Blade

Bladeとは

Bladeは、Laravelに用意されているテンプレートエンジンとなり、シンプルながら強力なテンプレート機能を提供してくれます。

Bladeはテンプレート内でプレーンなPHPコードの使用を制限しません。

Bladeファイルの設置場所

Bladeファイルは「 ○○○.blade.php 」ファイル拡張子を使用し、通常は「 resources/views 」フォルダ内に保存します。

コントローラーからBladeを呼び出す

Bladeは、グローバルなviewヘルパーを使用してルートまたはコントローラから返します。

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class Sample01_1Controller extends Controller
{
    //
    public function index() {
        return view('sample01_1');
    }
}
```

```
use App\Http\Controllers\Sample01_Controller;

Route::get('sample01_1', [Sample01_1Controller::class, 'index']);
```

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>sample01 - サーバーサイドスクリプト演習２</title>
</head>
<body>
    <h1>Bladeを使ったページの表示</h1>
</body>
</html>
```

Bladeへデータを渡す

viewヘルパーの２番目の引数にBladeへ渡したいデータを指定することで、コントローラーからBladeファイルへデータを渡すことができます。

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
```

```
class Sample01_Controller extends Controller
{
    //
    public function index() {
        $message = "コントローラーからBladeへ渡されたデータです。";
        return view('sample01_1', ['message' => $message]);
    }
}
```

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>sample01 - サーバーサイドスクリプト演習２</title>
</head>
<body>
    <h1>Bladeを使ったページの表示</h1>
    <p>{{ $message }}</p>
</body>
</html>
```

Bladeディレクティブ

Bladeは、一般的なPHP構文のような役割を果たす短縮記法を提供してくれています。

Bladeディレクティブを利用することで、HTMLの中にPHP制御構文を簡単に美しく簡潔な形でテンプレートに簡単に組み込むことができます。

データの表示

変数を中括弧で囲むことで「 {{ $variable }} 」Bladeに渡されたデータを表示することができます。

エスケープせずにデータの表示

Bladeはデフォルトで XSS対策として、htmlspecialchar関数を通してデータをHTMLエンティティへエンコーディングします。

データをエスケープしたくない場合は、「 {!! $variable !!} 」を使用します。

コメント

Blade内でHTMLへ出力されずにコメントを記載したい場合は、「 {{-- コメント --}} 」を使用します。

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class Sample01_Controller extends Controller
{
    //
    public function index() {
        $message = "コントローラーからBladeへ渡されたデータです。";
        $records = ['PHP', 'Ruby', 'Python', 'NodeJs'];
        return view('sample01_1', compact('message', 'records'));
    }
}
```

PHP Manual - compact
https://www.php.net/manual/ja/function.compact.php

条件分岐

if文

「 @if 」「 @elseif 」「 @else 」「 @endif 」ディレクティブを使用してifを作成できます。

```
@if (count($records) === 1)
    <p>１レコード</p>
@elseif (count($records) > 1)
    <p>複数レコード</p>
@else
    <p>レコードがありません</p>
@endif
```

switch文

switchは、「 @switch 」「 @case 」「 @break 」「 @default 」「 @endswitch 」ディレクティブを使用して作成できます。

```
@switch($i)
    @case(1)
        最初のケース
        @break
    @case(2)
        ２番目のケース
        @break
    @default
        デフォルトケース
@endswitch
```

繰り返し

PHPのループ構造を操作するための簡単なディレクティブを提供します。繰り返しますが、これらの各ディレクティブは、対応するPHPと同じように機能します。

for文

forは、「 @for 」「 @endfor 」ディレクティブを使用して作成できます。

```
@for ($i = 0; $i < 10; $i++)
    <p>The current value is {{ $i }}</p>
@endfor
```

foreach文

foreachは、「 @foreach 」「 @endforeach 」ディレクティブを使用して作成できます。

```
@foreach ($users as $user)
    <p>{{ $user }}</p>
@endforeach

{{-- ネストされた要素がオブジェクトの場合はアロー演算子を使います --}}
@foreach ($users as $user)
    <p>{{ $user -> name }}</p>
@endforeach
```

while文

whileは、「 @while 」「 @endwhile 」ディレクティブを使用して作成できます。

```
@while (true)
    <p>メビウスの輪</p>
@endwhile
```