サーバーサイドスクリプトⅡ

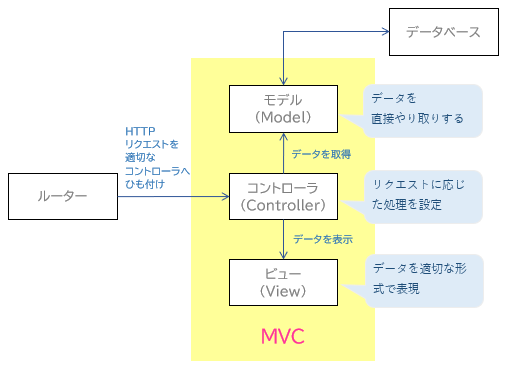
９．DB処理（DELETE）

第９回目では、データ削除を行っていきます。

下記③の手順を行っていきます。（①②は第5回、④は第6回で作成および設定済みのものを使用。削除処理のみなので⑤はナシ）

1. **マイグレーション**
2. **モデルの作成**
3. **コントローラの作成**
4. **ルーティングの設定**
5. **ビューの作成**

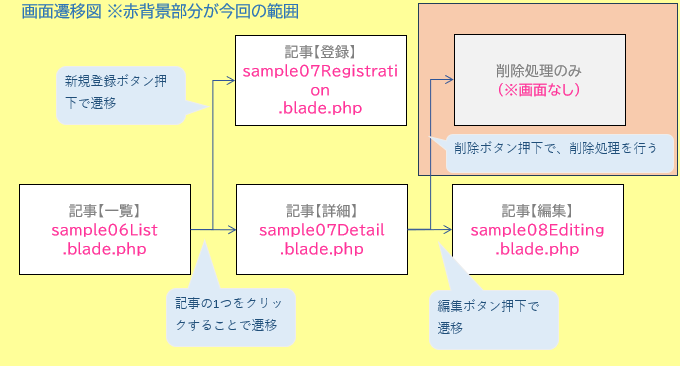
■図：　MVC



今回の処理範囲は下図の通りです。

　＊記事詳細（sample07Detail.blade.php）は７回目で作成済み。

　　記事詳細で「削除」ボタン押下時に、DBからデータ削除処理をします。削除用の画面はありません。



９－１． DELETE

記事詳細画面の「削除」ボタン押下時、該当データをDBから削除する処理を作成します。

■使用メソッド＆ルーティング名

|  |  |  |  |
| --- | --- | --- | --- |
| HTTPメソッド | パス | メソッド | 役割 |
| DELETE | /sample06/{sample06} | **destroy** | 指定されたリソースを削除 |

＊一部過去資料にて、メソッドもdeleteになっていましたが、誤植です。

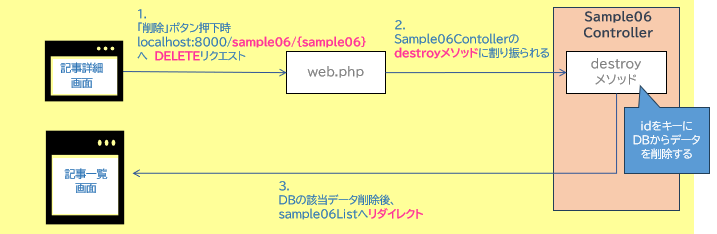
メソッドはdestroyが正しいです。【最新版は修正済】

＊HTTPメソッドはDELETEが正なので注意。

記事詳細画面で「削除」ボタンが押された時、DELETEデータを受け取り、DBから該当データを削除する処理はコントローラのdestroyメソッドが担います。

処理の流れは下図の通りです。

■記事編集画面でのDB削除　処理の流れイメージ



９－１－１． 記事詳細画面の動作を追加する

記事編集画面で必要事項を入力後、「削除」ボタンを押したときのリクエスト送信設定を行います。

削除ボタンを押すと、入力したデータを送信します。ただし、この時、DELETEリクエストを送信するものとする。

■フォーム設定①　<form>タグの属性指定。

|  |  |
| --- | --- |
| 属性 | 値 |
| action | コントローラのデータ削除用のメソッドをルーティング名にて指定。記事のidを引数として渡す。  （サンプル環境では、「sample06.destroy」を指定） |
| method | POST  ＊**HTMLのformタグではGETとPOSTにしか対応していない**ため、今回はPOSTを指定。 |

■@methodディレクティブ 前回（第8回目でも使用）



■フォーム設定②　formタグ内に追記

@method('DELETE')

　＊ルーティングでDELETEメソッドを指定しているため、formタグ内に正しいメソッドを指定。

以上をビューに反映します。

■sample07Detail.blade.php

　＊削除ボタンのフォーム部分のみ抜粋。

　＊赤枠部分を設定。

<form action="{{ route('sample06.destroy', $sample\_data->id) }}" method="POST">

    @csrf

    @method('DELETE')

    <button type="submit"

        class="block w-16 text-white text-center bg-red-600 hover:bg-red-500 mr-5 px-3 py-2 rounded-md">削除</button>

</form>

詳細画面で「削除」ボタンを押すと、DELETEリクエストが送信され、destroyメソッドが呼び出される動作となります。destroyメソッドは引数として記事idとリクエストデータを受け取ります。

９－１－２． DBから削除する

destroyメソッドの処理を記述していきます。destroyメソッドは、記事idを引数として受け取ります。

受け取ったidのデータをDBから削除します。

削除にはEloquentメソッドのdestroyもしくはdeleteを使用します。

■destroyメソッド（Eloquentメソッド）

　＊コントローラのdestroyメソッドとは違います。



■使用例　＊変数$idにはid番号が格納されていると仮定。

// インスタンス化せずに使用

Sample::destroy($id);

もしくは

// インスタンス化して使用

$sampleDao = new Sample();

$sampleDao->destroy($id);

■deleteメソッド（Eloquentメソッド）



■使用例　　＊変数$idにはid番号が格納されていると仮定。

　＊先にデータを特定する必要がある。例ではfindメソッドを使用。

// インスタンス化せずに使用

Sample::find($id)->delete();

もしくは

// インスタンス化して使用

$sampleDao = new Sample();

$sampleDao->find($id)->delete();

■Sample06Controller.php

＊コントローラのdestroyメソッドのみ抜粋

＊transactionは第7回目に記載済。

public function destroy(string $id)

{

    //　インスタンス化

    $sampleDao = new Sample();

    DB::transaction(function () use ($sampleDao, $id) {

        // destroyメソッドにてidを指定して削除

        $sampleDao->destroy($id);

    });

// 一覧画面にリダイレクト

    return redirect()->route("sample06.index");

}

９－１－２． 動作確認する（重要！）

実装したら、データ削除処理が正しく行われることを確認します。データ削除後は一覧画面が表示されること。

ただし、以下に注意してください。

|  |  |
| --- | --- |
| 警告 単色塗りつぶし | （＊）論理削除  今回のテーブル設計では「論理削除」使用していますので、画面表示上はデータが表示されずに削除されたように見えますが、**実際のデータベースのテーブルからは削除されていないことを確認**してください。 |

■一覧画面

グラフィカル ユーザー インターフェイス, Web サイト

自動的に生成された説明

**記事選択**

■詳細画面

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

**「削除」ボタン押下**

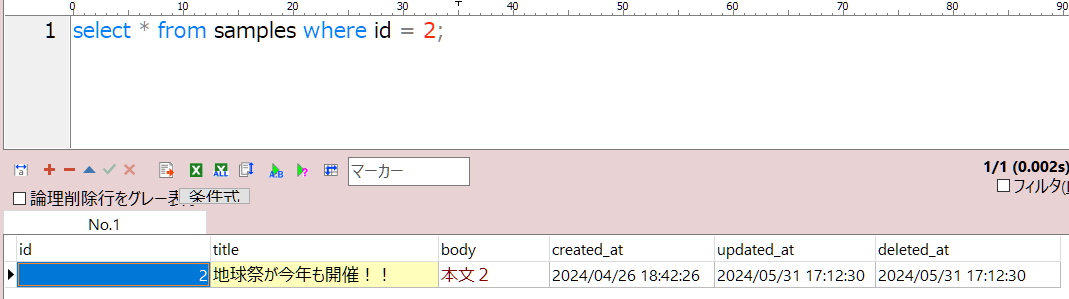
■削除後、一覧画面

グラフィカル ユーザー インターフェイス, Web サイト

自動的に生成された説明

**2件目の「地球祭～～」のタイトルのデータが削除されている。**

■DBテーブル確認（samplesテーブル）



**＊idが２のデータは存在しているが、「deleted\_at」に日時が入っているため、削除扱いとなる。（selectでデータが抽出されない。）**

９回目は以上です。