**Задача**: создать шаблонизатор наподобие Blade, который находится в Laravel.

**Вход**:

@foreach($products as $product)

<td class="bg-warning">

@if($product->priceWithDiscount < $product->price)

<small class="text-secondary">

<s>${{ $product->price }}</s>

</small>

<span class="text-danger font-weight-bold">

${{ $product->priceWithDiscount }}

</span>

@else

<span class="font-weight-bold">

${{ $product->price }}

</span>

@endif

</td>

@endforeach

**Выход** (приблизительно):

|  |
| --- |
| <?php $\_\_currentLoopData = $products; $\_\_env->addLoop($\_\_currentLoopData); foreach($\_\_currentLoopData as $product): $\_\_env->incrementLoopIndices(); $loop = $\_\_env->getLastLoop(); ?>                         <th scope="col">                             <img src="<?php echo e($product->photos[0]->url); ?>" style="width: 150px">                             <br>                             <form action="<?php echo e(route('comparison.destroy', $product)); ?>" method="POST" class="d-inline">                                 <?php echo csrf\_field(); ?>                                 <?php echo method\_field('DELETE'); ?>                                 <button class="btn btn-outline-danger btn-sm p-2">                                     <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-trash" viewBox="0 0 16 16">                                         <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5zm2.5 0a.5.5 0 0 1 .5.5v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5zm3 .5a.5.5 0 0 0-1 0v6a.5.5 0 0 0 1 0V6z"/>                                         <path fill-rule="evenodd" d="M14.5 3a1 1 0 0 1-1 1H13v9a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2V4h-.5a1 1 0 0 1-1-1V2a1 1 0 0 1 1-1H6a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1h3.5a1 1 0 0 1 1 1v1zM4.118 4 4 4.059V13a1 1 0 0 0 1 1h6a1 1 0 0 0 1-1V4.059L11.882 4H4.118zM2.5 3V2h11v1h-11z"/>                                     </svg>                                 </button>                             </form>                             <a href="<?php echo e(route('product.description', $product)); ?>" target="\_\_blank" class="ml-3">                                 <?php echo e($product->name); ?>                             </a>                         </th>                     <?php endforeach; $\_\_env->popLoop(); $loop = $\_\_env->getLastLoop(); ?> |

**Функционал**:

* основной класс шаблонизатор - **TemplateEngine** отвечает за:
  + **отображение представлений** ( ВЫПОЛНЕНО TemplateEngine::**view**(«compiled.view.name.with.dot.syntax.as.a.path», [„parameter\_passed\_to\_the\_view“ => parameter\_value, ...]) :string) — представление загружается из папки со скомпилированными представлениями (/compiled) и ему передаются необязательные параметры. Точечный синтаксис пути к представлению (например, admin.payments) превращается в настоящий путь к папке (**/compiled/**admin/payments**.blade.php**) — при этом представление должно иметь специальное расширение .blade.php, иначе будет считаться, что оно не найдено и тогда будет исключение.
  + **Загрузку и временную подмену настроек** ВЫПОЛНЕНО , но пока без точечного синтаксиса (TemplateEngine::**config**(«configuration.parameter.name.with.dot.syntax.as.a.path», $new\_temporary\_value\_of\_parameter = null). Метод работает с настройками, сохраненными в сеансе. Он интерпретирует сохраненный в сеансе массив настроек через точечный синтаксис.
    - При вызове с одним параметром возвращает текущее значение данной настройки из сеанса (TemplateEngine::**config**(«parameter\_name»)).
    - При передаче второго параметра $new\_temporary\_value\_of\_parameter временно подменяет значение данной настройки в сеансе (TemplateEngine::**config**(«parameter\_name», new\_parameter\_value)).
  + **Загрузку всех настроек в сеанс** ВЫПОЛНЕНО (TemplateEngine::**loadConfig**()) - это позволяет получить настройки из любого места приложения или временно подменить их значение (в сеансе, не в файле). Все настройки храняться в файле /config.php и возвращаются в виде массива, который записывается в сеанс.
  + **Компиляцию представлений**. В метод TemplateEngine::**compile**(«path/to/file») :string передается путь к представлению (относительно папки /views). Метод преобразовует представление и возвращает результат в виде строки.
  + **Окончательную сборку всех представлений**. Метод TemplateEngine::**build**() перебирает все папки с представлениями (/views) и компонентами (/components), компилирует их, и сохраняет результат в соответствующие файлы в папке /compiled.
* Класс файловой системы — **Views****Mapper** выводит пути представлений до и после компиляции — этот список потом передается шаблонизатору, чтобы он знал, откуда читать и куда сохранять файлы.
  + **ViewsMapper**::**run**($base\_directory = null) :array находит все файлы, которые считаются представлениями (имеют расширение .blade.php и находятся в папке $base\_directory или её подпапках — т. е. Поиск рекурсивный) и возвращает массив путей к файлам представлений из основной папки ($base\_directory) и соответствующие им пути в папке скомпилированных представлений (/compiled). По умолчанию $base\_directory равна параметру «views\_directory» из файла конфигурации. Пример результата: [„/views/test.blade.php“ => „/compiled/test.blade.php“, ...]