VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

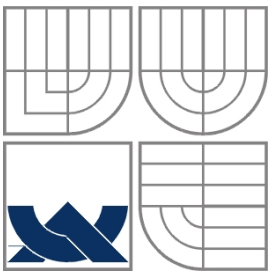
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATOVÁ MŘÍŽKA VE WEBOVÉM PROHLÍŽEČI

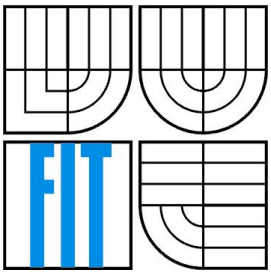
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE RICHARD MIKÚŠEK
AUTHOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATOVÁ MŘÍŽKA VE WEBOVÉM PROHLÍŽEČI

DATA GRID IN THE WEB BROWSER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RICHARD MIKÚŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ MÁČEL

BRNO 2011

Abstrakt

Tato práce se zabývá návrhem a implementací komponenty datové mřížky v prostředí webového prohlížeče. Práce popisuje problematiku moderních webových technologií a ich využití při implementaci interaktivních komponent. Podstatná část popisuje problematiku získávaní dat, ich uložení, rozložení jednotlivých elementů, události spřevázející životní cyklus komponenty. Reálné využití navrhované aplikace je možné například jako grafické uživatelské rozhraní pro databázi nebo jiný datový zdroj.

Abstract

This work deals with design and Implementation of data grids component in a web browser environment. The work describes the problems of modern web technologies and their use in implementation of interactive component. A substantial part devoted to the issue of data mining, the deposit, the distribution of elements , the events accompanied by life cycle component. Real use of designed application can be such as a graphical user interface for database or other data source.

Klíčová slova

Web, Webová komponenta, Java Script, Objektové programování, Dojo, ExtJS, Mootools, Bohaté Internetové Aplikace, Datová Mřížka, Klientská Cache

Keywords

Web, Web Browser, Java Script, Object Oriented Programming, Dojo, ExtJS, Mootools, Rich Internet Applications, Data Grid, Browser Cache

Citace

Richard Mikúšek: Datová mřížka ve webovém prohlížeči, bakalářská práce, Brno, FIT VUT v Brně, 2011

Dátová mriežka vo webovom prehliadači

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením...
Další informace mi poskytli...
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Richard Mikúšek
18.05.2011

Poděkování

Děkuji svému vedoucímu panu Ing. Lukášovi Máčelovi za odbornou pomoc a časovou dostupnost.
Dále bych rád poděkoval své rodině a přátelům za cenné rady a psychickou podporu při spracování bakalářské práce.

Obsah

Obsah....1
1 Úvod....4
2 Terminológia a prehľad používaných technológií....5
2.1 Komponenta z pohľadu softwarového inžinierstva....6
2.2 Koncepty....7
2.2.1 Rich Internet Apllications....7
2.2.2 Document Object Model....8
2.3 Jazyky....9
2.3.1 HTML....9
2.3.2 CSS....10
2.3.3 JavaScript....11
2.3.4 AJAX....12
2.4 Serverové technológie....14
2.5 Formáty dát....16
2.5.1 JSON....16
2.5.2 XML....17
2.6 Frameworky....18
2.6.1 Dojo toolkit....18
2.6.2 ExtJS....19
3 Popis existujících rešení....19
3.1 Popis komponenty....19
3.2 Vytvorenie dátovej mriežky knižnici DOJO....20
3.2.1 Dátová vrstva - Získavanie dát....20
3.2.2 Vytvorenie samotnej mriežky....20
3.3 Zhodnotenie frameworkov....22
4 Návrh komponenty dátovej mriežky....22
4.1 Služby komponenty....23
4.1.1 Sorting....23
4.1.2 Editing....24
4.1.3 Filter....26
4.1.4 Live search....27
4.1.5 Column-Sizing....27
4.2 Dátová vrstva....28

4.3 Grafické užívateľské rozhranie....28
4.4 Udalosti....29
4.5 Vytvorenie komponenty....31
4.6 Navigácia....32
5 Implementácia....32
5.1 Navigácia....33
5.2 Sorting....34
5.3 Editácia....34
5.4 Dátová vrstva....34
6 Záver....35
Literatúra....36
Zoznam obrázkov a výpisov....37
Zoznam použitých skratiek....38
Prílohy....38

1 Úvod

V poslednom desaťročí sa Internet masovo rozšíril do celého sveta a čím ďalej sa stáva každodennou potrebou miliónov ľudí. Prenos dát po Internete sa stále zlepšuje a rapídne sa znásobuje rýchlosť prenosu oproti rokom minulým. Od jeho vzniku sa jeho rýchlosť znásobila niekoľko stonásobne. So zvýšením rýchlosti sa zrýchlila práca s ním a otvorili sa možnosti používať technológie a koncepty, ktoré by s pomalým pripojením mali obmedzujúci význam a priniesli by viac premrhaného času čakaním na výsledok, ako efektívnu prácu. Zvýšenie rýchlosti Internetu spôsobilo masívny vývoj webových aplikácií a snahu dostávať na web všemožné aplikácie, ktoré boli výsadou desktopu.

Za posledných pár rokov zaznamenali dynamické webové technológie veľký boom. Začali sa rapídne vyvíjať a používať pre krajšiu a efektívnejšiu prácu vo webovej platforme. Vyvinul sa nový koncept Web 2.0, ktorý bol zadefinovaný v roku 2004 a zmenil chápanie webu. Zmenil pohľad na web, ktorý nemá byť reprezentovaný len ako zhluk textu a grafiky ale ako prostredie poskytujúce interaktivitu na čo najviac typoch platforiem. Prináša so sebou koncepty ako sociálne siete, blogy, galérie, knižnice a veľké množstvo operácií prístupných z webového rozhrania. Tým sa zmenil kľúčový vzťah čitateľa k médiu, kde zo statického prístupu prehliadania stránok sa stal obojsmerný tok, kde užívateľ na webe publikuje, prezentuje sa, jednoducho je „tvorcom webu“. Tento koncept sa stále vyvíja a odhaľuje nové cesty vedúce k lepším a efektívnejším riešeniam.

Táto bakalárska práca sa zaoberá interpretáciou a manipuláciou s dvojrozmernými dátami. Dáta sa reprezentujú jednotlivými záznamami, ktoré majú viacero atribútov. Sú väčšinou spracúvané a ukladané do databáz pre potrebu neskoršej manipulácie s nimi. Avšak práca s databázami nie je úplne zrozumiteľná pre klasického užívateľa PC. Ideálne a zrozumiteľné zobrazenie takýchto dát umožňuje komponenta dátovej mriežky, ktorá sa používa v počítačovom svete už zopár rokov. Vo webových prehliadačoch je klasicky reprezentovaná statickou tabuľkou. Medzi masovo najznámejšiu dátovú mriežku vo svete desktopových aplikácií patrí rozhranie programu Microsoft Excel, ktorý ponúka široké možnosti manipulácie s dátami. Umožňuje užívateľovi dáta medzi sebou prepájať, vytvárať závislosti, aplikovať rôzne matematické a iné funkcie. Avšak v oblasti webových aplikácii sú možnosti obmedzené a dosiahnutie takejto interaktivity ťažko dosiahnuteľné.

Práca je rozdelená do šiestich kapitol. Od čitateľa sa očakáva aspoň základný prehľad v oblasti webových technológií a internetu.

Prvá kapitola sa venuje úvodu do problematiky a motiváciou.

V druhej kapitole sú predstavené použité technológie a vysvetlené konkrétne koncepty.

Tretia kapitola opisuje implementácie obdobných komponent a ich prístup k dátam a komponentám navzájom.

Vo štvrtej kapitole je opísaný návrh vlastnej komponenty dátovej mriežky inšpirovaný zaužívanými riešeniami s pridaním ďalších vhodných doplnkov.

Piata kapitola opisuje implementáciu, jej úskalia a problémy.

Posledná, šiesta kapitola, obsahuje záver a zhrnutie významu práce a jej výsledkov. Na konci práce sú potom uvedené použité zdroje, použité obrázky a prílohy bakalárskej práce.

Motivácia pre výber tejto témy plynula z narastajúceho stretávania sa s modernými webovými technológiami. Vo svete webu je badateľný progres doslova každý deň, čo ma fascinuje a pramení z toho chuť naučiť sa pracovať s týmito technológiami. Najväčším vplyvom pre mňa boli aplikácie firmy Google, ktoré svojou funkčnosťou, intuitívnosťou a prístupnosťou majú pre mňa väčší význam ako obdobné aplikácie vo svete desktopu.

2 Terminológia a prehľad používaných technológií

V tejto kapitole sú bližšie popísané technológie použité pre implementáciu komponenty dátovej mriežky a bližšie teoretické znalosti z problematiky. V oblasti vývoja webových aplikácií sa jedná o technológie tradičné a najviac používané v daných situáciách a implementačných problémoch. Ich výber závisel na tom, aby boli najviac rozšíriteľné, fungujúce vo všetkých webových prehliadačoch a multiplatformne podporované. Za vybranými technológiami stoja veľké rady vývojárov a široká škála projektov. Výber implementačných nástrojov vo veľkej miere závisí aj od množstva literatúry a dostupných zdrojov.

Skúsenosti s implementáciou evokovali u vývojárov potrebu uľahčovať rutinne sa opakujúce problémy a rozdiely medzi rôznymi platformami vývojom knižníc a frameworkov. Tieto nástroje majú za úlohu ušetriť čas pri implementácii tým, že v sebe ukrývajú zaužívané postupy a návrhové vzory pri riešení klasických programátorských problémov. V tejto práci budem využívať JavaScriptových frameworkov, ktoré vyriešia odlišnú implementáciu JavaScriptu a asynchrónnych prenosov AJAXu vo všetkých webových prehliadačoch a umožnia sa sústrediť na jadro problému namiesto zameriavania sa na fungujúce zabehnuté princípy a implementačné postupy.

Na serverovú réžiu som zvolil jazyk PHP v kombinácii s relačným databázovým systémom MySQL. Použitie týchto vcelku „tradičných“ technológií záviselo vo veľkej miere na tom, že sú

podporované väčšinou hostingových služieb a ich podpora je bezplatná. Tento fakt zaručuje vysokú prenositeľnosť a znovupoužiteľnosť do budúcna.

Čo sa týka užívateľského rozhrania, výber bol celkom jasný, ide o klasické technológie na strane klienta. O sémantiku obsahu sa stará jazyk HTML, vizuálny štýl jeho elementov zabezpečujú kaskádové štýly CSS a napokon o funkcionalitu, zvýšenie interakcie a zlepšenie užívateľskej skúsenosti sa stará jazyk JavaScript.

2.1 Komponenta z pohľadu softwarového inžinierstva

Komponentne založené softwarové inžinierstvo zaznamenáva v posledných rokoch veľký rozmach hlavne medzi webovskými aplikáciami, desktopovými a grafickými aplikáciami. Hlavným cieľom komponent je ich použiteľnosť a znovu-použiteľnosť. Komponenty sa spájajú s inými komponentami a tvoria softwarový systém.

Komponenta je softwarový balík alebo modul ktorý zapúzdruje skupinu príbuzných funkcií alebo dát. Všetky systémové procesy sú umiestnené v oddelených komponentách, takže všetky dáta a funkcie vnútri každej komponenty sú sémanticky príbuzné. Kvôli tomuto princípu sú často nazývané ako modulárne a kohezívne. Kvôli lepšej koordinácii systému komunikujú komponenty medzi sebou pomocou rozhrania. Keď komponenta ponúka služby zbytku systému, vyberie si rozhranie ktoré definuje služby ktoré môžu byt použité a spôsob akým môžu byť použité. Toto rozhranie sa môže zdať ako označenie komponenty - klient nepotrebuje vedieť nič o vnútornúch procesoch komponenty (implementácii) za účelom použiť to. Ďalšou výhodou komponenty je *nahraditeľnosť,* ktorá umožňuje nahradiť komponentu nejakou inou, napríklad aktualizovanou verziou alebo nejakou alternatívnou bez pádu systému v ktorom komponenta pracuje[10].

Rozhranie komponenty ponúka v našom prípade *kontajner*. Kontajner je objekt, ktorý uchováva iné objekty.[1] Zvyčajne je implementovaný ako nejaká dátová štruktúra ako napríklad pole, zoznam, strom, a veľa iných. V prípade webovej komponenty ide o grafické kontajnery, ktoré zhlukujú jednotlivé widgety dokopy. V tomto prípade má grafický kontajner rovnaké správanie ako klasický s rozdielom, že za svojich potomkov má interaktívne webové elementy namiesto klasickýh dátových štruktúr.

Životný cyklus komponenty zahŕňa všetky aktivity od špecifikácie a analýzy požiadaviek po softwarový návrh a samotnú implementáciu a testovanie.

1 Viz.: http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/container.html

2.2 Koncepty

2.2.1 Rich Internet Apllications

Rich Internet Apllications (RIA) je súhrnné označenie pre webové technológie, ktoré sa používajú pri tvorbe webových stránok. Tento koncept je reakciou na obmedzenú funkcionalitu a interaktivitu zabehnutých webovských technológií ako HTML a CSS. Preto je nutné hľadať nové riešenia, ktoré tieto obmedzenia odstránia, jedných z nich sú práve RIA, ktoré nám poskytujú priestor pre vývoj týchto riešení a snažia sa priblížiť „bohatými“ možnosťami desktopovým technológiám.

Súčasné aplikácie musia riešiť dva primárne problémy. Musia sa vyrovnať s HTTP protokolom a jeho modelom žiadosť/odpoveď. I najmenšia zmena stavu na klientovi musí vyvolať žiadosť na server, ktorý ho obslúži a vráti späť všetky predchádzajúce dáta. Druhým problémom sú obmedzené možnosti prezentačných technológií (HTML, CSS) pre kompaktnejšie grafické rozhrania.

Na jednej strane je teda množina žiadostí a na druhej strane konečné množstvo súčasných technológií. Za Rich Internet Applications preto považuje také aplikácie, ktoré dokážu splniť aj tie najnáročnejšie požiadavky. Medzi ich hlavné znaky patria animácie, väčšie možnosti grafických efektov, práca s audio a video súbormi. Celá réžia RIA sa odohráva na strane klienta bez inštalačného procesu alebo po nainštalovaní pluginu.

Najznámejšie technológie, ktoré sa v súčasnosti využívajú na tvorbu týchto aplikácií sú:

- AJAX (Asynchronous JavaScript and XML)
- Adobe Flash
- Java
- Microsoft Silverlight

AJAXu sa budem bližšie venovať v kapitole o skriptovacích technológiách.

Adobe Flash[2], podobne ako Microsoft Silverlight[3] sú univerzálne klientské technológie, ktoré umožňujú beh interaktívnych aplikácií. Ich veľkou výhodou je možnosť tvorby užívateľských

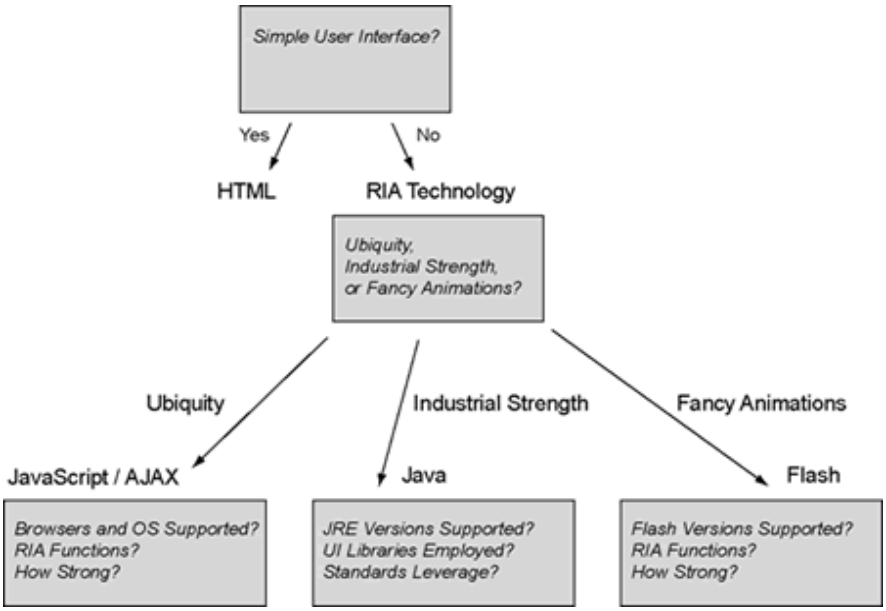
2 http://www.adobe.com/products/flashplayer/

3 http://www.silverlight.net/

rozhraní. Nevýhodou je ich absencia v prehliadačoch, do ktorých sa musia dodatočne nainštalovať zásuvné moduly schopné spustiť tieto prvky.

Java[4] je ďalším jazykom využívajúcim koncept RIA. Vo webových stránkach beží ako applet používajúci Java Virtual Machine, ktorý prekladá kód aby bol prístupný a zrozumiteľný vo všetkých platformách a prehliadačoch. Pre prácu s Java appletmi je tiež treba inštalovať zásuvné moduly.

Obrázok 2.1: **Vzťahy a vlastnosti kľúčových RIA technológií**[5]



2.2.2 Document Object Model

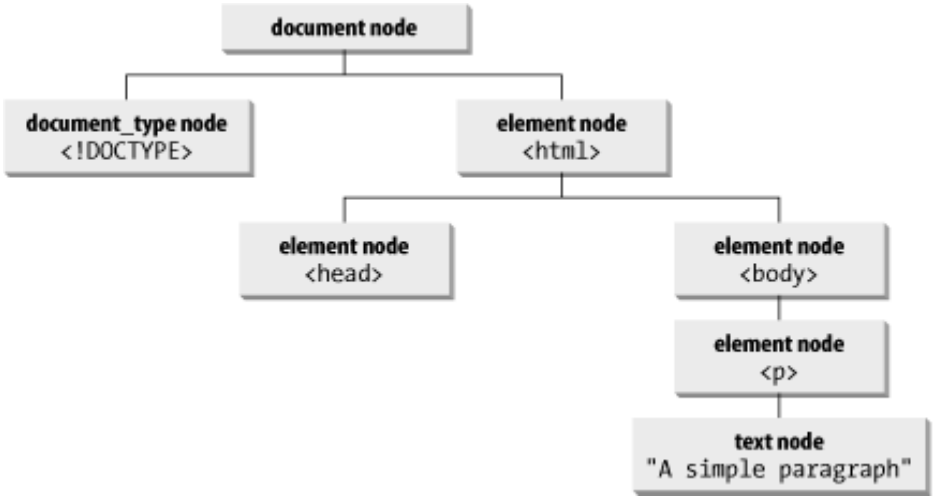
Ďalšou technológiou, ktorá sa využíva na klientskej strane je DOM (Document Object Model). Objektový model dokumentu je API – Aplication Programming Interface (aplikačné programové rozhranie). Toto rozhranie umožňuje pracovať s webovou stránkou samostatne a DOM umožňuje dynamicky pristupovať k jednotlivým objektom XML alebo HTML dokumentu a pracovať s nimi. Týmito objektmi sú elementy, atribúty, text, komentáre, atď. Model DOM reprezentuje dokument ako hierarchicky usporiadanú stromovú štruktúru zloženú z jednotlivých uzlov. DOM umožňuje na presne určené miesto pridať akýkoľvek obsah, toto mu umožňuje napríklad JavaScript, o ktorom sa budem zmieňovať neskôr. DOM je však implementovaný i v ďalších skriptovacích jazykoch. Vznikol ako reakcia na rozdielnu manipuláciu prehliadačov s HTML dokumentami. Tento koncept je

4 Viz.: http://en.wikipedia.org/wiki/Java_applet

5 Viz.: http://www.javalobby.org/articles/ajax-ria-overview/

jazykovo a platformne nezávislý a implementovaný do všetkých populárnych webových prehliadačov.

Obrázok 2.2: **Hierarchická stromová štruktúra modelu DOM**[6]



2.3 Jazyky

2.3.1 HTML

Prvý jazyk, ktorý sa používal pre definíciu štruktúry WWW dokumentov je HTML (HyperText Markup Language). HTML je značkovací jazyk, oficiálny publikovací jazyk pre web. Jeho hlavnou úlohou je rozdelenie obsahu dokumentu podľa sémantickej významnosti do značiek pre ich lepšie ďalšie spracovanie a manipuláciu. Vplyvom nekontrolovateľného rozvoja WWW sa do HTML dostali i značky pre definíciu vzhľadu. V roku 1999 prehlásilo W3C jazyk XHTML sa zástupca jazyka HTML.

HTML ponúka autorom dokumentov tieto možnosti:

- publikáciu online štruktúrovaných dokumentov s nadpismi, podnadpismi, odstavcami, formátovaním, tabuľkami, formulármi, zoznamami, atď.,
- získavanie online dokumentov a vzájomné prepojovanie dokumentov pomocou hypertextových odkazov,

6 http://docstore.mik.ua/orelly/webprog/dhtml/ch01_06.htm

- navrhovanie formulárov ako súčasť dokumentov pre zaistenie vkladania užívateľských dát a prenos dát na server k ďalšiemu spracovaniu,
- vkladanie ďalších formátov hypermediálnej povahy priamo do dokumentov – tabuľky excel, dokumenty, pdf, video, zvuk, animácie, atď.

Výsledkom písania kódu v HTML je textový ASCII súbor. Vonkajším znakom HTML je súbor s príponou .html (príp. .htm). HTML súbory je možné vytvárať v akomkoľvek textovom editore, v súčasnosti však existujú špecializované programy na tvorbu týchto dokumentov, ktoré intuitívne navádzajú užívateľa a ten si môže jednoducho sám vytvoriť webové stránky.

Za dôležité rysy HTML pokladám:

- podporu pre tvorbu dokumentov s rešpektovaním národných zvyklostí rôznych krajín sveta,
- zaistenie dobrej prístupnosti obsahu dokumentu pre osoby so zvláštnymi potrebami,
- kvalitná podpora tabuliek,
- kombinované dokumenty,
- štýly dokumentu (oddelenie definície vzhľadu dokumentu od obsahu),
- podpora skriptovacích jazykov

Jazyk HTML vyvinul Tim Berners-Lee v laboratóriách CERNu v roku 1990. Jeho prvotným zámerom pri vývoji bolo prenášať medzi všetkými zariadeniami pripojenými na web. Jazyk sa postupne rozširoval s vývojom samotného webu a v roku 1995 sa dočkal významného momentu, v ktorom sa začalo pracovať na jeho špecifikácii. Postupne sa stal jazykom úplne prenositeľným a nezávislým na platforme a prehliadačoch. V súčasnosti sa o jeho vývoj a optimalizáciu stará W3C (World Wide Web consortium).

2.3.2 CSS

V súvislosti s tvorbou webových stránok sú asi najčastejšie spomínaným jazykom kaskádové štýly. Pôvodný návrh HTML počítal s tým, že formátovanie dokumentu bude prebiehať priamo v dokumente pomocou formátovacích tagov a atribútov jednotlivých elementov. Kaskádové štýly

(CSS) vznikli okolo roku 1997 a na trh ich uviedla spoločnosť Microsoft.[7] CSS predstavujú kolekciu metód pre grafickú úpravu webových stránok. Samotný názov kaskádové štýly upozorňuje na možnosť vrstvenia definícií štýlov.

CSS sa využívajú hlavne v prípadoch, kedy autori stránok píšu veľké množstvo stránok a potrebujú ušetriť čas tým, že si vytvoria jeden formátovací súbor a do každej webovej stránky tento súbor pripoja odkazom naň.

Možnosti CSS[8]:

- nastavenie ľubovoľnej a presnej veľkosti písma,
- odsadenie prvého riadku odstavca, nastavenie riadkovania,
- zrušenie alebo zväčšenie prázdneho priestoru po odstavci,
- automatické formátovanie nadpisov,
- zvýrazňovanie odkazov,
- grafické odrážky,
- zviditeľnenie určitých častí textu,
- nastavenie pozadia (stránky, tabuľky, odstavca, atď.),
- a mnoho ďalších.

Najväčšou výhodou CSS je v ich automatickom fungovaní, vzhľad celej webovej stránky je možné deklarovať len v jednom súbore. Odpadá zdĺhavé kopírovanie a neprehľadnosť.

2.3.3 JavaScript

Hlavným skriptovacím jazykom na strane klienta je JavaScript, ktorý funguje, pokiaľ užívateľ nemá vypnutú podporu JavaScriptu v prehliadači. Je interpretovaný vo webových prehliadačoch ako súčasť webových stránok[4]. JavaScript sa začal používať v časoch pomalého pripojenia na kontrolu formulárov pred odoslaním na server (odpadá HTTP žiadosť a čakanie na odpoveď v prípade chybne zadaných dát). Syntax patrí do rodiny jazykov C/C++/Java.

7 [1 – s. 92]

8 http://www.jakpsatweb.cz/css/css-uvod.html

Javascript je jazyk[9]:

- interpretovaný,
- objektový,
- závislý na implementácii v prehliadači,
- podobný jazyku C a Java

JavaScriptový kód môže byť vložený priamo v HTML súbore alebo v externom súbore. Sú ním najčastejšie ovládané rôzne interaktívne prvky užívateľského rozhrania alebo animácie a efekty s textom alebo obrázkami. Syntax patrí do rodiny jazykov C/C++/Java. S programovacím jazykom Java, ako vyplýva z názvu, nemá nič spoločné, aj keď syntax je trochu podobná.

2.3.4 AJAX

V súčasnosti sa do webových stránok pridáva veľké množstvo webových aplikácií. Podobne ako aj v staviteľstve sú jednotlivé druhy materiálov nahrádzané inými, i vo webových stránkach bolo potrebné vyvinúť niečo, čo by im dávalo nový štýl, nový lepší prístup. Týmto nástrojom boli Ajaxové aplikácie.

AJAX (Asynchronous JavaScript and XML) je obecné označenie pre technológie vývoja interaktívnych klientských webových aplikácií. AJAX nie je sám o sebe technológioul, je to návrhový vzor pre RIA. Ajax umožňuje webovej aplikácii získať dáta zo serveru na pozadí a bez zmeny chovania vzhľadom k zobrazenej stránke. Je akýmsi sprostredkovateľom v klasickom modeli otázka/odpoveď, kde sa obsah načíta ako nová stránka po užívateľskej aktivite. Takto sa môže zamerať len na určitú časť stránky a len tú stiahnuť bez toho aby sa zbytočne sťahovala celá stránka.

Používanie AJAXových techník zvyšuje interaktivitu a dynamiku rozhraní webových stránok. Dáta sú zvyčajne získavané pomocou objektu XMLHttpRequest. Táto technológia je spájaná s jazykom JavaScript, napriek tomu to nie je jediný jazyk implementujúci AJAXovské technológie. Názov kľúčového objektu XMLHttpRequest neprikazuje pracovať iba s XML ale umožňuje manipulovať aj s alternatívnymi formátmi dát ako JSON, ale taktiež HTML alebo čistý text.

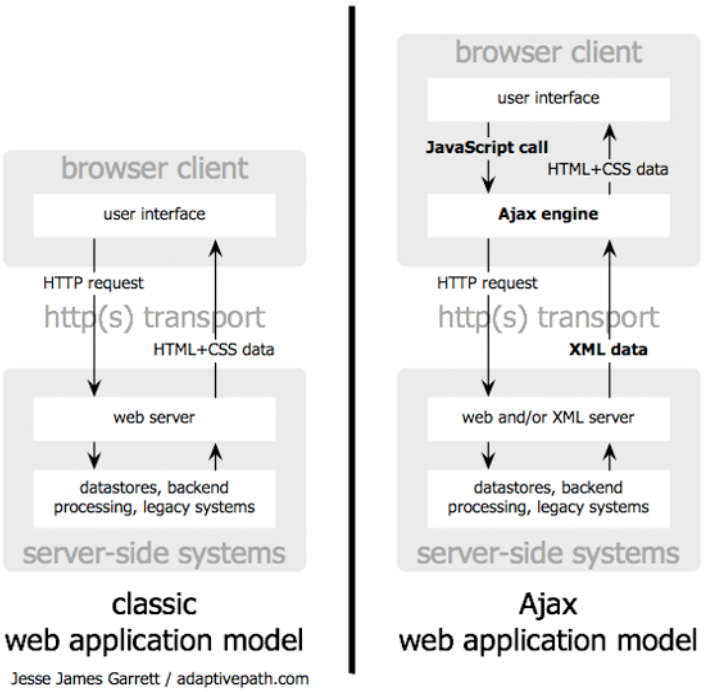
9 [1 – 273 s.]

Technológie spolupracujúce pri práci s AJAXom:

- HTML alebo XHTML a CSS na prezentáciu dát
- Document Object Model na dynamické zobrazenie dát a interakciu s nimi
- XML,XSLT, JSON na výmenu, manipuláciu a zobrazenie dát
- XMLHttpRequest objekt na vykonávanie asynchrónnej komunikácie
- JavaScript spája tieto technológie dohromady, čím sa vytvárajú bohaté možnosti pre tvorbu kvalitného webového produktu.

Koncept AJAXu tu je už pár rokov, avšak popularity sa dočkala až keď ju postupne začali podporovať viaceré vyhľadávače.

Obrázok 2.3: **Princíp fungovania technológie Ajax**[10]



Obrázok znázorňuje rozdiel medzi klasickým webovým modelom a modelom na báze AJAXu. Na ľavej strane je zobrazený model na princípe žiadosť/odpoveď, kde činnosť užívateľa v rozhraní

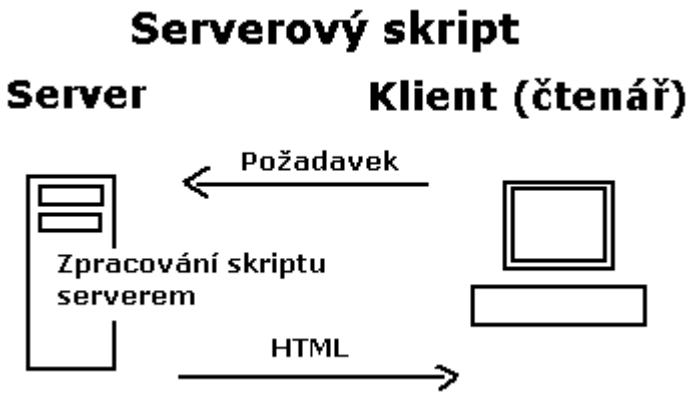
10 http://coronet.iicm.tugraz.at/lectures/mmis/material/slides_ajax.html

prehliadača pošle serveru žiadosť o dáta. Server žiadosť vyhodnotí a pošle požadované dáta, zvyčajne webovú stránku. Prehliadač tým pádom stránku načíta, čím sa úplne zmení obsah okna prehliadača.

Na pravej strane je v modeli pridaný AJAX engine, akýsi prostredník medzi užívateľským prostredím a serverom. Klient zavolá JavaScriptový skript, ktorý spustí AJAXové volanie. Toto volanie obslúži AJAX engine, prijme dáta zo serveru a JavaScriptom ich ďalej spracúva. Prijímané dáta sú zvyčajne vo formáte XML alebo JSON. Tieto sa podsúvajú klientovi ako parametre nejakej funkcie na základe ktorej sa vytvárajú sémantické HTML prvky.

2.4 Serverové technológie

Technológie na strane servera umožňujú vytvárať plnohodnotné aplikácie, spoluprácu s formulármi, databázami a má mnoho iných využití. Nevýhodami týchto technológií je to, že sú to väčšinou komerčné technológie, hosting na serveri býva drahší než pre statické stránky (ak je zadarmo, býva nekvalitný), skripty môžu zaťažovať server, obvykle sa musia ladiť online. Princíp týchto technológií spočíva v tom, že pri žiadosti server vykoná program a klientovi posiela len dáta vo forme HTML. Klient s programom neprichádza do styku. Pre názornosť uvádzam nasledujúci obrázok:



Obrázok 2.4: Schéma princípu serverového skriptu.[11]

Všetky serverové skriptovacie technológie si vyžadujú nainštalovanú podporu serveru, práva na spustenie programov alebo skriptov a všeobecnú priazeň správcu serveru. Preto som vybral zaužívané a navjviac rozšírené skriptovacie technológie na serverovej strane, a to PHP a MySQL.

PHP (HyperText Preprocessor) je freeware, tzn. že je voľne šíriteľný software. Je to jazyk nezávislý na platforme (ale na serveroch Apache ponúka viacero možností).

11 http://www.jakpsatweb.cz/php/jak-zacit.html

Príkazy sa zadávajú do HTML medzi tagy <?php a ?> a výsledkom je v HTML kóde. Stránka sa ukladá väčšinou so špeciálnou príponou (*.php). Táto skriptovacia technológia vznika v roku 1994 (Rasmus Lerdorf) a v súčasnosti je k dispozícii verzia PHP 5, pracuje sa na verzii 6.

PHP používa veľké množstvo vývojárov na miliónoch serverov. Vývojový tím PHP zahŕňa desiatky vývojárov a ďalšie desiatky ľudí pracujú na projektoch spojených s PHP (napr. PEAR, dokumentačný projekt, atď.).

K tomu aby bolo možné vidieť výsledok skriptu napísaného v jazyku PHP je nutné ho predať interpretovi k spracovaniu. K tomu však už nestačí mať len webový prehliadač.

Je potrebné mať:

- Textový editor (najlepšie so zvýrazňovaním syntaxe)
- webový prehliadač (najlepšie niekoľko a rôzne verzie, na prezeranie vygenerovaného kódu)
- www server (napr. Apache, prípadne sa dá využiť vzdialený server k čomu je nutné mať pripojenie k internetu a FTP klient)
- nainštalovanú podporu skriptovacieho stroja PHP (najlepšie modul Apache, príp. iným spôsobom)
- prístup k databázovému serveru (najčastejšie MySQL) ak pracujeme s databázou

Jazyk PHP je podobný jazyku C či Perlu. Od verzie 5 sa jedná o objektovo orientovaný programovací jazyk. Jednotlivé príkazy sa oddeľujú bodkočiarkou. PHP rozlišuje veľkosť písmen v názvoch premenných, ale nie u funkcií. „Program“ v jazyku PHP sa vkladá priamo do HTML stránok, ktoré mávajú koncovku .php ale je možné nastaviť aj inú. V prípade, že prehliadač požiada server o dokument, v ktorom je PHP kód, server najskôr predá skript interpretovi PHP (parser) čo je program, ktorý beží na serveri a stará sa o prevedenie PHP kódu. Interpret vygeneruje kód, text (výstup skriptu), ktorý je serverom odoslaný ako odpoveď danému prehliadaču. Z vyššie uvedeného je preto zrejmé, že návštevníkovi stránok sa PHP kód nezobrazí, uvidí len výsledok (na rozdiel technológii na strane klienta – napr. JavaScript).

MySQL je otvorený viacvlákninový užívateľský SQL relačný databázový server. MySQL je populárny databázový systém, podporuje viacero platforiem (Linux, Windows,Solaris) a je implementovaný vo viacerých programovacích jazykoch ako PHP, C++, či Perl. Databázový systém je relačný typu DBMS (database management system). [12]

12 [5]

Každá databáza je tvorená z tabuliek (môže byť aj z jednej). Riadky tabuliek predstavujú jednotlivé záznamy a stĺpce dátový typ týchto záznamov, spolu vytvárajú pole. Práca s MySQL databázou sa vykonáva pomocou „dotazov“, ktoré zadáva programovací jazyk SQL (Structured Query Language).

MySQL databázy sú prístupné vo všetkých programovacích jazykoch a oficiálne ich knižnice sú implementované v ANSI C/ANSI C++. MySQL sa najčastejšie využíva vo webových aplikáciách. Jeho obľúbenosť je úzko spojená s popularitou spomínaného PHP. PHP a MySQL sú základné komponenty pre tvorbu systémov (CMS), napríklad Ebay, wikipédia, atď.

Všetky informácie potrebné k používaniu MySQL sa dajú nájsť na stránke SQL[9] pod sekciou dokumenty. Na administráciu databázy sa používa príkazový riadok, prípadne sa dá použiť GUI administrátorské nástroje (MySQL Administrators a MySQL Query Browser) zo stránky MySQL. Obidva tieto nástroje sú obsiahnuté v jednom balíčku pod názvom MySQL GUI Tools.

Voľne šíriteľné administrátorské rozhranie realizované v PHP používané pre tvorbu databáz v MySQL je phpMyAdmin.

2.5 Formáty dát

2.5.1 JSON

JSON (JavaScript Object Notation) je jednoduchý formát dát, ktorý tiež umožňuje jednoduchú konverziu na iný formát. Je zrozumiteľný a jednoducho sa píše i číta. Je podmnožinou JavaScriptu, založený na štandarde ECMA 262 v decembri 1999. JSON je textový formát, ktorý je úplne nezávislý na jazyku, avšak používa konvencie z jazyku C. Všetky tieto vlastnosti robia JSON ideálnym jazykom na výmenu dát.

JSON je postavený na dvoch štruktúrach [8]

- Páry názov/hodnota. V rôznych jazykoch sa tento pár realizuje ako záznam, štruktúra, hešovacia tabuľka, unikátny kľúč zoznamu, či asociatívne pole.
- Zoradený zoznam hodnôt. Vo väčšine jazykoch sa realizuje ako pole, vektor, zoznam alebo sekvencia.

Jedná sa o univerzálne dátové štruktúry, podporované prakticky všetkými modernými programovacími jazykmi, ktoré ich podporujú v rôznych podobách. Je zrejmé, že i jazyky, ktoré podporujú tento formát musia byť založené na príbuzných štruktúrach.

Výpis 2.1 zobrazuje štruktúru syntaxe jazyka JSON.

Výpis 2.1: **príklad kódu vo formáte JSON**

```
1. {
2.       "meno": "Peter",
3.       "priezvisko": "Sveter",
4.       "vek": 33,
5.       "adresa": {
6.           "ulica": "Hollého 1203",
7.           "mesto": "Senica",
8.           "PSČ": "90501"
9.       }
10.        }
```

2.5.2 XML

Ďalším formátom dát používaných pri tvorbe webových stránok je XML.

XML, zo skratky eXtensible Markup Language, v preklade rozšíriteľný značkovací jazyk, umožňuje vytváranie rôznych značkovacích jazykov na základe rôznych požiadaviek. Obsahuje štrukturované dáta – tabuľky, adresáre, konfigurácie, atď., je to súbor pravidiel tvorby textových formátov, ktoré umožňujú štrukturovať dáta. XML uľahčuje počítaču tvoriť, čítať či zapisovať dáta a tým zaistiť jednoznačnosť štruktúry dát. XML je nezávislé na platforme, rozšíriteľné a podporuje lokalizáciu. XML plne vyhovuje štandardu UNICODE. XML začalo vznikať v roku 1996 a od roku 1998 ho doporučuje a štandardizuje W3C.

Jazyk XML je jednoduchý na učenie porozumenie, príklad syntaxe ukazuje obrázok:

Výpis 2.2: **Príklad kódu v XML**

```
1.    <Osoba>
2.       <meno>Pavel</meno>
3.       <priezvisko>Dobrý</priezvisko>
4.       <vek>35</vek>
5.       <addresa>
6.          <ulica>Kounicova 34</ulica>
7.          <mesto>Brno</mesto>
8.          <PSČ>90532</PSČ>
9.       </addresa>
10.      <telefon type="home">775 3453621</telefon>
11.      <telefon type="fax">608 0988764</telefon>
12.   </Osoba>
```

XML dovoľuje definovať nový formát kombináciou a opätovným použitím iných formátov. Pri kombinácii formátov je potrebné dávať pozor na rovnaké názvy elementov alebo atribútov, aby sa predišlo tejto zámene. XML obsahuje mechanizmus menných priestorov (XSL a RDF).
XML formát podporuje popis zdrojov a uplatnenie metadát, RDF napríklad umožňuje označiť ľudí vo webovom fotoalbume.[13]

2.6 Frameworky

2.6.1 Dojo toolkit

Dojo toolkit[6] je multiplatformná knižnica jazyka Javascript založená na open-source princípe, čiže jej zdrojový kód je verejný, má otvorený obsah. Knižnica dojo je vyvinutá na uľahčenie a urýchlenie vytvárania moderných interaktívnych webových prvkov. Multiplatformnosť si vydobuje takmer rovnakou funkčnosťou pre odlišné platformy v rôznych prehliadačoch s podporou technológií ako JavaScript a AJAX . Skúsenosti ale ukazujú odlišnú funkčnosť u rôznych prehliadačov, u niektorých úplnú nefunkčnosť.

13 Viz.: http://www.w3.org/2004/08/XML-in-10-points-sk.html

2.6.1.1 Widgety

Knižnica Dojo sa skladá z prvkov zvaných Widgets. Widgety sú komponenty zahrnujúce javascriptový kód, HTML značky a CSS rozloženie, čo zabezpečuje multiplatformnosť a prenositeľnosť. Príkladmi dojo widgetov sú napríklad grafy, menu, tabuľky, validačné formuláre, grafické efekty a veľa ďalších interaktívnych prvkov. Dojo obsahuje aj pár prednastavených css štýlov na zmenu vzhľadu jednotlivých prvkov.

2.6.2 ExtJS

Ext JS umožňuje vytvárať perfektné multiplatformové webové aplikácii pomerne rýchlo a jednoducho. Komponentový model Ext JS vytvára prehľadne štruktúrovaný kód, takže sa v ňom ľahko udržujú i väčšie aplikácie. Ext JS poskytuje encyklopedickú zbierku widgetov užívateľského rozhrania. Z tohto dôvodov je medzi vývojármi na celom svete veľmi obľúbený[7].

3 Popis existujících rešení

3.1 Popis komponenty

Jednou z najpoužívanejších spôsobov zobrazenia dát je formou dátovej mriežky. Mriežky majú široké spektrum možností a využitia. Či už je to využitie čisto na čítanie, ako v prípade adresára alebo telefónneho zoznamu, alebo ide o rozsiahlejšie a zložitejšie využitie s možnosťou spracovávať dáta a bližšie s nimi pracovať ako v prípade administrácie objednávok v obchode alebo inventári. Firmy využívajúce takéto mriežky sú nútené uchovávať dáta na jednom mieste a z jedného miesta ich spravovať. To síce nesie svoje výhody avšak s trendom internetu je snaha namierená iným smerom, a to ku internetovým a intranetovým aplikáciám aby bolo možné k dátam pristupovať a administrovať ich všade na svete. Oproti desktopovým aplikáciám toto so sebou nesie isté obmedzenia a nevýhody. Kvôli rôznym požiadavkám na mriežku bývajú webové riešenia časovo zdĺhavé, závislé na znovunačítaní stránky a náročnejšie na implemetáciu čo znižuje ich použiteľnosť.

Internet ponúka mnoho známych implementácií dátovej mriežky v rôznych prostrediach. Napríklad v ASP.NET je dátová vrstva implementovaná pomocou pripojenia xml súboru a jeho zobrazenie v HTML tabuľke. Takáto implementácia je málo interaktívna lebo sa musí pri každej operácii nanovo načítavať obsah stránky. Pri technológii AJAX sa znovu-načítaniu vyhneme,

respektíve ho nebudeme vnímať lebo všetko pracuje transparentne a neznepríjemňuje ďalšiu prácu. Iné implementácie exportujú mysql databázu pomocou požiadavku SELECT do xml súborov zrozumiteľných pre ďalšie čítanie. Technológie spojené s javascriptom používaju v zásade datové sklady ktoré zabezpečia poriadok a transparentnosť dát na klientskej strane.

3.2 Vytvorenie dátovej mriežky knižnici DOJO

3.2.1 Dátová vrstva - Získavanie dát

Webová komponenta dátovej mriežky potrebuje zdroj dát pre klientskú stranu. Problém nastáva ak získava dáta z rozličných nehomogénnych zdrojov. Takými môžu byť zdroje vo formátoch JSON, XML, CSV, HTML. Tieto dáta potrebujú nejaký adaptér, ktorý z nich spraví homogénne dáta, alebo to aspoň tak bude vyzerať pre potreby užívateľa.

Úlohou týchto dát je, aby boli zrozumiteľné pre prvky užívateľského rozhrania a aby táto réžia bola úplne transparentná. Problém heterogénnosti zdrojov rieši v knižnici dojo dátový sklad. Ten umožňuje odtieniť dátovú a komunikačnú vrstvu. Každý zdroj má definovaný svoj dátový sklad. Ten transformuje dáta zo zdroja do jednotného formátu. Ak pribudne nejaký dátový zdroj, stačí mu len implementovať nový dátový sklad a dáta sú pripravené na spracovanie. Prostredníctvom technológie AJAX získame dáta zo všetkých zdrojov a potom pre každý typ vytvoríme dátový sklad s homogénnymi dátami. Pomocou skladu môžeme dáta spracúvať, vytvárať modifikovať, rušiť, radiť a vyhľadávať v nich. Tieto možnosti nám sprostredkúvajú rozhrania knižnice DOJO akými sú Read API ktoré je určené na čítanie, Write API určené na širšiu manipuláciu s dátami ako je pridávanie, rušenie a modifikácia záznamov, Identify API ktoré pomáha pri vyhľadávaní položiek a Notification API ktoré dovoľuje reagovať na udalosti vznikajúce pri manipulácii s lokálnymi dátami a ich zmenou. Pre každý typ operácie bude vhodné iné rozhranie na implementáciu.

3.2.2 Vytvorenie samotnej mriežky

Dátová mriežka môže byť definovaná deklaratívne pomocou HTML značiek a programovateľne pomocou Javascriptu. Pomocou značkovania v HTML vyzerá definícia podobne ako na výpise 3.1.

Výpis 3.1: **Vytvorenie gridu v Dojo pomocou deklarácie v HTML**

```
1. <table jsId="grid" dojoType="dojox.grid.DataGrid" query="{login:
   '*'}" store="store">
2.       <thead>
3.         <tr>
4.           <th field="login" width="300px" >login</th>
5.           <th field="name" width="200px" editable="true">name</th>
6.           <th field="int" width="200px" editable="true">cislo</th>
7.           <th field="no" width="200px" editable="true">poradie</th>
8.         </tr>
9.       </thead>
10. </table>
```

Hlavnými, deklaratívnymi značkami sú <table> a <th>. Značka <table> spolu s parametrom *dojoType* definuje, že komponenta DataGrid bola vytvorená. Značka <th> definuje rozloženie jednotlivých stĺpcov mriežky a je nevyhnutná pre jej vytvorenie.

Ďalšiu skupinu tvoria voliteľné atribúty platné zvlášť pre jednotlivé stĺpce:

- *field* – definuje názov stĺpca mriežky
- *width* – definuje šírku stĺpca
- *cellType* – udáva typ bunky, je možnosť si vybrať z boolovskej varianty definovanej v `dojox.grid.cells.Bool`, kde bunka bude nadobúdať iba hodnoty true a false. Ďalej možnosť selektačnej bunky (`dojox.grid.cells.Select`), kde si užívateľ vyberá hodnotu bunky z viacerých variant. Východiskovým formátom je textový vstup priamo v bunke.
- *options* – používa sa iba s parametrom *cellType* a definuje mu obor hodnôt ktoré môže bunka nadobudnúť.
- *editable* – možnosť vyžadujúca prístup k dátam pomocou skladu `dojo.data.ItemFileWriteStore`. Tento parameter nadobúda iba hodnoty true alebo false, povoľuje/zakazuje editovanie vybranej bunky.
- *draggable* – parameter zakazujúci premiestňovanie vybranej bunky

Definovali sme si nastavenia pre stĺpce, avšak je tu ešte jedna skupina parametrov pre mriežku samotnú:

- jsId – identifikuje meno javascriptovej premennej, ktorá odkazuje na inštanciu triedy *DataGrid.*
- *Store* – parameter určujúci sklad, z ktorého bude mriežka sťahovať dáta pre manipuláciu.
- *ColumnReordering* – vlastnosť definujúca nastavenie dynamického znovu zoradenia stĺpca.

3.3 Zhodnotenie frameworkov

Implementácia mriežky v oboch frameworkoch bola obdobná, principiálne sa jednotlivé prístupy vo veľa veciach nelíšia. Rýchlejšie fungovanie zaznamenal framework Dojo, avšak zase zaostával vo funkčnosti v ostatných prehliadačoch.

4 Návrh komponenty dátovej mriežky

Dátová mriežka je štandardnou komponentou implementovateľnou v mnohých programovacích jazykoch (Java, ASP.NET, PHP). Predstavuje klientské rozhranie, ktoré sprostredkúva užívateľovi veľký objem dát a umožňuje následnú manipuláciu s nimi. Ponúka náhľad a orientáciu medzí dátami. Je reprezentovaná ako tabuľka, v ktorej riadky reprezentujú jednotlivé záznamy databázy a stĺpce identifikujú ich jednotlivé atribúty.

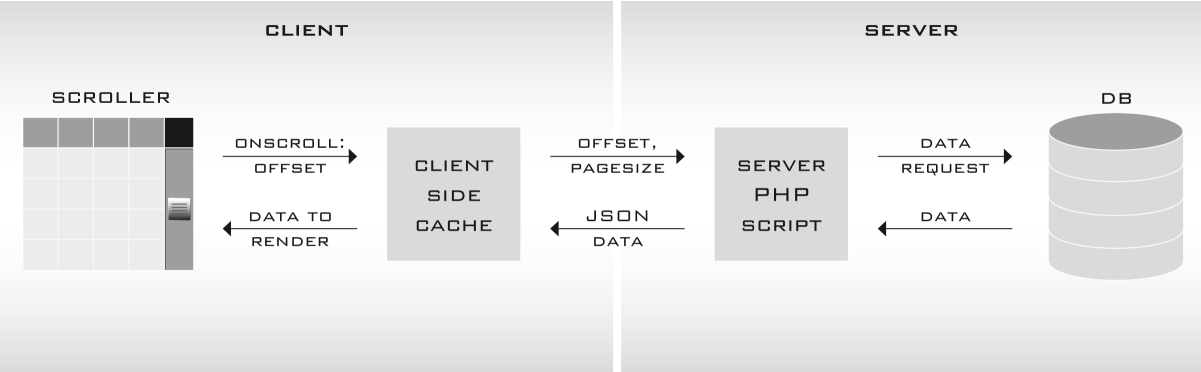
Cieľom tejto bakalárskej práce je navrhnúť vlastnú komponentu dátovej mriežky podľa zaužívaných postupov a existujúcich riešení. V prvom rade je treba rozdeliť komponentu na časti pričom každá má inú funkcionalitu a jej réžia sa odlišuje aj technológiami ktorými je implementovaná.

Prvú časť reprezentuje užívateľské rozhranie (GUI), ktoré sa stará o vzhľad komponenty, vykresľovanie dát, užívateľskú prívetivosť (user-friendly) a o celkovú funkcionalitu na strane klienta. Je reprezentovaná HTML elementami s definíciou vzhľadu v CSS. O vykresľovanie záznamov sa stará JavaScript.

Ďalšou časťou sú dáta a ich réžia, ktorá sa odohráva na pozadí komponenty. Táto časť je realizovaná na klientovi prostredníctvom lokálnej pamäte (cache), kde sa ukladajú načítané položky od serveru; a na serveri, ktorý posiela záznamy z databázy podľa požiadaviek.

Poslednú nosnú časť komponenty reprezentuje vrstva udalostí, ktoré ovládajú správanie ostatných zložiek komponenty.

Obrázok 4.1: **Princíp komunikácie webovej dátovej mriežky.**



Obrázok 4.1 popisuje spôsob komunikácie v rámci komponenty, rozdeľuje na klientskú a serverovú stranu. Vyplýva z neho, že celá funkčnosť sa začína pri vykreslení užívateľského rozhrania. Rozhranie ponúka možnosť navigácie, ktorá je v tejto práci implementovaná pomocou rolovacieho mechanizmu (scrollbar), ktorým sa pohybuje vo vertikálnom smere. Scrollbar detekuje svoju pozíciu vzhľadom na veľkosť elementu v ktorom je vytvorený. Po zmene tejto pozície sa vyvolá udalosť *onscroll*, kedy sa pošle pozícia scrollbaru, takzvaný offset, klientskej pamäti. Pamäť zistí, či obsahuje položky na danej pozícii. Ak obsahuje, vráti ich užívateľskému rozhraniu na vykreslenie. V opačnom prípade pošle požiadavku serveru s parametrami o offsete a počte požadovaných položiek (pagesize). Serverový skript stiahne zo serveru požadovaný zhluk dát, prevedie dáta na formát JSON a vráti ich klientskej pamäti. Tu sa uložia a pošlú rozhraniu na vykreslenie.

4.1 Služby komponenty

Funkčná komponenta dátovej mriežky sa snaží byť pre užívateľa 'niečím viac' ako len statickou tabuľkou ako ju poznáme z HTML. Mala by interaktívnosťou reflektovať všeobecne známe požiadavky na manipulácii s dátami, akými sú:

- radenie(abecedne, podľa veľkosti čísel),
- filtrovanie obsahu, zmena šírky stĺpca,
- vyhľadávanie v reálnom čase,
- editovanie dát

4.1.1 Sorting

Radenie (sorting) je proces, pri ktorom sa transformuje zoznam položiek na zoradený zoznam v stanovenom poradí. Najčastejšie je numerické radenie podľa hodnoty výrazu a lexikografické radenie podľa abecedy, resp. umiestnenia v ASCII tabuľke. Radenie musí byť prispôsobené formátu dát jednotlivých atribútov v dátovej mriežke, či už to budú číselné hodnoty, text alebo špeciálne formáty ako napríklad dátum. V prípade dátumu pre väčšinu štandardov jeho zobrazovania platí lexikografické radenie, lebo časové údaje sú usporiadané hierarchicky od najväčšej časovej veličiny po najmenšiu. Nie vždy je to ale tak a treba určiť štruktúru časového záznamu a zadefinovať smerodatné veličiny.

Pre radenie som vybral klasickú metódu vyvolania radiaceho mechanizmu, a to po kliknutí na hlavičku samotnej mriežky. Pri ďalšom kliknutí na hlavičku totožného stĺpca sa vykoná radenie položiek v danom stĺpci v opačnom poradí (stúpajúco /klesajúco).

Obrázok 4.2 znázorňuje mechanizmus viacnásobného radiaceho mechanizmu, ktorý zoraďuje položky podľa viacerých kritérií podľa ich radiacej priority. V mojom návrhu som zvolil jednoduchšie riešenie, a to radenie iba podľa jediného kritéria.

Obrázok 4.2: **Príklad viacnásobného radiaceho mechanizmu**

Id (Sort Order: 1)	FirstName (Sort Order: 0)	MiddleName	LastName (Sort Order: 2)
15	Abraham		Lincoln
6	Andrew		Jackson
16	Andrew		Johnson
22	Benjamin		Harrison
40	Bill		Clinton
28	Calvin		Coolidge
20	Chester	A.	Arthur
32	Dwight	D.	Eisenhower

4.1.2 Editing

Medzi hlavné ovládacie prvky mriežky patrí možnosť editovania jednotlivých buniek. Editácia (Editing) umožní užívateľovi zmeniť alebo upraviť hodnotu atribútu vybraného prvku. Možnosť editovania môže byť pre každý stĺpec rozdielna, treba túto vlastnosť zadefinovať metadátami pri vytváraní komponenty. Zakázanie tejto možnosti treba deklarovať v atribútoch modelu mriežky nastavením atribútu editable na logickú hodnotu false. Táto možnosť bez deklarovania je implicitne nastavená na true, čiže všetky stĺpce majú povolené sa editovať.

Obrázok 4.3 zobrazuje vytvorenie jednoduchého prednastaveného textového poľa

Obrázok 4.3**: Editačné pole**

id	Meno	Priezvisko	Dátum narodenia	Pohlavie
1	Peter	Janík	123456	M
2	Karol	sdfs	123456	M
3	Michal	Varga	123456	M
4	Jozef	Balák	123456	M

Výber editovaného prvku sa uskutoční prostredníctvom dvojkliku na cieľovú bunku. Po tejto udalosti sa objaví editovateľné okno na zadávanie nových hodnôt. Typ zadávacieho formuláru závisí na dátovom type zobrazovanom v danom stĺpci. Keď ten nie je nastavený, zobrazí sa textové zadávacie pole. Vstupné rozhranie nemusí byť všade len to isté textové pole. Implementované môžu byť klasické ovládacie prvky, známe aj z HTML, ako napríklad combo box, radio button, select. Každý z týchto prvkov ponúka určité editačné rozhranie, avšak občas mriežka uchováva zložitejšie dátové typy, ktoré majú špecifickú štruktúru. Takýto typ dát treba dodatočne spracovávať na zrozumiteľnejšie dielčie časti. Príkladom je napríklad dátum, ktorý je prirodzene chápaný ako reťazec čísel a pomocných znakov. Avšak keď má program definované metódy na rozdeľovanie daného reťazca podľa určitého, dohodnutého formátu dátumu, ľahko zistí jednotlivé časové premenné. Napríklad určíme si formát času *2010-05-23*. Vieme, že prvý údaj je rok, druhý mesiac a tretí údaj reprezentuje deň. Podľa toho text rozdelíme do troch celo-číselných premenných s ktorými môžeme ďalej efektívne pracovať a využívať ich. Môžu nám byť nápomocné napríklad pri hierarchizácii, respektíve pri radení podľa tohto dátumu. Moderné webové technológie však umožňujú vytvárať efektívnejšie vstupné prvky či už pre základné dátové typy, tak aj pre heterogénne záznamy zložitejších dátových typov. Takéto dátové typy sa definujú parameterom columnEditor, ktorý nesie sebou informáciu o type vstupného rozhrania. Prvky umožňujúce editovať zložitejšie dátové typy sa nazývajú widgety a ako príklad uvádzam kalendár použitý pri dátach s dátovým typom dátum, v ktorom si užívateľ prehľadne vyberie presný dátum pomocou jednoduchého kalendárového rozhrania.

Obrázok 4.4 znázorňuje pokročilé editačné formuláre.

Obrázok 4.4: **Editačné možnosti widgetov** [14]

Jediným stĺpcom, ktorý nemôže byť editovateľný, je stĺpec reprezentujúci identifikátory jednotlivých záznamov v sklade. Označuje sa parametrom *"identifier"* na začiatku JSON súboru s dátami pre mriežku. Hodnoty záznamov v identifikačnom stĺpci musia byť pre každý záznam jedinečné a rozdielne.

V prípade, ak sa snažíme editovať bunku, ktorá nemá vlastnosť editácie deklarovanú, bunka žiadnym spôsobom nebude reagovať na danú udalosť a bude sa chovať staticky. Pri editácii sa v bunke mriežky zobrazí textový formulár s aktuálnou hodnotou označenej bunky pripravený na prepísanie novou hodnotou.

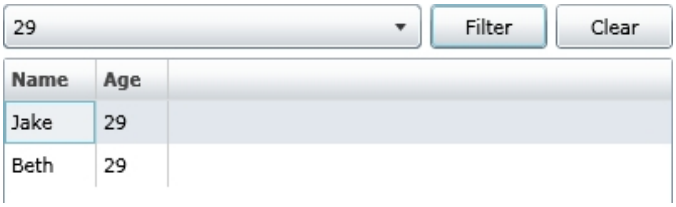
4.1.3 Filter

Umožňuje filtrovanie obsahu podľa zadaného reťazca. Zobrazia sa iba záznamy, ktoré obsahujú text zadaný užívateľom. Pod každým stĺpcom je filtrovací formulár do ktorého užívateľ vkladá text, podľa ktorého sa na serveri vyfiltrujú vyhovujúce záznamy a ktoré klient obdrží a vykreslí do užívateľského rozhrania.

Variantu filtra znázorňuje obrázok 3.3, v mojom riešení som využil priamy filter pomocou zadávaného reťazca v textovom poli..

14 http://wiki.forum.nokia.com/index.php/File:Dojo_input.jpg

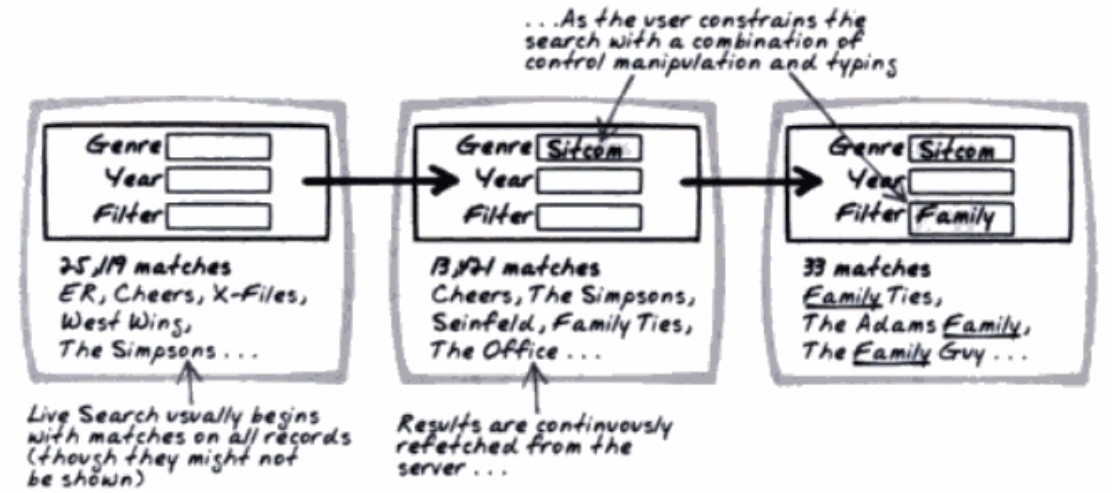
Obrázok 4.5: **Selektačný filter**[15]



4.1.4 Live search

V reálnom čase filtruje prvky z tabuľky a vyberá len tie ktoré maju spoločné znaky ako sú práve zadané. Namiesto toho, aby klient posielal požiadavku na dáta pri stlačení každej klávesy, posiela dáta po určitých intervaloch. Má to väčší význam pre presnosť a relevantnosť hľadanej položky. Live Search nielenže pomáha užívateľovi hľadať určité atribúty ale aj pomáha zrýchliť vyhľadávanie vybraním ponúknutých výsledkov pomocou klávesnice. Ponuka výsledkov neobsahuje iba reťazce začínajúce hľadaným reťazcom ale všetky ktorých je hľadaný reťazec podmnožinou.

Obrázok 4.6: **Mechanizmus Live Search [5]**



4.1.5 Column-Sizing

Zmena šírky stĺpca (Column-Sizing) umožňuje roztiahnuť šírku stĺpca pomocou kurzora myši. Implicitná hodnota šírky stĺpca zodpovedá hodnote parametra *width* uvedenej v metadátach modelu pri danom stĺpci.

15 http://www.mostlydevelopers.com/mostlydevelopers/blog/image.axd?picture=2009/2/collection_filter.jpg

4.2 Dátová vrstva

Práca dátovej mriežky spočíva v spracovaní dát. Avšak pri prechádzaní veľkého množstva dát a ich následnom vykreslení nastáva problém. Tento problém spočíva v neustálych požiadavkách na server. Pre jednoduchšie rozhranie a minimalizovanie sieťových nárokov je použitá implementácia „klienskej pamäte“ (client-side cache). Tá poskytuje dáta v homogénnom stave. Homogénne dáta do tejto pamäte už vstupujú, sú dekódované hneď po prijatí od serveru PHP skriptom a posielané klientskej strane v zrozumiteľnom formáte.

Pre potreby dátovej mriežky som zvolil formát JSON v ktorom bude prijímať komponenta dáta na svoje vykreslenie. Takisto XML môže byť variantou dátového zdroja pre klientskú pamäť, avšak v mojej práci budem používať iba formát JSON.

4.3 Grafické užívateľské rozhranie

Pri návrhu GUI komponenty dátovej mriežky bola pre mňa najdôležitejšia najmä jednoduchosť a intuitívnosť obsluhy. Ide o to, aby každý užívateľ prišiel ľahko na to, čo má spraviť aby sa vykonala požadovaná operácia. Vzhľad mriežky je podobný klasickej tabuľke z jazyka HTML a sémanticky to môže byť aj tabuľka, alebo iné, napríklad číslovaný zoznam naštýlovaný do podoby tabuľky s bunkami.

Rozhranie mriežky je uložené v triede risogrid. V HTML je deklarovaný element, ktorý sa lokalizuje pomocou špecifického id v DOM. V danom uzle sa vyhľadá element kontajneru, v ktorom sa vykreslí celá komponenta mriežky s priradenými vlastnosťami.

Na vytvorenie novej mriežky slúži objekt *risoGrid*, v ktorom sa deklarujú následovné vlastnosti:

- rowCount: určuje počet zobrazených riadkov
- width/height: šírka/výška vytváranej komponenty
- header: nadpis uvedený v záhlaví mriežky
- url: nesie v sebe adresu serveru, na ktorý sa majú posielat požiadavky na dáta
- resizeColumn: príznak editovateľnosti stĺpca,
- sortBy: smer v akom sa majú radiť položky,
- editable: moťnosť editovať položky,
- url: adresa serverového skriptu,
- page: počet záznamov, ktoré sa naraz prijímajú od serveru,
- offset: poradové číslo od ktorého sa má načítať zhluk dát veľkosti *page*,

- visibleRows: počet zobrazených riadkov tabuľky,
- totalRows: určuje počet všetkých záznamov v databáze,
- columnModel: definuje názov modelu komponenty,

O vlastnosti komponenty sa stará trieda model, ktorá sa definuje pri vytvorení samotnej komponenty a obsahuje metadáta potrebné k vytvoreniu mriežky. Zo zadefinovaných metadát určí, či je stĺpec editovateľný, akú má šírku, určí možnosť zväčšovania šírky stĺpca. Užívateľsky príťažlivá je implementácia rôznych typov widgetov, ktoré uľahčujú prácu ako vstupné rozhrania pri editácii prvkov.

Rozloženie prvkov a generovanie jednotlivých HTML elementov riadi vykreslovacia metóda v GUI. Tá pomocou nadobudnutých premenných a požiadaviek užívateľa vloží HTML kód s elementami do kontajneru určenému na vykreslovanie celej komponenty. Tieto sémantické elementy prislúchajú k určitým triedam, ktorých vizuálne vlastnosti sa definujú v externom CSS súbore *gridlayout.css*. Triedy elementov sa menia po určitých udalostiach a tým sa zmení aj ich vzhľad, len pre inú triedu a s inými vlastnosťami.

4.4 Udalosti

To, čo z tabuľky robí funkčnú dátovú mriežku sú udalosti (events). Ide hlavne o udalosti modelu DOM, ale aj iné, nadefinované udalosti ktoré nie sú súčasťou modelu DOM. Udalosti sa generujú pri vykonaní určitej operácie nad prvkom dokumentu.

Udalosti v návrhu tejto práce zaberajú podstatné miesto a dôležitosť. Nadefinované udalosti sú rozšírením klasických Javascriptových udalostí. Sú používané prepájaním jednotlivých udalostí s inými funkciami.

Udalosti sa deklarujú pri určitej funkcii ako proces, ktorý sa vykoná po ich vykonaní. Určujú prvky, s ktorými bola spojená vykonaná udalosť. V mriežke sa využívajú pri každej interakcii mriežky s kurzorom myši a klávesnicou. Ide o udalosti spojené s udalosťou DOM alebo ako indikátor začatia resp. skončenia nejakej funkcie. Sú vyvolané tiež ako zmena obsahu, vymazanie prvku, resp. každou vyvolanou činnosťou.

Opis jednotlivých udalostí ktoré môžu v mriežke nastať a nesú so sebou nejakú činnosť:

- click : kliknutie na hlavičku mriežky spustí radiaci mechanizmus
- double-click : dvojité kliknutie na bunku tabulky vyvolá editačný mechanizmus
- scroll : pohyb a navigáciu po prvkoch zabezpečuje scrollovanie

- mouseover : pri prechode myšou nad záznamom sa farba riadku záznamu premení
- mouseout : nastáva pri skončení mouseover udalosti, čiže keď sa kurzor nenachádza nad záznamom.

Ďalšími možnými udalosťami sú udalosti z rozhrania klávesnice ktoré môžu byť pre zrýchlenie práce doimplementované.

4.5 Vytvorenie komponenty

Moja komponenta dátovej mriežky sa vytvára deklarovaním jej parametrov v HTML kóde. Týmito parametrami sú metadáta charakterizujúce možnosti a vlastnosti komponenty. V týchto dátach sú deklarované vizuálne parametre ako napríklad počet zobrazených riadkov alebo príznak možnosti zmeny veľkosti stĺpcov. Ďalšími parametrami sú adresa vzdialeného skriptu a parametre pre serverové volania, názov elementu v HTML kóde, takzvaného kontajneru, v ktorom sa vyobrazí celá komponenta. Dôležitým parametrom je model, ktorý v sebe nesie informácie o vzhľade a nastaveniach jednotlivých stĺpcov. Tieto obsahujú rozmery stĺpcov, dátový typ položiek v stĺpci, možnosť editovania položiek stĺpca. V niektorých implementáciách sa stretneme aj s parametrom ako napríklad hidden, ktorý informuje či sa má vôbec daný stĺpec zobraziť. Model mriežky môže definovať aj zobrazenie zložitejších položiek, napríklad počet stĺpcov pod ktorými sa bunka vykreslí (colspan, rowspan), čo som ale v tejto práci neimplementoval. Skript podľa týchto parametrov vytvorí samotnú mriežku.

Zápis 4.0 zobrazuje zadefinovanie mriežky a jej vlastností pomocou metadát.

Zápis 4.0

```
datagrid = new risoGrid('kontajner', {
        columnModel: colModel,
        url:"server/db.php",
        resizeColumns: true,
        width:600,
        rowCount: 20,
        headerHeight:35
});
```

```
var colModel = [{
                        label: "ID",
                        dataType:'number',
                        width:120
                    },
                {
                        label: "Meno",
                        dataType:'string'
                    },
                {
                        label: "Priezvisko",
                        dataType:'string'
                          , width:130
                    }
                ];
```

4.6 Navigácia

Dátová mriežka je určená na načítavanie dát z databázy a pre ich spracovanie na strane klienta. Prehliadač sprostredkuje dáta komponenty v prehľadnej forme a s cieľom sprostredkovať užívateľovi komponentu, ktorej ovládanie bude intuitívne a uživateľsky priateľské.

Medzi primárne funkcie patrí možnosť navigácie, čiže prechádzanie medzi prvkami a ich prezeranie. Základnými navigačnými konceptami v zobrazovaní dvojrozmerných dát sú stránkovanie a rolovanie (scrolling).

Klasickým riešením je stránkovanie, čiže rozdelenie záznamov do jednotlivých podstránok. Pri tomto riešení užívateľ vidí len pár záznamov a musí prepínať na inú stránku aby videl ďalšie. Pre potreby dátovej mriežky je akceptovateľné aj takéto riešenie, avšak môže sa stať, že užívateľ pri veľkom počte strán stratí prehľad a celková navigácia naprieč záznamami stratí na prehľadnosti a stane sa pre užívateľa neefektívnou.

Tento problém nastáva ak databáza obsahuje veľké množstvo dát. Prijať tisícky záznamov naraz a následne ich vykresliť v prehliadači je náročné na sieťové požiadavky, tým pádom to má vplyv aj na rýchlosť behu celej komponenty. Aby sa dalo s komponentou plnohodnotne pracovať, musí odpovedať na serverové požiadavky čo najskôr, najlepšie hneď po vykonaní požiadavky.
Preto sa potreby dátovej mriežky sa javí praktickejšie vytvoriť takzvaný virtuálny pracovný priestor (Virtual Workspace)[5]. Ten poskytuje akýsi pohľad na strane klienta do serverovej časti, pričom umožňuje užívateľovi navigovať sa naprieč záznamami ako keby boli všetky lokálne. Vyvoláva ilúziu, že celý pracovný priestor je v prehliadači, avšak server posiela časti obsahu dynamicky na

požiadanie. Preto sa mi najpraktickejším riešením javí použitie scrollbaru, ktorý inkrementálne naviguje po obsahu a umožňuje "skákať" na vybranú pozíciu.

5 Implementácia

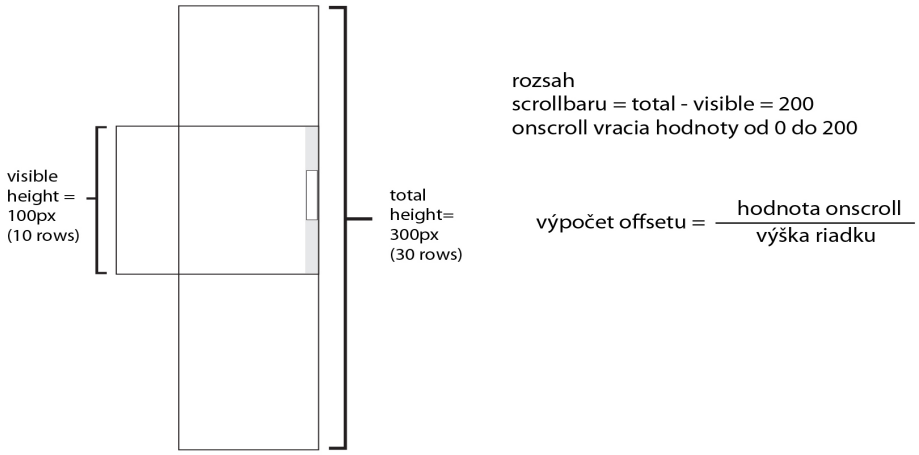
Implementácia mojej práce so sebou niesla zopár problémov vyplývajúcich či už zo slabých zdrojov k danej problematike alebo zo slabých skúseností s prácou s danými technológiami. Prácu som implementoval s využitím knižnice Mootools. Je v nej implementovaná celá škála pred-pripravených metód pre zjednodušenie implementácie a lepšie objektovo orientované programovanie .

5.1 Navigácia

Keďže náš virtuálny pohľad do databázy nevykresľuje naraz celý obsah databázy ale len zvolený počet riadkov, vytvorenie scrollbaru ako takého by nebolo príliš reálne zrealizovateľné. Z charakteru scrollbaru totiž vyplýva, že sa vytvorí keď sa vačší HTML element nachádza v menšom elemente. Tým pádom by musela naša komponenta naraz obsahovať všetky položky aby sa mohol scrollbar prispôsobiť svojimi parametrami výške vnútorného obsahu. Riešením využívaným v knižnici DOJO je vytvoriť pre nenačítané položky prázdnu tabuľku, ktorej obsah sa vypĺňa pri načítaní dát. Pri implementácii som zvolil prístup s možnosťou nastaviť si v parametroch komponenty počet zobrazených stĺpcov. Obsah riadkov sa bude vykreslovať do jednotlivých položiek a navigácia sa odohráva pomocou scrollbaru. Tento scrollbar však nie je vlastný elementu so zobrazenými položkami, pretože jeho výška je rovná výške počtu zvolených stĺpcov. Pre tieto účely sa vytvorí nový prázdny element *div* s šírkou 1pixel a výškou rovnou počtu zobrazených riadkov a do neho ďalší div s výškou všetkých riadkov v databáze. Týmto sa vytvorí scrollbar, ktorý je parametricky identický ako keby bola vykreslená celá databáza.

Posúvaním scrollbaru sa vyvolá udalosť *scroll,* ktorá detekuje pohyb po scrollbare. Podľa jeho pozície vypočíta odchýlku (offset), čo je index prvku z databázy, ktorý sa má vykresliť. Predtým než komponenta pošle serveru požiadavku na dáta, skontroluje lokálnu pamäť, či už táto položka nie je načítaná. Ak ju už máme v pamäti, ušetrí sa serverové volanie a položky sa vykreslia z lokálnej pamäte. Keď sa však v pamäti nenachádza, uskutoční sa serverové volanie a dáta kým sa vrátia na vykreslenie, uložia sa do pamäte.

Obrázok 5.1: **Zobrazenie scrollbaru a výpočet jeho pozície**



Na obrázku 5.1 je znázornený princíp, z akého vlastne vzniká scrollbar, a to vložením väčšieho HTML elementu do menšieho. Pre príklad je uvedený menší element obsahujúci 10 prvkov, čo zodpovedá 100 pixelom pri výške riadka 10 pixelov. Väčší zodpovedá tridsiatim riadkom. Z toho vyplýva, že trajektória scrollbaru sa pohybuje rozmedzí 0 až 200 px. Z pozície scrollbaru vypočítame, na ktorej pozícii sa nachádza vzhľadom ku všetkým dátam vo väčšom elemente tak, že ju vydelíme výškou riadku. Výška menšieho elementu zodpovedá počtu riadkov (visibleRows) a výška väčšieho zodpovedá výške všetkych záznamov v databáze.

5.2 Sorting

Proces radenia sa vykoná kliknutím na hlavičku mriežky s názvom stĺpca. Vytvorí sa udalosť, pomocou ktorej v javascripte voláme ajaxové požiadavky na PHP skript, ktorým vyberieme z databázy položky v požadovanom poradí. Pri radení požiadavka na server zmení o parameter sorting, ktorého hodnota nesie názov stĺpca, podľa ktorého sa majú položky radiť. Tento reťazec so sebou nesie aj príznak smeru radenia. Štandardnou hodnotou je stúpajúci smer (ascending), príznak smeru radenia sa posiela až pri druhom kliknutí na hlavičku. Príznakom bude reťazec pred hodnotu parametra. V jazyku PHP sa detekuje výskyt tohto reťazca a podľa toho sa položky radia.

5.3 Editácia

Editácia jednotlivých atribútov prebieha po zmenení hodnoty atribútu vo vstupnom formulári. Formulár reaguje na udalosť onEdit, pri ktorej hneď ajaxovým volaním pošle na server hodnotu

zmeneného atribútu a jednoznačný identifikátor položky. Na serveri sa pomocou skriptu uloží položka do databázy.

5.4 Dátová vrstva

Dátová vrstva predstavuje zdroj dát, ktoré sú komponentou spracovávané a interpretované pre užívateľa. V prípade tejto práce sa jedná o dáta získavané asynchrónne, technológiou AJAX. Serverový skript je nastavený na posielanie zhlukov dát vo formáte JSON. Ďalšou alternatívou by bol formát XML, ale vybral som si formát JSON pre natívnu javaScriptovskú syntax. Dáta sú, ako som spomínal, získavané asynchrónne pomocou požiadaviek pospájaných z užívateľskej interakcie na klientskej strane. Sú to atribúty objektu mriežky, ktoré sa menia podľa vyvolaných udalostí. Parametre asynchrónneho požiadavku sa rozširujú o atribút sortBy, ktorý v sebe drží informáciu o tom, v akom smere je vybraný stĺpec zoradený, parameter sort určujúci podľa akého stĺpca je mriežka zoradená. Tieto radiace udalosti vyvolá kliknutie na hlavičku mriežky.

Ďalším parametrom na serverové volanie je filter, ktorý v sebe nesie reťazec vyskytujúci sa medzi položkami v databáze. Tieto parametre sa menia globálne ako atribúty objektu mriežky a pri každom volaní sa všetky volajú ak sú nenulovo nastavené. Operácie ako radenie položiek by sa dali realizovať aj lokálne, ale pravdepodobne by položky z databáze neboli načítané všetky, čiže by sa tieto operácie realizovali iba medzi čiastočnou zbierkou dát. Preto pri každom inicialiozvaní týchto operácií posiela serverovú požiadavku so zmenenými parametrami, čiže všetky tieto operácie sa dejú na strane servera. Predtým, než komponenta odošle požiadavku na nový súbor dát, skontroluje či už nemá dáta sa daným kľúčom lokálne uložené. Tým sa, ako som už spomínal, redukuje počet serverových volaní a zvyšuje sa rýchlosť odozvy a vykreslovania žiadaného obsahu.

Implementácia klientskej lokálnej pamäte je implementovaná objektom vo formáte JSON, kde ukladáme dvojice kľúč/hodnota. Kľúč je jednoznačný identifikátor v poli položiek a hodnota je v mojej implementácii taktiež pole ktoré pri manipulácii prevádzame na objekt a naopak. Poskytuje rozhranie na nastavenie položky podľa kľúča a hodnoty, zistenie či pamäť obsahuje položku s daným kľúčom, vyprázdnenie celej pamäte a vymazanie položky.

6 Záver

Vo svojej bakalárskej práci som sa zameral na návrh a implementáciu dátovej mriežky za použitia webových technológií. Preskúmal som veľa dostupných implementácií v Java Scriptových knižniciach, no napokon som si vybral a hlbšie skúmal implementácie v knižniciach Dojo a ExtJS. V týchto dvoch knižniciach mi komponenta dátovej mriežky pripadala najviac prepracovaná a priebežne vyvíjaná, čiže logicky viac a častejšie optimalizovaná.

Implementácia funguje a bola vyvíjaná v Mozilla Firefox a Google Chrome, a mala by fungovať aj na IE od verzie 8.

Implementáciu sa mi nepodarilo dokončiť tak, ako som si predstavoval. Tento fakt bol ovplyvnený mojimi slabými skúsenosťami s objektovými jazykmi a Javascriptom v tejto podobe. Rozhodol som sa implementovať svoju prácu s využitím knižnice Mootools. Táto knižnica ponúka bohatú dokumentáciu a veľa nápomocných článkov po celom internete.

Práca mi ukázala technológie, ktoré som v oblasti webu príliš nepoznal. Osobný prínos vidím v objavení nových, zaujímavých a veľmi efektných pomôcok pri tvorbe kvalitného webu a silný potenciál cítim v objektovo orientovanom JavaScripte s ktorým budem ďalej experimentovať a zavádzať do praxe.

Projekt možno rozširovať rôznymi vylepšeniami. Na dátovej vrstve to môžu byť implementácie nových „adaptérov“ rôznych formátov pre klientskú pamäť. V oblasti užívateľského rozhrania sú to buď vstupné editačné polia v podobe widgetov, implementovanie iných navigačných prvkov alebo stromovú hierarchizáciu záznamov mriežky. Ďalším zaujímavým vylepšením by bolo implementovanie udalostí z klávesnice alebo viacnásobné filtre pre vyhľadávanie.

Literatúra

[1] DOM: *DOM Specifications [online]. 2005 [cit. 2011-01-17]. www.w3schools.com.* Dostupné z WWW: <http://www.w3.org/DOM/ >. [web]

[2] *HTML: HTML 4.01 Specification [online]. 1*999 [cit. 2011-01-17]. www.w3schools.com. Dostupné z WWW: <http://www.w3.org/TR/html401/>. [web]

[3] *CSS: CSS Reference* [online]. 2009 [cit. 2011-04-17]. www.w3schools.com. Dostupné z WWW: < http://www.w3schools.com/css/css_reference.asp>. [web]

[4] *ASLESON, Ryan; T.SCHÜTT , Nathaniel. AJAX - Vytváříme vysoce interaktivní webové aplikace. Praha : Computer Press, 2006. 272 s. ISBN 80-251-1285-3, 978-80-251-1285-4 . [kniha]*

[5] MAHEMOFF, Michael. *Ajax Design Patterns*. Vyd. 1. [s.l.] : O'Reilly Media, 06.07.2006. 656 s. ISBN 9780596101800.

[6] *Dojo: Dojo Toolkit Reference Guide* [online]. 2008 [cit. 2011-02-11. Dostupné z WWW: <http://docs.campus.org/manual/>. [web]

[7] *Sencha : Ext JS 4 JavaScript Framework for Rich Apps in Every Browser* [online]. 2011 [cit. 2011-05-01]. Dostupné z WWW: <http://www.sencha.com/products/extjs/>. [web]

[8] *Json.org* [online]. 2002 [cit. 2011-04-10]. Dostupné z WWW: <www.json.org>. [web]

[9] *Mysql.com* [online]. 2010 [cit. 2011-04-17]. MySQL.com. Dostupné z WWW: <www.mysql.com>. [webová stránka]

[10] Component-based software engineering. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 02 October 2005, last modified on 11 April 2011 [cit. 2011-05-16]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Component-based_software_engineering>. [e-příspěvek]

Zoznam obrázkov a výpisov

Obrázok 2.1: Vzťahy a vlastnosti kľúčových RIA technológií – strana 8

Obrázok 2.2: Hierarchická stromová štruktúra modelu DOM– strana 9

Obrázok 2.3: Princíp fungovania technológie Ajax – strana 13

Obrázok 2.4: Schéma princípu serverového skriptu. - strana 14

Obrázok 4.1: Princíp komunikácie webovej dátovej mriežky – strana 23

Obrázok 4.2: Príklad viacnásobného radiaceho mechanizmu – strana 24

Obrázok 4.4: Editačné pole – strana 25

Obrázok 3.3: Selektačný filter – strana 17

Obrázok 4.5: Editačné možnosti widgetov – strana 26

Obrázok 4.6: Princíp Live Search – strana 27

Obrázok 5.1: Zobrazenie scrollbaru a výpočet jeho pozície – strana 33

Výpis 2.1: Príklad kódu vo formáte JSON – strana 9

Výpis 2.2: Príklad kódu v XML – strana 9

Výpis 3.1: Vytvorenie gridu v Dojo pomocou deklarácie v HTML – strana 12

Zoznam použitých skratiek

AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
DOM	Document Object Model
GUI	Graphical User Interface
HTML	HyperText Markup Language
JSON	Java Script Object Notation
RIA	Rich Internet Applications
URL	Uniform Resource Locator
XML	eXtensible Markup Language
W3C	World Wide Web Consortium
WWW	World Wide Web

Prílohy

A. Obsah priloženého CD

Priložený text obsahuje text práce, príkladné implementácie, prehliadače potrebné na spustenie, pomocné knižnice a implementáciu.

Štruktúra zložiek na CD je následovná:

- ***Browsers*** – zložka v ktorej sa nachádzajú potrebné prehliadače
- ***Grid*** – zložka obsahujúca implementáciu a potrebné moduly