VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PŘEKLAD DOTVVM STRÁNEK DO ACCELERATED MOBILE PAGES

DOTVVM TO ACCELERATED MOBILE PAGES TRANSPILER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL TICHÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN PLUSKAL**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Tichý Michal**
Program: Informační technologie
Název: **Překlad DotVVM stránek do Accelerated Mobile Pages**
DotVVM to Accelerated Mobile Pages Transpiler
Kategorie: Web

Zadání:

1. Seznamte se s technologií AMP (accelerated mobile pages) a DotVVM.
2. Prozkoumejte, které prvky/funkce DotVVM mohou být přeloženy do AMP. Vytvořte testovací sadu stránek, která je bude demonstrovat.
3. Navrhněte způsob, jak překlad do AMP integrovat s DotVVM.
4. Implementujte prototyp překladu DotVVM stránek do AMP.
5. Zhodnoťte úspěšnost překladu za použití testovací sady stránek i reálných webových aplikací. Diskutujte budoucí možná rozšíření.

Literatura:

- O'Donoghue, R. (2017). AMP: Building Accelerated Mobile Pages: Create lightning-fast mobile pages by leveraging AMP technology. Packt Publishing Ltd.
- Richter, J. (2012). *CLR via C#*. Pearson Education.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Pluskal Jan, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Tato práce přidává to webového .NET frameworku DotVVM možnost převodu DotVVM stránek do technologie AMP. V rámci tohoto textu je také obsažen přehled aktuálního stavu obou technologií. Práce obsahuje souhrn výhod a nevýhod obou technologií a náhled do jejich fungování. Druhá polovina práce popisuje implantaci knihovny DotVVM.AMP. V závěru práce je popsán postup integrace tohoto překladu do stávajícího DotVVM aplikace.

Abstract

This thesis proposes an extension to the DotVVM web framework that adds support for conversion of DotVVM pages into AMP pages. This thesis describes the current status and inner workings of DotVVM and AMP. The thesis contains an overview of the advantages and disadvantages of both technologies. Second part of this thesis describes implementation of DotVVM.AMP. Final part of this thesis shows integration of DotVVM.AMP into an existing DotVVM application.

Klíčová slova

webové aplikace, AMP, DotVVM, .NET Framework

Keywords

web applications, AMP, DotVVM, .NET Framework

Citace

TICHÝ, Michal. *Překlad DotVVM stránek do Accelerated Mobile Pages*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal

Překlad DotVVM stránek do Accelerated Mobile Pages

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Pluskala. Další informace mi poskytli členové DotVVM týmu. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Tichý

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Janu Pluskalovi a také Stanislavu Lukešovi, Tomáši Hercegovi a dalším členům DotVVM týmu, kteří mě nasměrovali správným směrem.

Obsah

Kapitola 1

Úvod

Svět webových technologií se neustále vyvíjí. Rok za rokem stoupá výkon klientských zařízení a spolu s ním i velikost a komplexita webových aplikací. Co se ale bohužel zlepšuje jen velmi pozvolna, je rychlost webových aplikací. Problém nízké rychlosti se navíc postupně zvětšuje kvůli zvyšujícímu se zastoupení uživatelů, kteří pro přístup k webovým aplikacím používají mobilní zařízení s malým výkonem a nepříliš dobrou konektivitou.

V posledních letech se stále častěji můžeme setkat s webovými stránkami, které používají open source technologii Accelerated Mobile Pages neboli zkráceně AMP, která se snaží problém rychlosti webových aplikací řešit pomocí omezení složitosti, optimalizací vykreslování (díky fixnímu rozložení), přednačítání webové stránky a dalších optimalizací v rámci AMP distribuční infrastruktury.

Výsledkem všech těchto optimalizací jsou webové stránky, jejichž čas do první interakce[1] je v nízkých desetinách sekundy. Pro porovnání v roce 2018 průměrně trvalo načtení webové stránky 8.66 s [19]. Udržení rychlosti načítání stránky na takto nízkých hranicích ovlivňuje, jak se uživatel po příchodu na naši stránku bude chovat (a zda si nerozmyslí naši stránku vůbec nenavštívit).

Zvýšená rychlost AMP stránek má kromě přímých efektů na uživatele také vedlejší efekt na pozici v rámci výsledků vyhledávání a tím i na větší počet návštěvníků.

Jako každá jiná technologie tak i AMP technologie není všemocná a pro její použití musí vývojáři obětovat většinu interaktivity pomocí javascriptu, značně omezit velikost kaskádových stylů a zcela vynechat některé zakázané HTML tagy. Kvůli těmto omezením jsou cílovými použitími pro AMP jednoduché statické stránky, popřípadě značně zjednodušené verze stránek plnohodnotných.

Cílem této práce je přinést výše zmiňované výhody technologie AMP uživatelům open source webového frameworku DotVVM založeného na platformě ASP.NET. Přestože cílovým použitím DotVVM jsou složité interaktivní webové aplikace (tedy opak stránek ideálních pro přepis do AMP), tak i v rámci těchto aplikací často nalezneme stránky, které jsou z velké části statické, popřípadě se dynamicky skládají na serveru, ale ne už u klienta [15]. Příkladem takových stránek můžou být úvodní a další veřejné/informační stránky u jinak dynamických webových aplikací.

[1] Čas do první interakce: anglicky se tato metrika nazývá Time to Interactive a vyjadřuje, jak dlouho trvá, než se stránka načte do stavu, kdy je uživatel schopný s ní bez omezení interagovat.

Kapitola 2

Co to je AMP?

Technologie AMP má za cíl zajistit uživatelům velmi rychlé webové stránky nezávisle na tom, kde se uživatelé nacházejí a jaké zařízení používají.

AMP byl původně vydán v roce 2015 společností Google (ve spolupráci s firmou Twitter) pod názvem Accelerated Mobile Pages jako odpověď na technologii Facebooku s názvem Instant Articles. Na rozdíl od Instant Articles se AMP, díky tomu že je open-source, rozšířil mimo platformu zakládající organizace a nyní existují více než dvě miliardy AMP stránek [23].

Amp zajišťuje slibované zvýšení rychlosti pomocí kombinace dvou přístupů. Redukce komplexity webových stránek, která je vynucovaná mnohými restrikcemi technologie AMP, a přiblížení obsahu uživatelům pomocí distribuovaných AMP Cache a dalších součásti AMP infrastruktury.AMP infrastruktura je popsána na konci části 2.2. Kromě společnosti Google dnes poskytují kompletní infrastrukturu pro AMP stránky také společnosti Microsoft a CloudFlare.

2.1 Nové typy obsahu, které AMP přináší

Kromě celých webových stránek AMP podporuje i další nové typy obsahu. Tyto nové formáty přináší novou funkcionalitu a zvýšení rychlosti do domén jako je například email nebo reklama. Žádný z těchto nových formátů se bohužel prozatím nerozšířil mimo Google platformu. Uživatelé se nejčastěji mohou setkat s AMP adds, ale to jen z toho důvodu, že Google AdSense[1] je hlavní provozovatel reklamních služeb na internetu.

AMP Email

Amp Email umožňuje používat v rámci emailů AMP komponenty. Díky tomu mohou emaily obsahovat formuláře, interaktivní grafiku a další prvky, které byly dosud vyhrazené jen pro web. AMP Email má formát HTML s atributem amp4email. V rámci jednoho AMP emailu by měla být vždy posílána i textová a případně webová verze, jinak by nastával problém při starších emailových klientech [20, Ch. 6, p. 285]. Aktuálně je AMP Email podporován jen v klientech: Outlook, Gmail a Mail.ru [8].

[1] Google AdSence je platforma společnosti Google, která umožňuje si zakoupit reklamní prostor na různých webových stránkách.

AMP Add

Reklamy bez javascriptu fungovat skoro nemohou. A přestože pro koncové uživatele by byl web bez reklam ideální, tak z pohledu některých provozovatelů by bylo toto omezení likvidační. AMP tento problém řeší pomocí AMP komponenty AMP Add, která umožňuje vložit do stránky reklamní bannery. Díky tomu, že AMP přesně ví, které prvky na stránce jsou reklama, tak může zařídit, aby se na tyto pro uživatele nedůležité prvky nečekalo. Stránky, které budou zobrazovány v rámci AMP Add, musí být vydány schváleným provozovatelem a HTML reklamního banneru musí být anotováno atributem amp4add [20, Ch. 6, p. 280]. V lednu 2019 tvořily AMP reklamy 12% všech reklam v rámci Google AdSense [1].

AMP Stories

AMP Stories jsou celoobrazovkové "prezentace" vizuálně velmi podobné konceptu stories od společnosti Facebook. Tento typ obsahu je určen především pro zpravodajské portály, které jej mohou využít pro vytvoření krátkých "prezentací" k danému tématu. Na tyto prezentace poté uživatelé mohou přistoupit po hledání jména vydavatele v rámci prohlížeče nebo při hledání k samotné události, kterou AMP story popisuje [20, Ch. 6, p. 290].

2.2 Základní principy technologie AMP

Technologie AMP se skládá ze dvou hlavních částí: AMP frameworku a AMP distribuční infrastruktury.

Tabulka 2.1: Tabulka jednotlivých součástí technologie AMP

FRAMEWORK	DISTRIBUCE
AMP HTML	AMP Cache
AMP CSS	AMP Viewer
AMP Komponenty	AMP Prerender

Jednotlivé komponenty AMP frameworku slouží pro sestavení samotné AMP stránky. Na rozdíl od svých ne-AMP ekvivalentů jsou ořezané o některé funkce, při jejichž použití mohou být stránky pomalé, nebo by docházelo k dynamickým změnám rozložení. Pokud by docházelo k dynamickým změnám rozložení, tak by nebylo možné provádět prerendering.

Prvky distribučního procesu se různými metodami snaží minimalizovat čas potřebný k zobrazení stránky uživateli. Distribuční AMP infrastrukturu provozují aktuálně Google, Microsoft a CloudFlare [20, Ch. 1, p. 49].

AMP Framework

AMP stránky jsou tvořeny ve zjednodušeném jazyku HTML, který je rozšířený o AMP atributy a AMP komponenty. Omezení se nevyhnulo ani kaskádovým stylům. Jejich velikost je omezena na pouhých 75 kB, musí být umístěny přímo v dokumentu a nelze použít všechny typy animací a některé další konstrukty CSS. Největší omezení AMP technologie ale rozhodně představuje absence skoro jakéhokoliv javascriptu. Kvůli těmto omezením mají AMP stránky statický layout. Statický layout umožňuje, aby byla AMP stránka před vykreslená návštěvníkovi dříve než se vůbec rozhodne, že na danou stránku chce přistoupit.

AMP Prerendering je hlavní důvod, proč je AMP schopen zobrazit před stránky v čase blížícím se nule [24].

AMP Html

AMP Html vychází z HTML 5, je ale značně upravené. Některé základní HTML tagy jsou nahrazeny jejich AMP variantami a některé jsou zakázány. Oproti klasickému HTML můžeme AMP Html rozeznat podle bezhodnotového atributu s názvem amp (případně emotikonu High Voltage).

Výpis 2.1: Ukázka AMP Html

```
<!doctype html>
<html amp lang="cs">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,minimum-scale=1">
    <script async src="https://cdn.ampproject.org/v0.js"></script>
    <style amp-custom>
        /*INLINE CSS*/
    </style>
    <style amp-boilerplate>
        /* AMP BOILER PLATE CSS */
    </style>
    <link rel="canonical" href="UmisteniPuvodniStranky">
    <title>AMP</title>
</head>
<body>
    <h1>Toto je AMP stranka</h1>
</body>
</html>
```

Hlavička

V rámci hlavičky AMP dokumentu je nutné přidat AMP knihovnu a amp-boilerplate css[2] [3]. Toto css má za úkol skrýt stránku, dokud není plně vyrenderované a následně ji pomocí animace prolnutí zobrazit. Díky tomuto je stránka pocitově rychlejší. Vynechání těchto dvou prvků by způsobilo nevalidnost AMP stránky. AMP stránka se pak nedostane do cache.

V rámci hlavičky také můžeme přidat odkaz na požadované fonty. V rámci AMP stránek jsou povoleny jen fonty dostupné na zdrojích uvedených v tabulce 2.2 [2].

[2]Plné znění amp-boilerplate css je možné nalézt na https://amp.dev/boilerplate/

Tabulka 2.2: Tabulka povolených poskytovatelů fontů

TYPOGRAPHY.COM	`https://cloud.typography.com`
FONTS.COM	`https://fast.fonts.net`
GOOGLE FONTS	`https://fonts.googleapis.com`
TYPEKIT	`https://use.typekit.net`
FONT AWESOME	`https://maxcdn.bootstrapcdn.com` `https://use.fontawesome.com`

AMP Html elementy

Kvůli požadavku na fixní layout není možné použít všechny HTML elementy, některé jsou zakázané (viz. tabulka 2.3), některé byly nahrazeny jejich AMP variantami (viz. tabulka 2.4) a u jiných je omezena jejich funkcionalita (viz. tabulka 2.5) [22, p. 41].

Tabulka 2.3: Tabulka zakázaných HTML elementů

base	picture	frame	frameset	object	param	applet	embed

Tabulka 2.4: Tabulka nahrazených HTML elementů

img	amp-img
video	amp-video
audio	amp-audio
iframe	amp-iframe

Tabulka 2.5: Tabulka HTML elementů s omezenou funkcionalitou

script	Povoleny jen typu ***application/ld + json*** nebo ***text/plain***.
input	Typy: ***image***, ***button***, ***password*** a ***file*** jsou zakázané.
style	Povolen jen jediný styl v hlavičce (kromě amp-boilerplate), tento styl musí mít atribut. $\boldsymbol{amp-custom}$.
a	Jsou zakázané javascriptové odkazy.

AMP CSS

V rámci AMP stránky jsme omezeni pouze na max 75 kB (dříve jen 50 kB) inline CSS [20, Ch. 2 p. 100]. Důvodem, proč styly musí být přiloženy v rámci AMP stránky, je snaha o minimalizaci počtu potřebných request k tomu, aby mohla být načtena stránka (nebo

alespoň části ovlivňující její rozložení). Výjimkou jsou CSS keyframes[3], které můžou být obsaženy v separátním inline CSS elementu s maximální velikostí 500 kB [4].

Kromě omezení velikosti a nutnosti styly přikládat k hlavnímu dokumentu, AMP také zakazuje použití klíčového slova !important a stylování AMP komponent (selektory cílící na tagy začínající na -amp- nebo i-amp-) [6]. Obě tato omezení jsou nutná kvůli tomu, aby amp byl schopný zajistit, že elementy budou mít předpokládanou velikost.

Další omezení, se kterým musí vývojáři bojovat, je zákaz většiny animací. Jediné povolené animace jsou nad vlastnostmi, jejichž animování je GPU-accelerated, jde tedy o animace nad vlastnostmi opacity a transform [6].

Omezení použití javascriptu

Vlastní javascript je aktuálně kompletně zakázaný v rámci stránky. Existuje však několik způsobů, jak javascript spustit.

iframe

V rámci amp-iframe můžeme používat jakýkoliv javascript. Stejně jak u klasického HTML iframe i tady nemůžeme ovlivňovat nic mimo daný iframe. Toto řešení se hodí, pokud potřebujeme například zobrazit mapy, apod.

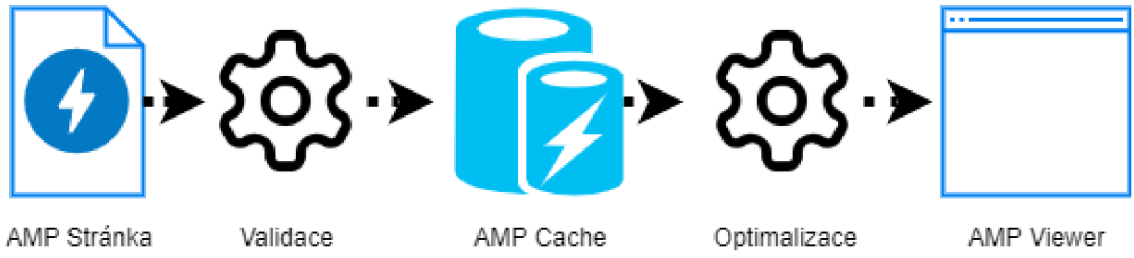
AMP-script

Koncem roku 2019 byla vydaná komponenta AMP-script. Tato dodatečná komponenta umožňuje spouštět javascript v rámci Web Workeru, tento kód běží na pozadí. V rámci tohoto kódu můžeme přidat interaktivitu do stránky například úpravami virtuálního DOMu stránky nebo úpravou stavu komponenty amp-state.

V rámci AMP-script nejsou povoleny všechny javascriptové API (podpora API pro manipulaci s prohlížečem chybí) a některé, jako například metoda querySelektor, jsou omezené [5].

Distribuční infrastruktura pro AMP stránky

Přestože jsme schopni zobrazovat AMP stránky bez využití distribučních mechanizmů AMPu, tak jsme přišli o výhody přednačítání stránek a dalších optimalizací, za které jsou zodpovědné AMP Cache a AMP Viewer. Bez těchto optimalizací je jediný benefit, který získáme použitím technologie AMP ten, že nás tato technologie nutí tvořit jednoduché stránky a omezuje nás ve velikosti externě načítaných zdrojů. Část takto ušetřeného času bychom vzápětí zase ztratili kvůli nutnosti stažení AMP knihovny.



Obrázek 2.1: Popis distribuční infrastruktury

[3]CSS keyframes obsahují definice animací.

Amp distribuční infrastruktura se skládá ze dvou hlavních částí: AMP Cache a AMP Vieweru. Schéma této infrastruktury je znázorněno na obrázku 2.1.

Amp stránka se do AMP cache dostane jen tehdy, je-li validní a provozovatel AMP cache usoudí, že se mu vyplatí danou stránku cachovat. Díky umístění stránky v AMP cache získáme kromě přiblížení obsahu ke koncovým uživatelům také různé typy optimalizací.

Amp Viewer má na starosti prerendering stránek z AMP Cache. Dalo by se říci, že je webovou aplikací obalující naši AMP stránku.

AMP Cache

Základní funkcionalitou AMP cache je poskytovat Content Delivery Network[4] pro AMP stránky. Stránky uložené v AMP Cache jsou poté přístupné pomocí AMP Vieweru. Kromě této základní funkcionality AMP Cache také poskytuje mechanismy pro validaci a optimalizaci AMP stránek.

Validace

Validace probíhá před samotným zařazením AMP stránky do cache a její výsledek rozhoduje o tom, jestli bude do cache zařazena. AMP validace kontroluje, zda není porušeno některé z omezení a požadavků AMP frameworku. Například zda stránka obsahuje jen validní elementy, nebo zda obsahuje AMP JS a CSS boiler plate. [7].

Optimalizace

Po zařazení AMP stránky do cache probíhají dodatečné optimalizace obsahu. Proběhne úprava HTML, během které dojde k odstranění prázdných znaků a komentářů, a stránka se upraví tak, aby nebylo možné provést Cross-site scripting [9]. Obrázky jsou zkomprimovány a jsou z nich odstraněna nepotřebná metadata. Pokud je to možné, dojde ke konverzi na WebP formát a ke generaci srcsetů. Externí zdroje jsou anotovány prefetch značkami, toto umožní AMP Prerender[5]. Nakonec dojde k agresivní minifikaci[6] HTML a CSS.

Aktualizace

Standardně AMP cache provádí aktualizace uložených stránek dle principu stale-while-revalidate, pokud si tedy uživatel vyžádá dokument, o kterém si AMP cache myslí, že je neplatný (například je překročená Max-Age hlavička), tak mu AMP cache předá tuto starou verzi a zažádá si o aktualizace, takže další uživatel dostane již verzi novou [16]. Majitel domény, v rámci které je AMP stránka hostovaná, může také “manuálně“ zažádat o aktualizaci stránky pomocí update-cache dotazu. Tento dotaz musí být podepsán privátním RSA klíčem asociovaným s danou doménou [17].

AMP Viewer

AMP Viewer je prostředí, ve kterém běží jednak kód hostované AMP stránky, tak z části kód samotného AMP Vieweru. Hlavní úlohou AMP Vieweru je optimalizace zobrazení AMP stránky a o její prerendering.

[4]Content Delivery Network je služba, která umožňuje, aby se externí obsah stahoval ze serveru, který je blíže k cílovému uživateli.

[5]Prerender: vykreslení stránky nebo její části ještě předtím, než si uživatel o obsah této stránky zažádá.

[6]Minifikace: odstranění nepotřebných prvků (mezery, komentáře,..) ze zdrojového kódu. Také může například docházet ke zkrácení názvů proměnných nebo funkcí.

AMP Prerendering
Prerendering probíhá u AMP stránek, o kterých si poskytovatel AMP Vieweru myslí, že na ně budeme chtít přistoupit, ještě předtím než na ně přistoupíme, v jednu chvíli může být tedy předpřipaveno více stránek. V daném momentu probíhá jen pro jednu stránku.

1. AMP Viewer stáhne AMP HTML z AMP Cache.
2. Načte potřebné JS knihovny.
3. Stáhne obrázky a další externí obsah vhodný pro dané zařízení.
   Dochází ke stažení jen toho obsahu, který se skutečně uživateli zobrazí okamžitě po příchodu na stránku [25].

Nevýhody AMP Vieweru
Z důvodu zobrazení naší AMP stránky v rámci AMP Vieweru dochází k tomu, že adresa, na které se uživatel nachází, nepřísluší naší stránce, ale je to adresa AMP Vieweru. Toto AMP Viewer aktuálně, pouze částečně, řeší pomocí falešného navigačního baru, který zobrazuje skutečnou zdrojovou adresu. Toto chování může zjednodušovat phishing útoky [20, Ch. 3, p. 50]. Řešením tohoto problému je technologie Signed HTTP Exchange, která umožní oddělit autora nebo zdroj obsahu od jeho poskytovatele [20, Ch. 3, p. 51].

Kapitola 3

DotVVM

DotVVM je open source MVVM[1] webový framework [15]. Backend část je založená na technologii ASP.NET (případně na její .NET Core variantě). Frontend je založený na značně upravené verzi javascriptové knihovny Knockout. Díky tomu, že DotVVM pokrývá frontend i backend, jsme schopni psát relativně komplexní aplikace jen s použitím jazyka C# (popřípadě jiného jazyka z platformy .NET) a jazyka DotHtml[2]. Z velké části se tak vyhneme psaní javascriptu pro běžné a repetetivní úkony, jako například komunikace se serverem a mapování dat mezi formáty, které se používají na backendu a frontendu. Díky těmto vlastnostem a snadné rozšiřitelnosti DotVVM o vlastní komponenty a komponenty třetích stran, je programátor se znalostí platformy .NET schopen velmi rychle vytvořit v DotVVM aplikaci.

DotVVM je vhodný převážně pro tvorbu Line of Business aplikací[3] [15]. Pro čistě statické prezentační weby většinou nepřináší dostatek benefitů, které by vyvážily snížení rychlosti oproti stránkám používajícím kombinaci HTML a jednoduchého javascriptu. Ve chvílích, kdy ale potřebujeme mít v rámci DotVVM Line of Business aplikace prezentační stránku, se může vyplatit i na tyto stránky použít DotVVM. Pro tento typ stránek by měl sloužit překlad DotVVM stránek do technologie AMP. Díky AMPu odstraníme problém rychlosti u DotVVM stránek bez příliš velkých nákladů na implementaci.

3.1 Návrhový vzor MVVM

MVVM je návrhový vzor, který zajišťuje separaci kódu popisujícího jak data zobrazovat od kódu, který s nimi manipuluje. Návrhový vzor MVVM se skládá ze tří hlavních částí: **M**odelu,**V**iew a **V**iew**M**odelu. Znázornění návrhového vzoru MVVM je ukázáno v obrázku 3.1.

Model

Model by měl být nositelem dat bez jakékoliv další funkcionality. To znamená, že by model měl být třída, která data uchovává (property), ale neobsahuje žádnou logiku (metody) pro manipulaci s nimi.

[1] MVVM: návrhový vzor model - view - viewmodel.

[2] DotHtml: HTML rozšířené o nové tagy, vazby na ViewModel a předpisy vazby DotHtml na Viewmodel.

[3] Line of Business aplikace: Rozsáhlé informační systémy, které obsahují velké množství formulářů, tabulek, atd.

Některé zdroje uvádějí jako výjimku pro toto pravidlo velmi jednoduchý kód v rámci get metody[4] jednotlivých vlastností a případně implementaci INotifyPropertyChanged[5] [21].

View

View specifikuje, jak by měla být data a akce obsažené v rámci ViewModelu reprezentovány. V rámci webových aplikací to bude tedy specifikace toho, jak bude vypadat výsledné HTML. Toto oddělení logiky, dat a zobrazování umožňuje mít pro jeden viewmodel více views. Časté použití pro tento scénář je, že potřebuje mít oddělenou stránku pro zobrazení na velkých displejích a na mobilních zařízeních. Aby bylo možné takto recyklovat viewmodel, tak je nutné, aby všechny Views zobrazovaly stejnou podmnožinu dat.

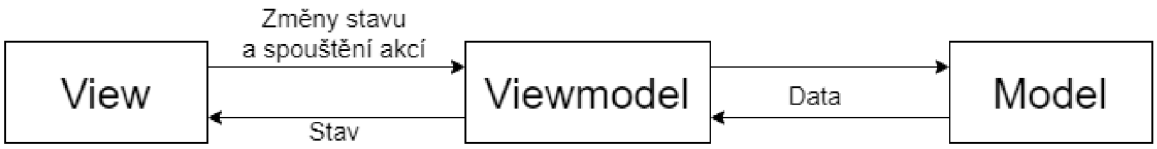
Viewmodel

Viewmodel je abstrakce View a vystavuje všechny Modely a akce, které jsou potřeba v rámci View. V rámci Viemodelu se provádí finální transformace dat, které získáme z Modelu/ů. Viewmodel také obsahuje metody pro všechny akce, které měly být spouštěny z View.

Na viewmodel je dobré se dívat jen jako na mezistupeň mezi tím co vidí uživatel a vnitřní aplikační logikou[6]. Tato separace se hodí zejména v případě, že potřebujeme zobrazovat stejná data na více různých stránkách, ale v natolik odlišné podobě, že nemůžeme použít jen více views pro stejný viewmodel. Díky tomu, že veškerou složitejší logiku přesuneme do samostatné vrstvy[7](třídy), tak se vyhneme duplikaci kódu.

Binder

Kromě tří hlavních částí MVVM obsahuje také komponentu, nazývanou Binder, která se stará o obousměrnou synchronizaci dat mezi View a Viewmodelem. Tato komponenta zajišťuje, že View bude zobrazovat aktuální data z Viewmodelu a také že viewmodel bude mít v době vykonávání akcí všechna data, které uživatel změnil v rámci View.



Obrázek 3.1: Diagram návrhového vzoru MVVM.

[4]Vlastnosti třídy, které jsou jen pro čtení a v rámci get metody obsahují kód pro složení výsledné hodnoty, se v rámci C nazývají calculated properties.

[5]Interface INotifyPropertyChanged slouží k oznámení zbytku aplikace, že se některá z hodnot objektu změnila.

[6]aplikační logika: Business Logic

[7]Pokud používáme vícevrstvou architekturu.

3.2 DotVVM ViewModel

DotVVM ViewModel je C# třída dědící od DotvvmViewModelBase, která na straně serveru reprezentuje data, která jsou klientské straně zobrazována, a akce, které je možné provádět. Díky tomu, že viewmodel musí být jak pro přenos ze serveru na klienta, tak pro přenos zpátky serializován, je nutné, aby všechna data, která potřebujeme u klienta, byla definována jako public vlastnosti. Také je nutné, aby veškeré objekty, které chceme poslat na klienta, měly bezparametrický konstruktor. Akce[8], které může stránka spouštět, jsou ve viewmodelu definovány jako veřejné metody.

Bázová třída DotvvmViewModelBase zajišťuje, že viewmodel bude mít přístup k objektu Context, který obsahuje například informace o příchozím HTTP požadavku nebo informace o tom, zda je uživatel přihlášený. Tato bázová třída také definuje virtuální metody Init, Load a Prerender, které jsou volány v různých fázích životního cyklu stránky (viz. kapitola 3.6).

DotVVM Viewmodel také podporuje použití validačních a autorizačních atributů.

Na jednotlivých vlastnostech jsme schopni pomocí standardních validačních atributů zajistit, že nedojde ke spuštění specifických akcí, pokud nejsou data ve viewmodelu validní [13].

To, že je uživatel autorizovaný k přístupu k danému viewmodelu, můžeme zajistit tak, že viewmodel anotujeme autorizačním atributem. Autorizaci můžeme provádět nejen na úrovni celého viewmodelu, ale i na úrovni jednotlivých metod [10]. V ukázce 3.1 je uveden příklad viewmodelu s třemi vlastnostmi a jednou akcí.

Výpis 3.1: Ukázka DotVVM Viewmodelu

```
1  public class ExampleViewmodel : DotvvmViewModelBase
2  {
3      public int Number { get; set; }
4
5      public string Text { get; set; }
6
7      public ExampleClass ComplexClass { get; set; }
8
9      public void CustomCommand()
10     {
11     }
12 }
```

3.3 DotVVM View

DotVVM View je stránka napsaná v jazyce DotHtml.DotHtml je HTML rozšířeném o vlastní komponenty (v ukázce 3.2 <dot:TextBox . . . /> , <dot:Literal . . . /> a <dot:Button . . . />) a takzvaný Binding systém, který umožňuje definovat vztah na vlastnost / metodu ve viewmodelu.

[8] Akce jsou v rámci DotVVM nazývané Commands. [14]

Výpis 3.2: Ukázka DotHtml

```
1  @viewModel TestSamples.Common.ViewModels.ExampleViewmodel, TestSamples.Common
2  <!DOCTYPE html>
3
4  <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
5  <head>
6      <meta charset="utf-8" />
7      <title>{{value: Text}}</title>
8  </head>
9  <body>
10     <dot:TextBox Text="{value: Number}" ValueType="Number" />
11     <dot:Literal Text="{value: Text}" />
12     <dot:Button Click="{command: CustomCommand()}"
13                 Text="Run custom command" />
14 </body>
15 </html>
```

Na začátku každého view se nachazí definice toho, který viewmodel bude použit pro dané view. Pokud bychom na stránce chtěli používat Static Command Services[9], nebo enum hodnoty, které jsou v jiném namespace než daný viewmodel, tak v této sekci view také můžeme importovat potřebné prostředky. Pod touto sekcí se již nachází DotHtml kód.

V rámci DotHtml je možné používat "šablony"[10], které nám umožňuji složit výslednou stránku z více částí. Takto můžeme například sdílet hlavičku stránky. V případě použití tohoto systému jsou na stránkách, které používají nějakou šablonu namísto celého DotHtml, obsaženy jen sekce[11], které hlavní stránka šablony specifikuje jako obsah k doplnění.

Komponenty

DotVVM komponenty jsou v rámci View zaznačeny jako HTML tag, jehož název se skládá z názvu skupiny komponent a názvu vlastní komponenty, tyto dva údaje jsou odděleny dvojtečkou. Za takto složeným názvem následují jednotlivá nastavení dané komponenty. Tato nastavení můžou být rovnou specifikována v rámci view, nebo pomocí binding systému se můžou odkazovat na hodnotu ve viewmodelu. Každá komponenta také může mít ve svém otevíracím tagu klasické HTML atributy.

HTML komponenty, které do stránky přidávají dynamické chování, jsou v DotVVM v základu tvořeny samostatnými DotVVM komponentami (například dot:TextBox,dot:Button nebo dot:GridView). Další balíky komponent lze dokoupit (DotVVM Bootstrap, DotVVM BusinessPack) případně existují komunitou spravované balíky komponent. Na každém větším projektu také většinou vznikají komponenty, které jsou specifické pro daný projekt.

[9] Static Command Services představují způsob, jak zavolat metodu na serveru bez nutnosti přenášet celý viewmodel.

[10] DotVVM masterpage

[11] Každá šablona obsahuje ve své hlavní stránce komponenty ContentPlaceholder, které specifikují, že na toto místo bude dosazen kód, který musí být obsažen ve všech stránkách, které tuto šablonu používají. Kód k doplnění je obalen komponentou Content, která specifikuje, že její obsah patří na určité místo v dané šabloně.

CodeOnly controls

Vzhled a nastavení komponenty jsou definovány v rámci C# kódu. Vzhled komponenty je tvořen pomocí manuálního složení výsledného stromu prvků.

Markup controls

Komponenta jen obaluje větší množství DotHtml kódu do jedné komponenty. Používá se, pokud chceme větší bloky DotHtml používat opakovaně, ale nevyžadujeme větší customizace pro jednotlivá použití komponenty.

Markup controls with Codebehind

Markup controls with Codebehind představují kombinaci dvou předchozích přístupů včetně všech jejich výhod a nevýhod. Nevyžadují ruční skládání stromu komponent, ale umožňuje rozsáhlejší konfiguraci. Bohužel ale není možné dělat složitější věci, které lze v Codeonly controls, a nejsou ani tak jednoduché na použití jako markup controls.

Binding systém

Pro napojení View na Viewmodel se používá binding systém. Tento systém podporuje přenos hodnot definovaných ve veřejných vlastnostech uvnitř viewmodelu do view a také volání veřejných metod ve viewmodelu. Binding systém se zapisuje ve formátu: typBindingu: vlastnost/metoda() [11]. Jednotlivé typy bindingů jsou uvedeny v tabulce 3.1.

Tabulka 3.1: Tabulka typů bindingu

value	Napojení se na hodnotu ve viewmodelu, hodnota se vyhodnocuje na klientovi.
resource	Napojení se na hodnotu ve viewmodelu, hodnota se vyhodnocuje na serveru.
command	Spuštění akce na serveru.
staticCommand	Spuštění akce na klientovi: Pro jednoduché akce jako například přiřazení do proměnné. Spuštění akce na serveru: Lze spustit jen metody anotované atributem static command. Na server se neposílá celý viewmodel.

Na straně klienta se používá javascriptový framework knockout.js . DotVVM/Knockout.js uchovává viewmodel ve skrytém input elementu[12] na konci HTML body elementu. Jednotlivé napojení typu value jsou v rámci vygenerovaného HTML nahrazeny knockout.js výrazy, které zajistí napojení na viewmodel.

Kromě přímého využití hodnot ve viewmodelu, také můžeme do napojení typu value zapsat pravdivostní výraz. Takto můžeme například na základě toho, jestli některá z vlastností má námi vyžadovanou hodnotu nebo skrývat / zobrazovat elementy. Také je možné

[12]Skrytý input element umožňuje, že obsah viewmodelu nebude ztracen (narozdíl od stavu JS) po návratu na předchozí stránku.

v těchto situacích využívat ternární operátory a další jednoduché operace jako například zřetězení.

3.4 DotVVM Route Table

Ke každé DotVVM stránce by měl být přiřazen minimálně jeden záznam v DotVVM route table [12]. Díky tomuto záznamu DotVVM dokáže z URL příchozího požadavku určit, které DotVVM View je požadováno, jestli jsou součástí cesty nějaké parametry a zda má použít jiný než defaultní DotVVM presenter. Využití jiného než defaultního presenteru se může hodit, příchozí požadavek zpracovat jiným způsobem než vykreslením DotVVM stránky. Příklad registrace do DotVVM route table je uveden v ukázce 3.3.

Výpis 3.3: Ukázka přidání záznamu do DotVVM route table v rámci DotvvmStartup.cs.

```
config.RouteTable.Add(
    // nazev cesty
    "Example",

    // specifikace URL
    "/ExampleUrl/{param}",

    // cesta k~DotVVM View
    "Views/Example",

     // defaultn hodnota parametru
    new {param="test"},

    //specifikace vyzadovaneho presenteru
    provider => provider.GetService<LocalizablePresenter>()
);
```

Jiný než standardní DotVVM presenter se hodí například pokud potřebujeme lokalizovat jednotlivé stránky (na to použijeme LocalizableDotvvmPresenter) nebo pokud chceme například pro request na specifickou adresu vracet JSON[13] namísto HTML.

3.5 Externí zdroje

Dotvvm obsahuje mechanismus na správu externích prostředků[14]. Tento mechanismus je užitečný, například když potřebujeme v rámci DotVVM stránek referencovat obsah (například javascript nebo css) třetích stran. Různé typy externích prostředků jsou uvedeny v tabulce 3.2. Pomocí Dotvvm Resource Manageru můžeme zaregistrovat tyto zdroje v aplikaci a později jen specifikovat, které komponenty vyžadují přítomnost dané knihovny. V rámci této registrace specifikujeme zapamatovatelný název pro daný prostředek, jeho typ a způsob jeho získávání[15] a také jeho závislostí. Různé typy zdrojů externích prostředků jsou uvedeny v tabulce 3.3. V ukázce 3.4 je uveden příklad registrace externího CSS prostředku, který se nachází na disku a má jednu závislost.

[13] JavaScript Object Notation
[14] DotVVM Resource Manager
[15] Resource Location

Výpis 3.4: Registrace CSS

```
config.Resources.Register("bootstrap-css", new StylesheetResource()
{
    Location = new FileResourceLocation("~/Content/bootstrap.min.css"),
    Dependencies = new[] { "bootstrap-js" }
});
```

Tabulka 3.2: Tabulka typů prostředků pro Dotvvm Resource Manager

ScriptResource	Vloží do stránky link na JS soubor.
InlineScriptResource	Vloží do strány HTML script element s obsahem vyžadovaného JS souboru.
StylesheetResource	Vloží do strány link na CSS soubor.
InlineStylesheetResource	Vloží do strány HTML sctyle element s obsahem vyžadovaného CSS souboru.
NullResource	Neudělá nic, tento typ se používá, pokud chceme zaregistrovat prostředek, ale chceme jeho typ a zdroj doplnit až za běhu aplikace.

Tabulka 3.3: Tabulka typů zdrojů externích prostředků

FileResourceLocation	Pro prostředky přístupné přes souborový systém.
UrlResourceLocation	Pro prostředky přístupné přes URL.
EmbeddedResourceLocation	Pro prostředky vložené přímo v DLL (toto je učitečné zejména pokud vyvíjíme knihovnu, která bude později používána v rámci DotVVM aplikace.
InlineResourceLocation	Prostředek je specifikován přímo v rámci zdrojového kódu.
UnknownResourceLocation	Slouží jako dočasný location.

Pomocí DotVVM Resource Manageru můžeme tedy uchovávat informace o externích prostředcích použitých v rámci aplikace na jednom místě, což nám v případě změny jejich verze, způsobu získávání nebo jejich závislosti, umožní aplikovat požadované změny na jednom místě.

V rámci samotné DotVVM stránky, nebo komponenty, specifikujeme požadované externí prostředky pomocí jejich zaznamenání do RequiredResources kolekce v rámci DotVVM kontextu (viz. ukázka 3.5), případně pomocí komponenty dot:RequiredResource v rámci DotHtml (viz. DotHtml v ukázce 3.6). Všechny požadované prostředky budou doplněny do stránky na konec HTML tagu head nebo body v závislosti na jejich typu.

Výpis 3.5: Specifikace požadovaného prostředku v rámci C kódu

```
context.ResourceManager.AddRequiredResource("bootstrap-css");
```

Výpis 3.6: Specifikace požadovaného prostředku v rámci DotHtml

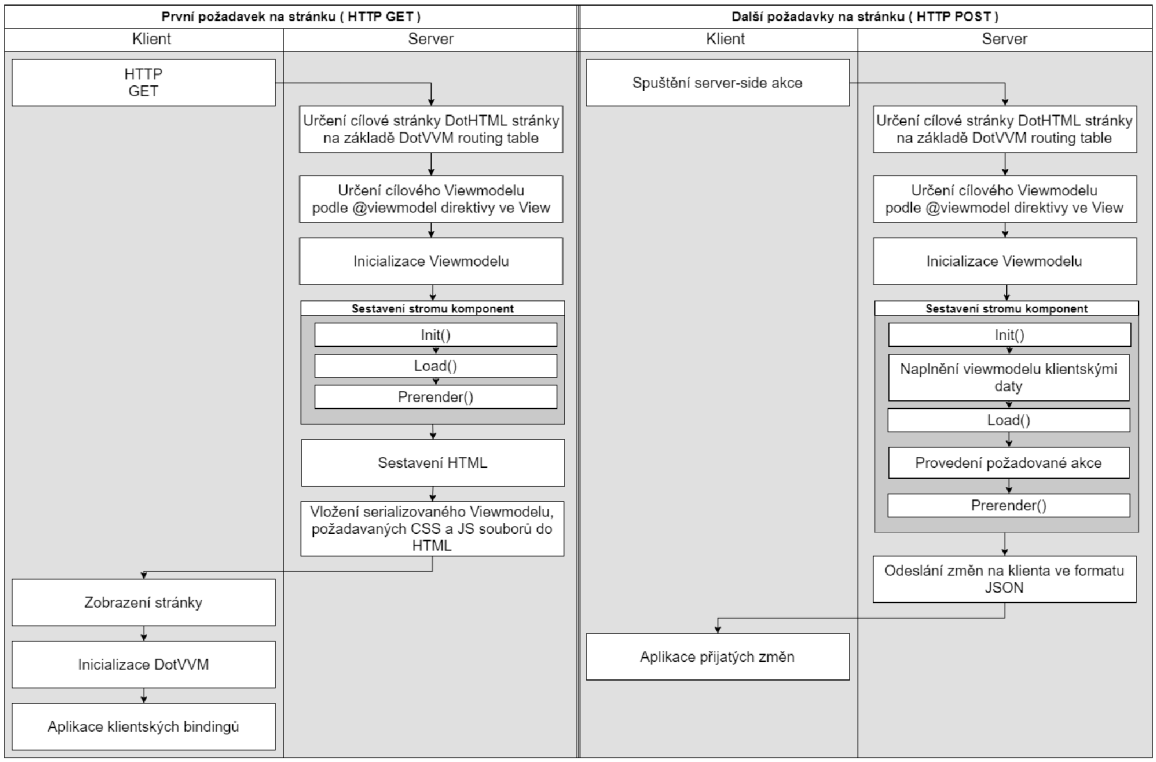
```
1 <dot:RequiredResource Name="bootstrap-css" />
```

Některé prostředky jsou nutné pro správný běh DotVVM stránky, a proto jsou zaregistrovány a vyžadovány automaticky.

- **dotvvm**: Esenciální množina funkcí pro korektní běh DotVVM stránky.
- **dotvvm.debug**: Pomocný kód, který usnadňuje ladění DotVVM stránek. Tento prostředek je obsažen jen v rámci Debug módu.
- **dotvvm.fileUpload-css**: CSS styly pro fileupload komponentu.
- **knockout**: Upravená verze Knockout.JS.
- **globalize**: Upravená verze Globalize.JS.
- **globalize:*JAZYK***: Informace o formátu data a času pro aktuální jazyk.

3.6 Životní cyklus DotVVM stránek

Životní cyklus DotVVM stránky se liší na základě toho, jestli jde o první požadavek na stránku (GET), nebo o požadavek na vykonání nějaké akce (POST) [14]. Tento algoritmus je ve zjednodušené podobě znázorněn v obrázku 3.2.



Obrázek 3.2: Popis životního cyklu DotVVM stránek při HTTP GET a POST požadavcích.

Každé DotHtml je přeloženo do stromové struktury a následně je z tohoto stromu vygenerováno výsledné HTML, které se odešle jako odpověď na klientský dotaz. Po každé fázi generování HTML se volají metody, do kterých může uživatel doplnit vlastní logiku. Tyto metody se volají na celém viewmodelu, a na každé komponentě.

1. Do příslušného DotVVM presenteru dojde request od klienta. Výběr tohoto presenteru zajišťuje DotVVM routing.
2. DotvvmViewBuilder sestaví z DotVVM View markapu strom reprezentující dané view.
   2.1. Podle informací obsažených v tomto stromu se do jeho uzlů doplní .net typy, příslušný datakontext a další sémantické informace => vznikne sémantický strom.
   2.2. Z tohoto sémantického stromu se vytvoří strom komponent. Jednotlivé komponenty jsou reprezentovány jako .net třídy, které popisují, jak pro danou komponentu vytvořit HTML. V tomto stromu jsou reprezentované i normální HTML tagy jako HtmlGenericControl.
3. Nastaví se RequestContext[16].
4. Vytvoření viewmodelu.
5. Zavolání metody OnPageInitializedAsync na zaregistrovaných filterch typu IViewModelActionFilter[17].
6. Vygenerovanému stromu se nastaví DataContext[18].
7. Zavolání OnPreInit na jednotlivých komponentách.
8. Zavolání Init metody na viewmodelu.
9. Zavolání OnInit na jednotlivých komponentách.
10. Zavolání metody OnViewModelCreatedAsync na zaregistrovaných filtrech typu IPageActionFilter[17].

[16] RequestContext: Objekt obsahující informace o HTTP požadavku.
[17] ActionFilter: Třída implementující rozhraní IActionFilter, která umožňuje provádění dodatečných akcí během kompilace a inicialize DotVVM stránky.
[18] DataContext: Reference na objekt ze které se budou brát všechny navázané vlastnosti.

První request.

10.1. Zavolání Load metody na viewmodelu.

10.2. Zavolání OnLoad na jednotlivých komponentách.

Postback

10.1. Načtení viewmodelu z příchozího HTTP požadavku.

10.2. Zavolání metody OnViewModelDeserializedAsync na zaregistrovaných filterch typu IViewModelActionFilter[17].

10.3. Validace CSRF Tokenu[19].

10.4. Zavolání Load metody na viewmodelu.

10.5. Zavolání OnLoad na jednotlivých komponentách.

10.6. Zavolání metody OnCommandExecutingAsync na zaregistrovaných filterch typu ICommandActionFilter[17].

10.7. Spuštění požadované akce.

10.8. Zavolání metody OnCommandExecutedAsync na zaregistrovaných filterch typu ICommandActionFilter[17].

11. Zavolání PreRender metody na viewmodelu.
12. Zavolání OnPreRender na jednotlivých komponentách.
13. Zavolání PreRenderComplete na jednotlivých komponentách.
14. Vygenerování CSRF tokenu[19].
15. Zavolání metody OnViewModelSerializingAsync na zaregistrovaných filterch typu IViewModelActionFilter[17].
16. OutputRenderer sestaví výslednou odpověď na klientský požadavek.
    (a) Nastaví požadované HTTP hlavičky.
    (b) Zavolá metodu Render na sestaveném view (kořenu stromu), ten pak metodu Render zavolá na svých potomcích.
    (c) Zkontroluje se, že došlo k vyrenderování externích prostředků.
    (d) Vytvoří nový HtmlWriter, který bude zapisovat do odpovědi.
17. Odešle se odpověď na klientský požadavek.

[19] CSRF Token slouží k obraně před **C**ross **S**ite **R**equest **F**orgery útoky.

Kapitola 4

Integrace technologie AMP do DotVVM

Hlavní myšlenkou vytvoření integrace technologie AMP do již existujících DotVVM stránek bylo umožnit napsat celou webovou aplikaci, včetně statických veřejných stránek, v jedné technologii bez toho, abychom přišli o rychlost, kterou nám čistě statický web nabízí, ale kterou při použití webových frameworků částečně ztrácíme.

Aby tato myšlenka v praxi fungovala, je nutné minimalizovat objem změn a konfigurace, kterou musí uživatel udělat, aby jeho stránka využila místo DotVVM technologii AMP. Proto je použití tohoto rozšíření navrženo tak, aby stránky, které neobsahují žádný technologií AMP zakázaný prvek, fungovaly bez jakýchkoli úprav. V tomto ideálním scénáři by mělo stačit jen označit danou stránku jako stránku s podporou technologie AMP. Po této jednoduché konfiguraci by měla vzniknout verze původní stránky, která nebude na straně klienta používat technologii DotVVM, ale technologii AMP.

Aplikace velké části potencionálních uživatelů bohužel nejspíše nebudou splňovat všechny podmínky nutné pro převod do technologie AMP bez jakýchkoliv dodatečných úprav nebo konfigurací. Pro tyto případy je nutné podporovat způsob, jak v rámci View anotovat, které části se používají jen v rámci AMP stránky a které jen v rámci DotVVM stránky. Díky možnosti takto specifikovat různé verze pro jednotlivé technologie bude možné se vyhnout nucené úpravě již existující DotVVM stránky jen proto, aby její AMP verze byla validní.

Posledním důležitým aspektem této integrace je umožnit (ale nevynucovat) možnost konfigurovat, případně nahradit, jednotlivé komponenty zajišťující převod DotVVM stránky do technologie AMP.

Z těchto poznatků vyplývají základní prvky, které by měl tento projekt splňovat:

- Jednoduchý na použití.
- Minimalizace potřebného množství zásahů do původních stránek.
- Velká míra rozšiřitelnosti.

4.1 Registrace a konfigurace DotVVM.AMP

Pro správné chování všech prvků DotVVM.AMP je nutné nejdříve zaregistrovat potřebné prostředky do DotVVM a IOC[1] kontejneru. API[2] pro registraci DotVVM.AMP je navrženo tak, aby se co nejvíce podobalo registračnímu API dalších rozšíření DotVVM[3].

Registrace prostředků

Pro registraci prostředků a komponent, které DotVVM.AMP vyžaduje, se používá zavolání extension metody[4] AddDotvvmAmp na typu DotvvmConfiguration. Tuto registraci bych doporučoval provádět v rámci metody Configure v třídě DotvvmStartup.

Tato metoda zaregistruje do DotvvmResourceRepository dva prostředky: AMP CSS boilerplate a AMP JS boilerplate. Tyto prostředky vloží do HTML hlavičky ampem vyžadovaný CSS a JS kód.

Registrace AMP komponent

V rámci volání extension metody[4] AddDotvvmAmp se také zaregistrují všechny AMP komponenty, které jsou v namespace DotVVM.AMP.AmpControls jako DotVVM komponenty s prefixem amp[5].

Registrace používaných tříd do IOC kontejneru

Kvůli požadavku na snadnou modifikaci defaultního chování jednotlivých částí DotVVM.AMP bylo zapotřebí co nejvíce omezit vznik pevných závislostí mezi jednotlivými částmi knihovny. Tento problém jsem se rozhodl vyřešit pomocí DI[6].

Všechny závislosti jednotlivých komponent nejsou definovány pomocí konkrétní implementace, ale pouze pomocí jejich rozhraní. IOC kontejner poté při vytváření instance doplní ke každému požadovanému rozhraní příslušnou implementaci. To jakou implementaci (a to jestli má vytvářet nebo znovu používat dané instance) je definováno v rámci registrací komponent do IOC kontejneru.

Tato registrace se provádí v rámci volání extension metody[4] AddDotvvmAmpSupport na typu IDotvvmServiceCollection.

Konfigurace DotVVM.AMP

Veškerá konfigurace DotVVM.AMP se uchovává v rámci třídy DotvvmAmpConfiguration. V rámci celé webové aplikace by vždy měla existovat jen jedna instance této třídy. Tímto způsobem je zjištěno, že všechny stránky se chovají stejně. Konfigurace se dá pomyslně rozdělit na tři části: instance sdílených vlastností, globální nastavení a nastavení chování jednotlivých typů validací.

---

[1] IOC: Inversion of Control

[2] Programové rozhraní, které využívají ostatní části aplikace pro kontrolu našeho kódu.

[3] Nejrozšířenější rozšíření DotVVM jsou balíky komponent DotVVM Bootstrap a DotVVM BusinessPack.

[4] Extension metoda je statická metoda, které je přidána k určitému typo na jiném místě než v rámci jeho definice.

[5] Jejich použití bude tedy <amp:*JMÉNO* ... /> .

[6] DI: Dependency Injection

Sdílené instance

- *ControlTransforms*
  Registr obsahující registrace jednotlivých transformací DotVVM komponenty na AMP komponentu.
- *RouteManager*
  Třída obsahující logiku pro pojmenovávání a správu cest na jednotlivá AMP stránky.
- *ExternalResourceMetadataCache*
  Cache obsahující metadata (například rozměry) o externích zdrojích (primárně obrázcích).

Globální nastavení

- *TryToDetermineExternalResourceDimensions*
  Určuje, zda v případě nalezení obrázku, u kterého neznáme jeho velikost, dojde k pokusu zjistit jeho rozměry.
- *StyleRemoveForbiddenImportant*
  Určuje, zda se mají být z CSS kódu odstraněny AMPem zakázané modifikátory !important.
- *AmpJsUrl*
  Určuje jaký zdroj povinného AMP JS. Změnou tohoto parametru můžeme změnit používanou verzi technologie AMP.
- *MaximumAmpCustomStylesheetSize*
  Určuje, jaká velikost v bytech vlastního CSS bude ještě akceptována.

Nastavení validací

Všechny validace podporují dva režimy:

- Throw
  Při první chybě je překlad ukončen a dojde ke vzniku záznamu v logu a výjimky, která danou chybu popisuje.
- LogAndIgnore
  Při chybě bude daná chyba zaznamenána, ale kompilace bude pokračovat dále. Element (nebo jeho část) obsahující chybu bude ve většině případů vynechán.

Kvůli zajištění větší granulity jsou validace v různých kontextech nastavovány samostatně pomocí těchto vlastností v rámci DotvvmAmpConfiguration:

- *AttributeHandlingMode*
  Nastavení chování při nalezení nevalidního HTML atributu.
- *KnockoutHandlingMode*
  Nastavení chování při nalezení knockout bind atributu (xml:*).
- *HtmlTagHandlingMode*
  Nastavení chování při nalezení nevalidního HTML tagu.

- *StylesHandlingMode*
  Nastavení chování při nalezení nevalidního CSS.

- *UnsupportedControlPropertiesHandlingMode*
  Nastavení chování při nalezení nevalidně nastavené DotVVM komponenty.

Veškerá nastavení by měla probíhat výlučně v rámci volání metody AddDotvvmAmpSupport. Této metodě je možné jako parametr předat Action<DotvvmAmpConfiguration>[7]. Tato Action by měla sloužit jako jediné místo, kde bude zasahováno do DotvvmAmpConfiguration. Takto provedené úpravy konfigurace budou aplikovány hned, jakmile si nějaká část DotVVM.AMP poprvé vyžádá přístup ke konfiguraci.

Výpis 4.1: Registrace DotvvmAmpConfiguration

```
serviceCollection.Services.AddSingleton<DotvvmAmpConfiguration>(
    provider =>
{
    var registry = provider.GetService<IAmpControlTransformsRegistry>();
    var routeManager = provider.GetService<IAmpRouteManager>();
    var logger = provider.GetService<ILogger>();
    var resourceMetadataCache =
        provider.GetService<IAmpExternalResourceMetadataCache>();
    var config =
        new DotvvmAmpConfiguration(registry,
                                   routeManager,
                                   resourceMetadataCache);

    RegisterTransforms(config, logger);
    modifyConfiguration?.Invoke(config);
    return config;
});
```

4.2 Rozšíření DotVVM routing table

Jelikož DotVVM.AMP nenahrazuje již vytvořené DotVVM stránky, ale jen k nim vytváří jejich AMP verzi, tak se jako nejlepší místo pro definici, které stránky budou přeloženy do technologie AMP, jeví kód provádějící registrace DotVVM stránek do DotVVMRouteTable (příklad registrace DotVVM stránky je uveden v ukázce 4.2).

Výpis 4.2: Registrace DotVVM stránky do DotvvmRouteTable

```
config.RouteTable.Add("Default", "index", "Views/Default.DotHtml");
```

[7] Action<DotvvmAmpConfiguration> : Anonymní metoda s návratovým typem void, která jako jediný parametr dostane instanci DotvvmAmpConfiguration.

Pro jednoduché stránky nepotřebuji od uživatele jiné informace, které jsou povinné v rámci registrace klasické cesty. Tudíž se jako ideální řešení způsobu registrace jeví přidání DotVVM.AMP pomocí nahrazení volání metody Add nad DotvvmRouteTable extension metodou AddWithAmp, která se volá nad stejným typem a má skoro stejné vyžadované parametry. Takto upravená registrace do DotVVM route table je uvedena v ukázce 4.3.

Výpis 4.3: Registrace AMP stránky do DotvvmRouteTable

```
1 config.RouteTable.AddWithAmp("Default",
2                              "index",
3                              "Views/Default.DotHtml",
4                              config);
```

Jediné co se na signatuře metody změnilo (kromě jejího názvu) je nový parametr vyžadující instanci DotVVMConfiguration.

Pro náročnější scénáře je možné jako parametry předat defaultní hodnoty url parametrů jak DotVVM stránky tak její AMP verze. Také je možné specifikovat, který DotVVM presenter bude použit pro DotVVM stránku a který pro její AMP verzi. Specifikace presenteru umožňuje například podporu lokalizací pomocí LocalizablePresenter (resp DotvvmAmpLocalizablePresenter pro AMP variantu).

Zavolání metody AddWithAmp vygeneruje dvě odlišné DotVVM cesty, jednu pro původní DotVVM stránku a druhou pro její AMP verzi.
Generování těchto cest probíhá následovně:

- Získání AMP konfigurace z IOC kontejneru.
- Vygenerování názvu DotvvmRoute pro AMP verzi pomocí AmpRouteManageru.
  V základu mají cesty tento formát: *PŮVODNÍ-NÁZEV*-amp .
- Vygenerování URL pro AMP verzi pomocí AmpRouteManageru.
  V základu má tento formát: *amp/PŮVODNÍ-URL* .
- Pokud nejsou nastaveny separátní defaultní hodnoty pro AMP verzi, použijí se tak defaultní hodnoty pro původní verzi.
- Přidání cesty pro DotVVM stránku.
- Přidání cesty pro AMP stránku.
- Registrace překladu do AmpRouteManageru.
  Tato informace se používá pro nalezení DotVVM varianty AMP stránky a naopak.

4.3 AmpPresenter

DotvvmPresenter funguje jako centrální bod, který kontroluje průběh zpracování requestu na DotVVM stránku. Tato třída má jako závislosti (které jsou inicializovány pomocí DI[6]) všechny části DotVVM infrastruktury. Z tohoto důvodu je vlastní DotvvmPresenter ideální místo pro nahrazení DotVVM závislostí jejich DotVVM.AMP verzemi.

AmpPresenter funguje jako dekorátor defaultního DotvvmPresenteru. Tento dekorátor modifikuje dvě části původního chování.

Nahrazení původních závislostí

Závislosti na defaultní DotVVM třídy jsou nahrazeny jejich AMP variantami v rámci konstruktoru[8].

- IDotvvmViewModelBuilder => IAmpDotvvmViewBuilder
  AmpViewBuilder rozšiřuje DefaultDotvvmViewBuilder o podporu transformací během sestavování výsledného stromu komponent.
- IOutputRenderer => IAmpOutputRenderer
  AmpOutputRenderer nahrazuje HtmlWriter za AmpHtmlWriter. AmpHtmlWriter umožňuje provádět poslední validace před vygenerováním HTML.

Registrace resource preprocesorů

Kvůli nutnosti upravovat externí prostředky (hlavně CSS) je nutné pro každý request zaregistrovat dvě třídy typu IResourceProcessor do ResourceManageru.

- AmpCustomCssResourceProcessor
  AmpCustomCssResourceProcessor zajišťuje spojení všech CSS do jednoho[9] inline stylu.
- AmpResourcesProcessor
  AmpResourcesProcessor vyfiltruje všechny nepovolené prostředky.

DotvvmAmpLocalizablePresenter

Druhým presenterem, který je v rámci DotVVM.AMP implementován, je DotvvmAmpLocalizablePresenter. Tento presenter je AMP variantou na DotVVM LocalizablePresenter. Účel localizable presenteru je nastavit do CultureInfo.CurrentCulture uživatelem vyžádaný jazyk[10]. Po nastavení jazyka je další zpracování požadavku delegováno na defaultní DotvvmPresenter.

Jediný rozdíl v DotvvmAmpLocalizablePresenteru oproti LocalizablePresenteru je že namísto toho, aby se delegovalo zpracování dalšího requestu na DotvvmPresenter, tak se zpracování deleguje na AmpPresenter.

4.4 Rozšíření DotHtml

Některé technologií AMP vyžadováné prvky jsou v rámci DotVVM.AMP řešené náhradou stávající komponenty (HTML elemetntu,...) za separátní komponentu. Mimo interní komponenty existují také komponenty, které lze použít v rámci DotHtml. Pomocí těchto komponent lze například přidat do již existujícího HTML prvky, které budou obsažené jen v AMP verzi (nebo jen v DotVVM verzi).

Interní komponenty

Většina komponent v DotVVM.AMP jsou interní komponenty použité jen pro účely transformací. Interní komponenty nelze používat v rámci DotHtml.

[8] Constructor Injection: Závislosti vyžadované v rámci konstruktoru jsou doplněny patřičnými instancemi dle pravidel DI kontejneru.

[9] V některých případech je možné, že vzniknou dva inline styly.

[10] Vyžadovaný jazyk je nejčastěji zadán v rámci nepovinného URL parametru.

AmpBody

DotVVM se v rámci renderování HtmlGenericControl[11], která má nastaven svůj HTML tag na body, pokusí vyrenderovat všechny externí prostředky. Tomuto chování ale musíme v rámci DotVVM amp zabránit, protože tyto prostředky defaultně obsahují javascriptové soubory nutné pro běh DotVVM.

Tento mechanismus bohužel není navržen tak, aby bylo možné tomuto chování jednoduše zabránit. Z tohoto důvodu jsem se na tomto místě musel uchýlit k využití reflection[12]. Při překladu první stránky se získá informace o vlastnosti[13] BodyRendered na typu ResourceManager. Tato informace o vlastnosti se následně využije pro nastavení této vlastnosti na hodnotu true, což zabrání renderování nechtěných prostředků.

RouteLink

Komponenta RouteLink rozšiřuje chování defaultního RouteLinku logiku, která zajišťuje použití Amp varianty odkazované stránky namísto původní DotVVM stránky. Pokud AMP varianta této stránky neexistuje, je použita původní stránka.

Pro vyhledání AMP varianty se používá AmpRouteManager. AmpRouteManager obsahuje tabulku, která eviduje, jaké DotVVM stránky mají AMP verzi. Registrace do této tabulky vznikají při registraci cest do DotVVM route table.

AmpLinkToAnotherVersion

AmpLinkToAnotherVersion je varianta na AMP RouteLink. Tato komponenta přidává odkaz na původní DotVVM stránku (nebo naopak z DotVVM stránky na její AMP verzi). Tato komponenta se používá místo komponenty AMP RouteLink z důvodu nutnosti správně určit URL parametry pro původní stránku.

Pokud původní DotVVM stránka má defaultní parametry a pro přístup na aktuální stránku byly použity defaultní parametry pro AMP verzi, vygenerovaný odkaz vede na URL používající defaultní parametry DotVVM stránky. To znamená, pokud byly parametry defaultní, defaultní zůstanou, jen se použijí defaultní parametry pro správnou verzi dané stránky.

Pokud výše uvedené pravidlo splněno není, použijí tak se aktuální parametry.

AmpRepeater a AmpGridView

AmpRepeater a AmpGridView v DotVVM verzi používají vnitřní šablony.

- Komponenta Repeater zduplikuje tuto šablonu pro všechny prvky v jejím zdroji dat[14]. Každý takto zduplikovaný prvek obsahuje data, která jsou získána z daného prvku.
- Komponenta GridView obsahuje v šabloně definice (šablony) jednotlivých sloupců tabulky, které jsou pak použity pro vykreslení jednotlivých řádků.

[11] Každý HTML tag (ne DotVVM komponenty) je v rámci stromu komponent reprezentován jako HtmlGenericControl.

[12] Reflection například umožňuje dynamicky za běhu měnit hodnoty jinak nedostupných vlastností (například privátní vlastnosti).

[13] ProperetyInfo: Informace o vlastnosti sloužící pro nastavení nebo získání hodnoty pomocí reflection.

[14] DataSource: zdroj dat pro danou komponentu

Obsah těchto šablon není ve stromu komponent během transformací, a proto se na něj transformace neaplikují. Z tohoto důvodu je nutné všechny transformace aplikovat dodatečně a tudíž jsou defaultní komponenty Repeater a Gridview nahrazeny jejich AMP variantami.

Amp komponenty

Některé komponenty v DotVVM.AMP jsou určené i k použití v rámci DotHtml. Scénář pro použití těchto komponent například je, aby výsledné nastavení dané komponenty bylo jiné, než jak tyto komponenty nastavuje transformace.

Image

Tato komponenta reprezentuje náhradu za HTML img tag. Této komponentě můžeme nastavit běžné vlastnosti obrázku jako například velikost a zdroj, ale také i vlastnosti specifické pro amp-img. Hlavní z těchto nových vlastností je atribut Layout. Tento atribut určuje, jak bude výsledný obrázek napozicován v rámci AMP stránky.

Z důvodu modifikaci URL oproti původní stránce je v rámci generování HTML pro obrázek nutné zajistit, aby zdroj obrázku obsahoval absolutní cestu. Pokud je detekována relativní cesta, dojde k její transformaci na cestu absolutní pomocí doplnění kořenové cesty, která se získá z příchozího požadavku.

Hodnota tohoto atributu Layout také ovlivňuje, zda je nutné, aby obrázek měl zadanou výšku anebo šířku. Pokud není Layout nastaven, je nutné, aby obrázek měl výšku i šířku. V případě že velikost obrázku nastavená není, a v rámci DotvvmAmpConfiguration je nastavena vlastnost TryToDetermineExternalResourceDimensions na hodnotu true, tak se pokusím velikost obrázku zjistit.

Detekce velikosti obrázku se provádí v rámci třídy AmpExternalResourceMetadataCache. Obrázek se nejprve stáhne a pak se pokusím vyčíst jeho velikost. Tento mechanismus bohužel selže pro vektorové obrázky[15]. Pro každý obrázek (pro každou URL) je detekce velikosti provedena pouze jednou. Pro každý další dotaz se použije dříve získaná hodnota.

Exlude a Include

Pro kontrolu, zda se nějaká stránka má vyrenderovat jen v rámci AMP verze nebo jen v rámci původní Dotvvm stránky, lze použít komponenty Exlude a Include.

Obsah Exclude komponenty se vyrenderuje jen v původní stránce a obsah Include komponenty jen v AMP verzi. Pro tento účel lze také použít attached propertu[16] AMP.Exclude.

Detekce toho, zda se komponenta nachází v AMP nebo původní verzi stránky, je založena na tom, že pouze v AMP verzi probíhají transformace. Na detekci, zda proběhly transformace, se používá transformace, která doplňuje instanci DotvvmAmpConfiguration do vlastnosti Configuration v rámci dekorátorů. Pokud je vlastnost Configuration prázdná (hodnota je null), to znamená že se nenacházíme na AMP verzi stránky.

To, jestli dojde nebo nedojde k vygenerování obsahu těchto stránek se provádí pomocí podmíněného přeskočení fáze RenderContents dané komponenty.

[15] Vektorové obrázky nemají specifikovanou velikost.

[16] Attached properties: Vlastnosti, které lze použít v rámci jakékoliv komponenty.

4.5 Sestavení DotVVM View

O sestavení Dotvvm View se v klasických DotVVM stránkách stará DefaultDotvvmViewBuilder. Tato třída nejprve načte příslušné DotHTML a z něj pak sestaví strom komponent. V případě, že stránka používá šablony, tak se v rámci sestavení stromu komponent načte a zpracuje DotHTML i těchto stránek. Na konci tohoto procesu vznikně DotVVM komponenta DotvvmView, která reprezentuje strom komponent.

V rámci DotVVM.AMP se pro sestavení view používá AmpViewBuilder. Tato třída dekoruje IDotvvmViewBuilder. AmpViewBuilder nejprve nechá původní implementaci IDotvvmViewBuilder sestavit strom komponent pro původní DotVVM stránku a následně na takto sestavený strom začne aplikovat transformace. Proces výběru a aplikace transformací je znázorněn v ukázce 4.4.

Výpis 4.4: Ukázka aplikace transformací

```
1  public void ApplyTransforms(DotvvmControl root,
2                              IDotvvmRequestContext context)
3  {
4      ApplyTransforms(root.GetAllDescendants().ToList(), context);
5  }
6
7  public void ApplyTransforms(List<DotvvmControl> allControls,
8                              IDotvvmRequestContext context)
9  {
10     foreach (var control in allControls)
11     {
12         GetTransform(control)?.Transform(control, context);
13     }
14 }
```

Transformace

Transformace jsou třídy implementující IControlTransform. Tyto třídy mají za úkol transformovat strom komponent reprezentující DotVVM stránku, na strom komponent reprezentující AMP stránku. Transformace dělíme do několika kategorií: klasické, validační a nahrazující. To jestli, je možné danou transformaci použít, zjistíme pomocí metody CanTransform, kterou musí každá transformace implementovat.

Klasické transformace

Klasické transformace dědí z ControlTransformBasem. Jsou to transformace, které mohou pouze měnit nastavení dané komponenty. Tyto transformace se používají například pro nastavení vykreslovacího módu na serverové vykreslování. Samotná transformace probíhá v několika krocích:

1. Rozhodnutí, zda by tato komponenta měla být zahrnuta ve výsledném stromu komponent.
   V případě, že je na komponentě nastaveno například Amp.Exclude na hodnotu true, dojde k jejímu vyřazení ze stromu komponent a dále se již v transformaci nepokračuje.

2. Provedení BeforeTransform metody.
   Tuto metody aktuálně žádná z transformací neimplementuje. Metoda existuje z důvodu pozdější rozšiřitelnosti.

3. Provedení TransformCore metody.
   Každá transformace tuto metodu musí implementovat. V této transformaci by měly proběhnout všechny úpravy.

4. Nastavení vyžadovaných vlastností.
   Všechny AMP komponenty musí být vyrenderovány již na serveru. Tato metoda nastavení RenderSettings.ModeProperty na hodnotu RenderMode.Server.

5. Aplikace attached properties.
   V tomto kroku dojde k přidání AMP atributů fallback nebo placeholder, pokud je komponenta vyžaduje.

6. Provedení AfterTransform metody.
   Metoda existuje z důvodu pozdější rozšiřitelnosti. Některé transformace v rámci této metody provádějí finální úpravy komponenty.

Validační transformace

Validační transformace jsou rozšířením klasických transformací. Jsou to transformace, které dědí z ControlValidatorTransformBase. Validační transformace provádí ověření, zda je danou komponentu v jejím aktuálním nastavení možné použít v rámci AMP stránky. Tyto transformace v rámci metody TransformCore provedou validaci požadovaných vlastností. Pokud daná vlastnost není validní, dojde k výjimce nebo k vynechání dané komponenty a zalogování chyby (záleží na nastavení UnsupportedControlPropertiesHandlingMode). Validační transformace se používají například pro kontrolu toho, zda není v rámci komponenty GridView zapnuta podpora editačního módu.

Nahrazující transformace

Nahrazující transformace jsou rozšířením validačních transformací. Jsou to transformace, které dědí z ControlReplacementTransformBase. Nahrazující transformace nahradí jednu komponentu za jinou v rámci TransformCore metody. V rámci nahrazení dojde ke zkopírování všech atributů a vlastností. Při nahrazení komponenty je nutné přenést všechny potomky dané komponenty a také je důležité správně nastavit odkaz na rodičovskou komponentu a původní komponentu ze stromu odebrat. Kód zajišťující nahrazení komponenty je uveden v ukázce 4.5.

Výpis 4.5: Nahrazení jedné DotVVM komponenty za jinou.

```
public void ReplaceControl(DotvvmControl currentControl,
                           DotvvmControl newControl)
{
    var parent = currentControl.Parent as DotvvmControl;
    DotvvmControl[] children =
                           new DotvvmControl[currentControl.Children.Count];
    currentControl.Children.CopyTo(children, 0);
    if (parent != null)
    {
        var childIndex = parent.Children.IndexOf(currentControl);
        parent.Children.Remove(currentControl);
        parent.Children.Insert(childIndex, newControl);
    }
    currentControl.Children.Clear();
    currentControl.Parent = newControl.Parent;
    newControl.Children.Add(children);
}
```

4.6 CSS infrastruktura

AMP zakazuje externích CSS styly a omezuje styly vložené do stránky na max 75 kB vlastního CSS v jediném HTML style tagu. Jedinou výjimkou pro toto pravidlo je maximálně 500 kB CSS keyframes, které můžou být obsaženy v samostatném style elementu.

Díky těmto pravidlům je nutné nejprve všechny externí a embedované[17] styly spojit do jediného CSS kódu a ten následně rozdělit na CSS keyframes a zbytek. Spojení probíhá v několika fázích.

Modifikace stromu komponent

V rámci aplikace transformací dochází odebrání všech embedovaných stylů a odkazů na externí styly. Tyto styly jsou odebrány ze stromu komponent a přidány do vyžadovaných prostředků.

Sjednocení CSS v rámci externích prostředků

V průběhu vyhodnocování externích prostředků dochází k aplikaci ResourceProcessorů[18]. Pro sjednocení CSS se používá AmpCustomCssResourceProcessor, který je registrován v rámci DotvvmAmpPresenteru. Tento processor vždy, při nalezení CSS prostředeku, tak jej uloží do AmpStylesheetResourceCollection. Když projde všechny vyžadované prostředky, vrátí ještě dva další prostředky: AmpCustomStylesheetResource a AmpCustomStylesheetResource.

[17] Embedované: Vložené přímo na stránku.

[18] ResourceProcessor: Třída implementující IResourceProcessor. Tato třída v rámci metody Process bere jako parametr množinu externích prostředků a vrací množinu externích prostředků. Tuto třídu lze tedy použít pro modifikaci (například přidání nebo odebrání) externích prostředků. ResourceProcessory jsou při aplikaci volány postupně => Výsledky jednoho dostane na vstupu následující ResourceProcessor.

Vygenerování amp-custom a amp-keyframes elementů

Během vykreslení prostředku AmpCustomStylesheetResource nebo AmpCustomStylesheetResource dojde k vygenerování výsledného CSS. O generování CSS se stará AmpStylesheetResourceCollection. Pokud AmpStylesheetResourceCollection již někdy generovalo daný CSS kód, tak vrátí původní verzi, jinak dojde k vygenerování CSS pro amp-custom a CSS pro amp-keyframes.

Generování CSS

1. Asynchronně se stáhnou všechny požadované CSS.
2. Všechny CSS kódy se spojí do jediného CSS.
3. Dojde k ověření na předčastné ukončení.
4. CSS se naparsuje.[19]
5. Pokud je v rámci DotvvmAmpConfiguration nastavena vlastnost StyleRemoveForbiddenImportant na true, dojde k odstranění všech CSS !important anotací.
6. Naparsovaný CSS dokument se rozdělí na CSS Keyframes a ostatní.
7. Obě výsledné CSS projdou minifikací.[20]

4.7 Sestavení finální stránky

Když proběhnou veškeré transformace na stromu komponent a jsou připraveny sjednocené CSS prostředky, dochází k vygenerování finálního HTML. Za tento úkon je v rámci klasické DotVVM stránky zodpovědný HtmlWriter. HtmlWriter obsahuje mimo jiné metody pro přidávání různých html atributů, vygenerování počátečního a koncového html tagu a generování čistého textu nebo html kódu. Každá komponenta dostane instanci IHtmlWriter v rámci své Render fáze a využívá tuto třídu pro sestavení výsledného HTML.

V rámci DotVVM amp jsem HtmlWriter využil jako místo, ze kterého se provádí finální validace. AmpHtmlWriter dekoruje HtmlWriter a každé z jeho volání obaluje kontrolou na validnost požadovaného konstruktu. Pokud validace tedy selže, tak se daný konstrukt nevygeneruje. Když takto dojde k přeskočení počátečního tagu, je následně nutné ignorovat vše až do příslušného ukončovacího tagu. Kód zajišťující toto chování je uveden v ukázce 4.6.

[19] Parsování CSS se provádí pomocí knihovny ExCSS-Core.
https://github.com/HockeyJustin/ExCSS_Core

[20] Minifikace: odstranění všech přebytečných znaků pro účel snížení velikosti.
Pro minifikaci CSS se používá kód publikovaný Madsem Kristensenem na jeho blogu.
https://madskristensen.net/blog/efficient-stylesheet-minification-in-c

Výpis 4.6: Metoda RenderBeginTag na třídě AmpHtmlWriter

```
1  public virtual void RenderBeginTag(string name)
2  {
3      if (validator.CheckHtmlTag(name,Attributes))
4      {
5          Attributes.Clear();
6          writer.RenderBeginTag(name);
7      }
8      else
9      {
10         StartTagSkipped = true;
11     }
12 }
```

4.8 Validace

Mnoho DotVVM aplikací používá i v rámci svých úvodních (a jiných jednoduchých) stránek některou z forem obsahu, které nejsou technologií AMP podporované. Nejčastější takové prvky jsou kontaktní formuláře, webové galerie vyžadující javascript nebo jiné prvky, které bohužel nelze jednoduše převést do technologie AMP. Pro tyto případy je nutné dát uživatelům vědět, jaké části jejich webových stránek nemohou být převedeny do technologie AMP a z jakého důvodu. Z diskuze s lidmi, vyvíjejícími v technologii DotVVM, vyplynulo, že část chce jen vědět, že se na stránce vyskytují nevalidní prvky (spolu s informací o místě a typu problému), ale i přesto chtějí, aby stránka byla do technologie AMP převedena. A to i v případě, že by nevalidní elementy nebo jejich části byly v průběhu překladu vynechány. Druhá skupina uživatelů preferuje čistě opačný přístup. Tato skupina chce, aby ve chvíli, kdy je nalezen nevalidní element, došlo k výjimce a nedocházelo tak k tomu, že z výsledné stránky budou vynechány některé elementy.

Z tohoto důvodu si uživatel pro každou skupinu validačních pravidel může nastavit, jak se má validace chovat v případě nalezení nevalidního prvku. Pro každou skupinu existují dva módy: Throw a Log and Ignore. Bližší popis jednotlivých módů lze nalézt v rámci popisu nastavení validací v této kapitole.

V rámci generování HTML pro některé komponenty se stává, že komponenta i zkusí přidat klientský DataBind atribut. V rámci AMP stránek bohužel tyto atributy nemohou fungovat, protože javascript, který tyto atributy zpracovává, není možné do stránky vložit. Některé tyto dataBind výrazy nám ale v rámci AMP stránek nevadí, protože zajišťují aktualizaci hodnot při postbacku. Namísto toho, aby v těchto případech nastala výjimka (v případě že je pro danou validační skupinu její chování nastaveno na Throw), tak jen zalogujeme upozornění, že byl tento atribut vynechán.

4.9 Rozšiřitelnost

Technologie AMP v základu obsahuje jen tři vestavěné komponenty: amp-layout, amp-image a amp-pixel [20, Ch. 3, p. 160]. Tyto tři komponenty by měly pokrýt potřeby pro vytvoření úplně základní stránky. Pokud ale chceme pokročilé komponenty, na které jsme zvyklí ze světa plnohodnotných webových stránek (jako například fotogalerii nebo youtube video) tak musíme sáhnout po některé z rozšiřujících komponent.

Aktuálně je v rámci technologie AMP dokumentace k 240 komponentám. V rámci DotVVM.AMP jsou ale podporovány jen výše zmíněné tři komponenty. Pro přidání podpory pro další komponenty je možné pomocí vytvoření vlastní transformace a případně i vlastní komponenty.

Pokud bychom chtěli přidat podporu pro komponentu amp-iframe[21], tak musíme nejdříve vytvořit DotVVM komponentu [22], kterou nahradíme všechny iframe na stránce. Tato komponenta bude obsahovat všechny vlastnosti, které budeme chtít podporovat. Dále budeme do stránky potřebovat vložit odkaz na požadovaný javascript. Toto uděláme stejně jako kdybychom potřebovali přiložit jakýkoliv jiný prostředek k DotVVM komponentě. Zaregistrujeme prostředek, který vyrenderuje potřebný script HTML tag do DotVVMResourceManageru a ten v rámci OnInit metody přidáme jako required resource.

Jako poslední krok musíme vytvořit transformaci, která všechny HtmlGenericControl s tagem iframe na stránce nahradí za náš iframe a zkopíruje jeho atributy do vlastností, které máme v rámci naší DotVVM komponenty. Pokud bychom nechtěli kopírovat atributy, ale přesto využívat výhody Dotvvm properties, můžeme upravit getter příslušných vlastností podobně jako v ukázce 4.8. Tato transformace by měla dědit z ControlReplacementTransformBase. Bázová třída za nás vyřeší nahrazení nové komponenty ve stromu komponent a nakopíruje všechny DotVVM properties a atributy na novou komponentu. Výsledná transformace by mohla vypadat podobně jako ukázka kódu 4.7. Poslední zbývající krok je zaregistrovat naši transformaci do AmpControlTransformRegistry. Tuto registraci můžeme udělat v rámci volání extension metody AddDotvvmAmpSupport nad typem IDotvvmServiceCollection. Příklad této registrace je uveden v ukázce kódu 4.9.

Výpis 4.7: Ukázka možné implementace tranformace pro iframe

```
1  public class IframeTransform : ControlReplacementTransformBase
2  {
3   public IframeTransform(
4                          DotvvmAmpConfiguration ampConfiguration,
5                          ILogger logger = null
6                         ) : base(ampConfiguration, logger)
7   {
8   }
9
10  public override bool CanTransform(DotvvmControl control)
11  {
12     return control is HtmlGenericControl generic &&
13            generic.TagName == "iframe";
14  }
15
16  protected DotvvmControl CreateReplacementControl(DotvvmControl con)
17  {
18      return new AmpIframe();
19  }
20 }
```

[21] amp-iframe: https://amp.dev/documentation/components/amp-iframe/?format=websites

[22] Tutoriál na tvorbu komponent lze nalézt v rámci DotVVM dokumentace v sekci Control Development. https://www.dotvvm.com/docs/tutorials/control-development-code-only-controls/2.0

Výpis 4.8: Možný způsob jak se vyhnout kopírování HTML atributů do DotVVM property.

```
public virtual void RenderBeginTag(string name)
{
    public string Src
    {
       get {return GetPropertyValueOrAttribute(SrcProperty, "src");}
       set {SetValue(SrcProperty, value); Attributes.Remove("src");}
    }
}
```

Výpis 4.9: Ukázka možné implementace tranformace pro iframe

```
public class IframeTransform : ControlReplacementTransformBase
{
    options.AddDotvvmAmpSupport((configuration, logger) =>
    {
        configuration.ControlTransforms.Register(
               new IframeTransform(configuration,logger));
    });
}
```

4.10 Testování

V rámci vývoje DotVVM.AMP jsem pro jednotlivé komponenty a transformace vytvářel testovací stránky a automatizované UI testy k těmto stránkám. Pro UI testy jsem použil knihovnu Selenium, která umožňuje ovládat prohlížeč přímo z testů. Celkově vzniklo 15 integračních testů[23]. Pro testování stránek se ukázalo užitečné rozšíření do prohlížeče Chrome AMP Validator[24]. Toto rozšíření provádí validace AMP stránek na základě aktuálních validačních pravidel AMP.

Během vývoje komponent a mechanismů DotVVM.AMP vznikly testovací DotVVM stránky. Stručný popis těchto testovacích stránek je uveden v tabulce 4.1. Na takto vzniklých stránkách jsem testoval, zda došlo ke správnému vykreslení výsledného AMP HTML a zda je toto AMP HTML validní. Jako finální ukazatel toho, jestli je stránka validní, jsem používal rozšíření prohlížeče chromu AMP Validator[24].

Pro některé infrastrukturní komponenty a mechanismy jsem kromě manuálního testování vytvořil integrační UI testy. Konkrétně se testují komponenty Literal a HtmlLiteral a testovací stránka, která vykresluje prázdnnou AMP stránku. Tyto testy vyžadují, aby na předem určené adrese běžela aplikace obsahující sadu testovacích stránek. Každý test spustí pomocí testovacího frameworku Selenium webový prohlížeč. Prohlížeč je spouštěn pomocí driver aplikace pro příslušný prohlížeč. Tento driver umožňuje instanci webového prohlížeče programově ovládat. Běžící test po nastartování prohlížeče přejde na požadovanou stránku a na ní provede příslušné operace a kontroly. Díky tomu, že DotVVM.AMP slouží především pro zobrazení statických stránek, tak testy jen zkontrolují, zda výsledné

[23] Testy a testovací stránky jsou dostupné v rámci repozitáře projektu. https://github.com/MichalTichy/DotVVM.AMP

[24] Rozšíření AMP Validator je dostupné z https://chrome.google.com/webstore/detail/amp-validator/nmoffdblmcmgeicmolmhobpoocbbmknc.

HTML splňuje daná kritéria. Příklad testovacího kódu je uveden v ukázce 4.10. Ve verzi 0.1.1 všechny automatizované testy procházejí.

Výpis 4.10: Ukázka testovacího kódu.

```
1  public class IframeTransform : ControlReplacementTransformBase
2  {
3      [Fact]
4      public void HtmlTagHasAmpAttribute()
5      {
6          RunInAllBrowsers(SampleUrl, wrapper =>
7          {
8              var html = wrapper.Single("html", By.TagName);
9              AssertUI.HasAttribute(html, "amp");
10         });
11     }
12 }
```

Nejvíce vracejících se chyb objevily testy, které kontrolovaly vykreslení prázdné stránky. Vykreslení AMP validní prázdné stránky bylo komplikované kvůli tomu, že bylo třeba zabránit, aby DotVVM do stránky přidalo jakýkoliv kus javascriptu nebo jiný v technologii AMP nevalidní prvek.

Měření výkonosti

Měřením prováděném na reálné aplikaci jsem naměřil, že DotVVM stránka se zpracuje za 43 ms a zpracovaní AMP stránky trvá 177.44 ms. Toto měření bylo provedeno na aplikaci, která měla inicializované všechny vnitřní cache. Měřen byl čas od odeslání požadavku do přijetí odpovědi. Toto zpomalení je způsobné z velké míry nutností stahovat a zpracovávat externí prostředky. Z výsledků měření vyplývá, že vygenerování AMP verze je pomalejší než vygenerování DotVVM verze. Toto zpomalení je zapříčiněno tím, že DotVVM.AMP jen rozšiřuje vykreslovací infrastrukturu DotVVM. Toto zpomalení by v produkční verzi nemělo mít vliv, protože stránka bude koncovému uživateli poskytnuta ne z našeho serveru ale z AMP Cache.

Tabulka 4.1: Tabulka testovacích stránek.

Empty	Stránka testující vygenerování prázdné AMP stránky.
CssSingle CssMultiple CssExternalCombined CssInlineCombined WithDependendCss	Stránky testující různé formy vložení CSS do stránky.
ContentPlaceHolderPage ContentPlaceHolder- PageContentTest DoubleContentPlaceHolder- PageContentTest	Stránky testující funkčnost šablon.
RouteLinkUrlGen RouteLinkUrlGenToAmp RouteLinkUrlGenBinding TestRouteAmp TestRouteAmp2	Stránky testující navigační odkazy.
LinkToAmpPage LinkToNonAmpPage	Stránky testující navigační odkazy z DotVVM stránky na AMP stránky a naopak.
RepeaterNamedTemplate RepeaterNestedRepeater RepeaterRepeaterWrapperTag RepeaterSeparator	Stránky testující komponentu Repeater. Repeater opakuje svůj vnitřní obsah.
BasicGridView EmptyDataHidden EmptyDataDisplayed	Stránky testující komponentu GridView. GridView vykresluje tabulku.
Literal HtmlLiteral	Stránky testující komponenty Literal a HtmlLiteral. Tyto komponenty vykreslují text/html.
Img	Stránky testující transformace obrázků.
Include Exclude	Stránky testující komponenty Include a Exclude, které zajišťují podmíněné vykreslovaní svého obsahu.

Kapitola 5

Postup integrace překladu stávajících DotVVM stránek do technologie AMP

V této kapitole popíši postup, jak DotVVM.AMP použít pro přidání AMP verze k již existující DotVVM stránce v rámci reálné webové aplikace. V rámci této ukázky použiji DotVVM.AMP ve verzi 0.1.1[1]. Jako aplikaci, na které budu převod do AMP prezentovat, jsem zvolil webovou aplikaci, která slouží pro prezentaci portfolia. Konkrétně jsem zvolil stránku, která obsahuje detail projektu. Tuto aplikaci (a konkrétní stránky v rámci aplikace) jsem zvolil z toho důvodu, že je dynamicky generovaná[2], ale přesto neobsahuje mnoho prvků, které by bránily vytvoření její AMP verze. Levá část obrázku 5.1 ukazuje, jak vypadá původní DotVVM stránka.

5.1 Přidání DotVVM.AMP do projektu

Jako první krok je třeba přidat knihovnu DotVVM.AMP do naší aplikace. Nejjednodušší způsob, jak tohoto docílit, je přidat referenci na nuget balík DotVVM.AMP[1]. Přidáním tohoto balíku přidáme referenci na knihovnu DotVVM.AMP.

Konfigurace DotVVM.AMP

Pro správné fungování DotVVM.AMP musíme upravit třídu DotvvmStartup.

1. Zavoláme metodu AddDotvvmAmp nad instancí DotvvmConfiguration.
2. Zavoláme metody AddDotvvmAmpSupport nad instancí IDotvvmServiceCollection. V rámci této metody můžeme měnit AMP konfiguraci.

Příklad toho, jak může konfigurace vypadat, nalezneme v ukázce 5.1.

[1] Knihovna DotVVM.AMP ve verzi 0.1.1 je veřejně dostupná na nuget.org.
https://www.nuget.org/packages/DotVVM.AMP/0.1.1

[2] View je napsáno obecně pro jakýkoliv projekt a specifické informace o dané referenci jsou doplněné v rámci modelu.

Výpis 5.1: Ukázka možné AMP konfigurace

```
serviceCollection.AddDotvvmAmpSupport((configuration,_) =>
{
        configuration
            .AttributeHandlingMode = ErrorHandlingMode.LogAndIgnore;
        configuration
            .HtmlTagHandlingMode = ErrorHandlingMode.LogAndIgnore;
        configuration
            .KnockoutHandlingMode = ErrorHandlingMode.LogAndIgnore;
        configuration
            .HtmlTagHandlingMode = ErrorHandlingMode.LogAndIgnore;
        configuration
            .StylesHandlingMode = ErrorHandlingMode.LogAndIgnore;
        configuration
            .StyleRemoveForbiddenImportant = true;
});
```

5.2 Registrace DotVVM.AMP pro jednotlivé cesty

V rámci naší ukázkové aplikace je registrace jednotlivých cest trochu odlišná od standardního scénáře. V této aplikaci probíhá registrace přes separátní metodu, která v rámci registrace cesty nastaví pro danou cestu lokalizaci. Úpravou této metody tedy dojde k registraci DotVVM.AMP na všechny stránky. V našem případě to nevadí, ale pokud bychom potřebovali zaregistrovat DotVVM.AMP jen pro některé cesty, museli bychom vytvořit variantu této metody pro AMP a variantu pro verzi bez technologie AMP. Kód výše zmiňované metody je uveden v ukázce kódu 5.2.

Výpis 5.2: Registrace cesty s DotVVM.AMP

```
config.RouteTable.AddWithAmp(
        routeName,
        "{Language?:length(2)}",
        virtualPath,
        config,
        dotvvmPagePresenterFactory:
        LocalizablePresenter.BasedOnParameter("Language"),
        ampPagePresenterFactory:
        DotvvmAmpLocalizablePresenter.BasedOnParameterWithAmp("Language")
        );
```

5.3 Úpravy stránky s detailem projektu

Pokud by byla daná stránka dostatečně jednoduchá, tak by nám výše zmíněné úpravy stačily k tomu, abychom mohli vygenerovat AMP verzi dané stránky. V našem případě ale z logu po přístupu na stránku zjistíme, že AMP verze není plně validní. V této chvíli je nejlepší začít postupně odstraňovat nebo upravovat jednotlivé nepřeveditelné prvky.

Ideální je začít od šablon, které stránka používá, protože změny v šablonách se projeví na více místech. V našem případě v šabloně, kterou naše stránka používá je třeba upravit

skripty, které se do stránky přidávají v rámci hlavičky. Tento problém vyřešíme pomocí nastavení attached property Amp.Exclude na true na Html script elementech. Tímto zajistíme, že dané elementy nebudou v rámci AMP verze vygenerované.

Další krok je úprava komponenty navigační bar. Problémem v této komponentě je, že u loga na její levé straně musíme specifikovat jeho velikost. Dané logo je vektorové[3], a proto ho nelze určit automaticky. Při prozkoumání původní stránky jsem zjistil, že logo se zobrazuje vždy s šířkou 12rem a výškou 4rem. Tyto dva parametry nastavíme původnímu obrázku pomocí attached properties Amp.Width a Amp.Height. Tuto úpravu můžeme vidět v příkladu 5.3.

Výpis 5.3: Upřesnění velikosti obrázku.

```
<img src="/Resources/images/Logos/riganti-logo-white.svg"alt="logo"
    Amp.Width="12rem"
    Amp.Height="4rem" />
```

Další část, kterou musíme na upravit, je samotné View stránky pro zobrazení detailu projektu. Na této stránce jsou tři prvky, které nelze převést do technologie AMP.

Stránka obsahuje kanonický odkaz[4]. V rámci AMP stránek musí ale jediný povolený kanonický odkaz směřovat na původní (ne AMP) verzi dané stránky. Z tohoto důvodu tento odkaz odstraníme z AMP verze pomocí attached property Amp.Exclude nastavenou na hodnotu true.

Výpis 5.4: Odstranění komponenty fancybox.

```
<amp:Exclude>
       <a data-fancybox="Reference gallery"
          href="{value: Image.Src}"
          IncludeInPage="{resource: _this.Image.Src != null}">
               <img class="reference-image"
                    src="{value: Image.SrcSmall}"
                    alt="{value: Title}" />
       </a>
</amp:Exclude>
<amp:Include>
       <a>
              <img src="{value: Image.SrcSmall}"
                    alt="{value: Title}"
                    IncludeInPage="{resource: _this.Image.Src!=null}"/>
       </a>
</amp:Include>
```

Dále stránka používá pro zobrazení detailu obrázky javascriptovou knihovnu fancybox. V rámci technologie AMP ale nelze javascriptové knihovny použít. AMP framework sice obsahuje dodatečné komponenty s podobnou funkcionalitou, ale pro tuto stránku jsem se rozhodl, že tuto funkcionalitu vynechám pro AMP verzi. Původní zobrazení obrázku obalíme komponentou amp:Exclude. Obsah této komponenty nebude v rámci AMP verze vygenerován. Původní zobrazení obrázků musíme nyní nahradit verzí, která bude kompatabilní

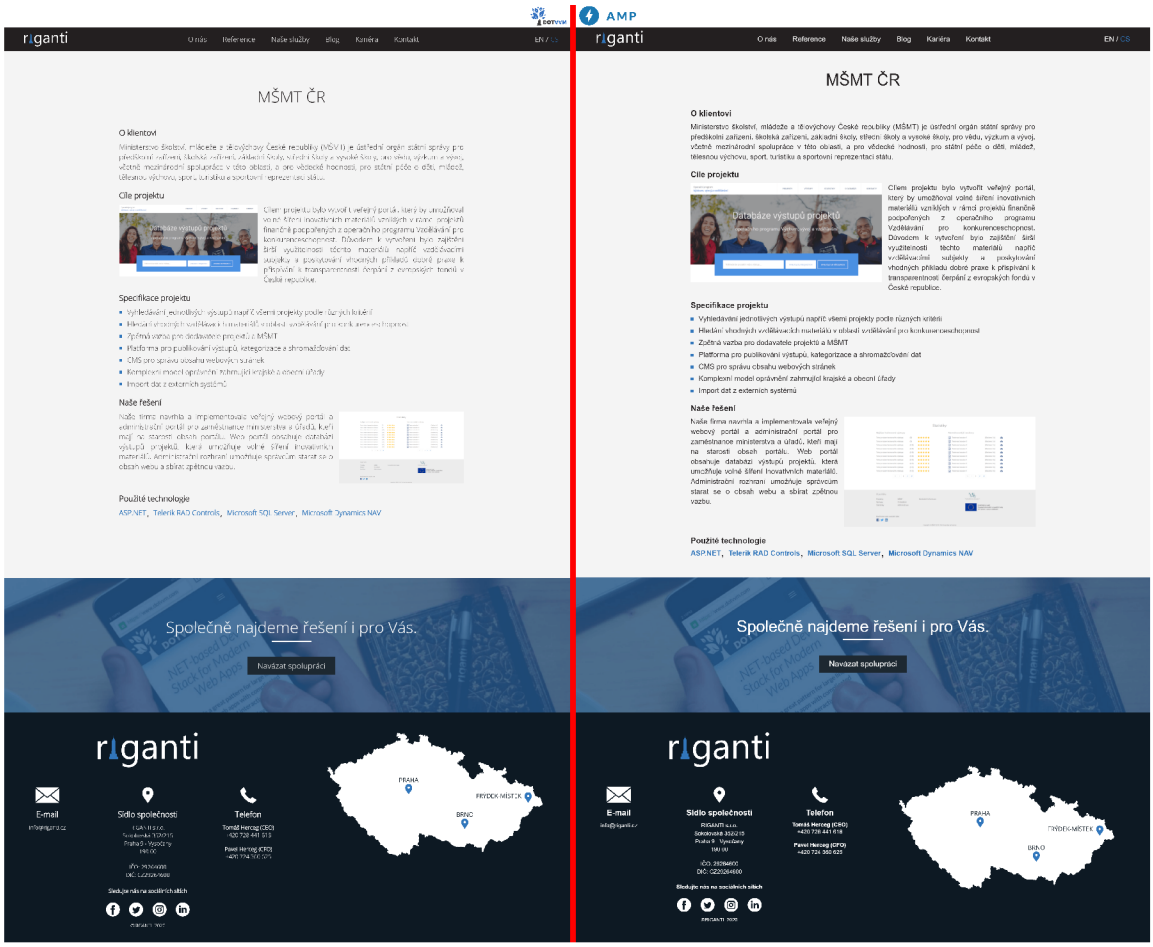
[3]Vektorová grafika nemá pevně dané rozměry. Je definovaná pomocí křivek, takže je možné ji jakkoliv zvětšit bez ztráty kvality.

[4]Kanonický odkaz je odkaz na stejnou stránku na jiné url.

s technologií AMP. Pro tento účel do stránky umístíme komponentu amp:Include, která se vyrenderuje pouze v rámci AMP verze. Dovnitř této komponenty můžeme již umístit náhradu za původní zobrazení obrázku. V našem případě to bude HTML link (<a>. . .) element, který nikam neodkazuje, ale slouží jen k tomu, aby byla zachována struktura elementů z DotVVM verze stránky. Dovnitř tohoto elementu umístíme původní html image element. Detail této úpravy je uveden v ukázce 5.4.

V rámci odstranění komponenty fancybox musíme vyřešit také odstranění javascriptových knihoven, které sloužily pro zobrazení fancybox komponenty. Tyto skripty obalíme komponentou amp:Exclude.

5.4 Zhodnocení výsledků



Obrázek 5.1: Porovnání původní stránky (vlevo) a její AMP verze (vpravo).

Na obrázku 5.1 je zobrazeno porovnání obou verzí. Stránky vypadají velmi podobně a jediný na první pohled zřetelný rozdíl je použitý font a mírné odlišnosti v pozici a velikosti obrázků. Další odlišností je absence možnosti zobrazit detail obrázků, kvůli absenci komponenty fancybox.

Jak na testovací sadě příkladů[5], tak při použití na reálných aplikacích se ukázalo, že DotVVM.AMP zvládne základní scénáře zcela automaticky. Při použití na komplikovanějších aplikacích je možné pomocí jednoduchých úprav docílit toho, že ke stránkám budou vygenerovány jejich AMP verze i bez výrazných ztrát funkcionality. AMP verze a původní verze nejsou vzhledově zcela identické, ale rozdíly jsou minimální a daly by se odstranit malou úpravou stránky a CSS, které se na danou stránku aplikuje.

[5]Sada testovacích stránek je dostupná v rámci zdrojového kódu nebo na https://github.com/MichalTichy/DotVVM.AMP/tree/master/TestSamples

Kapitola 6

Závěr

Cílem práce bylo navrhnout a vyvinout knihovnu, která by umožňovala překlad stránek napsaných v technologii DotVVM do technologie AMP. Hlavním důvodem pro existenci takového překladu je zvýšení rychlosti DotVVM stránek.

Překlad do technologie AMP je vhodný pouze pro stránky, které neobsahují velkou míru interaktivity. Do tohoto typu stránek často patří například úvodní a jiné prezentační stránky, u kterých nám na rychlosti nejvíce záleží, ale kvůli použití komplexního webového frameworku o část této rychlosti přicházíme. V rámci této práce vznikla knihovna DotVVM.AMP, umožnující vytvořit AMP verzi k již existující DotVVM stránek, která tento cíl plní. Tato knihovna je v době psaní tohoto textu dostupná ve verzi 0.1.1 jako nuget balíček[1]. DotVVM.AMP jsem otestoval jak na sadě testovacích stránek[2], tak na reálné webové aplikaci. Na všech testovaných stránkách se povedlo vytvořit jejich AMP validní verzi bez větších zásahů do původní stránky.

Díky tomu, že DotVVM.AMP rozšiřuje vykreslovací proces frameworku DotVVM, není možné dosáhnout rychlejší odpovědi ze serveru u AMP verze oproti DotVVM verzi. Vykreslení stránky na straně klienta rychlejší je. DotVVM.AMP musí navíc oproti DotVVM provést transformace a další náročné akce jako například stažení obrázků a jiných externích zdrojů. Tento problém s rychlostí není ve většině situací důležitý, protože většina přístupů na AMP stránku proběhne přes AMP Cache a budou tedy takřka instantní.

Problém s rychlostí bych chtěl vyřešit v rámci dalšího vývoje DotVVM.AMP. Řešením tohoto problému by byla cache, která by zaručila, že pro každou URL se AMP verze vygeneruje pouze jednou. Další možné rozšíření DotVVM.AMP by mohlo být nahrazení ručně psané validační logiky za parser validačních pravidel, které se používají přímo v rámci AMP infrastruktury[3].

Doufám že dokončením této práce vývoj DotVVM.AMP nekončí a do vývoje se zapojí další členové DotVVM komunity. Podpora technologie AMP v rámci DotVVM byla oznámena v rámci DotVVM Virtual Conference, takže věřím, že si DotVVM.AMP najde jak uživatele tak a další vývojáře [18].

[1] Knihovna DotVVM.AMP ve verzi 0.1.1 je veřejně dostupná na nuget.org. https://www.nuget.org/packages/DotVVM.AMP/0.1.1

[2] Sada testovacích stránek je dostupná v rámci zdrojového kódu nebo na https://github.com/MichalTichy/DotVVM.AMP/tree/master/TestSamples

[3] Validační pravidla jsou dostupná na https://github.com/ampproject/amphtml/tree/master/validator

Literatura

[1] Abner, L. *Google displaying AMP-based ads on regular web pages - 9to5Google*. Feb 2019. Dostupné z: https://9to5google.com/2019/02/20/google-amp-ads-regular-web/.

[2] AMP contributors. *Add custom fonts*. AMP Project. Dostupné z: https://amp.dev/documentation/guides-and-tutorials/develop/style_and_layout/custom_fonts/.

[3] AMP contributors. *AMP Boilerplate*. Dostupné z: https://amp.dev/boilerplate/.

[4] AMP contributors. *AMP HTML Specification*. AMP Project. Dostupné z: https://amp.dev/documentation/guides-and-tutorials/learn/spec/amphtml/#keyframes-stylesheet.

[5] AMP contributors. *Amp Script*. AMP Project. Dostupné z: https://amp.dev/documentation/components/amp-script/.

[6] AMP contributors. *AMP supported CSS*. AMP Project. Dostupné z: https://amp.dev/documentation/guides-and-tutorials/develop/style_and_layout/style_pages/.

[7] AMP contributors. *How AMP pages are cached*. AMP Project. Dostupné z: https://amp.dev/documentation/guides-and-tutorials/learn/amp-caches-and-cors/how_amp_pages_are_cached/.

[8] AMP contributors. *Supported Email Platforms*. Dostupné z: https://amp.dev/support/faq/email-support/.

[9] AMP contributors. *Why AMP Caches exist*. The AMP Blog, Jan 2017. Dostupné z: https://blog.amp.dev/2017/01/13/why-amp-caches-exist/.

[10] DotVVM team. *DotVVM Authentication and Authorization*. Dostupné z: https://www.dotvvm.com/docs/tutorials/advanced-authentication-authorization/latest.

[11] DotVVM team. *DotVVM Binding Syntax*. Dostupné z: https://www.dotvvm.com/docs/tutorials/basics-binding-syntax/latest.

[12] DotVVM team. *DotVVM Routing*. Dostupné z: https://www.dotvvm.com/docs/tutorials/basics-routing/latest.

[13] DotVVM team. *DotVVM Validation*. Dostupné z: https://www.dotvvm.com/docs/tutorials/basics-validation/latest.

[14] DotVVM team. *DotVVM ViewModels*. Dostupné z: https://www.dotvvm.com/docs/tutorials/basics-viewmodels/latest.

[15] DotVVM team. *Introduction*. Dostupné z: https://www.dotvvm.com/docs/tutorials/introduction/latest.

[16] Google developers. *Google AMP Cache*. Google. Dostupné z: https://developers.google.com/amp/cache/overview.

[17] Google developers. *Update AMP Content*. Google. Dostupné z: https://developers.google.com/amp/cache/update-cache.

[18] Herceg, T. *DotVVM Virtual Conference*. Apr 2020. Dostupné z: https://youtu.be/FwvKoSQKBPA?t=13168.

[19] Marvin, MachMetrics, Raphael, Chris, Sansar et al. *Average Page Load Times for 2018 - How does yours compare?* Feb 2018. Dostupné z: https://www.machmetrics.com/speed-blog/average-page-load-times-websites-2018/.

[20] Michálek, M. a Pokorný, R. *Vzhůru do AMP*. Martin Michálek - Vzhůru dolů, 2019. ISBN 978-80-88253-04-4.

[21] Microsoft. *5: Implementing the MVVM Pattern Using the Prism Library 5.0 for WPF*. May 2014. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/gg405484(v=pandp.40)?redirectedfrom=MSDN.

[22] ODonoghue, R. *AMP: building accelerated mobile pages: create lightning-fast mobile pages by leveraging AMP technology*. Packt Publishing, 2017. ISBN 978-1786467317.

[23] Project, T. A. *Turbocharging AMP*. The AMP Blog, May 2017. Dostupné z: https://blog.amp.dev/2017/05/18/turbocharging-amp/.

[24] Steinlauf, E. *The Speed Benefit of AMP Prerendering*. Aug 2019. Dostupné z: https://developers.googleblog.com/2019/08/the-speed-benefit-of-amp-prerendering.html.

[25] Zhou, R. *Building the perfect AMP Viewer, with Google Search*. Dostupné z: https://www.youtube.com/watch?v=hyd84z_qX8Q&feature=youtu.be.