Univerzita Karlova

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Systém pro generování statického webu

System for static web generation

Emil Miler

Vedoucí bakalářské práce: PhDr. Josef Procházka, Ph.D.

Studijní program: Specializace v pedagogice

Studijní obor: Informační technologie se zaměřením na vzdělávání

Praha 2020

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Poděkování.

Název práce: Systém pro generování statického webu

Autor: Emil Miler

Katedra: Katedra informačních technologií a technické výchovy

Vedoucí bakalářské práce: PhDr. Josef Procházka, Ph.D., Katedra informačních technologií a technické výchovy

Abstrakt: Abstrakt.

Klíčová slova: www web generátor

Title: System for static web generation

Author: Emil Miler

Department: Name of the department

Supervisor: PhDr. Josef Procházka, Ph.D.,

Abstract: Abstract.

Keywords: www web generator

Obsah

Úvod

1. Staticky generovaný web

Princip statické webové stránky sahá ke až vzniku WWW, kdy existovaly pouze stránky statické, tedy stejné pro každého uživatele. Jejich obsah může být průběžně aktualizován, ovšem negenerují se zvlášť pro každého uživatele na základě různých proměnných. U statických webů tedy dochází k vytvoření čistého HTML ve chvíli, kdy je změněn zdrojový obsah, nebo kdy autor ručně spustí generátor. Magazine (2020)

Dynamické stránky jsou generovány speciálně pro každého uživatele na základě jeho nastavení, různých vstpů, proměnných a dalších vlastností. Ke generování dochází ve chvíli, kdy si uživatel stránku vyžádá, nikoliv předem, jako je tomu u staticky generovaných stránek. Magazine (2017)

1.1 Výhody statických webových stránek

Pro sdílení statického obsahu mezi různé uživatele stačí velmi jednoduchý HTTP server bez jakýchkoliv dalších modulů typu *PHP*, *Pythnon* a dalších systémů, které by obsah dynamicky generovaly například z dat vytažených z databáze, nebo z uživatelského vstupu. Na straně serveru tedy nedochází ke zpracování obsahu těsně před jeho odesláním uživateli, čímž se v komunikaci mezi klientem a serverem se drasticky snižuje „Time To Firt Byte“[1] a tím dochází ke snížení celkové latence. Hoffman (2013-09-26)

Snížení samotné latence může pozitivně přispět ke spokojenosti uživatelů, což dokazuje nespočet výzkumů na toto téma, například analýza z webového portálu Financial Times, kde se uvádí, že rychlost webové stránky negativně ovlivňuje hloubku jejího užívání, ať už je odezva sebemenší. Jak je zde rovněž uvedeno, data ukazují, že pohledu uživatelské spokojenosti a finančního dopadu existují jasné a důležité výhody při zrychlení webové stránky. Z tohoto výzkumu se autoři rozhodli v následujících měsících od vydání analýzy zainvestovat více času do úprav všech aspektů jejich nové stránky FT.com s cílem jejího zrychlení.

Nechat opravit překlad

Eliminováním dynamického obsahu se také předchází nevyžádaným vstupům od uživa-

[1] Time To First Byte — čas mezi odesláním požadavku a přijmutím prvního bajtu dat.

tele, které mohou být i cílené na prolomení bezpečnostních nedostatků webové aplikace a v některých případech mohou vést k úniku citlivých dat, převzetí kontroly útočníka nad webovou aplikací nebo celým serverem, podstrčení falešných dat uživateli a mnoho dalším běžně se stávajícím útokům. Statický web eliminuje tento problém, jelikož nemá žádný uživatelský vstup.

Sledování a analýze nejčastějších chyb webových aplikací a serverů se věnuje organizace OWASP[2], která vydává aktualizované seznamy a statistiky. Podle OWASP byly v roce 2017 nejčastější tyto chyby a bezpečnostní nedostatky:

1. Injekce
2. Rozbitá autentizace
3. Odhalení citlivých dat
4. XML External Entities (XXE)
5. Nefunkční řízení přístupu
6. Špatná konfigurace zabezpečení
7. Cross-Site Scripting (XSS)
8. Nezabezpečná deserializace
9. Uživání komponent se známými zranitelnostmi
10. Nedostatečné logování a monitorování

OWASP (2017)

Většina těchto chyb se vztahuje právě k dynamickým webovým aplikacím. Bezpečnost tedy závisí nejen na programátorovi který aplikaci vytváří, ale také na tom, že programovací jazyk je bezpečně implementován. To nelze tvrdit o nejpoužívanějším jazyce PHP, který nejen že obsahuje spousty chyb, ale zároveň nevede programátora ke psaní bezpečného kódu a ve výsledku vzniká opravdu děravá aplikace, pokud si

doložit zdroj - seznam bezpečnostních záplat, analýza, ...

[2]The Open Web Application Security Project — `https://owasp.org/`.

autor programu nedá pozor na správné ošetření vstupů a další bezpečnostní aspekty programu.

Údržba velkých webových aplikací je také často problematická. Kód je nutné udržovat v návaznosti na aktualizace daného jazyka, databázového systému a dalších věcí. Těmto aktualizacím se z bezpečnostních důvodů nelze vyhýbat. Statický web nemusí udržovat funkční propojení s databázemi a různými frameworky a je tedy mnohem méně náročný na dlouhodobou údržbu. Při zvolení správného generátoru není nutná ani údržba šablon a celý systém při zachování stejného prostředí nepřestane fungovat. Protože statický generátor nepracuje s uživatelským vstupem, vyhýbá se bezpečnostním chybám a tím i nutným aktualizacím.

Lepši slovo?

Zdroj?

Jako každý jiný systém, i tento má nevýhody. Hlavním z problémů je to, že správa statického generátoru a tvorba obsahu je náročnější, než klasické webové rozhraní s administračním panelem, různými uživateli a jednoduchou správou pro běžné technicky nenadané uživatele. Pro přidání nebo úpravu obsahu je nutné pracovat s lokálními soubory ve stromové struktuře a při generování je často potřebný zásah do shellu[3]. Tvorba systému pro automatizované generování je také náročnější než instalace některého z běžných CMS[4]. Cimpanu (2015)

1.2 Princip generátorů

Generátor statického obsahu je tvořen ze tří hlavních částí. První částí jsou soubory šablon, které popisují rozložení stránky, vizuální vlastnosti, typografii, ale také vstupní a výstupní kódování a formáty. V podstatě definují jak a kam se bude obsah vkládat. Druhou částí je obsah obecně psaný v některém ze značkovacích jazyků, nejčastěji v jazyce Markdown. Obsah je strukturován do vlastních sekcí a souborů, aby bylo snadné rozlišit, do které části výsledné stránky patří. Třetí a poslední částí je samotné jádro generátoru, které zpracovává obsah, vkládá ho do šablon a renderuje statickou webovou stránku.

Většina generátorů zároveň umí pracovat s konfiguračními soubory, kterými jde na-

[3]Program pro interpretování příkazů v prostředí příkazové řádky.

[4]Content Management System

stavit chování generátoru na jednom centralizovaném místě. Část z nich má také integrovaný jednoduchý web server, který umožňuje autorovi náhled výstupních stránek během tvorby obsahu.

Cimpanu (2015)

2. Webová paradigmata

Ve světě webových stránek se setkáváme se spoustou forem a paradigmat, která se hodí pro obsažení různých druhů informací. Neexistují žádné formální zařazení druhů webových stránek do skupin, ovšem některé webové portály se pokouší určit základní druhy webů, které se na Internetu objevují. Na základě těchto portálů a jejich rozřazení do skupin[1][2][3], které jsou často mířené na specifický obsah, lze vytvořit tři základní paradigmata, do kterých lze tyto weby zařadit. Jsou jimi:

Takhle je to v pořádku?

- Webová prezentace
- Index všeobecných informací
- Technická dokumentace
- Sociální sítě a fóra

V této práci byl ke každému z paradigmat vybrán systém vhodný pro generování a správu daného druhu obsahu. Výjimkou je skupina sociálních sítí a fór, kde staticky generovaný obsah není z důvodu často se měnícího obsahu vhodným řešením.

2.1 Webová prezentace

Nejbližší původním webům z dob vzniku WWW jsou webové prezentace, tedy stránky s jednoduchým obsahem, které slouží k předání informací čtenáři například formou článků nebo galerie. Do této skupiny lze zařadit portfolia, blog, online noviny a časopisy, firemní stránky, foto alba a podobně. Tento druh stránek se skvěle hodí ke statickému generování obsahu, který se odesílá všem uživatelům stejný a nemění se často.

Opravit slovosled

Jako nejvhodnější systém pro generování webových prezentací byl vybrán software Zola. Ten je oproti jiným systémům výhodný tím, že je napsaný v jazyce Rust a je tedy rychlejší, než všichni jeho soupeři. Také ho jde zkompilovat do jednoho staticky

Citovat benchmark

[1] http://www.xislegraphix.com/website-types.html
[2] https://www.hostgator.com/blog/popular-types-websites-create
[3] https://www.quora.com/What-are-the-different-types-of-websites

linkovaného binárního souboru, se kterým se pracuje mnohem lépe, než se složitým frameworkem.

2.2 Index všeobecných informací

Za obecného zástupce tohoto druhu stránek lze považovat Wikipiedii, která podnítila vznik spousty jiných takzvaných „Wiki systémů“ a stránek.

2.3 Technická dokumentace

Na rozdíl od Wiki stránek se technická dokumentace liší organizováním svého obsahu, který je cílený na přesný popis systému či objektu.

3. Značkovací jazyky pro popis obsahu

3.1 Principy značkovacích jazyků

Vysvětlení principu značkovacích jazyků, nebo také takzvaně „makrup jazyků“, můžeme najít například v RFC 7764[1], tedy že v počítačových systémech jsou kontextuální data ukládána a zpracována několika technikami. Informaci lze kódovat jako čistý text bez speciálních formátovacích znaků. Tento přístup je jednoduchý pro implementaci i použití, ovšem neumožňuje složitější formátování textu.

Kódovat lze můžeme i do binárních dat určených ke zpracování a interpretaci specializovaným programem. Zřejmou nevýhodou je to, že zdroj není čitelný bez programu určeného pro jeho interpretaci.

Markup jazyky se snaží o spojení toho nejlepšího z obou světů, tedy o obsah čitelný v čistém textu s možností formátování. To je dosaženo tím, že běžným znakům jsou přiděleny speciální významy nedefinované původní znakovou sadou. Uživatel je schopen tyto znaky psát jako čistý text a vyjádřit tím speciální význam. Například v rámci jazyka Markdown se znak `#` změní z běžného křížku na definování nadpisu první úrovně, nebo také kombinace znaků `<p>` značí začátek odstavce v HTML. Leonard (2016)

3.2 Nejběžnější jazyky

Ke dnešnímu dni vnikl nespočet značkovacích jazyků. Nejpoužívanějším z nich jednoznačně HTML, ovšem tato práce se věnuje těm nejpoužívanějším jazykům, které mají uživateli usnadnit psaní a sázení obsahu. Uživatel tedy nemusí nutně řešit typografii a formátování obsahu při jeho psaní, tedy o věci, o které se později stará generátor pomocí šablon. U HTML je tomu naopak, kdy uživatel řeší samotný obsah i formátování v jednu chvíli skrze různé druhy formátovacích tagů. O vyplňování obsahu do HTML

[1] Jako *RFC* se označují standardy vydané organizací IETF (Internet Engineering Task Force).

se v případě staticky generovaných webů stará právě samotný generátor.

Vybrané jazyky jsou zároveň cílené na čitelnost samotného zdrojového obsahu v čistém textu bez nutnosti jeho interpretace speciálním prostředím či zpracováním do jiného formátu, například do PDF, DjVu, PostScript apod. Například podtržení textu je v nějakém pseudo-jazyce reprezentováno opravdovým podtržením pomocí spojovníků, nikoliv obalením nadpisu ve speciální deklaraci, jako je tomu například u HTML. Podtržení je poté pro čtenáře mnohem jasnější, jelikož nemusí přemýšlet, co v případě HTML daný tag vůbec způsobuje, ale podtržený vyplývá z kontextu.

Seznam nejoblíbenějších jazyků je sestaven podle aktuálních statistik ze serveru Slant, který se věnuje obecnému určení oblíbenosti na základě hodnocení ze strany uživatelů. Slant (2020)

3.2.1 Markdown

Vznik jazyka Markdown byl 14. prosince roku 2014, když John Gruber vydal jeho první popis syntaxe a referenční implementaci.

Hlavním z cílů syntaxe jazyka je vytvářet co možná nejčitelnější obsah v syrové podobě. Dokument psaný v Markdownu by měl být publikovatelný sám o sobě jako čistý text bez dalších úprav a zpracování. Jazyk byl ovlivněn několika již existujícími specifikacemi jiných jazyků, ovšem největším zdrojem inspirace pro jeho vznik jsou čisté emailové korespondence. Gruber (2004-12-17)

První specifikaci Gruber vydal společně s referenční implementací v jazyce Perl, která slouží pro konverzi Markdownu do HTML. Program také nese stejný název „Markdown“, ovšem mluvíme-li o „Markdownu“, máme nejčastěji na mysli samotnou syntaxi. Ta je dnes již implementována v mnoha různých jazycích a programech. Gruberova specifikace ovšem není formální standard, kvůli čemuž vznikl veliký počet alternativních a více či méně pozměněných implementací, které nemusí být navzájem kompatibilní. Nejčastějšími z nich jsou například Github Markdown, CommonMark, R Markdown a mnoho dalších.

Citace

3.2.2 Org-mode

... Schulte a kol. (2012) The Org Mode Developers (2020)

3.2.3 AsciiDoc

...

3.2.4 reStructuredText

...

3.2.5 TEX

Tento jazyk se již vzdaluje od původního konceptu čitelnosti zdroje, ovšem ve statických generátorech ho lze stále efektivně využít a jeho části se velmi často objevují jako rozšíření dříve zmíněných jazyků. Jedním z hlavních rozšíření jsou zápisy matematických rovnic, které z TEXu vychází.

Rozšířit o popis TeXu a matiky.

Většina uživatelů se setkala spíše s jazykem LATEX, tedy s nadstavbou původního TEXu, která má uživateli zjednodušit práci svými makry a rozšířeními. Realita je ovšem taková, že LATEX dělá celou práci složitější, jak popisuje doktor Olšák:

> Představte si, že si nějaký uživatel přečte LATEXovou příručku a nabyde dojmu, že mu bude stačit rozumět problematice sazby na úrovni této příručky. Pak se jednou překlepne třeba při sestavování tabulky a na terminálu na něj TEX křičí: `Extra alignment tab has been changed to "\cr".` Uživatel začne znovu listovat ve své příručce a zjistí, že tam o žádném "`\cr`"není jediná zmínka. Má pak tři možnosti: (1) Zmáčkne Enter a podobně se zachová i u dalších chyb. Pomyslí si, že ten LATEX je něco tajemného a mystického. (2) Propadne zoufalství a jde od toho. Dojde k závěru, že je lepší zůstat u Wordu. Vždyť stačí vzít tabulku v Excelu a jednoduše ji přemístit do Wordu a jaképak smolení se s nějakým podezřelým "`\cr`".

(3) Pořídí si TEXbook a po intenzivním studiu nakonec řekne: „aha“. V tuto chvíli ale už nepotřebuje, aby mu LATEX zakrýval složitost TEXu.

Olšák (1997)

Je tedy lepší použít samotný TEX.

3.2.6 Troff

4. Taxonomie požadavků

4.1 Obecná kritéria

4.2 Kritéria specifická pro modelový web

4.3 Kritéria pro šablony a design

5. Modelová implementace

5.1 Požadavky na modelový web

K této práci byl jako modelová implementace zvolen web pro distribuci výukových materiálů. Webové stránky byly objednány Ústavem výzkumu a rozvoje vzdělávání Pedagogické fakulty Univerzity Karlovy za účelem usnadnění práce již aktivních učitelů a jsou tedy plně využívány v praxi mnoha učiteli z celé republiky. Materiály jsou určeny pro učitele během vyhlášeného stavu nouze v době šíření viru COVID-19 a mají učitelům pomoci s přípravou distanční výuky a úkolů pro studenty během karantény. Tuto implementaci lze ovšem použít na distribuci jakýchkoliv jiných výukových materiálů.

Hlavním požadavkem je možnost dělit obsah na sekce dle druhu školy (základní škola, střední škola, vysoká škola atd.) a dále pak na subsekce podle předmětů a oborů. Obsah každé stránky s obsahem je třeba dělit na sekci odkazů, sekci se soubory a sekci s videi. Všechny tyto soubory musí být distribuovatelné přímo z webových stránek, nikoliv s externích zdrojů. Všechna videa je nutné vložit do stránky a musí je být možné přehrát přímo v nativním přehrávači prohlížeče bez nutnosti otevírání externích webů.

Stránky musí být staticky generované a není tedy žádoucí v rámci webu řešit uživatelské účty, přihlašování apod. Zároveň je důležité, aby byl obsah zobrazitelný na každém druhu zařízení, tedy jak na monitorech s nadstandardní velikostí, tak na mobilních zařízeních. Z důvodu potencionálního vytížení sítě je nutné, aby byl celý obsah optimalizován za účelem předejití vysoké latence, a to z důvodů probíraných v předchozích částech práce.

5.2 Výběr vhodného systému

Pro správu obsahu i šablon a statických souborů byl zvolen systém Git. Hlavní výhodou tohoto verzovacího systému ...

5.3 Tvorba šablony

5.4 Rozšíření šablony

Ve výchozím stavu generátor neumí vklládat nic jiného, než je uvedeno ve specifikaci CommonMark[1]. Dle požadavků modelového webu je nutné, aby generátor uměl vkládat videa přímo do stránky. Taková funkce není soušástí specifikace CommonMark a je tedy potřeba rozšířit funkcionalitu generátoru. Nejvhodnějším způsobem přidání vlastní funkce je využití vlastních filtrů, které se v rámci generátoru nazývají „shortcode“.

Principem vlastních filtrů je to, že uživatel si vytvoří vlastní šablonu, kterou lze vyvolat speciální řídící sekvencí přímo z obsahu. Každý tento shortcode může pracovat s libovolným množstvím proměnných a po zpracování vloží do místa vyvolání zkompilovaný HTML kód. Lze tedy tvrdit, že shortcode je v své podstatě imperativní funkce, která umí pracovat s parametry.

Pro tvorbu těchto filtrů je v generátoru Zola určena složka `templates/shortcodes`, která obsahuje jejich HTML šablony. Název HTML souboru definuje název vlastního filtru. Vytvoříme-li soubor `templates/shortcodes/video.html`, budeme schopni využívat vlastní filtr s názvem `video`.

V následujícím příkladu bude filtr očekávat atribut `src` a bude vracet jednoduchý HTML kód pro vložení videa do stránky.

```
<video controls><source src="{{ src }}"></video>
```

Tento filtr lze vyvolat kdekoliv v obsahu, tedy v kterémkoliv souboru s koncovkou `.md`.

```
{{ video(src="video.webm") }}
```

Výstupem této direktivy bude následující HTML kód.

```
<video controls><source src="video.webm"></video>
```

Součástí požadavků pro modelový web jsou i citace přiložených souborů a videí. Exis-

[1] `https://commonmark.org/`

tující filtr je tedy třeba rozšířit o možnost přiložení různých metadat. Tato metadata ovšem nejsou pro vložení videa povinná. Ve specifikaci vlastních filtrů lze využívat všechny operátory, které generátor nabízí. Nejlepším přístupem k tomuto problému je tedy využití jednoduchých podmínek, které kontrolují, zda je každá z hodnot zadána jako parametr a v případě že ano, vepíše se do obsahu. V následujícím příkladu jsou přidány tři podmínky pro kontrolu a případné vložení, jimiž jsou název videa (`title`), jméno autora (`author`) a datum vytvoření (`date`).

```
{% if title %}
    <div class="title">{{ title }}</div>
{% endif %}
    <video controls><source src="{{ src }}"></video>
{% if author %}
    <div class="metadata">{{ author }}</div>
{% endif %}
{% if date %}
    <div class="metadata">{{ date }}</div>
{% endif %}
```

Pouze atribut `src` je podle tohoto filtru povinný. Filtr lze opět vyvolat pomocí stejné direktivy kdekoliv v obsahu, ovšem nyní lze libovolně přidat parametry pro metadata.

```
{{ video(
    src="video.webm",
    title="Nazev videa",
    author="Jmeno autora",
    date="2020-03-22"
) }}
```

Opravit háčky a čárky v blocích kódu.

Výtupem toho filtru bude tedy následující HTML.

```
<div class="title">Nazev videa</div>
<video controls><source src="video.webm"></video>
<div class="metadata">Jmeno autora</div>
<div class="metadata">2020-03-22</div>
```

5.5 Optimalizace

5.6 Požadavky na rozšíření

6. Vyhodnocení modelové implementace

6.1 Návrhy pro rozříšení systému

6.2 Implementace rozšíření

Závěr

Seznam použité literatury

Cimpanu, C. (2015). How static site generators work. https://web.archive.org/web/20200316165614/https://news.softpedia.com/news/How-Static-Site-Generators-Work-482007.shtml. Cit. 2020-03-16.

Gruber, J. (2004-12-17). Markdown. https://web.archive.org/web/20200227143926/https://daringfireball.net/projects/markdown/. Cit. 2020-02-27.

Hoffman, B. (2013-09-26). Improving search rank by optimizing your time to first byte. https://web.archive.org/web/20190416124447/https://moz.com/blog/improving-search-rank-by-optimizing-your-time-to-first-byte. Cit. 2020-02-12.

Leonard, S. (2016). Guidance on markdown: Design philosophies, stability strategies, and select registrations. RFC 7764. URL https://tools.ietf.org/html/rfc7764.

Magazine, P. (2017). Definition of: dynamic web page. https://web.archive.org/web/20170117040526/https://www.pcmag.com/encyclopedia/term/42199/dynamic-web-page. Cit. 2020-02-12.

Magazine, P. (2020). Definition of: static web page. https://web.archive.org/web/20200223095514/https://www.pcmag.com/encyclopedia/term/static-web-page. Cit. 2020-02-12.

Olšák, P. (1997). Proč nerad používám latex. http://petr.olsak.net/ftp/olsak/bulletin/nolatex.pdf.

OWASP (2017). Owasp top ten 2017. Technical report.

Schulte, E., Davison, D., Dye, T. a Dominik, C. (2012). A multi-language computing environment for literate programming and reproducible research. *Journal of Statistical Software*, **46**(3), 1–24. ISSN 1548-7660. URL http://www.jstatsoft.org/v46/i03.

Slant (2020). What are the best markup languages? https://web.archive.org/web/20200210061112/https://www.slant.co/topics/589/~best-markup-languages. Cit. 2020-02-10.

The Org Mode Developers (2020). *The Org Manual*.

A. Přílohy

A.1 První příloha