

Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

## **Bakalářská práce**

# **jQuery – vytvoření pluginu pro manipulaci se stromem štítků**

Plzeň, 2013

Libor Vohanka

## **Prohlášení**

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. 5. 2013 Libor Vohanka .....

## **Poděkování**

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Martinu Dostalovi za poskytnutí odborných rad, věcných připomínek, ochotu a vstřícný přístup během zpracování této práce.

# Abstract

## **jQuery – creating of plugin for hanging with tree of tags**

Goal of my bachelor thesis is to make a plugin for jQuery framework. Plugin loads tags structure from database and display them in web browser using tree structure. Then there is an option to add, edit and delete tags. Changes in the tree are performed using the contextual menu. After each of these operations is the result saved back to database. There is also included a simple administration of predicates of tags. All this is done using Javascript and Ajax. Plugin works asynchronously without need to reload the page. The purpose of this work is comfortable modification of the database generated by external program. There was further used language PHP and HTML. Data between PHP and Javascript was transferred using JSON technology.

# Abstrakt

## **jQuery – vytvoření pluginu pro manipulaci se stromem štítků**

Cíl mé bakalářské práce byl vytvořit plugin pro jQuery framework. Plugin načte strukturu štítků z databáze a s využitím stromové struktury je zobrazí v prohlížeči. Dále umožní přidávat, upravovat a mazat štítky. Změny ve stromu jsou prováděny s využitím kontextového menu. Po provedení každé akce je výsledek uložen zpět do databáze. V pluginu je také jednoduchá správa predikátů tagů. Vše je vytvořeno za použití Javascriptu a Ajaxu. Plugin pracuje asynchronně bez nutnosti obnovy stránky. Účel této práce je pohodlná úprava databáze generované externím programem. Dále bylo využito jazyků PHP a HTML. Data mezi PHP a Javascriptem jsou přesouvána pomocí technologie JSON.

# Obsah

1 Úvod.....	1
2 Použité technologie.....	2
2.1 Technologie pro klientskou část .....	2
2.1.1 HTML a jiné značkovací jazyky.....	2
2.1.2 CSS .....	4
2.1.3 Javascript .....	4
2.1.4 Coffeescript.....	5
2.2 Serverové technologie.....	6
2.2.1 PHP .....	6
2.2.2 MySQL .....	7
2.3 Přenos dat klient - server .....	7
2.4 Záloha dat .....	8
2.5 Vývojová prostředí .....	9
3 Použité frameworky.....	10
3.1 jQuery .....	10
3.2 Twitter Bootstrap .....	11
4 Podobné práce.....	12
4.1 jQuery tree view.....	12
4.2 jqTree .....	12
4.3 jsTree .....	12
5 Tvorba jednoduchého jQuery pluginu .....	13
6 Adresářová struktura pluginu.....	17
7 Databáze.....	18
7.1 Návrh použité databáze.....	18
7.2 Práce s databází a záměna za vlastní .....	19

8 Plugin jqTagTree .....	21
8.1 Vykreslování stromu.....	21
8.2 Kontextové menu.....	23
8.3 Změny položek stromu .....	23
8.3.1 Přidávání .....	25
8.3.2 Úprava.....	26
8.3.3 Mazání .....	27
8.3.4 Zobrazení predikátů .....	27
9 Použití pluginu ve webové aplikaci .....	29
10 Závěr .....	31
Použitá literatura .....	32
Seznam kódů.....	34
Seznam obrázků.....	34
Seznam tabulek .....	35

# 1 Úvod

Cílem této práce je plugin do frameworku jQuery, který přehledně zobrazí štítky uložené v databázi v prohlížeči. Důvodem pro tvorbu tohoto programu je pohodlná úprava databáze a její specifická struktura, kterou bylo nutné dodržet. Kromě zobrazování štítků musí obsahovat další vlastnosti, a to přidávání nových štítků, změna stávajících a jejich mazání včetně potomků. To vše se musí opět projevit v databázi beze změny její struktury.

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 velké množství pluginů rozšiřující jeho možnosti různými směry. Velké množství z nich různým způsobem zobrazuje fotografie ve fotogaleriích, další animují rozevírací menu stránek, ať už hlavní, nebo vedlejší, některé spravují kalendáře, nebo okna pro výběr dat přímo v prohlížeči. Dále jsou pluginy porovnávající text psaný uživatelem s databází, přičemž mu zobrazují nápovědu u podobných slov a umožňují doplnit celé slovo. Další spravují záložky, animují progressbary a v neposlední řadě zobrazují stromy.

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



## 2 Použité technologie

### 2.1 Technologie pro klientskou část

Do této kategorie spadají technologie běžící na počítači uživatele a mající na starosti pouhé zobrazení a uspořádání dat přijatých od serveru do formy, která bude pro uživatele přehledná.

#### 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 formátovány. Cílem značek je přidání sémantické informace, která zvýší jak lidskou, tak strojovou čitelnost dokumentu. Dále je uveden příklad některých značek. Kompletní seznam lze nalézt na webu spravující standard HTML [1].

- 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 2.3 Přenos dat klient - server. Později byl uveden jazyk XHTML syntaxí vycházející z XML. Je však jen úpravou HTML. Následuje stručný výpis zásadních změn, ve specifikaci XHTML oproti HTML [2]:

- 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 hotová, prohlížeče již HTML5 více či méně podporují. Na stránkách <http://html5test.com> lze otestovat podporu kteréhokoli prohlížeče. V závislosti na tom,

kolik funkcí HTML 5 podporuje, dostane bodové ohodnocení do celkového počtu 500 bodů. Pořadí a skóre nejpoužívanějších prohlížečů [3] (viz Obrázek 2.1).

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

Obrázek 2.1: Žebříček prohlížečů seřazený podle počtu bodů

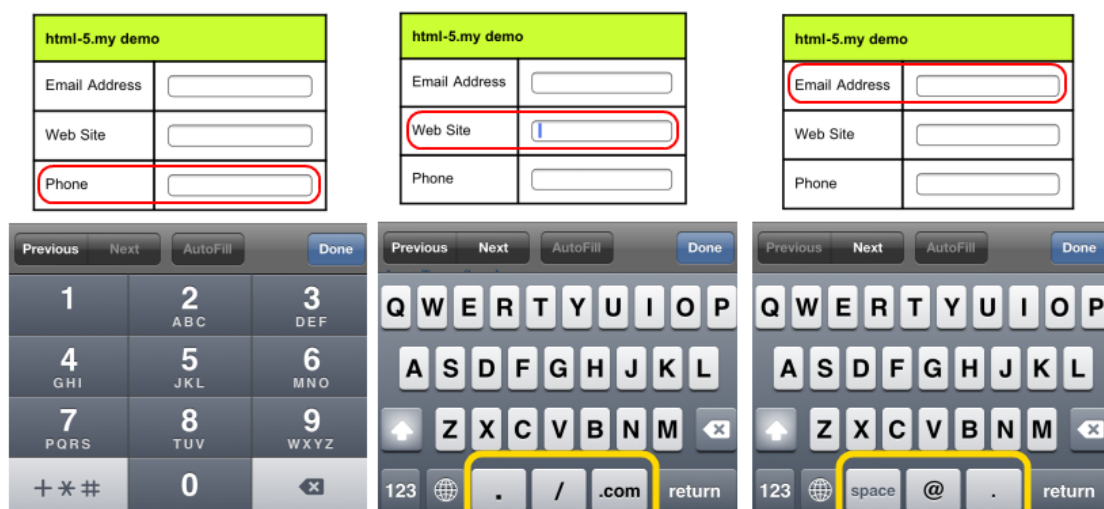
Mezi nejzajímavější rozšíření nové verze patří třeba přidání nového atributu data, který může pomoci například při odesílání formulářů nebo při výběru tagu Javascriptem. Hlavním přínosem nové verze jsou nové tagy [4] zlepšující sémantickou čitelnost webu.

- 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. Tím je opět zvýšena čitelnost webu nejen pro uživatele, ale také pro stroje. To je významné také pro zařízení s dotykovými obrazovkami. Pokud je typ vstupního pole například email, webová adresa nebo telefonní číslo (viz Kód 2.1), může prohlížeč díky dobré strojové čitelnosti dokumentu uzpůsobit klávesnici pro jejich vkládání (viz Obrázek 2.2). Stejně tak lze použít třeba barvu, vyhledávání, heslo nebo datum. Touto tematikou se zabývá <http://html5doctor.com/html5-forms-input-types/>. Tento příklad byl inspirován [5].

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

Kód 2.1: Vstupní pole formuláře v HTML 5 [5]



Obrázek 2.2: Změna klávesnice na dotykových zařízeních [5]

## 2.1.2 CSS

Kaskádové styly [6] 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 krátký 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>`, zpravidla vkládané do hlavičky dokumentu a třetí nejpoužívanější možnost je vložit vlastnosti do externího souboru, na který je odkázáno v hlavičce HTML souboru.

Specifikace třetí verze kaskádových stylů také ještě není zcela hotova. O podpoře jednotlivých vlastností v prohlížečích informuje server W3Schools<sup>1</sup>. 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 [7] [8] je skriptovací jazyk běžící na straně klienta. Existují také řešení spouštěné na straně serveru jako například Node.js [9]. Slovo Java je v názvu jazyka jen

<sup>1</sup> [http://www.w3schools.com/cssref/css3\\_browsersupport.asp](http://www.w3schools.com/cssref/css3_browsersupport.asp)

z marketingových důvodů a s programovacím jazykem Java má společnou pouze syntaxi. Základy Javascriptu jsou formovány jazykem ECMAScript<sup>2</sup>, jehož začátky sahají až do roku 1997. Vhodné je využití při tvorbě menu a různých kontejnerů, které se po přejetí myši rozevrou a ušetří tak místo na obrazovce. Lze ho také použít při tvorbě webového editoru textu. Javascript stránkám přidává na dynamičnosti, to v praxi znamená, že uživatel má dojem rychlejších stránek, jelikož se mu například menu rozevře ihned po kliknutí.

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. Dále lze takto volat PHP kód, který může například vybrat určitá data z databáze a předat je Javascriptu ke zpracování a zobrazení.

#### 2.1.4 Coffeescript

Javascript je poměrně starý a hojně používaný jazyk. Proto vznikají knihovny, které v určitých směrech zlepšují programátorovi psaní zdrojového kódu. Jsou to jak frameworky, které obalují Javascript vlastními funkcemi a přidávají mu různou funkcionalitu, tak jazyky velmi se lišící syntaxí, které se do Javascriptu před spuštěním kompilují. Takových jazyků je celá řada [10] a proto zde uvedu jednoho z nejpopulárnějších zástupců a to Coffeescript [11].

Kód napsaný v CoffeeScriptu je po spuštění zkompilován do Javascriptu. Výrobce poznamenává, že efektivita výsledného zdrojového kódu je minimálně stejná, jako by byl psán v samotném Javascriptu v určitých případech dokonce větší [12]. Jelikož je kód ve výsledku pouhý Javascript, lze použít knihoven i frameworků třetích stran bez

---

<sup>2</sup> <http://www.ecmascript.org/>

omezení. Vygenerovaný soubor je dobře čitelný a dále upravitelný. Lze proto k němu přiřpat funkce v Javascriptu a do stránek může být vložen až po úpravě. Dále uvedu některé z rozdílů mezi coffeescriptem a Javascriptem [12] (viz Tabulka 2.1).

Javascript	Coffeescript
<pre>if (opposite) {   number = -42; }</pre>	<pre>number = -42 if opposite</pre>
<pre>square = function(x) {   return x * x; };</pre>	<pre>square = (x) -&gt; x * x</pre>
<pre>if (   typeof elvis !== "undefined"   &amp;&amp; elvis !== null) {   alert("I knew it!"); }</pre>	<pre>alert "I knew it!" if elvis?</pre>

Tabulka 2.1: Porovnání syntaxe Javascriptu a Coffeescriptu [12].

## 2.2 Serverové technologie

### 2.2.1 PHP

Skriptovací jazyk PHP je zdarma distribuovaný jazyk zpravidla poskytovaný jako modul, který můžeme nahrát na webové servery na velkém množství operačních systémů a mnoha platformách. PHP se zapisuje přímo do HTML kódu stránky a je zpracováván na straně serveru - je tak hlavním článkem při validaci formulářů odesílaných uživatelem. Od čtvrté verze lze v PHP programovat objektově [13]. Do páté verze byl však objektový model PHP přepracován [14]. PHP je pravděpodobně nejrozšířenější skriptovací jazyk pro tvorbu webu a díky tomu pro něj existuje mnoho knihoven a frameworků, například Nette<sup>3</sup> nebo Zend<sup>4</sup>. Syntaxí spadá do rodiny C/Java.

Hlavním rozdílem PHP oproti jazykům C nebo Java je typovost a způsob deklarace proměnných. V PHP nejsou programátorem předem nijak inicializovány a vznikají až

<sup>3</sup> <http://nette.org/cs/>

<sup>4</sup> <http://framework.zend.com/>

v případě, že do nich bylo něco uloženo. Java i C vyžadují při deklaraci proměnné uvést datový typ - zda se jedná o číslo, znak a podobně.

## 2.2.2 MySQL

MySQL je nejznámější multiplatformní databázový systém používaný na většině hostingů a vycházející z normy jazyka SQL. Tuto normu však striktně nerespektuje. 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.

MySQL je velmi surový systém, jehož dotazování může být v některých případech velice nepřehledné. Nejen z tohoto důvodu jsou součástí některých frameworků obalové třídy, které práci s ním zjednodušují. Výhodou tohoto řešení je i následná přenositelnost dotazů mezi databázovými systémy.

## 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 využíváno spíše XML. Je však na ústupu před modernějším JSON formátem. Příčinou toho je převážně objem dat, který je potřeba přenést při komunikaci. Formát XML vychází ze syntaxe HTML. To znamená, že veškerá informace o datech je uložena jak mezi, tak ve značkách definovaných uživatelem. Příklad dat zapsaných pomocí XML viz Kód 2.2. Kód 2.3 popisuje předchozí příklad, zapsaný ve formátu JSON.

```
<osoba>
  <jmeno type="string">Petr</jmeno>
  <prijmeni type="string">Novák</prijmeni>
</osoba>
```

Kód 2.2: Příklad XML kódu

```
{
  "osoba": {
    "jmeno": "Petr",
    "prijmeni": "Novák"
  }
}
```

Kód 2.3: Příklad JSON kódu

Pokud porovnáme Kód 2.2 a Kód 2.3, lze si všimnout, v tuto chvíli nepatrných, rozdílů ve velikosti jednotlivých kódů. Jakmile se ale přenášejí větší data, tento rozdíl bude znatelnější a JSON může uživateli ušetřit značné množství času při přesunu dat mezi serverem a prohlížečem.

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 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 vedoucího práce je nejvhodnější poslední jmenovaná možnost. Druhý systému pro správu zdrojových kódů je více. Pro příklad uvedu čtyři nejpoužívanější:

- CVS
- SVN
- GIT
- Mercurial

Jedno z nejhlavnějších dělení je na centralizované a decentralizované [15]. SVN [16], stejně jako CVS [17] 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 v nejlepším případě na pár hodin. Je-li repositář decentralizovaný, jako v případě Mercurial [15] a GITu [15], tak každý, kdo s ním pracuje, má staženou jeho verzi u sebe v počítači a lze tím pádem pracovat i na cestě vlakem, bez připojení k internetu. Po návratu do sítě internet změny odeslat, případně si stáhnout úpravy ostatních členů projektu, spojit je se svými a repositář dále zůstane aktuální.

Na konci minulého roku (2012) byl společností Redmonk proveden výzkum nejpoužívanějších verzovacích systémů v porovnání s rokem 2010 [18]. Je z něho patrné, že jsou stále nejvíce používány centralizované systémy. Jsou však pomalu na ústupu. Nejpoužívanějšími zástupci jsou SVN a Git. Největší změnu zaznamenal právě systém Git, který se v poslední době rozšířil především díky službě github.com.

Github umožňuje spravovat 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.

## 2.5 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 prostředí Netbeans. Ten umí, stejně jako řada jiných, kromě zvýrazňování klíčových slov, také doplňovat názvy proměnných a funkcí. Má také správu projektů a další funkce, které usnadňují vývoj jak webových, tak desktopových aplikací. Navíc je zde implementován i klient pro správu zdrojových kódů podporující systém GIT.



## 3 Použité frameworky

### 3.1 jQuery

jQuery je Javascriptový framework, který si klade za úkol zjednodušit programování v Javascriptu a sjednotit jeho práci v prohlížečích. Od jeho založení v roce 2006 jeho popularita raketově vzrostla. Nyní je používán nejnavštěvovanějšími webovými servery [19], jako například <http://amazon.com>, <http://microsoft.com> nebo <http://twitter.com>. Více než 62% nejpopulárnějších internetových stránek [20] používá jQuery.

Jednou z nejpoužívanějších funkcí jQuery je výběr elementu na stránce. Cokoli je potřeba v dokumentu změnit, musí být nejdříve vybráno. Pro porovnání, jak vypadá vybrání prvku s `id = menu` v jazyce Javascript a v jQuery viz Kód 3.1.

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

Kód 3.1: 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

Nahrání konkrétní verze jQuery lze zajistit buď stažením zdrojového kódu pluginu<sup>5</sup> a nahrání na web, s využitím služeb typu CDN<sup>6</sup> (Content Delivery Network).

Framework jQuery je postaven tak, aby vývojářům co možná nejvíce zjednodušil práci s dokumentem, ale zaměřil se i na vývojáře a tvorba pluginů je také velmi jednoduchá. Nejslavnějším pluginem pro jQuery je jQuery UI vytvořený samotnou společností jQuery. Jedná se o sadu Javascriptových funkcí pro tvorbu uživatelského

<sup>5</sup> <http://code.jquery.com/jquery-1.9.1.min.js>

<sup>6</sup> [//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js](http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js)

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<sup>7</sup>.

## 3.2 Twitter Bootstrap

Twitter Bootstrap<sup>8</sup> je kolekce kaskádových stylů pro tvorbu vzhledu webových aplikací. Během krátké chvíle se stal velmi populárním projektem na Githubu. Jednou z nejzásadnějších věcí, které Bootstrap řeší, je nekompatibilita zobrazení různých prvků na stránce v odlišných 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
- Vzhled všech prvků

Dále obsahuje styly pro menu, tlačítka, stránkování, štítky nebo náhledy obrázků. Tato knihovna je dále dodávána se sadou ikon Glyphicons<sup>9</sup>. Součástí Bootstrapu je i několik jQuery pluginů. Bootstrap se během necelých dvou let od svého vzniku stal velmi oblíbený. Svědčí o tom nejen stránky <http://builtwithbootstrap.com/>, které shromažďují stránky postavené buď se základem, nebo výhradně s použitím Bootstrapu, ale i web <https://wrapbootstrap.com/>, který nabízí k prodeji šablony založené opět na Bootstrapu.

---

<sup>7</sup> <http://opensource.org/licenses/MIT>

<sup>8</sup> <http://twitter.github.io/bootstrap/>

<sup>9</sup> <http://glyphicons.com/>

## 4 Podobné práce

Pluginů pro zobrazení, či úpravu stromu lze na internetu nalézt velké množství. Nelze však říci, že by některý zcela vyhovoval požadavkům vedoucím mé práce. Nyní budou některé z nich představeny a uvedeny jejich přednosti a nedostatky.

### 4.1 jQuery tree view

Plugin [21] pro pouhé zobrazování stromové struktury, jednoduché přidávání složek, mazání položek stromu a složek. Využívá například 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.

### 4.2 jqTree

Součástí funkcionality jqTree [22] 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.

### 4.3 jsTree

Plugin `jstree` [23] má více funkcí než výše zmíněné 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. Bohužel však neumožňuje k položce stromu přidat URI adresu, ani jiný text, což je v zásadní vlastnost požadovaná vedoucím práce.

## 5 Tvorba jednoduchého jQuery pluginu

Základní kostra dokumentu v HTML 5 kterou budeme dále rozšiřovat viz Kód 5.1.

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

Kód 5.1: Šablona HTML dokumentu.

Vytváření pluginu do jQuery je velmi jednoduché. Aby bylo možné použít jQuery, tak stačí do hlavičky dokumentu přidat následující kód (viz Kód 5.2).

```
<script
src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.
min.js" type="text/Javascript"></script>
```

Kód 5.2: Vložení Javascriptu.

Ten načte zdrojové kódy jQuery frameworku ze stránek webu a vloží je do našeho. 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- je tím zrychleno načítání celého webu. Další výhodou je, že při přechodu na novější verzi frameworku lze jenom přepsat odkaz a je hotovo. V porovnání se stahováním knihovny z webu jQuery a následné ukládání na vlastní je to výrazná časová úspora. Odkaz na konkrétní verzi jQuery lze zaměnit za odkaz na verzi s názvem latest neboli poslední (viz Kód 5.3). Takový přístup s sebou nese jisté výhody v tom, že se dá předpokládat, že na nejnovější verzi poběží velký počet webů, a tudíž je vysoce pravděpodobné, že uživatel bude mít jQuery již načteno. Je zde ale i nevýhoda. Byla-li v naší aplikaci použita funkce, označovaná jako zastaralá (deprecated), je velice pravděpodobné, že bude v novější verzi odstraněna a v tu chvíli by aplikace mohla přestat fungovat. Je také možné že v době, kdy byla aplikace napsána, tak funkce deprecated nebyla a stane se to

v budoucnu. Plánuje-li programátor webovou aplikaci vytvořit a nechat běžet bez údržby, nedoporučuje se používat odkaz na poslední verzi jQuery.

```
http://code.jquery.com/jquery-latest.min.js
```

Kód 5.3: Odkaz na nejnovější verzi jquery.

Nyní můžeme všechny funkce jQuery používat dle libosti. Řekněme, že budeme chtít vytvořit plugin, který nám jednou a 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 tag `div`. To je element, který má za úkol pouze obalit nějaký text či bloku kódu, aby byla adresovatelný (např. kaskádovými styly nebo Javascriptem) (viz Kód 5.4). Následně bude pomocí Javascriptu vybírán.

```
<div>Jste přihlášen/a.</div>
```

Kód 5.4: 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 nejdříve proběhlo načtení jQuery. V opačném případě by plugin nefungoval, jelikož knihovna, do které by byl plugin vkládán, by neexistovala.

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. Důvod tohoto čekání je podobný jako u pořadí vkládání Javascriptů. Výběrem dokumentu a zavolání funkce `ready` to můžeme považovat za hotové. Viz Kód 5.5. To je nezbytné, protože jinak bychom mohli skrývat něco, co tam ještě není. Tento případ nemusí být tak závažný, konzole by vypsalala 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. Další kód bude psán do anonymní funkce vytvořené v argumentu funkce `ready` (viz Kód 5.5).

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

Kód 5.5: Funkce čekající na dokončení načítání web stránky.

Plugin budeme psát do jmenného prostoru, `Jquery.fn.jmenoPluginu`. Dále je kód navíc uzavřen do anonymní funkce, která předejde možným kolizím našeho kódu s jiným pluginem, nebo knihovnou. (viz Kód 5.6).

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

Kód 5.6: Anonymní funkce v jQuery pluginu.

Upozornění uživatele pomocí jednoduchého zobrazení a skrytí textu provedeme za pomoci tří funkcí jQuery. Nejdříve prvek skryjeme funkcí `hide()`, aby ho uživatel neviděl dříve než je potřeba. 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 a plugin máme hotov. Pokud bychom v tuto chvíli stránku zobrazili, nic by se nestalo. Musíme plugin totiž zavolat, jinak by nevěděl kdy a kde pracovat (viz Kód 5.7).

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

Kód 5.7: 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*“ (viz Kód 5.8).

```
<!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(){
```

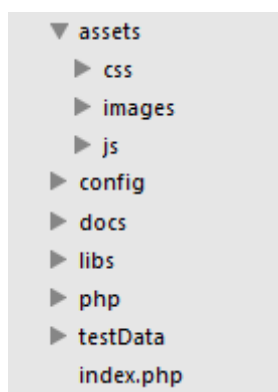
```
(function($){
    $.fn.helloWorldPlugin = function(){
        this.hide();
        this.slideDown(1000);
        this.delay(1000);
        this.slideUp(1000);
    };
})(jQuery);
$('#div').helloWorldPlugin();
</script>
</head>
<body>
    <div>Jste přihlášen/a.</div>
</body>
</html>
```

Kód 5.8: 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 `testData` obsahující `*.sql` soubor s testovacími daty. Do složky `php`, 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<sup>10</sup> pro úpravu vzhledu selectboxu ve stylu Twitter Bootstrap. Složka `docs` zastřešuje soubory s dokumentací. Dále `config` obsahuje nastavení konstant, které se týkají PHP a přístupů do databáze. A nakonec složka `assets`, do které patří kaskádové styly, Javascriptové soubory a obrázky. Je tam také umístěn kód mého pluginu s názvem `jquery.jqTagTree.js`. Vzhled adresářové struktury viz Obrázek 6.1.



Obrázek 6.1: Adresářová struktura pluginu

---

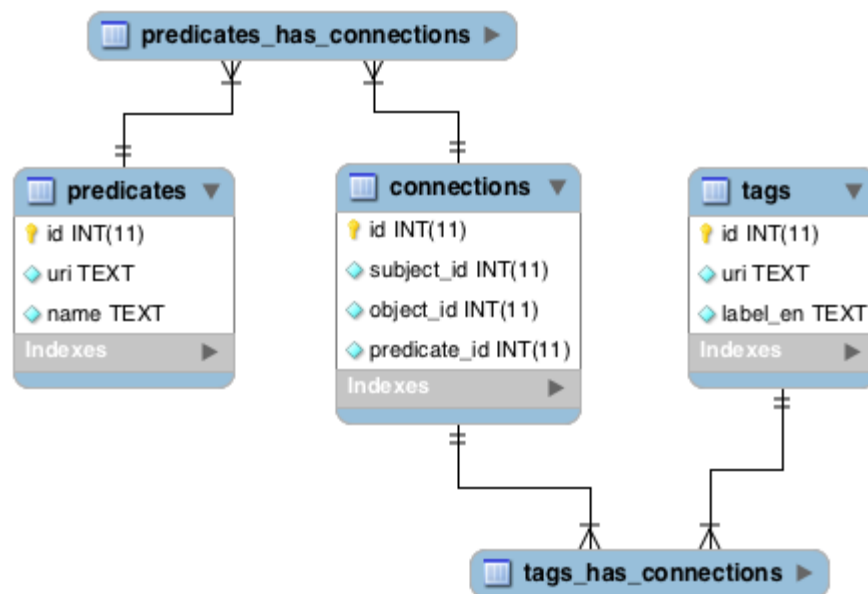
<sup>10</sup> <https://github.com/silviomoreto/bootstrap-select>



# 7 Databáze

## 7.1 Návrh použité databáze

Jeden z požadavků programu bylo zachování stávající struktury databáze. Ta obsahuje tři tabulky. ERA model databáze viz Obrázek 7.1. Jedná se o spojení tabulek štítků a predikátů tabulkou vazeb. Data, která nese tabulka štítků, jsou jméno a URI. Dále obsahuje sloupeček id, který je primárním klíčem. Stejně tak je tomu i u tabulky predikátů i vazeb. Tabulka vazeb pak určuje nadřazené a podřazené tagy pomocí sloupečků `object_id` a `subject_id`. Sloupec `predicate_id` určuje typ vazby mezi jednotlivými štítky. Ve stromové struktuře v prohlížeči je dále zobrazován sloupec se jmény štítků a to `label_en`.



Obrázek 7.1: ERA model databáze

Dále budou uvedeny vazby mezi tabulkami na příkladu. Máme v tabulce štítky se jmény A a B s `id` 1 a 2. Řekněme, že B je potomek A. Pak bude v tabulce vazeb pouze záznam: `object_id = 1`, `subject_id = 2`. Žádná jiná vazba tam není, z čehož vyplývá, že A nemá žádného rodiče, a tudíž je kořenem stromu. Je-li v tabulce predikátů pouze záznam s `id = 1`, pak bude `predicate_id` z tabulky vazeb rovno jedné (viz Tabulka 7.1).

Predicates			Connections				Tags		
Id	Uri	name	Id	Sub_id	Obj_id	Před_id	Id	uri	name
1	http://...	Some name	1	2	1	1	1	http://...	A
							2	http://...	B

Tabulka 7.1: Vzorová data pro demonstraci vazeb

## 7.2 Práce s databází a záměna za vlastní

Ve složce `php` je obsaženo několik souborů pracujících s databází. Veškerá komunikace klienta se serverem v PHP souborech probíhá asynchronně s využitím technologie AJAX.

Všechna logika datové vrstvy aplikace je obsažena v souboru `jqTagTree.db.class.php`, kde se nachází třída pro práci s MySQL databází. Je zde několik základních funkcí usnadňujících komunikaci s databází. Například funkce zkracující název z původního `mysql_query` na pouhé `query` nebo funkce `numberOfRows` vracející počet řádek zadaného dotazu (viz Kód 7.1).

```
private function query($query){//odeslání dotazu do DB
    return mysql_query($query);
}
private function numberOfRows($query){//počet řádek
    $result = $this->query($query);
    return $this->numRows($result);
}
```

Kód 7.1: Jednoduché funkce v `jqTagTree.db.class.php`

Složitější funkce databázové třídy mají na starosti už samotou práci se stromem štítků. Nejdůležitější funkce s názvem `loadFirstLevelFromDb()` načítá vůbec první úroveň stromu. Její činností je pouhé odeslání dotazu na dabázi a vrácení pole s výsledkem. Kořeny stromu se hledají spojením tabulky štítků s tabulkou vazeb pomocí syntaktické konstrukce `LEFT_JOIN`. Následně se ze spojení odeberou indexy, které mají záznam ve sloupečku `subject_id` (viz Kód 7.2). Dotaz je včetně prefixových konstant, jako prevence před překrytím názvů konstant ostatními pluginy, knihovnamí

nebo programátorem implementujícím tento plugin. Konstanty slouží k jednoduché záměně názvů tabulek databáze.

```
public function loadFirstLevelFromDb(){
    $record = $this->readFromDb( //dotaz na databázi
        'SELECT a.'. JQTT_TAG_TABLE_ID.',
        a.'. JQTT_TAG_TABLE_NAME .' as name,
        a.'. JQTT_TAG_TABLE_URI .',
        count(c.'. JQTT_TAG_TABLE_ID.') AS children
        FROM '. JQTT_TAG_TABLE .' a
        LEFT JOIN ' . JQTT_CONNECTIONS_TABLE . ' c ON
        a.'. JQTT_TAG_TABLE_ID.'=
        c.'. JQTT_CONNECTIONS_TABLE_PARENT.'
        WHERE a.'. JQTT_TAG_TABLE_ID .'
        NOT IN (
            SELECT b.'. JQTT_CONNECTIONS_TABLE_CHILD .'
            FROM ' . JQTT_CONNECTIONS_TABLE .' b) GROUP BY
            a.'. JQTT_TAG_TABLE_ID .' order by
            a.'. JQTT_TAG_TABLE_NAME);

    return $record;
}
```

Kód 7.2: Funkce výběru kořenových prvků stromu

## 8 Plugin jqTagTree

Testovacími daty naplněný plugin viz Obrázek 8.1.

### jQuery plugin - jqTagTree

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

Obrázek 8.1: Plugin jqTagTree s testovacími daty

### 8.1 Vykreslování stromu

Tento odstavec má nastínit co se bude dít ještě před samotným vykreslením stromu. Ihned po načtení stránky, kde má plugin jqTagTree běžet, se provede změna nastavení proměnných, zadané uživatelem při volání pluginu. Lze takto změnit použité ikony, cesty k PHP souborům nebo dobu a barvu zvýraznění právě přidaného prvku. Seznam konstant viz Kapitola 9. Následuje zjištění jména kořenového tagu, vložení potřebného HTML a zapnutí kontextového menu.

Najít způsob jak vykreslovat strom, který je zadaný 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. Například je-li dat málo. Pro tento případ to je ale krajně nevhodné. Mnohem lepší způsob je dynamické načítání stromu po jednotlivých úrovních. Kliknutím na libovolný štítek mající své podřazené prvky, se z databáze načtou jeho přímí potomci.

Inspiraci ke tvorbě stromu jsem získal ve knize *jQuery novice to ninja* [24]. Funguje na principu duplikace šablony. Potřebné HTML vložené před začátkem vykreslování je šablona struktury jednoho štítku, která bude později duplikována. Viz Kód 8.1.

```

<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 8.1 : 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`, odpovídající záznamu v databázi. Dále je cíl odkazu nasměrován na adresu, kam má odkazovat. Mezi tagy `<span>` a `</span>` 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 složena z předpony `JQTT`, která ošetřuje problém s překrytím proměnných. 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 8.1.

Stejný postup dynamického načítání objektů z databáze a kopírování jedné šablony je použit i pro načtení vůbec prvního elementu stromu. Při načítání jednotlivých úrovní

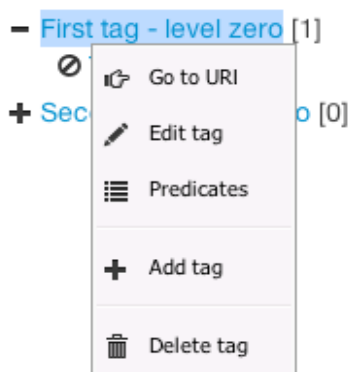
stromu je volána funkce s parametrem představující unikátní identifikátor prvku. Načítá-li se první úroveň stromu, je tento parametr roven nule.

## 8.2 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 obsahuje čtyři položky. V případě, že element obsahuje predikáty, je přidána pátá. Více o predikátech v kapitole 8.3.4. V pluginu jsou odkazy v nabídce rozděleny do třech polí, které se dále spojují. 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 8.2.

### jQuery plugin - jqTagTree



Obrázek 8.2 : 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.

## 8.3 Změny položek stromu

Výše zmíněná 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 do dokumentu a volanou 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 viz Kód 8.2. 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.

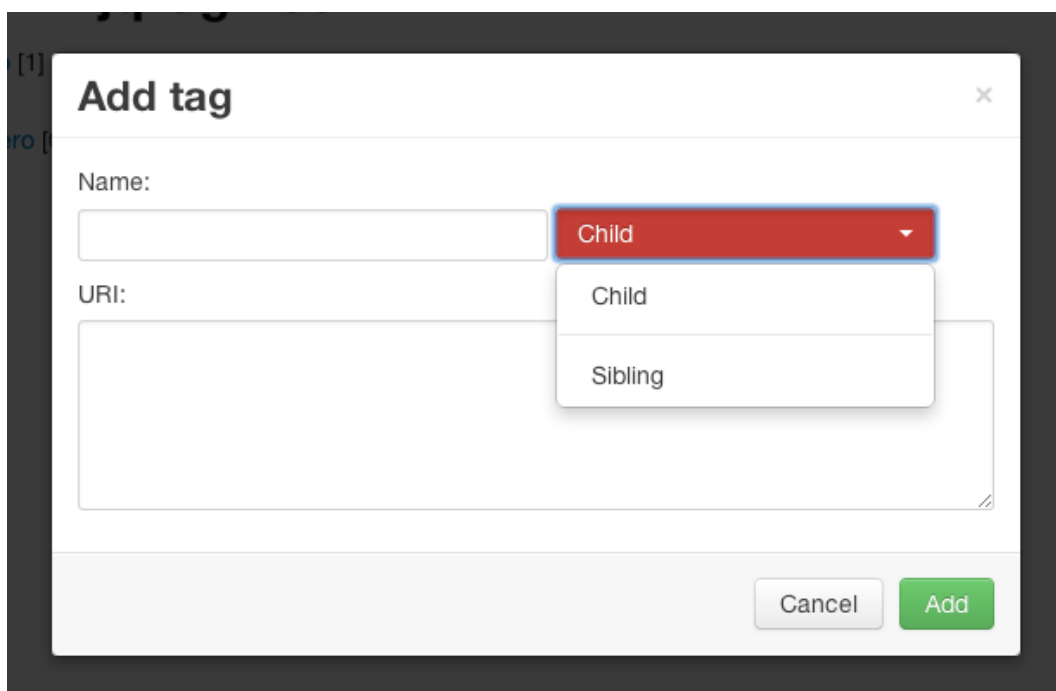
Všechny změny stromové struktury probíhají jak na straně klienta v prohlížeči pomocí Javascriptu, tak v databázi na straně serveru pomocí technologie AJAX a jazyka PHP. Každá změna je interně rozdělena do tří menších funkcí. První upravuje šablonu modálního okna pro potřeby přidávání, úpravy, mazání 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'><!--hlavička modalu->
    <button type='button' class='close' data-
      dismiss='modal' aria-hidden='true'>×
    </button><h3 id='modalLabel'></h3>
  </div>
  <div class='modal-body'></div><!--tělo modalu->
  <div class='modal-footer'><!--patička modalu->
    <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 8.2 : Šablona modálního okna

### 8.3.1 Přidávání

Funkce pro přípravu přidávání štítku upraví a zobrazí modální okno - Obrázek 8.3. 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-select<sup>10</sup>. 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 8.3 : Modální okno pro přidávání štítku

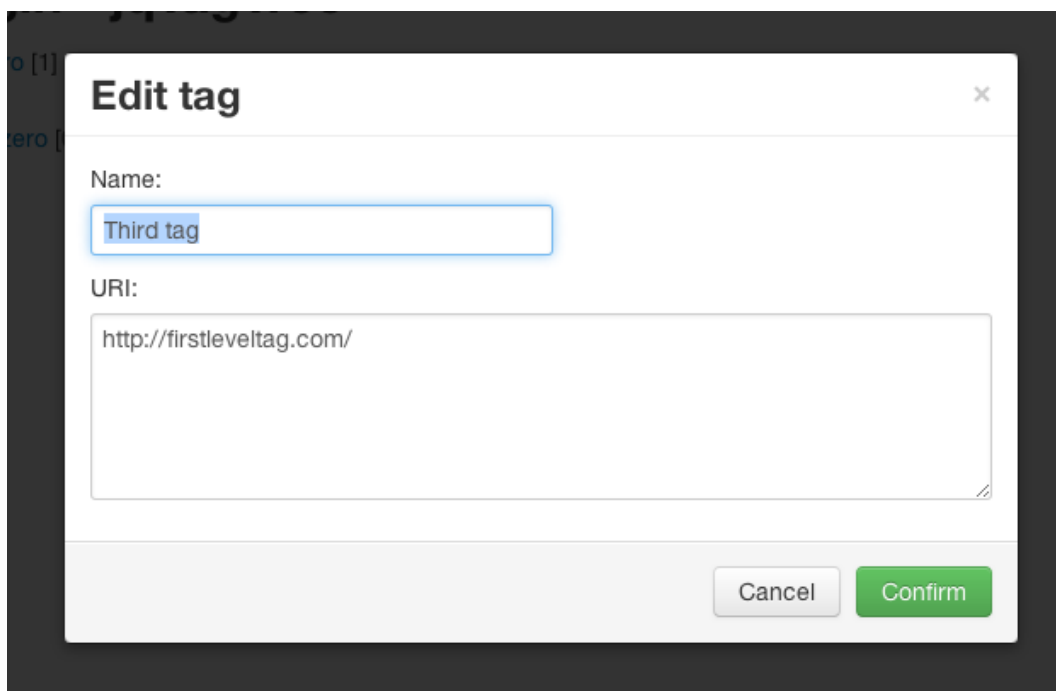
Kliknutím na tlačítko „add“ nastupuje validace vstupů z formuláře. V případě, že jsou řetězce neprázdné a nejsou delší než povolená délka, jsou data odeslána na server. Tam se zjišťuje jméno vybraného predikátu a porovnává s databází. Pokud je správně zadané, tak se zapíše data do tabulky štítků i vazeb. 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. Jiná větev je načtená, ale skrytá úroveň. 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.

### 8.3.2 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 8.4). 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 uloženo do databáze. Nakonec se provede úprava stromu v prohlížeči.

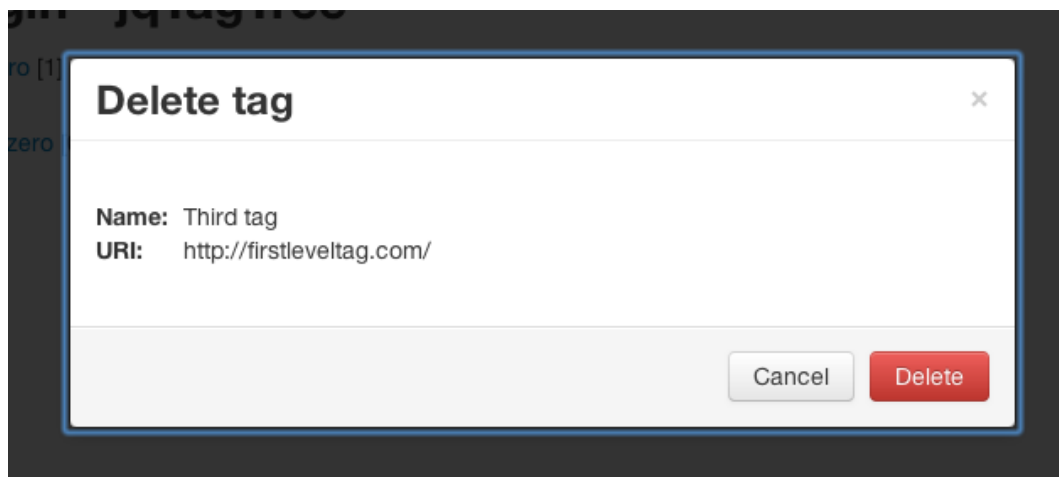


Obrázek 8.4: Modální okno pro úpravu štítků

### 8.3.3 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 8.5). Po potvrzení mazání, je pomocí AJAXu zavolán soubor `delNode.php`. Jeho cílem je smazat štítek včetně jeho potomků. Postup je k tomu 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 8.5: Modální okno pro smazání štítku

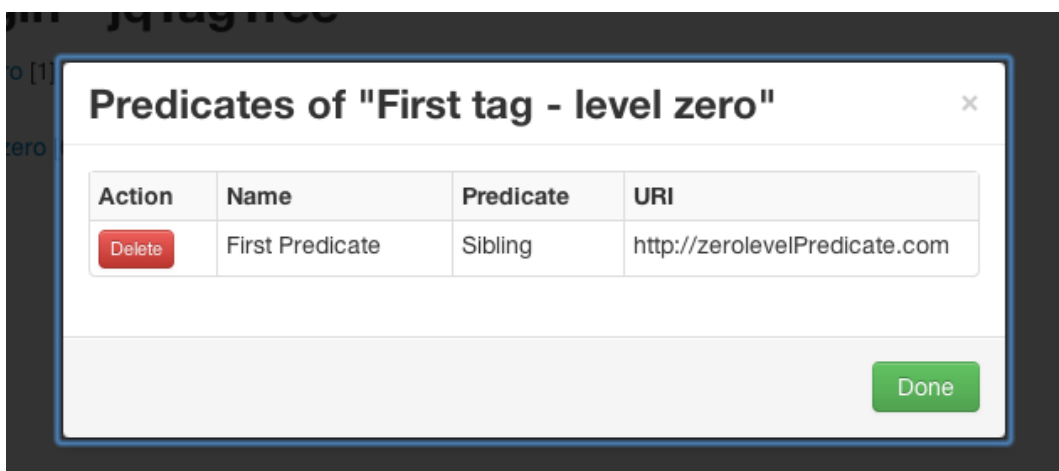
### 8.3.4 Zobrazení predikátů

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 číslem v databázi v tabulce vazeb a sloupečkem predicate\_id 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 8.1). Na toto místo se počet predikátů ukládá při úpravě vzorové šablony, jak bylo zmíněno v kapitole 8.1. 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 počet větší 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 které jsou vypsány predikáty. K vložení tabulky a záznamů do ní, je opět použita funkce `ajax` z `jQuery`. Po stlačení 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 8.6: Modální okno pro zobrazení predikátů

## 9 Použití pluginu ve webové aplikaci

Před použitím pluginu s názvem jqTagTree, musí být nejprve provedeno stažení zdrojových kódů z veřejného repozitáře na githubu<sup>11</sup> (lze samozřejmě použít kódy z CD přiloženého k bakalářské práci). Po rozbalení souboru ve formátu zip uživatel uvidí adresářovou strukturu popsanou v kapitole 6. Je vhodné použít tuto strukturu. V opačném případě bude nutné přenastavit cesty k souborům. Dále je ve složce testData přiložen soubor s příponou sql, který po importu do uživatelem připravené databáze vytvoří tabulky a naplní je testovacími daty. Následně je nezbytné v konfiguračním souboru config/constants.php změnit údaje o přístupu k databázi. Navíc lze v tomto souboru změnit názvy sloupců a tabulek tak, aby odpovídaly databázi. Jsou-li zdrojové kódy ve složce s běžícím php serverem, tabulky importované v databázi a změněny přihlašovací údaje k databázi, spuštění ukázkové stránky nic nebrání.

V případě, že je potřeba změnit strukturu složek, veškeré nastavení cest k souborům je soustředěno na dvě místa. Oboje lze ovlivnit ze souboru index.php. Zaprvé jsou to cesty k souborům kaskádových stylů a Javascriptových souborů. Zadruhé to je umístění souborů pro práci s databází, které lze měnit pomocí nastavení, které je odesíláno pluginu při jeho volání. Tímto způsobem je možné upravit i jiná nastavení pluginu (viz Kód 9.1). Další proměnné, které změní chování pluginu viz Tabulka 9.1.

```
$('#jqTagTree').jqTagTree({
  iconOpen : 'icon-arrow-right', //ikona zavřené větve
  iconClose : 'icon-arrow-down' //ikona otevřené větve
});
```

Kód 9.1: Nastavení vnitřních proměnných pluginu

<sup>11</sup> <https://github.com/vohanka/jqTagTree>

<b>Název proměnné</b>	<b>Činnost proměnné</b>
<b>ajaxLoadTreePhpPath</b>	Cesta k php souboru načítající strom.
<b>ajaxEditNodePhpPath</b>	Cesta k souboru upravující položku stromu.
<b>ajaxAddNodePhpPath</b>	Cesta k souboru přidávající položku stromu.
<b>ajaxAddNodeLinkPhpPath</b>	Cesta k souboru načítající jména predikátů do formuláře pro přidávání.
<b>ajaxDelNodePhpPath</b>	Cesta k souboru odstraňující položku stromu.
<b>ajaxLoadPredicatesPhpPath</b>	Cesta k souboru zobrazující predikáty položky.
<b>iconLoadingPath</b>	Cesta k souboru obrázku značící načítání větve.
<b>iconOpen</b>	Název ikony uzlu k otevření.
<b>iconClose</b>	Název ikony uzlu k otevření.
<b>iconEmpty</b>	Název ikony uzlu který nelze otevřít.
<b>highlightAddColor</b>	Barva zvýraznění právě přidaného prvku.
<b>highlightEditColor</b>	Barva zvýraznění právě upraveného prvku.
<b>highlightDuration</b>	Doba zvýraznění prvku.
<b>slidingDuration</b>	Doba rozevírání a zavírání větve stromu.

Tabulka 9.1: Konstanty pluginu upravitelné uživatelem..

## 10 Závěr

Cílem mé bakalářské práce bylo naprogramovat zásuvný modul do knihovny jQuery. Práce splňuje celkem specifické požadavky vedoucího práce, a proto nejspíše nebude, tak jak je, použitelná pro širokou veřejnost. Je tomu tak především kvůli uspořádání tabulek v databázi. Práce je ale veřejně uložena na serveru github.com a je tedy možné využít funkce fork, která je zde nabízena a upravit si ji k vlastním potřebám. Funkčnost pluginu byla ověřena na webové aplikaci, která je přiložena k bakalářské práci (viz kapitola 9).

Aplikace umožňuje vkládání, úpravu a mazání štítků z kontextového menu. Navíc lze z téhož menu přejít na URI adresu štítku. Po domluvě s vedoucím práce práce p. Ing. Dostalem jsem nad rámec zadání zrealizoval možnost přidání, zobrazení a mazání predikátů štítku, neboť to vhodně doplňuje funkcionalitu pluginu.

Při práci bylo využito knihoven jQuery, jQuery UI a Twitter Bootstrap a jelikož jsem v pluginu použil prefixové pojmenování proměnných a jmenných prostorů, tak je zaručena kompatibilita při použití jiných pluginů v jedné webové aplikaci. S využitím Bootstrapu jsem se snažil vytvořit intuitivní a příjemné uživatelské rozhraní. Rozšíření Bootstrap-select mi k tomu pomohlo i v případě select boxu u přidávání štítků.

Zadávající povolil umístění zdrojových kódů do webové služby github pod většinou licencí umožňující volné použití. Tím se mi zjednodušila záloha a navíc je takto moje práce veřejně přístupná široké veřejnosti. Takový plugin doposud neexistoval a může tak být dále používán.

Důvodem, proč měl být tento plugin vytvořen, je usnadnění změn dat v databázi generované jiným softwarem. Jsem přesvědčen, že úprava dat v přehledném stromovém zobrazení je daleko jednodušší a přehlednější, než v databázi. Díky tomu se domnívám, že cíl mé bakalářské práce byl splněn. Jsou splněny i všechny body zadání.

## Použitá literatura

1. W3SCHOOLS. *HTML Reference* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://www.w3schools.com/tags/>
2. W3SCHOOLS. *Introduction to XML* [online]. 2013 [cit. 2013-05-03]. Dostupné z: [http://www.w3schools.com/html/html\\_xhtml.asp](http://www.w3schools.com/html/html_xhtml.asp)
3. How well does your browser support HTML5?. *The HTML5 test* [online]. 2012 [cit. 2013-05-04]. Dostupné z: <http://html5test.com/results/desktop.html>
4. W3SCHOOLS. *HTML new elements* [online]. 2013 [cit. 2013-05-03]. Dostupné z: [http://www.w3schools.com/html/html5\\_new\\_elements.asp](http://www.w3schools.com/html/html5_new_elements.asp)
5. HTML5 TUTORIAL. *Input type: Email, URL, Phone* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://www.html5tutorial.info/html5-contact.php>
6. W3C. *Cascading style sheets* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://www.w3.org/Style/CSS/>
7. MOZILLA. *Mozilla developer network* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <https://developer.mozilla.org/en-US/docs/JavaScript>
8. FLANAGAN, David. *JavaScript*. O'Reilly Media, Inc., 1998.
9. JOYENT. *Node.js* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://nodejs.org/>
10. ASHKENAS, Jeremy. GITHUB. *List of languages that compiles to JS*. [online]. 2013 [cit. 2013-05-03]. Dostupné z: <https://github.com/jashkenas/coffee-script/wiki/List-of-languages-that-compile-to-JS>
11. ASHKENAS, Jeremy. *Coffeescript*. 2012.
12. COFFEESCRIPT. *CoffeeScript* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://coffeescript.org/>
13. THE PHP GROUP. *PHP: Classes and Objects (PHP 4) - Manual* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://php.net/manual/en/oop4.php>
14. THE PHP GROUP. *PHP: Classes and Objects - Manual* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://php.net/manual/en/language.oop5.php>
15. O'SULLIVAN, Bryan. Making sense of revision-control systems. *Communications of the ACM* [online]. 2009-09-01, vol. 52, issue 9, s. 56-62 [cit. 2013-05-03]. DOI: 10.1145/1562164.1562183. Dostupné z: <http://portal.acm.org/citation.cfm?doid=1562164.1562183>

16. COLLINS-SUSSMAN, Ben; FITZPATRICK, Brian; PILATO, Michael. *Version control with subversion*. O'Reilly Media, 2007.
17. CEDERQVIST, Per, et al. Version management with CVS. *Available online with the CVS package*. Signum Support AB, 1992
18. Centralized vs Decentralized Version Control: 2010 vs 2012 – tecosystems. *Redmonk* [online]. 2012 [cit. 2013-05-03]. Dostupné z: <http://redmonk.com/sogradey/2012/11/05/dvcs-2012/>
19. Websites using jQuery. *Build with* [online]. 2013 [cit. 2013-05-04]. Dostupné z: <http://trends.builtwith.com/websitelist/jquery>
20. *BuiltWith Technology Lookup: Trends* [online]. 2013-4-12 [cit. 2013-04-17]. Dostupné z: <http://trends.builtwith.com/javascript/JQuery>
21. Jzaefferer/jquery-treeview · GitHub. *Github.com* [online]. 2007 [cit. 2013-05-03]. Dostupné z: <https://github.com/jzaefferer/jquery-treeview>
22. Mbraak/jqTree · GitHub. *Github.com* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <https://github.com/mbraak/jqTree>
23. *JsTree* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://www.jstree.com/>
24. CASTLEDINE, Earle a Craig SHARKIE. *JQuery: novice to ninja*. 2nd ed. Collingwood, Vic.: SitePoint, 2012, xxv, 453 p. ISBN 978-098-7153-074.



## Seznam kódů

Kód 2.1: Vstupní pole formuláře v HTML 5 .....	3
Kód 2.2: Příklad XML kódu .....	7
Kód 2.3: Příklad JSON kódu .....	8
Kód 3.1: Porovnání jQuery a Javascriptu .....	10
Kód 5.1: Šablona HTML dokumentu. ....	13
Kód 5.2: Vložení Javascriptu.....	13
Kód 5.3: Odkaz na nejnovější verzi jquery.....	14
Kód 5.4: Zobrazovaný div. ....	14
Kód 5.5: Funkce čekající na dokončení načítání web stránky.....	15
Kód 5.6: Anonymní funkce v jQuery pluginu. ....	15
Kód 5.7: Volání testovacího pluginu. ....	15
Kód 5.8: Jednoduchý jquery plugin. ....	16
Kód 7.1: Jednoduché funkce v jqTagTree.db.class.php .....	19
Kód 7.2: Funkce výběru kořenových prvků stromu .....	20
Kód 8.1 : Šablona prvku stromu .....	22
Kód 8.2 : Šablona modálního okna.....	24
Kód 9.1: Nastavení vnitřních proměnných pluginu.....	29

## Seznam obrázků

Obrázek 2.1: Žebříček prohlížečů seřazený podle počtu bodů.....	3
Obrázek 2.2: Změna klávesnice na dotykových zařízeních.....	4
Obrázek 6.1: Adresářová struktura pluginu .....	17
Obrázek 7.1: ERA model databáze.....	18
Obrázek 8.1: Plugin jqTagTree s testovacími daty .....	21
Obrázek 8.2 : Kontextová nabídka s predikáty.....	23
Obrázek 8.3 : Modální okno pro přidávání štítku .....	25
Obrázek 8.4: Modální okno pro úpravu štítků .....	26
Obrázek 8.5: Modální okno pro smazání štítku .....	27
Obrázek 8.6: Modální okno pro zobrazení predikátů .....	28

## Seznam tabulek

Tabulka 2.1: Porovnání syntaxe Javascriptu a Coffeescriptu. ....	6
Tabulka 7.1: Vzorová data pro demonstraci vazeb.....	19
Tabulka 9.1: Konstanty pluginu upravitelné uživatelem. ....	30