Udržitelnost WordPress webu

Petr Hlavička

hlavicka.cz

Obsah

- Prémiové šablony a pluginy
- Struktura
- Nástroje
- Verzování
- Deployment
- Testování
- Dokumentace

Co pro mě znamená, že je projekt udržitelný?

Cíl přednášky.

Proč je dobré tu udržitelnost řešit?

Myslet dopředu.

Jaké problémy vidím s premiovými šablonami?

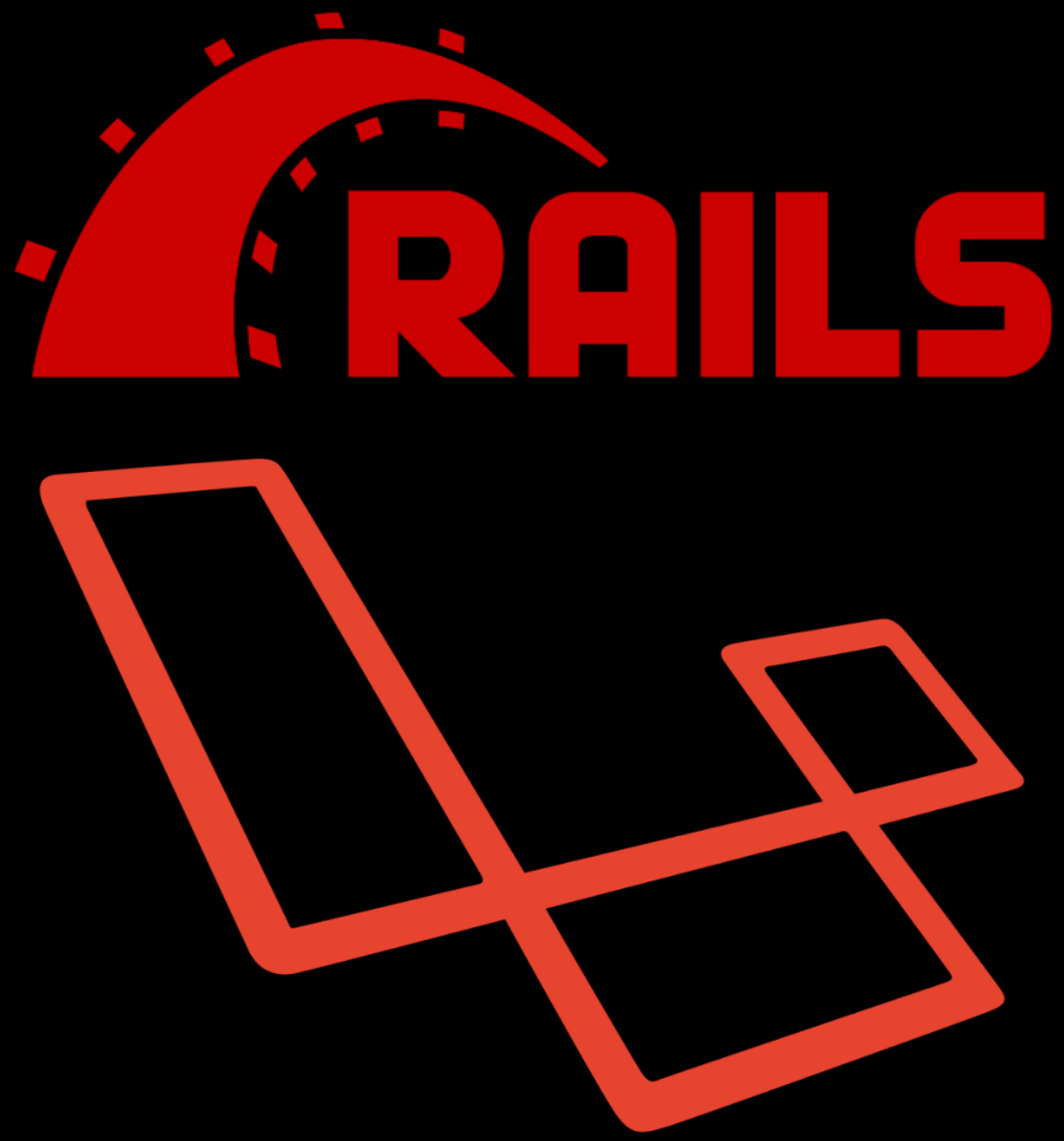
- aktualizace
- co maj "pod kapotou"?
- uzamčení obsahu (*theme/plugin lock-in*)

Nad čím se zamyslet při výběru pluginu?

- Opravdu je potřeba?
- Jak kritický bude pro web?
- Jaká je podpora? Vývoj?
- Je bezpečný? (WPScan Vulnerability Database)

Shrnutí

- Nemáte kontrolu nad vývojem šablony / pluginu.
- ↘ závislostí ↗ kontrola ➡ **udržitelnější**.
- ↘ závislostí ↗ čas ➡ **dražší**.



Webové frameworky

- Struktura
- Nástroje
- Verzování
- Deployment (CI/CD)
- Testování
- Dokumentace

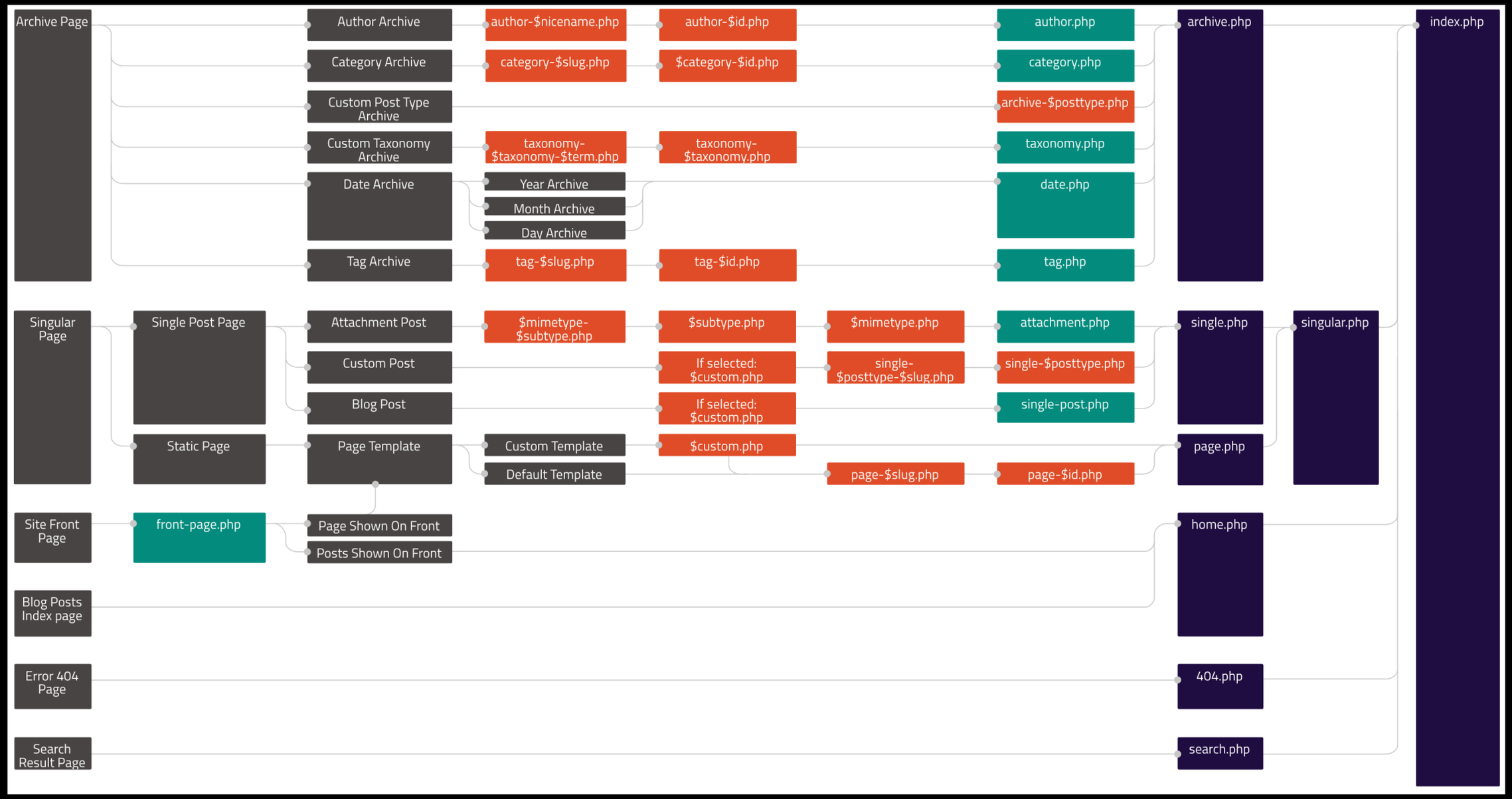
Struktura WordPressu

```
site/
├── index.php
├── license.txt
├── readme.html
├── wp-activate.php
├── wp-admin/
├── wp-blog-header.php
├── wp-comments-post.php
├── wp-config.php
├── wp-config-sample.php
├── wp-content/
│   ├── themes/
│   ├── languages/
│   └── plugins/
├── wp-cron.php
├── wp-includes/
├── wp-links-opml.php
├── wp-load.php
├── wp-login.php
├── wp-mail.php
├── wp-settings.php
├── wp-signup.php
├── wp-trackback.php
└── xmlrpc.php
```

Struktura WordPressu

```
site/
├── index.php
├── license.txt
├── readme.html
├── wp-activate.php
├── wp-admin/
├── wp-blog-header.php
├── wp-comments-post.php
├── wp-config.php
├── wp-config-sample.php
├── wp-content/
│   ├── themes/
│   ├── languages/
│   └── plugins/
├── wp-cron.php
├── wp-includes/
├── wp-links-opml.php
├── wp-load.php
├── wp-login.php
├── wp-mail.php
├── wp-settings.php
├── wp-signup.php
├── wp-trackback.php
└── xmlrpc.php
```

Struktura WordPress šablony



Struktura WordPress šablony

Proč to řešit?

- Povinné soubory šablony:
  - `index.php`
  - `style.css`

Struktura WordPress šablony

Jaké máme možnosti?

- Oficiální (doporučená)[1]
- Použití WordPress frameworku
- Vlastní

[1] Organizing Theme Files | WordPress Developer Resources

Oficiální (doporučená)

```
theme/
├── 404.php
├── archive.php
├── assets/
├── comments.php
├── footer.php
├── front-page.php
├── functions.php
├── header.php
├── inc/
├── index.php
├── page.php
├── README.txt
├── rtl.css
├── screenshot.png
├── search.php
├── searchform.php
├── sidebar.php
├── single.php
├── style.css
└── template-parts/
```

Oficiální (doporučená)

```
theme/
├── 404.php
├── archive.php
├── assets/
├── comments.php
├── footer.php
├── front-page.php
├── functions.php
├── header.php
├── inc/
├── index.php
├── page.php
├── README.txt
├── rtl.css
├── screenshot.png
├── search.php
├── searchform.php
├── sidebar.php
├── single.php
├── style.css
└── template-parts/
```

Použití WordPress frameworku

- Usnadnění vývoje.
- Možný odklon od standardního vývoje šablon.
- Další závislost v projektu.
- Hotová dokumentace.

Ukázka struktury Assely

```
theme/
├── app
├── bootstrap
├── composer.json
├── composer.lock
├── config
├── functions.php
├── gulpfile.js
├── index.php
├── package.json
├── public
├── README.md
├── resources
├── screenshot.png
├── storage
├── style.css
└── vendor
```

Ukázka struktury Assely

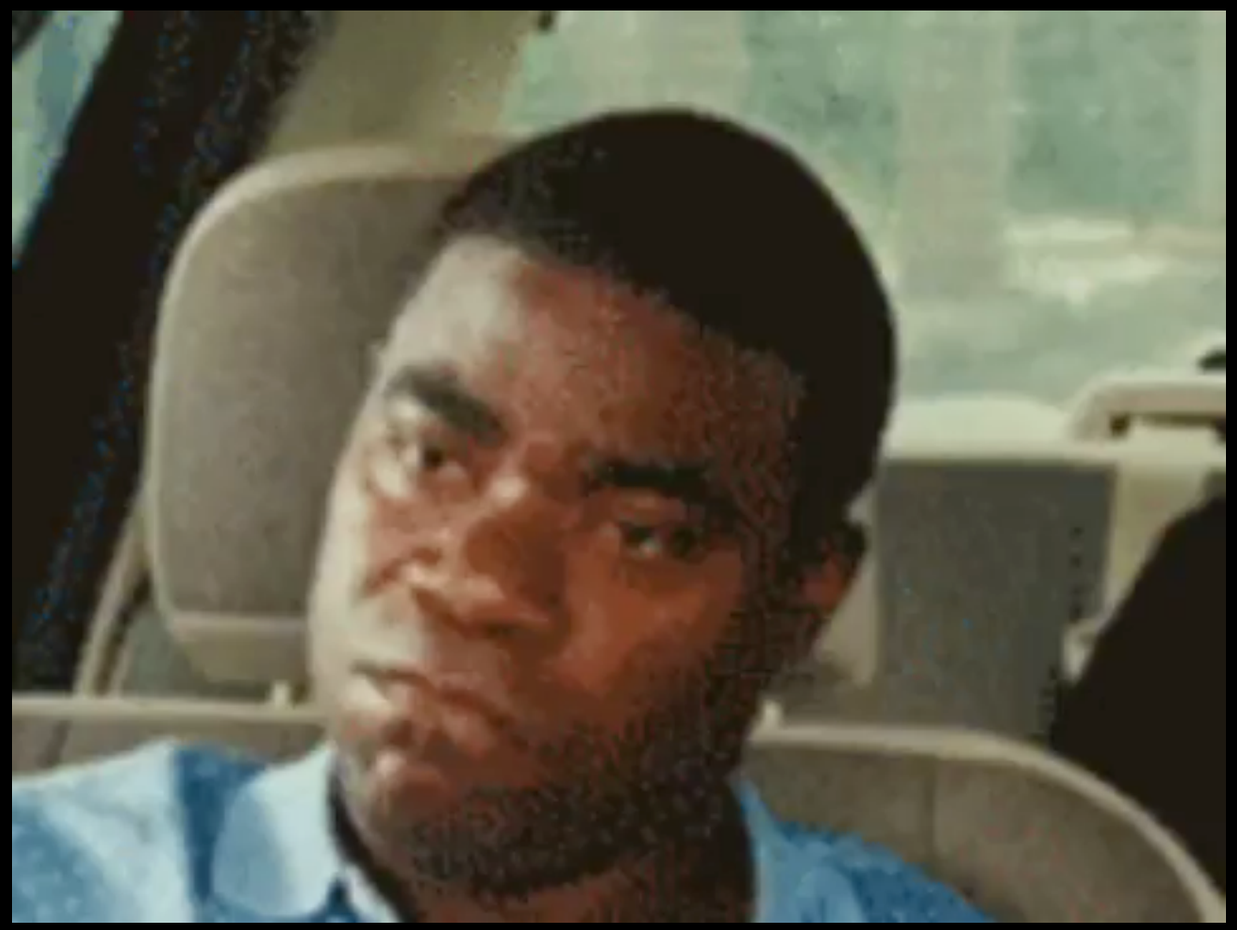
```
theme/
├── app
├── bootstrap
├── composer.json
├── composer.lock
├── config
├── functions.php
├── gulpfile.js
├── index.php
├── package.json
├── public
├── README.md
├── resources
├── screenshot.png
├── storage
├── style.css
└── vendor
```

Použití WordPress frameworku

Doporučení

- Druh implementace.
- Zvážit omezení, které mají.
- Vývoj a podpora frameworku.
- Vyzkoušet!

Nelíbí se ani jedna z variant?



Vlastní struktura
Doporučení

- Samostatný git repozitář.
- S dokumentací.
- Nechte se inspirovat.
- Myslet dopředu.

Struktura WordPress pluginu

- Princip stejný jako u šablon.
- Kde se inspirovat:
  - WordPress Plugin Boilerplate a s tím spojený Plugin Directory Boilerplate.
  - Frameworky: WordPress Plugin Framework, Herbert, WordPress Plugin Template.

Struktura souborů je pouze základ

- Standardizovat si kód
  - Vlastní standardy
  - Cizí standardy např. WordPress Coding Standards + kontrola (WordPress Coding Standards for PHP_CodeSniffer)
- Standardizovat si nastavení editoru
  - Soubor `.editorconfig` (editorconfig.org)

Je vhodné předem informovat klienta o nestandardních komponentách projektu.

Nástroje

- WP-CLI
- GenerateWP
- Plugin Developer + jim doporučené pluginy
- Případné nástroje frameworků.

Lokální vývojové prostředí

- přímo
- hotové řešení
  - WPLib Box, Local (bývalý Pressmatic) či Trellis pro Bedrock.
- Vagrant
  - VCCW
- Docker
  - Local WordPress Development with Docker: 3 Easy Steps & Using wp-cli with Docker

Offline dokumentace

- Dash - macOS, iOS
- Zeal - Windows, Linux
- Velocity - Windows

Verzování

- Nic 2x
- Jen svoji práci
- Vše kritické

GitHub

Jaké jsou možnosti?

- Git submoduly
- Composer
- VersionPress

Git submoduly

- Propojování git repositářů.
- WordPress na GitHub
- WP Plugins na GitHub

Git submoduly

Ukázka struktury

```
site/
├── index.php
├── wp-config.php
├── wp-content/
│   ├── plugins/
│   │   └── wordpress-seo/  # Plugin jako submodul
│   └── themes/
│       └── theme/          # Vlastní šablona
├── wp-cli.yml              # Info pro WP-CLI, kde je WP
└── wp/                     # WordPress jako submodul
```

Git submoduly

Co se výsledně verzuje

```
site/
├── index.php
├── wp-config.php
├── wp-content/
│   ├── plugins/
│   │   └── wordpress-seo/  # Plugin jako submodul
│   └── themes/
│       └── theme/          # Vlastní šablona
├── wp-cli.yml              # Info pro WP-CLI, kde je WP
└── wp/                     # WordPress jako submodul
```

Git submoduly

Výhody

- Jen důležité.
- Verze WP a pluginů.
- Žádný další nástroj.
- Jednoduchý deployment pomocí Git-ftp.

Git submoduly

Nevýhody

- Náročnější správa verzí WP a pluginů.
- Nelze aktualizovat pluginy a ani WP z admina či WP-CLI.
- Klade vyšší nároky na znalost gitu.

Composer

- Balíčkovací systém pro PHP.
- Možné díky WordPress Packagist - WordPress sám o sobě nepodporuje Composer[2].
- Využívá např. Bedrock, WP Starter a další.

[2] #23912 (Add Composer package description) – WordPress Trac

Composer

Ukázka: instalace Bedrocku

1. `composer create-project roots/bedrock`
2. Nastavení DB, WP Salt a další v souboru `.env`.
3. Hotovo.

Composer

Práce s Composerem

- Instalace pluginu: `composer require wpackagist-plugin/wordpress-seo`
- Aktualizace WP a pluginů: `composer update`

Ukázka struktury Bedrocku

```
site/
├── config/
│   ├── environments/
│   │   ├── development.php
│   │   ├── staging.php
│   │   └── production.php
│   └── application.php   # Primary wp-config.php
├── vendor/               # Composer dependencies
└── web/                  # Virtual host document root
    ├── app/              # WordPress content directory
    │   ├── mu-plugins/
    │   ├── plugins/
    │   ├── themes/
    │   └── uploads/
    └── wp/               # WordPress core
```

Composer

Výhody

- `composer.json`

Composer

Nevýhody

- Nutný deployment.
- Vyšší požadavky na hosting.
- Nelze aktualizovat pluginy a ani WP z admina či WP-CLI.

VersionPress

- Trochu jiné verzování, než předchozí varianty.
- Stále ve vývoji.

Další možnost

`.gitignore`[3]

- Zakázání verzování všeho kromě vlastní šablony a podpůrných souborů.
- Neverzuje se verze WP a pluginů.
- WP-CLI, aktualizace přímo, jednoduchý deployment.

---

[3] WordPress .gitignore

Zamyšlení: WP-CLI pro správu verzí WP a pluginů?

- `wp-cli.json`
- `wp plugin install wordpress-seo`, `wp plugin update --all`, `wp core update`, …
- nové příkazy `wp install`, `wp update`
- tak kdyby se někdo nudil… 👍

Možnosti verzování databáze

- DBV - webové rozhraní, umí vytvářet migrace
- MMP - CLI, umí vytvářet migrace
- DBVC - CLI, neumí vytvářet migrace
- VersionPress - pro podporované pluginy

Deployment

- Development -> Staging -> Production.
- Součást CI[4]/CD[5].
- Automaticky.

[4] Continuous integration

[5] Continuous delivery

Deployment

Jak si vybrat?

- Dle požadavků
- Dle rozpočtu
- Dle preferencí

Deployment

Obecné požadavky

- Specifickou strukturu (pro služby nebo nástroje pro WP)
- Repositáře na GitLab/GitHub (pro automatický deploy)
- FTP nebo nejlépe VPS

Deployment: Nástroje

- Jednoduché (nízkonákladové):
  - Git-ftp - development -> production
  - Git na hostingu (některé sdílené mají podporu gitu) - vlastní řešení (`git pull`) nebo VersionPress plugin
- Nástroje pro WP (nutný SSH přístup a další utility na straně serveru):
  - wp-deploy - Capistrano
  - Evolution WordPress - Vagrant, Ansible, Capistrano
  - WP Stack - Capistrano
  - Wordmove - Ruby Gem pro development <=> production
  - DebOps for WordPress - Python, Ansible, DebOps
  - bedrock-capistrano - Capistrano pro Bedrock
  - Trellis - Vagrant, Ansible pro Bedrock

Deployment: Služby

- Univerzální služby:
  - DeployBot
  - DeployHQ
- WP služby:
  - Pantheon
  - WP Engine
  - VersionPress.com
  - Presslabs, Gitium
  - Flywheel

Deployment: Zajímavé

- Deploy pouze šablony či pluginu
  - WP Pusher
  - GitHub Updater

Testování

- Plugin Unit Tests - PHPUnit, WP-CLI
- Codeception pro WP - přes Composer, Integration Tests, WordPress Functional Tests, Acceptance Tests, BDD

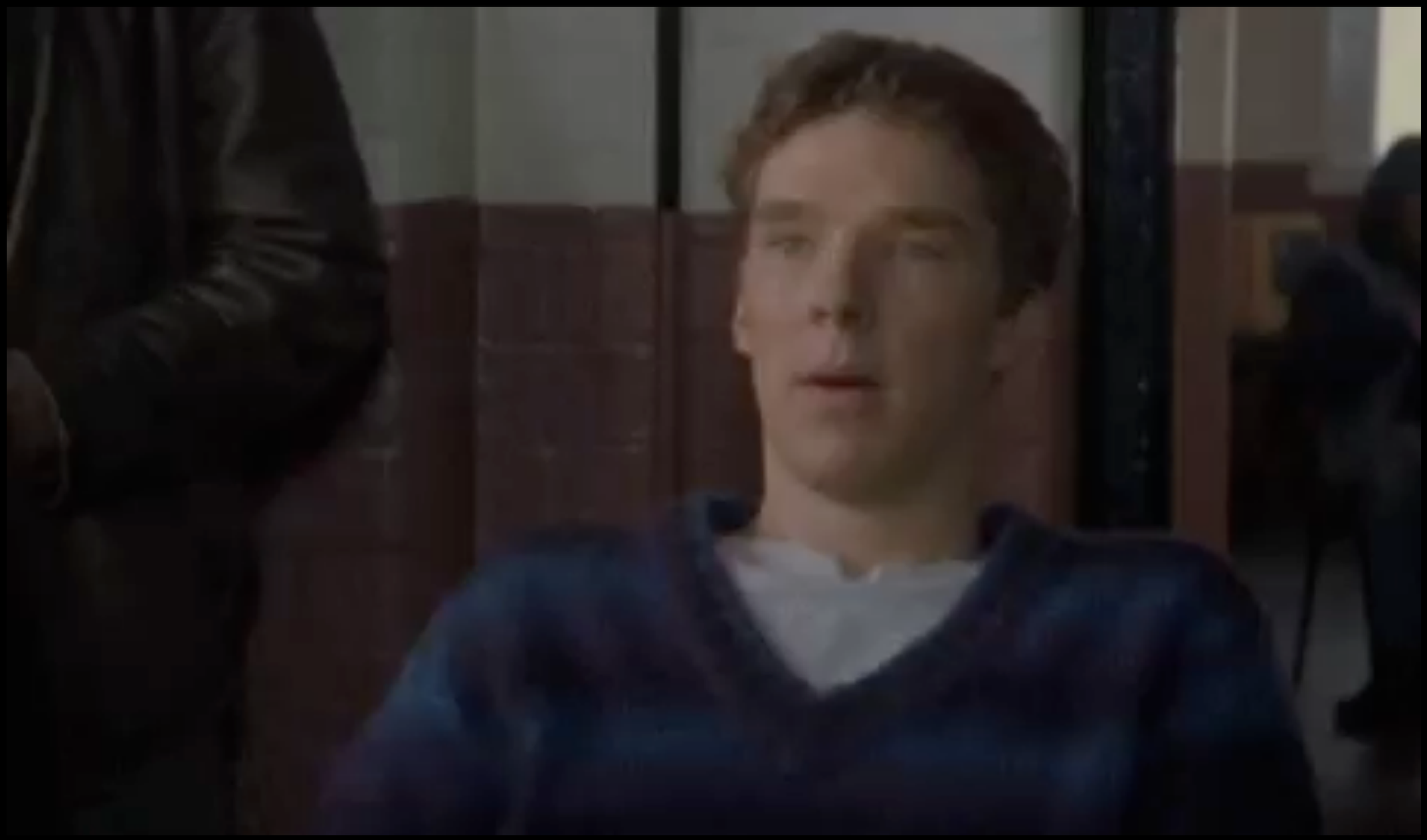
- Zajímavé odkazy:
  - WP Test - kolekce dat pro testování WP
  - Unit Testing WordPress Plugins with PHPUnit
  - Introduction to WordPress Unit Testing

Dokumentace

- Rychlejší orientace v projektu.
- Stručně. Jasně.
- Wiki u GitLab/GitHub.
- Dokumentovat i kód (PHP Documentation Standards).

Shrnutí

Dotazy?



Prezentace bude dostupná na GitHubu. Sledujte @WordCampPraha na Twitteru.

Děkuji za pozornost.

Petr Hlavička

hlavicka.cz