

Laravel 資料 2 リクエストの受渡し

ログイン結果ページの作成

資料 1 に引き続き、今回はログイントップページで入力された内容を表示する画面を作成する。作成の流れはログイントップページを作成した際と同様である。本資料ではログイン結果ページに以下の機能を実装するまでを説明する。

- ログイントップページで入力されたリクエストを受け取る
- リクエストからパラメータを取り出す
- 受け取ったパラメータをログイン結果ページのビューに渡し表示する

今回は段階的にログイン結果ページの実装を行っていく。

ルーティング設定～ビューの表示確認

■ ルーティング設定

`routes/web.php` に以下を追記。

```
Route::post('/login', 'LoginController@postIndex');
```

ログイントップページへのルーティングと見比べてみると以下の点が異なっている

```
Route::get('/login', 'LoginController@getIndex');  
Route::post('/login', 'LoginController@postIndex');
```

- `Route::`の後に **post**
- @以降で指定している関数が違う

指定している URL は同じだが、HTTP メソッド (get / post) で振り分けられるため問題ない。

なお、HTTP メソッド名と「～@○○ Index」の○○が対応しているが、全く関係ない名前でも構わない。

例えば「Route::**get**('/login', 'LoginController@**post**Index);」と書いても、非常に紛らわしいが動作する。

■ postIndex メソッドの作成

postIndex メソッドを作成する。

現時点では getIndex() と名前と View 以外全く同じである。

【app\Http\Controllers\LoginController.php】

```
public function postIndex()  
{  
    // view ファイルを返却  
    return view('login/result');  
}
```

■ result.blade.php の作成

login.blade.phpと同じくLaravelSample/resources/views/login にresult.blade.phpを作成する。

{{!-- --}}で囲まれた部分はコメントである。blade の場合はこう記述する。

【resources\views\login\result.blade.php】

```
@extends('layout/layout') @section('content')
<h1>ログイン入力内容</h1>
<div class="row">
  <div class="col-sm-12">
    <a href="/login" class="btn btn-primary" style="margin:20px;">
      ログイントップページに戻る
    </a>
  </div>
</div>

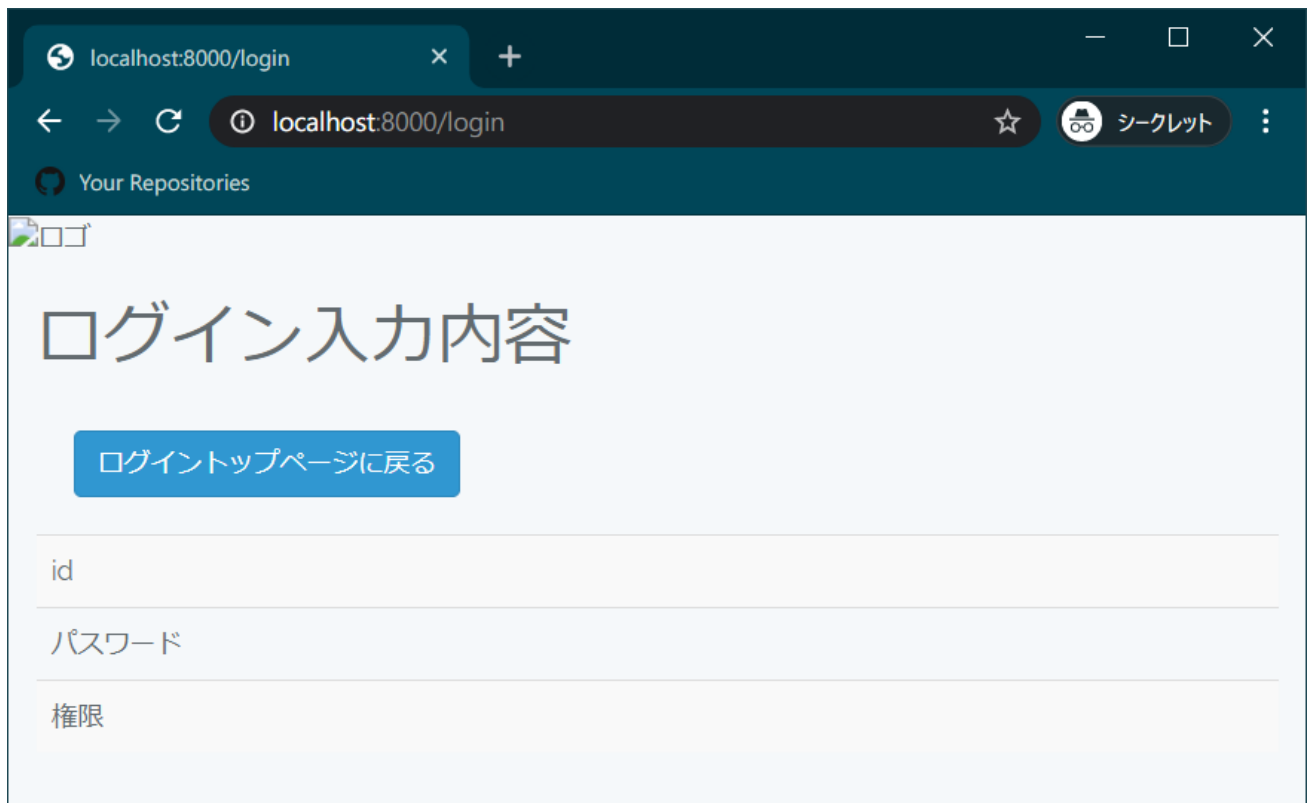
<table class="table table-striped">
  <tr>
    <td>id</td>
    <td>
      {{!-- ここにpostIndex()から渡されたデータが入る --}}
    </td>
  </tr>
  <tr>
    <td>パスワード</td>
    <td>
      {{!-- ここにpostIndex()から渡されたデータが入る --}}
    </td>
  </tr>
  <tr>
    <td>権限</td>
    <td>
      {{!-- ここにpostIndex()から渡されたデータが入る --}}
    </td>
  </tr>
</table>

@stop
```

■ 画面を確認

ここで一旦、動作するか確認しておく。

[ログイントップページ](#) にアクセスした後、ログインボタンを押下してログイン結果ページに遷移する。



The screenshot shows a web browser window with the address bar displaying 'localhost:8000/login'. The page has a dark blue header with the text 'Your Repositories'. The main content area has a light blue background and the title 'ログイン入力内容' (Login Input Content). Below the title is a blue button labeled 'ログイントップページに戻る' (Return to Login Top Page). There are three input fields with labels 'id', 'パスワード' (Password), and '権限' (Authority/Permission).

リクエストパラメータ取得

■ postIndex メソッドの編集

ログイントップページから送られてきたリクエストを受け取り、リクエストパラメータを取得するように変更する。-が付いている行が変更前、+が付いている行が変更後である。

【app\Http\Controllers\LoginController.php】

```
-public function postIndex()  
+public function postIndex(Request $request)  
{  
+ // リクエストパラメータを配列として全件取得  
+ $input = $request->all();  
+ dump($input);  
  // view ファイルを返却  
  return view('login/result');  
}
```

リクエストを受け取るには、単純にメソッドの引数に Request クラスの引数を追加すればよい。資料通りに写していれば問題ないだろうが、`use Illuminate\Http\Request;`を忘れていた場合 ReflectionException になる。

\$request->all()

Request クラスの all メソッドはリクエストパラメータを全件配列として返却する。

dump()

var_dump()のような機能を持つ Laravel が用意している関数。
色がついたり配列の開閉ができたりして var_dump()より使いやすい。

■ （補足）要素を指定して入力データを取得

以下のようにビューのinputタグで付けたnameを指定することで、その値のみを取得することもできる。

```
$request->input('name');
```

■ 画面を確認

再び画面で確認する。

[ログイントップページ](#) にアクセスした後、適当になにか入力してログインボタンを押下する。

dump()が上部に の中身を出力している。入力した通りのデータが出力されていれば問題ない。

"_token"には見覚えがないかもしれないが、これは CSRF トークンで、ログイントップページのビューの form タグ内にある`{{ csrf_field() }}`が生成している。

Laravel はフォームでデータを送信する場合、必ず CSRF トークンを使うよう定めているため、書かなかったらエラーになる。覚えておくこと。

The screenshot shows a web browser window with the address bar at `localhost:8000/login`. The page content displays a JSON array of login data:

```
array:4 [▼  
  "_token" => "FjEpP4PMfsEZ0dTRmSXFh5s8zoP5mOC8T1KhdUUr"  
  "loginid" => "確認"  
  "loginpassword" => "kakunin"  
  "authority" => "1"  
]
```

Below the dump, the page title is "ログイン入力内容" (Login Input Content). There is a blue button labeled "ログイントップページに戻る" (Return to Login Top Page). Below the button are three input fields labeled "id", "パスワード" (Password), and "権限" (Authority).

コントローラからビューに値を渡す

■ postIndex メソッドの編集

前項でデータが受け取れていることまでは確認した。

確認が終わったため、dump()を書いた行は消すなりコメント化するなりしておく。

あとはビューにデータを渡すだけである。

ビューにデータを渡す方法には3通りの方法がある。好きな方法で実装するとよい。

それぞれ書き方と、ソースの変更箇所を示す。

1.view()第2引数に連想配列として渡す

書き方

```
public function test () {  
  
    $iroha = '色は';  
    $nihoheto = '匂へと';  
  
    return view('ビュー', [  
        'test1' => $iroha,  
        'test2' => $nihoheto'  
    ]);  
}
```

postIndex()の編集

【app\Http\Controllers\LoginController.php】

```
public function postIndex(Request $request)  
{  
    // リクエストパラメータを配列として全件取得  
    $input = $request->all();  
-    dump($input);  
    // view ファイルを返却  
-    return view('login/result');  
+    return view('login/result', ['input' => $input]);  
}
```

2.view()第2引数に compact 関数で渡す

書き方

```
public function test () {  
  
    $iroha = '色は';  
    $nihoheto = '匂へと';  
  
    // 変数ではなく変数名を文字列として渡していることに注意  
    return view('ビュー', compact('iroha', 'nihoheto'));  
}
```

postIndex()の編集

【app\Http\Controllers\LoginController.php】

```
public function postIndex(Request $request)  
{  
    // リクエストパラメータを配列として全件取得  
    $input = $request->all();  
-    dump($input);  
    // view ファイルを返却  
-    return view('login/result');  
+    return view('login/result', compact('input'));  
}
```


3.view()の with メソッドで渡す

書き方

```
public function test () {  
  
    $iroha = '色は';  
    $nihoheto = '匂へと';  
  
    return view('ビュー')->with([  
        'test1' => $iroha,  
        'test2' => $nihoheto'  
    ]);  
}
```

postIndex()の編集

【app\Http\Controllers\LoginController.php】

```
public function postIndex(Request $request)  
{  
    // リクエストパラメータを配列として全件取得  
    $input = $request->all();  
-    dump($input);  
    // view ファイルを返却  
-    return view('login/result');  
+    return view('login/result')->with('input', $input);  
}
```

コントローラから渡されたデータを表示するように変更する。

【resources\views\login\result.blade.php】

```
(略)
<tr>
  <td>名前</td>
  <td>
-    {{-- ここにpostIndex()から渡されたデータが入る --}}
+    {{ $input['name'] }}
  </td>
</tr>
<tr>
  <td>パスワード</td>
  <td>
-    {{-- ここにpostIndex()から渡されたデータが入る --}}
+    {{ $input['password'] }}
  </td>
</tr>
<tr>
  <td>権限</td>
  <td>
-    {{-- ここにpostIndex()から渡されたデータが入る --}}
+    @if(empty($input['authority']))未選択
+    @elseif($input['authority'] === '1')管理者
+    @elseif($input['authority'] === '2')一般
+    @endif
  </td>
</tr>
(略)
```

コントローラからはどの方法でもという変数（中身も）で渡されて来ている。

`{{ 変数 }}`は`<?php echo htmlspecialchars(変数) ?>`と同じ動作をする。

`$_POST`の場合と同じく、ログイントップページのフォームの name 要素で設定した名前と同じキーで値が格納されているため、`{{ }}`内に書くことで値を表示することができる。

blade には制御構文も用意されており、権限の表示では if 文を使って表示する文字列を変えている。

資料最終段階のソースコード

■ 【app\Http\Controllers\LoginController.php】

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class LoginController extends Controller
{
    public function getIndex()
    {
        // view ファイルを返却
        return view('login/login');
    }

    public function postIndex(Request $request)
    {
        // リクエストパラメータを配列として全件取得
        $input = $request->all();

        // 好きな方法でViewに値を渡す
        // return view('login/result', ['input' => $input]);
        // return view('login/result', compact('input'));
        return view('login/result')->with('input', $input);
    }
}
```

```
@extends('layout/layout') @section('content')
<h1>ログイン入力内容</h1>
<div class="row">
  <div class="col-sm-12">
    <a href="/login" class="btn btn-primary" style="margin:20px;">
      ログイントップページに戻る
    </a>
  </div>
</div>

<table class="table table-striped">
  <tr>
    <td>名前</td>
    <td>
      {{ $input['name'] }}
    </td>
  </tr>
  <tr>
    <td>パスワード</td>
    <td>
      {{ $input['password'] }}
    </td>
  </tr>
  <tr>
    <td>権限</td>
    <td>
      @if(empty($input['authority']))未選択
      @elseif($input['authority'] === '1')管理者
      @elseif($input['authority'] === '2')一般
      @endif
    </td>
  </tr>
</table>

@stop
```

付録 ログ画像を設定する

ログ画像を設定しているのは以下の箇所である。

【layout.blade.php】

```

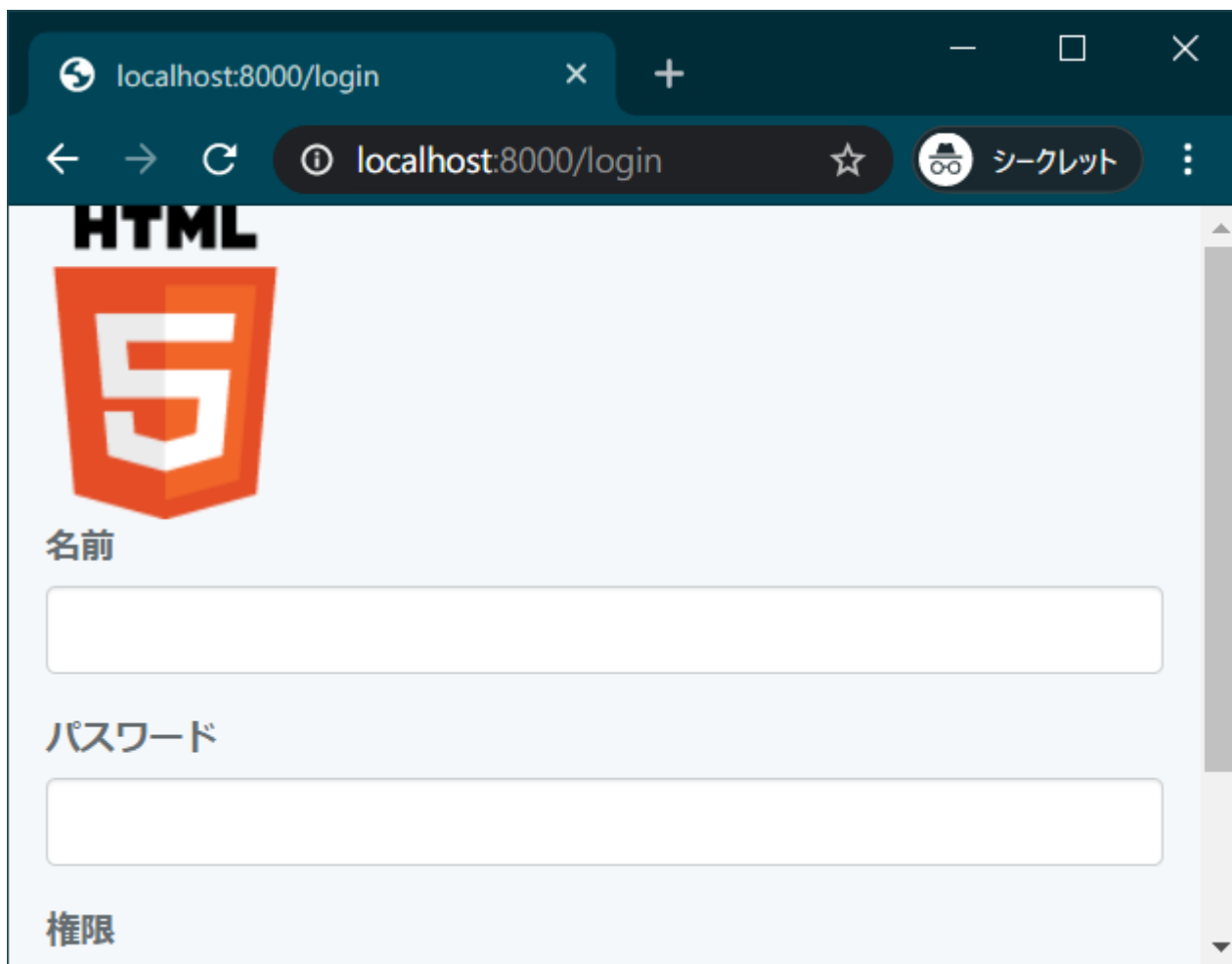
```

asset()はpublicフォルダのパスを返す。

resources下にassetsフォルダが存在するが関係ないようだ。

ビュー側で指定している通りにpublicフォルダに新たにimgフォルダを作成し、そこにhtml5b.pngという画像を配置すればロゴ画像が表示される。

画像は[W3C公式](#)からダウンロードするとよい。



位置調整は必要だが、ロゴ画像が表示された。

付録 よく見がちなエラーと対処

とりあえずエラー文を翻訳すれば概ねどんなエラーであるかは分かる。
あとエラー文で検索すれば大抵解決方法が出てくる。

■ このサイトにアクセスできません

localhost で接続が拒否されました。

web サーバが起動していない。

Laravel は XAMPP の Apache ではなく Laravel の web サーバを起動しないと動かない。

`php artisan serve`した shell は閉じずに置いておくこと。

閉じたらサーバも終了する。

■ MethodNotAllowedHttpException

ルーティングに該当する URL が存在しない。

`web.php`のルーティングの設定が正しくできているか確認してみる。

■ ReflectionException

Class App\Http\Controllers\Request does not exist

クラスが見つからない。

上の例では Request クラスが見つからないというエラー。

`use` でクラスが存在する場所を設定する。