Obchodní akademie, Vyšší odborná škola a Jazyková škola
s právem státní jazykové zkoušky Uherské Hradiště

POLOLETNÍ PROJEKT

DOKUMENTACE PROJEKTU NETTE-MAPY

2021 Vojtěch Vlachynský, 4.B

ANOTACE

Pololetní projekt do předmětů Projekt programování a Dokumentace projektu, který se zabývá vytvořením tutoriálu pro framework Nette podle projektu Todolist.

OBSAH

1 INSTALACE

1.1 Proč Nette používat a jak funguje?

Z vlastní zkušenosti můžeme říct, že Nette je vlastně velký balík knihoven (kolem 20) pro PHP. Základy v Nette jsou hračka, ale zároveň stále programujete v PHP a tím pádem musíte znát jeho vlastnosti. Framework vám vychází vstříc při programování i debugování za pomoci tzv. laděnky. Framework má obsáhlou komunitu, která ráda poradí s jakýmkoliv problémem a také byl zvolen jako nejpoužívanější framework v české republice. Jeho autorem a vývojářem je David Grudl, který má na starosti spoustu knihoven týkajících se Nette.

Na Nette jede hned několik významných projektů, jako je například GE Money, ČSFD, ESET a další. Tutoriál je určený pro aktuální verzi Nette 3.x a PHP 7.4 nebo novější.

1.2 Composer

Composer je jeden z klíčových nástrojů pro správu balíčků v PHP a umožňuje jednoduše instalovat přídavné knihovny do našeho projektu.

Windows instalátor: https://getcomposer.org/Composer-Setup.exe
Vyžaduje již předinstalované PHP. Například si můžete stáhnout PHP prostředí XAMPP.

Linux, MacOS instalátor:
K tomuto vám postačí tyto řádky kódu. Instalátor zároveň zkontroluje soubor php.ini a poté stáhne nejnovější verzi composeru composer.phar. Zadejte níže uvedené 4 očíslované řádky do příkazového řádku (v pořadí jak jdou za sebou).

```
1. php -r "copy('https://getcomposer.org/installer', 'composer-
   setup.php');"
2. php -r "if (hash_file('sha384', 'composer-setup.php') ===
   '756890a4488ce9024fc62c56153228907f1545c228516cbf63f885e036d3
   7e9a59d27d63f46af1d4d07ee0f76181c7d3') { echo 'Installer
   verified'; } else { echo 'Installer corrupt';
   unlink('composer-setup.php'); } echo PHP_EOL;"
3. php composer-setup.php
4. php -r "unlink('composer-setup.php');"
```

2 ZPROVOZNĚNÍ

S nainstalovaným composerem lze jednoduše stáhnout a nainstalovat základní kostru webu s knihovnami, také pod pojmem Web Project, který již obsahuje Nette Framework. K založení projektu si vytvoříme složku v kořenovém adresáři například /var/www (v Linuxu), nebo htdocs/www ve Windowsu.
Pomocí příkazového řádku spustíme příkaz:

```
composer create-project nette/web-project nette-blog
```

2.1 Kostra webu

Tutorial-na-framework-Nette-na-projektu-To-do-list/
├── app/ ← adresář s aplikací
│ ├── Presenters/ ← třídy presenterů
│ │ └── templates/← šablony
│ ├── Router/ ← konfigurace URL adres
│ └── Bootstrap.php ← zaváděcí třída Bootstrap
├── bin/ ← skripty spouštěné z příkazové řádky
├── config/ ← konfigurační soubory
├── log/ ← logování chyb
├── temp/ ← dočasné soubory, cache, …
├── vendor/ ← knihovny instalované Composerem
│ └── autoload.php ← autoloading všech nainstalovaných balíčků
└── www/ ← veřejný adresář - jediný přístupný z prohlížeče
└── index.php ← prvotní soubor, kterým se aplikace spouští

Adresář www/ je určen pro ukládání kaskádových stylů, obrázků, javascriptů a podobně. Navíc je to veřejně dostupný adresář a jako jediný má povolený .htaccess soubor, který upravuje práva k přístupu. Nejdůležitější složky jsou app/. Zde najdeme soubor Boostrap.php, ve kterém je třída, která slouží k naštení celého frameworku, knihoven a nastavení aplikace. Další složkou, která pro nás bude důležitá je config, kde se nachází konfigurační soubory pro databázi local.neon a common.neon k přídavným nastavením knihoven (session apod.).

3 MVP

Podobně jako MVC (Model, View, Controller) má Nette (Model, View, Presenter). MVP nám slouží k oddělení logiky webové stránky, kterou vytváříme. Umožňuje nám zpřehlednit web a oddělit jednotlivé operace.
Více informací zde: https://youtu.be/FtTvadnj4KE

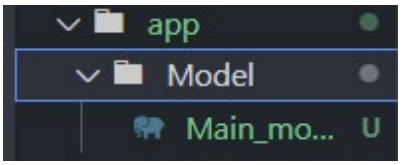
3.1 Model

Jedná se o část MVP, která slouží pro složitější operace s databází, které by nebylo příliš vhodné psát do presenteru, protože by bylo velmi obtížné se v kódu poté vyznat. My se modelem v podstatě v tomto tutoriálu nebudeme příliš zaobírat, protože pro základy, které se budeme učit jej budeme využívat jen velmi zřídka. Nicméně i tak nám docela usnadní práci.

3.1.1 Zprovoznění modelu v projektu

1. Musíme ve složce app vytvořit novou složku s názvem **Model** v této složce poté vytvoříme nový soubor s názvem Main_model.php přesně, jak můžete vidět na obrázku.

Obrázek 1-Model



2. Musíme říct Nette, že tento soubor chceme využívat jako model. Je tedy nutné jej nadefinovat v soboru common.neon ve složce config. Výsledek by měl poté vypadat následovně:

```
parameters:

application:
    errorPresenter: Error
    mapping:
       *: App\*Module\Presenters\*Presenter

session:
    expiration: 14 days
```

```
di:
    export:
        parameters: no
        tags: no

services:
    - App\Router\RouterFactory::createRouter
    - App\Model\Main_model
```

V podstatě pouze v tomto souboru do sevices: vložíme `- App\Model\Main_model` a soubor uložit.

3. Nyní musíme model implementovat do presenteru, abychom jej mohli využívat.

```
<?php

namespace App\Presenters;

use Nette;
use App\Model\Main_model;
use Nette\Database\Context;

final class HomepagePresenter extends Nette\Application\UI\Present
er
{
    private $main_model;

    public function __construct(Main_model $main_model)
    {
        $this->main_model = $main_model;
    }
```

4. Posledním krokem je nalinkovat databázi do Main_model, abychom s ní v něm mohli pracovat. Výsledný model tedy bude vypadat nějak takto:

```
<?php
```

```
namespace App\Model;

use Nette;

class Main_model
{

  public function __construct(Nette\Database\Explorer $database)
  {
    $this->database = $database;
  }
```

Nyní jsme připraveni využívat model naplno.
Podrobnější rozbor zde: https://youtu.be/Q8n3V3_LxMA

3.2 View

Pod pojmem view si můžete představit všechno, co chcete na svém webu zobrazit, jedná se o seskupení šablon a html atributů, obsahuje také tagy jiného jazyka.

3.2.1 Šablona

Nette jako takové funguje na tzv. Latte šablonách, tyto šablony jsou v podstatě totéž, jako typický html dokument, nicméně má své specifické funkce a knihovny. Tyto knihovny a funkce můžete najít na oficiálním API na webu Nette. Latte šablony jsou vždy soubory typu: název_šablony.latte. Velmi důležitou operací je možnost zobrazit nějaký prvek presenteru právě v latte šabloně.

Do šablony se většinou odesílá nějaká komponenta presenteru a ve zbylých případech vkládáte kód, který si nadefinujete přímo v šabloně pomocí latte kódu.

Základní šablonou je v Nette @layout.latte jedná se o šablonu, která je nadřazena všem ostatním. Její asi nejpozoruhodnější vlastností je to, že není statická, tudíž se její kód mění, na základě toho, jakou podřazenou šablonu zrovna zobrazuje. Pojďme si to ukázat na příkladu.

Máme @layout.latte šablonu:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>To do list</title>
     <title>{ifset title}{include title|stripHtml} | {/ifset}Todo-
List</title>
</head>

<body class="body container">
     <div n:foreach="$flashes as $flash" n:class="flash, $flash-
>type">{$flash->message}</div>

     {include content}

     {block scripts}

     {/block}
</body>
</html>
```

Celá šablona zůstává v průběhu všech operací na stránce stejná, nicméně je zde jedna část, kterou můžeme a chceme ovlivnit jednotlivými dílčími šablonami a sice část {include content}. Ve zkratce by se dalo říct, že vše co chcete, aby bylo pro všechny šablony společné dáváte do nadřazeného @layout.latte a naopak, pokud máme něco co bychom chtěli mít pouze pro jednu určitou šablonu, většinou se jedná o kód, který běžně píšeme do body v běžném HTML.

@layout.latte šablona pracuje tak, že {include content} nahradí kódem z konkrétní latte šablony, kterou chceme zobrazit.

Podřazená latte šablona vypadá takto:

```
{block content}
<p>Hello world</p>
{control postForm}

{/block}
```

Všiměte si, že v @layout.latte máme {include content} a v „podšabloně“ máme {block content}. Obsah {block content} poté nahrazuje {include content}.

3.3 **Presenter**

Jedná se o jednu z nejdůležitějších částí projektu, slouží jako prostředník mezi view a modelem, kteří o sobě nic neví. Právě proto je spojuje dohromady a vytvoří celek. Presenter je jen přejmenovaný controller v tradičním rozdělení MVC.

Každá šablona by měla mít ideálně pouze jeden presenter. Tento soubor, by měl poté vypadat nějak takto: NázevPresenter.php.
Je zde také možnost psát operace, které by byly normálně v presenteru do takzvaného modelu.
V takovém případě se budeme jednak lépe orientovat v presenteru a také v jednotlivých databázových operacích, je pouze nutné daný model implementovat do config/local.neon a pak již jen implementovat v presenteru a místo psaní dlouhého kódu, můžeme do presenteru pouze napsat název funkce, která je vytvořena v modelu, právě pro konkrétní databázovou operaci.

4 ZPROVOZNĚNÍ DATABÁZE

zprovoznění databáze v Nette není vůbec nic složitého, jedná se v podstatě pouze o dva jednoduché kroky a poté již můžete s databází směle pracovat.

1. Je potřeba, abyste měli vytvořenou databázi a věděli její název. Tento název je poté nutné zadat do souboru local.neon ve složce config projektu. Výsledek by měl vypadat následovně…

```
database:
    dsn: 'mysql:host=127.0.0.1;dbname=název vaší databáze'
    user: root
    password: *zde vložte heslo k databázi*
```

vysvětlení: dsn: jedná se o propojení se serverem databáze, je zde nutné zadat mysql hosta v drtivé většině případů se bude jednat o 127.0.0.1 neboli localhost. Localhost je loopback adresa používaná koncovým uživatelem čili vámi. Pokud používáte phpMyAdmin ke správě databází, hosta lze zjistit přímo když si otevřeme databázi v horní liště s popisem.

Obrázek 2-phpMyAdmin

Je-li databáze uložena lokálně, mělo by být defaultní nastavení údajů stejné, jako v našem případě. V opačném případě je nutné změnit uživatele (user) a vložit heslo (password), které jsou nastaveny v nastavení přístupu do databáze.

2. Pokud se nám úspěšně povedlo nakonfigurovat local.neon potřebujeme ještě zavolat databázi a to buď v presenteru a nebo modelu projektu. My jako začátečníci se nebudeme zabývat modelem, ale budeme vše vykonávat přes presenter. Upravíme tedy náš homepagePresenter následovně:

```
<?php

namespace App\Presenters;

use Nette;
use Nette\Database\Context;
final class HomepagePresenter extends Nette\Application\UI\Present
er
{
    public $database;

    public function injectContext(Context $database)
    {
        $this->database = $database;
    }
```

Tímto jsme vložili naši databázi do presenteru. Konkrétně jsme ji vložili do proměnné *database*, takže teď již budeme používat jen proměnou, nikoli $this database. ⇨

Těmito dvěma jednoduchými kroky jsme dokázali nastavit Nette tak, aby mělo přístup k databázi a mohli jsme ji dále využívat.

!!!POZOR!!!

Je nutné tento kód zkopírovat vždy na začátek každého nového presenteru, aby bylo možné využívat databáze. Nesmíme ovšem zapomenout na přejmenování třídy na název daného presenteru, který právě využíváme. Pokud tedy mám PostPresenter.php tak název třídy bude následovný:

```
final class PostPresenter extends Nette\Application\UI\Presenter
```

5 ZÁKLADY NETTE – TVORBA PROJEKTU NETTE-MAPY VIDEOTUTORIÁLY

5.1 Úvod + Default presenter

Podrobný video návod zde: https://youtu.be/FXhkf9tR-k0

5.2 Templates, model + ostatní presentery

Podrobný video návod zde: https://youtu.be/6LqVM5C5btk

Kód využitý v projektu naleznete zde: https://github.com/vovyy/nette-mapy.cz-skola.git použité scripty a doplňky projektu Nette-mapy.

5.3 Datatables

Datatables je velmi jednoduchý nástroj pro snadné nasazení, oživení a zvýšení funkčnosti vaší html tabulky. Mezi jejich největší přednosti patří možnost upravit si jednotlivé atributy, které chceme v samotném souboru Datatables.js.
Informace o instalaci a konfiguraci naleznete zde: https://datatables.net/
Konkrétní soubor Datatables.js použitý v projektu naleznete zde: https://github.com/vovyy/nette-mapy.cz-skola.git

5.4 .htaccess

.htaccess je důležitý soubor frameworku, jelikož pomocí něho se určuje přístup k souborům a složkám webové aplikace běžným uživatelům. Tento soubor je nezbytné upravit pro správnou funkčnost a zabezpečení webu. Jeho nesprávné nastavení má dopad především na bezpečnost, ukradení či odcizení aplikace a nastavení na samotném hostingu.
Soubor .htacces naleznete hned v kořenovém adresáři. Po nalezení tohoto souboru jej upravte tak, aby vypadal přesně takto:

```
<IfModule mod_rewrite.c>
RewriteEngine On

RewriteRule ^$ www/ [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_URI} !^/www/
RewriteRule ^(.*)$ www/$1
</IfModule>
```

!!!POZOR!!!

Používejte pouze v případě, že si chcete usnadnit přecházení z jednoho webu do druhého. Každý hosting si to řeší sám. Pokud byste použili tento soubor, učinilo by to pouze více škody než užitku. Hrozí odcizení dat z webové stránky. Pro nás bude stačit spouštění webu na portu 8000.
Spustíme aplikaci příkazový řádek (cmd.exe) ve Windows. Přejdeme do složky s webem a napíšeme:

```
php -S localhost:8000 -t www
```

Tyto .htaccess soubory jsou ve vašem projektu celkem dva. Jeden je jak bylo již zmiňováno v kořenové složce projektu, a druhý ve složce /www. Druhý zmiňovaný soubor ve složce /www vypadá takto:

```
# Apache configuration file (see
https://httpd.apache.org/docs/current/mod/quickreference.html)
Require all granted

# disable directory listing
<IfModule mod_autoindex.c>
    Options -Indexes
</IfModule>

# enable cool URL
<IfModule mod_rewrite.c>
    RewriteEngine On
```

```
    RewriteBase /

    # use HTTPS
    # RewriteCond %{HTTPS} !on
    # RewriteRule .? https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]

    # prevents files starting with dot to be viewed by browser
    RewriteRule /\.|^\.(?!well-known/) - [F]

    # front controller
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule
!\.(pdf|js|mjs|ico|gif|jpg|jpeg|png|webp|svg|css|rar|zip|7z|tar\.g
z|map|eot|ttf|otf|woff|woff2)$ index.php [L]
</IfModule>

# enable gzip compression
<IfModule mod_deflate.c>
    <IfModule mod_filter.c>
        AddOutputFilterByType DEFLATE text/html text/plain
text/xml text/css application/javascript application/json
application/xml image/svg+xml
    </IfModule>
</IfModule>
```

V případě, že by se ve vaší /www složce soubor .htaccess nenacházel, tak jej prostě a jednoduše vytvoříte, jako jakýkoli jiný soubor, zkopírujete do něj kód výše a uložíte jej.

ZÁVĚR

SEZNAM POUŽITÉ LITERATURY

(1) Nette: Začínáme s Nette [online]. [cit. 2021-4-30]. Dostupné z: https://doc.nette.org/cs/3.1/quickstart/getting-started

(2) ITnetwork: Nette sandbox a IDE [online]. [cit. 2021-4-30]. Dostupné z: https://www.itnetwork.cz/php/nette/zaklady/tutorial-nette-php-framework-zaciname

SEZNAM OBRÁZKŮ

Nenalezena položka seznamu obrázků.