

Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

jQuery – plugin pro manipulaci se stromem štítků

Plzeň, 2013

Libor Vohanka

1 Úvod.....	3
2 Použité technologie.....	4
2.1 Technologie pro klientskou část	4
2.1.1 HTML a jiné značkovací jazyky.....	4
2.1.2 CSS	26
2.1.3 Javascript	27
2.1.4 Coffeescript.....	28
2.2 Serverové technologie.....	28
2.2.1 PHP	28
2.2.2 MySQL	28
2.3 Přenos dat klient - server	28
2.4 Vývojová prostředí	29
2.5 Záloha dat	29
3 Použité frameworky.....	31
3.1 jQuery	31
3.2 Twitter Bootstrap	32
4 Podobné práce.....	33
5 Tvorba jednoduchého jQuery pluginu	34
6 Adresářová struktura pluginu.....	36
7 Databáze.....	38
7.1 Návrh použité databáze.....	38
7.2 Záměna za vlastní	38
7.3 Práce s databází.....	38
8 Plugin jqTagTree	39
8.1 Před vykreslením stromu	39

8.2 Vykreslování stromu.....	39
8.3 Kontextové menu.....	41
8.4 Změny položek stromu	41
8.4.1 Přidávání	42
8.4.2 Úprava.....	44
8.4.3 Mazání	44
8.4.4 Zobrazení predikátů	45
9 Tvorba ukázkové webové aplikace.....	47
10 Závěr	48
11 Použitá literatura	49

1 Úvod

Framework jQuery je velice oblíbený hlavně kvůli jednoduchosti použití a jeho rozšiřitelnosti. Framework je z obecného hlediska jakési rozšíření stávajícího jazyka o funkce a proměnné, které programátorům ulehčují práci v daném jazyce. jQuery je nadstavbou jazyka Javascript.

V současné době je velké množství programátorů, kteří používají Javascript bez frameworku. Ještě více je však používáno jQuery. Proto lze na internetu nalézt nepřeberné množství zásuvných modulů, neboli pluginů právě do jQuery.

Díky rozšířenosti jQuery lze na internetu nalézt velké množství pluginů, rozšiřující jeho možnosti různými směry. Velké množství z nich různým způsobem zobrazuje fotografie ve fotogaleriích, další animují rozevírací menu stránek, některé spravují kalendáře, nebo okna pro výběr data přímo v prohlížeči. Dále jsou pluginy které doplňují text psaný uživatelem, tvoří slidery spravují záložky, animují progressbary a v neposlední řadě zobrazují stromy.

Cílem této práce je plugin do frameworku jQuery, který přehledně zobrazí štítky uložené v databázi v prohlížeči. Důvodem pro tvorbu tohoto programu je specifická struktura tabulek databáze. Dalšími vlastnostmi pluginu jsou přidávání nových štítků, změna stávajících, jejich mazání včetně potomků a správa predikátů štítků.

Ve druhé kapitole bude popsán přehled technologií a jazyků použitých při tvorbě zásuvného modulu. V další kapitole zaměřím na pluginy pro manipulaci se stromy které již byly publikovány na webu, jejich výhody a nevýhody. Čtvrtá kapitola shrnuje použité frameworky, pátá obsahuje návod na tvorbu jednoduchého pluginu do jQuery. V šesté je popsána struktura souborů a složek pluginu. Obsahem sedmé kapitoly je struktura databáze, které bylo nutné se držet. Osmá kapitola popisuje implementaci metod a algoritmů, kterých bylo v práci použito.

2 Použité technologie

2.1 Technologie pro klientskou část

2.1.1 HTML a jiné značkovací jazyky

HTML je nejrozšířenější značkovací jazyk, který je používán pro tvorbu webových prezentací. Stavebními kameny jsou značky, neboli tagy, většinou párové. Mezi ně jsou uzavírány části dokumentu, které mají být nějakým způsobem odlišeny. Cílem značek je přidání sémantické informace, která zvýší jak lidskou, tak strojovou čitelnost dokumentu.

- některé značky formátují vzhled dokumentu – například span
- jiné značky, například div, slouží k poskládání prvků na stránce
- značky table, se používají k výpisu tabulkových dat

Vše ale prochází určitým vývojem a s příchodem čtvrté verze jazyka HTML byl uveden jazyk XML, více v kapitole 9.3 Přenos dat klient - server. Později byl uveden jazyk XHTML syntaxí vycházející z XML. Je však jen úpravou HTML. Následuje stručný výpis zásadních změn, co požaduje XHTML oproti HTML¹:

- Zákaz křížení tagů
- Párové tagy jsou párové povinně
- Nepárové tagy musejí končit lomítkem
- Tagy, atributy ani hodnoty atributů nesmějí být psány velkými písmeny
- Hodnoty atributů musejí být v uvozovkách

Dalším vývojovým stupněm je HTML 5. Sestává z mnoha částí, a přesto, že ještě není jeho definice zcela hotova, prohlížeče již HTML5 více či méně podporují. Na stránkách <http://html5test.com> lze otestovat kterýkoli prohlížeč na jeho podporu. V závislosti na tom, kolik funkcí HTML 5 podporují, dostanou prohlížeče bodové ohodnocení do celkového počtu 500 bodů. Pořadí a skóre nejpoužívanějších prohlížečů dle výsledků z výše jmenované webové adresy k datu 15.4.2013 viz Obrázek 9.2: Změna klávesnice na dotykových zařízeních

¹ http://www.w3schools.com/html/html_xhtml.asp

2.1.2 CSS

Kaskádové styly slouží k definici vzhledu stránek vytvořených v HTML, XHTML, nebo XML. Díky nim mohou být elementy po webové stránce posouvány, upravovány, měněna jejich velikost nebo barva. U textu lze změnit font, barvu či velikost. To je samozřejmě jen výběr toho, co kaskádové styly umí. Dokáží ale také výrazným způsobem zlepšit přehlednost webu. Existují tři způsoby, jak vložit styly do dokumentu. Přímou do tagu `<div style = "color : red">...</div>`, mezi tagy `<style></style>` a třetí nejpoužívanější možnost je vložit vlastnosti do externího souboru, na který se odkáže v hlavičce HTML souboru.

Specifikace třetí verze kaskádových stylů také ještě není zcela hotova. O podpoře jednotlivých vlastností v prohlížečích informuje server W3Schools. Mezi významnější zjednodušení při tvorbě vzhledu internetových stránek patří možnost přidat elementům stíny, zaoblit rohy nebo třeba možnost definice průhlednosti elementů.

2.1.3 Javascript

Javascript je skriptovací jazyk běžící na straně klienta. Existují také řešení spouštěné na straně serveru jako například Node.js. Slovo java je v názvu jazyka jen z marketingových důvodů a s programovacím jazykem Java má společnou pouze syntaxi. Javascript stránkám přidává na dynamičnosti. Vhodné je využití při tvorbě menu a různých kontejnerů, které se po přejetí myši rozevřou a ušetří tak místo na obrazovce. Lze ho také použít při tvorbě webového editoru textu.

Své využití má i při validaci formulářů. Je-li odeslán s daty ve špatném formátu, zobrazí upozornění bez komunikace se serverem a uživatel tak dostane informaci o datech rychleji. Javascript si ale uživatel v prohlížeči může vypnout, a proto je nutné odeslaná data validovat ještě na straně serveru. Je vhodné při práci s Javascriptem používat některý z frameworků. Obsahují množství funkcí výrazně zjednodušující práci s dokumentem.

Asynchronní Javascript a XML technologie (AJAX) je používána pro dynamické načítání obsahu stránek bez nutnosti znovu načítat celou internetovou stránku. Lze ji také používat pro pouhé načítání statických HTML stránek, po kliknutí na odkaz načíst

příslušný formulář nebo lze takto volat PHP kód, který může například vybrat určitá data z databáze, předat je dále Javascriptu ke zpracování.

2.1.4 Coffeescript

2.2 Serverové technologie

2.2.1 PHP

Skriptovací jazyk PHP je zpracováván na straně serveru a je tak hlavním článkem při validaci formulářů odesílaných uživatelem. Od čtvrté verze lze v PHP programovat objektově. Do páté verze byl však objektový model PHP přepracován. PHP je pravděpodobně nejrozšířenější skriptovací jazyk pro tvorbu webu, a díky tomu pro něj existuje mnoho knihoven a frameworků, například Nette nebo Zend. Syntaxí spadá do rodiny C/Java.

2.2.2 MySQL

MySQL je databázový systém vycházející z normy jazyka SQL. Tuto normu však striktně nerespektuje. MySQL je používán na většině webových hostingů. Od převzetí firmy MySQL firmou Oracle na podzim roku 2010 je nabízen jak pod bezplatnou, tak pod placenou licenci. Od verze 5.0 podporuje například i uložené procedury, trigery a pohledy.

2.3 Přenos dat klient - server

Nejpoužívanější formáty pro přenos dat mezi klientem a serverem jsou JSON a XML. Oba formáty lze použít jak v internetových aplikacích, tak v off-line aplikacích. V současné době je při přenosu dat více využíván spíše JSON. Je tomu tak převážně kvůli objemu dat, který je potřeba přenést při komunikaci. Formát XML vychází ze syntaxe HTML. To znamená, že veškerá informace o datech, je uložena mezi značkami, které si uživatel může definovat vlastní. V Kód 9.2 je uveden příklad dat zapsaných

pomocí XML. Značka osoba s atributy jmeno a prijmeni. Kód 9.3 dává za příklad, jak vypadají data odesílána pomocí JSON.

```
<osoba jmeno="Petr" prijmeni="Novak">  
</osoba>
```

Kód 9.2: Příklad XML kódu

```
"jmeno": "Petr", "prijmeni": "Novak"
```

Pokud porovnáme Kód 9.2 a Kód 9.3, lze si všimnout, v tuto chvíli nepatrných, rozdílů ve velikosti jednotlivých kódů. Pokud se přenáší větší data, tento rozdíl bude znatelnější a JSON může uživateli ušetřit velké množství času.

Další výhodou JSONu je jednoduchost interpretace přijatých dat. PHP dokonce obsahuje funkci pro převod mezi polem a JSON strukturou. Tuto funkci má v sobě také framework jQuery pro Javascript. Z těchto důvodů jsem si pro přenos dat vybral právě JSON.

2.4 Vývojová prostředí

Je dobrým zvykem psát kód v programu, který alespoň zvýrazňuje syntaxi jazyka, ve kterém programátor píše. Zástupcem nejjednodušších prostředí je pro platformu Windows například program Notepad++. Pro Mac OS X je to pak třeba TextMate. Obě tato prostředí rozpoznají syntaxi velkého množství skriptovacích, značkovacích a programovacích jazyků. Mimo jiné také Javascript, HTML, CSS a PHP, které se většinou používají při tvorbě jQuery pluginů. Já pro práci na svých projektech používám Netbeans. Ten umí kromě zvýrazňování klíčových slov, také doplňovat názvy proměnných a funkcí. Má také správu projektů a další funkce, které usnadňují vývoj.

2.5 Záloha dat

Před samotným vývojem pluginu bylo nezbytné vyřešit problematiku zálohování zdrojových kódů a textu bakalářské práce. Způsobů, jak zálohovat, je hned několik. Od vypalování záloh na CD, přes kopírování dat na externí disk, zrcadlení disků, nahrávání dat na internet do externích úložišť až po verzovací systémy. Pro mé potřeby a potřeby

zadavatele je nejvhodnější poslední jmenovaná možnost. Druhý systému pro správu verzí je více. Pro příklad uvedu čtyři nejpoužívanější:

- CVS
- SVN
- GIT
- Mercurial

Jedno z nejhlavnějších dělení je na centralizované a decentralizované. CVS, stejně jako SVN spadají do kategorie centralizovaných verzovacích systémů, což v principu znamená, že mají jeden server, kde jsou nahrány veškeré zdrojové kódy. Jestliže vše funguje jak má, tak s tím není problém. Pokud se ale stane, že server spadne nebo nefunguje připojení k internetu, tak uživatelé ztrácejí k repositáři přístup a často to znamená nemožnost další práce. Je-li repositář decentralizovaný, jako v případě Mercurial a GITu, tak každý kdo s ním pracuje, má staženou jeho verzi u sebe v počítači a lze tím pádem pracovat i na cestě bez připojení k internetu. Po návratu do sítě změny odešle, případně si stáhne úpravy ostatních členů projektu, a repositář dále zůstane aktuální.

Na konci minulého roku byl společností Redmonk proveden výzkum nejpoužívanějších verzovacích systémů v porovnání s rokem 2010 (viz <http://redmonk.com/sogady/2012/11/05/dvcs-2012/>). Je z něho patrné, že jsou stále nejvíce používány centralizované systémy. Jsou však pomalu na ústupu. Nejpoužívanějšími zástupci jsou SVN a Git

Díky službě <http://github.com>, kde lze umístit a verzovat jakýkoli kód zdarma v případě, že je repositář veden jako veřejný. Na základě domluvy s vedoucím jsem zvolil tuto variantu. Jelikož je mou povinností mít bakalářskou práci veřejně dostupnou je služba github zcela ideální. Další výhodou je pak jednoduchá kontrola stavu práce vedoucím.

3 Použité frameworky

3.1 jQuery

jQuery je Javascriptový framework, který si klade za úkol zjednodušit programování v Javascriptu a sjednotit jeho zobrazování v prohlížečích. Od jeho založení v roce 2006 jeho popularita raketově vzrostla. Nyní je používán nejnavštěvovanějšími webovými servery, jako například <http://amazon.com>, <http://microsoft.com>, <http://twitter.com>. Více než 62% nejpopulárnějších internetových stránek [1] používá jQuery.

Jednou z nejpoužívanějších funkcí jQuery je výběr elementu na stránce. Cokoli je potřeba v dokumentu změnit, musí to být nejdříve vybráno. Zde je porovnání, jak vypadá vybrání prvku s `id = menu`.

```
document.getElementById('menu'); //Javascript  
$('#menu')                       //jQuery
```

Další funkce poskytované jQuery :

- Změna DOM (Document Object Model) elementů na stránce – přidávání či mazání html tagů ze stránky
- Úprava kaskádových stylů po výběru elementu
- Funkce pro práci s AJAXem
- Systém událostí
- Animace a efekty
- Snadné přidávání pluginů

Framework jQuery je postaven tak, aby vývojářům co možná nejvíce zjednodušil tvorbu pluginů. Nejslavnějším pluginem pro jQuery je jQuery UI vytvořený samotnou společností jQuery. Jedná se o sadu Javascriptových funkcí pro tvorbu uživatelského rozhraní. Mezi nejzajímavější patří funkce pro posuv elementů po stránce, dále obsahuje různé doplňky jako tooltip, datepicker nebo autocomplete a v neposlední řadě přidává oproti jQuery řadu efektů zobrazování a skrývání elementů. jQuery UI je stejně jako jQuery poskytován pod MIT licenci.

3.2 Twitter Bootstrap

Twitter Bootstrap je kolekce nástrojů pro tvorbu webových aplikací. Během necelého půl roku od vydání se stal nejpopulárnějším projektem na GitHubu. Jednou z nejzásadnějších věcí, které Bootstrap řeší, je nekompatibilita zobrazení různých prvků na stránce v různých prohlížečích. Pokud programátor použije k tvorbě designu svého webu výhradně stylů Bootstrapu, nemusí se nekompatibility bát.

Bootstrap je knihovna obsahující naprostou většinu stylů, které jsou zapotřebí ke tvorbě jednoduchých internetových prezentací. Mezi základní styly, díky kterým stojí zato Twitter Bootstrap použít, patří:

- Rozložení prvků na stránce
- Základní šablony pro web
- Responsivní design
- Typografie

Dále obsahuje styly pro menu, tlačítka, stránkování, štítky nebo náhledy obrázků. Tato knihovna je dále dodávána se sadou ikon Glyphicons. Součástí Bootstrapu je i několik jQuery pluginů. Bootstrap se během necelých dvou let stal velmi oblíbený. Svědčí o tom nejen stránky <http://builtwithbootstrap.com/>, které shromažďují stránky postavené na základu Bootstrapu, ale i web <https://wrapbootstrap.com/>, který nabízí k prodeji šablony založené opět na Bootstrapu.

4 Podobné práce

Pluginů pro zobrazení, či úpravu stromu lze na internetu nalézt velké množství. Dále budou některé z nich představeny a uvedeny jejich přednosti a nedostatky.

jQuery tree view

Plugin pro pouhé zobrazování stromové struktury, a jednoduché přidávání složek a mazání položek stromu a složek. Využívá funkcí `slideUp` a `slideDown` pro zobrazování respektive skrývání větví. Kliknutím na prvek stromu je uživatel přesměrován na stránku v odkazu. Podporuje funkci rozevřít a zavřít všechny větve stromu. První úroveň je zobrazena s jiným formátem písma než ostatní. Vývoj tohoto projektu byl však už zastaven.

jqTree

Součástí funkcionality jqTree je zobrazování a skrývání položek stromu a jejich přesouvání technikou `drag and drop`. Data pro strom lze načíst pomocí funkce `jquery` `getJSON`, nebo z pole které má stejnou strukturu. Kód pluginu je napsán v Coffeescriptu.

jsTree

Plugin `jstree` má více funkcí než předešlé pluginy a to, nejen zobrazování a přesouvání technikou `drag and drop`, ale také přidávání složek, souborů, jejich přejmenování a mazání. Dále plugin také umožňuje vyhledávání ve stromu.

5 Tvorba jednoduchého jQuery pluginu

Základní kostra dokumentu v HTML 5, kterou budeme dále rozšiřovat, vypadá následovně.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=UTF-8">
  </head>
  <body>
  </body>
</html>
```

Kód 12.1: Šablona HTML dokumentu.

Vytváření pluginu do jQuery je velmi jednoduché. Abychom mohli používat jQuery, stačí do hlavičky dokumentu přidat následující řádek.

```
<script src="http://code.jquery.com/jquery-latest.min.js"
type="text/JavaScript"></script>
```

Kód 12.2: Vložení Javascriptu.

Ten načte zdrojové kódy jQuery frameworku ze stránek jquery.com a vloží jej na naše. Má to tu výhodu, že někteří uživatelé přicházející z jiných stránek, kde jQuery také používají, a tudíž tyto kódy mohou mít staženy a nemusí je stahovat znova. Nyní můžeme všechny funkce jQuery používat dle libosti. Řekněme, že budeme chtít vytvořit plugin, který nám jednou na vteřinu ukáže nějaký text. Mohlo by to být třeba upozornění, že jsme se na stránky přihlásili.

Nejdříve budeme potřebovat prvek, který budeme zobrazovat a skrývat. V našem případě div. To je jen jednoduché ohraničení nějakého textu či bloku kódu. Následně ho budeme pomocí Javascriptu vybírat. Náš div bude vypadat následovně.

```
<div>Jste přihlášen/a.</div>
```

Kód 12.3: Zobrazovaný div.

Tímto jsme s HTML hotovi a můžeme se pustit do tvorby pluginu. Budeme ho vkládat do hlavičky hned za načtení jQuery. Je nezbytné, aby nejdřív proběhlo načtení jQuery. V opačném případě by plugin nefungoval, jelikož by nebylo do čeho vkládat.

Veškerý kód Javascriptu začíná párovým tagem `<script>` a končí `</script>`. Dále bude následovat znak dolaru “\$”, který odpovídá volání funkce jQuery. Má pouze funkci zkrácení zápisu. Za ním v kulatých závorkách bude argument funkce, neboli takzvaný selektor, kterým vybereme potřebný element na stránce. Nejprve je potřeba počkat, než se celá stránka stáhne k uživateli a připraví k použití. Výběrem dokumentu a zavolání funkce `ready` to můžeme považovat za hotové. Viz Kód 12.4: Funkce čekající na dokončení načítání web stránky. To je nezbytné, protože jinak bychom mohli skrývat něco, co tam ještě není. Tento případ nemusí být tak závažný, konzole by vypsala nějakou chybu, kterou uživatel při prohlížení stránek ani neuvidí. Lze si ale představit případ vedoucí k horším výsledkům. Navíc se může projevit jen jednou za čas v závislosti na rychlosti načtení dokumentu a odhalení těchto chyb by nemuselo být jednoduché. Dále budeme vše psát do anonymní funkce vytvořené v argumentu funkce `ready`.

```
$(document).ready(function(){});
```

Kód 12.4: Funkce čekající na dokončení načítání web stránky.

Plugin budeme psát do jmenného prostoru k tomu vyhrazenému frameworkem jQuery, a to `jQuery.fn.jmenoPluginu`, do kterého uložíme anonymní funkci s kódem pluginu. Je dobré kvůli bezpečnosti uzavřít ho ještě do anonymní funkce, díky které nemusíme mít strach, že by náš plugin kolidoval s nějakou jinou knihovnou nebo pluginem.

```
(function($){  
/*kód našeho pluginu v anonymní funkci*/  
})(jQuery);
```

Kód 12.5: Anonymní funkce v jQuery pluginu.

Upozornění uživatele pomocí jednoduchého zobrazení a skrytí textu provedeme za pomoci tří funkcí jQuery. Nejdříve prvek skryjeme funkcí `hide()`, jako že tam není. Pak ho pomalu zobrazíme, `slideDown(1000)` a nakonec zase schováme `slideUp(1000)`. Argumentem funkcí `slide` je čas v milisekundách, jak dlouho má skrývání, popřípadě zobrazování, trvat. Tedy jednu vteřinu každé. Plugin máme hotov. Pokud bychom v tuto chvíli zkusili stránku načíst, nic by se nestalo. Musíme ho totiž zavolat, jinak by nevěděl kdy a kde pracovat.

```
$('#div').helloWorldPlugin();
```

Kód 12.6: Volání testovacího pluginu.

Následuje celý zdrojový kód, který po uložení do souboru s příponou HTML a spuštění v prohlížeči, zobrazí a zase skryje text „*Jste přihlášen/a*“.

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;
  charset=UTF-8">
  <script src=http://code.jquery.com/jquery-latest.min.js
  type="text/Javascript"></script>
  <script>
    $(document).ready(function() {
      (function($) {
        $.fn.helloWorldPlugin = function() {
          this.hide();
          this.slideDown(1000);
          this.delay(1000);
          this.slideUp(1000);
        };
      })(jQuery);
      $('#div').helloWorldPlugin();
    });
  </script>
</head>
<body>
  <div>Jste přihlášen/a.</div>
</body>
</html>
```

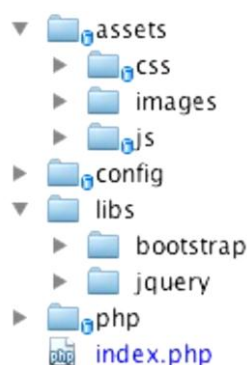
Kód 12.7: Jednoduchý jquery plugin.

6 Adresářová struktura pluginu

Umístění jednotlivých souborů je klíčové pro přehlednost jak moji, tak potenciálních uživatelů mého pluginu, kteří ho budou chtít implementovat do svých webových stránek.

Kořenový adresář obsahuje čtyři složky a jeden soubor. Index.php musí být vždy hned v první úrovni struktury. Nad ním je složka php, kam spadá veškerý kód v jazyce PHP. V mém případě je to pouze třída pracující s databází a soubory volané AJAXem při zobrazování nebo úpravě stromu. Ve složce libs se nacházejí knihovny použité v

pluginu. Jedná se o jQuery, jQuery UI a Twitter Bootstrap. Dále jsem použil sadu funkcí s názvem Bootstrap-Select pro úpravu vzhledu selectboxu ve stylu Twitter Bootstrap. Složka config obsahuje nastavení konstant, které se týkají PHP a přístupů do databáze. Na konci je složka assets, do které patří kaskádové styly, javascriptové soubory a obrázky. Je tam také umístěn kód mého pluginu s názvem `jqTagTree.js`. Vzhled adresářové struktury viz Obrázek 13.1.



Obrázek 13.1: Adresářová struktura pluginu

7 Databáze

----- DODĚLAT -----

7.1 Návrh použité databáze

Jeden z požadavků programu bylo zachování stávající struktury databáze. Ta obsahuje tři tabulky. První v sobě udržuje informace o štítku, jedinečné id, viz Tabulka 14.1 a v druhé jsou uloženy informace o vazbách mezi štítky, viz Tabulka 14.2.

id – primary key	label_en	uri
int(11)	text	text

Tabulka 14.1 : struktura tabulky se štítky

Id	Subject_id	Object_id	Predicate_id
Int(11)	Int(11)	Int(11)	Int(11)

Tabulka 14.2 : struktura tabulky s vazbami mezi štítky

Dalším požadavkem byla správa predikátů. K tomu bylo nezbytné přidat ještě jednu databázovou tabulku, viz Tabulka 14.3.

Id	Name
Int(11)	text

Tabulka 14.3 : struktura tabulky s predikáty

7.2 Záměna za vlastní

7.3 Práce s databází

Pro práci s databází Soubory se zdrojovými kódy obsahují

----- / DODĚLAT -----

8 Plugin jqTagTree

Testovacími daty naplněný plugin viz Obrázek 15.1.

jQuery plugin - jqTagTree

- First tag - level zero [1]
 - ⊖ Third tag [1]
- + Secon tag - level zero [0]

Obrázek 15.1: Plugin jqTagTree s testovacími daty

8.1 Před vykreslením stromu

Ihned po načtení stránky, kde má plugin jqTagTree běžet, se provede změna nastavení proměnných, které uživatel zadal při volání pluginu. Lze takto změnit použité ikony, cesty k PHP souborům nebo dobu a barvu zvýraznění právě přidávaného prvku. Následuje zjištění jména kořenového tagu, vložení potřebného HTML a zapnutí kontextového menu.

8.2 Vykreslování stromu

Najít způsob jak vykreslovat strom, který je zadán v databázi, nebylo jednoduché. Pokud bych chtěl načíst celou databázi do prohlížeče najednou, uživatel by dlouhou dobu viděl pouze informaci o načítání stromu a po nějaké době závislé na velikosti databáze a rychlosti připojení k ní, by se mu zobrazil celý strom. Tento způsob nemusí být vždy špatný. Pro tento případ to je ale krajně nevhodné. Mnohem lepší způsob je dynamické načítání stromu po jednotlivých úrovních. Kliknutím na nějaký prvek se z databáze načtou jeho přímí potomci.

Inspiraci ke tvorbě stromu jsem získal ve knize *jQuery novice to ninja*. Funguje na principu duplikace šablony. Potřebné HTML vložené před začátkem vykreslování je šablona struktury jednoho štítku, která bude později duplikována. Viz Kód 15.1.

```
<li style='display: none' class='tagNameTemplate liId'>
  <div style='position: relative'>
    <a href='#' class='showUls name' ></a>
    <span class='jqttNumOfPredic'></span>
    <img src='"+JQTT.globUserVar.iconLoadingPath+"'
      class='hidden'>
```

```

        </div>
    </li>

```

Kód 15.1 : Šablona prvku stromu

Po zkopírování šablony do pomocné proměnné se musí upravit pro zobrazení ve stromu. Nejprve je položce nečíslovaného seznamu odebrána třída `tagNameTemplate` a přidán atribut `id` s jedinečným identifikátorem `id`, který odpovídá záznamu v databázi. Dále je cíl odkazu nasměrován na adresu, kam má odkazovat. Mezi tagy `` a `` je vloženo číslo s počtem predikátů navázaných na daný prvek stromu. Značka `img` je většinu času skryta. Jedná se o obrázek načítání potomků objektu, na který bylo kliknuto. Je zobrazen pouze při čtení z databáze a ihned po načtení je skryt. Jelikož je tento vzor vkládán pomocí Javascriptu samotným pluginem, může zde být i řetězec `JQTT.globUserVar.iconLoadingPath`, zastupující proměnnou nastavitelnou při spouštění pluginu. Je v ní uložena cesta k souboru formátu `*.gif`. Adresa, kde se nachází informace o cestě k obrázku, je složen z předpony `JQTT`, která ošetřuje problém s překrytím proměnných jiných pluginů použitých uživatelem mnou vytvořeného pluginu. Dále je v adrese obsažen název `globUserVar`, který odkazuje v rámci pluginu `jqTagTree` na jmenný prostor s proměnnými. Následuje jméno samotné proměnné.

Součástí dotazu na databázi při načítání jedno či více prvků stromu je i zjistit počet potomků každého z nich. Tento údaj se použije k výběru ikony vkládané do šablony ještě před odkaz. Je-li počet větší než 0, bude vybráno zobrazení ikony značící možnost otevření větve. V případě prvků, u kterých je vybrána druhá možnost, je zobrazena ikona značící nemožnost rozevření větve a uživatel je po kliknutí na tento prvek přesměrován na adresu v odkazu. O které ikony jde, viz Obrázek 15.1: Plugin `jqTagTree` s testovacími daty.

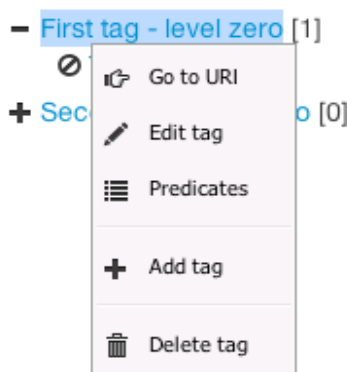
Stejný postup dynamického načítání objektů z databáze a kopírování jedné šablony je použit i pro načtení vůbec prvního elementu stromu. Při načítání jednotlivých úrovní stromu je volána funkce s parametrem představující unikátní identifikátor prvku. Načítá-li se první úroveň stromu, je tento parametr roven nule.

8.3 Kontextové menu

S každým prvkem stromu je při jeho tvorbě svázána událost čekající na kliknutí pravého tlačítka myši. To vede z pravidla k vyvolání kontextového menu prohlížeče. Díky této události lze vyřadit menu, které je běžně zobrazováno, a nahradit ho vlastním. V jiné části stránky ovšem funguje výchozí menu beze změn.

Kontextová nabídka se obsahuje čtyři položky. V případě, že element obsahuje predikáty, je přidána pátá. Více o predikátech v kapitole 15.4.4. V pluginu jsou odkazy v nabídce rozděleny do třech polí. Jedno obsahuje přechod na adresu URI a úpravu štítku, druhé položku predikátů a třetí přidávání a mazání. Těsně před vykreslením se skládá v závislosti na počtu predikátů konkrétního prvku. Menu včetně odkazu na predikáty viz Obrázek 15.2 : Kontextová nabídka s predikáty.

jQuery plugin - jqTagTree



Obrázek 15.2 : Kontextová nabídka s predikáty

Jakmile je nabídka zobrazena, je zbytek stránky pokryt neviditelným tagem `<div>`. Je-li kliknuto na nějakou položku menu, provede se příslušná akce. V opačném případě se zavolá funkce, která skrývá kontextovou nabídku.

8.4 Změny položek stromu

Zmiňovaná kontextová nabídka slouží k úpravě položek zobrazeného stromu. Po kliknutí na libovolnou položku, se zobrazí modální okno knihovny Twitter Bootstrap. Toto okno je druhou šablonou vkládanou před vykreslením stromu. Oproti předchozí šabloně z tvorby stromu, která byla duplikována, je pouze jedna. Její data se pozměňují tak, aby vyhovovala příslušnému účelu. Šablona ve tvaru vkládaném do stránky je v

Kód 15.2. Nejjednodušší změny prováděné vzhledem k výchozí šabloně jsou úprava textu nadpisu v hlavičce a změna cílů odkazů v patičce modálního okna. Co se zobrazuje v rámci těla, bude popsáno v dalších kapitolách.

Všechny změny stromové struktury probíhají jak na straně klienta v prohlížeči pomocí Javascriptu, tak v databázi na straně serveru pomocí technologie AJAX a jazyka PHP. Každá změna je interně rozdělena do tří menších funkcí. První upravuje šablonu modálního okna pro potřeby přidávání, úpravy, mazání nebo zobrazení predikátů. Druhá provádí validaci a výměnu zadaných dat se serverem. A nakonec třetí část mění data ve stromu.

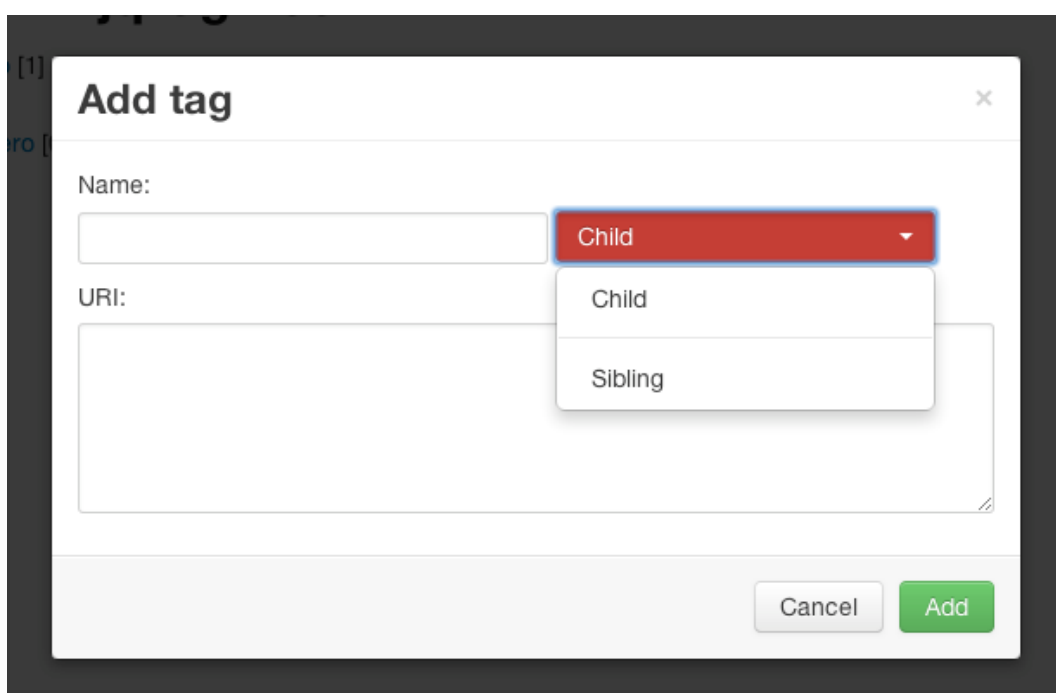
```
<div id='jqttModal' class='modal hide fade' tabindex='-1'
  role='dialog' aria-labelledby='myModalLabel' aria-
  hidden='true'>
  <div class='modal-header'>
    <button type='button' class='close' data-
      dismiss='modal' aria-hidden='true'>×
    </button><h3 id='modalLabel'></h3>
  </div>
  <div class='modal-body'></div>
  <div class='modal-footer'>
    <img src='"+
      JQTT.globUserVar.iconLoadingPath +"
      class='hidden'>
    <button class='btn' id='jqttModalCancel' data-
      dismiss='modal' aria-hidden='true'>Cancel
    </button>
    <button class='btn btn-success'
      id='jqttModalSave'>
    Save changes
    </button>
  </div>
</div>"
```

Kód 15.2 : Šablona modálního okna

8.4.1 Přidávání

Funkce pro přípravu přidávání štítku upraví a zobrazí modální okno - Obrázek 15.3. Při vkládání nových štítků do databáze je nezbytné znát jeho jméno, URI adresu a rozhodnout o jeho vazbě k prvku, na něhož bylo kliknuto. Jméno i adresu zadává

uživatel z klávesnice. Vazbu však vybírá jako jednu z řádek tabulky predikátů v databázi. Tento výběr probíhá pomocí select boxu, který je zobrazen u jména štítku. Styl pro pole výběru není bohužel v rámci Bootstrapu dodáván, a tak jsem stál před rozhodnutím, zda nechám výchozí vzhled, napíši vlastní styly nebo použiji něco již hotového. Nakonec jsem použil projekt Bootstrap-select²³. Je to plugin do jQuery využívající knihovny Bootstrapu distribuován, stejně jako Bootstrap pod licenci MIT. V nabídce je zobrazováno pouze jméno určitého predikátu. Jedná-li se o výchozí predikát, je namísto jména z databáze zobrazeno jméno „Child“ – jedná se o vazbu rodič – potomek.



Obrázek 15.3 : Modální okno pro přidávání štítku

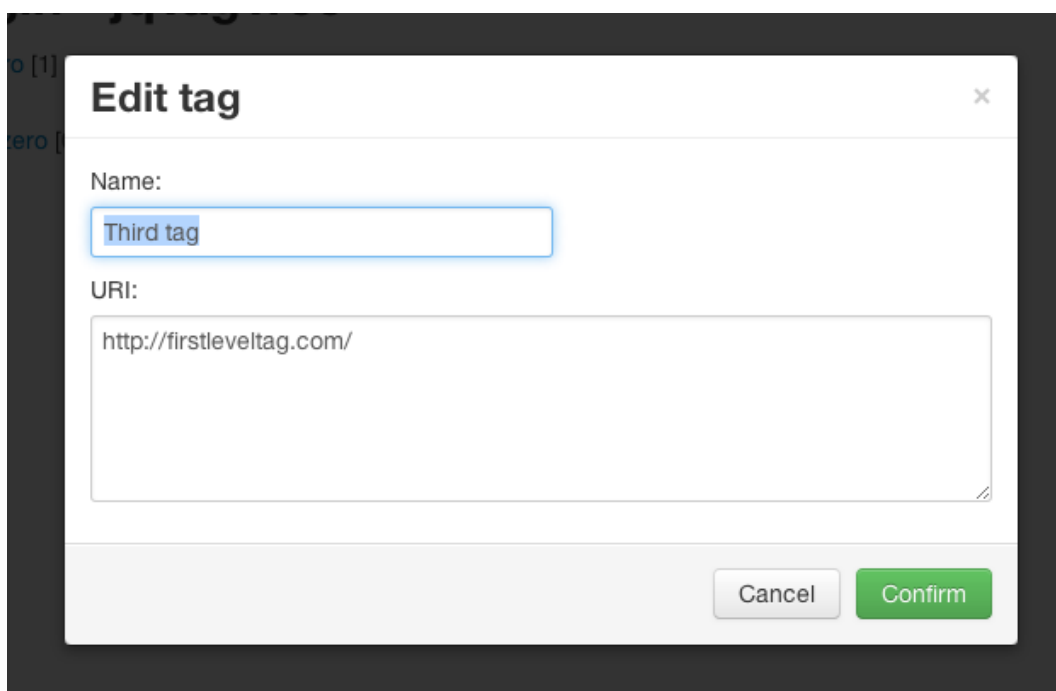
Kliknutím na tlačítko „add“ nastupuje validace vstupů z formuláře. V případě že jsou řetězce neprázdné a nejsou delší než povolená délka, jsou data odeslána na server. Tam se zjišťuje jméno vybraného predikátu a porovnává s databází. Pokud je správně zadané, tak se zapíše data do tabulky štítků i vazeb. Po úspěšné úpravě dat v databázi následuje funkce překreslující data ve stromu.

Zde je více větvení, jelikož je nutné odlišit několik zobrazení. Je-li štítek přidáván do větve, která nebyla uživatelem ještě otevřena, musí se nejprve stáhnout ze serveru. K tomu je použita funkce napsána při vykreslování stromu. Dále podle id najít prvek, který byl přidán a zvýraznit ho. Další věc je načtená, ale skrytá větev. V tomto případě

se musí zobrazit, změnit ikony zavřené větve na otevřenou, vložit právě přidávaný prvek a zvýraznit ho. Podobně to funguje i v případě prázdné větve. Nejjednodušší práce je s otevřenou. Tam probíhá pouze vkládání a zvýrazňování.

8.4.2 Úprava

Úprava je v tomto případě z programátorského hlediska nejjednodušší ze všech akcí prováděných pomocí kontextového menu. Jedná se o pouhou změnu názvu a URI adresy štítku v databázi a prohlížeči (viz Obrázek 15.4). Funkce překreslující modální okno má tři vstupní parametry id, jméno a URI. Ty jsou získány z elementu, na který bylo kliknuto. Tato data se zde vyplní do formulářů a mohou být upravena. Rozhodne-li se uživatel změny uložit a zmáčkne tlačítko „Confirm“, je zkontrolováno, zda nejsou formuláře prázdné a uložena do databáze. Nakonec se provede úprava stromu v prohlížeči.



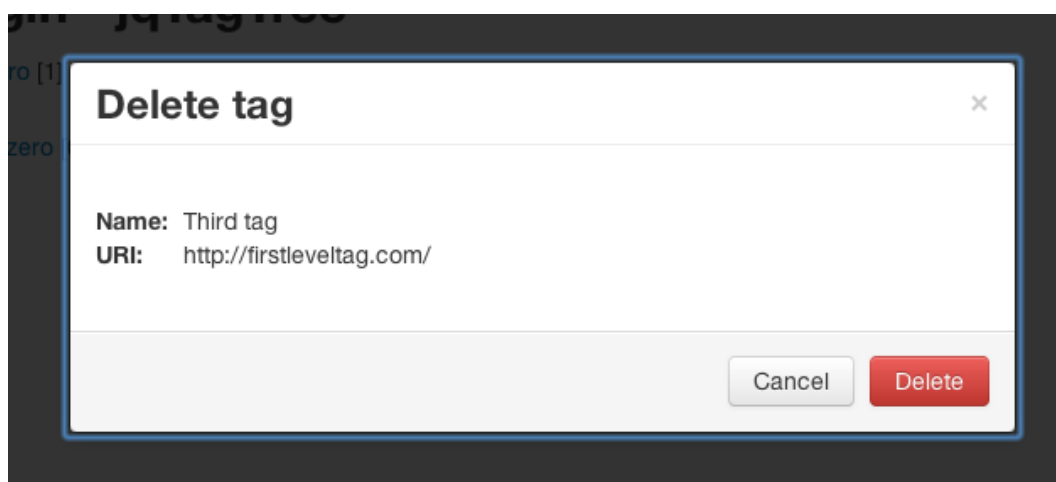
Obrázek 15.4: Modální okno pro úpravu štítků

8.4.3 Mazání

Úprava okna před smazáním tagu je stejná jako u úpravy štítku jen s tím rozdílem, že se data nevyplňují do formulářů, ale zobrazena jako prostý text (viz Obrázek 15.5:

Modální okno pro smazání štítku. Po potvrzení mazání, je pomocí AJAXu zavolán soubor `delNode.php`. Jeho cílem je smazat štítek včetně jeho potomků. Postup je k tomu následující následující:

- Vložit právě mazaný prvek do pole kam se budou ukládat všechny, jejichž přímí potomci byli již nalezeni.
- Najít všechny potomky mazaného prvku.
- Uložit do dočasného pole, ze kterého se budou hledat jeho potomci.
- Dále postupně procházet dočasné pole a hledat v databázi potomky jednotlivých záznamů.
- Prohledané prvky uložit do prvního pole, smazat z dočasného a nově nalezené vložit do dočasného.
- Jakmile je dočasné pole prázdné, zavolá se funkce pro smazání prvků z databáze a je jí předáno pole s prvky ke smazání.



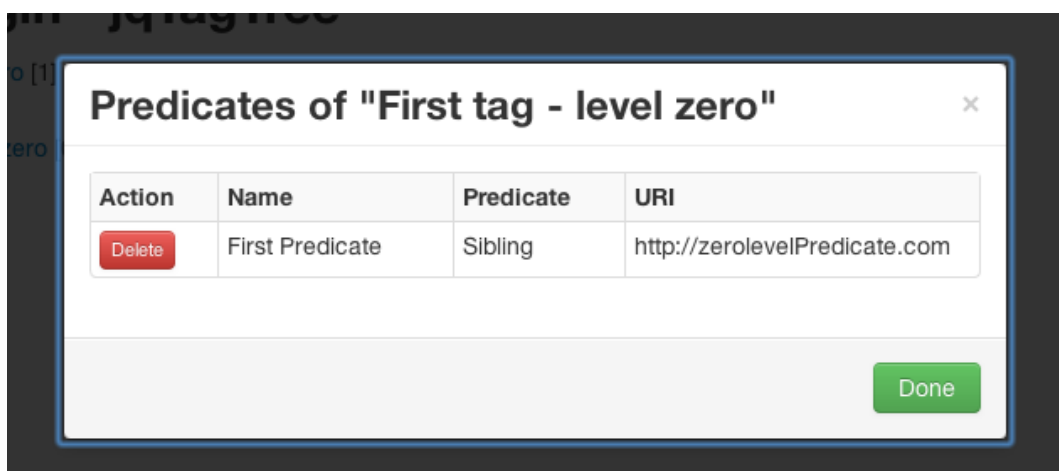
Obrázek 15.5: Modální okno pro smazání štítku

8.4.4 Zobrazení predikátů

Predikáty jsou v této aplikaci pojaty jako jakési vazby mezi prvky. V tabulce predikátů jsou záznamy, které představují jednotlivé vazby. Mají id, které je spojeno s každým štítkem. Je-li v konfiguračním souboru konstanta `JQTT_DEFAULT_PREDICATE_ID` nastavena na hodnotu jedna, jsou všechny tagy s tímto predikátem zobrazeny ve stromu. Jakmile se změní konstanta na jiné číslo, jsou jako prvky stromu brány jiné štítky.

Kolik má který štítek predikátů, je indikováno číslem v hranatých závorkách hned za jménem ve stromu (viz Obrázek 15.1: Plugin jqTagTree s testovacími daty. Na toto místo se počet predikátů ukládá při úpravě vzorové šablony jak bylo zmíněno v kapitole 15.2. Před vložením úrovně do prohlížeče je v databázi nalezeno kolik má daný štítek potomků a kolik z nich jsou predikáty. Je-li po odečtení potomků stále více než nula, je před jménem tagu zobrazena ikona značící možnost otevření uzlu. Počet predikátů ať už je jakýkoli, je zobrazen ve výše zmíněných závorkách.

Vybráním položky *Predicates* z kontextového menu, se zobrazí modální okno s tabulkou ve kterém jsou vypsány predikáty. K vložení tabulky a záznamů do ní, je opět použita funkce *ajax* z *jQuery*. Po zmáčknutí tlačítka *delete* v tabulce, je predikát smazán z databáze. Dále je třeba změnit číslo s počtem predikátů ve stromové struktuře. Název každé položky ve stromu je rozdělen pomocí tříd kaskádových stylů na jméno štítku a počet predikátů. Lze tedy ze jména „*First tag - level zero* [1]“ vybrat pouze text [1]. Z něj jsou dále oříznuty závorky a hodnota čísla snížena o jedna. Následně je modální okno skryto.



Obrázek 15.6: Modální okno pro zobrazení predikátů

9 Tvorba ukázkové webové aplikace

Dále je sepsán návod jak postupovat při tvorbě webové stránky s využitím pluginu jqTagTree.

).

	Score
Maxthon 4.0 »	476
Chrome 26 »	468
Opera 12.10 »	419
Firefox 20 »	394
Safari 6.0 »	378
Internet Explorer 10 »	320
<i>Microsoft Surface and others</i>	

Obrázek 9.1: Žebříček prohlížečů seřazený podle počtu bodů

Mezi nejzajímavější rozšíření nové verze patří třeba přidání nového atributu data, který může pomoci například při odesílání formulářů nebo při výběru tagu Javascriptem. Hlavním přínosem nové verze jsou nové tagy² zlepšující sémantickou čitelnost webu.

- header – text dokumentu v záhlaví (header není head)
- footer – text patřící do zápatí dokumentu
- article – definuje prostor pro článek
- figure – prostor pro obrázky, grafy, ilustrace, kódy apod.
- audio, video – multimediální soubory

Dalším rozšířením je přidání několika typů vstupních polí u formulářů. To opět přidá na čitelnosti webu a navíc je výhodné pro zařízení s dotykovými obrazovkami. Pokud je typ vstupního pole například email, webová adresa nebo telefonní číslo (viz <input id="email" type="email"/> <input id="website" type="url"/> <input id="phone" type="tel" />

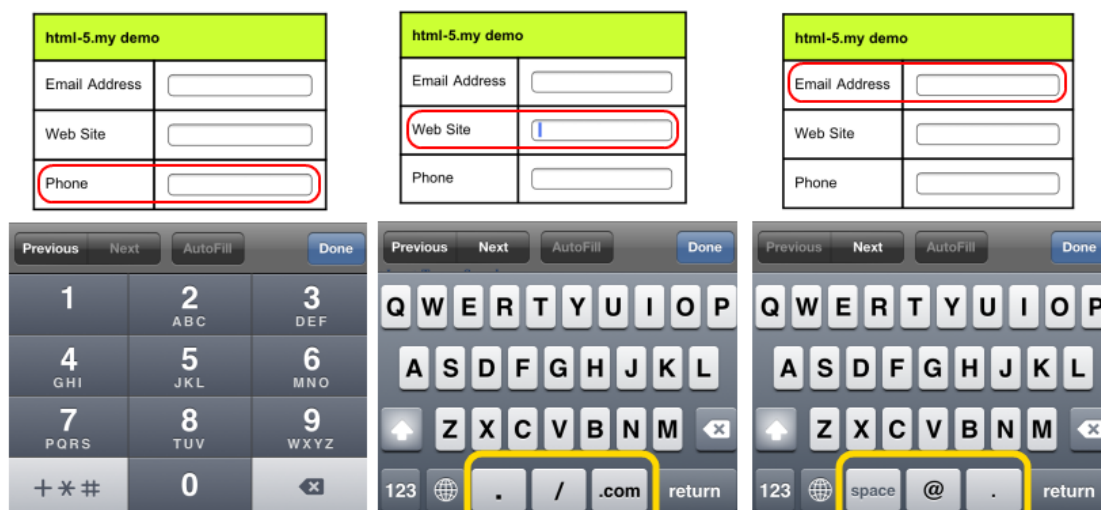
Kód 9.1: Vstupní pole formuláře v HTML 5

² http://www.w3schools.com/html/html5_new_elements.asp

), může prohlížeč díky dobré strojové čitelnosti dokumentu uzpůsobit klávesnici pro jejich vkládání (viz Obrázek 9.2). Stejně tak lze použít třeba barvu, vyhledávání, heslo nebo datum. Podrobnější popis lze nalézt na stránkách <http://html5doctor.com/html5-forms-input-types/>. Zdroj příkladu viz <http://www.html5tutorial.info/html5-contact.php>.

```
<input id="email" type="email"/>
<input id="website" type="url"/>
<input id="phone" type="tel" />
```

Kód 9.1: Vstupní pole formuláře v HTML 5



Obrázek 9.2: Změna klávesnice na dotykových zařízeních

9.1.1 CSS

Kaskádové styly³ slouží k definici vzhledu stránek vytvořených v HTML, XHTML, nebo XML. Díky nim mohou být elementy po webové stránce posouvány, upravovány, měněna jejich velikost nebo barva. U textu lze změnit font, barvu či velikost. To je samozřejmě jen výběr toho, co kaskádové styly umí. Dokáží ale také výrazným způsobem zlepšit přehlednost webu. Existují tři způsoby, jak vložit styly do dokumentu. Přímě do tagu `<div style = "color : red">...</div>`, mezi tagy `<style></style>` a třetí nejpoužívanější možnost je vložit vlastnosti do externího souboru, na který se odkáže v hlavičce HTML souboru.

³ www.w3.org/Style/CSS/

Specifikace třetí verze kaskádových stylů také ještě není zcela hotova. O podpoře jednotlivých vlastností v prohlížečích informuje server W3Schools⁴. Mezi významnější zjednodušení při tvorbě vzhledu internetových stránek patří možnost přidat elementům stíny, zaoblit rohy nebo třeba možnost definice průhlednosti elementů.

9.1.2 Javascript

Javascript⁵ je skriptovací jazyk běžící na straně klienta. Existují také řešení spouštěné na straně serveru jako například Node.js⁶. Slovo java je v názvu jazyka jen z marketingových důvodů a s programovacím jazykem Java má společnou pouze syntaxi. Javascript stránkám přidává na dynamičnosti. Vhodné je využití při tvorbě menu a různých kontejnerů, které se po přejetí myši rozevřou a ušetří tak místo na obrazovce. Lze ho také použít při tvorbě webového editoru textu.

Své využití má i při validaci formulářů. Je-li odeslán s daty ve špatném formátu, zobrazí upozornění bez komunikace se serverem a uživatel tak dostane informaci o datech rychleji. Javascript si ale uživatel v prohlížeči může vypnout, a proto je nutné odeslaná data validovat ještě na straně serveru. Je vhodné při práci s Javascriptem používat některý z frameworků. Obsahují množství funkcí výrazně zjednodušující práci s dokumentem.

Asynchronní Javascript a XML technologie (AJAX) je používána pro dynamické načítání obsahu stránek bez nutnosti znovu načítat celou internetovou stránku. Lze ji také používat pro pouhé načítání statických HTML stránek, po kliknutí na odkaz načíst příslušný formulář nebo lze takto volat PHP kód, který může například vybrat určitá data z databáze, předat je dále Javascriptu ke zpracování.

⁴ http://www.w3schools.com/cssref/css3_browsersupport.asp

⁵ <https://developer.mozilla.org/en-US/docs/JavaScript>

⁶ <http://nodejs.org/>

9.1.3 Coffeescript

9.2 Serverové technologie

9.2.1 PHP

Skriptovací jazyk PHP je zpracováván na straně serveru a je tak hlavním článkem při validaci formulářů odesílaných uživatelem. Od čtvrté verze lze v PHP programovat objektově⁷. Do páté verze byl však objektový model PHP přepracován⁸. PHP je pravděpodobně nejrozšířenější skriptovací jazyk pro tvorbu webu, a díky tomu pro něj existuje mnoho knihoven a frameworků, například Nette⁹ nebo Zend¹⁰. Syntaxí spadá do rodiny C/Java.

9.2.2 MySQL

MySQL je databázový systém vycházející z normy jazyka SQL. Tuto normu však striktně nerespektuje. MySQL je používán na většině webových hostingů. Od převzetí firmy MySQL firmou Oracle na podzim roku 2010 je nabízen jak pod bezplatnou, tak pod placenou licenci. Od verze 5.0 podporuje například i uložené procedury, trigery a pohledy.

9.3 Přenos dat klient - server

Nejpoužívanější formáty pro přenos dat mezi klientem a serverem jsou JSON a XML. Oba formáty lze použít jak v internetových aplikacích, tak v off-line aplikacích. V současné době je při přenosu dat více využíván spíše JSON. Je tomu tak převážně kvůli objemu dat, který je potřeba přenést při komunikaci. Formát XML vychází ze syntaxe HTML. To znamená, že veškerá informace o datech, je uložena mezi značkami, které si uživatel může definovat vlastní. V Kód 9.2 je uveden příklad dat zapsaných

⁷ <http://php.net/manual/en/oop4.php>

⁸ <http://php.net/manual/en/language.oop5.php>

⁹ <http://nette.org/cs/>

¹⁰ <http://framework.zend.com/>

pomocí XML. Značka osoba s atributy jmeno a prijmeni. Kód 9.3 dává za příklad, jak vypadají data odesílána pomocí JSON.

```
<osoba jmeno="Petr" prijmeni="Novak">  
</osoba>
```

Kód 9.2: Příklad XML kódu

```
"jmeno": "Petr", "prijmeni": "Novak"
```

Kód 9.3 : Příklad JSON kódu

Pokud porovnáme Kód 9.2 a Kód 9.3, lze si všimnout, v tuto chvíli nepatrných, rozdílů ve velikosti jednotlivých kódů. Pokud se přenášejí větší data, tento rozdíl bude znatelnější a JSON může uživateli ušetřit velké množství času.

Další výhodou JSONu je jednoduchost interpretace přijatých dat. PHP dokonce obsahuje funkci pro převod mezi polem a JSON strukturou. Tuto funkci má v sobě také framework jQuery pro Javascript. Z těchto důvodů jsem si pro přenos dat vybral právě JSON.

9.4 Vývojová prostředí

Je dobrým zvykem psát kód v programu, který alespoň zvýrazňuje syntaxi jazyka, ve kterém programátor píše. Zástupcem nejjednodušších prostředí je pro platformu Windows například program Notepad++. Pro Mac OS X je to pak třeba TextMate. Obě tato prostředí rozpoznají syntaxi velkého množství skriptovacích, značkovacích a programovacích jazyků. Mimo jiné také Javascript, HTML, CSS a PHP, které se většinou používají při tvorbě jQuery pluginů. Já pro práci na svých projektech používám Netbeans. Ten umí kromě zvýrazňování klíčových slov, také doplňovat názvy proměnných a funkcí. Má také správu projektů a další funkce, které usnadňují vývoj.

9.5 Záloha dat

Před samotným vývojem pluginu bylo nezbytné vyřešit problematiku zálohování zdrojových kódů a textu bakalářské práce. Způsobů, jak zálohovat, je hned několik. Od vypalování záloh na CD, přes kopírování dat na externí disk, zrcadlení disků, nahrávání dat na internet do externích úložišť až po verzovací systémy. Pro mé potřeby a potřeby

zadavatele je nejvhodnější poslední jmenovaná možnost. Druhý systému pro správu verzí je více. Pro příklad uvedu čtyři nejpoužívanější:

- CVS
- SVN
- GIT
- Mercurial

Jedno z nejhlavnějších dělení je na centralizované a decentralizované¹¹. CVS¹², stejně jako SVN¹³ spadají do kategorie centralizovaných verzovacích systémů, což v principu znamená, že mají jeden server, kde jsou nahrány veškeré zdrojové kódy. Jestliže vše funguje jak má, tak s tím není problém. Pokud se ale stane, že server spadne nebo nefunguje připojení k internetu, tak uživatelé ztrácejí k repositáři přístup a často to znamená nemožnost další práce. Je-li repositář decentralizovaný, jako v případě Mercurial¹⁴ a GITu¹⁵, tak každý kdo s ním pracuje, má staženou jeho verzi u sebe v počítači a lze tím pádem pracovat i na cestě bez připojení k internetu. Po návratu do sítě změny odešle, případně si stáhne úpravy ostatních členů projektu, a repositář dále zůstane aktuální.

Na konci minulého roku byl společností Redmonk proveden výzkum nejpoužívanějších verzovacích systémů v porovnání s rokem 2010 (viz <http://redmonk.com/sogady/2012/11/05/dvcs-2012/>). Je z něho patrné, že jsou stále nejvíce používány centralizované systémy. Jsou však pomalu na ústupu. Nejpoužívanějšími zástupci jsou SVN a Git

Díky službě <http://github.com>, kde lze umístit a verzovat jakýkoli kód zdarma v případě, že je repositář veden jako veřejný. Na základě domluvy s vedoucím jsem zvolil tuto variantu. Jelikož je mou povinností mít bakalářskou práci veřejně dostupnou je služba github zcela ideální. Další výhodou je pak jednoduchá kontrola stavu práce vedoucím.

¹¹ <http://biz30.timedocor.com/git-mecurial-and-cvs-comparison-of-svn-software/>

¹² <http://cvs.nongnu.org/>

¹³ <http://subversion.apache.org/>

¹⁴ <http://hgbook.red-bean.com/read/>

¹⁵ <http://git-scm.com/documentation>

10 Použité frameworky

10.1 jQuery

jQuery je Javascriptový framework, který si klade za úkol zjednodušit programování v Javascriptu a sjednotit jeho zobrazování v prohlížečích. Od jeho založení v roce 2006 jeho popularita raketově vzrostla. Nyní je používán nejnavštěvovanějšími webovými servery¹⁶, jako například <http://amazon.com>, <http://microsoft.com>, <http://twitter.com>. Více než 62% nejpopulárnějších internetových stránek [1] používá jQuery.

Jednou z nejpoužívanějších funkcí jQuery je výběr elementu na stránce. Cokoli je potřeba v dokumentu změnit, musí to být nejdříve vybráno. Zde je porovnání, jak vypadá vybrání prvku s id = menu.

```
document.getElementById('menu'); //Javascript  
$('#menu')                       //jQuery
```

Kód 10.1: Porovnání jQuery a Javascriptu

Další funkce poskytované jQuery :

- Změna DOM (Document Object Model) elementů na stránce – přidávání či mazání html tagů ze stránky
- Úprava kaskádových stylů po výběru elementu
- Funkce pro práci s AJAXem
- Systém událostí
- Animace a efekty
- Snadné přidávání pluginů

Framework jQuery je postaven tak, aby vývojářům co možná nejvíce zjednodušil tvorbu pluginů. Nejslavnějším pluginem pro jQuery je jQuery UI vytvořený samotnou společností jQuery. Jedná se o sadu Javascriptových funkcí pro tvorbu uživatelského rozhraní. Mezi nejzajímavější patří funkce pro posuv elementů po stránce, dále obsahuje různé doplňky jako tooltip, datepicker nebo autocomplete a v neposlední řadě přidává

¹⁶ <http://trends.builtwith.com/websitelist/jquery>

oproti jQuery řadu efektů zobrazování a skrývání elementů. jQuery UI je stejně jako jQuery poskytován pod MIT licenci¹⁷.

10.2 Twitter Bootstrap

Twitter Bootstrap¹⁸ je kolekce nástrojů pro tvorbu webových aplikací. Během necelého půl roku od vydání se stal nejpopulárnějším projektem na GitHubu. Jednou z nejzásadnějších věcí, které Bootstrap řeší, je nekompatibilita zobrazení různých prvků na stránce v různých prohlížečích. Pokud programátor použije k tvorbě designu svého webu výhradně stylů Bootstrapu, nemusí se nekompatibility bát.

Bootstrap je knihovna obsahující naprostou většinu stylů, které jsou zapotřebí ke tvorbě jednoduchých internetových prezentací. Mezi základní styly, díky kterým stojí zato Twitter Bootstrap použít, patří:

- Rozložení prvků na stránce
- Základní šablony pro web
- Responsivní design
- Typografie

Dále obsahuje styly pro menu, tlačítka, stránkování, štítky nebo náhledy obrázků. Tato knihovna je dále dodávána se sadou ikon Glyphicons¹⁹. Součástí Bootstrapu je i několik jQuery pluginů. Bootstrap se během necelých dvou let stal velmi oblíbený. Svědčí o tom nejen stránky <http://builtwithbootstrap.com/>, které shromažďují stránky postavené na základu Bootstrapu, ale i web <https://wrapbootstrap.com/>, který nabízí k prodeji šablony založené opět na Bootstrapu.

¹⁷ <http://opensource.org/licenses/MIT>

¹⁸ <http://twitter.github.io/bootstrap/>

¹⁹ <http://glyphicons.com/>

11 Podobné práce

Pluginů pro zobrazení, či úpravu stromu lze na internetu nalézt velké množství. Dále budou některé z nich představeny a uvedeny jejich přednosti a nedostatky.

jQuery tree view²⁰

Plugin pro pouhé zobrazování stromové struktury, a jednoduché přidávání složek a mazání položek stromu a složek. Využívá funkcí `slideUp` a `slideDown` pro zobrazování respektive skrývání větví. Kliknutím na prvek stromu je uživatel přesměrován na stránku v odkazu. Podporuje funkci rozevřít a zavřít všechny větve stromu. První úroveň je zobrazena s jiným formátem písma než ostatní. Vývoj tohoto projektu byl však už zastaven.

jqTree²¹

Součástí funkcionality jqTree je zobrazování a skrývání položek stromu a jejich přesouvání technikou `drag and drop`. Data pro strom lze načíst pomocí funkce `jquery` `getJSON`, nebo z pole které má stejnou strukturu. Kód pluginu je napsán v Coffeescriptu.

jsTree²²

Plugin jstree má více funkcí než předešlé pluginy a to, nejen zobrazování a přesouvání technikou `drag and drop`, ale také přidávání složek, souborů, jejich přejmenování a mazání. Dále plugin také umožňuje vyhledávání ve stromu.

²⁰ <https://github.com/jzaefferer/jquery-treeview>

²¹ <https://github.com/mbraak/jqTree>

²² <http://www.jstree.com>

12 Tvorba jednoduchého jQuery pluginu

Základní kostra dokumentu v HTML 5, kterou budeme dále rozšiřovat, vypadá následovně.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=UTF-8">
  </head>
  <body>
  </body>
</html>
```

Kód 12.1: Šablona HTML dokumentu.

Vytváření pluginu do jQuery je velmi jednoduché. Abychom mohli používat jQuery, stačí do hlavičky dokumentu přidat následující řádek.

```
<script src="http://code.jquery.com/jquery-latest.min.js"
type="text/JavaScript"></script>
```

Kód 12.2: Vložení Javascriptu.

Ten načte zdrojové kódy jQuery frameworku ze stránek jquery.com a vloží jej na naše. Má to tu výhodu, že někteří uživatelé přicházející z jiných stránek, kde jQuery také používají, a tudíž tyto kódy mohou mít staženy a nemusí je stahovat znova. Nyní můžeme všechny funkce jQuery používat dle libosti. Řekněme, že budeme chtít vytvořit plugin, který nám jednou na vteřinu ukáže nějaký text. Mohlo by to být třeba upozornění, že jsme se na stránky přihlásili.

Nejdříve budeme potřebovat prvek, který budeme zobrazovat a skrývat. V našem případě div. To je jen jednoduché ohraničení nějakého textu či bloku kódu. Následně ho budeme pomocí Javascriptu vybírat. Náš div bude vypadat následovně.

```
<div>Jste přihlášen/a.</div>
```

Kód 12.3: Zobrazovaný div.

Tímto jsme s HTML hotovi a můžeme se pustit do tvorby pluginu. Budeme ho vkládat do hlavičky hned za načtení jQuery. Je nezbytné, aby nejdřív proběhlo načtení jQuery. V opačném případě by plugin nefungoval, jelikož by nebylo do čeho vkládat.

Veškerý kód Javascriptu začíná párovým tagem `<script>` a končí `</script>`. Dále bude následovat znak dolaru “\$”, který odpovídá volání funkce `jQuery`. Má pouze funkci zkrácení zápisu. Za ním v kulatých závorkách bude argument funkce, neboli takzvaný selektor, kterým vybereme potřebný element na stránce. Nejprve je potřeba počkat, než se celá stránka stáhne k uživateli a připraví k použití. Výběrem dokumentu a zavolání funkce `ready` to můžeme považovat za hotové. Viz Kód 12.4: Funkce čekající na dokončení načítání web stránky. To je nezbytné, protože jinak bychom mohli skrývat něco, co tam ještě není. Tento případ nemusí být tak závažný, konzole by vypsala nějakou chybu, kterou uživatel při prohlížení stránek ani neuvidí. Lze si ale představit případ vedoucí k horším výsledkům. Navíc se může projevit jen jednou za čas v závislosti na rychlosti načtení dokumentu a odhalení těchto chyb by nemuselo být jednoduché. Dále budeme vše psát do anonymní funkce vytvořené v argumentu funkce `ready`.

```
$(document).ready(function(){});
```

Kód 12.4: Funkce čekající na dokončení načítání web stránky.

Plugin budeme psát do jmenného prostoru k tomu vyhrazenému frameworkem `jQuery`, a to `jQuery.fn.jmenoPluginu`, do kterého uložíme anonymní funkci s kódem pluginu. Je dobré kvůli bezpečnosti uzavřít ho ještě do anonymní funkce, díky které nemusíme mít strach, že by náš plugin kolidoval s nějakou jinou knihovnou nebo pluginem.

```
(function($){  
/*kód našeho pluginu v anonymní funkci*/  
})(jQuery);
```

Kód 12.5: Anonymní funkce v `jQuery` pluginu.

Upozornění uživatele pomocí jednoduchého zobrazení a skrytí textu provedeme za pomoci tří funkcí `jQuery`. Nejdříve prvek skryjeme funkcí `hide()`, jako že tam není. Pak ho pomalu zobrazíme, `slideDown(1000)` a nakonec zase schováme `slideUp(1000)`. Argumentem funkcí `slide` je čas v milisekundách, jak dlouho má skrývání, popřípadě zobrazování, trvat. Tedy jednu vteřinu každé. Plugin máme hotov. Pokud bychom v tuto chvíli zkusili stránku načíst, nic by se nestalo. Musíme ho totiž zavolat, jinak by nevěděl kdy a kde pracovat.

```
$('#div').helloWorldPlugin();
```

Kód 12.6: Volání testovacího pluginu.

Následuje celý zdrojový kód, který po uložení do souboru s příponou HTML a spuštění v prohlížeči, zobrazí a zase skryje text „*Jste přihlášen/a*“.

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;
  charset=UTF-8">
  <script src=http://code.jquery.com/jquery-latest.min.js
  type="text/Javascript"></script>
  <script>
    $(document).ready(function() {
      (function($) {
        $.fn.helloWorldPlugin = function() {
          this.hide();
          this.slideDown(1000);
          this.delay(1000);
          this.slideUp(1000);
        };
      })(jQuery);
      $('#div').helloWorldPlugin();
    });
  </script>
</head>
<body>
  <div>Jste přihlášen/a.</div>
</body>
</html>
```

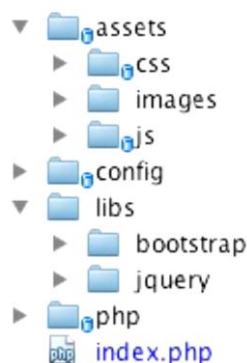
Kód 12.7: Jednoduchý jquery plugin.

13 Adresářová struktura pluginu

Umístění jednotlivých souborů je klíčové pro přehlednost jak moji, tak potenciálních uživatelů mého pluginu, kteří ho budou chtít implementovat do svých webových stránek.

Kořenový adresář obsahuje čtyři složky a jeden soubor. Index.php musí být vždy hned v první úrovni struktury. Nad ním je složka php, kam spadá veškerý kód v jazyce PHP. V mém případě je to pouze třída pracující s databází a soubory volané AJAXem při zobrazování nebo úpravě stromu. Ve složce libs se nacházejí knihovny použité v

pluginu. Jedná se o jQuery, jQuery UI a Twitter Bootstrap. Dále jsem použil sadu funkcí s názvem Bootstrap-Select²³ pro úpravu vzhledu selectboxu ve stylu Twitter Bootstrap. Složka config obsahuje nastavení konstant, které se týkají PHP a přístupů do databáze. Na konci je složka assets, do které patří kaskádové styly, javascriptové soubory a obrázky. Je tam také umístěn kód mého pluginu s názvem `jqTagTree.js`. Vzhled adresářové struktury viz Obrázek 13.1.



Obrázek 13.1: Adresářová struktura pluginu

²³ <https://github.com/silviomoreto/bootstrap-select>

14 Databáze

----- DODĚLAT -----

14.1 Návrh použité databáze

Jeden z požadavků programu bylo zachování stávající struktury databáze. Ta obsahuje tři tabulky. První v sobě udržuje informace o štítku, jedinečné id, viz Tabulka 14.1 a v druhé jsou uloženy informace o vazbách mezi štítky, viz Tabulka 14.2.

id – primary key	label_en	uri
int(11)	text	text

Tabulka 14.1 : struktura tabulky se štítky

Id	Subject_id	Object_id	Predicate_id
Int(11)	Int(11)	Int(11)	Int(11)

Tabulka 14.2 : struktura tabulky s vazbami mezi štítky

Dalším požadavkem byla správa predikátů. K tomu bylo nezbytné přidat ještě jednu databázovou tabulku, viz Tabulka 14.3.

Id	Name
Int(11)	text

Tabulka 14.3 : struktura tabulky s predikáty

14.2 Záměna za vlastní

14.3 Práce s databází

Pro práci s databází Soubory se zdrojovými kódy obsahují

----- / DODĚLAT -----

15 Plugin jqTagTree

Testovacími daty naplněný plugin viz Obrázek 15.1.

jQuery plugin - jqTagTree

- First tag - level zero [1]
 - ⊖ Third tag [1]
- + Secon tag - level zero [0]

Obrázek 15.1: Plugin jqTagTree s testovacími daty

15.1 Před vykreslením stromu

Ihned po načtení stránky, kde má plugin jqTagTree běžet, se provede změna nastavení proměnných, které uživatel zadal při volání pluginu. Lze takto změnit použité ikony, cesty k PHP souborům nebo dobu a barvu zvýraznění právě přidávaného prvku. Následuje zjištění jména kořenového tagu, vložení potřebného HTML a zapnutí kontextového menu.

15.2 Vykreslování stromu

Najít způsob jak vykreslovat strom, který je zadán v databázi, nebylo jednoduché. Pokud bych chtěl načíst celou databázi do prohlížeče najednou, uživatel by dlouhou dobu viděl pouze informaci o načítání stromu a po nějaké době závislé na velikosti databáze a rychlosti připojení k ní, by se mu zobrazil celý strom. Tento způsob nemusí být vždy špatný. Pro tento případ to je ale krajně nevhodné. Mnohem lepší způsob je dynamické načítání stromu po jednotlivých úrovních. Kliknutím na nějaký prvek se z databáze načtou jeho přímí potomci.

Inspiraci ke tvorbě stromu jsem získal ve knize *jQuery novice to ninja*. Funguje na principu duplikace šablony. Potřebné HTML vložené před začátkem vykreslování je šablona struktury jednoho štítku, která bude později duplikována. Viz Kód 15.1.

```
<li style='display: none' class='tagNameTemplate liId'>
  <div style='position: relative'>
    <a href='#' class='showUls name' ></a>
    <span class='jqttNumOfPredic'></span>
    <img src='"+JQTT.globUserVar.iconLoadingPath+"'
      class='hidden'>
```



```

        </div>
    </li>

```

Kód 15.1 : Šablona prvku stromu

Po zkopírování šablony do pomocné proměnné se musí upravit pro zobrazení ve stromu. Nejprve je položce nečíslovaného seznamu odebrána třída `tagNameTemplate` a přidán atribut `id` s jedinečným identifikátorem `id`, který odpovídá záznamu v databázi. Dále je cíl odkazu nasměrován na adresu, kam má odkazovat. Mezi tagy `` a `` je vloženo číslo s počtem predikátů navázaných na daný prvek stromu. Značka `img` je většinu času skryta. Jedná se o obrázek načítání potomků objektu, na který bylo kliknuto. Je zobrazen pouze při čtení z databáze a ihned po načtení je skryt. Jelikož je tento vzor vkládán pomocí Javascriptu samotným pluginem, může zde být i řetězec `JQTT.globUserVar.iconLoadingPath`, zastupující proměnnou nastavitelnou při spouštění pluginu. Je v ní uložena cesta k souboru formátu `*.gif`. Adresa, kde se nachází informace o cestě k obrázku, je složen z předpony `JQTT`, která ošetřuje problém s překrytím proměnných jiných pluginů použitých uživatelem mnou vytvořeného pluginu. Dále je v adrese obsažen název `globUserVar`, který odkazuje v rámci pluginu `jqTagTree` na jmenný prostor s proměnnými. Následuje jméno samotné proměnné.

Součástí dotazu na databázi při načítání jedno či více prvků stromu je i zjistit počet potomků každého z nich. Tento údaj se použije k výběru ikony vkládané do šablony ještě před odkaz. Je-li počet větší než 0, bude vybráno zobrazení ikony značící možnost otevření větve. V případě prvků, u kterých je vybrána druhá možnost, je zobrazena ikona značící nemožnost rozevření větve a uživatel je po kliknutí na tento prvek přesměrován na adresu v odkazu. O které ikony jde, viz Obrázek 15.1: Plugin `jqTagTree` s testovacími daty.

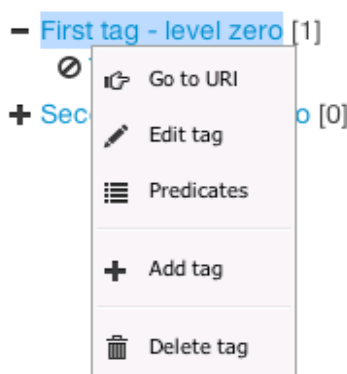
Stejný postup dynamického načítání objektů z databáze a kopírování jedné šablony je použit i pro načtení vůbec prvního elementu stromu. Při načítání jednotlivých úrovní stromu je volána funkce s parametrem představující unikátní identifikátor prvku. Načítá-li se první úroveň stromu, je tento parametr roven nule.

15.3 Kontextové menu

S každým prvkem stromu je při jeho tvorbě svázána událost čekající na kliknutí pravého tlačítka myši. To vede z pravidla k vyvolání kontextového menu prohlížeče. Díky této události lze vyřadit menu, které je běžně zobrazováno, a nahradit ho vlastním. V jiné části stránky ovšem funguje výchozí menu beze změn.

Kontextová nabídka se obsahuje čtyři položky. V případě, že element obsahuje predikáty, je přidána pátá. Více o predikátech v kapitole 15.4.4. V pluginu jsou odkazy v nabídce rozděleny do třech polí. Jedno obsahuje přechod na adresu URI a úpravu štítku, druhé položku predikátů a třetí přidávání a mazání. Těsně před vykreslením se skládá v závislosti na počtu predikátů konkrétního prvku. Menu včetně odkazu na predikáty viz Obrázek 15.2 : Kontextová nabídka s predikáty.

jQuery plugin - jqTagTree



Obrázek 15.2 : Kontextová nabídka s predikáty

Jakmile je nabídka zobrazena, je zbytek stránky pokryt neviditelným tagem `<div>`. Je-li kliknuto na nějakou položku menu, provede se příslušná akce. V opačném případě se zavolá funkce, která skrývá kontextovou nabídku.

15.4 Změny položek stromu

Zmiňovaná kontextová nabídka slouží k úpravě položek zobrazeného stromu. Po kliknutí na libovolnou položku, se zobrazí modální okno knihovny Twitter Bootstrap. Toto okno je druhou šablonou vkládanou před vykreslením stromu. Oproti předchozí šabloně z tvorby stromu, která byla duplikována, je pouze jedna. Její data se pozměňují tak, aby vyhovovala příslušnému účelu. Šablona ve tvaru vkládaném do stránky je v

Kód 15.2. Nejjednodušší změny prováděné vzhledem k výchozí šabloně jsou úprava textu nadpisu v hlavičce a změna cílů odkazů v patičce modálního okna. Co se zobrazuje v rámci těla, bude popsáno v dalších kapitolách.

Všechny změny stromové struktury probíhají jak na straně klienta v prohlížeči pomocí Javascriptu, tak v databázi na straně serveru pomocí technologie AJAX a jazyka PHP. Každá změna je interně rozdělena do tří menších funkcí. První upravuje šablonu modálního okna pro potřeby přidávání, úpravy, mazání nebo zobrazení predikátů. Druhá provádí validaci a výměnu zadaných dat se serverem. A nakonec třetí část mění data ve stromu.

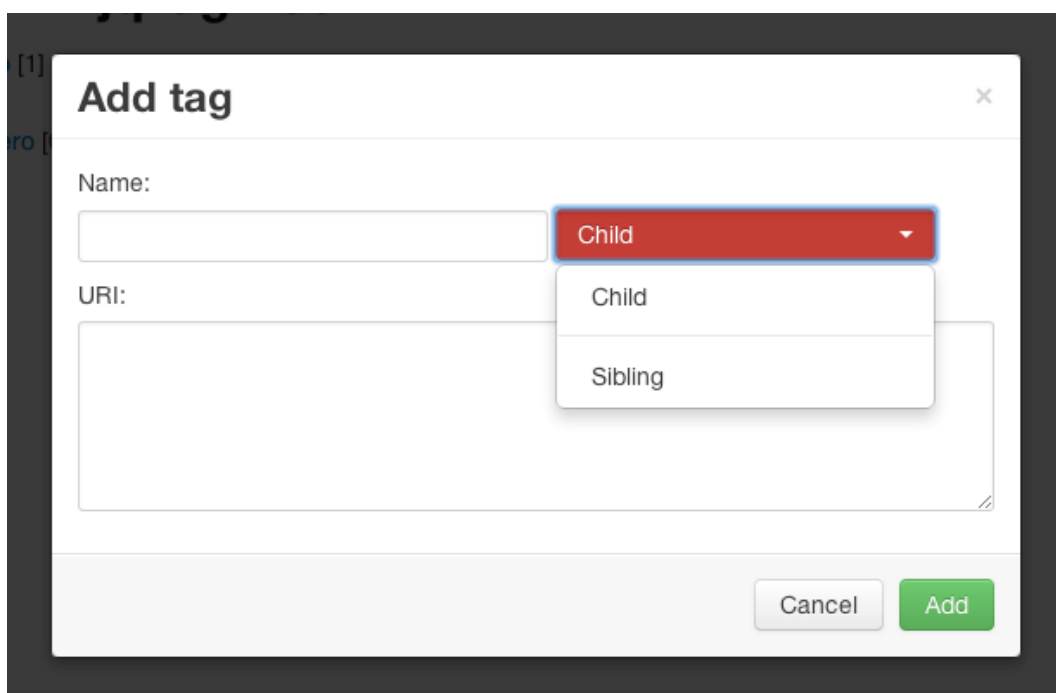
```
<div id='jqttModal' class='modal hide fade' tabindex='-1'
  role='dialog' aria-labelledby='myModalLabel' aria-
  hidden='true'>
  <div class='modal-header'>
    <button type='button' class='close' data-
      dismiss='modal' aria-hidden='true'>×
    </button><h3 id='modalLabel'></h3>
  </div>
  <div class='modal-body'></div>
  <div class='modal-footer'>
    <img src='"+
      JQTT.globUserVar.iconLoadingPath +"
      class='hidden'>
    <button class='btn' id='jqttModalCancel' data-
      dismiss='modal' aria-hidden='true'>Cancel
    </button>
    <button class='btn btn-success'
      id='jqttModalSave'>
      Save changes
    </button>
  </div>
</div>"
```

Kód 15.2 : Šablona modálního okna

15.4.1 Přidávání

Funkce pro přípravu přidávání štítku upraví a zobrazí modální okno - Obrázek 15.3. Při vkládání nových štítků do databáze je nezbytné znát jeho jméno, URI adresu a rozhodnout o jeho vazbě k prvku, na něhož bylo kliknuto. Jméno i adresu zadává

uživatel z klávesnice. Vazbu však vybírá jako jednu z řádek tabulky predikátů v databázi. Tento výběr probíhá pomocí select boxu, který je zobrazen u jména štítku. Styl pro pole výběru není bohužel v rámci Bootstrapu dodáván, a tak jsem stál před rozhodnutím, zda nechám výchozí vzhled, napíši vlastní styly nebo použiji něco již hotového. Nakonec jsem použil projekt Bootstrap-select²³. Je to plugin do jQuery využívající knihovny Bootstrapu distribuován, stejně jako Bootstrap pod licenci MIT. V nabídce je zobrazováno pouze jméno určitého predikátu. Jedná-li se o výchozí predikát, je namísto jména z databáze zobrazeno jméno „Child“ – jedná se o vazbu rodič – potomek.



Obrázek 15.3 : Modální okno pro přidávání štítku

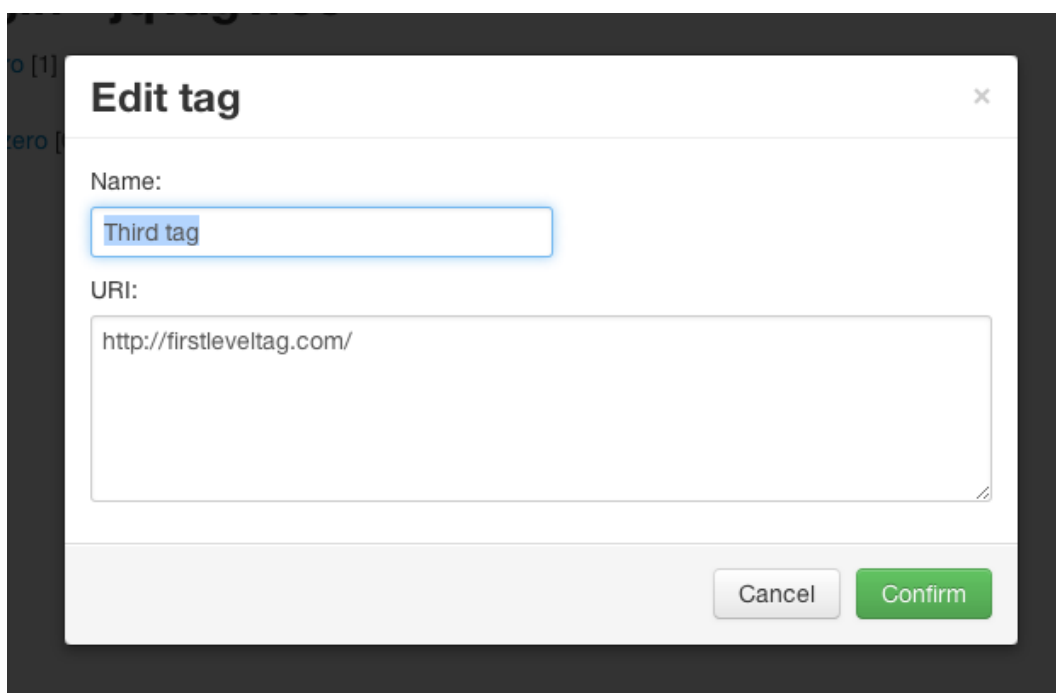
Kliknutím na tlačítko „add“ nastupuje validace vstupů z formuláře. V případě že jsou řetězce neprázdné a nejsou delší než povolená délka, jsou data odeslána na server. Tam se zjišťuje jméno vybraného predikátu a porovnává s databází. Pokud je správně zadané, tak se zapíše data do tabulky štítků i vazeb. Po úspěšné úpravě dat v databázi následuje funkce překreslující data ve stromu.

Zde je více větvení, jelikož je nutné odlišit několik zobrazení. Je-li štítek přidáván do větve, která nebyla uživatelem ještě otevřena, musí se nejprve stáhnout ze serveru. K tomu je použita funkce napsána při vykreslování stromu. Dále podle id najít prvek, který byl přidán a zvýraznit ho. Další věc je načtená, ale skrytá větev. V tomto případě

se musí zobrazit, změnit ikony zavřené větve na otevřenou, vložit právě přidávaný prvek a zvýraznit ho. Podobně to funguje i v případě prázdné větve. Nejjednodušší práce je s otevřenou. Tam probíhá pouze vkládání a zvýrazňování.

15.4.2 Úprava

Úprava je v tomto případě z programátorského hlediska nejjednodušší ze všech akcí prováděných pomocí kontextového menu. Jedná se o pouhou změnu názvu a URI adresy štítku v databázi a prohlížeči (viz Obrázek 15.4). Funkce překreslující modální okno má tři vstupní parametry id, jméno a URI. Ty jsou získány z elementu, na který bylo kliknuto. Tato data se zde vyplní do formulářů a mohou být upravena. Rozhodne-li se uživatel změny uložit a zmáčkne tlačítko „Confirm“, je zkontrolováno, zda nejsou formuláře prázdné a uložena do databáze. Nakonec se provede úprava stromu v prohlížeči.



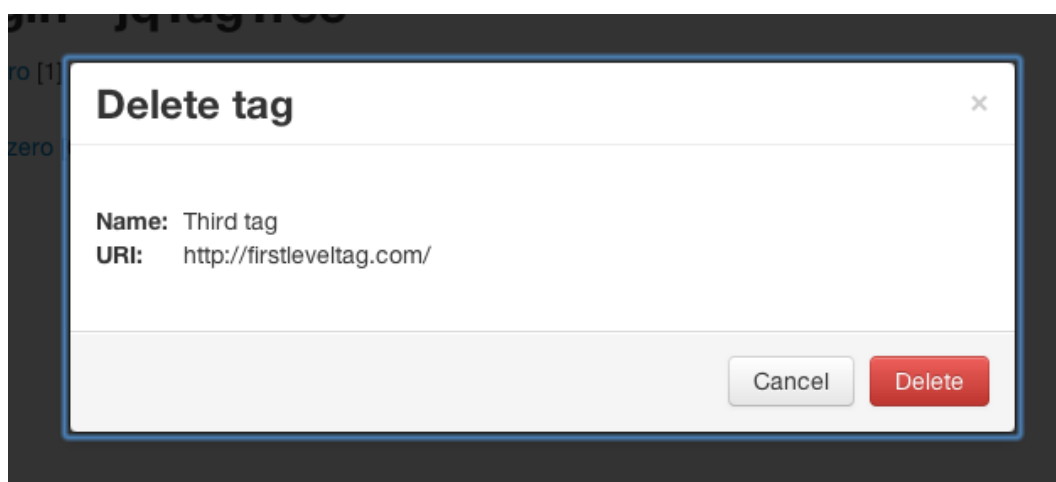
Obrázek 15.4: Modální okno pro úpravu štítků

15.4.3 Mazání

Úprava okna před smazáním tagu je stejná jako u úpravy štítku jen s tím rozdílem, že se data nevyplňují do formulářů, ale zobrazena jako prostý text (viz Obrázek 15.5:

Modální okno pro smazání štítku. Po potvrzení mazání, je pomocí AJAXu zavolán soubor `delNode.php`. Jeho cílem je smazat štítek včetně jeho potomků. Postup je k tomu následující následující:

- Vložit právě mazaný prvek do pole kam se budou ukládat všechny, jejichž přímí potomci byli již nalezeni.
- Najít všechny potomky mazaného prvku.
- Uložit do dočasného pole, ze kterého se budou hledat jeho potomci.
- Dále postupně procházet dočasné pole a hledat v databázi potomky jednotlivých záznamů.
- Prohledané prvky uložit do prvního pole, smazat z dočasného a nově nalezené vložit do dočasného.
- Jakmile je dočasné pole prázdné, zavolá se funkce pro smazání prvků z databáze a je jí předáno pole s prvky ke smazání.



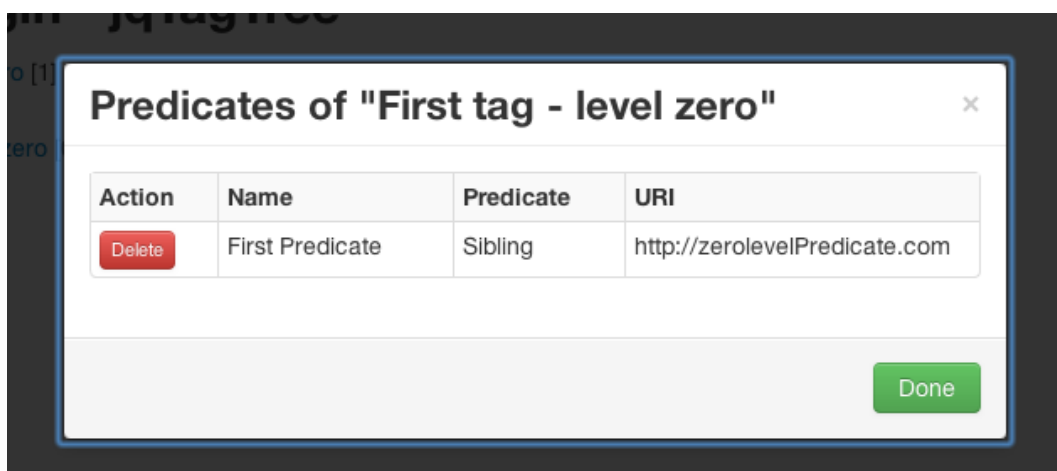
Obrázek 15.5: Modální okno pro smazání štítku

15.4.4 Zobrazení predikátů

Predikáty jsou v této aplikaci pojaty jako jakési vazby mezi prvky. V tabulce predikátů jsou záznamy, které představují jednotlivé vazby. Mají id, které je spojeno s každým štítkem. Je-li v konfiguračním souboru konstanta `JQTT_DEFAULT_PREDICATE_ID` nastavena na hodnotu jedna, jsou všechny tagy s tímto predikátem zobrazeny ve stromu. Jakmile se změní konstanta na jiné číslo, jsou jako prvky stromu brány jiné štítky.

Kolik má který štítek predikátů, je indikováno číslem v hranatých závorkách hned za jménem ve stromu (viz Obrázek 15.1: Plugin jqTagTree s testovacími daty. Na toto místo se počet predikátů ukládá při úpravě vzorové šablony jak bylo zmíněno v kapitole 15.2. Před vložením úrovně do prohlížeče je v databázi nalezeno kolik má daný štítek potomků a kolik z nich jsou predikáty. Je-li po odečtení potomků stále více než nula, je před jménem tagu zobrazena ikona značící možnost otevření uzlu. Počet predikátů ať už je jakýkoli, je zobrazen ve výše zmíněných závorkách.

Vybráním položky *Predicates* z kontextového menu, se zobrazí modální okno s tabulkou ve kterém jsou vypsány predikáty. K vložení tabulky a záznamů do ní, je opět použita funkce *ajax* z *jQuery*. Po zmáčknutí tlačítka *delete* v tabulce, je predikát smazán z databáze. Dále je třeba změnit číslo s počtem predikátů ve stromové struktuře. Název každé položky ve stromu je rozdělen pomocí tříd kaskádových stylů na jméno štítku a počet predikátů. Lze tedy ze jména „*First tag - level zero* [1]“ vybrat pouze text [1]. Z něj jsou dále oříznuty závorky a hodnota čísla snížena o jedna. Následně je modální okno skryto.



Obrázek 15.6: Modální okno pro zobrazení predikátů

16 Tvorba ukázkové webové aplikace

Dále je sepsán návod jak postupovat při tvorbě webové stránky s využitím pluginu jqTagTree.

- Vytvoření tabulek v databázi
- Vložení složkové struktury pluginu do složky s webem

17 Závěr

Jelikož je v pluginu použito prefixové pojmenování proměnných a jmenných prostorů, tak je zaručeno, že lze na stránkách použít i jiných pluginů.

18 Použitá literatura

1. *BuiltWith Technology Lookup: Trends* [online]. 2013-4-12 [cit. 2013-04-17].

Dostupné z: <http://trends.builtwith.com/javascript/JQuery>