Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Prispôsobiteľný Widget

(dokumentácia k 2. šprintu)

Tím č.10 : the 6_p@ck

Bc. Michal Immer
Bc. Jakub Korch
Bc. Jozef Macho
Bc. Peter Petriľák
Bc. Igor Repka
Bc. Ján Sivuľka

Študijný program: Softvérové inžinierstvo/Informačné systémy
Ročník: 1.ročník inžinierskeho štúdia
Predmet: Tímový projekt
Vedúci projektu: Ing. Tomáš Kuzár
Ak. rok: 2010/11

Analýza

Google Spreadsheet

Ide o jednoduché úložisko dát, ktoré dokáže pri menších objemoch dát nahradiť klasickú databázu, čím je možné ušetriť množstvo nákladov spojených s prevádzkovaním a sprístupnením dát v takejto databáze.

Do Spreadsheet-u je možné pristupovať priamo cez webový prehliadač alebo pomocou Google Rest API. Druhý spomínaný spôsob sme využili v našom projekte ako spôsob na pristupovanie k dátam v takej forme, aká nám vyhovuje.

Dáta v Spreadsheet-e sú reprezentované v podobe tabuliek v jednom alebo viacerých hárkoch (podobný spôsob aký sa využíva v kancelárskom produkte Excel). K týmto údajom je možné pristupovať s použitím Rest API, ktoré nám ponúka niekoľko spôsobov zobrazenia dát vo formáte XML feed-u. Načítanie údajov je možné dvomi spôsobmi a to tzv. "list-based feed", ktorý je založený na princípe "jeden riadok v tabuľke = jeden **entry** tag". Druhou verziou založenou na princípe získavania dát je tzv. "cell-based feed". V tomto prípade tvorila jeden záznam jedna samostatná bunka zo spreadsheet-u.

Dizajn widgetu

Vizualizáciu widgetu sme vzhľadom na použitie technológie HTML riešili pomocou kaskádových štýlov(CSS), teda pomocou mechanizmu na jednoduché formátovanie webových dokumentov. Tento spôsob štruktúrovania dokumentu umožňuje oddeliť obsah dokumentu od jeho vzhľadu. CSS umožňujú definovať vzhľad dokumentu 3 spôsobmi a to:

- externým súborom
- vnorením CSS bloku do hlavičky HTML dokumentu
- definovaním štýlu priamo v elemente, ktorý používame v HTML

Tieto spôsoby svojou kombináciou tvoria tzv. kaskádový štýl, pričom podľa priority vytvárajú nový štýl. Štýl HTML dokumentu je teda daný použitím jedného z predchádzajúcich spôsobov definovania, kde priority použitia jednotlivých štýlov sú nasledovné:

1. štýl definovaný priamo v elemente
2. štýl definovaný v hlavičke HTML súboru
3. štýl definovaný v externom súbore

4. štýl definovaný prehliadačom

, kde 1. znamená najvyššiu prioritu.

Návrh

Google Spreadsheet

Po zvážení a otestovaní viacerých spôsobov sme sa rozhodli, že viac výhod poskytuje prvá možnosť získavania údajov, teda pomocou jedného riadku v tabuľke. Implementovali sme aj druhý spôsob, avšak okrem iného bolo veľkou nevýhodou niekoľkonásobné zvýšenie dátového objemu výsledného dokumentu v porovnaní s prvou spomínanou metódou.
Samotné získanie obsahu takéhoto feed-u je možné uskutočniť pomocou knižničnej funkcie jazyka PHP nazývanej cURL, ktorá umožňuje skopírovať obsah ľubovoľného webového zdroja alebo dokumentu a následne s ním pracovať. Tak sme mohli vygenerovaný feed zo Spreadsheet-u uložiť a následne ho spracovávať na servere.
Dizajn widgetu
Podľa požiadavky zadávateľa návrh vizualizácie widgetu riešime pomocou niekoľkých vlastností widgetu. Jednotlivým elementom widgetu môže používateľ meniť vlastnosti podľa požadovaného výzoru, ktorý chce dosiahnuť na svojej stránke. Používateľ má možnosť si zvoliť pri vytváraný svojho špecifického widget-u z niekoľkých možností vzhľadu, pričom má na výber rôzne farebné kombinácie, zmenu rozmeru a typu zobrazovaných dát.
Zároveň sa mu ihneď zobrazuje aj náhľad toho, ako bude v skutočnosti ním nakonfigurovaný widget vyzerať. To je možné vďaka technológii Jacascript, ktorá umožňuje pružne reagovať na používateľove akcie.

Implementácia

Google Spreadsheet

V nasledujúcej časti je popísané parsovanie XML dokumentu získaného z Google Spreadsheet. Požadované dáta zo Spreadsheet-u sú dostupné na servere. V ďalšej etape je potrebné dáta z XML dokumentu spracovať. Spôsobov na spracovávanie alebo inak "parsovanie" XML dokumentu existuje aj v rámci webových technológií viacero spôsobov. Opäť sme najskôr skúšobne implementovali dva spôsoby takéhoto spracovávania. Prvá v poradí bola metóda využívajúca technológiu DOM a fakt, že XML dokument je možné prechádzať podobne ako DOM dokument nakoľko majú identickú štruktúru. Tento spôsob je

funkčný, avšak nebol úplne ideálny. Okrem iného bolo využitie tejto technológie pomerne komplikované a ťažkopádne. Ako vhodnejšie riešenie sa ukázala možnosť využiť schopnosti Javascript-u, konkrétne zdokonalenú verziu v technológii AJAX. V rámčeku nižšie je kompletný kód v jazyku AJAX, ktorý nám umožňuje spracovanie XML dokumentu a získania všetkých kategórii z daného dokumentu, ktoré sú následne odoslané do html formulára, konkrétne do tzv. combo box-u (alebo tiež dropdown list-u). Tento sa zobrazuje v klientskej časti aplikácie používateľovi a umožňuje mu vybrať si konkrétnu kategóriu, podľa ktorej chce filtrovať udalosti z bázy všetkých akcií.

Obrázok č.1: Ukážka spôsobu parsovania XML

```
<script>
$(document).ready(function(){
  $.ajax({
    type: "GET",
    url: "feed.xml",
    dataType: "xml",
    success: function(xml){
      var select = $('#mySelect');
      var data = "category";
      var pom = "";
      $.xmlns["gsx"] =
      "http://schemas.google.com/spreadsheets/2006/extended";
      $(xml).find('entry').each(function(){
        var title = $(this).find(data).text();
        if(pom.indexOf(title) == -1){
          select.append("<option value='"+title+"'
          class='ddheader'>"+title+"</option>");
          pom = pom + title;
        }
      });
      document.getElementById("mySelect").remove(this.selectedIndex);
    }
  });
});
</script>
```

Podobným spôsobom je možné zo zdroja, teda XML súboru, získať akékoľvek dáta a následne ich zobraziť v okne nášho widgetu.

Dizajn widgetu

Naša implementácia pozostáva z dvoch častí, prvou je voľba vlastností pre widget a druhou je zobrazenie vlastností widgetu. Pri vizualizácii samotného widgetu bol kladený dôraz hlavne na prehľadnosť a jednoduchosť rozhrania pre túto vizualizáciu. Implementované rozhranie je viditeľné na Obrázku č.2, Obrázku č.3 a Obrázku č.4, ktoré sú priložené v prílohe tejto dokumentácie. Používateľ si môže vybrať jedno z preddefinovaných pozadí podľa svojho výberu. Tieto sú volené tak, aby sa mal čo najväčšiu možnosť priblížiť sa dizajnu svojej stránky. Pri výbere veľkosti widgetu je ohraničenie pre šírku <150, 500> pixelov a ohraničenie pre šírku <150, 800> pixelov. Tieto ohraničenia sme vybrali po zvážení vhodnosti veľkosti takéhoto widgetu na stránke používateľa. Príliš malý widget by totiž nezobrazoval dostatočne veľa informácií bez nutného použitia kurzora myše a príliš veľký widget by zase pôsobil rušivo na stránke. Na spomínaných obrázkoch je vidieť i správanie prispôsobenia sa widgetu pri zadaní nezmyseľne veľkých rozmerov. Pokiaľ sa zadá menší rozmer, ako je minimum, daný atribút sa nastaví na minimum, pokiaľ sa nastaví väčší rozmer, daný atribút sa nastaví na maximum. Používateľ môže ešte zvoliť počet zobrazených článkov a názov widgetu.

Testovanie

Testovanie bolo vykonávané pre nastavenie veľkosti widgetu, zobrazenie správy vo widgete a filtrovanie nad RSS úložiskom.

Tabuľka č.1: Akceptačný test pre nastavenie veľkosti widgetu

ID	1	**Názov**	Nastavenie veľkosti widgetu		
Úroveň splnenia testu			Musí – ~~Mal by~~ – ~~Mohol by~~	**Autor**	Jozef Macho
Rozhranie		Stránka widgetizéru(vizualizácia widgetu)			
Účel		Overenie správnej funkčnosti zmeny veľkosti widgetu			
Vstupné podmienky		Nastavené parametre dĺžky a šírky widgetu v rámci povoleného rozsahu			
Výstupné podmienky		Zmena parametrov dĺžky a šírky widgetu na nastavené a zobrazenie widgetu v danej veľkosti			

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Zväčšenie(zmenšenie) parametra výšky widgetu	Vertikálne roztiahnutie(zúženie) zobrazeného widgetu.	Rovnaká ako očakávaná.
2	Zväčšenie(zmenšenie) parametra šírky widgetu	Horizontálne roztiahnutie(zúženie) zobrazeného widgetu.	Rovnaká ako očakávaná.
3	Zadanie väčšej(menšej) hodnoty výšky, ako je horná(dolná) hranica	Vertikálne roztiahnutie(zúženie) zobrazeného widgetu na maximálnu(minimálnu) hodnotu výšky.	Rovnaká ako očakávaná.
4	Zadanie väčšej(menšej) hodnoty šírky, ako je horná(dolná) hranica	Horizontálne roztiahnutie(zúženie) zobrazeného widgetu na maximálnu(minimálnu) hodnotu výšky.	Rovnaká ako očakávaná.

Tabuľka č.2: Akceptačný test pre zobrazenie názvu udalosti vo widgete

ID	2	**Názov**	Zobrazenie názvu udalosti vo widgete		
Úroveň splnenia testu			~~Musí~~ – Mal by – ~~Mohol by~~	**Autor**	Jozef Macho
Rozhranie		Stránka widgetizéru(vizualizácia widgetu)			
Účel		Overenie správnej funkčnosti zobrazenia názvu udalosti vo widgete			
Vstupné podmienky		Načítané správy z dátového zdroja, zobrazenie vzhľadu widgetu			
Výstupné podmienky		Vypísanie názvu udalosti pri zobrazení widgetu			
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**
1	Zobrazenie widgetu na stránke widgetizéru(kustomizácia)		Zobrazenie názvu správy získanej z dátového zdroja vo vizualizácii widgetu v podobe hyperlinku.		Nič sa vo widgete zo správy nezobrazuje.

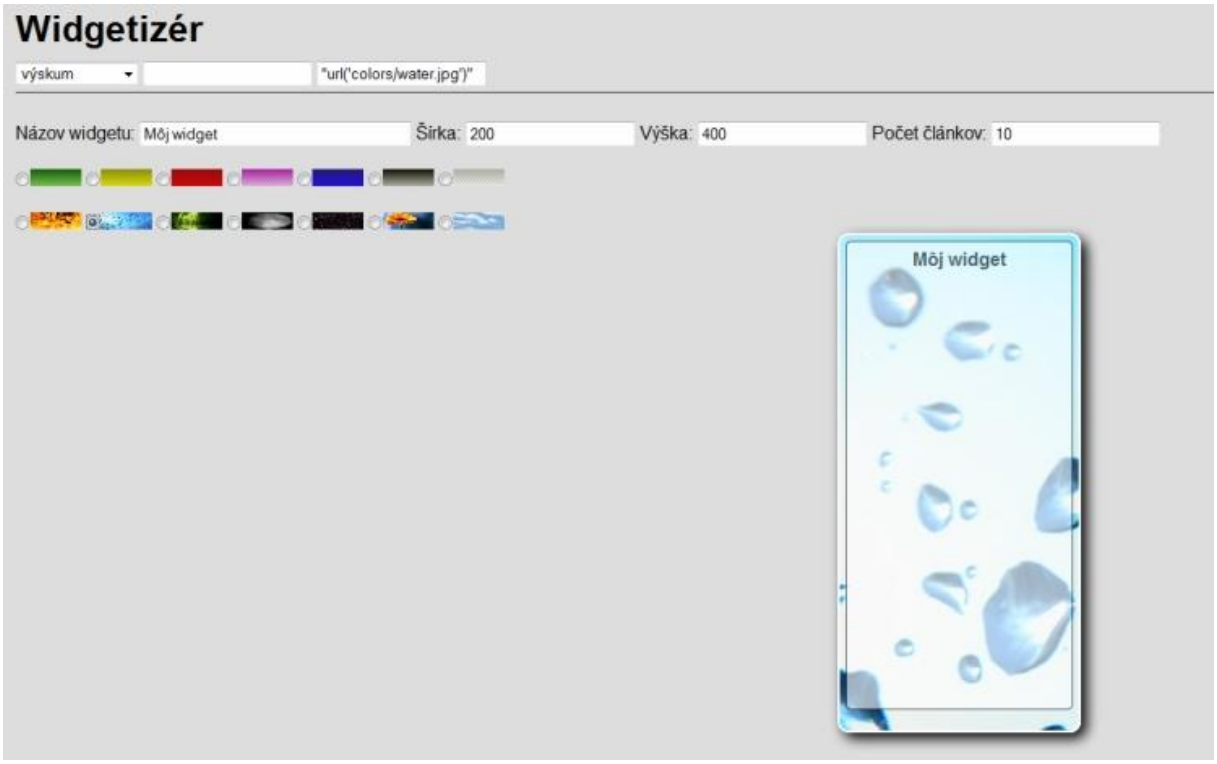
Tabuľka č.3: Akceptačný test pre filtrovanie kategórií nad RSS úložiskom.

ID	3	**Názov**	Filtrovanie kategórií nad RSS úložiskom		
Úroveň splnenia testu			Musí – ~~Mal by~~ – ~~Mohol by~~	**Autor**	Jozef Macho
Rozhranie		Hlavná stránka			
Účel		Overenie správnej funkčnosti filtrovania kategórií nad RSS úložiskom			
Vstupné podmienky		Správna cesta k RSS úložisku, kategórie v RSS správe			
Výstupné podmienky		Získané kategórie z RSS správy			
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**
1	Vloženie cesty k RSS úložisku a stlačenie tlačidla na získanie kategórií		Vložené kategórie z RSS správ do ComboBoxu.		Skutočná ako očakávaná.

Prílohy

Obrázok č.2: Rozhranie pre vytváranie a vizualizáciu widgetu(najmenšia veľkosť)

Obrázok č.3: Rozhranie pre vytváranie a vizualizáciu widgetu(zvolená veľkosť)



Obrázok č.4: Rozhranie pre vytváranie a vizualizáciu widgetu(maximálna veľkosť)