- Heroku で動作できる Sinatra アプリと設定ファイルの例 Deploying Rack-based Apps | Heroku Dev Center
- 例を見ながら，エディタを用いて，次の 3 つのファイルを作成する

`hello.rb`	Ruby による Web アプリ本体（作成済み）
`config.ru`	Web アプリサーバ（Rack）の設定
`Gemfile`	アプリで利用するライブラリ（Gem）

コマンド

```
emacs config.ru
emacs Gemfile
```

アプリを GitHub に push する

内容

- Heroku で動かす前に，commit が必要
- ついでに，GitHub にコードを push しておく
  - この場合の push 先は `origin master`

コマンド

```
git add .
git commit -m 'Add configuration files for Heroku'
git push origin master
```

Heroku にアプリを作る

アプリを作る

- Heroku が自動生成した URL が表示されるので，メモする
- `git remote -v` で heroku という名前の remote が追加されたことが分かる
- Web ブラウザで Heroku の管理画面を開くと，アプリができていることが確認できる

コマンド

```
heroku create
git remote -v
```

Heroku にアプリを配備する

配備する方法

- Heroku のリモートリポジトリに push する
- Web ブラウザでアプリの URL を開き，動作を確認する

コマンド

```
git push heroku master
```

Sinatra アプリの作成

演習課題

- Sinatra アプリを作成して，Heroku で動作させなさい
- Sinatra の DSL について調べ，機能を追加しなさい
- コミットのログは詳細に記述し，どんな作業を行ったかが他の人にも分かるようにしなさい
- 完成したコードは GitHub にも push しなさい

Sinatra アプリの共同開発

演習課題

- 隣の席の人と協同で Sinatra アプリを開発しなさい
- 一方が GitHub のリポジトリを作成し，もう一人が Fork する
- 最初に，どんな機能をもたせるかを相談しなさい
  - メンバーのスキルに合わせて，できるだけ簡単なもの
  - データベースは使わない
- ブランチを作成し，Pull Request を送る

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

第 6 章 Ruby on Rails アプリの開発

RoR を使った Web アプリケーション

Ruby on Rails（RoR）とは？

- Web アプリケーションを作成するためのフレームワーク

参考文献

- Ruby on Rails

Heroku で動かす方法

Getting Started

- Getting Started with Rails 4.x on Heroku | Heroku Dev Center

DB について

- Databese は PostgreSQL を使用する
  - RoR 標準の sqlite は使わない

rails_enpit リポジトリを作成する

内容

- rails は予め，仮想化環境にインストールしてある
- rails new コマンドを用いて，RoR アプリの雛形を作成する

コマンド

```
rails new ~/rails_enpit --database=postgresql
cd ~/rails_enpit
git init
git create
git add .
git commit -m 'Generate a new rails app'
git push -u origin master
```

Gemfile の変更

変更する内容

- Gemfile に Rails 内部で動作する JavaScript の実行環境を設定する
- 当該箇所のコメントを外す
- 変更を commit しておく

変更前

```
# gem 'therubyracer',  platforms: :ruby
```

変更後

```
gem 'therubyracer',  platforms: :ruby
```

Bundle install の実行

bundle install

- Gemfile を読み込み，必要な gem をインストールする
- `rails new` をした際にも，`bundle install` は実行されている
- 今回は therubyracer と，それが依存している gem でまだインストールしていないものをインストール
- インストールする先は `~/.rbenv` 以下の特定のディレクトリ

コマンド

```
bundle install
git commit -a -m 'Run bundle install'
```

Rails server の起動

Rails server を起動

- この段階で，アプリケーションを起動できるようになっている
- Host OS の Web ブラウザで，`http://localhost:3000` にアクセスして確認
- 端末にもログが表示される
- 確認したら，端末で Ctrl-C を押してサーバを停止する

コマンド

```
rails server
```

Hello World を表示する Controller

Controller とは？

- MVC 構造でいう Controller
- HTTP のリクエストを処理し，View に引き渡す
- `rails generate controller` コマンドで作成する

コマンド

```
rails generate controller welcome
```

View の作成

View とは？

- HTML 等で結果をレンダリングして表示する
- app/views/welcome/inde.html.erb を作成する
- erb で作成するのが一般的で，内部で Ruby コードを動作させることができる

index.html.erb

```
<h2>Hello World</h2>
<p>
  The time is now: <%= Time.now %>
</p>
```

root となる route の設定

Route とは？

- HTTP のリクエスト（URL）とコントローラを紐付ける設定
- ここでは `root` へのリクエスト（`GET /`）を `welcome` コントローラの `index` メソッドに紐付ける
- `rake routes` で確認する

`config/routes.rb` の当該箇所をアンコメント

```
root 'welcome#index'
```

Controller と View の動作確認

動作確認の方法

- 再度，`rails server` でアプリを起動する
- Web ブラウザで `http://localhost:3000/` を開いて確認する

コマンド

```
rails server
```

ここまでをコミットしておく

ここまでの内容

- ここまでの作業で，controller と view を 1 つ備える RoR アプリができた
- 作業が一区切りしたので，commit する（commit はひとかたまりの作業に対して行う）

コマンド

```
git add .
git commit -m 'Create welcome controller and view'
```

Gemfile の設定

Heroku 用 Gem

— `Gemfile` に `rails_12factor` を追加する

- Ruby のバージョンも指定しておく
- `Gemfile` を変更したら必ず `bundle install` すること

`Gemfile` に追加する内容

```
gem 'rails_12factor', group: :production
ruby '2.1.3'
```

Git にコミット

コミットする必要性

- Heroku にコードを送るには，git を用いる
- ローカルで最新版を commit しておく必要がある
- ついでに GitHub にも push しておく

コマンド

```
git commit -a -m 'Set up for Heroku'
git push # origin master -> GitHub が省略されている
```

Heroku アプリの作成とディプロイ

作成とディプロイ

- `heroku` コマンドを利用してアプリを作成する
- `heroku create` で表示された URL を開く
- `git push` でディプロイすると，Heroku からのログが流れてくる

コマンド

```
heroku create
git push heroku master
```

RoR アプリの作成

演習課題

- ここまでの説明に従い，Heroku で動作する RoR アプリを完成させなさい

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

第 7 章 RoR アプリで DB を使う

PostgreSQL に DB を作成

開発で利用する DB

rails_enpit_development 開発作業中に利用

rails_enpit_test テスト用に利用

rails_enpit_production 本番環境で利用（ローカルには作成しない）

コマンド

```
createdb rails_enpit_development
createdb rails_enpit_test
```

Scaffold

Scaffold とは

- scaffold - Google 検索
- RoR では，MVC の雛形を作る
- CRUD 処理が全て実装される

コマンド

```
rails generate scaffold book title:string author:string
```

DB の migrate

migrate とは

- Database のスキーマ定義の更新
- Scaffold を追加したり，属性を追加したりした際に行う

コマンド

```
rake db:migrate
```

動作確認

動作確認の方法

- Web ブラウザで `http://localhost:3000/books` を開く
- CRUD 処理が完成していることを確かめる

テストについて

ガイド

- A Guide to Testing Rails Applications —Ruby on Rails Guides

コマンド

```
rake test
```

Scaffold の作成

Travis CI の Web 管理画面

GitHub と Travis CI 連携

Travis 経由での Heroku への deploy

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

第 7 章 Web API を活用したサービス構築

- 楽天ウェブサービス: ご利用ガイド

- rakuten-ws/rws-ruby-sdk

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

< 演習 > Ruby on Rails を用いた開発演習 (1)

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

< 演習 > Ruby on Rails を用いた開発演習 (2)

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

[講義] ミニプロジェクト

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

< 演習 > ミニプロジェクト演習 (1)

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

< 演習 > ミニプロジェクト演習 (2)

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

memo

33 演習で作成したリポジトリ

34 バージョン管理の概念

- my_enpit
- our_enpit -> 事前に教員が用意
- 二人で行う演習のもの
- sinatra_enpit
- rails_enpit

33 演習で作成したリポジトリ

34 バージョン管理の概念

シナリオ

HTML による Web ページ

index.html を作りブラウザで開く

バージョン管理の基礎知識

diff

patch

sha1

演習課題

演習課題

- あなたがよく知っている「歴史上の有名人」を一人取り上げる
- その人を紹介する Web ページを作成する
- HTML を作成する（リンクや画像の埋め込みにもチャレンジ）
- git でバージョン管理
- GitHub に push する

GitHub に push

コマンド

```
git commit -a -m 'First commit'
git push -u origin msater
```

ビジネスアプリケーション演習

中鉢欣秀・上田隆一

産業技術大学院大学 (AIIT)

Tasks

35 **DONE** chocolatey のインストールを kazam でキャプチャする

36 **TODO** 英語の原典を読めるようになること

37 **TODO** よくある間違い cd しないで git init するとか.

38 **TODO** OS をインストールし，手順書を参照しながら長々とコマンドを打つ，ということが不要になった.

39 **TODO** アンケートを作成する

40 **TODO** .bash_profile から.bashrc を読み込む（カラー化）

35 DONE chocolatey のインストールを kazam でキャプチャする

36 TODO 英語の原典を読めるようになること

37 TODO よくある間違い cd しないで git init するとか.

38 TODO OS をインストールし，手順書を参照しながら長々とコマンドを打つ，ということが不要になった.

39 TODO アンケートを作成する

40 TODO .bash_profile から.bashrc を読み込む（カラー化）

35 **DONE** chocolatey のインストールを kazam でキャプチャする

36 **TODO** 英語の原典を読めるようになること

37 **TODO** よくある間違い cd しないで git init するとか.

38 **TODO** OS をインストールし，手順書を参照しながら長々とコマンドを打つ，ということが不要になった.

39 **TODO** アンケートを作成する

40 **TODO** .bash_profile から.bashrc を読み込む（カラー化）

35 **DONE** chocolatey のインストールを kazam でキャプチャする

36 **TODO** 英語の原典を読めるようになること

37 **TODO** よくある間違い cd しないで git init するとか.

38 **TODO** OS をインストールし，手順書を参照しながら長々とコマンドを打つ，ということが不要になった.

39 **TODO** アンケートを作成する

40 **TODO** .bash_profile から.bashrc を読み込む（カラー化）

35 **DONE** chocolatey のインストールを kazam でキャプチャする

36 **TODO** 英語の原典を読めるようになること

37 **TODO** よくある間違い cd しないで git init するとか.

38 **TODO** OS をインストールし，手順書を参照しながら長々とコマンドを打つ，ということが不要になった.

39 **TODO** アンケートを作成する

40 **TODO** .bash_profile から.bashrc を読み込む（カラー化）

- 調査の目的
  - モダンなソフトウエア開発の理解度（これは 2 回やる）
    - git について 90%（業務でのソフトウエア開発に利用できる）, 70%, 50%, 30%, 10% （ほとんど知らない・使ったことはない）
  - PBL のために，事前学習が役に立ったか（これは PBL 後）事前学習をした人とそうでない人とで，PBL の満足感，達成感が違うか円滑に PBL をすすめることができたか
- 方法論

あなたは BizApp 演習の内容を学習しましか？

1. 授業を履修した
   1. ビデオを視聴した
      1. 学習していない
2. 道具

35 **DONE** chocolatey のインストールを kazam でキャプチャする

36 **TODO** 英語の原典を読めるようになること

37 **TODO** よくある間違い cd しないで git init するとか.

38 **TODO** OS をインストールし，手順書を参照しながら長々とコマンドを打つ，ということが不要になった.

39 **TODO** アンケートを作成する

40 **TODO** .bash_profile から.bashrc を読み込む（カラー化）