

# 情報システム設計論 2 レポート

Shen yizhu

吉川馬研究室

6930-27-3766

## 全体説明 :

私が作ったのは、オンライン英語単語管理システム。このシステムはユーザーに登録機能、単語検索機能、ノート作成機能、そして単語をノートに保存する機能を提供する。

### 1. モデル :

このシステムは以下のいくつのモデルで構成される :

**User** : 登録アカウントのデータを保存するモデル

--name : アカウントのユーザー名

--password : アカウントのパスワード

--password\_confirmation : 確認用のパスワード

**Note** : 生成されたノートのデータを保存するモデル

--note\_id : ノートの ID

--name : ノートの名前

--comment : ノートに関するメモ

--mark : ノートの重要度を表す数値

**Word** : 保存された単語の基本データを保存するモデル。一つの Note が複数の Word を持つことができる。

--title : 単語そのものを表す文字列

--ipa : 単語の発音

--pron : 単語のオーディオトラックの URL

**Word\_def**：保存された単語の定義を保存するモデル。一つの Word が複数の Word\_def を持つことができる

--title\_id：この定義を持つ Word の ID

--jp: 単語の品詞

--en: 単語の解釈

--eg: 単語の用例

## 2. コントローラー：

**Application\_controller.rb**：全体に適用するアクションを記述するコントローラーで。

--authorize :ページが呼び出される前に、登録状態をチェックするアクションで、不登録と判断したら登録ページに移動する。

**Sessions\_controller.rb**：主にログインページに関する機能を実現するコントローラー。

--new：ログインページを表示する

--create：入力されたユーザー名とパスワードを認証し、通過すればホームページに移動し、サーバーに登録情報を記録する。通過しなければエラーメッセージを表示する。

--destroy：ログアウトのとき用いられるアクションで、ユーザー登録情報をサーバーからか消除する

**Notes\_controller.rb**：主にノート管理機能を実現するコントローラー

--welcome：ホームページを表示する

--index：全てのノート情報を表示する

--show：あるノートを詳細表示する。

--create・new：ノートを作成するとき用いられるアクションで、作成失敗したらエラーメッセージを提出する。

--edit・update：ノート情報を修正するとき用いられるアクションで、修正失敗したらエラーメッセージを提出する。

--destroy : ノートを消すアクション。

**Words\_controller.rb** : 主に単語検索、管理機能を実現するコントローラー

--index : 全ての単語情報を表示する

--show : ある単語を詳細表示する。

--create・new : 単語を作成するとき用いられるアクションで、作成失敗したらエラーメッセージを提出する。

--destroy : 単語を消すアクション。

--search\_suggest : 外部 API を用いて、単語を入力するときユーザーに相關単語を提示するアクションで、提示結果が AJAX の方式で表示する

--search : 外部 API を用いて、検索単語の品詞や発音、定義などを表示する。

**Word\_defs\_controller.rb** : scaffold で自動的に生成されるもので、Word\_def モデルを制御するコントローラーであるが、今回はこれを利用しない。Word\_def モデルに対する制御を全て Words\_controller で実現する。

3. ビュー :

**Layouts** : ページのベースと成るテンプレート

--application.html.erb : 全てのページに適用するレイアウトで、デフォルトのまま。

--note.html.erb : notes のパスにおけるページに適用するレイアウトで、外部の CSS と JS、またはパーシャルがここで読み込まれる。

--session.html.erb : sessions のバスにおけるページに適用するレイアウトで、外部の CSS と JS、またはパーシャルがここで読み込まれる。

**Partials** : ページに用いられる部分テンプレート

--bgswitch.html.erb : 登録ページに用いられるスキン変更ダッシュ

ユボードであり、あらかじめ用意されたスキンのサムネイルを表示し、ユーザーに選択させてもらう。JQueryで変更機能を実現する。

--bgsnav.html.erb : 登録ページ以外のページに用いられる変更ダッシュボードであり、スキンのサムネイルを表示しない。JQueryで変更機能を実現する。

--bgstransfer.html.erb : ページが読み込まれるとき、選択させたスキンのCSSをレンダリングする。

--headbar.html : 登録ページ以外のページに用いられるもので、システムの名前と検索入力スペースを提供する。

--sidebar.html : 登録ページ以外のページに用いられるもので、ホームページ、ノート表示ページ、単語表示ページ、Aboutページ、ログアウトへのリンクを提供する。

--form.html.erb : ノートを詳細表示するとき用いられるフォームテンプレート。

Sessions : ログイン機能に関するビュー

--new.html.erb : 登録ページ。登録用フォームを提示する

Notes : 主にノート管理に関するビュー

--welcome.html.erb : ホームページ。Fullcalendar.jsを用いて、単語が保存された日付をカレンダーの形で提示する。

--index.html.erb : 全てのノートを表示するページ。

--show.html.erb : あるノートの情報とこのノートに保存される単語を表示するページ。

--new.html.erb&&edit.html.erb : ノート作成・編集ページで、同じフォームパーシャルを用いる。

--about.html.erb : aboutページ。

Words : 単語管理・検索機能に関するビュー

--form.html.erb : 単語を詳細表示するとき用いられるパーシャル。

--index.html.erb : 全ての単語を表示するページ。

--show.html.erb : ある単語の基本情報と持つ定義を表示するペー

ジ。form.html.erb を用いる

-- search.html.erb : 単語の検索結果を表示するページ。  
form.html.erb を用いる。このページで単語を保存できる。

公開先の URL:

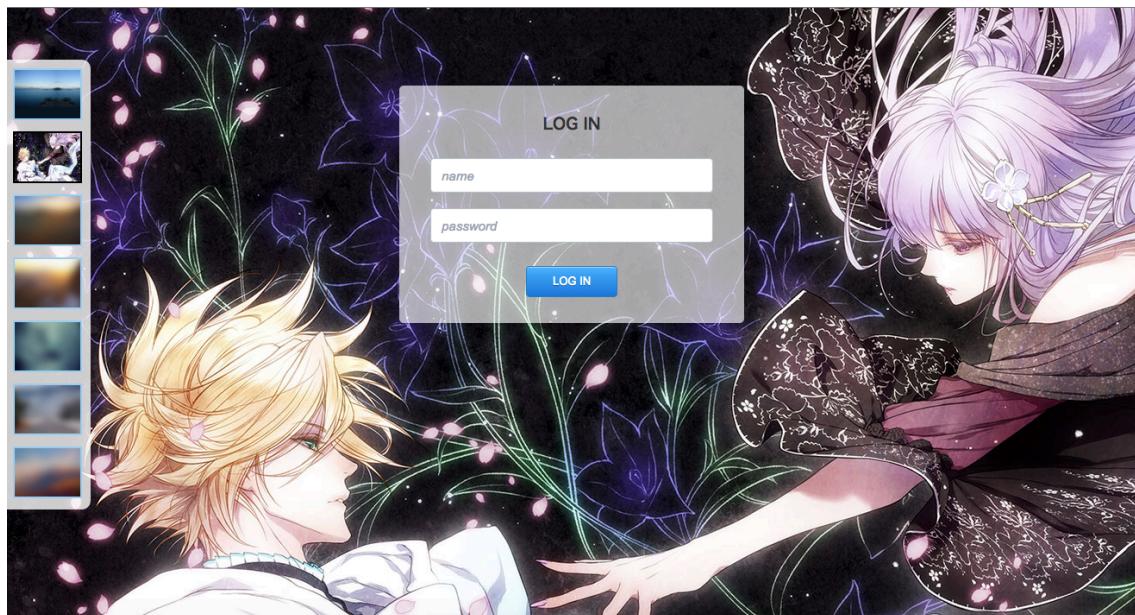
[shen-yi-zhu.herokuapp.com](http://shen-yi-zhu.herokuapp.com)

注意 :

1. 初使用の時ローディング時間が長い
2. スキン変更機能を利用できない場合、refresh してください

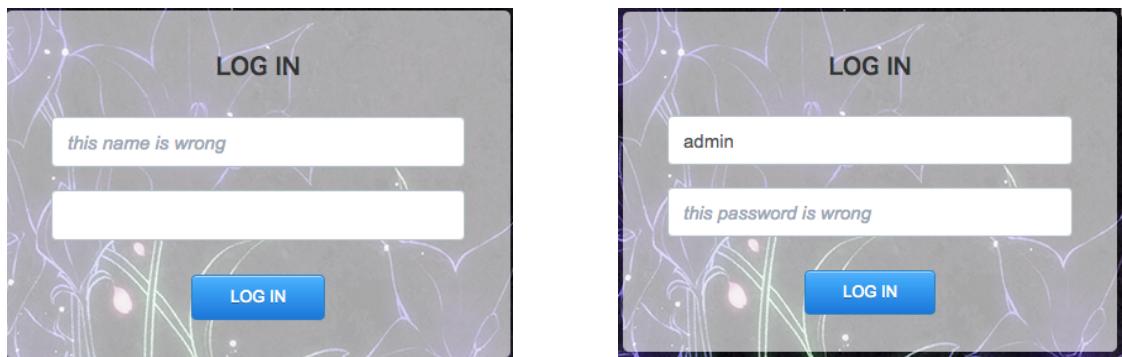
利用シナリオ :

1. ログイン画面 :



左の bgswitch でスキンを変更できる。

ID: admin Pw:123456 以外のユーザー名やパスワードを入力すると、エラーメッセージが提出される。



## 2。ホームページ：

MY WORDS NOTE

please input the word

[Home](#)

[My Notes](#)

[My Words](#)

[About](#)

[Log out](#)

July 2015

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1

右にいる bgsnav でスキンを変更できる。

MY WORDS NOTE

please input the word

[Home](#)

[My Notes](#)

[My Words](#)

[About](#)

[Log out](#)

July 2015

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1

skin 1  
skin 2  
skin 3  
skin 4  
skin 5  
skin 6  
skin 7

カレンダーで単語が保存された日付を表示する。1日中六つ以上を超えると、略の形で表示する。

	Fri	Sat
30		1
	apple	informal
		cat
		dog
		mouse
		white
		+2 more
7		8
		9

month

Fri	Sat
	Saturday, May 2
apple	informal cat dog mouse white cargo bus

左にいる sidebar でノートページ、単語ページ、about ページをリンクでき、ログアウトできる。

### 3. ノートページ：

	NAME	COMMENT	MARK	DATE	ACTION
<input type="checkbox"/>	myNote1	this is my first note	2	2015-04-19 11:46:59 UTC	Show Edit Destroy
<input type="checkbox"/>	myNote2	this my second note	3	2015-04-19 12:24:34 UTC	Show Edit Destroy

右にいるアクションクリックするとこのノートを詳細表示、編集、削除ページにリンクする。Add Note ボタンを押すと新規作成ページにリンクする

### 4. ノート詳細表示ページ：

WORD	POS	MEANING	VOICE
car	1. noun	1. a vehicle (as a railroad coach or an automobile) that moves on wheels	<ul style="list-style-type: none"><li>Show</li><li>Destroy</li></ul>
god	1. noun	1. the supreme or ultimate reality	<ul style="list-style-type: none"><li>Show</li><li>Destroy</li></ul>
test	1. noun 2. verb 3. noun	1. a critical examination, observation, or evaluation 2. to put to test or proof 3. a firm or rigid outer covering (as a shell) of many invertebrates	<ul style="list-style-type: none"><li>Show</li><li>Destroy</li></ul>

詳細表示ページで、単語に対応するアクションをクリックすると、単語を詳細表示、消除できる。

## 5. ノート新規作成ページ：

MY WORDS NOTE

please input the word

Edit this note

Name: myNote1

Mark: 1

Comment: this is my first note

Submit OR CANCEL

## 6. ノート編集ページ：

MY WORDS NOTE

please input the word

New note

Name:

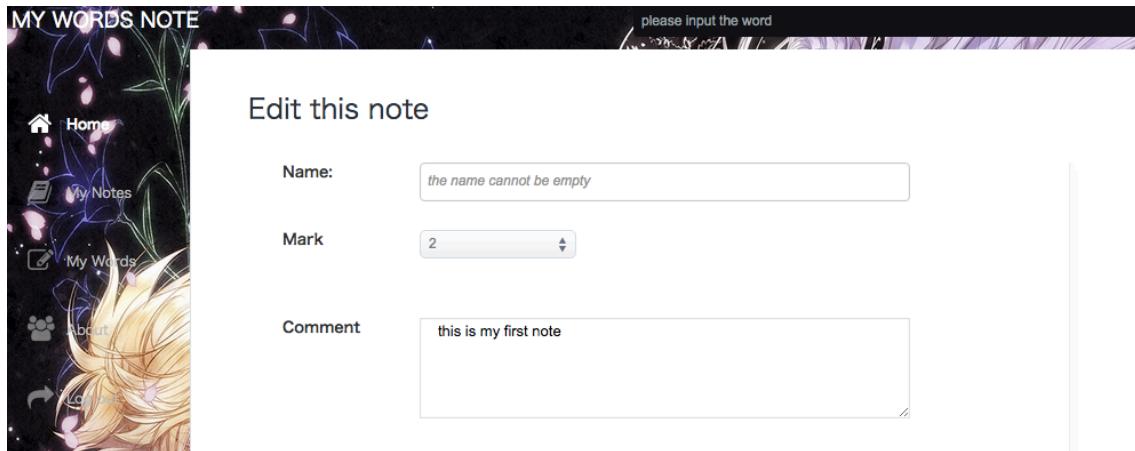
Mark: 1

Comment:

Submit OR CANCEL

新規作成ページと編集ページで、Submit ボタンを押すと、入力されたデータを送信できる。また、Cancel ボタンを押すと、ページを更新しないまま、入力が初期状態戻る。(新規作成の場合は、空に戻る。編集の場合は元の内容に戻る)。

name を空いたまま送信すると、エラーメッセージが提出される。



## 6. 単語ページ：

The screenshot shows the 'Words' page. The sidebar on the left is identical to the previous one. The main content area has a title 'Words'. Below it is a table with three entries:

Title	NoteID	Jp	En	Action
car	1	1. noun	1. a vehicle (as a railroad coach or an automobile) that moves on wheels	<a href="#">Show</a> <a href="#">Destroy</a>
god	1	1. noun	1. the supreme or ultimate reality	<a href="#">Show</a> <a href="#">Destroy</a>
test	1	1. noun 2. verb 3. noun	1. a critical examination, observation, or evaluation 2. to put to test or proof 3. a firm or rigid outer covering (as a shell) of many invertebrates	<a href="#">Show</a> <a href="#">Destroy</a>

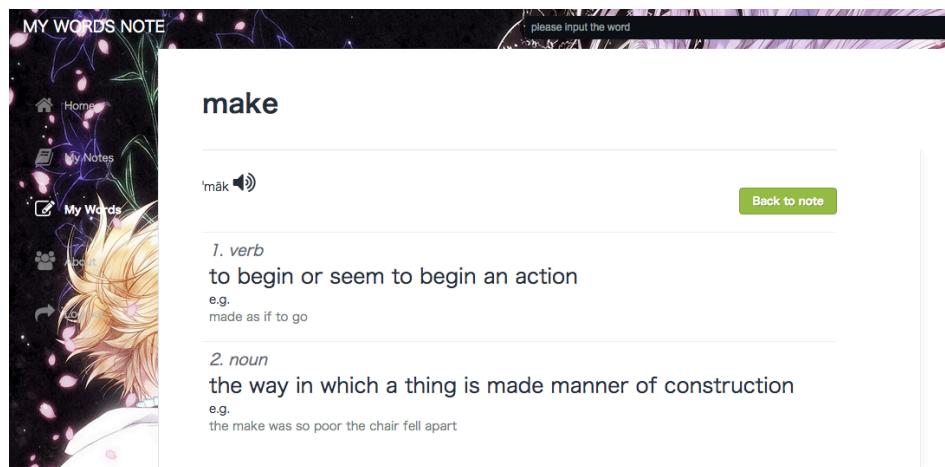
At the bottom, there is a message 'Showing 1 to 3 of 3 entries' and a navigation bar with buttons for First, Previous, Next, and Last.

jquery.dataTables.js を用いて、保存された全ての単語を提示する。単語検索、一ページ中の単語数の設定ができる。

単語に対応するアクションをクリックすると、単語を詳細表示、消除できる。

右下で総単語の数を確認できる。

## 7. 単語詳細ページ：



喇叭の図を押すとオーディオトラックを放送できる。

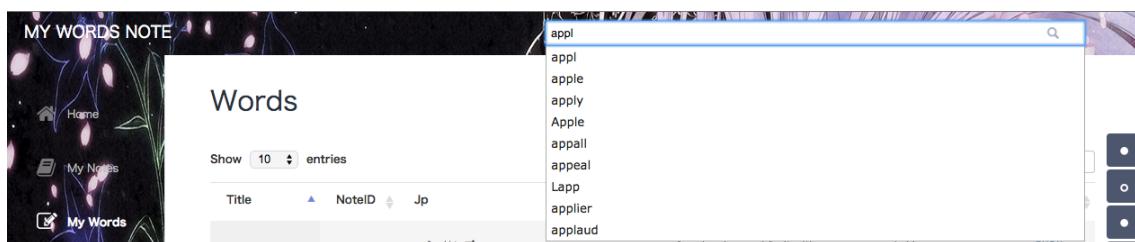
Back to note ボタンを押すと、この単語を保有するノートの詳細表示ページに移動できる。

上にいる headbar の入力スペースで単語を検索できる方法は二つ。

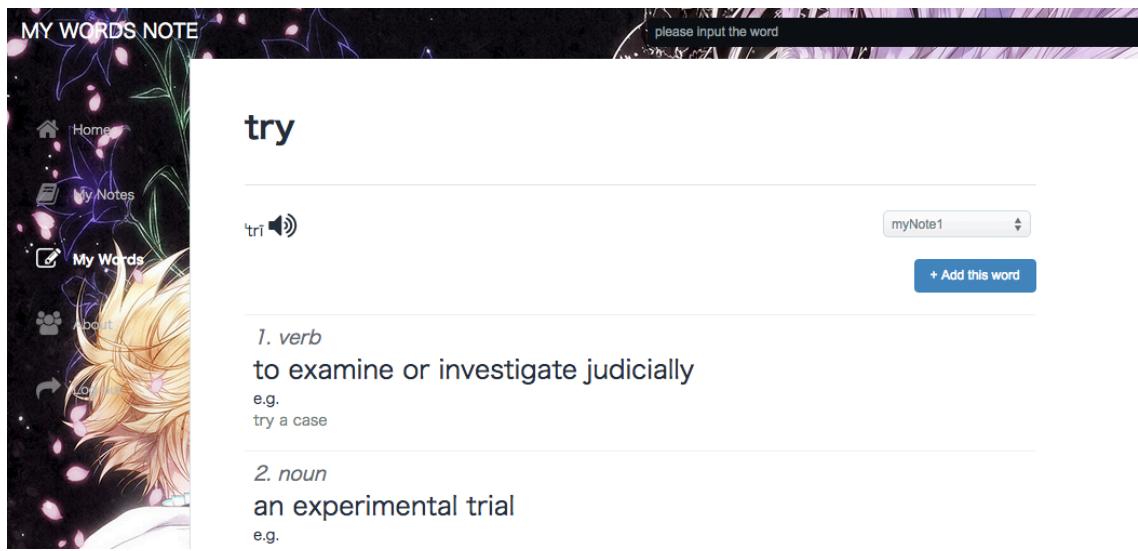
方法 1：単語を入力し、Enter ボタンを押す。

方法 2：単語を入力し、AJAX で提示された相關単語を選択する。

## 8. 相関単語提示：



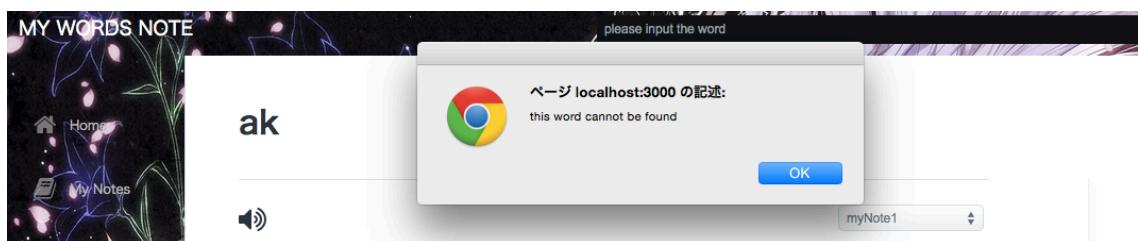
## 9. 検索結果ページ



右のボタンで単語をノートに保存することができる。

もし、検索された単語はすでにノートに保存された場合、直接に保存された単語の詳細表示ページにリンクする。

また、入力された単語が見つからなかった場合、エラーメッセージが提示された後、単語ページに移動する。



10. about ページ:



アピールポイントとそれに対応しているソースコード：

アピールポイントは全てのオプション機能を実現したこと

1. CSS や外部ライブラリを利用した UI 設計。

今回は主に **Bootstrap** という CSS フレームワーク、**JQuery**、**JQuery** に基づく **dataTables** というテーブルプラグイン、**fullcalendar** というカレンダープラグインを利用した。

フォームやテーブルテンプレートなど、**Bootstrap** は主にシステム全体的なインターフェース設計に用いる。そして **JQuery** は主にスキン変更など、インターフェクション設計に用いられる。

ノートを表示する時と違って、単語を表示する時 **dataTable** が使われる。**dataTable** に付加された検索機能や、ランキング機能で、より便利な単語管理を提供できる。

コード：

```

<script type="text/javascript">
$(document).ready(function() {
    $('#example').DataTable({
        "sPaginationType": "full_numbers"
    });
});
</script>

```

そして、**fullcalendar** をベースとして、カレンダーの形でユーザー

に単語の保存の日付を可視化することで、ユーザーに単語の全体状況を把握させられる。

コード：

```
<script>

$(document).ready(function() {

    var date = new Date();
    var d = date.getDate();
    var m = date.getMonth();
    var y = date.getFullYear();
    $('#calendar').fullCalendar({
        header: {
            left: 'prev,next today',
            center: 'title',
            right: 'month'
        },
        selectable: true,
        selectHelper: true,
        editable: true,
        eventLimit: true,
        events: [
            <% @words.each do |word| %>
            {
                title: "<%= raw(word.title) %>",
                start: new Date("<%= raw(word.created_at) %>"),
                allDay: false
            },
            <%end%>
        ],
    });
});

</script>
```

## 2. 外部 Web サービス

今回は二つの API を利用した。一つは入力された単語に対して、相關単語を提示する API--「Spellcheck」。単語提示機能があれば、例え検索したい単語を正しく覚えなくても、提示された単語リストから目標単語を得られる。

コード：

```

def search_suggest

response = Unirest.get "https://montanaflynn-spellcheck.p.mashape.com/check/?text="+ params[:search],
headers:{
  "X-Mashape-Key" => "u6IwQ3lgE9mshD9JiaffujdP2ltzp1TKoiajsnPvKMG7XR8hAVA",
  "Accept" => "application/json"
}

temp = response.body["corrections"][params[:search]]
if temp == nil
  temp = Array.new
end
temp.insert(0,params[:search])
render json:temp.to_json

end

```

もう一つは Merriam-Webster が提供される単語の定義検索 API。この API は入力された単語に対して、品詞や、発音、または辞書で書かれている定義を XML で返信するもので、そのシステムの使い道のコアをサポートする。

コード：

```

def search

@note = Note.all

@word = Word.find_by_title(params[:word])
if @word != nil
  redirect_to @word
else
  @word = Word.new
end

baseUrl= "http://www.dictionaryapi.com/api/v1/references/sd4/xml/" +params[:word]+ "?key=7b6540c6-117a-4264-8c39-070748265b39";
uri = URI.parse(baseUrl);
userAgent = Net::HTTP.new(uri.host, uri.port);
req = Net::HTTP::Get.new(uri.request_uri);
response = userAgent.request(req);
doc = REXML::Document.new(response.body)

```

### 3. Ajax

よりよいユーザ－体験を目指すため、「Spellcheck」で得た相關単語は、Ajax の方式で表示する。Ajax を元に、ページ更新を必要なく効率的な検索機能を提供できる。

コード：

```

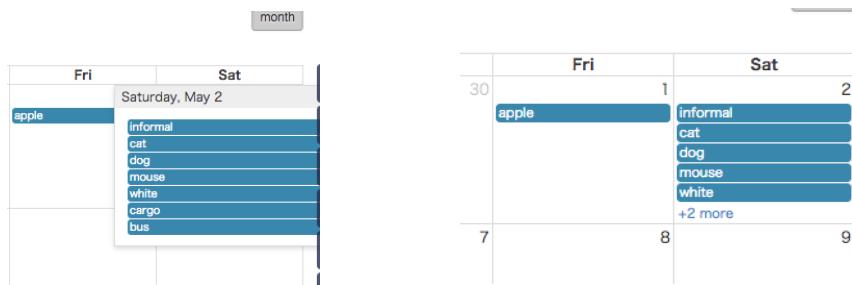
<script>
$(function() {
    $("#tags").autocomplete ({
        source: function(request,response){
            var term = request.term
            $.ajax({
                url: '/words/search_suggest',
                data: {search:term},
                type: "POST",
                datatype: "json",
                success: function(data) {
                    response(data);
                }
            })
        },
        select: function( event, ui ) {
            location.href = '/words/search?word=' + ui.item.value
        }
    });
    $('#tags').bind('keypress',function(event){
        if(event.keyCode == "13")
        {
            location.href = '/words/search?word='+$('#tags').val();
        }
    });
})
</script>

```

## レビューで貰ったコメントとその対応：

**Q** :一日中数が多い単語を保存する場合、カレンダーはどう表示するのか？

**A** :説明不足ですみません。六つ以上を超えると、略な形で表示する。クリックすると、詳細が出られる。



**Q** :英語の単語から日本語の定義を検索できるが、逆はダメですか？

**A** (レビュー時) このシステムは基本 API 次第のもので、適切な API がいないと、実現しにくいです。すみません。(今)：レビュー時は英語から日本語の定義を得るシステムを紹介したが、この後元の API が急に使

えなくなったので(ライセンスの問題)、英語単語から英語定義を得る API に変更した。不便をかけてすみません。

## 感想 :

今回は初めて `Rails` で Web アプリケーションを作った。前に `ThinkPHP` で Web アプリケーションを作ったことがあるので、`Rail` は `ThinkPHP` より便利であることを認識した。

例えば `Rail` は `Gem` というライブラリパッケージの仕組みがあり、ライブラリの管理が効率的に行われる。また、`scaffold` というツールで MVC 構造を簡単に生成する。そして PHP と違って、`Rails` のコントローラーの場合は、変数ではなく、ビューにモデルのオブジェクトを渡すことができる。よって、ビューからモデルメソッドを直接に呼び出すことで、同じロジックが少ないコードで実現できる。

そして今回は色々な外部ライブラリを利用するなどを挑戦した。自分から見るといいインターフェース設計を作成したが、各機能や設計を実現するライブラリを用いる結果として、システム全体にリンクされるライブラリ間の依存関係が混雑になった。それらのライブラリを整理したいと思うが、なかなかその異なるバージョン間の差を判明できない。勝手に消除すると問題を起こる可能性があるので、全てのライブラリをそのまま残した。今後の学習として、ライブラリを使うだけでなく、ライブラリの内容を理解することが大事であることを意識した。