株式会社万葉の新入社員教育用カリキュラム　https://everyleaf.com/

README.md

el-training

このカリキュラムについて

この文書は、万葉で必須とされるRuby on Railsとその周辺技術の基礎を習得するための新入社員教育用カリキュラムです。 新入社員の能力によらず、必ず一通りのステップを実施していただきます。 研修期間は特に定めておりません。 すべてのステップを完了した時点で研修修了となります。

本カリキュラムでは、以下の登場人物を想定しています。

- 新入社員（メンティー）：本カリキュラムの受講者です。
- メンター：新入社員の教育・指導・助言を行います。また、新入社員と相談して仕様を一緒に決めたりする役割も担います。

指導について、メンターがどの程度関与するかどうかはメンターの裁量に一任します。また、研修期間については、新入社員のスキルレベルや社内の案件状況を考慮して、メンターの方であらかじめ目安を設定する予定です。

ライセンス

概要

システムの要件

本カリキュラムでは、課題としてタスク管理システムを開発していただきます。 タスク管理システムでは、以下のことを行いたいと考えています。

- 自分のタスクを簡単に登録したい
- タスクに終了期限を設定できるようにしたい
- タスクに優先順位をつけたい
- ステータス（未着手・着手・完了）を管理したい
- ステータスでタスクを絞り込みたい
- タスク名・タスクの説明文でタスクを検索したい
- タスクを一覧したい。一覧画面で（優先順位、終了期限などを元にして）ソートしたい
- タスクにラベルなどをつけて分類したい
- ユーザ登録し、自分が登録したタスクだけを見られるようにしたい

また、上記の要件を満たすにあたって、次のような管理機能がほしいと考えています。

- ユーザの管理機能

サポートブラウザ

- サポートブラウザはmacOS/Chrome各最新版を想定しています

アプリケーション（サーバ）の構成について

以下の言語・ミドルウェアを使って構築していただきたいです（いずれも最新の安定バージョン）。

- Ruby
- Ruby on Rails
- MySQL

サーバについては以下を利用していただきたいです。

- Heroku
- 【オプション要件】AWSアカウントで、EC2のインスタンスを作成して構築する
  - t2.micro（Amazon Linux）
  - RDS

※ 性能要求・セキュリティ要求は特に定めませんが、一般的な品質で作ってください。 あまりにサイトのレスポンスが悪い場合は改善をしていただきます

本カリキュラムの最終目標

本カリキュラムの終了時点で、以下の項目を習得することを想定しています。

- Railsを利用した基本的なWebアプリケーションの実装ができるようになること
- Railsアプリケーションで一般的な環境を使ってアプリケーションを公開できること
- 公開されたRailsアプリケーションに対して、機能の追加やデータのメンテナンスができること
- GitHubでPRをしてマージする一連の流れを習得すること。また、それに必要なGitのコマンドを習得すること
  - 適切な粒度でコミットができること
  - 適切にPRの説明文が書けること
  - レビューに対する対応と修正が一通りできること
- 不明な点を適切なタイミングでチームメンバーや関係者に（今回はメンターになります）口頭やチャットなどで質問ができること

課題ステップ

ステップ1: Railsの開発環境を構築しよう

1-1: Rubyのインストール

- rbenvを利用して最新バージョンのRubyをインストールしてください
- `ruby -v` コマンドでRubyのバージョンが表示されることを確認してください

1-2: Railsのインストール

- Gemコマンドでrailsをインストールしましょう
- 最新バージョンのRailsをインストールしてください
- `rails -v` コマンドでRailsのバージョンが表示されることを確認してください

1-3: データベース（MySQL）のインストール

- 手元のOSでMySQLをインストールしましょう
  - macOSの場合は `brew` などでインストールしてください

ステップ2: GitHubにリポジトリを作成しよう

- 手元にGitをインストールしましょう
  - macOSの場合は `brew` などでインストールしてください
  - `gitconfig` でユーザ名、メールアドレスを登録しましょう
- アプリ名（=リポジトリ名）を考えましょう
- リポジトリを作成しましょう
  - アカウントがない場合は取得します
  - その上で空のリポジトリを作成します

ステップ3: Railsプロジェクトを作成しよう

- `rails new` コマンドでアプリケーションに最低限必要なディレクトリやファイルを作成しましょう
- `rails new` してできたプロジェクトのディレクトリ（アプリ名のディレクトリ）の直下に `docs` というディレクトリを作り、この文書ファイルをコミットしましょう
  - このアプリの仕様を管理下に置き、いつでも見られるようにするためです
- 作成したアプリをGitHub上に作成したリポジトリにpushしましょう
- バージョンを明示するため、利用するRubyのバージョンを `Gemfile` に記載しましょう（Railsは既にバージョンが記載されていることを確認しましょう）

ステップ4: 作りたいアプリケーションのイメージを考える

- 設計を進める前に、どのようなアプリになるか完成イメージを（メンターと一緒に）考えてみましょう。ペーパープロトタイピングによる画面設計などがおすすめです
- システムの要件を読んで、必要なデータ構造を考えてみましょう
  - どのようなモデル（テーブル）が必要そうか
  - テーブル内にどのような情報が必要そうか
- データ構造を考えたら、それをモデル図に手書きで書いてみましょう
  - 完成したら撮影して、リポジトリに入れましょう
  - `README.md` にテーブルスキーマを記載しましょう（モデル名・カラム名・データ型）

※ 現時点で正解のモデル図を作成する必要はまだありません。現時点での想定として作ってみましょう（今後のステップで間違いと思ったら改修していくイメージです）

ステップ5: データベースの接続設定（周辺設定）をしましょう

- まずGitで新たにトピックブランチを切りましょう
  - 以降、トピックブランチ上で作業をしてコミットをしていきます
- Bundlerをインストールしましょう
- `Gemfile` で `mysql2` （MySQLのデータベースドライバ）をインストールしましょう
- `database.yml` の設定をしましょう
- `rails db:create` コマンドでデータベースの作成をしましょう
- `rails db` コマンドでデータベースへの接続確認をしましょう
- GitHub上でPRを作成してレビューしてもらいましょう
  - コメントがついたらその対応を行ってください。LGTM（Looks Good To Me）が2つついたらmasterブランチにマージしましょう

ステップ6: タスクモデルを作成しましょう

タスクを管理するためのCRUDを作成します。 まずは名前と詳細だけが登録できるシンプルな構成で作りましょう。

- `rails generate` コマンドでタスクのCRUDに必要なモデルクラスを作成しましょう
- マイグレーションを作成し、これを用いてテーブルを作成しましょう
  - マイグレーションは1つ前の状態に戻せることを担保できていることが大切です！ `redo` を流して確認する癖をつけましょう
- `rails c` コマンドでモデル経由でデータベースに接続できることを確認しましょう
  - この時に試しにActiveRecordでレコードを作成してみる
- GitHub上でPRを作成してレビューしてもらいましょう

ステップ7: タスクを登録・更新・削除できるようにしよう

- タスクの一覧画面、作成画面、詳細画面、編集画面を作成しましょう
  - `rails generate` コマンドでコントローラとビューを作成します
  - コントローラとビューに必要な実装を追加しましょう
  - 作成、更新、削除後はそれぞれflashメッセージを画面に表示させましょう
- `routes.rb` を編集して、 `http://localhost:3000/` でタスクの一覧画面が表示されるようにしましょう
- GitHub上でPRを作成してレビューしてもらいましょう
  - 今後、PRが大きくなりそうだったらPRを2回以上に分けることを検討しましょう

ステップ8: テスト（feature spec）を書こう

- specを書くための準備をしましょう
  - `spec/spec_helper.rb` 、 `spec/rails_helper.rb` を用意しましょう
- feature specをタスク機能に対して書きましょう
- Circle CIなどのCIツールを導入して、Slackに通知するようにしましょう
  - オプションですが、メンターが利便性のために追加しても可です
- 参考書籍：https://leanpub.com/everydayrailsrspec-jp

ステップ9: Railsのi18nを使ってアプリの日本語部分を共通化しましょう

- Railsのi18nの仕組みを利用して、日本語リソースの共通化をしましょう

※ i18n化を行うと後のステップでメッセージを出したりするのが楽になるなどのメリットがあります

ステップ10: Railsのタイムゾーンを設定しましょう

- Railsのタイムゾーンを日本（東京）に設定しましょう

ステップ11: タスク一覧を作成日時の順番で並び替えましょう

- 現在IDの順で並んでいますが、これを作成日時の降順で並び替えてみましょう
- 並び替えがうまく行っていることをfeature specで書いてみましょう

ステップ12: バリデーションを設定してみよう

- バリデーションを設定しましょう
  - どのカラムにどのバリデーションを追加したらよいか考えてみましょう
  - 合わせてDBの制約も設定するマイグレーションを作成しましょう
  - マイグレーションファイルだけを作成するため、 `rails generate` コマンドで作成します
- 画面にバリデーションエラーのメッセージを表示するようにしましょう
- バリデーションに対してモデルのテストを書いてみましょう
- GitHub上でPRを作成してレビューしてもらいましょう

ステップ13: デプロイをしよう

- masterブランチにシンプルなタスク管理システムができたので、デプロイしてみましょう。
- Herokuにデプロイを実施してみましょう
  - アカウントがなければ作成します
- デプロイされたHeroku上のアプリを触ってみよう
  - これからはこのアプリにタスクを登録して開発を進めましょう
  - ※ ただし、Herokuのアプリケーションはインターネットでどこでも参照できるので、公開してはまずい情報は載せないようにしましょう
    - Basic認証をこの時点でいれてもいいかもしれません
  - 今後、ステップが終わるたびにHerokuへ自分のアプリを継続的にデプロイしましょう
- デプロイの方法を `README.md` に記載しましょう
  - その際に、このアプリで使っているフレームワークのバージョン情報なども記載しておくとなおよいです

ステップ14: 終了期限を追加しよう

- タスクに対して、終了期限を登録できるようにしてみましょう
- 一覧画面で、終了期限でソートできるようにしましょう
- specを拡充しましょう
- PRしてレビューをしてもらったら、リリースしてみましょう

ステップ15: ステータスを追加して、検索できるようにしよう

- ステータス（未着手・着手中・完了）を追加してみよう
  - 【オプション要件】初学者ではない場合はstateを管理するGemを導入しても構いません
- 一覧画面でタイトルとステータスで検索ができるようにしよう
  - 【オプション要件】初学者ではない場合はransackなどの検索の実装を便利にするGemを導入しても構いません
- 絞り込んだ際、ログを見て発行されるSQLの変化を確認してみましょう
  - 以降のステップでも必要に応じて確認する癖をつけましょう
- 検索インデックスを貼りましょう
- 検索に対してmodel specを追加してみよう（feature specも拡充しておきましょう）

ステップ16: 優先順位を設定しよう（※類似した実装経験のある人は省略可）

- タスクに対して、優先順位（高中低）を登録できるようにしよう
- 優先順位でソートできるようにしましょう
- feature specを拡充しましょう
- PRしてレビューをしてもらったら、リリースしてみましょう（以降続けてください）

ステップ17: ページネーションを追加しよう

- KaminariというGemを使って一覧画面にページネーションを追加してみましょう

ステップ18: デザインを当てよう

- Bootstrapを導入して、これまで作成したアプリにデザインを当てましょう
  - 【オプション要件】自分でCSSを書いてデザインする

ステップ19: 複数人で利用できるようにしよう（ユーザの導入）

- ユーザモデルを作成してみましょう
- 最初のユーザをseedで作成してみましょう
- ユーザとタスクが紐づくようにしましょう
  - 関連に対してインデックスを貼りましょう
  - N+1問題を回避するための仕組みを取り入れましょう
  - ※ Herokuにデプロイした際に、すでに登録されているタスクとユーザが紐づいているようにしてください（データメンテナンス）

ステップ20: ログイン/ログアウト機能を実装しよう

- 追加のGemを使わず、自分で実装してみましょう
  - DeviseなどのGemを使わないことで、HTTPのCookieやRailsにおけるSessionなどの仕組みについて理解を深めることが目的です
  - また、一般的な認証についての理解を深めることも目的にしています（パスワードの取り扱いについてなど）
- ログイン画面を実装しましょう
- ログインしていない場合は、タスク管理のページに遷移できないようにしましょう
- 自分が作成したタスクだけを表示するようにしましょう
- ログアウト機能を実装しましょう

ステップ21: ユーザの管理画面を実装しよう

- 画面上に管理メニューを追加しましょう

- 管理画面にはかならず `/admin` というURLを先頭につけるようにしましょう
  - `routes.rb` に追加する前に、あらかじめURLやルーティング名（ `*_path` となる名前）を想定して設計してみましょう
- ユーザ一覧・作成・更新・削除を実装しましょう
- ユーザを削除したら、そのユーザが抱えているタスクを削除するようにしてみましょう
- ユーザの一覧画面で、ユーザが持っているタスクの数を表示するようにしてみましょう
- ユーザが作成したタスクの一覧が見られるようにしてみましょう

ステップ22: ユーザにロールを追加しよう

- ユーザに管理ユーザと一般ユーザを区別するようにしてみましょう
- 管理ユーザだけがユーザ管理画面にアクセスできるようにしてみましょう
- ユーザ管理画面でロールを選択できるようにしましょう
- 管理ユーザが1人もいなくならないように削除の制御をしましょう
- ※ Gemの使用・不使用は自由です

ステップ23: タスクにラベルをつけられるようにしよう

- タスクに複数のラベルをつけられるようにしてみましょう
- つけたラベルで検索できるようにしてみましょう

ステップ24: エラーページを適切に設定しましょう

- Railsが用意しているデフォルトのエラーページを自分が作った画面にしてみましょう
- 状況に応じて、適切にエラーページを設定しましょう
  - ステータスコードの404ページと500ページの2種類の設定は少なくとも必須とします

あとがき

お疲れさまでした。 あなたは教育カリキュラムを一通り完遂しました!!

このカリキュラムでは触れきれませんでしたが、今後は以下のトピックなどが必要になると思うので、学習を進めていくとよいと思います（案件を通じて学ぶことも多いと思います）。

- Webアプリケーションの基本的な理解を深める
  - HTTPとHTTPSに関する理解
- Railsのもう少し進んだ使い方を習得する
  - STI
  - ロギング
  - 明示的なトランザクション
  - 非同期処理
  - アセットパイプライン（どちらかというとリリース系のトピック）
- JavaScriptやCSSなどのフロントエンドに関するより高度な理解
- データベースに対する理解を深める
  - SQL
  - よりパフォーマンスを重視したクエリの構築
  - インデックスの理解をより深める
- サーバ環境に関するより多くの理解
  - Linux OS
  - Webサーバ（Nginx）の設定
  - アプリケーションサーバ（Unicorn）の設定
  - MySQLに関する設定への理解
- リリースに関するツールの理解
  - Capistrano
  - Ansible

（番外編）オプション要件

必須要件とは別に、タスク管理システムに関するオプション要件を以下に示します。 メンターと相談しつつ必要に応じて実施してみてください。

オプション要件1: 終了間近や期限の過ぎたタスクがあった場合、アラートを出したい

- ログインした時点で、どこかに終了間近や期限の過ぎたタスクを表示してみましょう
- 既読などが分かるようになるとよりよい

オプション要件2: ユーザの間でタスクを共有できるようにしたい

- 複数人で同じタスクを参照・編集できるようにしたい
  - 例：メンターとメンティーでタスクを共有できること
- 作成者を表示する

オプション要件3: グループを設定できるようにしたい

- オプション要件2の続き
- グループを設定できるようにして、グループ内だけでタスクを参照できるようにしたい

オプション要件4: タスクに添付ファイルをつけられるようにしたい

- タスクに対して添付ファイルをつけられるようにしてみましょう
- Herokuの場合はS3バケットにアップロードした添付ファイルを管理するようにします
- 適切にGemを選別して利用してみましょう

オプション要件5: ユーザにプロフィール画像を設定できるようにしてみましょう

- ユーザがプロフィール画像を設定できるようにしましょう
- アップロードした画像はアイコンとして使用するので、遅くならないようにサムネイル画像も作りたい
- 適切にGemやライブラリを選別しましょう

オプション要件6: タスクカレンダーの機能がほしい

- 終了期限を可視化するために、カレンダーに終了期限別にタスクを表示するようにしてみましょう
- ライブラリの使用・不使用は自由です

オプション要件7: タスクをドラッグ&ドロップで並べ替えられるようにしたい

- タスク一覧でタスクをドラッグ&ドロップして並べ替えできるようにしてみましょう

オプション要件8: ラベルの使用頻度をグラフ化してみましょう

- 統計情報を可視化するために、グラフを導入してみましょう
- グラフの種類は見やすいものを提案してみましょう

オプション要件9: 終了間近のタスクを作成ユーザ宛てにメール通知してみましょう

- 終了間近のタスクがある場合、バックグラウンドでメール通知してみましょう
- メール送信はクラウドサービスを利用します
  - HerokuならばSendGrid
  - AWSならばAmazon SESなど
- 1日1回のバッチで送信するようにしてみましょう
  - HerokuならばHeroku Scheduler（アドオン）
  - AWSだったらcronを設定してみる

オプション要件10: AWS にインスタンスを作って環境構築する

- AWSに環境構築をしてデプロイしてみましょう
- ミドルウェアはNginx+Unicornを推奨構成とします

- EC2のインスタンス等はサーバ要件を参照してください