Aplikační PHP frameworky

Principy rámcových řešení, existující řešení

Doc. Ing. Radek Burget, Ph.D.
burgetr@fit.vutbr.cz

Aplikační frameworky

- Rámcová řešení pro implementaci serverových aplikací
  - Neplést s klientskými frontend frameworky (v JavaScriptu)!
- Poskytují API (třídy a metody) pro běžné úlohy a situace
  - Zpracování požadavku apod.
- Vynucují pevnou strukturu aplikace
- Tvůrce se může soustředit na tvorbu vlastního řešení
  - Implementace aplikační logiky
  - Uživatelské rozhraní nebo API (volitelně)

Vrstvy aplikace

Nejdůležitější úlohy frameworku

- Zpracování HTTP požadavků
  - Dekódování požadavku a parametrů
  - Volání příslušné aplikační logiky
  - Generování odpovědi
- Tvorba GUI na serveru
  - Snadné generování výstupního HTML (šablony)
  - Znovupoužitelné komponenty (např. formuláře)
- Databázová vrstva
  - Jednotná komunikace s DB serverem
  - Jednotná konfigurace

Další běžné funkce

- Jednotná konfigurace aplikace
- Správa instancí objektů a závislostí
  - Dependency injection
- Autentizace a autorizace

Hlavní PHP frameworky

Populární v ČR:

- Laravel – https://laravel.com/
- Symfony – https://symfony.com/
- Nette – https://nette.org/

Širší přehled:

- např. 10 Popular PHP frameworks in 2020
- (ale podobných žebříčků je mnoho)

Je dobré použít framework?

ANO

(Pro produkční, déle udržovanou aplikaci)

- Přehledná struktura aplikace – udržovatelnost, rozšiřitelnost
- Produktivita
- Bezpečnost (záleží na frameworku)

Architektura frameworku

Jak to uvnitř funguje?

Specifika PHP

- PHP využívá dávkové zpracování
  - Nic z aplikace na serveru trvale neběží
- Celý framework se „startuje“ znovu při příchodu každého HTTP požadavku
  - Lze optimalizovat vyrovnávacími paměťmi apod.
- Na druhou stranu poměrně přímočará implementace

Bootstraping

– viz definice na Wikipedii

- Aktivace funkcionality frameworku při příchodu HTTP požadavku
- Všechny požadavky se směrují na jeden vstupní skript
  - `index.php`, `bootstrap.php`, …
  - Zařízeno nejčasteji konfigurací HTTP serveru (apache, nginx, …)
- Vstupní skript
  - Nastartuje framework (vlastní *bootstraping*)
    - Přečte konfiguraci, vytvoří instance služeb, spojení s databází, atd.
  - Dekóduje požadavek, aktivuje logiku frameworku a vrátí výsledek (HTTP odpověď)

Zpracování HTTP požadavku

(Symfony: The Symfony Application Flow)

Architektura aplikace

- Nejčastěji návrhový vzor MVC
- **Model**
  - Třídy implementující business logiku (operace, zahrnuje práci s databází)
- **View**
  - Generované HTML, šablona naplněné daty
- **Controller**
  - Přiřazený k HTTP požadavku (routing)
  - Volá metody modelu
  - Na základě výsledku vytváří View

Routing

- Mapování URL na controller, který se bude volat
- Konfigurace
  - Anotace u controllerů, speciální konfigurační soubory, nebo PHP

    např. Konfigurace Symfony

    nebo Nette Router
- Konvence
  - Pevně definovaný způsob pojmenování controllerů a mapování URL

    např. Nette SimpleRouter

View (Laravel) – Blade templates

```
<!- View stored in resources/views/greeting.blade.php ->

<html>
    <body>
        <h1>Hello, {{ $name }}</h1>
    </body>
</html>
```

Controller (Laravel)

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\User;

class UserController extends Controller
{
    /**
     * Show the profile for the given user.
     *
```

View (Symfony) – Twig

```
{# template/users/welcome.html.twig #}
<!DOCTYPE html>
<html>
    <head>
        <title>Welcome to Symfony!</title>
    </head>
    <body>
        <h1>{{ page_title }}</h1>

        {% if user.isLoggedIn %}
            Hello {{ user.name }}!
        {% endif %}
```

Controller (Symfony)

```php
// src/Controller/UserController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class UserController extends AbstractController
{
    // …

    public function welcome()
    {
        // the template path is the relative file path from `templates/`
```

View (Nette) – Latte

```
{block content}
    <h1 n:block="title">Můj blog</h1>

    {foreach $posts as $post}
    <div class="post">
        <div class="date">{$post->created_at|date:'F j, Y'}</div>

        <h2>{$post->title}</h2>

        <div>{$post->content}</div>
    </div>
    {/foreach}
```

Presenter (Nette)

```php
namespace App\Presenters;
use Nette;

class HomepagePresenter extends Nette\Application\UI\Presenter
{
    /** @var Nette\Database\Context */
    private $database;

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

Model (Nette)

```php
namespace App\Model;

use Nette;

class ArticleManager
{
    use Nette\SmartObject;

    /** @var Nette\Database\Context */
    private $database;

    public function __construct(Nette\Database\Context $database)
```

Dependency injection

- Framework zabezpečuje vytváření instancí objektů
  - DI container
- Každá část specifikuje požadované služby (objekty)
  - Nejčastěji jako parametru konstruktoru
- Framework vytvoří a dodá instanci
  - Způsob vytvoření je definován v konfiguraci aplikace

Dependency injection (Nette)

```php
namespace App\Presenters;

use Nette;
use App\Model\ArticleManager;


class HomepagePresenter extends Nette\Application\UI\Presenter
{
    /** @var ArticleManager */
    private $articleManager;

    public function __construct(ArticleManager $articleManager)
```

Databázová vrstva

- Jednoduché abstrakce nad SQL
  - Fluent query builder (Laravel)
  - Nette database
- Objektově relační mapování
  - Doctrine ORM (preferované v Symfony)
  - Eloquent (Laravel)

Jak začít?

Instalace a první aplikace

Composer

- Většina frameworků používá Composer
  - Vytvoření a správa projektu
  - Instalace závislostí
- Konfigurace projektu v souboru `composer.json`

Struktura projektu

- Aplikace (`/app`, `/src`, apod.)
  - Implementace modelů, kontrolerů, … (PHP stuff)
- Webové soubory (`/www`, `/public`)
  - Obsah přístupný serveru (statický)
  - `index.php`
  - `.htaccess`
- Konfigurační soubory aplikace
- Dočasné soubory
  - *Adresáře s právem zápisu*
  - `/var/*`, `/temp`, `/log` apod.

Spolupráce s HTTP serverem Apache

- Apache musí mít přístup jen do adresáře s webovými soubory!
- Konfigurace v souboru `.htaccess`
- Nejčastěji přepisování URL pomocí `mod_rewrite`
  - Přesměrování na front controller `index.php`
  - Ošetření výjimek
- Nastavení viditelnosti adresářů apod.

Laravel

- Instalace a první aplikace https://laravel.com/docs/8.x/installation
- Vlastní nástroj `laravel` nebo `composer`
- Demo aplikace
  - Jednoduchá https://github.com/laravel/quickstart-basic
  - RealWorld https://github.com/gothinkster/laravel-realworld-example-app/tree/master/app

Symfony

- Instalace a první aplikace
  https://symfony.com/doc/current/setup.html
- Vlastní nástroj `symfony` nebo `composer`
- Demo aplikace
  - https://github.com/symfony/demo

Nette

- Instalace a první aplikace
  https://doc.nette.org/cs/3.0/quickstart/getting-started
- Používá `composer`
- Demo aplikace
  - https://github.com/nette/tutorial-quickstart/tree/v3.0

A to je vše!

Dotazy?