Studentská Konference Fakulty Mechatroniky, informatiky a mezioborových studií
10. červen 2013, Liberec, Česká republika

Návrh a implementace architektury pro tvorbu single page web aplikací

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Bc. Ondřej Smola
Ing. Jan Silovský, PhD.
FM, ITE

Abstract

The aim of this work is the design and implementation of application framework for creating single page web applications. Described framework is currently used during development of application at Faculty of Mechatronics TUL. Developed framework provides an infrastructure that greatly simplifies the creation of applications and solves most of the major problems related to design and implementation of single page web applications.

Motivace

Mezi hlavní výhody single page web aplikací patří možnost interaktivity na úrovni desktopových aplikací. Mezi další výhody patří značné snížení množství přenášených dat, jelikož jsou přenášena pouze data potřebná pro aktualizaci pohledu namísto všech dat nutných pro zobrazení cílové webové stránky. Redukce množství přenášených dat a požadavků zároveň může vést ke snížení nároků na hardware.

Z pohledu uživatele patří mezi největší výhody především citelné zvýšení rychlosti odezvy až na úroveň, kdy se může pohybovat po webové stránce bez nutnosti neustále čekat na reakci webové aplikace. Tvorba single page aplikací ale zároveň přináší celou řadu problémů, se kterými se vývojář doposud nemusel zabývat. Mezi hlavní problémy patří především přepínání jednotlivých stránek a řešení vzájemné komunikace mezi komponentami.

Cíl

Mezi hlavní cíle této práce patří návrh a implementace aplikačního rámce pro tvorbu single page web aplikací. Mezi hlavní úkoly při návrhu patří definice způsobu tvorby a uspořádání stránky. S ohledem na zvolené rozdělení potom definovat vzájemnou komunikaci jak v rámci komponent tak i mezi jednotlivými stránkami. Mezi poslední úkol z hlediska návrhu patří definice způsobu jakým budou jednotlivé komponenty načítány.

Implementační část se bude zabývat konkrétními algoritmy použitými při řešením vytvořeného návrhu společně s popsáním realizovaných postupů.

Návrh

Ovladač aplikace za chodu načítá a mění perspektivy, které utvářejí jednotlivé pohledy na celou aplikaci. Perspektivy obsahují moduly, které mohou dále obsahovat další moduly. Komunikaci jak mezi perspektivami, tak mezi moduly je zajišťována rozesíláním zpráv kontroléry.

Perspektiva

Perspektiva je reprezentována jako kořen stromu modulů. V klasickém pohledu na webové stránky si lze perspektivu představit jako jednu klasickou webovou stránku.

Modul

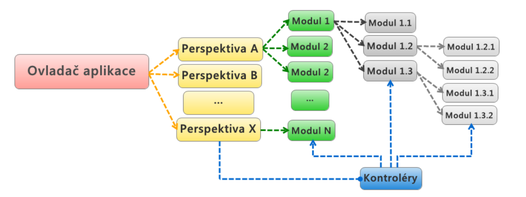
Modul slouží pro reprezentaci jednotlivých komponent aplikace a jedná se o základní stavební prvek celého rámce. Reprezentace modulu je řešena objektem obsahujícím pouze reference na funkci pro inicializaci modulu a funkci pro příjem zpráv. S okolními moduly probíhá komunikace pomocí kontroléru a identifikátoru přiděleného během inicializace modulu.

Kontrolér

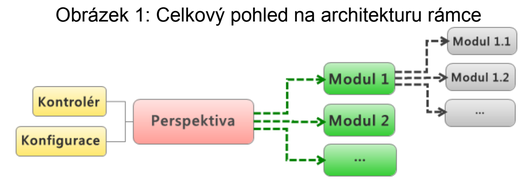
Hlavním úkolem kontroléru je řízení a zprostředkovávání komunikace mezi moduly. Kontrolér je jediná část rámce, která nese informaci o přítomných modulech a možnostech komunikace mezi nimi. Na základě konfigurace provede ověření zprávy a její přeposlání.

Konfigurace modulů

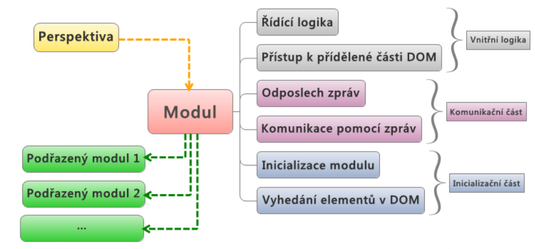
Návrh konfigurace modulů se snaží o striktní oddělení místa definice modulů od jejich umístění v hierarchii. Definicí modulu se rozumí cesty k souborům (soubory jazyka Javascript, kaskádové styly) společně s umístěním podřazených modulů. Umístění v hierarchii určuje v jakém kontextu (z hlediska umístění) bude modul umístěn a je zároveň místem konfigurace.



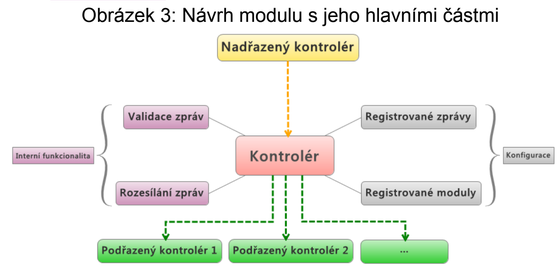
Obrázek 1: Celkový pohled na architekturu rámce



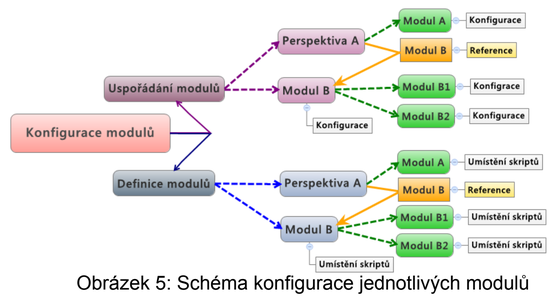
Obrázek 2: Návrh perspektivy s hierarchií modulů



Obrázek 3: Návrh modulu s jeho hlavními částmi



Obrázek 4: Návrh hierarchie kontrolérů



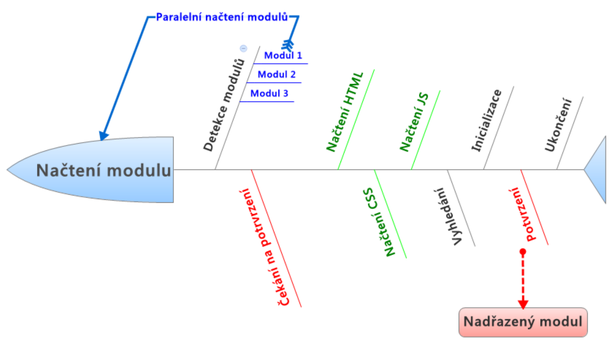
Obrázek 5: Schéma konfigurace jednotlivých modulů

Implementace

Hlavními body z hlediska implementace je způsob načtení modulu, způsob rozesílání zpráv mezi nimi. Pro zajímavost je uvedeno schéma modulu pro asynchronní komunikaci.

Načtení modulu

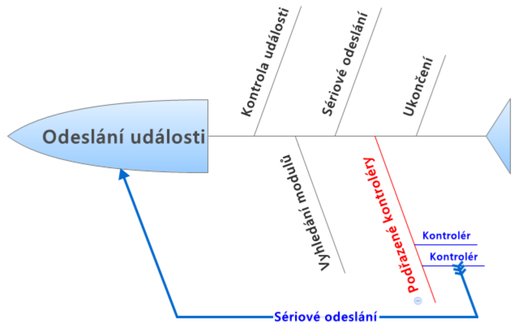
Načtení modulu je realizováno pomocí rekurzivní algoritmu vycházející z *bottom-up* návrhu, neboli nejprve jsou načteny moduly bez závislostí a následně postupně všechny další moduly, které již mají načteny všechny závislosti.



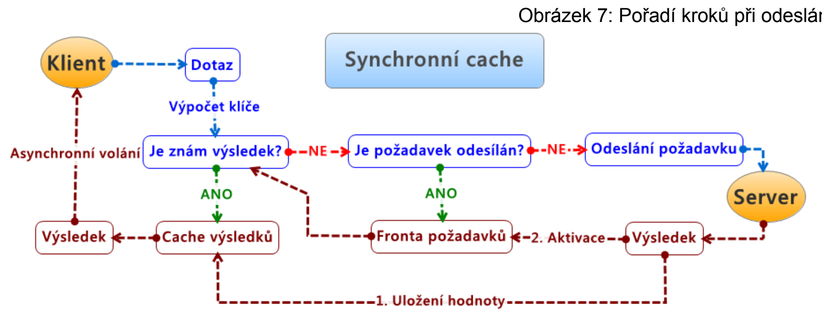
Obrázek 6: Pořadí kroků při načítaní modulu

Odeslání události

Událost je v systému realizována pomocí zprávy. Zpráva je odeslána pomocí kontroléru, který následně rozešle zprávu všem známým modulům. V případě, že cílový modul obsahuje vlastní kontrolér, je zpráva kontrolérem dále rozeslána. Implementace zároveň částečně řeší vznik nekonečných cyklů možností vyloučit zdrojový modul z cílových pomocí identifikátoru.



Obrázek 7: Pořadí kroků při odeslání události



Obrázek 8: Stavový automat synchronní cache pro obsluhu asynchronní dotazy

Závěr

Stávající řešení je plně funkční a jeho vývoj probíhá v souladu s tím, jak se vyvíjejí požadavky na aplikaci pro niž byl rámec původně navržen. Aktuálně probíhá implementace správy historie a řešení doby validity záznamů v systému cache. Mezi budoucí cíle patří podpora pro internacionalizaci.

Dílčí nedostatky budou dále řešeny společně s reálným nasazením aplikace, jelikož především při návrhu a implementaci knihoven obecně je základem reakce uživatelů a většina problémů je objevena až při využití více uživateli.

Reference

[1] O'REILLY MEDIA / YAHOO PRESS. JavaScript: The Good Parts [online]. 2008. ISBN978-0-596-15873-6. Dostupné z: http://shop.oreilly.com/product/9780596517748.do

[2] O'REILLY MEDIA. JavaScript Patterns [online]. 2010. ISBN 978-0-596-80677-4. Dostupné z: http://shop.oreilly.com/product/9780596806767.do

[3] Javascript. *MOZILLA DEVELOPER NETWORK* [online]. 2013 [cit. 2013-04-29]. Dostupné z: https://developer.mozilla.org/en-US/docs/JavaScript

Kontakt

Bc. Ondřej Smola
ondrej.smola@tul.cz

Prezentace této práce byla podpořena z projektu SGS 2013