Gymnázium, Praha 6, Arabská 14

Předmět Programování

MATURITNÍ PRÁCE

Knihovna pro záznam kotev v textu

Petr Chalupa IV.E

Praha, 2023/24

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V …… dne …… Petr Chalupa …….

ANOTACE

Práce se zabývá návrhem algoritmů, které by umožnili ukládání tzv. textových kotev (označení, poznámky aj.) do statického i dynamického textu (formátu XML) tak, aby je bylo možné opětovně do textu vložit i po jeho úpravě (a případně vyhodnotit chybu při vkládání). Takovýto program by pak měl být použitelný jako knihovna např. pro webové aplikace.

KLÍČOVÁ SLOVA

Algoritmus, textová kotva, XML, knihovna, webová aplikace

ANNOTATION

The thesis deals with the design of algorithms that would enable the storage of so-called text anchors (labels, notes etc.) in static and dynamic text (XML format) so that they can be re-inserted into the text even after its editing (and possibly evaluate the error during insertion). Such a program should then be usable as a library for e.g., web applications.

KEY WORDS

Algorithm, text anchor, XML, library, web application

OBSAH

1 ZADÁNÍ

Autor: Petr Chalupa

Téma: Knihovna pro záznam kotev v textu

Popis: Knihovna s algoritmy pro zapamatování vložených kotev (označení, poznámek aj.) do textu uživatelem. Cílem je, aby fungovala i pro dynamický text, tedy aby se kotvy automaticky přizpůsobovaly změnám v konkrétním textu (v rámci možností) a případně aby poskytla „zpětnou vazbu“ ohledně chyb - např. nepovedené zařazení do textu apod. Měla by fungovat na formátu XML (HTML) - použití primárně ve webových aplikacích, jako je například projekt Digitálního učebnicového systému, kterého jsem spoluautorem.

Platforma: JS/TS, Vue.js

2 Úvod

V dnešní době se mnoho textů přesouvá do digitální podoby. Druhy takových textů sahají od původně fyzických textů přesunutých do digitální formy, jako jsou například knihy – tento proces se nazývá digitalizace, až po zcela nově vytvořené texty, jako jsou například články. Takové texty jsou zpravidla dostupné na webu, a proto se nabízí možnost poskytnout uživatelům dané webové stránky/aplikace užitečné nástroje pro manipulaci s nimi.

Tato práce se zaměřuje na problematiku ukládání a opětovného vkládání textových kotev do statického i dynamického textu ve formátu XML. Pomocí navrhnutých algoritmů může uživatel označit klíčové body v textu a jejich snadnou editaci a aktualizaci i po provedených úpravách v původním textu.

Cílem této práce tedy je vytvořit knihovnu, která bude sloužit jako nástroj pro ukládání a manipulaci s textovými kotvami, a která bude využitelná ve webových aplikacích. Implementace této knihovny poskytne uživatelům flexibilitu a efektivitu při práci s textem, přičemž bude zajišťovat nejen správnost manipulace s kotvami, ale i detekce a řešení chyb, které by mohly nastat v průběhu procesu.

3 TERMINOLOGIE

Pro tuto práci je nutné definovat několik základních pojmů, které jsou pro její správné chápání zcela zásadní.

3.1 STATICKÝ TEXT

Statický text je chápán jako řetězec znaků, který se v průběhu času nemění. Takový text se v ideálním případě dá rozdělit na odstavce, věty, slova a znaky. Pracovat s takovým textem tedy lze předvídatelně.

3.2 DYNAMICKÝ TEXT

Dynamický text se odlišuje od statického pouze tím, že se v průběhu času mění, což může vést ke ztížení práce s ním a ke zkomplikování operací na něm prováděných – v extrémních případech až k jejich úplnému selhání.

3.3 TEXTOVÁ KOTVA

Textová kotva je pojem, který označuje specifický bod v textu, který je v ideálním případě nehybný. Takový bod je charakteristický zejména svojí odlišností od textu. Pokud rozšíříme tuto definici na právě dva sousední body, označující začátek a konec kotvy, začne mít význam i vizuální charakteristika kotvy – např. zabarvené pozadí. Z toho vyplývá například použití pro označování částí textu, což je hlavní motivace této práce.

3.3.1 Z POHLEDU PROGRAMU

Pro algoritmické operace pozbývá význam vizuální reprezentace kotvy, ale je zcela nutné, aby byla každá kotva unikátní např. pomocí UUID[1]. Kotvy jsou zároveň chápány jako nejmenší celky, které jsou pospojované do jednolitého bloku, který může pak kotvu reprezentovat vizuálně. Důvodem pro tento rozdíl v rozlišení na kotvy (Anchor) a bloky kotev (AnchorBlock) je to, že ve formátu XML se každá kotva vkládá do páru tagů, který ohraničuje např. odstavec textu, ovšem označený text může přesahovat mezi více než jedním takovým úsekem textu.

[1] UUID – Universally Unique Identifier ~ Univerzálně Unikátní Identifikátor

4 ALGORITMUS VYTVOŘENÍ KOTVY

K vytvoření kotvy je nejdříve potřeba impulsu od uživatele. Takový impuls musí následovat po označení libovolné oblasti textu – horizontálně i vertikálně. V případě, že by žádné označení neexistovalo, není co vytvářet. Vše je situováno tak, aby bylo možné tvořit kotvy zvlášť pro každý jeden definovaný blok textu (rootNode). Pokud výběr přesahuje za hranice tohoto bloku, je selekce nesprávná a algoritmus končí.

V případě, že výběr v textu je validní, je možné pokusit se o vytvoření kotvy. Algoritmus zpracovává jednotlivé části výběru (objektu Selection) – tzv. Range. Nejprve je nutné získat nejmenší možný blok, který obsahuje celý Range – tzv. commonAncestorContainer. Zbytek práce může probíhat už pouze v tomto bloku. Z tohoto bloku jsou dále vybrány všechny jeho potomci – tzv. Node, které jsou textové (TEXT_NODE) a zároveň jsou součástí označení. Ty jsou vkládány do pole v pořadí jejich výskytu a následně jsou vyřazeny (nahrazeny null) ty, které jsou již součástí nějakého Anchoru – nesmí v cestě k němu skrze DOM[2] existovat žádný další Anchor, čímž je zabráněno překryvům mezi kotvami.

Pro každý souvislý úsek těchto potomků je pak vytvořen AnchorBlock, který kromě referencí na jednotlivé menší části nese i další přídavné informace, které jsou pro celou kotvu společné (barva apod.). Pro první Anchor prvního AnchorBloku je použit startOffset z Range a pro poslední Anchor posledního AnchorBlocku je použit endOffset z Range. Zbytek pokrývá všechen doposud nepokrytý text mezi nimi. Nakonec AnchorBlock spojí všechny své Anchory (leftJoin, rightJoin). Tím algoritmus končí.

Přidat diagram!!!

[2] DOM – Document Object Model ~ Objektový Model Dokumentu

5 ALGORITMUS ULOŽENÍ KOTVY

Něco málo k serializaci

6 ALGORITMUS VLOŽENÍ KOTVY

Pár checků a pak to samé jako vytvoření

7 SPECIÁLNÍ FUNKCE

xPath, invertHexColor…

8 KNIHOVNA

Nezapomenout na NPM! Zmínit README!

8.1 ARCHITEKTURA

8.2 POUŽITÍ

9 DEMO

9.1 FUNKCE

9.2 GENEROVÁNÍ TEXTU

10 ZÁVĚR

11 POUŽITÉ ZDROJE

Aktuální dokument neobsahuje žádné prameny.

12 SEZNAM OBRÁZKŮ

Nenalezena položka seznamu obrázků.

13 SEZNAM UKÁZEK KÓDU

Nenalezena položka seznamu obrázků.