# 情報理工学実験　web_intelligence

02170037

金子泰之

提出日　2019 年 12 月 4 日

## 課題の注意点

課題１－2 と１－３は同一ファイルに含まれています。
課題 3－1 と 3－2 は sinatra で行ったため課題４－１は day3 に含まれています。
課題４－２と課題 5 は同一ファイルに含まれています。

## 1 日目の課題に関して

初めて API というものを使ってみて大変便利だと感じた。YOLP というものを使ったが住所から緯度経度の情報を手軽に取得出来たり、2 点間の距離を測ることが出来た。今までいろいろなホームページを利用していて、地図が表示されていることもあったがどうやって表示して処理を行っているのか知らなかったので知る良い機会だった。単に地図を表示するだけでなく、縮尺を変えるバーを作ったり、地図の中心を変更したりと使用目的によって変更できる点が多くそういったところでも工夫を行った。

## 2 日目の課題に関して

英語の検索は普通に URL に打ち込んで検索できるが日本語になると ASCII にエンコードする必要があり、エンコードを行って URL に入れる必要があった。また、存在しない本だったり、住所だったりした場合にエラーを出力する必要がありそういった処理も大変だったが実装した。実行した感じとして毎回問い合わせをしているので情報を取得するまで時間がかると感じた。

## ３日目の課題に関して

Sinatra を用いて作成した。やっていることは 1 日目と 2 日目の課題を合わせただけだがデータを ruby ファイルと erb ファイルで受け渡す際に情報量が多くなったり、渡すのが配列やハッシュだった時に難しいと感じた。基本的なことだが入力フォームは入力しやすく、結果は見やすいように心がけた。

## 4 日目の課題に関して

3 日目の課題を Sinatra で書いていたのであまり改変することなく終わった。課題 4－2 で不正な文字列を受け取ったときは入力を進ませないようにしたり、例外を見つけてそれを一つ一つ解決していきどのような条件でもプログラムがうまくいくように作るのが大変だ

った。

## 5 日目の課題に関して

今回最初のページでキーワードと基準点、そして読み込む本の件数を読み込んでいるが住所と読み込み件数を遷移後のページでも編集できるようにしようとした。住所に関しては地図が表示されているページで新しい住所を入力するとページが更新されて新たな住所からの距離順にソートしたものを表示することが出来た。一方で、読み込み件数の変更だが、結果は実装することが出来なかった。考えていた仕組みとしては ruby ファイルですべてのデータを読み込んでおき、javascript で表示する件数を制御するという仕組みにしたかったが最後の段階でページから得た情報を他の式に代入することが出来なかったため止む無く最初の段階で表示件数も指定するプログラムにした。いろいろ変更してみたが結局実装できなかったので残念である。実行したかった処理を乗せたプログラム「本来」とやむなく変更したプログラム「妥協」をレポートの最期に添付する。

それ以外の工夫点や全体にかかわる工夫点は下の「工夫した点」という項目に記述する

## 工夫した点

まず地図の表示に関して、スクロースやダブルクリックで地図を動かすことが出来、さらに Ctrl＋スクロールもしくは左のバーを用いて地図の拡大と縮小を自在に行えるようにしたため使いやすさが向上した。次に読み込む本の数を変更出来るようにした。最大でも 200 件までしか読み込めないため、初めに 200 件すべてを読み込んでおき特定のページで指定された件数だけ表示できるようにした。この時、図書の名前＝＞図書の ID にハッシュ化しているため図書の名前が被った場合図書の ID が上書きされてしまう。これを回避するために同じ図書が出てきたときは名前を図書（1）のように変えてハッシュ化を行った。また基本的なこととして、URL エラー、住所のエラー、蔵書図書のエラー、空入力などのエラーが起こった時には実行させずに再度適切な情報を入れなおさせるようにエラー処理を行った。稀に本は存在しても図書館に蔵書が無い本もあるのでその本の蔵書場所を検索しようとしたときは蔵書が無いことを伝える警告文を出してそれ以上処理を進めないようにした。さらに、プログラムを実行していると図書館の住所を受け取る処理で通信時間が長い箇所があり、複数のデータを送受信する際は大きな問題になることが分かった。これを解消するために図書館の名前、住所、緯度、経度を格納するデータベースを用意しておき、一度出現した図書館の場合はデータベースからデータを参照することで処理時間の短縮化を図った。結果は、72 件の図書館の住所などを読み込んで地図を描画するまでにデータベース導入前は 94.04 秒かかり、データベース導入後は 1.37 秒まで短縮することが出来たのでデータベースを導入することで効率化に成功したといえる。

## マッシュアップが持つ利点と欠点・感想

マッシュアップを使う利点としてその手軽さが考えられる。自分の必要な機能すべてを構築する必要性がないので、アイデアがあればすぐに始められる。そのため開発時間やコストといった面を抑えることが出来るので生産性が向上し、さらに必要な機能の追加や不要になった機能の削除なども簡単に行うことが出来るので更なる効率化が見込める。一方で、マッシュアップの欠点として多くを組み合わせて使うことによる弊害も起こる。例えば、データのアクセスに時間がかかるため全体としての処理時間が遅くなる可能性があり、もし多数の API を用いている場合はその影響が無視出来なくなる。さらに API を提供している団体が提供を停止、もしくは変更することによって今まで使えていた機能が急に使用できなくなる可能性もある。加えて多くの場合、使用にはトークンが必要でこのトークンの期限切れによってAPIの使用制限がかかることや、そもそも利用自体が有料であることもある。

今回初めてマッシュアップを行ってみて便利さが良く分かった。地図などを挿入することで見栄えもよくなり、利用者の使いやすさも向上した。一方で通信にかかる時間がとても長く、取得データ数が少ないときはあまり気にならない遅延でもデータ数が多くなるとそれが酷くなり、利用に支障を与え API の欠点を如実に感じた。使いやすさの面では、一般に公開されていて使い方も出ているので少し調べればいろいろな情報が出てくるため使いやすいと感じた。