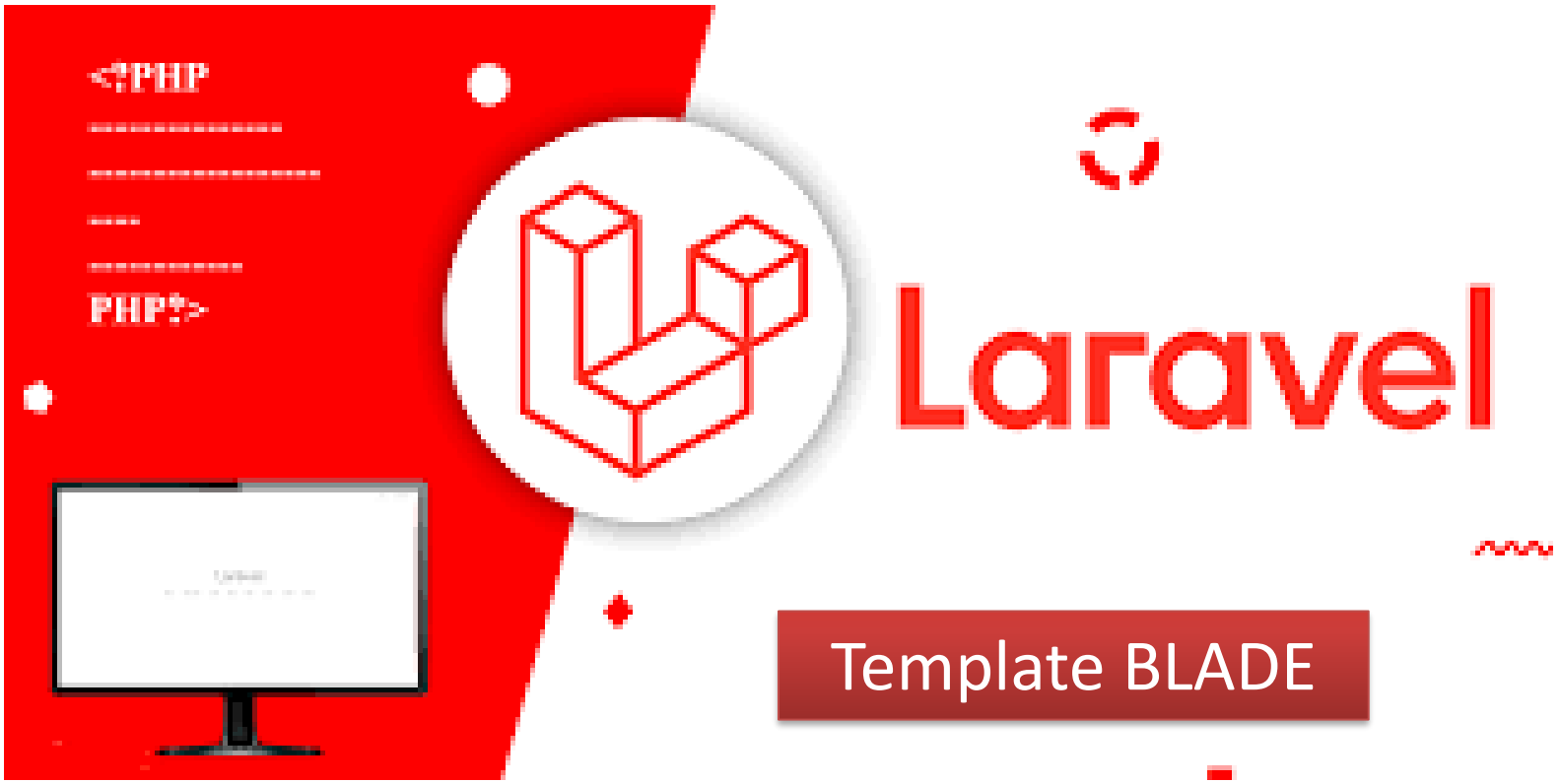
Développer en back-end



Réalisé par M. Hamid Belyazidi

Développer en back-end

Introduction

A. Découvrir le Framework PHP Laravel

1. Découvrir les notions fondamentales des Frameworks PHP
2. Préparer l'environnement de Laravel

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel
2. Maîtriser le Framework Laravel

C. Approfondir la programmation Laravel

1. Gérer la sécurité
2. Interagir avec la base de données
3. Manipuler l'ORM Eloquent
4. Prendre en charge les tests

D. Administrer un site à l'aide d'un CMS

1. Manipuler les éléments essentiels d'un CMS
2. Personnaliser graphiquement un site à l'aide d'un CMS
3. Manipuler les outils avancés d'un CMS

Conclusion

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

- ✓ Gestion du routage
- ✓ Utilisation des Middleware (définition, enregistrement, paramétrage, terminate)
- ✓ Protection CSRF
- ✓ Manipulation des contrôleurs
- ✓ Manipulation des requêtes http
- ✓ Manipulation des réponses http
- ✓ Manipulation des vues
- ✓ **Création des template Blade**
- ✓ Génération d'URL
- ✓ Manipulation des sessions HTTP
- ✓ Validation des données d'entrée
- ✓ Gestion des erreurs
- ✓ Journalisation (logging)

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓Définition

Blade est le moteur de Template utilisé par Laravel. Son but est de permettre d'utiliser du PHP sur notre vue mais d'une manière assez particulière. Pour créer un fichier qui utilise le moteur de Template Blade il vous faut ajouter l'extension ***".blade.php".*** Comme nous l'avons vu dans la présentation de l'architecture de Laravel, les fichiers de vos vues se situent dans le dossier ***resources/views***.

La première fonctionnalité la plus basique de Blade est l'affichage d'une simple variable :

Exemple :

Si je transmets à ma vue la variable ***$maVariable*** = ***'Hello World !'*** Dans ma vue ***{{ $maVaribale }}*** affichera 'Hello World' .

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Pour vous permettre de ne pas échapper les caractères html

Exemple

```
$mavariable = '<h1>Mon Titre</h1>';
{{ $maVariable }}  <!-- donnera : '<h1>Mon Titre</h1>' -->
{!! $maVariable !!}  <!-- donnera : 'Mon Titre' dans une balise HTML h1 -->
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle

Blade vous permet de manipuler des données comme le ferait le PHP, il vous est donc possible d'utiliser les différentes structures de contrôle PHP existantes. À la différence que leur écriture diffèrent un tout petit peu.

Pour mettre en place une condition :

@if / @elseif / @else, @switch, @isset, @empty

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle (@if, @elseif, @else)

```
$animal = 'cheval';
@if ( $animal === 'chien' )
        <p>L'animal est un chien.</p>
    @elseif ( $animal === 'chat' )
        <p>L'animal n'est pas un chien.</p>
    @else
        <p>L'animal n'est ni un chat ni un chien.</p>
@endif
```

```
<!-- donnera : -->
<p>L'animal n'est ni un chat ni un chien.</p>
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle (@switch)

```
@switch($age)
    @case( $age < 18 )
        <p>La personne est mineure.</p>
        @break
    @case( $age > 18 )
        <p>La personne est majeure.</p>
        @break
    @default
        <p>valeur par défaut.</p>
@endswitch
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle (@isset)

```
$produit = 'costume';

@isset($produit)
    <p>Le produit existe</p>
@endisset
```

```
<!-- donnera : -->
<p>Le produit existe</p>
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle (@empty)

```
$produit = '';
@empty($produit)
    <p>Le produit n'existe pas.</p>
@endempty
```

```
<!-- donnera : -->
<p>Le produit n'existe pas.</p>
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle

Pour réaliser une boucle

@for, @foreach, @forelse et @while

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle (@while)

```
$i = 1;
@while ($i < 3)
    <p>$i est égal à {{ $i ++ }}</p>
@endwhile
```

```
<!-- donnera : -->
<p>$i est égal à 1</p>
<p>$i est égal à 2</p>
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle (@foreach)

```
$letters = ['a', 'b', 'c'];
@foreach ( $letters as $letter )
<!-- mon code exemple : -->
<p>Lettre : {{ $letter }} </p>
@endforeach
```

```
<!-- donnera : -->
<p>Lettre : a</p>
<p>Lettre : b</p>
<p>Lettre : c</p>
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle (@for)

```
$numbers = [1, 2, 3];
@for ($i = 0; $i < count($numbers); $i++)
    <p>Nombre : {{ $numbers[$i] }} </p>
@endfor
```

```
<!-- donnera : -->
<p>Nombre : 1</p>
<p>Nombre : 2</p>
<p>Nombre : 3</p>
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle (@forelse)

le @forelse est un foreach qui vous permet de retourner ce que vous souhaitez si le tableau est vide. Cela vous économisera l'ajout d'une condition if

```
$animals = ['chien', 'chat', 'cheval'];
@forelse ($animals as $animal)
    <li> {{ $animal }} </li>
@empty
    <p>Aucun animal existant.</p>
@endforelse
```

```
<!-- donnera : -->
<li>chien</li>
<li>chat</li>
<li>cheval</li>
```

```
<!-- Si le tableau $animals est vide -->
$animals = [];
<!-- Cela donnera : -->
<p>Aucun animal existant.</p>
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Les structures de contrôle ($loop)

Dans vos boucles vous avez également accès à une variable **$loop** qui permet de récupérer certaines informations concernant l'itération en cours dans la boucle. Par exemple si vous voulez mettre une condition sur la première ou la dernière itération :

```
$days = ['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi', 'samedi', 'dimanche'];
@foreach ($days as $day)
    @if ($loop->first)
        <p>C'est le premier jour de la semaine : {{ $day }} </p>
    @endif
    @if ($loop->last)
        <p>C'est le dernier jour de la semaine : {{ $day }} </p>
    @endif
@endforeach
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Créer un layout

la création de layout et l'organisation de nos fichiers de vues.

Définir le template

```
<!-- Voici le template situé dans resources/views/layout.blade.php -->
<!DOCTYPE html>
<html lang="fr">
<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>App Name - @yield('title')</title>
</head>
```

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Créer un layout

Définir le template(suite)

```
<body>
    @section('sidebar')
        Contenu de la section 'sidebar' du parent
    @show
    <div class="container">
        @yield('content')
    </div>
</body>
</html>
```

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Créer un layout

Les éléments de Blade : @section et @yield

@section() permet de déterminer la section d'un contenu.

@yield() permet de définir une zone qui permettra à l'enfant du Template d'y établir sa valeur. (yield signifie "céder" en anglais).

Voici comment étendre le Template que nous venu de voir :

Développer en back-end

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel

Création des Template Blade

✓ Créer un layout

```
<!-- Voici l'enfant du parent situé dans resources/views/child.blade.php -->
@extends('layout')
@section('title', 'Titre de la page enfant')

@section('sidebar')
    <!-- Ajout de contenu avant le contenu du parent -->
    @parent <!-- Contenu de la section 'sidebar' du parent -->
    <!-- Ajout de contenu après le contenu du parent -->
@endsection
@section('content')
    <!-- Contenu de la section 'content' de l'enfant -->
@endsection
```