ネットワークプログラミング I
—Web アプリケーションの作成—

学籍番号：16426
4 年 電子情報工学科 23 番
福澤 大地

提出日：2020 年 2 月 17 日

1　目的

Ruby で Web アプリケーションを作成することで、講義で身につけたサーバーサイドプログラミング、フロントエンドプログラミング、データベースなどの総合的な技術を強化する。

また、外部者からの攻撃などを想定し、それに対する対策を施すことで、セキュリティに関する知識を身に付ける。

2　作成したアプリケーション

イラスト共有を主な目的とした、掲示板サイトを作成した。テキストに加え、Web アプリケーション上で描いたイラストを投稿することができる。さらに、他人が描いたイラストを元にして絵を描き加えられる機能も実装した。

また、アカウント作成とログイン機能を実装し、自身が行った投稿の管理ができるような仕様にした。

3　開発環境

アプリケーションを開発するにあたって、仮想化ソフトフェアである VurtualBox を用いて仮想環境を構築した。ホスト OS の環境を表 1, 仮想環境を表 2 に示す。

表 1　ホスト OS の環境

CPU	Intel Core i5-7400 @ 3.0GHz
メモリ	8GB
OS	Microsoft Windows 10 Home
システム	64bit
Web ブラウザ	Google Chrome 77.0.3865.120

表 2　仮想環境

仮想化ソフトフェア	Oracle VirtualBox 6.0.12
割り当てメモリ	2GB
OS	CentOS 7.6
システム	64bit
開発言語	Ruby 2.6.2
データベース	SQLite3 3.7.17

今回は仮想環境上で Web アプリケーションを実行し、ホスト OS の Web ブラウザからアクセスする。つまり、ホスト OS と仮想環境を別々の 2 つのコンピュータと捉えて通信を行う。URL には次のように入力する。

```
http://localhost:9998/
```

`localhost` というのはループバックアドレスであり、自分自身にパケットを返す。末尾の`:9998` で 9998 番ポートでパケットを受け取ることを指定している。

4　Gem パッケージのインストール

Ruby でアプリケーションを作るにあたって、まずは必要な Gem パッケージをインストールしておく必要がある。Gem パッケージとは Ruby で使われるライブラリのことである。Gem パッケージをインストールするために、アプリケーションフォルダにパッケージを整えてくれるソフトウェア (Bundler) を使用する。アプリケーションフォルダに移動し、次のように入力して実行する。

```
bundle init
```

実行すると自動的に作成される Gemfile というファイルを、リスト 1 のように書き換える。Gemfile とは、Bundler の設定ファイルである。

リスト 1　Gemfile

```
1 source "https://rubygems.org"
2
3 gem "sinatra"
4 gem "sqlite3"
5 gem "activerecord", "< 6.0"
```

Gemfile で指定したパッケージの概要について次に示す。

- sinatra ... Web アプリケーションフレームワークである Sinatra を使うためのライブラリ
- sqlite3 ... データベース管理システムである SQLite3 を使うためのライブラリ
- activerecord ... Ruby でデータベースを扱うためのライブラリ

最後に次のように入力して。必要なパッケージをインストールする。

```
bundle install --path vendor/bundle
```

vendor というフォルダが作成され、このフォルダの中にパッケージがインストールされる。実際にインストールされたパッケージは、Gemfile と同じディレクトリの Gemfile.lock に記述されている。

5　データベースの設計

今回作成した掲示板の Web アプリケーションでは、書き込みを格納するためにデータベースを使用する。データベース管理システムは SQLite3 である。テーブルを作成するにはプロンプトから一行ずつコマンドを打ち込んでつくるという方法もあるが、テーブルに変更を加えたり、初期化したりするときにいちいち最初から打ち込むのは面倒である。そのため、リスト 2 のような SQL 文を記述したファイルを作成しておくと便利である。

リスト 2　bbs.sql

```
1 CREATE TABLE accounts (
2     userid VARCHAR(32) PRIMARY KEY,
3     salt   CHAR(32),
4     hashed CHAR(32),
5     name   VARCHAR(64)
6 );
7
8 CREATE TABLE posts (
```

```
    number INTEGER PRIMARY KEY,
    exist  INTEGER,
    kind   INTEGER,
    time   CHAR(24),
    userid VARCHAR(64),
    text   VARCHAR(1024),
    origin INTEGER
);
```

これを bbs.sql として保存し、次のように打ち込むと bbs.db が作成される。

```
sqlite3 bbs.db < bbs.sq3
```

投稿を格納するテーブル posts の設定を表 3、アカウント情報を格納するテーブル accounts の設定を表 4 に示す。

表 3　posts テーブルの概要

フィールド名	型	内容
number	INTEGER	投稿番号
exist	INTEGER	投稿が存在してるか (削除されていないか)
kind	INTEGER	投稿の種類 (0:テキスト, 1:イラスト)
time	CHAR(24)	投稿した時刻
userid	VARCHAR(64)	投稿者のユーザー ID
text	VARCHAR(1024)	投稿内容
origin	INTEGER	元にしたイラストの投稿番号

表 4　accounts テーブルの概要

フィールド名	型	内容
userid	INTEGER	ユーザー ID
salt	CHAR(32)	ハッシュ化の際に用いるソルト
hashed	CHAR(32)	認証用のハッシュ値
name	VARCHAR(64)	アカウント名

6　YAML について

YAML (YAML Ain't a Markup Language) は XML のように構造化されたデータを、XML よりも人間に読み書きしやすい形にしたものである。アプリケーションで bbs.db を扱うために、ソースコード 3 ように database.yml を作成する。プログラム内でこのファイルを読み込むと、指定したデータベースを操作できるようになる。

リスト 3　database.yml

```
development:
    adapter: sqlite3
    database: bbs.db
```

7　ActiveRecord について

ActiveRecord はデータベースのレコードをオブジェクト指向言語のオブジェクトに対応させるライブラリである。そうすることでオブジェクト指向言語でオブジェクトを操作すると、内部的にデータベース管理システムのコマンドを発行し、適切にデータベースを操作して結果を返してくれる。SQLite3 や MySQL などのデータベース管理システムの違いを ActiveRecord が吸収することで、様々なデータベースをオブジェクト指向言語のオブジェクトへ対応させている。

8　Web アプリケーションの構造

今回作成したアプリケーションでは、処理部分と表示部分を分離させている。処理本体は Ruby で記述するが、表示は HTML コードの中に Ruby のコードを埋め込むことができる Embedded Ruby で記述する。Embedded Ruby は、表 5 に示すルールで Ruby のコードを埋め込むことができる。

表 5　erb のルール

記述	意味
`<%= code %>`	囲まれた部分を実行して結果を埋め込む。
`<% code %>`	囲まれた部分を実行するが、結果は埋め込まない。
`<%# comment %>`	囲まれた部分をコメントアウトする。

見た目を表す Embedded Ruby のファイルは、views ディレクトリの中に作成する。アプリケーションフォルダのディレクトリ構造を図 1 に示す。

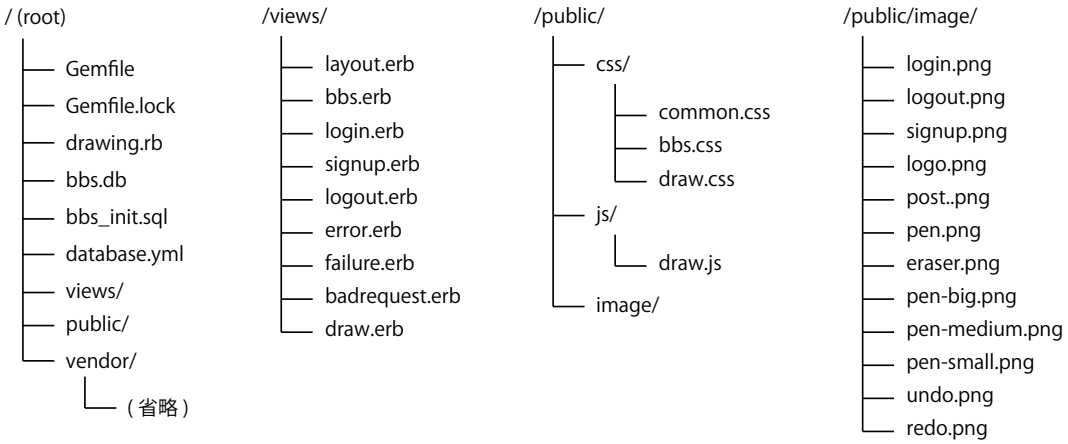

図 1　ディレクトリ構造

9　プログラムリスト

図 1 で示した各プログラムのソースコードを、リスト 4-17 に示す。

リスト 4　drawing.rb

```
1 require 'cgi'
2 require 'sinatra'
```

```
3  require 'digest/md5'
4  require 'active_record'
5
6  # 投稿種類
7  TYPE_TEXT = 0
8  TYPE_DRAW = 1
9
10 # 各種設定
11 NAME_MAX = 32   # 名前の最大文字数
12 USERID_MAX = 32 # ユーザーIDの最大文字数
13 PASS_MAX = 32   # パスワードの最大文字数
14 PAGE_MAX = 10   # 1ページに表示できる最大件数
15 TEXT_MAX = 512  # 書き込みの最大文字数
16
17 set :environment, :production
18 set :sessions,
19     expire_after: 7200,
20     secret: 'naganokosen3818850'
21
22 ActiveRecord::Base.configurations = YAML.load_file('database.yml')
23 ActiveRecord::Base.establish_connection :development
24
25 class Account < ActiveRecord::Base
26 end
27
28 class Post < ActiveRecord::Base
29 end
30
31 # 投稿の件数に対するページ数を返す
32 def page_num(len)
33     if len == 0
34         return 1
35     else
36         return ((len - 1) / PAGE_MAX).to_i + 1
37     end
38 end
39
40 # 1ページ目にリダイレクト
41 get '/' do
42     posts_len = Post.all.length
43     redirect "/#{page_num(posts_len)}"
44 end
45
46 # エラーページ
47 get '/error' do
48     @title = 'Error'
49     @css = 'error.css'
50     erb :error
51 end
52
53 # テキスト投稿
54 post '/new_text' do
55     if session[:login_flag]
56         post = Post.new
57
58         # 入力内容を取得
```

```
59          text = params[:text].slice(0, TEXT_MAX)
60          text = CGI.escapeHTML(text)
61          text = text.gsub(/(\r\n|\r|\n)/, '<br>')
62
63          # データベースへ書き込み
64          post.number = Post.all.length + 1
65          post.exist = 1
66          post.kind = TYPE_TEXT
67          post.time = Time.now.strftime('%Y/%m/%d(%a) %H:%M:%S')
68          post.userid = session[:login_userid]
69          post.text = text
70          post.origin = 0
71          post.save
72
73          redirect '/'
74      else
75          redirect '/badrequest'
76      end
77  end
78
79  # 投稿を削除
80  delete '/delete' do
81      post = Post.find(params[:number])
82
83      if session[:login_flag] && post.userid == session[:login_userid]
84          post.exist = 0
85          post.save
86          redirect '/'
87      else
88          redirect '/badrequest'
89      end
90  end
91
92  # お絵かき
93  get '/draw/:origin' do
94      @origin = params[:origin]
95
96      if Post.exists?(@origin)
97          if Post.find(@origin).kind != TYPE_DRAW
98              @origin = 0
99          end
100     else
101         @origin = 0
102     end
103
104     @text_max = TEXT_MAX
105     @title = 'イラスト投稿'
106     @css = 'draw.css'
107     erb :draw
108 end
109
110 # イラスト投稿
111 post '/new_draw' do
112     if session[:login_flag]
113         post = Post.new
114
```

```
115         # データベースへ書き込み
116         post.number = Post.all.length + 1
117         post.exist = 1
118         post.kind = TYPE_DRAW
119         post.time = Time.now.strftime('%Y/%m/%d(%a) %H:%M:%S')
120         post.userid = session[:login_userid]
121         post.text = params[:image]
122         post.origin = params[:origin].to_i
123         post.save
124 
125         redirect '/'
126     else
127         redirect '/badrequest'
128     end
129 end
130 
131 # 新規登録
132 get '/signup' do
133     @name_max = NAME_MAX
134     @userid_max = USERID_MAX
135     @pass_max = PASS_MAX
136     @title = '新規登録'
137     @css = 'signup.css'
138     erb :signup
139 end
140 
141 # ユーザー登録
142 post '/regist' do
143     name = params[:name]
144     userid = params[:userid]
145     password = params[:password]
146     re_password = params[:re_password]
147 
148     if !Account.exists?(userid) && password == re_password
149         r = Random.new
150         salt = Digest::MD5.hexdigest(r.bytes(20))
151         hashed = Digest::MD5.hexdigest(salt + password)
152 
153         a = Account.new
154         a.userid = userid
155         a.salt = salt
156         a.hashed = hashed
157         a.name = name
158         a.save
159 
160         session[:login_flag] = true
161         session[:login_userid] = userid
162         redirect '/'
163     else
164         redirect '/failure'
165     end
166 end
167 
168 # ログイン
169 get '/login' do
170     @userid_max = USERID_MAX
```

```
171     @pass_max = PASS_MAX
172     @title = 'ログイン'
173     @css = 'login.css'
174     erb :login
175 end
176 
177 # 認証
178 post '/auth' do
179     userid = params[:userid]
180     password = params[:password]
181 
182     if Account.exists?(userid)
183         a = Account.find(userid)
184         trial_hashed = Digest::MD5.hexdigest(a.salt + password)
185 
186         if a.hashed == trial_hashed
187             session[:login_flag] = true
188             session[:login_userid] = userid
189             redirect '/'
190         else
191             session[:login_flag] = false
192             redirect '/failure'
193         end
194     else
195         session[:login_flag] = false
196         redirect '/failure'
197     end
198 end
199 
200 # ログイン失敗
201 get '/failure' do
202     @title = 'Failed'
203     @css = 'failure.css'
204     erb :failure
205 end
206 
207 # ログアウト
208 get '/logout' do
209     session.clear
210     @title = 'ログアウト'
211     @css = 'logout.css'
212     erb :logout
213 end
214 
215 # バッドリクエスト
216 get '/badrequest' do
217     @title = 'Bad Request'
218     @css = 'badrequest.css'
219     erb :badrequest
220 end
221 
222 # ページ指定
223 get '/:page' do
224     # 数値が指定されているか
225     if params[:page] =~ /^[0-9]+$/
226         @page = params[:page].to_i
```

```
227     else
228         redirect '/error'
229     end
230
231     @last_page = page_num(Post.all.length)
232     @page_max = PAGE_MAX
233     @text_max = TEXT_MAX
234     @type_text = TYPE_TEXT
235     @type_draw = TYPE_DRAW
236
237     # 無効なページ数が指定されたらエラー
238     if @page <= 0 or @last_page < @page
239         redirect '/error'
240     else
241         @title = '掲示板'
242         @css = 'bbs.css'
243         erb :bbs
244     end
245 end
```

リスト 5　layout.erb

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8">
5    <title><%= @title %> - ABCDrawing</title>
6    <link rel="stylesheet" href="/css/common.css">
7    <link rel="stylesheet" href="/css/<%= @css %>">
8  </head>
9  <body>
10   <header id="head">
11     <a href="/"><img src="/image/logo.png" alt="ABCDrawing" width="300"></a><br>
12
13     <div class="buttons">
14       <% if session[:login_flag] %>
15         <a href="/logout"><img src="/image/logout.png" alt="ログアウト" width="147.5"></a>
16       <% else %>
17         <a href="/login"><img src="/image/login.png" alt="ログイン" width="147.5"></a>
18       <% end %>
19       <a href="/signup"><img src="/image/signup.png" alt="新規登録" width="147.5"></a>
20     </div>
21   </header>
22
23   <h2><%= @title %></h2>
24   <%= yield %>
25 </body>
26 </html>
```

リスト 6　bbs.erb

```
1    <!-- ページ操作 -->
2    <div class="page">
3      <a href="/1">&lt;&lt;</a>
4      <% if @page > 1 %>
5        <a href="/<%= @page - 1 %>">&lt;</a>
```

```
6       <% end %>
7
8       <%= @page %>/<%= @last_page %>
9
10      <% if @page < @last_page %>
11        <a href="/<%= @page + 1 %>">&gt;</a>
12      <% end %>
13      <a href="/<%= @last_page %>">&gt;&gt;</a>
14    </div>
15
16    <!-- 投稿一覧 -->
17    <div class="post">
18      <%# 表示件数を決定 %>
19      <% if Post.all.length == 0 %>
20        <% disp_len = 0 %>
21      <% elsif @page == @last_page %>
22        <% disp_len = (Post.all.length - 1) % @page_max + 1 %>
23      <% else %>
24        <% disp_len = @page_max %>
25      <% end %>
26
27      <%# 投稿を表示 %>
28      <% cur_posts = Post.all[(@page - 1) * @page_max, disp_len] %>
29      <% cur_posts.each do |post| %>
30        <% account = Account.find(post.userid) %>
31        <div id="post<%= post.id %>">
32          <hr>
33          <table cellspacing="10">
34            <tr align="left">
35              <td class="number"><%= post.number %></td>
36              <td class="name"><%= post.exist == 1 ? account.name : '削除されました' %></td>
37              <td class="time"><%= post.exist == 1 ? post.time : '削除されました' %></td>
38
39              <% if post.exist == 1 && post.userid == session[:login_userid] %>
40                <form method="post" action="/delete">
41                  <td><input type="submit" value="削除"></td>
42                  <input type = "hidden" name="number" value="<%= post.number %>">
43                  <input type = "hidden" name="_method" value="delete">
44                </form>
45              <% end %>
46            </tr>
47          </table>
48          <% if post.exist == 0 %>
49            削除されました<br>
50          <% else %>
51            <% if post.kind == @type_text %>
52              <div class="text">
53                <%= post.text %><br>
54              </div>
55            <% else %>
56              <img src="<%= post.text %>" width="300"><br>
57              <% if post.origin != 0 %>
58                <a href="/<%= ((post.origin - 1) / @page_max).to_i + 1%>#post<%= post.origin
    %>">>>><%= post.origin %></a>を元におえかきしました<br>
59              <% end %>
60              <a href="/draw/<%= post.number %>">この絵を元にしておえかき！</a><br>
```

```
61            <% end %>
62          <% end %>
63        </div>
64      <% end %>
65      <hr>
66    </div>
67
68    <!-- 入力フォーム -->
69    <% if session[:login_flag] %>
70      <div class="input">
71        <form method="post" action="/new_text">
72          <textarea required name="text" rows="8" cols="40" maxlength="<%= @text_max %>" \
73              placeholder="最大<%= @text_max %>文字"></textarea><br>
74          <input type="submit" value="書き込み">
75        </form>
76      </div>
77
78      <a href="/draw/0">イラストを書き込み</a>
79    <% end %>
```

リスト 7　login.erb

```
1   <form action="/auth" method="post">
2     ユーザーID <input required type="text" name="userid" size="40" maxlength="<%=
      @userid_max %>"><br>
3     パスワード <input required type="password" name="password" size="40" maxlength="<%=
      @pass_max %>"><br><br>
4     <input type="submit" value="ログイン">
5   </form>
```

リスト 8　signup.erb

```
1   <form action="/regist" method="post">
2     ユーザー名        <input required type="text" name="name" size="40" maxlength="<%=
      @name_max %>"><br>
3     ユーザーID        <input required type="text" name="userid" size="40" maxlength="<%=
      @userid_max %>"><br>
4     パスワード        <input required type="password" name="password" size="40" maxlength="
      <%= @pass_max %>"><br>
5     パスワード（確認）<input required type="password" name="re_password" size="40" maxlength
      ="<%= @pass_max %>"><br><br>
6     <input type="submit" value="新規登録">
7   </form>
```

リスト 9　logout.erb

```
1   <p>ログアウトしました。</p>
```

リスト 10　error.erb

```
1   <p>エラーが発生しました。無効なURLが指定された可能性があります。</p>
```

リスト 11　failure.erb

```
1   <p>認証に失敗しました。</p>
```

リスト 12　badrequest.erb

```
<p>不正なアクセスがありました。</p>
```

リスト 13　draw.erb

```
<script type="text/javascript" src="/js/draw.js"></script>
<script>window.onload = init;</script>

<div id="tool">
  <table id="tool-top">
    <tr>
      <td><img src="/image/eraser.png"></td>
      <td onclick="mode = Mode.eraser; weight = 20;"><img src="/image/pen-big.png"></td>
      <td onclick="mode = Mode.eraser; weight = 15;"><img src="/image/pen-medium.png"></td>
      <td onclick="mode = Mode.eraser; weight = 10;"><img src="/image/pen-small.png"></td>
    </tr>
    <tr>
      <td><img src="/image/pen.png"></td>
      <td onclick="mode = Mode.pen; weight = 15;"><img src="/image/pen-big.png"></td>
      <td onclick="mode = Mode.pen; weight = 10;"><img src="/image/pen-medium.png"></td>
      <td onclick="mode = Mode.pen; weight = 5;"><img src="/image/pen-small.png"></td>
    </tr>
    <tr>
      <td></td>
      <td><img onclick="undo()" src="/image/undo.png" ></td>
      <td><img onclick="redo()" src="/image/redo.png" ></td>
      <td></td>
    </tr>
  </table>
  <table id="pallet">
    <tr>
      <td><div id="color-DeepRed"     onclick="color = Color.deepRed"></div></td>
      <td><div id="color-Purple"      onclick="color = Color.purple"></div></td>
      <td><div id="color-Green"       onclick="color = Color.green"></div></td>
      <td><div id="color-VividGreen"  onclick="color = Color.vividGreen"></div></td>
    </tr>
    <tr>
      <td><div id="color-Red"         onclick="color = Color.red"></div></td>
      <td><div id="color-Blue"        onclick="color = Color.blue"></div></td>
      <td><div id="color-Yellow"      onclick="color = Color.yellow"></div></td>
      <td><div id="color-DarkGreen"   onclick="color = Color.darkGreen"></div></td>
    </tr>
    <tr>
      <td><div id="color-SalmonPink"  onclick="color = Color.salmonPink"></div></td>
      <td><div id="color-DeepBlue"    onclick="color = Color.deepBlue"></div></td>
      <td><div id="color-VividOrange" onclick="color = Color.vividOrange"></div></td>
      <td><div id="color-Gray"        onclick="color = Color.gray"></div></td>
    </tr>
    <tr>
      <td><div id="color-HotPink"     onclick="color = Color.hotPink"></div></td>
      <td><div id="color-LightBlue"   onclick="color = Color.lightBlue"></div></td>
      <td><div id="color-Orange"      onclick="color = Color.orange"></div></td>
      <td><div id="color-Black"       onclick="color = Color.black"></div></td>
    </tr>
```

```
50        <tr>
51          <td><div id="color-Pink"         onclick="color = Color.pink"></div></td>
52          <td><div id="color-VividBlue"    onclick="color = Color.vividBlue"></div></td>
53          <td><div id="color-Beige"        onclick="color = Color.beige"></div></td>
54          <td><div id="color-White"        onclick="color = Color.white"></div></td>
55        </tr>
56      </table>
57    </div>
58    <div id="canvas-box">
59      <img id="imageCanvas" src="<%= @origin == 0 ? '/image/white.png' : Post.find(@origin).
        text %>" width="466" height="466">
60      <canvas id="canvas" width="466" height="466"></canvas>
61    </div>
62
63    <a href="#" onclick="post();"><img id="post" src="/image/post.png" alt="投稿" width="200"
        ></a>
64
65    <form id="new_draw" method="post" action="/new_draw">
66      <input type="hidden" name="origin" value="<%= @origin %>">
67      <input type="hidden" name="image">
68    </form>
```

リスト 14　common.css

```
1  @charset "UTF-8";
2
3  html {
4      color: black;
5      background: #F0F0F0;
6  }
7
8  h2 {
9      background: #FFFFFF;
10     font-size: 200%;
11     border-bottom: double 5px #FFC778;
12 }
```

リスト 15　bbs.css

```
1  @charset "UTF-8";
2
3  .page {
4      font-size: 150%;
5  }
6
7  .number {
8      font-size: 12pt;
9      font-weight: bold;
10 }
11
12 .name {
13     color: navy;
14     font-size: 12pt;
15     font-weight: bold;
16 }
17
```

```
.time {
    font-size: 12pt;
    font-weight: bold;
}

.text {
    font-size: 14pt;
}
```

リスト 16　draw.css

```
#top {
    display: flex;
    align-items: center;
}

#tool {
    width: 25vw;
}

#tool-top {
    margin: 0 auto;
}

#tool-top img {
    width: 7vh;
    height: 7vh;
}

#pallet {
    margin: 3vh auto;
    border: 2px solid #666666;
    border-radius: 10px 10px 10px 10px;

    box-shadow: 0 0 3px #cacaca;
}

#pallet td {
    width: 7vh;
    height: 5vh;

    background-color: #f2f2f2;
}

#pallet div {
    width: 3.5vh;
    height: 3.5vh;

    margin: 0 auto;
    border-radius: 50%;

    box-shadow: -1px 1px 0px black;
}

#color-DeepRed {
    background-color: #c1272d;
```

```
}

#color-Purple {
    background-color: #a000a0;
}

#color-Green {
    background-color: #00ff00;
}

#color-VividGreen {
    background-color: #7fe021;
}

#color-Red {
    background-color: #ff0000;
}

#color-Blue {
    background-color: #0000ff;
}

#color-Yellow {
    background-color: #ffff00;
}

#color-DarkGreen {
    background-color: #556b2f;
}

#color-SalmonPink {
    background-color: #ed1e79;
}

#color-DeepBlue {
    background-color: #0071b0;
}

#color-VividOrange {
    background-color: #fbb03b;
}

#color-Gray {
    background-color: #808080;
}

#color-HotPink {
    background-color: #ff69b4;
}

#color-LightBlue {
    background-color: #32c8ff;
}

#color-Orange {
    background-color: #f7931e;
```

```
}

#color-Black {
    background-color: #141414;
}

#color-Pink {
    background-color: #ffc0cb;
}

#color-VividBlue {
    background-color: #32c8ff;
}

#color-Beige {
    background-color: #c69c6d;
}

#color-White {
    background-color: #ffffff;
}

#canvas-box {
    position: absolute;
    width: 50vw;
    top: 0px;
    left: 0px;
}

#imageCanvas, #canvas {
    display: block;
    position: absolute;

    top: 0;
    right: 0;
    bottom: 0;
    left: 0;

    margin: auto;
    border: 2px solid black;
}

#imageCanvas {
    z-index: 1;
}

#canvas {
    z-index: 2;
}
```

リスト 17　draw.js

```
// モード
let Mode = {
    pen: 0, // ペン
    eraser: 1, // 消しゴム
```

```
};

// ペンの色
let Color = {
    deepRed:     [193, 39, 45],
    red:         [255, 0, 0],
    salmonPink:  [237, 30, 121],
    hotPink:     [255, 105, 180],
    pink:        [255, 192, 203],
    purple:      [160, 0, 160],
    blue:        [0, 0, 255],
    deepBlue:    [0, 113, 176],
    lightBlue:   [50, 200, 255],
    vividBlue:   [0, 255, 255],
    green:       [0, 255, 0],
    yellow:      [255, 255, 0],
    vividOrange: [251, 176, 59],
    orange:      [247, 147, 30],
    beige:       [198, 156, 109],
    vividGreen:  [127, 200, 33],
    darkGreen:   [85, 107, 47],
    gray:        [128, 128, 128],
    black:       [20, 20, 20],
    white:       [255, 255, 255],
};

// 履歴を表すクラス
class History {
    constructor(canvas) {
        this.top = 0;
        this.current = 0;
        this.log = [canvas];
        this.canvas = canvas;
    }

    // 現在の状態を記録
    push() {
        // ログ用のキャンバスを用意
        let logCanvas = document.createElement("canvas");
        logCanvas.width = this.canvas.width;
        logCanvas.height = this.canvas.height;

        // 現在の状態をコピー
        let logContext = logCanvas.getContext("2d");
        logContext.drawImage(this.canvas, 0, 0);

        // 更新
        this.current++;
        this.top = this.current;
        this.log[this.current] = logCanvas;
    }

    // 元に戻す
    undo() {
        if (this.current <= 0) {
            return null;
```

```
61          } else {
62              this.current--;
63              return this.log[this.current];
64          }
65      }
66
67      // やり直し
68      redo() {
69          if (this.current >= this.top) {
70              return null;
71          } else {
72              this.current++;
73              return this.log[this.current];
74          }
75      }
76  }
77
78  // キャンバス
79  let canvas;
80  let context;
81  let history;
82
83  // 前回の座標
84  let px, py;
85
86  // ペン
87  let mode = Mode.pen;
88  let color = Color.black;
89  let weight = 10;
90  let drawing = false;
91
92  // 初期化
93  function init() {
94      // キャンバスを取得
95      canvas = document.getElementById("canvas");
96      context = canvas.getContext("2d");
97      history = new History(canvas);
98
99      // イベントリスナーを登録
100     canvas.addEventListener("mousemove", onMove, false);
101     canvas.addEventListener("mousedown", onClick, false);
102     canvas.addEventListener("mouseup", drawEnd, false);
103     canvas.addEventListener("mouseover", drawEnd, false);
104 }
105
106 // マウスアップ
107 function drawEnd() {
108     if (drawing)
109         history.push();
110
111     px = null;
112     py = null;
113     drawing = false;
114 }
115
116 // クリック
```

```
117 function onClick(e) {
118     drawing = true;
119     e.preventDefault();
120     const rect = e.target.getBoundingClientRect();
121 
122     x = e.clientX - rect.left;
123     y = e.clientY - rect.top;
124     drawLine(x, y);
125 }
126 
127 // ドラッグ
128 function onMove(e) {
129     // マウスが押されている場合にのみ処理を実行
130     if (!drawing)
131         return;
132 
133     e.preventDefault();
134     const rect = e.target.getBoundingClientRect();
135 
136     x = e.clientX - rect.left;
137     y = e.clientY - rect.top;
138     drawLine(x, y);
139 }
140 
141 // 線を引く
142 function drawLine(x, y) {
143     // キャンバスの描画モードを変更
144     switch (mode) {
145         case Mode.pen: // ペン
146             context.globalCompositeOperation = "source-over";
147             break;
148 
149         case Mode.eraser: // 消しゴム
150             context.globalCompositeOperation = "destination-out";
151             break;
152     }
153 
154     context.lineCap = "round";
155     context.strokeStyle = "rgb(" + color[0] + "," + color[1] + "," + color[2] + ")";
156     context.lineWidth = weight;
157     context.beginPath();
158 
159     if (px == null || py == null)
160         context.moveTo(x, y);
161     else
162         context.moveTo(px, py);
163 
164     context.lineTo(x, y);
165     context.stroke();
166     context.closePath();
167 
168     px = x;
169     py = y;
170 }
171 
172 // 元に戻す
```

```
function undo() {
    copyCanvas(history.undo());
}

// やり直し
function redo() {
    copyCanvas(history.redo());
}

// キャンバスをコピー
function copyCanvas(srcCanvas) {
    if (srcCanvas == null)
        return;

    if (mode == Mode.eraser)
        context.globalCompositeOperation = "source-over";

    context.clearRect(0, 0, canvas.width, canvas.height);
    context.drawImage(srcCanvas, 0, 0);

    if (mode == Mode.eraser)
        context.globalCompositeOperation = "destination-out";
}

// 投稿
function post() {
    // 合成用のキャンバスを作成
    let finalCanvas = document.createElement("canvas")
    finalCanvas.width = canvas.width;
    finalCanvas.height = canvas.height;

    // 元の画像と合成
    let finalContext = finalCanvas.getContext("2d");
    finalContext.drawImage(document.getElementById("imageCanvas"), 0, 0);
    finalContext.drawImage(canvas, 0, 0);

    // 投稿
    let image = finalCanvas.toDataURL("image/png");
    document.forms.new_draw.image.value = image;
    document.forms.new_draw.submit();
}
```

10　画像

プログラム中で用いられる画像を図 2–13 に示す。

図 2　logo.png

図 3　login.png

図 4　logout.png

図 5　signup.png

図 6　post.png

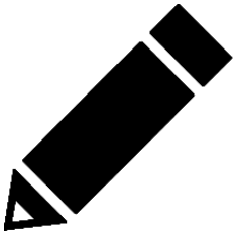

図 7　pen.png

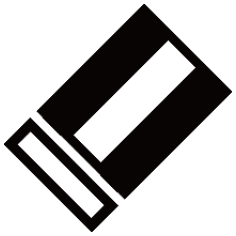

図 8　eraser.png

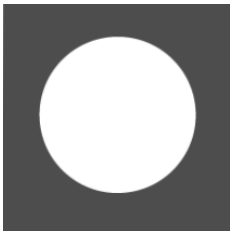

図 9　pen–big.png

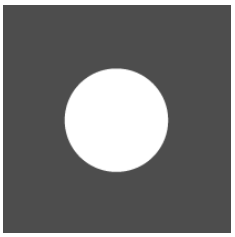

図 10　pen–medium.png

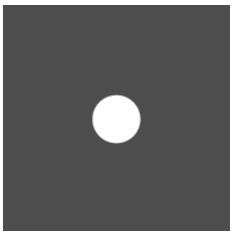

図 11　pen–small.png

図 12　undo.png

図 13　redo.png

11　実装機能

11.1　アカウント登録

アカウントを新規登録するためのフォームを図 14 に示す。ユーザー名、ユーザー ID、パスワードを入力し、新規登録ボタンを押すと、/regist に POST し、drawing.rb の 142 行目でキャッチする。ユーザー ID が既に存在していたり、確認用のパスワードが異なっていればエラーページを表示し、正常な入力なら accounts テーブルにアカウントを登録する。生のパスワードにランダムな文字列であるソルトを連結した文字列のハッシュ値を記録しておくことで、生のパスワードが保存されることはないので、テーブルの内容が盗み見られても安全である。

図 14　新規登録フォーム

11.2　ログイン

ログインをするためのフォームを図 15 に示す。ユーザー ID とパスワードを入力し新規登録ボタンを押すと、/auth に POST し、drawing.rb の 178 行目でキャッチする。パスワードにソルトを連結した文字列のハッシュ値と、新規登録の際に記録したハッシュ値を比較し、一致していたら Cookie にログイン中であることを記録する。ユーザー ID が存在していなかったり、パスワードが間違っていたら、図 16 のページを表示する。

図 15　ログインフォーム

図 16　ログイン失敗

11.3　ログアウト

ログアウトボタンを押した場合は、Cookie の内容を消去する。そうすることで、ログイン中の情報が消えるため、ログアウトできる。

11.4　テキスト投稿

図 17 のように、名前とテキストを入力し、“書き込み” ボタンを押下すると掲示板にその内容が書き込まれる。書き込まれた様子を図 18 に示す。図 18 を見ると、投稿内容の他に、投稿番号、書き込んだユーザーの名前、日時も記録されていることが分かる。

図 17　投稿前の様子

図 18　投稿後の様子

入力フォームの HTML は、bbs.erb の 68 行目以降に実装されている。ボタンを押すと/new_text に POST し、それをサーバーサイドのプログラム drawing.rb の 54 行目でキャッチする。ここでは、投稿番号、日時、投稿内容、ユーザー ID を取得し、ActiveRecord を介してデータベースに投稿を記録している。

11.5　投稿の削除

ここで、別のアカウントでログインし、投稿を行った状態を図 19 に示す。図 19 を見ると、2 つ目の投稿の横にだけ “削除” ボタンがあるのが分かる。これは、自身が行った投稿の横にのみ表示される。このボタンを押すと投稿が削除され、図 20 のように、投稿内容が表示されなくなる。

“削除” ボタンを押すと/delete にリダイレクトされるように設定されており、このとき bbs.rb の 80 行目以降が実行される。当該の投稿をデータベースから検索し、exist カラムの値を 0 にすることで、投稿が削除される。


掲示板 - ABCDrawing
localhost:4567/1
ABCDrawing
おえかき掲示板
ログアウト
新規登録
掲示板
<< 1/1 >>
1 だいちよ 2020/02/17(Mon) 11:17:52
Hello, world!
こんにちは、世界！
2 burdock 2020/02/17(Mon) 11:19:15
削除
Delete Test
最大512文字
書き込み
イラストを書き込み

図 19　削除前の状態

図 20　削除後の状態

11.6　サニタイズと改行

<や’ などの特殊な文字も入力できるように入力内容に対してサニタイズを行った。サニタイズをすることで、HTML タグを不正に実行される脆弱性を予防する効果もある。また今回は、改行コードを HTML タグ
に変換することで改行の入力を可能にした。

特殊な文字と改行を混じえた投稿の様子を図 21 に示す。図 21 を見ると、各記号が正しく投稿されており、改行もされていることが分かる。

図 21　サニタイズと改行

11.7　入力文字数制限

bbs.erb の 72, 73 行目を見ると、textarea 要素に required, maxlength 属性が付与されていることが分かる。これにより、文字を入力していなくても文字数が多すぎても投稿出来ないようになっている。図 22 は、文字を入力せずに投稿ボタンを押した時の挙動である。

図 22　入力必須フォーム

11.8　イラスト投稿

テキストの他に、自分で描いたイラストを投稿できる機能を実装した。イラストを描くフォームを図 23 に示す。消しゴム、ペンの太さを指定できる他、ペンの色指定と Undo/Redo をできるようにした。

図 23　おえかきフォーム

イラストを描いた様子を図 24 に示す。この状態で投稿ボタンを押すと、図 25 のように、描いたイラストが投稿できる。

図 24　イラストを描いた様子

図 25　イラストの投稿

11.9　おえかき

おえかきの処理は、リスト 17 の draw.js で行っている。

canvas 上に線を引く関数 `drawLine` は、142 行目から定義されている。canvas のメソッドである `strokeStyle`, `lineWidth` で先の色と太さを指定した後、`lineTo` で直前のマウス座標から現在のマウス座標に向かって線を引く。`drawLine` 関数を、マウスが動く度に呼び出されるイベントリスナー内で呼び出すこと

で、イラストが描ける。

操作履歴を管理するクラス `History` は、32 行目から定義されている。`push` メソッドを実行すると、現在の canvas の状態がそのまま `log` というリストに記録される。`undo` メソッドを実行すると、最新から 1 つ前の状態の canvas を `log` から取得し、`return` する。`redo` メソッドはその逆である。

投稿ボタンが押されると、198 行目の `post` 関数が呼び出される。描いたイラストを Base64 でエンコードした URL に変換し、それをサーバーに POST し、データベースに登録することで投稿する。

11.10　他人の絵を元にしたイラスト投稿

1 から自分でイラストを描く以外に、他人が描いたイラストを元にして追加で描き込める機能を実装した。図 25 の投稿にある、“この絵を元にしておえかき” というリンクをクリックすると、図 26 のようにそのイラストの上から描き込むことができる。この状態で投稿ボタンを押したときの様子を図 27 に示す。図 27 を見ると、元にした投稿へのリンクと共にイラストが投稿されているのが分かる。

図 26　他人の絵を元にイラストを描く様子

図 27　他人の絵を元にイラストを投稿した様子

11.11　ページ切り替え

10 件以上の投稿がある場合には、2 ページ以降が追加される機能を実装した。図 28 のように、すでに 10 件の投稿がある状態で投稿をすると 2 ページ目に投稿が追加される。投稿された後の状態を図 29 に示す。図 29 を見ると、新たに 2 ページ目が追加され、そこにリダイレクトされていることが分かる。

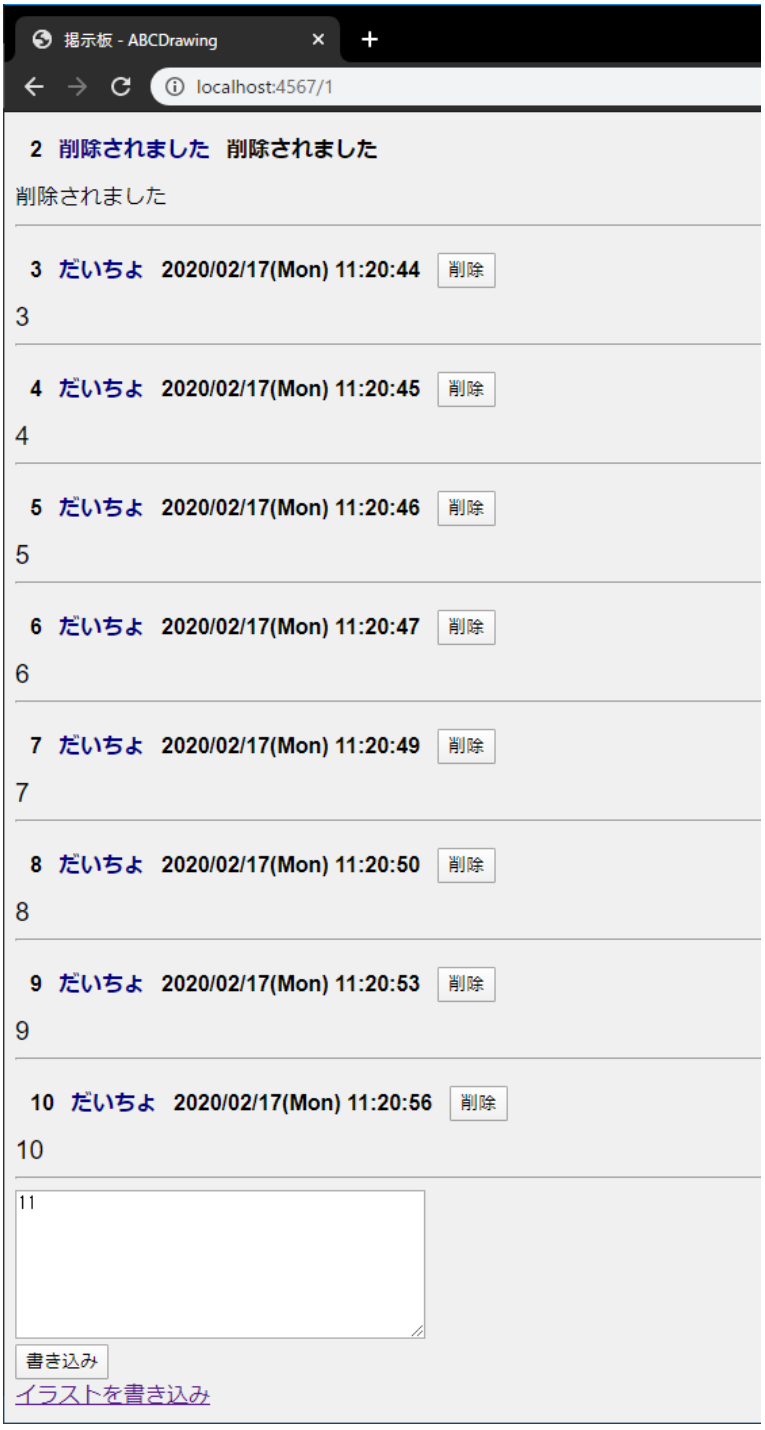

図 28　10 件の投稿がある状態

図 29　11 件目の投稿

ページ番号は URL に指定できる。指定されたページ番号をを bbs.rb の 223 行目で受け取り、そのページに相当する投稿を表示する。なお、範囲外の URL が入力された場合は、エラーページを表示する。エラーページが表示された様子を図 30 に示す。

図 30　エラーページ

11.12　CSS の設定

ページを見やすくするために、CSS を設定した。全てのページで共通するスタイルはリスト 14 の common.css で指定し, 各ページのみで適用させたいスタイルは別途指定するような仕様にすることで、メンテナンス性を向上させた。

CSS を設定していないページを図 31 に示す。CSS を設定することで色合いや間隔などを調整することで見やすくなることが分かる。

図 31　CSS を設定していないページ